НАЦІОНАЛЬНА АКАДЕМІЯ ПЕДАГОГІЧНИХ НАУК УКРАЇНИ
ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЗАСОБІВ НАВЧАННЯ

На правах рукопису

Шевчук Петро Георгійович

УДК 371.3 : 372.8 : 004 : 681.3.062

# Методика навчання програмування учнів класів технологічного профілю на основі використання мови C#

13.00.02 – теорія та методика навчання (інформатика)

Дисертація
на здобуття наукового ступеня
кандидата педагогічних наук

Науковий керівник:
Спірін Олег Михайлович,
доктор педагогічних наук, доцент

Київ – 2013

**Зміст**

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

| Скорочення, термін, позначення | Пояснення |
|---|---|
| ЗНЗ | Загальноосвітній навчальний заклад |
| ІКТ | Інформаційно-комунікаційні технології |
| МСН | Методична система навчання |
| ОО | Об’єктна-орієнтованість |
| ООП | Об’єктно-орієнтоване програмування |
| СД | Семантичний диференціал |
| .NET | Програмна платформа Microsoft .NET Framework |

# ВСТУП

**Актуальність теми.** Людство переживає активне становлення інформаційного соціуму, пріоритетом якого є випереджаючий розвиток науки та активне інтелектуальне новаторство у виробництві, культурі та управлінні. Висока креативність фахівця того чи іншого профілю – головна запорука успіху як окремої особистості на ринку праці, так і обов'язкова передумова становлення та розвитку виробничої чи іншої людської спільноти. Домінантою сучасного розвитку є новітні інформаційні технології з використанням сучасного програмного забезпечення. Фахівець з програмування – найбільш затребувана нині професія технологічного спрямування. В умовах упровадження профільного навчання в школі, програмування є необхідним та значною мірою вирішальним компонентом стратегічного інформаційного розвитку України.

Значний прогрес у сфері інформаційних технологій вимагає оновлення програмного забезпечення, яке використовується в навчальному процесі. Інновації можуть мати позитивний ефект лише за умови їх науково обґрунтованого та методично доцільного використання.

Національна доктрина розвитку освіти одним із головних пріоритетів визначає підготовку молодого покоління до життєдіяльності в інформаційному суспільстві [221]. Нині в українських школах значно збільшилася кількість комп'ютерного обладнання. Активно впроваджуються нові педагогічні та інформаційні технології. Здійснюється перехід до профільного навчання, що базується на широкому використанні інформаційно-комунікаційних технологій (ІКТ). Щоправда, бурхливе і переважно успішне впровадження інформаційних технологій в освіту не супроводжується підвищенням інтересу дітей до програмування як діяльності, що лежить в основі такого розвитку. Нові навчальні програми з інформатики, затверджені Міністерством освіти для загальноосвітніх навчальних закладів (ЗНЗ), залишають все менше часу на вивчення розділів, пов'язаних з програмуванням [97; 98; 116]. Навчання

програмування здійснюється нині в школі практично на тих же засадах, що й два десятиліття тому, коли інформатика лише з'явилась як шкільна дисципліна. Повільно впроваджуються нові парадигми програмування. В цілому навчання програмування в загальноосвітніх навчальних закладах погано інтегроване як зі змістом решти розділів інформатики, так і зі змістом навчального матеріалу інших шкільних дисциплін. За таких умов нагальною є потреба в удосконаленні методичних підходів щодо навчання алгоритмізації та програмування.

Разом з тим навчання програмування створює передумови формування та розвитку загальної креативності, логічного та операційного мислення учнів. Програмування є базою сучасних фундаментальних знань. Ця дисципліна лежить в основі суспільно значимих та економічно затребуваних професій. Без розробки алгоритмів та програм не може обійтися практично будь-яка науково-технічна творчість, в тому числі і дитяча. Елементи алгоритмічної культури активно проникають в соціальне середовище та мистецтво, чинять значний вплив на соціум в цілому.

Динамічно розвиваються концепції та парадигми розробки програмного забезпечення, технології та засоби програмування. Актуальним є використання в навчальному процесі, окрім традиційних мов та середовищ програмування (Pascal, Basic, Visual Basic, Delphi), новітніх та перспективних (Python, Java, Java Script, PHP, Visual Studio на основі платформи Microsoft .NET Framework). Поряд із цим, фірмою Microsoft спеціально створено мову, яка б максимально використовувала сучасні підходи у програмуванні. Вона отримала назву – C# (читається «Сі шарп») [127; 251; 328; 331;]. Використання цієї мови забезпечує високу швидкодію програм і вважається одним з найбільш досконалих засобів професійного програмування [224]. Тому важливо привернути увагу до неї педагогів під час навчання програмування. За умов упровадження профільної диференціації в загальноосвітній школі мови програмування, що використовуються для навчання, повинні забезпечувати як загальноосвітню, фундаментальну, так і спеціалізовану, професійну складову навчального

матеріалу. Особливо корисним буде використання сучасних професійних мов програмування, зокрема C#, в класах технологічного профілю загальноосвітнього закладу.

Сучасна педагогічна наука за останні роки збагатилася багатьма теоретичними дослідженнями в сфері профільної освіти. Зокрема, узагальнення та аналіз практичних результатів впровадження профільного навчання здійснювали М. М. Авраменко [1], С. П. Бондар [25], С. Є. Вольянська [43], А. Д. Сазонов [249], А. Л. Сейтешев [257], Л. П. Фаннінгер [289]. Навчання інформатики в умовах впровадження профільності досліджували С. О. Бешенков [19], С. А. Вернигоренко [34], Л. В. Галигіна [50], Н. А. Давидова [67], Ю. О. Дорошенко [75], І. О. Завадський [96], Т. Б. Захарова [110], М. І. Жалдак [83], С. В. Крапивка [138], О. Г. Кузьмінська [83], Н. В. Морзе [186], А. Г. Пекшева [210], Л. П. Фаннінгер [289], А. П. Шестаков [317].

Загальні питання навчання програмування розглядалися авторами шкільних програм та підручників, серед яких В. Є. Анохін [55], В. Ю. Биков [20], В. П. Вембер [181], В. В. Володін [40], І. Л. Володіна [40], Я. М. Глинський [54], Ю. О. Дорошенко [40], М. І. Жалдак [116], Л. А. Журавльова [116], І. О. Завадський [97 – 101], І. Т. Зарецька [109], О. П. Зеленяк [112], Т. П. Караванова [117], Б. Г. Колодяжний [109], В. П. Костюков [117], О. Г. Кузьмінська [181], Т. І. Лисенко [158], Н. В. Морзе [181], В. П. Пасько [208], Ж. В. Потапова [97], Н. С. Прокопенко [122], Т. Г. Проценко [122], Ю. С. Рамський [85], В. А. Ребрина [116], Й. Я. Ривкінд [158], В. Д. Руденко [245], В. А. Ряжська [55], О. В. Співаковський [270], О. М. Спірін [271], Л. А. Чернікова [158], В. В. Шакотько [158] та інші.

Різноманітні аспекти проблем впровадження в школі новітніх середовищ програмування, а також навчання програмування в умовах профільного навчання досліджуються вітчизняними та зарубіжними вченими

(С. А. Волошинов [41], В. Ю. Габрусєв [48], Я. М. Глинський [55], Ю. В. Горошко [62], Л. В. Гришко [24], М. І. Жалдак [84], Р. І. Заболотний [101], І.О. Завадський [96; 101], Т. П. Караванова [117; 118], Н. В. Морзе [185; 187], Ю. С. Рамський [235], В. Д. Руденко [245], З. С. Сейдаметова [254], С. О. Семеріков [260], О. В. Співаковський [270], О. М. Спірін [272], Ю. В. Триус [284]).

Зокрема, група московських науковців реалізувала компілятор мови програмування Pascal для платформи .NET і активно впроваджує його до навчання програмування. Українське видавництво BHV в 2008 році випустило навчальний посібник для учнів загальноосвітніх навчальних закладів І. О. Завадського та Р. І. Заболотного «Основи візуального програмування» в основі якого лежить використання середовища MS Visual Studio та мови програмування Visual Basic .NET для навчання візуальному проектуванню [101].

Проблеми впровадження об’єктно-орієнтованої парадигми до навчання програмування досліджували М. А. Бондаренко [27], Я. М. Глинський [55], Л. В. Гришко [64] Ю. С. Рамський [236, 237], С. О. Семеріков [259], О. І. Теплицький [281], Ф. В. Шкарабан [319], В. А. Білліг [22], А. С. Лесневский [156], А. Н. Петров [216] та інші.

Нині про використання C# у навчанні є достатньо відомостей на різноманітних ресурсах Інтернету [332; 330; 176; 177; 22]. Таке навчання широко представлене в системі дистанційної освіти на багатьох українських та закордонних сайтах [176]. Московський міжнародний університет «Інтуит.ру» проводить дистанційне навчання мовою C# для початківців дитячого віку «C# для начинающих [22]». Фірма Microsoft розробила навчально-ігрову надбудову до Visual Studio – Microsoft Visual Studio Learning Pack, призначену для навчання програмування в загальноосвітніх школах та вищих навчальних закладах, базовану на мові програмування C# [343]. Навчання професійних програмістів мові C# пропонують цілий ряд комерційних навчальних курсів

міста Києва та інших міст України [176]. Об’єктно-орієнтоване програмування, що базується на платформі Microsoft .NET, входить до навчальних планів багатьох спеціальностей вищих навчальних закладів, які готують фахівців у галузі розробки програмного забезпечення [64; 255 ].

Варто зазначити, що на платформі .NET може бути ефективно реалізована проектна діяльність школярів у програмуванні. Мовою C# цікавляться учні, що постійно беруть участь у конкурсах, олімпіадах, турнірах з інформатики. Оскільки підготовка учнів до участі в змаганнях та конкурсах не може обмежуватися вивченням предмету в межах шкільної програми, залучення педагогічно обґрунтованих і виважених новацій має значні перспективи.

Проблема навчання інформатики учнів загальноосвітнього навчального закладу в умовах профілізації середньої освіти до кінця не розв’язана, а це негативно відбивається на впровадженні профільного навчання, створенні умов для досягнення учнями із різним рівнем здібностей належного рівня компетентностей. Поза увагою дослідників залишилася проблема обґрунтування та розробки методичних засад навчання інформатики учнів класів технологічного профілю зорієнтованих на використання новітніх середовищ програмування. Відсутня цілеспрямована методика навчання програмування учнів класів технологічного профілю середньої школи на основі використання мови C#, яка знаходить все більше застосування на практиці і достатньо динамічно розвивається.

Таким чином, існує протиріччя між об’єктивною необхідністю впровадження профільного навчання в середніх загальноосвітніх навчальних закладах, потенціалом сучасних візуальних середовищ програмування щодо навчання інформатики учнів класів технологічного профілю і недостатньою розробленістю відповідного науково-методичного забезпечення на цій основі, що породжує актуальну соціальнозначущу проблему, на вирішення якої спрямоване наше дисертаційне дослідження: **«Методика навчання**

**програмування учнів класів технологічного профілю на основі використання мови C#»**.

**Зв'язок роботи з науковими програмами, планами, темами**

Дисертаційне дослідження виконано відповідно до тематичного плану науково-дослідної роботи відділу дослідження і проектування навчального середовища Інституту інформаційних технологій і засобів навчання НАПН України "Науково-методичні засади організації середовища дистанційного навчання в середніх загальноосвітніх навчальних закладах" (ДР № 0109U000175).

Тему дисертаційного дослідження затверджено вченою радою Інституту інформаційних технологій і засобів навчання НАПН України (протокол № 8 від 23 жовтня 2008 ) та узгоджено Радою з координації наукових досліджень у галузі педагогіки і психології НАПН України (протокол № 4 від 26. 05. 2009).

**Мета дослідження – науково** обґрунтувати та розробити методику навчання програмування учнів в класах технологічного профілю середньої школи на основі використанням мови C#.

Для досягнення зазначеної мети необхідним було вирішення таких **задач**:

1. З’ясувати психолого-педагогічні особливості організації шкільного навчально-виховного процесу за умов профілізації освіти та концептуальні потреби алгоритмічної підготовки учнів у класах технологічного профілю.

2. Проаналізувати стан дослідження проблеми використання середовища .NET в цілому та C# зокрема в навчанні програмування та визначити програмно-технологічні можливості використання мови C# для навчання програмування у ЗНЗ.

3. Визначити і створити основні компоненти методичної системи навчання програмування, розробити методику навчання програмування учнів класів технологічного профілю з використанням мови C# та визначити педагогічні умови впровадження мови C# в шкільний курс інформатики.

4. У ході педагогічного експерименту перевірити ефективність методики навчання програмування на основі використання мови C# в класах технологічного профілю.

***Об'єктом дослідження*** є процес навчання інформатики учнів загальноосвітніх навчальних закладів II-III ступенів в умовах профілізації освіти.

***Предметом дослідження*** є цілі, завдання, зміст, методи, форми і засоби навчання програмування учнів класів технологічного профілю середньої школи на основі використання мови C#.

***Методи дослідження:*** *аналіз, систематизація, узагальнення* наукової вітчизняної та зарубіжної фахової, педагогічної та навчально-методичної літератури (1.1 – 1.3, 2.1 (тут і далі підрозділи дисертації); *наукові узагальнення досвіду вчителів* інформатики загальноосвітніх навчальних закладів з метою виявлення й систематизації дослідницьких матеріалів (1.1, 2.1); *педагогічне прогнозування і моделювання* для побудови ефективного навчально-виховного процесу згідно з предметом дослідження (2.2, 2.3); *рефлексія власної професійної діяльності* (2.1, 2.2) сприяла визначенню взаємозв'язків під час суб'єкт-суб'єктної взаємодії, організації самостійної роботи та контролю успішності учнів; *спостереження* за навчальною діяльністю учнів загальноосвітніх навчальних закладів на уроках інформатики; *бесіди й опитування* вчителів сприяли вивченню досвіду вчителів інформатики з навчання програмування в класах технологічного профілю; *педагогічний експеримент* з метою перевірки ефективності методики навчання програмування в класах технологічного профілю загальноосвітніх навчальних закладів на основі мови програмування C#; *метод семантичного диференціалу* для отримання в процесі педагогічного експерименту кількісних значень оцінювання результатів навчання програмування в класах технологічного профілю; *математично-статистичний метод,* за допомогою якого визначено кількісні залежності між показниками, отриманими в результаті педагогічного

впливу на формувальному етапі дослідження та проведено якісний аналіз цих кількісних залежностей.

**Наукова новизна одержаних результатів** дослідження полягає в тому, що *вперше:*

– розроблено основні компоненти методичної системи навчання програмування учнів класів технологічного профілю з використанням мови C#, в межах якої забезпечується впровадження об'єктно-орієнтованої парадигми програмування;

– визначено педагогічні умови впровадження мови C# в шкільний курс інформатики;

– розроблено методику навчання програмування в класах технологічного профілю на основі використання мови C#;

*удосконалено:* методичні підходи до навчання об'єктно-орієнтованого програмування; теоретичні положення методичної системи навчання програмування в загальноосвітніх навчальних закладах; термінологію та систему понять, необхідних для навчання об'єктно-орієнтованого програмування в загальноосвітніх навчальних закладах.

*дістали подальшого розвитку:* методика навчання програмування учнів ЗНЗ, методичні підходи до забезпечення профільності навчання інформатики.

**Практичне значення одержаних результатів** полягає в розробці та апробації навчально-методичних матеріалів для курсу інформатики загальноосвітніх навчальних закладів, а саме: розроблено методичні рекомендації щодо навчання програмування на основі використання мови C# в класах технологічного профілю [308], удосконалено та адаптовано до навчання програмування мовою C# в класах технологічного профілю загальноосвітніх навчальних закладів рекомендоване календарне планування [231] згідно програми інформатики для класів технологічного профілю [116, с. 65-85]; розроблено практичні завдання, перелік тестових питань, завдання для перевірки знань учнів; для методичної підтримки навчання програмування

мовою C# в класах технологічного профілю створено сайт https://sites.google.com/site/c4plus/, який водночас використовувався для інформаційної підтримки педагогічного експерименту.

Положення дисертаційного дослідження використані у процесі безпосередньої педагогічної діяльності дисертанта в навчанні інформатики в Миропільській гімназії, Романівського р-ну, Житомирської обл., (довідка від 15 листопада 2012 р., № 210); в процесі навчання програмування в класах технологічного профілю Миропільської загальноосвітньої школи I-III ступенів № 2, Романівського р-ну, Житомирської обл. (акт про впровадження від 04 грудня 2012 р., № 203); Камінської загальноосвітньої школи, Романівського р-ну, Житомирської обл. (довідка від 11 жовтня 2012 р., № 48); Ружинської гімназії, Ружинського р-ну, Житомирської обл., (довідка від 21 вересня 2012 р., № 17-2); загальноосвітньої школи I-III ступенів № 7 м. Житомира (довідка від 12 листопада 2012 р., № 222-17); Іршанського навчально-виховного комплексу «Гімназія-дошкільний навчальний заклад» Володар-Волинського району Житомирської області (довідка від 04 жовтня 2012 р., № 236/1); спеціалізованої школи № 194 «Перспектива» з поглибленим вивченням природничих дисциплін, Оболонського р-н, м. Києва (довідка від 26 лютого 2013 р., № 11).

**Особистий внесок здобувача.** У статті, написаній у співавторстві з О. М. Кривоносом [305], автору належать систематизація стилів написання програм, що застосовується при розробці мовою C#. У статті, написаній у співавторстві з О. М. Шимоном [310], автору належать ідея класифікації Інтернет-сторінок присвячених мові C# за розробником, збір фактичного матеріалу, значна частина практичних рекомендацій, узагальнення та висновки.

**Апробація результатів дисертації**

Апробація результатів дисертації відбувалася шляхом публікації наукових праць автора, доповідей, повідомлень на конференціях та науково-методичних семінарах різного рівня:

**Міжнародні:** «Нові інформаційні технології в освіті для всіх: інноваційні методи та моделі" (м. Київ, 2009 р.), IX «Теорія та методика фундаментальних дисциплін у вищій школі» (м. Кривий Ріг, 2011 р.), «Інформаційні технології в освіті, науці і виробництві» (м. Луцьк 2011 р.), «Інженерія програмного забезпечення 2011» (м. Київ, 2011 р.);

**Всеукраїнські:** «VIII Всеукраїнська науково-практична конференція «Комп'ютерне моделювання та інформаційні технології в науці, економіці і освіті» (м. Одеса 2011 р.), Всеукраїнська науково-практична конференція «Освіта в інформаційному суспільстві: до 25-річчя шкільної інформатики» (м. Київ, 2010 р.); звітних наукових конференціях Інституту інформаційних технологій і засобів навчання НАПН України (м. Київ, 2009–2011рр.); семінарах – Всеукраїнському науково-методичному семінарі з питань використання засобів сучасних інформаційних технологій у навчальному процесі (м. Київ, 2011 р.); Всеукраїнському науково-методичному семінарі «Системи навчання і освіти в комп'ютерно орієнтованому середовищі» (м. Київ, 2010 р.)

**Публікації.** Основний зміст дисертаційного дослідження висвітлено у 17 публікаціях, з них: 5 одноосібних статей у вітчизняних фахових виданнях, 4 статті у збірниках наукових праць (3 одноосібні, 1 у співавторстві), 7 інших публікацій у збірниках матеріалів і тез конференцій (6 одноосібні, 1 у співавторстві) та 1 методичні рекомендації.

**Структура та обсяг дисертації**. Дисертація складається зі вступу, трьох розділів, висновків до розділів, загальних висновків, списку використаних джерел (359 найменувань, з них 12 – іноземними мовами), 7 додатків. Загальний обсяг дисертації – 319 сторінок, основного тексту – 178. Робота містить 14 таблиць, 19 рисунків.

# РОЗДІЛ 1.
# ПСИХОЛОГО-ПЕДАГОГІЧНІ ОСНОВИ НАВЧАННЯ ПРОГРАМУВАННЯ В СЕРЕДНЬОМУ ЗАГАЛЬНООСВІТНЬОМУ ЗАКЛАДІ ЗА УМОВ ЙОГО ПРОФІЛЬНОГО СПРЯМУВАННЯ

У розділі розглянуто та проаналізовано теоретичні основи сучасної галузі програмування, перспективи та тенденції її розвитку. Здійснено аналіз стану навчання програмування в загальноосвітніх навчальних закладах. Досліджено зміст шкільних програм, досягнення, проблеми та перспективи навчання програмування. Досліджено систему профільного навчання інформатики, значення та місце інформатики в класах технологічного профілю ЗНЗ. Визначено перспективність навчання об’єктно-орієнтованого програмування мовою C#.

## 1.1. Навчання основ алгоритмізації та програмування учнів середньої школи як педагогічна проблема

### 1.1.1. Актуальність навчання програмування

Нині людство настільки широко використовує ІКТ, що вони стали одним із основних чинників прогресу. Самі ж комп’ютерні технології, що лежать в основі становлення сучасного інформаційного суспільства, формуються завдяки розвитку технологій розробки програмного забезпечення. Отже, програмування, як новітня науково-виробнича галузь, є рушійною силою сучасного інформаційно-технологічного розвитку. Написання комп'ютерних програм – це не лише грандіозний виробничо-економічний напрям, а ще й соціально-культурна сфера людства. Програмування – це водночас високотехнологічне виробництво, передова галузь науки, суспільнозначима праця, різновид одного з найцікавіших людських захоплень, сучасна спортивна дисципліна і навіть, певним чином, надновий напрям мистецтва [124]. Навчання програмування – окремий вектор сучасної освіти. Загальні знання з програмування – обов’язковий компонент інформаційно-комунікаційної компетенції сучасного фахівця високого рівня. Необхідність навчання

програмування продиктована завданнями сучасної освіти стосовно підготовки молоді до життя та діяльності в умовах інформаційного суспільства [106; 107; 129; 130].

Поряд із цим, навчанню програмування в загальноосвітніх закладах приділяється останнім часом усе менше уваги. Цей розділ інформатики зазнав значного скорочення. Незважаючи на те, що концепції та парадигми розробки програмного забезпечення, технології та засоби програмування постійно та динамічно розвиваються, навчання програмування в сучасних загальноосвітніх закладах переважно здійснюється за методиками та технологіями, що застосовувались на початку введення до навчальних планів інформатики як шкільної дисципліни. Цей розділ шкільної інформатики не лише повільно розвивається, а й погано інтегрований із загальним змістом освіти. Про це неодноразово зазначають вітчизняні фахівці в галузі шкільної інформатики. Наприклад, серед виступів учасників круглого столу «Інформатика у школах України. Сучасний стан, проблеми, перспективи.», присвяченого 25-річчю шкільної інформатики, який проходив у лютому 2010 року, значна частина доповідачів вже в назві своїх виступів закликала звернути увагу на проблеми навчання програмування в загальноосвітніх навчальних закладах: «Користувацький ухил затягнувся недозволено довго [111]», «Розвиток алгоритмічного мислення – основна задача курсу інформатики [265]». На певні недоліки та необхідність розвитку навчання програмування в сучасній шкільній освіті звертається увага у багатьох публікаціях та наукових дослідженнях [48; 55; 24; 92; 253; 260; 319].

Ще відносно недавно навчання програмування лежало в основі навчання інформатики в школі. Впровадження шкільної дисципліни «інформатика» проходило під гаслом висунутим академіком Єршовим А. П. «Програмування – друга грамотність [77]». На користь масового навчання програмування наводилось надзвичайно багато вагомих аргументів [78, 77, 123, 185]. Протягом останніх років основні акценти цього предмету майже повністю змістилися в

напрямку практично-користувацьких аспектів. Гасло «Програмування – друга грамотність» так і не знайшло свого остаточного втілення. Необхідно з’ясувати та проаналізувати причини того, чому масове впровадження навчання програмування виявилось нежиттєздатним. Адже, на даному етапі вже назріла необхідність не лише зберегти чи, якоюсь мірою, відродити значимість алгоритмічної складової курсу шкільної інформатики, а й радикально оновити зміст навчання програмування в загальноосвітніх навчальних закладах.

### 1.1.2. Завдання навчання програмування

На початку активного застосування комп’ютерів авторитетні фахівці вважали, що для підтримки дорогої та продуктивної обчислювальної техніки, кількість якої неухильно зростала, потрібно підготувати багато програмістів. Тобто потрібно було готувати армію фахівців з мисленням «програміста». Це вважалось головним завданням школи стосовно підготовки учнів до використання комп’ютерної техніки. Одним з витоків таких завдань були не лише потреби програмної підтримки окремо локалізованої обчислювальної техніки, а й плани Радянського Союзу створити єдину державну мережу обчислювальних центрів (російська абревіатура ЕГСВЦ), що могла стати першою в історії повноцінною регіональною комп’ютерною мережею [188]. З цього приводу історик науки В. О. Герович наводить цитату з неопублікованих рукописів академіка В. М. Глушкова: «Сеть должна была вступить в строй в 1975 году. Для ее эксплуатации требовалось подготовить триста тысяч специалистов… [52]».

У своїй статті «Програмування – друга грамотність» [77] академік А. П. Єршов посилається на закономірність, яка носить назву «логістична крива Баррі Боем». Це – графічне відображення співвідношення витрат на розробку програмного забезпечення та витрат на створення апаратної частини комп'ютерних систем. Закономірно вважалося, що ресурси, які необхідні для розробки програмного забезпечення, значно перевершують затрати на виготовлення обладнання. Ось що пише з цього приводу сам А. П. Єршов:

«Конечно, специалисты по программированию трудятся вовсю, чтобы сделать труд программиста более производительным. Однако даже если мы примем гипотезу десятикратного увеличения производительности труда при производстве программ, элементарные расчеты показывают, что для того чтобы через двадцать лет запрограммировать все производимые микропроцессоры, нам надо будет посадить за программирование все взрослое население земного шара [77, с. 10]».

Незважаючи на серйозні аргументи на користь масового навчання програмування, а також на широке впровадження інформатики до навчальних програм практично всіх загальноосвітніх навчальних закладів колишнього Радянського Союзу, програмування не стало і найближчим часом не стане заняттям масовим та широко розповсюдженим. Це – беззаперечно. Уже в кінці вісімдесятих та на початку дев'яностих років було вказано на те, що масове навчання програмування недоцільне [82, с. 26-27; 87, с. 4]. Тоді ж було проголошено впровадження користувацького ухилу під час вивчення комп'ютерної техніки учнями загальноосвітніх навчальних закладів. Основоположником користувацького ухилу навчання інформатики у загальноосвітніх навчальних закладах України є академік НАПН України М. І. Жалдак [84].

І все ж практично всі нині діючі шкільні навчальні програми з інформатики містять розділи, що хоча б оглядово торкаються навчання програмування [97; 98; 116; 117]. Отже, загальні принципи алгоритмізації та написання комп'ютерних програм, в тих чи інших обсягах, нині вивчаються практично усіма учнями загальноосвітніх навчальних закладів.

Навчання програмування значно відрізняється від інших розділів інформатики специфічним та складним навчальним матеріалом. Значення навчання програмування для загального розвитку особистості дуже велике. Навчання програмування у загальноосвітніх навчальних закладах виконує

відразу кілька завдань щодо навчання учнів та розвитку їх особистостей. Умовно ці завдання нами поділяються на прямі та супутні.

Пряме завдання навчання програмування – вчити створювати комп'ютерні програми. Таке завдання, як уже зазначалось, не можна вважати основним для навчання програмування учнів загальноосвітніх навчальних закладів, адже компетентності професійного програміста знадобляться в майбутньому далеко не всім. Незважаючи на значний дефіцит спеціалістів з розробки програмного забезпечення, сучасна економіка не потребує такої великої кількості програмістів, як, скажімо, продавців чи водіїв автотранспорту. Поряд із цим, вимоги до професійного програміста надзвичайно високі, а, отже, лише найбільш підготовлені та здібні учні здатні проявити себе в даній галузі.

Супутні завдання можна умовно поділити на завдання сприяти загальному оволодінню ІКТ та завдання розвитку особистості учня в цілому. Навчання програмування безумовно допомагає навчанню використовувати уже існуючі комп'ютерні програми. Вже одне лише написання коду програми розвиває навики введення тексту, формує уміння зберігати та знаходити раніше створені файли, вчить активно користуватися буфером обміну. Усвідомлення принципів створення та функціонування програмного забезпечення допомагає раціонально та продуктивно його використовувати. Втім знання програмування не є необхідною умовою ефективного використання існуючого програмного забезпечення. Навіть для високопідготовленого користувача немає прямої необхідності вміти розробляти комп'ютерні програми. Тобто, навчати програмувати учнів лише заради формування у них користувацьких навиків, швидше за все, недоцільно.

Завдання розвитку особистості учня, що стоять перед сучасною школою, надзвичайно загальні і багатопланові. Це основне завдання навчального процесу в ЗНЗ. Навчання програмування, безумовно, сприяє формуванню світогляду, розвитку мислення, пам'яті, уваги, т. і. Нині актуальним є формування таких рис особистості учня, що будуть необхідні йому в найбільш

перспективних сферах інформаційного суспільства. В усі сфери виробництва, транспорту, медицини та побуту приходить передове обладнання та технології. Освіта повинна враховувати вимоги сьогодення та майбутнього. Навчання програмування, напевно, як ніяке інше сприяє формуванню вмінь та навичок актуальних в різних сферах сучасного інформаційного суспільства. Це завдання потрібно втілювати у відповідності до відомого в педагогіці способу супутнього формування в учнів прийомів інтелектуальної діяльності. Він знайшов своє відображення у працях Ю. К. Бабанського [15], В. В. Краєвського [135], І. Я. Лернера [154].

Варто вказати, що на значення навчання програмування для розвитку деяких додаткових навиків вказували дослідники, які вивчали систему підготовки професійних програмістів. Так, З. С. Сейдаметова відзначає, що для випускників-бакалаврів комп'ютерних наук важливі і деякі додаткові навички, які можуть бути сформовані в контексті комп'ютерних наук, проте мають загальний характер і корисні також і в багатьох інших контекстах [254, с. 81]. С. А. Волошинов, розглядаючи значення навчання програмування під час підготовки майбутніх судноводіїв, зазначає: «Алгоритмічна підготовка сучасного фахівця за спеціальністю «Судноводіння» сприяє розвитку динамічності мислення, його гнучкості, формуванню уміння розділяти складний об’єкт на прості складові, визначати взаємозв’язки між ними. Все це необхідно для вивчення і побудови формальних моделей в будь-якій наочній області і дозволяє навчитися такому підходу до будь-якого завдання, при якому його рішення виступає як об’єкт конструювання і винаходу [41, с. 104]». О. М. Спірін, визначаючи послідовність вивчення тем курсу інформатики педагогічного начального закладу, вказує, що розділ основи алгоритмізації та процедурного програмування повинен передувати вивченню елементів штучного інтелекту [272, с. 53].

Отже, з усіх можливих завдань, які вирішуються в процесі навчання програмування, найважливішою є розвиток тих рис особистості учня, що

допоможуть йому реалізувати себе в інформаційному суспільстві. Максимально ефективним можна вважати таку організацію навчання програмування в ЗНЗ, під час якої учні отримають найбільшу користь від реалізації всіх завдань, що виконує навчання програмування. Поряд із цим, пряме завдання навчання програмування не втрачає свого значення. Навчання програмування варто здійснювати так, щоб максимально уникнути переучування тих учнів, що в майбутньому оберуть фах програміста, якомога краще сприяти формуванню в учнів загальнокористувацьких навиків та якомога продуктивніше розвивати риси особистості, що знадобляться їм в різних сферах дорослого життя.

Виходячи з того, що основним завданням навчання програмування є загальний розвиток особистості учня, доцільно визначити, на які ж сторони особистості учня воно впливає, яким чином цей вплив відбувається.

### 1.1.3. Розвиток мислення

Протягом усієї історії розвитку та становлення комп'ютерної техніки виникала потреба з'ясувати, які риси, здібності, вміння потрібні людині, для того, щоб ефективно створювати комп'ютерні програми. Дане питання досліджували Ф. Брукс [28], Г. Вейнберг [359], Н. Вірт [36], Л. В. Гришко [61], Е. Дейкстра [70], Д. Кнут [123], С. Макконнелл [162], А. П. Єршов [77], М. І. Жалдак [82], Н. В. Морзе [185], Ф. С. Ільясова [319], З. С. Сейдаметова [254], М. Л. Смульсон [268], О. М. Спірін [272] Б. Шнейдерман [321], Ф. В. Шкарбан [319] та інші. Вивчаючи когнітивне значення навчання програмування, важливо з'ясувати не лише те, які розумові здібності необхідні програмісту, а й те, розвитку яких рис особистості сприяє навчання програмування. Це досить взаємопов'язані завдання. Оскільки під час навчання програмування учень здійснює активну діяльність, можна стверджувати, що ті здібності, які будуть при цьому задіяні, і будуть зазнавати найбільшого розвитку.

На початкових етапах становлення комп'ютерної техніки для характеристики розумових здібностей, необхідних програмісту, широко

використовувався термін «програмістський стиль мислення». Емпіричні спостереження феномена стилю мислення «програміста» були узагальнені у працях А. П. Єршова: нове поняття, яке неточно характеризує лише професійні особливості мислення, було замінене специфічною назвою «операційний стиль мислення» [76].

Можна узагальнити основні риси, що А. П. Єршов наводив як необхідні для стилю мислення програміста (програмістському, операційному мисленню):

1. Уміння планувати структуру дій, щоб досягти мету фіксованим набором засобів;

2. Уміння створювати інформаційні структури (моделі) для опису об'єктів і систем;

3. Уміння здійснювати пошук даних потрібних для розв'язання задачі;

4. Уміння правильно, чітко і однозначно сформулювати думку в зрозумілій співрозмовнику формі і правильно сприймати текстові повідомлення;

5. Звичка своєчасно звертатися до ЕОМ при вирішенні завдань з будь-якої області [76, с. 9].

Н. В. Морзе визначає операційне мислення як сукупність умінь:

– формалізовувати задачу;

– виділяти в ній логічно самостійні частини;

– визначати взаємозв'язки в цих частинах;

– спроектовувати алгоритм розв'язання з допомогою технологій «зверху-вниз» та «знизу-вверх»;

– добирати якомога ефективніший шлях отримання розв'язку;

– інтерпретувати та аналізувати результат [185, с. 3].

Особливості операційного стилю мислення спочатку пов'язували з необхідними для професійного програміста навиками і уміннями. Такі здібності необхідні були для ефективного використання комп'ютера, що вже на той час вважався надзвичайно перспективним інструментом вирішення інформаційних

завдань. Спрямовані на обслуговування лише комп’ютерного обладнання, ці уміння і навички здавалися вузько спрямованими, технологічними. Але вже тоді дослідники почали стверджувати важливість такої психологічної категорії, як операційний стиль мислення не лише у зв'язку з широким і постійно зростаючим розповсюдженням комп'ютерів. Ті чи інші алгоритми використовуються у практично всякій людській діяльності, вони властиві най різноманітнішим об’єктам, процесам, навіть далеким від обчислювальної техніки.

Дослідження того, які саме інтелектуальні здібності необхідні програмісту, проводив відомий засновник теорії програмування Дональд Кнут. Він намагався знайти відмінності розумової діяльності, що відбувається при розв’язанні математичних задач, та тієї, що має місце під час написання комп’ютерних програм. У статті «Алгоритмічне мислення і математичне мислення» Д. Кнут проводить опис довільних задач, що розглядаються у підручниках з математики. Усі розглянуті задачі виявилися надзвичайно різними. Автор робить висновок, що: «не існує «математичного мислення» як окремого, ізольованого поняття; математики використовують багато різних способів мислення, а не тільки один [123]». Поряд із цим свої спостереження Кнут оформив у таблицю, де вказав дев’ять різних типів розумових дій, необхідних під час розв’язання тих чи інших математичних задач:

1. Операції з формулами;
2. Відображення дійсності;
3. Поведінка (властивості) функцій;
4. Зведення до більш простого випадку;
5. Операції з нескінченністю;
6. Узагальнення;
7. Абстрактні міркування;
8. Використання структур даних;
9. Алгоритми.

Намагаючись знайти відмінності в мисленні математика та програміста, Д. Кнут зазначає: «Мені здається, що більшість типів мислення, перерахованих у таблиці, часто використовуються у програмуванні, так як і в математиці, за явним винятком «операцій з нескінченністю [123]».

О. В. Копаєв визначає алгоритмічний стиль мислення у його взаємозв'язку з алгоритмічною діяльністю: «Алгоритмічний стиль мислення є невід'ємним складником алгоритмічної діяльності. Тому специфіку алгоритмічного стилю мислення, як і будь-якого іншого, принципово неможливо виявити без аналізу предметної галузі, де цей стиль себе виявляє. Власне, алгоритмічний стиль мислення ми визначаємо як систему мисленнєвих способів дій, прийомів, методів та відповідних їм мисленнєвих стратегій, що спрямовані на розв'язування теоретичних та практичних задач, і результатом яких є алгоритми, як специфічні продукти людської діяльності. Під алгоритмічною діяльністю ми розуміємо діяльність, метою якої є створення, пізнання та перетворення алгоритму» [132].

Л. В. Гришко узагальнила перелік особистісних та професійних якостей, які відомі теоретики програмування (Е. Дейкстра, Б. Шнейдерман та М. Л. Смульсон) вважали необхідними програмісту [63]. Зокрема, Л. В. Гришко розглядає вимоги до програміста, розділивши їх на дві групи: «Якості, які властиві програмістові, що пов'язані безпосередньо із створенням програмного продукту» та «Психологічні і загальнолюдські риси, які повинні бути притаманні програмістові». До першої групи увійшло п'ятнадцять різних характеристик, кожна з яких є узагальненням певного комплексу знань, умінь та навичок. Наприклад, «здатність розуміти програми» розглядається на трьох рівнях: нижній рівень (розуміння кожного рядка коду), середній рівень (розуміння структури алгоритму і даних) і вищий рівень (розуміння загального призначення програми) [63, с. 117].

Друга група – «Психологічні і загальнолюдські риси, які повинні бути притаманні програмістові», містить перелік із двадцяти чотирьох якостей,

практично кожна з яких теж є комплексом різних особливостей особистості програміста. Наприклад, «наполегливість» уточнюється так: «наполеглива людина володіє необхідною для виконання роботи ініціативою [61, с. 118]».

На основі розглянутих досліджень можна зробити висновок, що діяльність програміста вимагає від нього самого високого розвитку різних психологічних та інтелектуальних якостей, серед яких першочергово виділяється мислення у більшості його формах та проявах. Переважна більшість таких якостей зазнає найактивнішого розвитку під час навчання програмування. Швидше за все, не має вирішального значення, за якою системою та у якому порядку розглядати розвиток особистості учня в процесі навчання програмування. Для цього, наприклад, можна застосувати популярну нині «таксономію Блума», яку вже широко впроваджують для визначення шляхів розвитку мислення учня під час навчання інформатики [83; 134].

Згідно таксономії Блума розрізняють шість досить різносторонніх та визначальних навичок мислення: запам’ятовування, розуміння, використання, аналіз, оцінювання, створення [164, с. 173]. Започаткована у п’ятдесятих роках минулого століття американським психологом та педагогом Бенджаміном Семюелом Блумом класифікація рівнів мислення стосується когнітивних процесів у вищій нервовій діяльності людини [328]. Закономірно, що в діяльності програміста присутні постійне та активне пізнання, застосування, аналіз, синтез, творчість, оцінювання. Отже, розроблена для характеристики навчальної діяльності учнів таксономія Блума також відображає надзвичайно необхідні складові професійної діяльності фахівця в сфері розробки програмного забезпечення.

### 1.1.4. Зв'язок природної мови та мислення

Комп’ютерні мови програмування, подібно до мов людського спілкування, розвиваються та змінюються у взаємозв’язку з інформаційними завданнями, які вони мають виконувати. Ще у першій половині 20-го століття відомий психолог П. П. Блонський описав зв’язок мислення з мовленням на

прикладі формування побутової розповіді представниками різних народів [23]. Для з’ясування процесу розвитку природних людських мов Блонський використовував той факт, що представники деяких етносів зберегли елементи спілкування, притаманні давньому суспільству. Блонський зазначає, що людська мова поступово відривалась від дії та предмету, які вона описувала, в сторону формування більш загальних абстрактних понять. Це проявляється в тому, що оповідачі народів, де мова знаходилась на більш ранніх стадіях розвитку, широко застосовували не лише міміку, жестикуляцію, інтонацію, а й багатослівне уточнення, повторення. Шлях формування мов різних народів проходив у напрямку відокремлення слова від предмету, який воно позначає. «Слово начинало выступать на первый план. Рассказ становится из рассказа-действия просто рассказом. Но на первых порах этой стадии, вероятно, как само рассказывание, так и слушание его были не таким простым делом, как сейчас [23, с. 142]». Фактично, людина проектувала дійсність на більш зручну і універсальну знакову систему, якою і є мова спілкування.

Стосовно навчання програмування, то воно також має багато схожого з тим, як людина вивчає природну мову. Мислення тісно пов’язане з мовленням. Формування мовлення дитини супроводжується розвитком її мислення [46]. Оволодіння мовою програмування неодмінно розвиває мислення, впливає на його розумові можливості.

Навчання програмування постійно супроводжує діяльність програміста. Програміст удосконалює своє володіння мовою програмування уже в процесі її застосування. Такий розвиток відбувається впродовж усієї кар’єри програміста. Це подібно до того, що відбувається з спілкуванням людей. Еволюція людства супроводжується розвитком людського спілкування, розвитком мов. Людина не лише створила мову, а й оволоділа нею. «Люди были теми, кто создал язык, но они же были и теми, кто усвоил язык, и каждый новый этап языкового творчества происходил на базе уже определенно усвоенного языка. Люди не только создавали язык, но они и выучивались ему [23, с. 153]».

Мовлення пройшло досить тривалий етап еволюції, що супроводжувався абстрагуванням від реальної дії оповідача. Свого часу, з появою писемності, слово відділилося ще й від процесу його промовляння, звучання. До появи писемності мислення було тісно пов’язане з усним мовленням. Слово, а отже і думка, що йому передувала, передбачали наявність слухача. Писемність породила нову форму існування слова зокрема та й мови в цілому. Слово змогло існувати беззвучно. «Блаженный Августин зафиксировал этот монумент. Его учитель, святой Амвросий, удивлял современников тем, что читал молча: «И необычайное зрелище: сидит в комнате человек с книгой и читает, не произнося слов [4, с. 28]».

У цих історичних довідках проглядається, за аналогією, важлива особливість програмування як розумової діяльності. Комп’ютерна програма для програміста, подібно до оповідання для оповідача, існує в двох дещо різних формах. У кінцевому випадку це має бути процес виконання програми – аналог конкретного звучання слова. Цьому передує більш абстрактна форма: програма (як і слово) спочатку формується у вигляді ідеї, у формі алгоритму записаного зрозумілим людині способом чи усвідомленого. Уміння програмування – це, подібно до усного читання без говоріння, уміння усвідомлювати програму окремо від її виконання. У літературі з галузі програмування феномену «розумінню програми» часто приділяється окрема увага. Наприклад, Д. Бентлі, описуючи алгоритм двійкового пошуку, виокремлює його аналіз главою під назвою «Розуміння програми» [18, с. 56].

Поряд із цим, у програмуванні розділення конкретної реалізації програми та створеного для цього коду існує не лише у свідомості програміста. Розроблена та відлагоджена, збережена на носіях програма та програма, що виконується, – це речі досить різні.

### 1.1.5. Добір мови навчання програмування

Навчання програмування передбачає вивчення певної мови програмування та розвиток певної системи мислення, притаманної даному

різновиду людської діяльності. У сучасному інформаційному суспільстві така діяльність, як навчання, супроводжує людину впродовж усього її життя. Усяке навчання – це, крім усього, ще й оволодіння певною системою мовних позначень, термінів, «жаргону», а отже, і пов'язаної з ними системи мислення та світосприйняття. Для продуктивного життя в інформаційному суспільстві важливо вміти швидко навчатися, оволодівати новими системами світобачення та мислення. Проте велике значення має певним чином спрямований розвиток вміння навчатися. Важливо вміти швидко оволодівати саме тими системами, що будуть мати місце у сучасному високотехнологічному виробництві, побуті, культурі. Оволодіння, наприклад, латинською мовою, беззаперечно, дає певний досить корисний досвід навчання. Але велика користь від масового навчання латинської мови сумнівна. Швидше за все, існують інші інтелектуальні заняття, які більш потрібні сучасній молодій людині, ніж практика навчання нині мертвої мови спілкування. Відому тезу Я. А. Коменського «усіх учити всього» [128, с. 126] не можна застосовувати буквально. Отже, має значення не лише сам факт навчання програмування, а й те, як воно буде здійснюватись, якою мовою програмування учні будуть оволодівати, яка система мислення буде формуватись у результаті такого навчання. Дейкстра писав: «Найважливішою, але більш непомітною властивістю будь-якого інструменту є вплив його на формування звичок людей, які ним користуються. Коли цей інструмент – мова програмування, його вплив, незалежно від нашого бажання, позначається на нашому способі мислення» [70, ст. 11].

Кількість різноманітних мов програмування постійно зростає. Поява нових мов програмування та парадигм, на яких вони базуються, – закономірний процес еволюції ІКТ [150; 282]. Серед значної кількості мов, на яких можна створити ту чи іншу комп'ютерну програму, існують такі, що не лише широко застосовуються, а й мають подібні характеристики. Поряд із цим, є й специфічні та малоперспективні мови програмування: рідко використовувані, морально застарілі, незручно організовані. Добір мови програмування, що буде

використовуватись для написання певної комп’ютерної програми, є надзвичайно важливим етапом розробки програмного забезпечення [124]. Не менш важливим є добір мови програмування, що буде використовуватися для навчання [302].

Стосовно принципів добору мови програмування, що буде використовуватися для написання програмного забезпечення, побутує два підходи. Перший спирається на певну універсальність більшості існуючих мов програмування. Його можна сформулювати так: «Будь-яка мова програмування дозволяє створити програму, яка вже була розроблена з використанням іншої мови». Зокрема, Тімоті Бад вважає таке твердження прямим наслідком з принципу Чьорча: «Будь-яке обчислення, для якого існує ефективна процедура, може бути реалізовано на машині Тьюрінга» [16, с. 9]. Твердження сформульоване аналогічно до принципу Чьорча стосовно мов людського спілкування буде таким: «Людську думку, сформульовану однією мовою спілкування, можна передати мовою іншою». Втім принцип Чьорча не підлягає доведенню з причини відсутності строгого визначення поняття «ефективна процедура».

Той же таки Т. Бад наводить твердження радикально протилежне до висновків, що випливають з принципу Чьорча. Він вказує на хоч і суперечливу також недоведену, але широковідому серед лінгвістів гіпотезу Сапіра–Ворфа: «Люди живуть не тільки в об'єктивному світі речей, не тільки у світі соціальної діяльності, як це зазвичай вважають, вони значною мірою перебувають під впливом тої конкретної мови, яка є засобом спілкування для даного суспільства. Було б помилковим вважати, що ми можемо повністю усвідомити дійсність, не вдаючись до підтримки мови, або що мова є побічним засобом вирішення деяких окремих проблем спілкування і мислення. Насправді «реальний світ» значною мірою несвідомо будується на основі мовних норм даної групи» [197, с. 199]. Цю гіпотезу ще називають гіпотезою лінгвістичної відносності і її скорочене трактування звучить так: «Мова визначає мислення і

спосіб пізнання» [140, с. 67]. В контексті цієї гіпотези Сапіра–Ворфа Т. Бад зазначає: «Індивідуум, який використовує деяку мову, має змогу уявити або придумати щось, що не може бути переведеним або навіть зрозумілим індивідуумами з іншого мовного середовища. Таке відбувається, якщо в мові другого індивідуума немає еквівалентних слів і відсутні концепції або категорії для ідей, залучених у розглянуту думку [16, с. 8]». Отже, другий підхід до добору мов програмування базується на тому факті, що від мови, якою здійснюється написання програми, залежить можливість її реалізації.

Відкинувши крайні точки зору, варто зважити на те, що вибір мови написання програми певною мірою визначає, як ефективність її функціонування, так і зручність самого процесу розробки. Дуже часто, щоб продемонструвати наскільки важливим є добір мови програмування, наводиться приклад зв'язку між мисленням і мовою людського спілкування. Так, у мові північного народу ескімосів більше сотні різних слів перекладаються, як сніг. Для ескімосів важливо розрізняти не лише сніг мокрий та пухкий, а й багато інших його станів [16, с. 6].

Добір мови програмування, зазвичай, визначається трьома важливими умовами:

– характеристиками, особливостями самої мови;

– наявністю зручного в навчанні, доступного до використання середовища програмування;

– наявність методичної підтримки (інформаційно-дидактичного та навчально-методичного забезпечення) [302].

Найважливішими характеристиками мови програмування, що використовується для навчання, є призначення, підтримувані парадигми, розповсюдженість, алфавіт та особливості синтаксису.

Специфіка завдань, до вирішення яких можна застосовувати мову програмування, вказує на її призначення.

Розповсюдженість мови програмування – це практично синонім її популярності. Розповсюдженість мови програмування в галузі професійної розробки програмного забезпечення є додатковим стимулом до її вивчення, стимулює інтерес учнів до навчання.

Стосовно алфавіту, то мови програмування, що створювалися з навчальною метою, загалом мають більш доступний для сприйняття набір службових слів та інших елементів алфавіту. Таку доступність важко досягти чимось іншим, ніж звуженням можливостей мови та зменшенням ефективності алгоритмічних конструкцій, що можуть бути реалізовані мовою програмування.

Мови програмування під час еволюції успадковували одна від одної правила написання та оформлення програм, закономірності використання та назви команд. Ці правила та домовленості, з певним уточненням, можна віднести до синтаксису мови програмування. Серед мов програмування, що традиційно використовуються для навчання, можна, досить не строго, за схожістю між собою, виділити три найбільш поширених різновидності синтаксисів:

1) синтаксис мов Basic та Visual Basic;

2) мов Pascal та Delphi;

3) синтаксис мов схожих до мови «С» (С++, С#, PHP, Java та ін.) [334].

Поряд із цим, за особливостями алфавіту та синтаксису існує низка спеціально розроблених навчальних мов програмування. Деякі з навчальних мов досить розвинені і носять універсальний характер: НАМ (навчальна алгоритмічна мова), Рапіда. Є й такі, що використовуються для розв’язання незначної кількості навчальних завдань, зокрема Лого (черепахова графіка). Мовами програмування, певною мірою, можна вважати набори та правила використання команд управління виконавцями в навчальних середовищах «Сходинки до інформатики» та «Скарбниця знань». Кожна з навчальних мов, зазвичай, має свій специфічний синтаксис та набір команд.

Віддаючи перевагу тому чи іншому синтаксису, варто враховувати, що здебільшого синтаксис мов подібний до синтаксису мови «С» дещо складніший. Проте основним аргументом на користі використання у навчанні «С-подібного» синтаксису є його розповсюдженість, решта різновидів синтаксису мов програмування не настільки широко використовується в професійній діяльності. Проблема переходу з навчальної на професійну мову програмування є актуальною для більшості фахівців з розробки програмного забезпечення. Використання в навчанні мов програмування з «С-подібним синтаксисом» дозволяє практично уникнути проблем такого переходу [311].

### 1.1.6. Еволюція методологій програмування

У перші роки використання комп'ютерної техніки процес розробки програми значною мірою визначався не лише можливостями обчислювальних машин, а й особистістю програміста. Основними характеристиками комп’ютерів вже тоді були обсяг оперативного запам'ятовуючого пристрою і швидкодія. Ці параметри, у порівнянні з сучасними комп'ютерам, були дуже і дуже малими.

Щоб розробити програму, яка використовує щонайменше оперативної пам'яті і працює якнайшвидше, програмістам доводилося використовувати певні нестандартні рішення. У результаті цього програми виходили несхожими одна на одну і заплутаними. Був такий час, коли взагалі вважалося, що чим більш нестандартно написана програма, тим вища кваліфікація програміста. Складність програми зростала за рахунок вставок з використанням команд безумовного переходу (в деяких сучасних мовах програмування це команда «goto»), безсистемного додавання команд, багаторазового використання одних і тих же регістрів пам’яті для даних різних типів, інших нетривіальних рішень.

Завдяки такому стилю програмування, який краще назвати відсутністю стилю, програміст згодом не міг розібратися навіть у власній програмі. Програми такої структури отримали назву BS-програми (абревіатура від "Bowl Spaghetti"), що означає "страва спагеті" [274].

Традиційно вважається, що в основі теорії програмування лежить поняття алгоритму. Зазвичай, знайомство з програмуванням розпочинають саме з вивчення цього поняття. Історію виникнення терміну «алгоритм» детально описав Кнут [124, с. 19-20]. Він же дав визначення властивостей алгоритму [124, с. 23-25]. Нині існує багато підходів до визначення поняття алгоритм та його використання у навчанні [139; 155; 165165]. Зокрема, досліджуючи це поняття у різних його трактуваннях, О. В. Копаєв. визначає алгоритм як модель алгоритмічного процесу. «Алгоритм і алгоритмічний процес – це різні об’єкти, причому вони мають різну природу. Разом з тим, вивчаючи алгоритм, можна зробити й певні висновки про алгоритмічний процес, який він задає. Адже алгоритм повністю детермінує алгоритмічний процес, а отже, й повністю його описує» [132].

Створення сучасних комп’ютерних програм – це не тільки і не стільки розробка окремих алгоритмів як побудова цілісних інформаційних моделей, які максимально відображають реальні явища чи предмети, що моделюється. Під час розвитку галузі програмування формувались та змінювались підходи до побудови комп’ютерних програм, що зумовлювалось постійним ускладненням самих програм, зростанням складності завдань моделювання предметної області. Сформувався термін, поняття, що вказує на ті чи інші підходи до написання комп’ютерної програми – *парадигма програмування.* Як модель чи приклад, а також як організуючий підхід – це слово використовував Роберт Флойд, лауреат премії Тюрінга 1979 року, в лекції «Парадигми програмування» [150, с. 159-174]. Парадигми в програмуванні – це спосіб концептуалізації, який визначає, як проводити обчислення і як робота, виконувана комп'ютером, повинна бути структурована і організована [150, с. 163].

Більшість сучасних парадигм програмування мають в своїй основі певні принципи та концепції. За своїм походженням одні парадигми якимось чином пов’язані з іншими: базуючись на них, продовжуючи їх ідеї чи, навпаки, використовуючи радикально протилежні принципи. Відносно найменування

парадигм програмування існує ціла низка термінів, що історично сформувалися [207]. Проте поки що не існує єдиної системи найменувань парадигм програмування та їх чіткої класифікації. Назви окремих парадигм мають декілька синонімів або розглядаються в складі інших, більш загальних підходів до написання програмного забезпечення. Наприклад, як противага парадигмі ООП (об'єктно-орієнтованого програмування), називають програмування процедурне, структурне чи імперативне, хоча між цими поняттями є суттєві відмінності. Наводимо найбільш розповсюджену, як на наш погляд, класифікацію парадигм програмування (див. табл. 1.1).

*Таблиця 1.1.*

**Сучасні парадигми програмування**

| Назва | Назва англійською | Ключові ідеї, принципи, концепції. | Зв'язок з іншими парадигмами |
|---|---|---|---|
| Імперативне програмування | Imperative programming | Процес отримання результатів описується як послідовність операцій, вказівок на їх виконання. | Перша історично. Є основою багатьох інших. |
| Процедурне програмування | Procedural programming | Концепція виклику процедур – підпрограм, методів або функцій. Будь-яку процедуру можна викликати з будь-якого місця програми в момент виконання програми, в тому числі з інших процедур або з самої себе. Процедура (інші назви підпрограма, метод) – це послідовність команд, які слід виконати. | Удосконалення імперативної парадигми, що полегшило розробку великих програм |

| | | | |
|---|---|---|---|
| Структурне програму-вання | Structured program-ming | – здійснюється за низсхідним принципом, зверху-вниз;<br>– увесь проект складається з модулів, з одним входом і одним виходом кожен;<br>– використовуються лише на три типи основних структур: лінійні, розгалуження та цикли;<br>– оператор передачі керування в будь-яке місце програми (goto) принципово не використовуються;<br>– документація створюється одночасно із написанням програми у вигляді коментарів [274, 275]. | Практично синонім процедурного програмування, наголос робиться на вимогах до структури програми |
| Модульне програму-вання | Module program-ming | Організація програми у вигляді сукупності незалежних блоків, структура і функції яких підпорядковуються певним вимогам. Такі блоки називаються модулями [33]. Переважно здійснюється за вверхсхідним принципом, знизу-вверх. | Удосконалення структурної та процедурної п-игм |
| Об’єктно-орієнтоване програму-вання (ООП) | Object-oriented program-ming | Програма розглядається як множина «об'єктів», що взаємодіють між собою. Кожен об’єкт є екземпляром певного класу, а класи є членами певної ієрархії наслідування. | Має все ту ж імперативн у основу |
| Прототипне програму-вання | Prototype-based program-ming | Різновид об'єктно-орієнтованого програмування, при якому відсутнє поняття класу, а повторне використання (успадкування) проводиться шляхом клонування існуючого примірника об'єкта — прототипу. | Удосконалення об’єктно-орієнтованої парадигми |
| Аспектно-орієнтоване програму-вання | Aspect-oriented program-ming | Дозволяє виокремити перехресну (наскрізну) функціональність. Побудова гнучких до змін програмних систем шляхом додавання нових аспектів (функцій), що забезпечують, наприклад, безпеку, взаємодію компонентів з іншим середовищем, а також синхронізацію одночасного доступу частин програмної системи до даних і виклик нових компонентів із загальносистемних середовищ [147]. | Удосконалення об’єктно-орієнтованої парадигми |

| | | | |
|---|---|---|---|
| Предметно-орієнтоване програму-вання. Синонім терміну суб'єктно-орієнтоване програмування (СОП) | Subject-oriented program-ming | Метод побудови об'єктно-орієнтованих систем, як композиції суб'єктів. В цілому СОП включає: розбиття системи на суб'єкти та написання правил їх композиції. СОП доповнює об'єктно-орієнтоване програмування, вирішуючи проблеми, що виникають при розробці великих систем, при вирішенні завдань інтеграції та переносимості [214]. | Розвиває ідеї, які певною мірою є протилагою ідеям об'єктно-орієнтованої парадигми |
| Функціо-нально-орієнтоване програму-вання | Feature-oriented program-ming | Загальна парадигма компонування програм у ряд програмних продуктів. Програма представляється стосом шарів. Кожен шар додає функціональність до попередніх, а різне комбінування цих шарів дає на виході різні програми. | Суттєво відрізняється від функціонального програмування. |
| Декларативн е програму-вання | Declarative program-ming | Програма описує який результат необхідно отримати, замість описання послідовності отримання цього результату. Замість алгоритму розв'язку задачі програміст складає її логічну специфікацію [271, с. 5]. | Мають не імперативну основу. |
| Логічне програму-вання | Logic program-ming | Логічне програмування має декларативну основу, засноване на автоматичному доказі теорем, базується на одному з розділів математичної логіки – логіці предикатів першого порядку [271, с. 4]. | |
| Функційне програму-вання | Functional program-ming | Програмування засноване на функціях, що приймають функції, як параметри, та повертають функції, як результати. Функційне програмування наголошує на застосуванні функцій, на відміну від імперативного програмування, яке наголошує на змінах у стані та виконанні послідовностей команд [9]. | |

У процесі розвитку галузі програмування нові парадигми поступово доповнювали уже існуючі новими концептуальними положеннями. Можна визначити таку основну траєкторію розвитку найбільш розповсюджених парадигм: імперативна – процедурна – структурна – модульна – об'єктно-орієнтована – різновиди подальшого розвитку ідей ООП: прототипна, аспектно-орієнтована, суб'єктно-орієнтована. Даний шлях розвитку започатковується імперативною парадигмою послідовного виконання завдань. В основі імперативної парадигми лежить «фоннейманівська» архітектура комп'ютера [150, с. 89]. Незважаючи на те що така архітектура вважалася застарілою ще 30 років тому, практично усі сучасні комп'ютери базуються саме на цій архітектурі [276, с. 36]. Розвиток парадигм програмування, певною мірою, зумовлений необхідністю долати обмеження моделі комп'ютера фон Неймана.

Рушійною силою еволюції методів та засобів програмування є розширення сфер застосування ІКТ. Поява все більш і більш досконалих мов програмування є наслідком прикладних досліджень з методології проектування, декомпозиції, абстрагування і ієрархій. Хоча ідеї ООП дещо розширили обмеження, що накладає імперативна парадигма програмування, цей напрям розвитку не вичерпує всіх концепцій розробки програмного забезпечення. Окрім того, ООП може реалізовуватись і на неімперативній основі. На противагу парадигмам, що були започатковані на імперативному програмуванні, існують та розвиваються неімперативні парадигми декларативного, логічного та функційного програмування. Логічне і функуційне програмування є, фактично, різновидами програмування декларативного. Ці парадигми мають багато переваг і надають значні можливості для розробки складних інтелектуальних систем, зокрема, систем штучного інтелекту [271; 272].

Більшість мов програмування є мультипарадигмальними, тобто придатними для написання програм у відповідності до різних парадигм. Наприклад, мови Pascal та C, створені за парадигмами процедурного,

структурного та модульного програмування. Мови C++, Object Pascal, Java, C# дозволяють розробляти об’єктно-орієнтовані програми, але їх можна використовувати і дотримуючись процедурної, структурної та модульної парадигм. Та ж таки мова C# має досить розвинені засоби декларативного, зокрема функціонального програмування.

Одночасне використання різних парадигм при розробці програм часто веде до серйозних труднощів при їх супроводі, значно ускладнює подальший розвиток та удосконалення. При виборі парадигми програмування потрібно враховувати багато передумов, тому що кожна з них має свої особливості і межі застосування.

### 1.1.7. Покоління мов програмування

З величезної кількості мов програмування, що з'явилися за період розвитку інформаційних технологій, лише найбільш зручні і досконалі були прийняті розробниками і відстояли своє право на існування. Аналізуючи мови програмування та обставини їх появи, можна виявити множину спільних рис. Це дозволяє згрупувати мови за основними використовуваними принципами і виділити покоління в їхньому розвитку (див. табл. 1.2). За основу розробленої нами таблиці було взято класифікацію поколінь мов програмування, що наводить Г. Буч [31, с. 22], доповнену відомостями про мови, що з’явилися пізніше цієї публікації.

На думку переважної більшості дослідників мов програмування п’ятого покоління все ще не існує. Основний висновок з узагальнення поколінь мов програмування: багато ідей, покладених в основу сучасних мов програмування, з'явилися в тому чи іншому вигляді вже до 1970 року.

Усі подальші мови, за невеликим винятком, є нащадками або результатом узагальнення і розвитку перерахованих вище. Цьому багато в чому сприяло як широке поширення міні- і мікро-ЕОМ і пов'язане з цим зростання числа як розробників програмного забезпечення, так і різноманіття операційних систем і різноманітних сфер застосування інформаційних технологій [31, с. 23].

*Таблиця 1.2.*

**Покоління мов програмування**

| Покоління (Generation Language) | Роки | Мови програмування | Призначення |
|---|---|---|---|
| Перше покоління (1GL) | (1954 – 1958) | FORTRAN I<br>ALGOL-58<br>Flowmatic<br>IPL V | Математичні формули |
| Друге покоління (2GL) | (1959 – 1961) | FORTRANII | Підпрограми, роздільна компіляція |
| | | ALGOL-60 | Блокові структури, типи даних |
| | | COBOL | Опис даних, робота з файлами |
| | | Lisp | Обробка списків, покажчики, збір сміття |
| Третє покоління (3GL) | (1962 – 1970) | PL/1 | FORTRAN + ALGOL + COBOL |
| | | ALGOL-68 | Більш строгий наступник ALGOL-60 |
| | | Pascal | Простіший наступник ALGOL-60 |
| | | Simula | Класи, абстрактні дані |
| Четверте покоління (4GL) | (1971 – дотепер) | SQL, SGML ( HTML, XML ), Prolog | Вузькоспеціалізовані декларативні мови |
| | | C ++ | об'єктно-орієнтоване розширення мови C |
| | | Perl | обробка текстів, динамічна генерація Web-сторінок |
| | | Python | близький до Perl, але більш строгий і логічний |
| | | Delphi | Візуальне ООП на базі мови Pascal |
| | | Java | ООП, кросплатформна проміжна машинна мова |
| | | C# | ООП, програмна платформа .NET |

### 1.1.8. Зародження об’єктно-орієнтованого програмування

У своїх дослідженнях Г. Буч вказує на особливості входження терміну об'єкт у програмування:

– поява об'єктного підходу пов'язана з прогресом у галузі архітектури ЕОМ, розвитком мов програмування, розвитком методології програмування;

– вперше термін об'єкт було використано при конструюванні комп'ютерів з особливою архітектурою;

– термін об'єкт з'явився практично одночасно в різних галузях, пов'язаних з комп'ютерами, для позначення того, що мало різні прояви, зберігаючи свою цілісність;

– загалом розвитком об'єктних абстракцій у програмуванні тісно пов'язаний зі створення об'єктно-орієнтованих операційних систем, технології побудови баз даних, дослідженням штучного інтелекту [31 с. 28].

Деякі поняття об’єктного підходу до написання комп’ютерних програм розглядалися вже в процесі розвитку структурної парадигми програмування. Зокрема У. Дал в збірнику статей зі структурного програмування розглядає терміни «клас» [68, с. 199] та «класи об’єктів» [68, с. 203].

Взаємодія даних та коду – це загальна схема побудови процедурно-орієнтованих програм. Дані описують за допомогою певних типів – цілочисельних, масивів, записів та ін. Код, що працює з даними, складається з підпрограм – процедур та функцій. Ці підпрограми різні для різних типів даних. При процедурному підході розробка великих проектів, де використовується багато різних структур даних, є досить складною. При створенні великих проектів центральне місце посідають дані, а не алгоритм. Зародження об’єктного підходу відбулося, коли виникла гостра необхідність мовної підтримки опису довільних об'єктів навколишнього світу. Це спонукало введення абстрактних типів даних. З цього приводу Шанкар зазначив: «Абстрагування, що досягається за допомогою використання процедур добре підходить для опису абстрактних дій, проте не годиться для опису абстрактних

об'єктів. Це серйозний недолік, оскільки в багатьох практичних ситуаціях складність об'єктів, з якими потрібно працювати, становить основну частину складності всього завдання [352]». В програмуванні терміном об'єкт позначили поєднання даних та коду, що безпосередньо опрацьовує ці дані. Підхід до написання програм на основі об'єктів отримав назву об'єктно-орієнтованого програмування – ООП.

Першою мовою, в якій знайшли своє вираження ідеї побудови програм на основі даних і об'єктів, стала мова Simula 67. Концепції, закладені в мові Simula, отримали свій розвиток в серії мов Smalltalk-72, -74, -76, -80, а також у мовах C ++ і Objective C. При внесенні об'єктно-орієнтованого підходу в мову Pascal з'явилася мова Object Pascal. У 90-х роках компанія Sun представила світу мову Java, яка є втіленням ідеї платформної незалежності і найбільш повної реалізації концепцій об'єктно-орієнтованого програмування, покладеної в основу мов Simula 67, Smalltalk, C ++.

Стосовно теперішнього використання мови програмування Smalltalk, то ця мова все ще надзвичайно актуальна [120; 355]. Нині широко використовується удосконалена версія цієї мови – Squeak. Існує навіть реалізація середовища об’єктно-орієнтованого програмування мовою Squeak для мобільних пристроїв [281].

Хоча принципи організації коду характерні для об'єктно-орієнтованої парадигми були, свого часу, новими для програмування, зародженням об’єктного підходу можна вважати появу теорії єдиної систематизації рослинного та тваринного світу Карла Ліннея. У 1735 році він опублікував свою основну працю під назвою «Система природи» («Systema Naturae»), що уславила його ім'я [24; 157]. Окремі елементи класифікації оточуючого світу, подібні до тих, що закладені в об’єктно-орієнтовану парадигму програмування, прослідковуються ще у працях древньогрецьких філософів, зокрема Платона [8].

Система втілена в об'єктно-орієнтованій парадигмі програмування широко використовується у повсякденному житті, що обґрунтовується особливостями людського світосприйняття, мислення, мови. В основу таких тверджень можна покласти дослідження авторів гіпотези лінгвістичної відносності [197; 140]. Здавалося б, далека від програмування гіпотеза лінгвістичної відносності неодноразово попадає в поле зору дослідників об’єктно-орієнтованої парадигми розробки програмного забезпечення. Зокрема, Брюс Еккель у книзі «Філософія Java» в епіграфі до розділу, де описуються принципи ООП, цитує Б. Л. Ворфа, одного з авторів цієї гіпотези: «Ми препаруємо природу, перетворимо її в концепції і приписуємо їм сенс так, як ми це робимо багато в чому, тому що ми всі є учасниками угоди, яка має силу в суспільстві поєднаному мовою і яка закріплена в структурі мови... Ми не можемо спілкуватися зовсім, окрім як погодившись з установленими цією угодою організацією та класифікацією даних [323, с. 18]». Напевно, природна притаманність ООП для людського мислення і є однією з причин того, що початківці у програмуванні часто здатні сприйняти основні ідеї об'єктно-орієнтованого програмування порівняно легко, в той час як люди більш обізнані в інформатиці через наявні уявлення вбачають складність в об’єктному написанні програм. Зокрема, так зазначав ще Алан Кей, основоположник об’єктно-орієнтованого програмування [120, с. 4].

Зародження характерних рис об’єктно-орієнтованого програмування розпочиналося з досить широкого застосування у програмуванні так званого способу «задавання завдання». Завдання ініціюється за допомогою передачі повідомлення, що містить вказівку на виконання завдання. Одержувач не лише приймає повідомлення, а й бере на себе відповідальність за виконання зазначеної в повідомленні дії. Далі розвивався принцип «маскування даних», згідно якого клієнтові не потрібно знати про засоби, за допомогою яких його запит буде виконано.

У традиційних мовах програмування маскування даних також є важливим принципом. Проте є відмінності між викликом процедури і передачею «доручення». Перша полягає в тому, що у «доручення» є цілком конкретний одержувач – агент, якому надіслано повідомлення. При виклику процедури немає настільки явно виділеного одержувача. У такому випадку можна домовитись, що одержувачем повідомлення є перший аргумент у виклику процедури. У більшості так і реалізуються одержувачі повідомлень, але це не є визначальним. Друга відмінність полягає в тому, що інтерпретація повідомлення (а саме метод, що викликається після прийому повідомлення) залежить від одержувача і є різним для різних одержувачів.

Загалом, сформувалися певні положення, фундаментальні характеристики ОО парадигми програмування у вигляді трьох ознак: інкапсуляції, поліморфізму, спадкування [16; 156; 236]. Проте один із перших основоположників ООП Алан Кей дещо по-іншому визначав основні ознаки цієї парадигми. За Кеєм їх також три [355]:

– **Об'єкт** – базова одиниця об'єктно-орієнтованої системи.

– Об'єкти можуть мати певні **стани** (перебувати в певних станах).

**– Надсилання повідомлення** – єдиний спосіб обміну даними між об'єктами.

На нашу думку, для щонайбільш повного опису сутності об’єктно-орієнтованого програмування доцільно узагальнити різні підходи до його трактування:

1. Усе є об'єктом.

2. Обчислення (операції) здійснюються шляхом взаємодії (обміну даними) між об'єктами, при цьому один об'єкт вимагає, щоб інший об'єкт виконав певну дію. Об'єкти взаємодіють, посилаючи й одержуючи повідомлення. Повідомлення – це запит на виконання дії, доповнений набором аргументів, що можуть знадобитися при виконанні дії.

3. Кожен об'єкт має незалежну пам'ять, яка складається з інших об'єктів. Також кожен об’єкт захищений від втручання у його внутрішню структуру.

4. Кожен об'єкт є представником класу, який виражає загальні властивості об'єктів.

5. У класі задається поведінка (функціональність) об'єкта. Тому всі об'єкти, що є екземплярами одного класу, можуть виконувати одні і ті ж дії.

6. Класи організовані в єдину деревоподібну структуру з загальним коренем, так звану ієрархією наслідування. Пам'ять і поведінка, що є екземплярами певного класу, автоматично доступні будь-якому класу, розташованому нижче в ієрархічному дереві.

### 1.1.9. Сучасна галузь програмування. Перспективи і розвиток галузі

Об'єктно-орієнтоване програмування все ж не вирішує всіх проблем, що постають під час розробки комп’ютерних програм. Сучасна галузь розробки програмного забезпечення бурхливо розвивається. Триває зародження нових парадигм програмування та розвиток існуючих. Важливо передбачити тенденції розвитку мов програмування, вже зараз мати уявлення про парадигми програмування, що з’являться у майбутньому. Це дасть змогу завчасно підготуватися до максимально ефективного використання майбутніх технологій. Передбачення подальшого розвитку технологій розробки програмного забезпечення важливі для галузі навчання програмування, адже дозволяють готувати учнів до вирішення завдань, що будуть їх у майбутньому чекати.

Передбачити подальші перспективи розвитку програмування надзвичайно важко, проте можна прослідкувати загальні тенденції, що притаманні сфері ІКТ та безпосередньо впливають на галузь розробки програмного забезпечення. В майбутньому неминучим стане:

– зростання швидкодії та загальних інформаційних потужностей комп’ютерної техніки, адже це диктується законом Мура [346];

– розширення і без того багатогранної сфери застосування ІКТ.

Існують окремі дослідження перспектив галузі розробки комп’ютерних програм. Одним із таких можна вважати розділ монографії Г. С. Теслера «Новітня кібернетика» – «Подальші шляхи розвитку мов та середовищ програмування [282, с. 206-211]». Зокрема Теслер передбачає:

1. Мови та системи програмування будуть поєднувати в собі різні орієнтації (системну і прикладну, символьні і обчислювальні, імперативні і декларативні і т.д.).

2. Отримає подальший розвиток абстрагування даних, знань, структур і засобів управління.

3. Мови матимуть ядро і спеціалізовані засоби для адаптації і саморозвитку, що дозволяють об'єднувати різні спеціалізовані та проблемно-орієнтовані підсистеми мови, а також створювати свої персоніфіковані версії.

4. На зміну нині поширеним структурно і об'єктно-орієнтованим технологіям прийде семантичне (смислове) програмування, що використовує у тій чи іншій мірі існуючі технології.

5. Майбутні мови і системи програмування будуть містити механізми забезпечення високого рівня відмовостійкості, гарантоздатності та прозорості створюваних програмних продуктів.

6. Нові мови забезпечувати накопичення отриманих даних і знань.

7. Поряд з мовами, орієнтованими на стаціонарні комп'ютерні платформи, будуть існувати їхні версії, орієнтовані на мобільні комп'ютерні платформи з засобами мережевої взаємодії.

8. Мови програмування матимуть засоби підтримки паралельної і паралельно-послідовної роботи.

9. Мови програмування будуть мати засоби підтримки особливостей використовуваної комп'ютерної платформи.

10. Мови програмування повинні забезпечувати стандарти на уніфікацію і переносимість програм.

Отже, загалом передбачення Г. С. Теслера стосуються екстенсивної складової розвитку галузі програмування. Тобто, вчений переважно говорить про нагромадження та урізноманітнення напрямків, можливостей, підходів.

У статті «Мови програмування через сто років [66]» дещо іншу картину змальовує відомий у світі програміст Пол Грем. Анонсує свій матеріал автор таким чином: «Це стаття про еволюцію мов програмування, сумне майбутнє Java і про те, на що витрачатимуть неймовірні обчислювальні ресурси, якими будуть володіти комп'ютери через сто років».

Грем розмірковує над тим, чи збережеться через 100 років потреба у програмуванні як такому а основним чинником розвитку мов програмування Грем вважає зростання швидкодії комп’ютерів [66, с. 2].

В цілому П. Грем прогнозує:

– розвиток мов програмування, що мають кращу, досконалішу і більш чітку логічну структуру;

– подальше зниження вимог до раціонального використання апаратних можливостей обчислювальної техніки та зростання вимог до раціонального використання зусиль програміста;

– повну чи часткову відмову від деяких типів даних;

– широке використання проміжної, до того ж багаторазової, трансляції коду програми на проміжні машинні мови для забезпечення кросплатформного використання програм [66, с. 4].

На жаль, автор практично не торкається питання розвитку парадигм програмування в цілому. Проте, у статті, окрім критики сучасних підходів до написання комп’ютерних програм, зазначається про певні перспективи використання об’єктно-орієнтованого програмування [66, с. 4].

Отже, в загальному зростання обчислювальних потужностей неминуче знизить вимоги до ефективності коду майбутніх комп’ютерних програм. Проте завжди існуватимуть галузі, що будуть вимагати максимально ефективного використання обчислювальних ресурсів. Сюди варто віднести, наприклад,

криптографію. Те, що сучасні мови програмування транслюються у проміжний код (байт код для Java чи Run Time Language для C#) також стало можливим завдяки зростанню потужностей комп'ютерів, адже кожне перекодування значно знижує швидкодію. Деякі мови програмування взагалі створені як інтерпретатори. Це потрібно для повторного використання коду, для забезпечення абсолютної кросплатформності.

### 1.2. Психолого-педагогічні особливості навчання інформатики за умов профілізації ЗНЗ

Впровадження профільного навчання є важливим етапом підвищення ефективності освіти. Ідеї профільності в педагогіці прослідковуються практично на всіх етапах її становлення. Але планомірне та масове впровадження профільного навчання відбувається лише впродовж кількох останніх десятиліть. Проблеми профільного навчання в педагогіці, в цілому, та в інформатиці, зокрема, знайшли своє відображення у працях багатьох українських та російських вчених [1; 19; 42; 50; 67; 110; 121; 138; 210; 249; 250; 257]. Профілізація сучасної шкільної освіти вимагає радикального оновлення освітньої парадигми, концептуальних підходів, цілей, форм та засобів навчання. Профільна школа найповніше реалізує принцип особистісно-орієнтованого навчання, що значно розширює можливості учня у виборі власної освітньої траєкторії [130] .

Визначальним є втілення в освітню науку, практику новітніх інформаційно-комунікаційних та психолого-педагогічних технологій, сучасне оновлення змісту та завдань навчання. При організації та здійсненні профільного навчання важливо орієнтуватися на швидкий розвиток інформаційно-комунікаційних технологій. Серед шкільних предметів, що зазнають найактивнішого впливу нових інформаційних технологій, важливе місце займає інформатика. Інформатика – наймолодша дисципліна, що вивчається в загальноосвітніх навчальних закладах. Зміст та інші складові цього предмету не лише швидко розвиваються, а й сприяють розвитку інших

шкільних дисциплін. Значною мірою розвиток та оновлення інформатики задає тон розвитку та оновленню шкільної освіти в цілому [83]. Частина розділів цього шкільного предмету вимагає оволодіння учнями спеціальними, професійно спрямованими знаннями та уміннями. До таких тем, у першу чергу, належать основи програмування.

Профільне навчання – вид диференційованого навчання, який передбачає врахування освітніх потреб, нахилів і здібностей учнів і створення умов для навчання старшокласників, відповідно до їхнього професійного самовизначення, що забезпечується за рахунок змін у цілях, змісті, структурі та організації навчального процесу [130]. Зустрічаються й інші означення профільного навчання. Наприклад, Н. Є. Мойсеюк вказує у визначенні профільного навчання три контексти:

– вид диференційованого навчання, який враховує освітні потреби, нахили, здібності учнів і створює умови для навчання старшокласників відповідно до їхнього професійного самовизначення;

– забезпечення можливостей здобуття загальноосвітньої, профільної, початкової професійної підготовки;

– у 10-11 класах здійснюється за суспільно-гуманітарним, природничо-математичним, технологічним, художньо-естетичним, спортивним напрямами [179].

Отже, в цілому профільне навчання спирається на два ключових поняття: диференціація навчання та професійне самовизначення. У взаємозв’язку цих понять нами сформульовано наступне визначення: профільне навчання – це різновид освітньої диференціації, покликаний максимально сприяти учню в його професійному самовизначенні.

### 1.2.1. Професійне самовизначення учня

В українському педагогічному словнику [286, с. 267] професійне самовизначення трактується, як процес вибору майбутньої трудової діяльності, що полягає в усвідомленні особистістю себе, як суб’єкта конкретної трудової

діяльності, і передбачає самооцінку людиною індивідуально-психологічних якостей та зіставлення своїх можливостей з психологічними вимогами професії до спеціаліста.

Професійне самовизначення, тобто вибір професії, – це одне з головних рішень в житті, один з центральних аспектів становлення особистості. Вибір професії обов'язково включає усвідомлення своїх можливостей та ролі в суспільстві. Професійне самовизначення тісно пов'язане з формуванням життєвих планів та пріоритетів.

Проблеми самовизначення особистості, включаючи професійне, вивчали цілий ряд дослідників: Ю. К. Бабанський [15], А. М. Леонтьєв [151], Д. Майерс [161], Р. С. Нємов [192], Р. С. Сазонов [247], П. А. Шавір [297], І. В. Шаповаленко [298]. Ними розглянуто різні аспекти проблеми професійного самовизначення особистості: професійна спрямованість, мотиви, потреби, інтереси, самоефективність, діяльність та інші.

Професійне самовизначення базується на професійній спрямованості особистості. Професійну спрямованість особистості визначають її змістова та динамічна характеристики. Змістова характеристика – коло мотивів. Динамічна – їх інтенсивність, тривалість, стійкість. Збагачення мотивів одна зі складових професійного самовизначення [297]. Мотиви формуються на основі потреб та інтересів. Для професійного самовизначення людина повинна узгоджувати свої здібності і можливості із зовнішніми умовами, що носять для неї характер об'єктивних і необхідних вимог [247]. Професійне самовизначення – важливий етап становлення особистості, тісно пов'язаний та обумовлений самоефективністю цієї особистості [161, с. 74]. Професійне самовизначення – це прояв вміння самостійно вирішувати складні питання [297, с. 268]. На сучасному етапі розвитку суспільства, коли практично будь-яка людська діяльність містить складову використання ІКТ, у професійному самовизначенні вбачається ще й усвідомлення особистістю значення та глибини ототожнення

своєї діяльності з використанням комп'ютера, Інтернету, інших засобів автоматизації.

Поряд з цим, в основі професійного самовизначення лежить вибір професії як вибір діяльності [15, 151, 192]. В процесі професійного самовизначення відбувається усвідомлення особистістю себе, як суб'єкта діяльності, та характеру (різновиду, змісту) діяльності, як її об'єкта. Діяльність – визначальний фактор формування і розвитку особистості, ключове поняття сучасної педагогічної теорії. Дослідники підкреслюють виключну роль діяльності у психічному розвитку особистості [53, 151, 192, 239, 244], розрізняють діяльність *зовнішню* – прояви особистості в суспільстві, та *внутрішню* – психічні процеси, що саморозвиваються. Зокрема, підкреслюється взаємозв'язок діяльності внутрішньої та зовнішньої [244, с. 680], вказується на тісне співвідношення діяльності та свідомості [151, с. 65]. Натомість, свідомість трактується, як образ, а діяльність, як процес його формування. Сама ж діяльність, як внутрішня, так і зовнішня, розглядається вченими у різних контекстах та проявах: пізнавальна – спрямована на когнітивне самовизначення, розумова – як діяльність без прямого контакту з об'єктами матеріального світу, практична – спрямована на перетворення світу у відповідності до визначених цілей. При цьому саме практична складова діяльності переважно обумовлює решту її проявів. Професійне самовизначення, як внутрішня та зовнішня діяльність, під час якої формується свідомість, обов'язково повинно містити складову практичної діяльності в ключі майбутніх життєвих планів та перспектив. Варто зазначити, що практична діяльність є провідною у навчанні інформатики в цілому та у навчанні програмування зокрема.

Будь-яка діяльність особистості обумовлена певними мотивами. «Діяльності без мотивів не буває; «невмотивована діяльність» – це діяльність не позбавлена мотиву, а діяльність на підставі суб'єктивно та об'єктивно прихованих мотивів [151, с. 54]». Мотиви не лише спонукають особистість до

діяльності, а й формуються в її процесі. Як уже зазначалося, формування мотивації – важлива складова професійного самовизначення особистості. За сучасних умов інформаційного розвитку суспільства в учнів широко проявляється мотивація до діяльності в сфері ІКТ.

Підсумовуючи та узагальнюючи розглянуті аспекти професійного самовизначення, нами зроблено такі висновки:

– професійне самовизначення базується на усвідомленні життєвих потреб, формуванні відповідних особистісних інтересів та мотивів;

– активна та вмотивована навчальна діяльність пов'язана з професійною сферою – необхідна умова професійного самовизначення особистості;

– навчання програмування вмотивоване перспективами суспільного розвитку та значною мірою сприяє професійному самовизначенню учнів.

### 1.2.2. Диференціація навчання

Як уже зазначалось, профільне навчання – це диференційоване навчання. В контексті множини індивідуальних відмінностей учнівських особистостей диференціація – необхідна умова забезпечення ефективності навчання. Диференціація (лат. differentia – відмінність) в освіті – процес та результат створення відмінностей між частинами освітньої системи (підсистем) [38, с. 271]. Підставами для створення відмінностей в освітньому процесі можуть бути стать, вік, соціальна належність, розумові здібності, успіхи в навчанні, пізнавальні інтереси тощо. Існує кілька видів диференціації навчання: за здібностями; за недостатністю здібностей; за інтересами учнів; за талантами дітей і за майбутньою професією [38, с. 271].

Диференціація, покладена в основу системи профільного навчання, переважно обумовлена майбутньою професійною орієнтацією учня та обов'язково спирається на здібності учнів, їх інтереси, таланти. Визначення змісту та обсягу навчання програмування, з одного боку, спирається на розумові здібності та пізнавальні інтереси учнів, а з іншого – відбувається у відповідності до обраного профільного спрямування.

Диференційований добір навчальної діяльності забезпечує її відповідне профільне спрямування. Отже, добір змісту навчання та обсягу навчального матеріалу є засобом профільної диференціації навчання.

### 1.2.3. Організація профільного навчання інформатики

Нормативно-правова база впровадження профільного навчання в загальноосвітніх навчальних закладах спирається на такі законодавчі документи та розпорядження:

– Національну доктрину розвитку освіти України [221];

– Закон України «Про загальну середню освіту» [106];

– Концепцію загальної середньої освіти [129];

– Концепцію профільного навчання у старшій школі [130].

– Концепція профільного навчання чітко визначає мету, спрямування, основні завдання, принципи профільного навчання.

Мета профільного навчання:

– забезпечення можливостей для рівного доступу учнівської молоді до здобуття загальноосвітньої профільної та початкової допрофесійної підготовки;

– забезпечення можливостей неперервної освіти впродовж усього життя;

– виховання особистості, здатної до самореалізації, професійного зростання й мобільності в умовах реформування сучасного суспільства [130].

Профільне навчання спрямоване на набуття старшокласниками навичок самостійної науково-практичної, дослідницько-пошукової діяльності; розвиток інтелектуальних, психічних, творчих, моральних, фізичних, соціальних якостей старшокласників; прагнення до саморозвитку та самоосвіти [130].

Досягнення таких цілей в контексті зазначеного спрямування неможливе без використання ІКТ, основою якого є навчання інформатики. Завдання, які стоять перед профільним навчанням, можна умовно поділити на ті, що безпосередньо торкаються навчання інформатики, та ті, виконанню яких в цілому сприяє оволодіння учнями ІКТ. «Основні завдання профільного навчання: створення умов для врахування й розвитку навчально-пізнавальних і

професійних інтересів, нахилів, здібностей і потреб учнів старшої школи під час їхньої загальноосвітньої підготовки; виховання любові до праці, забезпечення умов для їхнього життєвого і професійного самовизначення, формування готовності до свідомого вибору і оволодіння майбутньою професією; формування соціальної, комунікативної, інформаційної, технічної компетенції учнів на допрофесійному рівні, спрямування молоді щодо майбутньої професійної діяльності; забезпечення наступно-перспективних зв'язків між загальною середньою і професійною освітою відповідно до обраного профілю [130]».

**1.2.3.1. Структура профільного навчання**

На даному етапі основні напрями профільного навчання в 10-11 класах: суспільно-гуманітарний; природничо-математичний; технологічний; художньо-естетичний; спортивний. Профіль навчання охоплює: базові предмети; профільні предмети; курси за вибором. Базові загальноосвітні предмети становлять інваріантну складову змісту середньої освіти і є обов'язковими для всіх профілів. Зміст навчання і вимоги до підготовки старшокласників визначаються державним загальноосвітнім стандартом. Профільні загальноосвітні предмети реалізують цілі, завдання і зміст кожного конкретного профілю. Вони обов'язкові для учнів, які обрали даний профіль навчання. Вивчаються поглиблено.

Інформатика має базову та профільну складові. Інваріантна частина предмету може вивчатися за програмами двох рівнів: стандарту та академічним. Рівень стандарту – обов'язковий мінімум змісту навчальних предметів, який не передбачає подальшого їх вивчення. Академічний рівень – обсяг змісту достатній для подальшого вивчення предметів у вищих навчальних закладах – визначається для навчальних предметів, які є не профільними, проте є базовими або близькими до профільних.

### 1.2.4. Формування змісту навчального матеріалу

До змісту начального матеріалу, що вивчається в профільних класах, існує певна система вимог, обумовлених широким спектром чинників. Прикладом комплексного підходу до змістової складової навчання інформатики можуть слугувати дослідження О. М. Гончарової [59]. Розглядаючи систему формування інформатичних компетентностей студентів економічних спеціальностей, О. М. Гончарова вказує, що одна з проблем відбору змісту профільної технологічної підготовки фахівців економіки та управління в галузі інформатики як у вищих навчальних закладах, так і в старших класах середньої школи, полягає в необхідності формування в учнів єдиної технологічної картини світу, в якій відображаються уявлення, поняття і знання математики, економіки та управління з поняттями інформатики. Гончарова застерігає від диференційованого та спеціалізованого навчання інформаційних технологій, відірваного від формування єдиної технологічної картини світу [59, с. 114].

Для того, щоб применшити значення швидкого морального старіння та оновлення професійних вмінь в галузі застосування інформаційних технологій, О. М. Гончарова пропонує серед іншого формування в учнів достатньої фундаментальної підготовки та мотивації для подальшої самоосвіти, формулює принцип підсилення фундаментальної базової теоретичної складової змісту навчального матеріалу. Крім усього, у доборі змісту професійного навчання Гончарова пропонує спиратися на модель діяльності спеціаліста [59, с. 118].

Модель діяльності спеціаліста галузі ІКТ поряд з її фундаментальними основами проектуються на чинники, що визначають зміст навчання програмування в курсі інформатики:

– мова та парадигма програмування, що вивчаються;

– середовища програмування, що для цього використовуються;

– послідовність вивчення тем та підтем курсу;

– зміст навчальних завдань та прикладів їх розв’язання

Зміст навчання програмування у класах різного профільного спрямування може бути надзвичайно різноплановим. Передумовами для цього є:

– величезний діапазон підходів та парадигм;

– наявність різноманітних середовищ розробки: від рядкових до візуальних та автоматизованих;

– різноманітний характер задач, що можуть розв’язуватися;

– величезна варіативність у доборі змісту навчання в межах виділеного навчального часу.

Нині зміст навчання програмування визначається у відповідності до державних навчальних програм, практично кожна з яких надає вчителю надзвичайно широкі повноваження щодо самостійного визначення змісту навчання в межах теми та на конкретному уроці. За таких умов при доборі змісту навчання вчитель повинен враховувати не лише його доступність та психофізіологічні особливості учнів, а й профільне спрямування, яке цей матеріал повинен забезпечити. Ще одним фактором, який повинен враховувати вчитель під час добору навчального матеріалу, є необхідність реалізації міжпредметних та внутріпредметних зв’язків.

### 1.2.5. Міжпредметні та внутріпредметні зв’язки

Міжпредметні зв’язки – взаємне узгодження навчальних програм, зумовлене системою наук і дидактичною метою. Міжпредметні зв’язки відображають комплексний підхід до виховання й навчання, який дає можливість виділити як головні елементи змісту освіти, так і взаємозв’язки між навчальними предметами [291, с. 210].

Інформатика є наукою багатогалузевою, такою, що розглядає застосування ІКТ не лише у різних підходах (практично-користувацькому, програмістському, теоретичному), а й у різних напрямках та сферах (робота з текстами, зображеннями, звуками, використання у навчанні, у виробництві інше). Міжпредметні зв’язки інформатики з іншими дисциплінами тісно поєднуються та переплітаються зі зв’язками внутріпредметними. Пошук та використання під час навчання інформатики внутріпредметних зв’язків чи не єдиний шлях забезпечення цілісності навчальної дисципліни «інформатика».

Функції міжпредметних та внутріпредметних зв’язків: виховна, розвиваюча, детермінуюча – реалізовуються завдяки інтеграції знань, що підвищує продуктивність перебігу психічних процесів. Такі зв’язки формують конкретні знання учнів, включають їх в оперування пізнавальними методами: абстрагування, моделювання, аналогія, уявлення, інші.

Смислові зв'язки взагалі та їх різновидність міжпредметні зв’язки, забезпечують процес навчання, відіграють важливу роль у формуванні системного мислення учнів. Основою принципу системності в діяльності головного мозку є відомі результати дослідження фізіології нервової системи людини І. П. Павлова. Павлов обумовлював мислення, запам'ятовування та інші прояви вищої нервової діяльності утворенням тимчасових зв'язків у корі головного мезгу [206, с. 509]. Ці висновки склали основу сучасних педагогічних теорій навчання та розвитку особистості.

Одним із підходів до формування міжпредметних зв’язків інформатики з іншими дисциплінами є використання навчального програмного забезпечення в процесі навчання цих дисциплін. Зокрема Ю. В. Горошко розглядає використання, з метою формування між предметних зв’язків інформатики з математикою та фізикою, педагогічно-програмного засобу «Gran1» [61]. Сформована міжпредметна пов’язаність стане ще глибшою, якщо в процесі навчання програмування проаналізувати об’єктну модель та принципи функціонування того ж таки педагогічного програмного засобу «Gran».

Формування міжпредметних зв’язків відбувається також під час узагальнення навчального матеріалу. Узагальнення тісно пов’язане з систематизацією здобутих знань. Спираючись на принципи системності, Н. Ф. Тализіна зазначає, що узагальнення повинно відбуватися на основі системного підходу до досліджуваної області: «за тими властивостям предметів, які увійшли до складу орієнтовної основи дій, спрямованих на аналіз цих предметів». [**278,** с. 77].

Міжпредметні зв’язки лежать в основі профільного навчання інформатики. «Профіль навчання – це спосіб організації диференційованого навчання, який передбачає поглиблене і професійно зорієнтоване вивчення циклу споріднених предметів [130]». Поряд із цим інформатика забезпечує процес пізнання найсучаснішими засобами та технологіями, готує учнів до застосування ІКТ у різних сферах. На відміну від інших навчальних дисциплін, навчання інформатики, як ніяке інше, базується на підтримці практично всіх сфер людської діяльності. Сучасні телекомунікаційні мережні системи дозволяють автоматично збирати, класифікувати і структурувати інформацію з різних галузей людських знань, тим самим стираючи міжгалузеві межі. Реалізації міжпредметних зв’язків у процесі профільного навчання інформатики сприяють проектні та інші передові педагогічні технології.

Базове, фундаментальне значення програмування в курсі інформатики також потребує здійснення його навчання у взаємозв’язку з іншими розділами та дисциплінами.

В процесі нашого дослідження розглядалися такі шляхи реалізації міжпредметних зв’язків під час навчання програмування:

– узгоджене вивчення спільних та тотожних тем навчального матеріалу;

– розв’язання задач суміжного навчального предмету, інших тем інформатики;

– добір ілюстративних та пояснюючих матеріалів за матеріалами змісту навчання суміжних навчальних дисциплін;

– використання умов навчання, тотожних умовам навчального предмету, з яким зв’язок реалізовується.

Велике значення для реалізації міжпредметних зв’язків під час навчання програмування відіграє парадигма мови програмування, на основі якої навчання здійснюється. Навчання на основі мов імперативного програмування тісно поєднує інформатику з предметами фізико-математичного циклу та може супроводжуватись виконанням завдань най різноманітнішого змістового наповнення. Поряд із цим, навчання програмування на основі об’єктно-орієнтованого підходу дозволяє, крім усього, налагодити більш глибокі зв’язки задля систематизації та узагальнення навчального матеріалу різних розділів інформатики, інших шкільних дисциплін.

### 1.2.6. Програми для профільного навчання інформатики до 2008 року

У 2003 році було опубліковано єдиний збірник навчальних програм з інформатики, що передбачали профільну диференціацію, її вивчення в ЗНЗ [116]. До цього часу навчання інформатики в профільних класах було практично не регламентоване. Збірник «Інформатика. Програми для загальноосвітніх навчальних закладів» за редакцією М. І. Жалдака містить окремі програми для загальноосвітніх навчальних закладів 10 – 11 класів наступних профілів:

– універсального [116, с. 3];

– універсального, без використання комп’ютерів [116, с. 16];

– філологічного та суспільно-гуманітарного [116, с. 29];

– художньо-естетичного [116, с. 41];

– спортивного [116, с. 53];

– фізико-математичного, природничого та технологічного [116, с. 65].

Укладачами наведених програм є М. І. Жалдак, Н. В. Морзе, О. І. Мостіпанта, Г. Г. Науменко.

Окрім оцих основних програм, збірник включає досить широкий набір інших навчальних програм з інформатики для використання за умов профільного навчання:

1. Програми для загальноосвітніх навчальних закладів технологічного профілю інформатика та технології (інтегрований курс). 7 – 11 класи. Укладачі: Т. І. Астістова, С. М. Дзюба [116, с. 85]. До складу даного курсу, окрім основної змістової лінії, входять програми окремих курсів:

– Технології. 7 – 9 класи;

– Інформатика та технології. Основи веб-дизайну. 10 – 11 класи;

– Інформатика та технології. Об'єктно-орієнтовані мови програмування, використання табличного процесора у сфері економіки та бізнесу. 10 – 11 класи.

– Інформатика та технології. Архітектурне та ландшафтне проектування. 10 – 11 класи.

2. Програми для загальноосвітніх навчальних закладів, спеціалізованих шкіл, гімназій, ліцеїв. Інформатика (поглиблений курс). 8 –11 класи. Укладачі: М. І. Жалдак, Н. В. Морзе, О. І. Мостіпан.

3. Програми для загальноосвітніх навчальних закладів. Інформатика. 7 – 9 класи. Укладачі: М. І. Жалдак, Н. В. Морзе, Г. Г. Науменко.

4. Програми для спеціалізованих шкіл, гімназій, ліцеїв. Інформатика і програмування. 8 – 11 класи. Укладачі: Н. В. Голубнича, Т. П. Караванова, В. П. Костюков.

Друга частина збірника [116, с. 245] містить програми спецкурсів, факультативів, пропедевтичних курсів, гуртків. Автори-укладачі програм: Н. В. Морзе, О. І. Мостіпан, Т. І. Лисенко, Л. А. Чернікова, І. Є. Коровець, О. Б. Рудик.

### 1.2.7. Програми профільного навчання інформатики 2009 - 2012років.

У 2008 році було опубліковано нові програми з інформатики для 9-12 класів загальноосвітніх навчальних закладів (автори І. О. Завадський,

Ж. В. Потапова, Ю. О. Дорошенко) [99; 100]. Навчання інформатики передбачалося за двома рівнями «академічним рівнем» та «рівнем стандарту». У 2009 році було запроваджено обов'язкове навчання інформатики у 9 класах загальноосвітніх навчальних закладів. Перехід на нові навчальні плани відбувався у відповідності до переходу до 12-річної загальної середньої освіти [75, 129]. Поряд з тим, у той же період, була запроваджена навчальна програма авторів Т. П. Караванової та В. П. Костюкова з поглибленого вивчення інформатики для учнів 8-12 класів ЗНЗ (напрям: технологічний, профіль: інформаційно-технологічний).

У зв'язку з політичними змінами в України та прийняттям рішення про збереження 11- річного терміну навчання у ЗНЗ у 2010 році було проведено перегляд навчальних програм та внесено до них відповідні зміни. У тому ж таки році було опубліковано навчальні програми з інформатики для 9-11 класів академічного рівня та рівня стандарту авторів: І. О. Завадського, Ж. В. Потапової, Ю. О. Дорошенка [97; 98], що фактично є зміненими та доповненими редакціями програм для 9-12 класів цих же авторів [99; 100]. Також було дещо змінено організацію профільного навчання інформатики. Навчання інформатики в класах усіх профілів, окрім інформаційно-технологічного, базувалося на програмах академічного рівня та рівня стандарту.

Програми академічного рівня розроблені з розрахунку вивчення у 10 класі протягом однієї навчальної години на тиждень, а у 11 класі – 2 навчальних годин. Програми рівня стандарту передбачають на вивчення інформатики по одній навчальній годині в обох класах. Відмінними у програмах академічного рівня та рівня стандарту у 10 класах є підрозділи «Обробка мультимедійних даних», «Основи створення комп'ютерних публікацій» та «Системи обробки табличної інформації». «Обробка мультимедійних даних» та «Основи створення комп'ютерних публікацій» вивчаються тільки за програмою рівня стандарту. Тема «Системи обробки

табличної інформації» наявна тільки у програмі академічного рівня. Відмінними у програмах академічного рівня та рівня стандарту в 11 класах є підрозділи «Моделювання. Основи алгоритмізації» та «Основи алгоритмізації та програмування». Тема «Моделювання. Основи алгоритмізації» (5 годин) вивчається тільки за програмою рівня стандарту. А «Основи алгоритмізації та програмування» (28 годин) тільки у програмі академічного рівня.

Навчання інформатики у класах інформаційно-технологічного пропонується за окремою навчальною програмою поглибленого вивчення інформатики для учнів 10-11 класів загальноосвітніх навчальних закладів [223, с. 25]. «Характерною особливістю структури даної навчальної програми є те, що вона складається з двох паралельних змістовних ліній: сучасних інформаційно-комунікаційних технологій (ІКТ) та основ алгоритмізації та програмування (ОАП). Обидві ці лінії тематично взаємопов’язані і послідовно узгоджені [223, с. 26]».

### 1.2.8. Форми організації профільного навчання

Форми організації профільного навчання поділяються на внутрішньошкільні та зовнішні. До внутрішньошкільних належать: профільні класи в загальноосвітніх навчальних закладах; профільні групи в багатопрофільних загальноосвітніх навчальних закладах; профільне навчання за індивідуальними навчальними планами і програмами; динамічні профільні групи (в тому числі різновікові). Зовнішні форми це міжшкільні профільні групи; профільна школа інтернатного типу; опорна старша школа; навчально-виховний комплекс (НВК); міжшкільний навчально-виробничий комбінат (МНВК); загальноосвітні навчальні заклади на базі вищих навчальних закладів.

Стосовно інформатики, то на теперішньому етапі недостатньо використані напрямки та можливості реалізації профільного навчання. Поки що активно використовуються не всі форми організації профільного навчання.

### 1.2.8.1. Курси за вибором

Однією з форм профільного навчання є курси за вибором. Починаючи з 2010 року, курси за вибором є основним засобом відповідного профільного спрямування в класах усіх профілів, окрім інформаційно-технологічного. Вже в опублікованому у 2003 році збірнику програм з інформатики під редакцією М. І. Жалдака [116], що містить програми для профільного навчання інформатики у 10-11 класах, опубліковано окремі програми спецкурсів, факультативів, пропедевтичних курсів, гуртків:

– Основи інформаційних технологій. 7–11 класи. Укладачі: Н. В. Морзе, О. І. Мостіпан.

– Основи програмування. 10–11 класи. Укладач: Т. І. Лисенко

– Курс користувача. 7–9 класи. Укладачі: Н. В. Морзе, О. І. Мостіпан.

– Формальна логіка. 8–11 класи. Укладач: Л. А. Чернікова.

– Мова розмітки гіпертексту HTML. 10–11 класи. Укладач: Т. І. Лисенко

– Програмування Інтернет-орієнтованої графіки. 10 клас. Укладач: І. Є. Коровець

– Школа олімпійського резерву з програмування. 9–11 класи . Укладач: Т. І. Лисенко

– Інформаційна культура. 10–11 класи. Укладач: Н. С. Прокопенко

– Вступ до інформатики. 5–6 класи. Укладачі: Н. В. Морзе, О. І. Мостіпан.

– Прикладна математика. 8–11 класи. Укладач: О. Б. Рудик.

Ці програми фактично започаткували навчання інформатики в курсах за вибором. Проте, починаючи з 2004 року, на сторінках методичних видань почали з’являтися спеціально розроблені програми курсів за вибором для профільного навчання інформатики в ЗНЗ. Ось перелік назв деяких курсів за вибором, програми яких було опубліковано до 2011 року:

– Основи комп’ютерної графіки [74].

– Сучасні офісні інформаційні технології.

– Основи комп’ютерної безпеки.

– Основи Інтернету [73].

– Основи створення комп’ютерних презентацій [104].

– Інформаційний працівник.

– MS Excel у профільному навчанні.

– Основи веб-дизайну [102].

– Інформаційні технології проектування.

– Основи візуального програмування [103].

Курси за вибором тісно інтегруються з інваріантною складовою програми. Майже всі розділи програм рівня стандарту та академічного рівня можна розширювати шляхом викладання їх за програмами курсів за вибором.

**1.2.8.2. Дистанційне навчання як засіб профілізації**

Дистанційне навчання засобами сучасних ІКТ має цілий ряд переваг, що зумовлюють його активне використання в системі традиційної освіти. Це, в першу чергу, додатковий навчальний простір, що ефективно доповнює класно-урочну систему навчання. Також дистанційне навчання, як одна з новітніх та перспективних організаційних форм навчання програмування, має значний потенціал для забезпечення профільної диференціації. Зокрема, ресурси Інтернету присвячені дистанційному навчанню програмування мовою C#, можна широко використовувати для забезпечення профільного спрямування навчання інформатики в загальноосвітніх навчальних закладах [332].

**1.2.8.3. Допрофільна підготовка**

Важливим етапом здійснення профільного навчання є підготовка до такого навчання учнів середнього шкільного віку ще на етапах навчання, коли профільна диференціація не здійснюється. Нині широко поширені такі форми реалізації допрофільної підготовки, як введення курсів за вибором та поглиблене вивчення окремих предметів на диференційованій основі.

Основна функція курсів за вибором – профорієнтаційна. Для вивчення курсів за вибором професійного спрямування базового курсу інформатики у 7-9

класах та пропедевтичного курсу інформатики у 2-8 класах можуть використовуватися години варіативної частини робочого навчального плану та навчально-виробничої практики.

Поглиблене вивчення предмета має сприяти формуванню стійкого інтересу до предмета, розвитку відповідних здібностей, орієнтації на професійну діяльність, де використовуються одержані знання.

Допрофільна підготовка може здійснюватися також через факультативи, предметні гуртки, наукові товариства учнів, Малу академію наук, предметні олімпіади, кабінети профорієнтації.

### 1.2.9. Значення профільної організації навчання програмування

Навчання інформатики має велике значення у підготовці учнів до продовження освіти та професійного самовизначення [83]. З усіх розділів інформатики саме програмування дає учням найбільш повне, глибоке та правдиве уявлення про ті особливості сучасних інформаційних технологій, що залишаючись прихованими для більшості користувачів, фактично визначають принципи функціонування сучасної комп'ютерної техніки. Висока популярність ІКТ приваблює багатьох молодих людей оволодіти професією програміста. Іноді в школярів складається враження легкодоступності та простоти розробки програмного забезпечення. Упереджену картину професії програміста створюють засоби масової інформації та суспільна думка. Лише навчання програмування зорієнтоване на підготовку до професійної діяльності, надає можливість учню належним чином усвідомити сутність діяльності з розробки програмного забезпечення.

Фактично, навчання програмування використовувалось для профільної диференціації ще з перших років навчання інформатики в школі. Хоча термін «профільне навчання» у вісімдесяті роки минулого століття не був поширеним, численні гуртки та факультативи відігравали значну роль в професійному визначенні учнів. Зокрема, серед перших вітчизняних посібників з інформатики, що були надруковані українською мовою, був посібник

І. Ф. Следзінського та М. Я. Лященка для шкільних факультативів з інформатики «Програмування на ЕКОМ. Посібник для факультативних занять у 9 класі [159]».

### 1.3. Програмно-технологічні засади використання мови C# для навчання програмування в загальноосвітніх навчальних закладах

Мова, що надає можливість створювати програми на основі парадигми ООП, повинна певною мірою відповідати особливостям цієї парадигми. Загалом, можна виділити два різновиди мов, якими можна розробляти ОО програми. В першу чергу, це ті, що утворилися після певного розширення можливостей процедурної мови програмування. Сюди варто віднести мови Object Pascal, Delphi, C++. Крім того, існує багато мов програмування, що від моменту своєї появи передбачали написання ОО програм: Java, C# та інші. Швидше за все, написання ОО програм такими мовами більш ефективне та раціональне. Щоб з’ясувати наскільки та чи інша мова адаптована до особливостей ООП, варто розглянути самі ці особливості.

Однією з характерних рис об’єктно-орієнтованого програмування є широко використовуваний спосіб задавання завдання. Завдання в об'єктно-орієнтованому програмуванні ініціюється за допомогою передачі повідомлень агенту (об'єкту), що є відповідальним за завдання. Таке повідомлення – це вказівка на виконання завдання, що супроводжується додатковими даними (аргументами), необхідними для його виконання. Одержувач (receiver) – це агент, якому надсилається повідомлення. Якщо він приймає повідомлення, то на нього автоматично покладається відповідальність за виконання зазначеної дії. Як реакція на повідомлення одержувач запустить деякий метод, щоб задовольнити отриману вказівку.

Спочатку клієнт, якщо у нього є завдання, повинен знайти, кому можна було б його доручити. Це також неявний принцип. Якщо програміст довгий час розробляє код традиційними методами, то цілком нормальне вміння «перекласти свої завдання на чужі плечі» в нього зникає. Практично, будь-яка

мова програмування надає можливість йому самому розробити кожен окремий елемент програми від самого початку, не звертаючись до послуг ще когось. Розробка повторно використовуваних компонентів – важлива частина об'єктно-орієнтованого програмування.

В ООП діє принцип маскування даних щодо пересилання повідомлень, згідно якого клієнтові не потрібно знати про засоби, за допомогою яких його запит буде виконано. Якщо розглядати на рівні комп'ютерів і програм, то різниця між викликом процедури і пересиланням повідомлення полягає в тому, що в останньому випадку існує певний одержувач та інтерпретація (тобто вибір відповідного методу, який запускається у відповідь на повідомлення може бути різним для різних одержувачів). Зазвичай, конкретний одержувач невідомий аж до виконання програми, так, що визначити, який метод буде викликаний, заздалегідь неможливо. У такому випадку говорять, що має місце пізніше зв'язування між повідомленням (ім'ям процедури або функції) і фрагментом коду (методом). Ця ситуація протиставляється ранньому зв'язуванню (на етапі компіляції або компонування програми) імені з фрагментом коду, що відбувається при традиційних викликах процедур.

Ще одним важливим нюансом, фундаментальною концепцією ООП є відповідальність за виконання завдання. В ООП зростає рівень абстракції, незалежність агентів, а це, певною мірою, критичний фактор у складних завданнях [16, с. 11].

Об'єкти не можуть постійно реагувати на завдання тільки тим, що звертаються до інших із вказівкою виконати деяку дію. Це призведе до нескінченного циклу запитів. Якщо не всі, то хоча б деякі об'єкти повинні виконувати певну роботу, перш ніж звертатися з вказівкою до інших. Така функція по-різному реалізована в різних об'єктно-орієнтованих мовах програмування, але вона є обов’язковою в ООП.

Загалом, як уже зазначалося вище, називають три ознаки ООП: інкапсуляцію, поліморфізм, спадкування.

### 1.3.1. Мова програмування C# та платформа Microsoft .Net Framework

Мова програмування C# – складова частина загальної стратегії Microsoft .NET Framework. Термін ".NET" (читається "дот-нет") – це назва програмної технології, запропонованої фірмою Microsoft як платформа для створення різного роду комп'ютерних програм, у тому числі програм для Інтернету. .NET – одне з найбільш фундаментальних, значимих нововведень в галузі програмування за останні десятиліття (перша бета-версія розроблена в 2000 році).

Microsoft почала розробляти .NET Framework у кінці 1990-х під назвою Next Generation Windows Services (NGWS). Пізніше в 2000 році була випущена перша бета-версія .NET 1.0 [325; 326].

Офіційний сайт центру розробки Microsoft .NET Framework дає їй таке визначення: «.NET Framework – це модель програмування з керованим кодом від Microsoft, націлена на створення додатків для клієнтів Windows, серверів, а також мобільних та вбудованих пристроїв. Розробники можуть використовувати .NET для побудови різних типів додатків, таких як веб-додатки, серверні додатки, додатки інтелектуальних клієнтів, консольні додатки, додатки баз даних та багато інших [326]».

За іншими джерелами [338; 353] .NET – програмна технологія, яка багато в чому є продовженням ідей та принципів, покладених в основу технології Java, що на початку нового тисячоліття стала лідером серед засобів розробки програмного забезпечення.

Середовище розробки .NET, подібно до технології Java, теж створює байт-код, призначений для виконання віртуальною машиною [7, с. 25]. Вхідна мова цієї машини в .NET називається MSIL (Microsoft Intermediate Language – проміжна мова Майкрософт). Проте частіше використовується пізніша версія назви проміжної мови – CIL (Common Intermediate Language – загальноприйнята проміжна мова) або просто IL (проміжна мова) [266]. CIL найчастіше перекладають (тлумачать) українською мовою, як «проміжна

машинна мова». Сама ж віртуальна машина, в якій виконується CIL, носить назву .NET Runtime. .NET Runtime у перекладі означає «середовище виконання» чи, точніше, «середовище виконання в реальному часі».

Як уже зазначалося, .NET Runtime є аналогом віртуальної машини Java. Однією з переваг платформи .NET над Java є те, що для Java існує лише одна мова розробки комп'ютерних програм. Для .NET Runtime таких мов може бути багато. Середовище розробки MS Visual Studio підтримує такі мови, як Visual Basic, Visual C ++, C#. Існують .NET реалізації мов «Ada» (.A#), «Apl», Boo (заснований на Python), COBOL, Component Pascal, Delphi, Eiffel, F# (член сімейства мов програмування ML), Forth, FORTRAN, IKVM, Java, IronPython (реалізація мови Python), Lexico, Lisp, Mercury, Mondrian, Nemerle (гібридна функціонально/імперативна мова), Oberon/Zonnon, Perl, RPG, Smalltalk та інші. Кількість мов, для яких вже вдалося забезпечити підтримку .NET Runtime, постійно зростає [335].

Так само як і байт-код для віртуальної java-машини, програма для середовища .NET Runtime може виконуватися будь-якою операційною системою, в якій .NET Runtime встановлена. Кількість операційних систем, де може виконуватись .NET Runtime, теж постійно зростає, хоча найбільш повну підтримку проміжної машинної мови CIL реалізовано в операційних системах сімейства Windows починаючи з Windows XP.

Основні особливості сучасної платформи .NET Framework:

– кросплатформність – незалежність створюваного для роботи в межах платформи байт-коду від операційної системи та обладнання;

– гнучка система безпеки зумовлена кросплатформністю та існуванням проміжного середовища виконання;

– велика об'єктно-орієнтована бібліотека підпрограм (набір базових класів).

Поряд із цим для .NET Framework можна додати до переліку важливих її переваг підтримку та об'єднання в одному середовищі розроблення значної

кількості широко розповсюджених мов програмування. Така підтримка надає можливість розробниками, які програмували різними мовами програмування до використання .NET, і надалі застосовувати набуті знання, досвід, навики, раніше написаний програмний код.

Підтвердженням загальної перспективності платформи .NET Framework є те, що вона продовжує активно розвиватись та удосконалюватись [326].

Ще у вересні 2008 року Microsoft анонсувала .NET Framework 4.0. У квітні 2010 року відбувся офіційний реліз платформи та середовища розробки, що її підтримує – Visual Studio 2010. Станом на кінець 2012 року останніми офіційними версіями платформи .NET та середовища Visual Studio були .NET Framework 4.5 та Microsoft Visual Studio 2012, дата виходу яких 15 серпня 2012.

Існує значна кількість професійного програмного забезпечення, яке розроблене і функціонує на платформі Microsoft .NET Framework. Для ілюстрації можливостей таких програм можна використати графічний редактор Paint.NET. Ця програма постійно оновлюється, має хорошу підтримку в глобальній мережі Інтернет [349; 351]. Ось основні переваги Paint.Net:

– програма безкоштовна для розповсюдження та використання;

– оптимізована для роботи з двоядерними та чотириядерними процесорами;

– має зручний інтерфейс, схожий на Photoshop;

– може працювати з декількома документами одночасно;

– підтримує роботу з шарами;

– це простий у використанні графічний редактор [350].

Paint.Net зручний для використання в навчальних цілях завдяки тому, що надає можливість створювати як найпростіші малюнки, так і виконувати складну обробку зображень. Оскільки майже в усіх шкільних програмах з інформатики для загальноосвітніх навчальних закладів є теми, де вивчається комп’ютерна графіка та програми для роботи з нею, то використання Paint.Net у школі є вельми доцільним.

Цікавим у пізнавальному плані є міжнародний проект «Mono». Це проект повноцінного втілення системи .NET на базі вільного програмного забезпечення [343]. У 2010 році популярності набула одна із стабільних версій програми Mono Develop 2.2, яка вже мала реалізації, здатні працювати під управлінням операційних систем Windows, різних версій Linux, Mac OSX [345]. У жовтні 2012 року було випущено Mono 3.0, що забезпечує реалізацію в середовищі вільного програмного забезпечення останніх на той час нововведень платформи .NET Framework 4.0. Завдяки Mono той же Paint.NET можна використовувати на GNU/Linux та інших операційних системах, для яких реалізується Mono [349]. Хоча ідея та авторські права .NET належать Microsoft, Mono практично повністю наслідує .NET Framework.

Варто чітко визначити, які відомості про новітню платформу Microsoft .NET слід долучити до навчального матеріалу шкільного курсу інформатики. Нині діюча Концепція загальної середньої освіти зазначає такі вимоги до змісту освіти: «Зміст (загальної середньої освіти) визначається на засадах його фундаменталізації, науковості і системності знань, їх цінності для соціального становлення людини, гуманізації і демократизації шкільної освіти, ідей полікультурності, взаємоповаги між націями і народами, світського характеру школи [129, ст.4]». В тому ж таки документі зазначається: «За останні роки значно зросло навчальне навантаження учнів, зумовлене невідповідністю змісту освіти, навчальних технологій їхнім віковим психофізіологічним особливостям [129, ст.1]». Отже, будь-яке наповнення навчального матеріалу новими відомостями повинно відбуватися у строгій відповідності до психофізичних особливостей дітей певного віку та слугувати не лише підвищенню його науковості, системності, а й враховувати соціальну та загальнокультурну його цінність.

В межах даного дослідження нами сформульовано загальні рекомендації щодо вивчення в загальноосвітніх навчальних закладах відомостей про Microsoft .NET Framework та її аналоги (Mono, інші):

– доповнити навчальний матеріал загальними відомостями про платформу .NET, історію її розроблення та розвитку, перевагами стосовно розробки програмного забезпечення та перспективами її подальшого розвитку;

– знайомити учнів з перевагами Microsoft .NET можна під час вивчення та використання в навчальному процесі прикладного програмного забезпечення. Одним з прикладів такого програмного забезпечення є графічний редактор Paint.NET;

– за умови поглибленого вивчення інформатики у старших класах загальноосвітніх навчальних закладів доцільно включити до навчального матеріалу основні відомості про структуру віртуальної машини .NET Runtime;

– розповідати учням про .NET під час навчання програмування. Особливо актуальними є відомості про цю програмну платформу, якщо програмуванню навчають з використанням мов: Visual Basic, C++, C# у середовищі Microsoft Visual Studio.

Наведені вище та інші результати дослідження: «Платформа Microsoft .NET у змісті навчального матеріалу курсу інформатики середньої загальноосвітньої школи» опубліковано нами в електронному фаховому виданні «Інформаційні технології і засоби навчання» [314].

### 1.3.2. Передумови використання мови C# для навчання програмування

Як уже зазначалося, мова програмування C# – складова частина загальної стратегії Microsoft .NET Framework. Компанія Microsoft розробила мову C# у кінці 1990-х років. Автором мови є Андерс Хейльсберг [7]. У середині 2000 року була випущена альфа-версія мови. Хейльсберг, так само як автори мов C + + і Java, не створював нових наборів команд та правил побудови синтаксису, а, використавши як фундамент наявні мови, зосередився на покращеннях та інноваціях. C# надзвичайно схожа на широко вживані та популярні мови програмування C, C++, Java та PHP. Сьогодні практично всі професійні програмісти знають ці мови, тому перехід до C# відбувається без особливих труднощів [258]. Поряд із цим знання, набуті при оволодінні мовою C#, безумовно знадобляться бажаючим освоїти інші мови програмування.

Впровадження новітніх апаратно-програмних засобів навчання спирається на відповідні матеріальні умови та фахову підготовку педагогів, лаборантів [221]. Використання в школах для навчання програмування мови C# можливе лише за наявності відповідного програмно-технологічного забезпечення. Важливо з'ясувати, яке ж саме апаратне, програмне, фахове та інше забезпечення потрібне школам для повноцінного використання мови C# у навчанні програмування.

Для успішного використання мови C# під час навчання програмування в загальноосвітніх навчальних закладах необхідно врахувати такі умови:

– наявні та доступні середовища програмування, які доцільно використати при навчанні програмування мовою C#;

– операційні системи та додаткове програмне забезпечення, яке необхідне для повноцінного функціонування середовищ програмування орієнтованих на вивчення мови C#;

– апаратні вимоги, яким повинна відповідати техніка для повноцінної підтримки операційних систем, середовищ програмування та іншого програмного забезпечення, що може використовуватись для вивчення мови C#;

– рівень компетентності вчителів інформатики, необхідний для навчання програмування мовою C#.

Ці передумови сильно взаємопов'язані та взаємообумовлені. Середовища для програмування мовою C# працюють під управлінням тих чи інших операційних систем. Використання операційних системи, як і середовищ програмування, потребує від комп'ютерної техніки певних апаратних можливостей. І нарешті, для використання техніки в навчально-виховному процесі вчителі інформатики та лаборанти комп'ютерних класів повинні мати знання та компетентності, що стосуються комп'ютерного обладнання та програмного забезпечення.

### 1.3.3. Середовища програмування мовою C#

Нині найбільш поширеними є такі середовища розробки мовою C#:

– Microsoft Visual Studio Professional – найбільш повний та надзвичайно багатофункціональний пакет середовищ програмування для професійного використання. Існує можливість безкоштовного 90-денного використання цього середовища, як пробної версії та з навчально-ознайомлювальною метою;

– Microsoft Visual C# Express Edition безкоштовна, практично повнофункціональна версія, що відрізняється від професійної лише незначним обмеженням функцій та неповними бібліотеками класів;

– Sharp Develop – середовище розробки з відкритим кодом мовою C#.

– Mono Develop – кросплатформне середовище програмування для «Mono». Як уже зазначалось нами, Mono – аналог платформи .Net Framework для операційних систем відмінних від Windows, зокрема Linux;

Дуже багато фірм, що спеціалізуються на розробці програмного забезпечення, пропонують власні середовища програмування мовою C#. Наприклад, «Borland C# Editor» – середовище розробки від фірми Borland, яке, хоча нині вже не розповсюджується і не підтримується розробником, все ж є прикладом широкого інтересу відомих компаній до мови програмування C#.

Цікавим є в навчальному плані середовище Antechinus C# Editor. Ця програма використовує встановлений в системі компілятор платформи .NET. Antechinus C# Editor підтримує усі версії .Net Framework і здатна працювати навіть під управлінням дещо застарілих, проте дуже невибагливих до ресурсів операційних систем типу Windows 9x [327].

Для всіх операційних систем сімейства Windows, починаючи від Windows 98, за умови, що ця система підтримує Microsoft .Net Framework, існує можливість розробляти програми з використанням будь-якого, навіть найпростішого, текстового редактора. Програми, написані, наприклад, редактором «Блокнот», можна компілювати безпосередньо засобами .Net. Варто зазначити, що існує значна кількість текстових редакторів,

оптимізованих для написання комп'ютерних програм. Для цього вони мають так звану «підсвітку синтаксису», автоматичне завершення написання зарезервованих (службових) слів та інші спеціальні функції. Одним з досить поширених редакторів такого типу є Notepad++.

Для використання всіх названих середовищ програмування можна застосовувати операційні системи Windows XP, Windows Vista, Windows 7. Проте, завдяки активному розвитку проекту «Mono» та підтримки ним середовища програмування мовою C# – Mono Develop, розробка програм цією мовою доступна нині з використанням практично всіх сучасних операційних систем. Окрім операційної системи, додаткової програмної платформи (.Net або Mono) та середовища програмування, процес розробки мовою C# не потребує якогось додаткового програмного забезпечення [343].

Поряд із цим, додаткове програмне забезпечення може розроблятися та використовуватися з метою підтримки навчання програмування мовою C#. Прикладом програмного продукту, створеного для допомоги вивчення мови C#, може бути розроблене фірмою Microsoft доповнення до пакету Visual Studio – Microsoft Visual Studio Learning Pack, раніше відомий як Visual Studio Middle School Power Toy. Visual Studio Learning Pack 2.0 – це програмний пакет, створений для школярів і студентів, як допомога у вивченні програмування [343].

Технічні можливості використання мови C# визначаються, здебільшого, саме вимогами до обчислювальної техніки – апаратними вимогами. Як уже зазначалося, практично всі середовища програмування належно працюють під управлінням операційної системи Windows XP, проте для кожного середовища розробники вказують особливі вимоги до підтримуваного комп'ютерного обладнання. Найвищими є системні вимоги середовища розробки Microsoft Visual Studio Professional. Для версії Visual Studio 2008 вони такі:

– Операційна система Windows XP, Windows Server 2003, Windows 7.

– Процесор з тактовою частотою 1,6 ГГц чи більше.

– Оперативна пам’ять не менше 384 Мбайт.

– 2,2 Гбайт доступного місця на жорсткому диску.

– Екран з роздільною здатністю не нижче 1024 х 768 [347].

Більшість сучасних комп'ютерних класів, якими в останні роки забезпечуються українські школи, відповідають таким вимогам. Проте для навчання програмування мовою C# на комп’ютерах, що постачалися в школи до 2005 року, варто використовувати середовище Sharp Develop. Воно ще менш вибагливе до ресурсів і відносно нестара та стабільна його версія – Sharp Develop 3.2 працює на обладнанні, яке відповідає практично мінімальним вимогам операційної системи Windows XP:

– процесор з тактовою частотою 300 Мгц чи більше;

– оперативна пам’ять не менше 128 Мбайт;

– доступного місця на жорсткому диску 100 Мбайт [341].

У школах України незначна кількість комп’ютерних класів, які все ще працюють під управлінням операційних систем Windows 98 чи Windows Me і для яких вимоги середовища Sharp Develop 3.2 теж були б високими. Проте і за таких умов є можливість знайомити учнів з мовою C#, використовуючи платформу Microsoft.NET версії 1.0.3705. та середовища, які її підтримують: Sharp Develop v1.1 або Antechinus C# Editor. Функцій найпростіших варіантів середовищ розробки мовою C# більш ніж достатньо, навіть для поглибленого вивчення цієї мови на уроках інформатики загальноосвітніх шкіл.

### 1.3.4. Рівень підготовки вчителів інформатики

Хоча наявність спеціалістів не належить до програмно-технологічних умов тих чи інших впроваджень – цей фактор, у даному випадку, є дуже важливим. Для повноцінного використання наявних програмно-технологічних можливостей впровадження мови C# для навчання програмування в середніх загальноосвітніх навчальних закладах необхідні висококваліфіковані педагогічні кадри. Проблема полягає в тому, що більшість учителів інформатики не лише не знайомі з програмуванням на основі використання мов

з «С-подібним» синтаксисом, а й взагалі, недостатньо обізнані з сучасними апаратно програмними платформами типу .Net та Java. Проблема підготовки вчителів стоїть особливо гостро ще й тому, що переважна більшість навчає інформатики за сумісництвом і не в змозі приділити достатньо часу для ознайомлення та оволодіння відповідними технологіями програмування. У цьому випадку на допомогу можуть прийти різноманітні інтерактивні On-Line сервіси мережі Інтернет, які мають допомогти заповнити інформаційний вакуум, а також надати вчителям та учням своєчасну допомогу в оволодінні мовою C#.

## Висновки до розділу 1

Галузь розробки програмного забезпечення вже пройшла неабиякий шлях розвитку, що знаходить своє відображення в підходах до навчання інформатики. Навчання програмування дещо відстає від сучасних технологій написання комп'ютерних програм та не відповідає перспективам цієї галузі. Актуальним є навчання об'єктно-орієнтованого програмування в курсі інформатики ЗНЗ на основі сучасних мов та середовищ розробки програмного забезпечення.

Оскільки програмування, значною мірою, визначальний елемент сучасних виробничих та суспільних технологій, то й навчання цієї дисципліни є пріоритетним саме в класах технологічного профілю.

Зміст і обсяг навчального матеріалу визначає профільне спрямування навчання на основі поєднання професійної та фундаментальної складових знань. Навчання програмування можна використовувати для формування необхідного профілю. Навчання програмування – необхідна складова технологічного навчального профілю.

Профільне навчання охоплює базові предмети, профільні предмети, курси за вибором. Курси за вибором є основним засобом профільного навчання інформатики. Для покращення диференціації навчання інформатики у класах технологічного профілю доцільно розробити та впровадити курси за вибором

для навчання сучасним методам програмування на базі новітніх мов та середовищ.

Навчання програмування доцільно проводити за умов використання різних форм забезпечення профільної диференціації. Все ж ще багато форм організації профільного навчання не знайшли широкого використання. Такі форми навчання, як профільні групи в багатопрофільних загальноосвітніх навчальних закладах; профільне навчання за індивідуальними навчальними планами і програмами; динамічні профільні групи (у тому числі різновікові), потребують відповідного методичного забезпечення, вказівок та інструкцій щодо їх використання.

Допрофільна підготовка може суттєво допомогти учням визначитися зі своїм профільним спрямуванням, а отже, і оптимально сформувати профільні навчальні групи чи класи.

Відомості про платформи типу Microsoft .NET заслуговують якнайширшого впровадження до навчального матеріалу уроків інформатики у загальноосвітніх навчальних закладах.

Доцільним є використання сучасних професійних мов програмування, зокрема C#, для навчання в класах технологічного профілю загальноосвітнього закладу.

Наявні нині в школах програмно-технологічні можливості достатні для використання мови C# для навчання програмування. Вимоги, які ставить таке впровадження, суттєво не відрізняються від тих, що необхідні для навчання інших мов програмування, а в окремих випадках і суттєво нижчі. Потребує додаткової уваги проблема підготовки педагогічних кадрів.

Основні результати першого розділу опубліковані в роботах [302; 307; 311; 312; 314; 315].

# РОЗДІЛ 2.
# МЕТОДИЧНІ ЗАСАДИ НАВЧАННЯ ПРОГРАМУВАННЯ УЧНІВ У КЛАСАХ ТЕХНОЛОГІЧНОГО ПРОФІЛЮ З ВИКОРИСТАННЯМ МОВИ C#

У другому розділі розглядається загальна методика дослідження проблеми, обґрунтовується вибір напряму досліджень, викладається загальна методика проведення дисертаційного дослідження, наводяться методи вирішення задач та їх порівняльні оцінки, описуються основні тенденції, закономірності, методи розрахунків, гіпотеза, що розглядається.

Визначаються методичні та дидактичні аспекти пропонованої методичної системи навчання програмування учнів класів технологічного профілю з використанням мови C#. Визначено мету, завдання, зміст, форми, методи, засоби як основні компоненти методичної системи навчання програмування учнів класів технологічного профілю з використанням мови C#, проаналізовано педагогічні умови впровадження мови C# у шкільний курс інформатики. У зв’язку зі змінами у шкільних навчальних програмах, що відбулися за останні роки, методична система навчання програмування у класах технологічного профілю представлена у динаміці її змін у відповідності до програмних вимог.

Розглянуто можливі підходи щодо навчання об’єктно-орієнтованого програмування мовою C#. В результаті дослідження визначено зміст і методичні особливості навчання розділу «Основи алгоритмізації та програмування» з курсу інформатики в класах технологічного профілю.

Ілюструється використання мови C# для написання найпростіших програм для консольного виконання. Наводяться приклади розв'язування задач мовою програмування C#. Вказано суттєві особливості написання програм мовою C#. Розглядається методика ознайомлення та вивчення основних принципів об’єктно-орієнтованого програмування. Наведено застереження щодо можливих помилок, які можуть зумовити методично недосконале навчання програмування мовою C#.

## 2.1. Загальна методика дослідження проблеми

Актуальність навчання об’єктно-орієнтованого програмування на основі сучасних мов та середовищ, важливість впровадження такого навчання в курсі інформатики класів технологічного профілю ЗНЗ потребує переосмислення теоретико-методологічних засад та концептуальних підходів його здійснення.

*Провідною ідеєю дослідження* є положення про те, що навчання програмування на основі використання мови C# у класах технологічного профілю забезпечує відповідне профільне спрямування, надає можливість навчати основам об’єктно-орієнтованого підходу до написання комп’ютерних програм, забезпечує належний рівень внутріпредметних та міжпредметних змістових зв’язків навчального матеріалу.

*Концепція дослідження* ґрунтується на використанні об’єктно-орієнтованої парадигми програмування засобом мови програмування C# для створення авторської методичної системи навчання програмування в класах технологічного профілю загальноосвітніх навчальних закладів.

Провідні ідеї концепції дослідження відбиті у *гіпотезі*, яка ґрунтується на припущенні, що за умов використання науково-обґрунтованої та розробленої авторської методики навчання програмування учнів у класах технологічного профілю вдасться:

– забезпечити теоретичне та практичне знайомство з елементами ООП учнів класів технологічного профілю під час навчання програмування;

– покращити узгодженість, внутріпредметні зв’язки, змістову єдність розділу «Основи алгоритмізації та програмування» з іншими розділами шкільної інформатики;

– покращити узгодженість, міжпредметні зв’язки, змістову єдність інформатики з іншими предметами, що вивчаються в класах технологічного профілю загальноосвітніх навчальних закладів.

*Методологічну основу дослідження складають:*

– *на філософському рівні методології*: теоретичні положення щодо визначення філософських категорій взаємодії, інформації, структури та відображення; положення теорії пізнання; діалектичний принцип взаємозв'язку й взаємообумовленості закономірностей і явищ об'єктивної дійсності; взаємозв'язок теорії та практики у процесі пізнання; активна роль особистості у пізнанні й перетворенні дійсності; системний підхід до теоретичного осмислення та практичної організації процесу навчання програмування;

– *на загальнонауковому рівні методології:* концептуальні положення про взаємозв'язок, взаємозумовленість і взаємозалежність педагогічних явищ та процесів; концептуальні положення про зв'язок мови, мовлення і мислення, їх визначальне місце й роль у розвитку особистості; теорія розвивального навчання; теорія інформаційного суспільства; теоретичні основи побудови алгоритмів та програм; концепції сучасних парадигм програмування;

– *на конкретно-науковому рівні методології:* основні положення щодо формування системи та структури профільної організації навчання; системно-структурний підхід до аналізу навчальної діяльності; компетентнісний, особистісно-орієнтований, діяльнісний і комплексний підходи до розвитку, формування особистості учня.

Для досягнення мети і реалізації завдань дослідження застосовувався комплекс **методів**:

*теоретичні* – аналіз наукової вітчизняної та зарубіжної педагогічної та навчально-методичної літератури, наукові узагальнення досвіду вчителів інформатики загальноосвітніх навчальних закладів з метою виявлення й систематизації дослідницьких матеріалів; педагогічного прогнозування і моделювання для побудови ефективного навчально-виховного процесу, згідно з предметом дослідження; методи моделювання, у процесі якого була розроблена структурна модель навчання програмування учнів класів технологічного профілю загальноосвітніх навчальних закладів на уроках інформатики;

моделювання процесу навчання програмування в класах технологічного профілю ЗНЗ – уможливило розширення висновків з проблеми дослідження; рефлексія власної професійної діяльності сприяла визначенню взаємозв'язків під час суб'єкт-суб'єктної взаємодії, організації самостійної роботи та контролю успішності учнів;

*емпіричні* – цілеспрямоване спостереження за навчальною діяльністю учнів загальноосвітніх навчальних закладів на уроках інформатики, опитування учнів (усне й письмове), бесіди, опитування й анкетування вчителів сприяли вивченню досвіду вчителів інформатики з навчання програмування в класах технологічного профілю; застосування методу семантичного диференціалу – дозволило виявити позитивні сторони і недоліки навчання програмування в класах технологічного профілю на основі мови програмування C# та віднайти шляхи удосконалення методики такого навчання;

*експериментально-теоретичного рівня* – педагогічний експеримент, щодо оцінювання ефективності пропонованої методики навчання програмування. Застосування педагогічного експерименту допомогло реалізувати програму дослідження та виявити кількісні і якісні зміни під час навчання програмування в класах технологічного профілю загальноосвітніх навчальних закладів на основі мови програмування C#;

*математичної статистики* для обробки експериментальних даних, якісний і кількісний аналіз результатів проведеного дослідження за допомогою математичної статистики та комп'ютерної обробки даних експерименту забезпечили вірогідність отриманих результатів дослідження в динамічних змінах та часовій послідовності.

Дослідження проводилося протягом 2007–2011 років, воно охоплювало три етапи науково-педагогічного пошуку.

*Перший етап* (2007 – 2009 рр..) полягав у вивченні проблем навчання програмування в класах технологічного профілю. На цьому етапі були проведені:

– аналіз змісту і системи вимог до навчання інформатики у загальноосвітніх закладах, аналіз динаміки та причин зміни змісту;

– аналіз навчальної літератури з предмету, а також аналіз методичної, дидактичної, психологічної літератури дозволив виробити вихідні методологічні позиції дослідження;

– анкетування учнів і вчителів, бесіди, що сприяли конкретизації підходів до побудови методики навчання програмування в класах технологічного профілю;

– сформовані гіпотеза, цілі та методи дослідження.

*Другий етап* (2009–2011 рр.) – проведення формувального експерименту полягав у розробці та впровадженні методики навчання програмування на основі мови C# в класах технологічного профілю та включав: уточнення і конкретизацію дидактичних засад навчання програмування учнів профільних класів ЗНЗ на основі мови програмування C#; поняття "об’єктно-орієнтоване програмування", розробка класифікації умінь розробки комп’ютерних програм, їхніх структури і функцій, значення під час навчання інших шкільних предметів; визначення дидактичних умов навчання програмування учнів профільних класів загальноосвітніх навчальних закладів на основі мови програмування C#, критеріїв і показників рівнів сформованості умінь розробки комп’ютерних програм; розробка моделі й навчально-методичного забезпечення процесу навчання програмування учнів профільних класів ЗНЗ на основі мови програмування C#. Розробка методики навчання програмування в класах технологічного профілю, апробація теоретичних і методичних підходів у публікаціях і виступах, експериментальне навчання програмування в класах технологічного профілю, виявлення результативності розробленої методики навчання програмування мовою C# в класах технологічного профілю; запровадження моделі і навчально-методичного забезпечення навчання програмування учнів профільних класів загальноосвітніх навчальних закладів на основі мови програмування C#.

*Третій етап* (2010-2011 рр.) – проведення контролюючого експерименту, під час якого перевіряли ефективність моделі і навчально-методичного забезпечення навчання програмування учнів профільних класів ЗНЗ на основі мови програмування C#; коригували теоретичні положення та експериментальні методики, розробляли методичні рекомендації, уточнювали теоретичні положення. Оформлення дисертаційної роботи.

*Експериментальна база дослідження.* Дослідно-експериментальна робота здійснювалася на базі загальноосвітніх навчальних закладів Романівського району Житомирської області: Миропільської гімназії, Миропільської загальноосвітньої школи І-ІІІ ступенів № 2, Камінської загальноосвітньої школи І-ІІІ ступенів, Гордіївської загальноосвітньої школи І-ІІІ ступенів, Биківської загальноосвітньої школи І-ІІІ ступенів, Романівського ліцею. А також на базі чотирьох навчальних закладів інших районів Житомирської області: загальноосвітньої школи І-ІІІ ступенів № 7 м. Житомира, загальноосвітньої школи І-ІІІ ступенів № 21 м. Житомира, загальноосвітньої школи І-ІІІ ступенів № 3 м. Олевська та Озернянської загальноосвітньої школи І-ІІІ ступенів, Житомирського району.

Дослідно-експериментальною роботою було охоплено 255 учнів. З них 126 учнів в експериментальних та 128 учнів у контрольних групах класів технологічного профілю. Також участь в організації та проведенні експериментального дослідження взяли 12 учителів інформатики.

Результати експерименту визначались шляхом порівняння результатів анкетування учнів експериментальних та контрольних груп з результатами анкетування експертів. Формування експертних оцінок проводилось із залученням фахівців з галузі розробки програмного забезпечення провідних українських компаній, учителів-методистів інформатики Житомирської, Хмельницької, Львівської, Полтавської, Київської та інших областей України, викладачів вищих навчальних закладів, інших науковців, студентів старших

курсів, які навчаються у вищих навчальних закладах м. Києва та м. Житомира за спеціальностями, пов’язаними з розробкою програмного забезпечення.

*Вірогідність результатів дослідження* забезпечено обґрунтованістю вихідних положень щодо навчання програмування учнів класів технологічного профілю на основі використання мови C#; використанням комплексу взаємодоповнюючих методів наукового пошуку, адекватних меті, об’єкту, предмету і завданням дослідження; вивченням об’єкта дослідження з урахуванням усіх вимог щодо одержання інформації, надійності, обґрунтованості, точності пізнавальних процедур; репрезентативністю вибірки; поєднанням кількісного і якісного аналізів теоретичного й емпіричного матеріалів; використанням порівняльних методик; застосуванням елементів математичної статистики для обробки експериментальних даних.

## 2.2. Основні компоненти методичної системи навчання програмування учнів класів технологічного профілю з використанням мови C#

### 2.2.1. Методична система як освітній компонент

Система (від дав.-гр. σύστημα – «сполучення») – складне і багатогранне поняття. Означити поняття «система» можна як множину взаємопов'язаних елементів, відокремлених від середовища, що взаємодіє з ним, як ціле [211]. О. М. Гончарова, узагальнивши різні трактування терміну система, виділяє три основні властивості, якими повинен володіти об’єкт, щоб його можна було розглядати як систему:

1. Цілісність і членованість. Система являє собою цілісну сукупність елементів. Це означає, що з одного боку система являє собою єдине і цілісне утворення, що володіє властивостями емерджентності 2-го роду, а з іншого боку – в її складі можуть бути виділені окремі об'єкти – елементи. Зауважимо, що для системи цілісність є первинною ознакою [243].

2. Зв'язки. Функціонування об'єктів як систем забезпечується наявністю зв'язків і (або) відносин (абстрактна форма представлення зв'язків) між

окремими елементами системи і (або) їх властивостями, а також між досліджуваною системою в цілому і оточуючим світом – метасистемою. Зв'язки накладають обмеження на поведінку як окремих елементів усередині системи, так і досліджуваної системи в цілому.

3. Інтегративність. Системі в цілому притаманні інтегративні властивості (якості), які не властиві жодному з її елементів окремо. Властивості системи хоча і залежать від властивостей окремих її елементів, але не визначаються ними повністю [59, с. 150].

Методична система навчання (МСН) визначається дослідниками як складна система функціонування елементів, яка залежить від багатьох чинників [94; 182; 272]. Зокрема, зазначаються обов'язкові компоненти методичної системи: цілі навчання, його зміст, методи, засоби, організаційні форми [233]. Вказується на соціальний її характер [94; 186]. Також зазначаються чинники, які вона повинна враховувати: операційний склад умінь (сукупність дій і операцій, що входять в уміння, і особливості зв'язків між ними); розроблені рівні програмних вимог до формування умінь та знань, що їм відповідають; особливості взаємозв'язку змістової (профілі і рівні знань), процесуально-операційної (форми і стиль мислення, способи та орієнтири діяльності), мотиваційної (інтереси, потреби, мотиви), прогностичної (прийняття рішення, складання програми діяльності, передбачення результату діяльності) складових навчальної діяльності учнів [272, с. 77].

Взаємозв'язок між цими компонентами дослідники пояснюють схематично. Одним із перших таке подання упровадив А. М. Пишкало (див. рис. 2.1). Він запропонував підхід, у якому всі компоненти навчального процесу утворюють єдине ціле із визначеними внутрішніми зв'язками. [233].

Схематичне подання методичної системи навчання інформатики, яке пропонує Н. В. Морзе (див. рис. 2.2), вказує на не симетричний взаємозв'язок між її основними компонентами (цілі, зміст, методи, засоби, форми) через додаткові компоненти цієї системи: очікувані результати навчання; технології

добору змісту, методів, форм і засобів навчання; технології встановлення зв’язків між елементами методичної системи [186, с. 24–25].

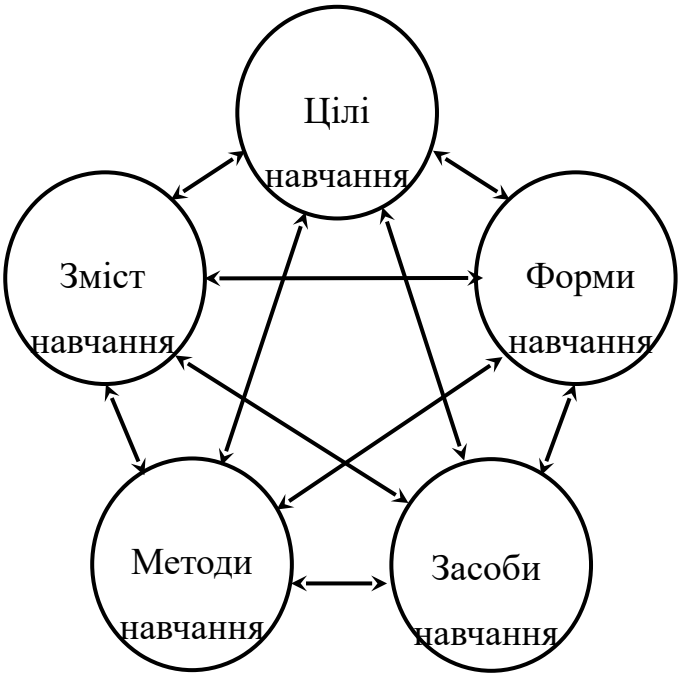



Рис 2.1. Поєднання компонентів методичної системи в їх взаємозв’язках

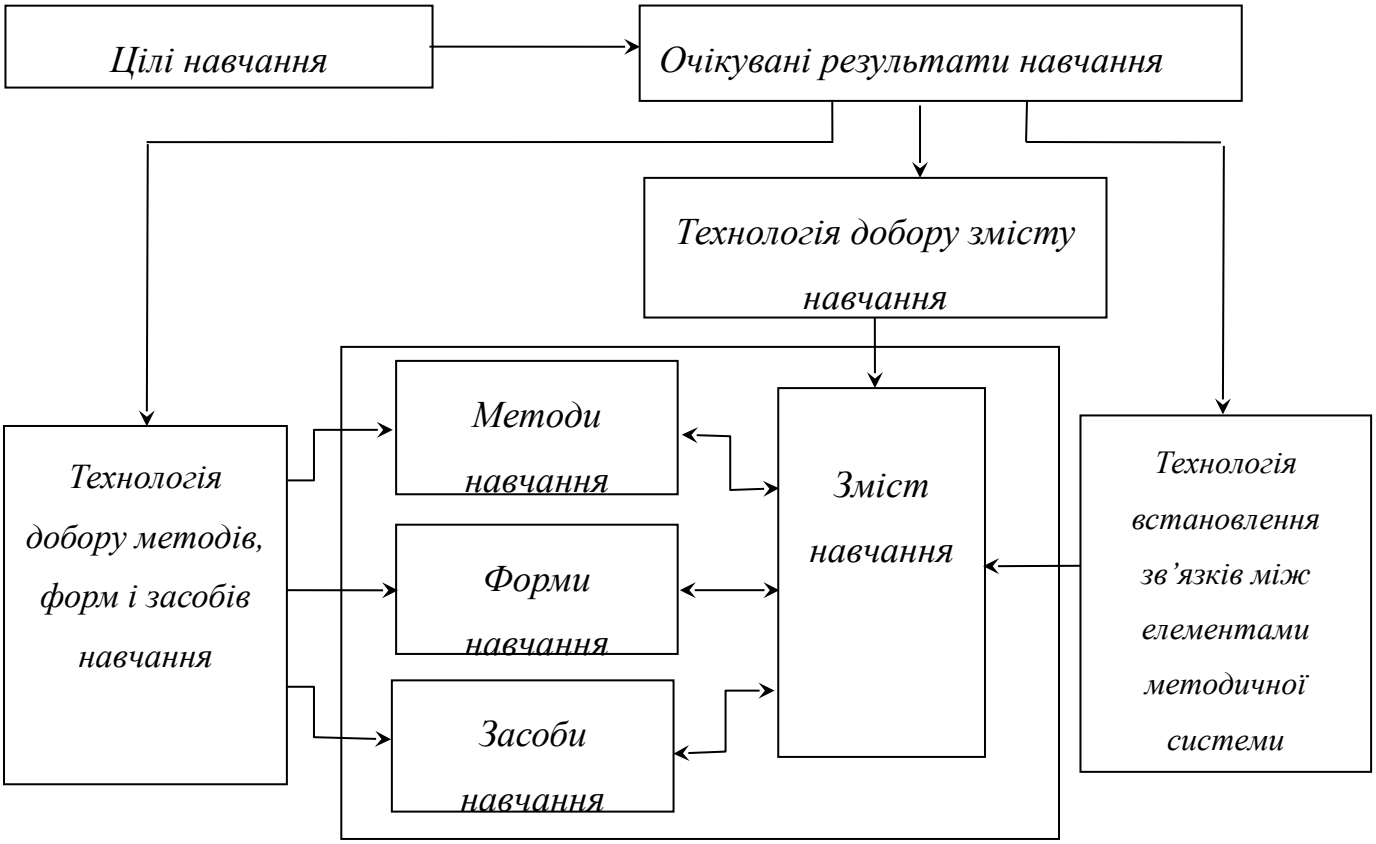



Рис. 2.2 Схематичне зображення методичної системи навчання в поєднанні з додатковими її компонентами (за Н. В. Морзе)

Оскільки МСН програмування в загальноосвітніх навчальних закладах є складовою методичної системи навчання інформатики, то їй притаманні практично одні і ті ж особливості. Зокрема, для МСН інформатики є характерним високий динамізм становлення та розвитку [182]. Н. В. Морзе вказує також на інші характерні особливості методичної системи навчання інформатики. Зокрема МСН інформатики значною мірою залежить від рівня інформатизації навчального процесу, розробки інформаційно-комунікаційних технологій (ІКТ) навчання і від використання ІКТ при вивченні різних навчальних предметів.

У нашому дослідженні, в загальному, будемо дотримуватись ідей побудови методичної системи навчання запропонованих Н. В. Морзе.

Вивчення будь-якого курсу, у тому числі і курсу навчання програмування, передусім, повинно бути цілеспрямованим, оперативним, гнучким і конкретним. Цього можна досягти, здійснюючи навчання за наступними принципами:

– відповідності змісту курсу сучасному замовленню суспільства щодо рівня освітньої підготовки учня;

– цілісності та комплексності (встановлення єдності всіх розділів курсу);

– систематичності та послідовності (охоплення всіх учнів різними формами навчальної діяльності протягом року з урахуванням структури курсу);

– пріоритетності (вибір і дотримання певної наукової проблеми, яка має вирішальне значення для даного курсу);

– науковості ( орієнтації на досягнення галузі сучасного програмування та психолого-педагогічної науки);

– практичної спрямованості (підпорядкування всіх аспектів навчального процесу завданням курсу).

### 2.2.2. Загальні риси методичної системи навчання програмування на основі використання мови C# на уроках інформатики в класах технологічного профілю

Розглянемо окремі компоненти методичної системи навчання програмування учнів класів технологічного профілю на основі мови програмування C#: цілі навчання, зміст, засоби, методи, організаційні форми. У зв'язку з появою та впровадженням нових шкільних навчальних програмах, що відбулися за останні роки, МСН програмування у класах технологічного профілю представлена у динаміці її змін у відповідності до програмних вимог. Усі її компоненти розглянуті нами у тому варіанті, що стосувався проведення педагогічного експерименту та зазначаються окремі доповнення, які методична система отримала у зв'язку з змінами у навчальній програмі з інформатики для загальноосвітніх навчальних закладів. Використання мови програмування C# на уроках інформатики в класах технологічного профілю детермінує відповідні зміни мети, завдань, змісту, методів, засобів та організаційних форм навчання на комплексній інтегрованій основі.

#### 2.2.2.1. Цілі навчання

Загальноосвітні цілі навчання пов'язані з набуттям певного рівня *компетентностей*. Структурними складовими таких компетентностей є відповідні *знання*, *уміння* та *навички.* Тобто, при визначенні цілей навчання програмування учнів класів технологічного профілю на основі мови програмування C# потрібно вказувати відповідні цілі навчального процесу: *освітні, розвиваючі, виховні*. Крім того, навчання програмування, як складова курсу інформатики, повинно здійснюватись у відповідності з цілями цієї начальної дисципліни. Також компетентнісний підхід тісно пов'язаний з концепцією неперервної освіти [149; 95]. Швидкий розвиток ІКТ зумовлює те, що цілі навчання повинні формуватись не лише з огляду на існуючі суспільно-економічні пріоритети, а й передбачати обов'язкове моральне старіння та неминуче зменшення актуальності набутих під час навчання компетентностей.

З огляду на це, навчання у загальноосвітніх навчальних закладах повинно складати фундамент для подальшої освіти та саморозвитку особистості. Досягти цього можна, забезпечивши достатню фундаментальну складову навчального матеріалу та шляхом його постійного оновлення.

**Освітня** мета навчання шкільного курсу інформатики – формування основ інформаційної культури школярів, тобто формування сукупності знань, умінь, навичок, що забезпечують учням можливість застосовувати комп'ютерну техніку в навчальній, а згодом – у професійній діяльності. Загальною метою навчання інформатики є формування складових інформаційної культури учнів [80]. Цілі навчання програмування формуються у контексті загальних навчально-виховних цілей та цілей курсу інформатики.

Навчання програмування **розвиває** в учнів загальнонаукові уміння та навички (організаційні, пізнавальні, пошукові). Ставить за необхідне створення умов для індивідуалізації та інтенсифікації навчального процесу, забезпечення самостійної роботи учнів. Результатом навчання програмування є сформованість умінь та навичок щодо побудови інформаційної моделі, необхідної для розв'язання відповідної задачі та розробки комп'ютерних програм за допомогою середовища програмування мовою C#.

**Виховні** цілі включають формування в учнів особистісних рис: наполегливості, старанності, послідовності, раціональності, виконавського сумління та групових навичок: роботи в колективі, у команді, міжособистісного спілкування. Важливе значення має усвідомлення витратності та економічної доцільності процесу розробки програмного забезпечення. Це сприяє вихованню свідомих громадян, які поважають авторські права на програмне забезпечення, засуджують використання неліцензійного програмного забезпечення [108].

#### 2.2.2.2. Зміст навчання

В цілому зміст навчання, як і інші його компоненти, визначається навчальними програмами. До 2010 року, на етапі реформи шкільної освіти, навчання інформатики у загальноосвітніх навчальних закладах здійснювалось, як

за програмами для 11-ї школи (у старших класах) [116; 115], так і за програмами, що передбачали перехід на новий 12-річний термін навчання [99; 100]. У зв'язку з відмовою від деяких положень концепції загальної середньої освіти [129] програми з інформатики, розраховані на 12-річний термін навчання, зазнали змін [97; 98]. В останні роки навчання інформатики здійснюється за програмами як для базового та академічного рівня, так і за програмами для поглибленого вивчення інформатики [117]. Одним із елементів, що визначає профільне спрямування навчання у загальноосвітніх навчальних закладах є курси за вибором, які спираються на відповідні навчальні програми [102; 103; 104; 101].

На етапі формуючого педагогічного експерименту навчання програмування на основі мови C# в класах технологічного профілю проводилося за програмою для 10-11 класів ЗНЗ фізико-математичного, природничого і технологічного профілів, автори: М. І. Жалдак, Н. В. Морзе, О. І. Мостіпан, Г. Г. Науменко [116, с. 65-85].

В результаті навчання, згідно вимог програми, повинні бути сформовані такі знання, уміння і навички:

***Учні повинні знати:***

– основні етапи розв'язування задачі з використанням комп'ютера;
– поняття інформаційної моделі задачі;
– поняття неформальної моделі задачі;
– визначення вхідних даних і результатів;
– поняття алгоритму;
– властивості алгоритму;
– способи та форми подання алгоритму;
– основні базові структури алгоритмів;
– сутність методу послідовного уточнення алгоритму;
– основні ідеї та принципи технології структурного програмування;
– порядок складання алгоритмів і програм;
– правила запису структурованих алгоритмів і програм;
– основні вказівки навчальної алгоритмічної мови;

– основні елементи однієї з мов програмування – алфавіт; основні поняття мови: числа, рядки, ідентифікатори, оператори, величини, операції;

– типи даних у мові програмування, набір функцій та операцій, допустимих для кожного з типів даних;

– принципи побудови опису програми мовою програмування;

– сутність вказівки надання значень; призначення та правила описування вказівок розгалуження й повторення; звернення до алгоритмів; поняття про алгоритми-процедури й алгоритми-функції;

– особливості використання табличних і рядкових величин;

– основні правила опису графічних операцій навчальною алгоритмічною мовою та мовою програмування;

– процедури та функції для побудови найпростіших графічних зображень.

***Учні повинні мати уявлення про:***

– класифікацію мов програмування;

– інтерпретацію та компіляцію;

– системи програмування;

– інтегровані середовища програмування.

***Учні повинні вміти:***

– застосовувати різні форми опису алгоритмів і переходити від однієї форми опису алгоритмів до іншої;

– визначати тип величини, описувати її навчальною алгоритмічною мовою та мовою програмування;

– використовувати прості й складені умови при побудові алгоритмів;

– застосовувати метод послідовних уточнень при побудові алгоритмів і програм;

– описувати алгоритми розв’язування задач різних типів навчальною алгоритмічною мовою та мовою програмування;

– складати й реалізовувати найпростіші лінійні, розгалужені, циклічні алгоритми на опрацювання табличних і рядкових величин, графічної інформації, на використання допоміжних функцій і процедур;

– вводити та налагоджувати програми на комп’ютері.

**Календарне планування**

Календарний план, згідно вище вказаної програми для загальноосвітніх навчальних закладів фізико-математичного, природничого та технологічного профілів. Інформатика, 10–11 класи [116, с. 65-85], розроблено на основі рекомендованого календарного планування Т. Г. Проценко [231]. Нижче наводиться тематичний зміст розробленого календарного плану, що включає перелік тем запланованих уроків, об’єднаних у певні теми навчальної програми, віднесені до розділу «Основи алгоритмізації та програмування». Перша цифра вказує номер розділу, друга – номер теми навчального матеріалу, а третя – номер уроку в даній темі:

9. **Основи алгоритмізації та програмування** (48 год)

9.1. **Інформаційна модель** (2 год)

9.1.1 Об'єкт. Класифікація та наслідування об’єктів. Модель. Моделювання об’єктів. Поняття інформаційної моделі.

9.1.2 Основні етапи розв'язування задач за допомогою комп'ютера. Побудова моделі.

9.2. **Алгоритми** (4 год.)

9.2.1 Методи об’єктів. Алгоритмічна сутність методів. Поняття алгоритму. Властивості алгоритмів.

9.2.2 Способи опису об’єктів та алгоритмів. Виконавець алгоритму. Основні алгоритмічні структури. Аналіз алгоритму.

9.2.3 Порядок складання алгоритмів. Конструювання «зверху донизу».

9.2.4 Структурний підхід до побудови алгоритмів. Сучасна галузь розробки програмного забезпечення. Об’єктно-орієнтоване програмування.

9.3. **Програма. Мова програмування** (10 год.)

9.3.1 Поняття програми. Мови та середовища програмування.

9.3.2 Класифікація мов програмування. Поняття редактора, транслятора, налагоджувача. Інтерпретація та компіляція. Компілятор. Поняття про системи програмування. Інтегровані середовища програмування.

9.3.3 Алфавіт мови програмування. Основні поняття мови: службові слова, ідентифікатори .

9.3.4 Практична робота № 6. Основні прийоми роботи в середовищі програмування.

9.3.5 Величини та їх опис мовою програмування. Дані, типи даних. Стандартні типи даних. Змінні та константи.

9.3.6 Арифметичні вирази. Бібліотеки підпрограм чи класів.

9.3.7 Структура програми. Уведення даних з клавіатури. Виведення даних на екран дисплею. Оператор присвоєння.

9.3.8 Створення лінійних програм.

9.3.9 Практична робота № 7. Створення лінійних програм.

9.3.10 Тематична атестація з теми: «Програма. Мова програмування».

9.4. **Звернення до алгоритмів і функцій** (5 год.)

9.4.1 Поняття підпрограми (метод як підпрограма). Процедури. Функції.

9.4.2 Формальні та фактичні параметри.

9.4.3 Область видимості змінних.

9.4.4 Практична робота № 8. Програми з процедурами та функціями.

9.4.5 Тематична атестація з теми: «Використання допоміжних алгоритмів».

9.5. **Вказівки повторення та розгалуження** (10 год.)

9.5.1 Оператори умовного переходу та вибору.

9.5.2 Використання операторів умовного переходу та вибору.

9.5.3 Практична робота № 9. Створення програм з використанням операторів умовного переходу та вибору.

9.5.4 Поняття циклу. Цикли з перед- та з післяумовою. Вкладені цикли.

9.5.5 Використання циклів з передумовою та післяумовою.

9.5.6 Практична робота № 10. Створення програм з використанням циклів.

9.5.7 Цикли з параметром.

9.5.8 Створення програм з використанням циклів з параметром.

9.5.9 Практична робота № 11. Створення програм з використанням циклів з параметром.

9.5.10 Тематична атестація теми: «Вказівки повторення та розгалуження».

9.6. **Масиви** (10 год.)

9.6.1 Обробка структурованих типів даних. Масиви.

9.6.2 Алгоритм роботи з таблицями. Введення та виведення значень масиву з використанням циклів.

9.6.3 Знаходження суми елементів масиву. Завдання обробки масивів.

9.6.4 Алгоритм пошуку елементів масиву із деякою властивістю.

9.6.5 Практична робота № 12. Складання та реалізація програм опрацювання одновимірних масивів.

9.6.6 Поняття сортування. Сортування масивів методом вибору.

9.6.7 Сортування одновимірних масивів методом обміну.

9.6.8 Практична робота № 13. Складання та реалізація програм сортування одновимірних масивів.

9.6.9 Поняття двовимірного масиву. Обробка двовимірного масиву.

9.6.10 Тематична атестація з теми: «Масиви».

9.7. **Рядкові величини** (5 год.)

9.7.1 Поняття рядка. Операції над рядковими величинами. Стандартні процедури та функції для обробки рядкових величин.

9.7.2 Найпростіші алгоритми роботи з рядками.

9.7.3 Створення алгоритмів для обробки рядкових величин.

9.7.4 Практична робота № 14. Створення алгоритмів для роботи з рядками у середовищі програмування.

9.7.5 Тематична атестація з теми: «Рядкові величини».

Підсумковий урок.

Цей же план зведений у таблицю для зручного використання учителем (див. додаток В.) завчасно опубліковано на сайті методичної підтримки педагогічного експерименту у розділі «Методичні матеріали» [332].

### 2.2.2.3. Розвиток змісту навчання програмування у контексті нових навчальних програм

Як уже зазначалось, починаючи з 2011 року навчання інформатики в 10-11 класах здійснюється за програмами для базового рівня, академічного рівня, поглибленого вивчення інформатики [117].

У програмі базового рівня майже відсутній матеріал, що торкається практичних аспектів навчання програмування. Проте розділ «Моделювання. Основи алгоритмізації» [97] передбачає вивчення двох фундаментальних понять галузі програмування «інформаційна модель» та «алгоритм». Такий підхід має на меті ознайомити учнів із загальними положеннями технологій розробки програмного забезпечення, моделюванням як способом досліджень, алгоритмічним підходом, застосовуваним у різних галузях. Проте, ці відомості подаються дещо розрізнено та відособлено. Відсутня змістова лінія, яка б єднала поняття алгоритму, інформаційної моделі та принципів функціонування сучасного програмного забезпечення. Такими єднаючими загально-пізнавальними та науково-теоретичними відомостями можуть бути навчальні матеріали про об'єктний підхід до розробки програмного забезпечення.

У програмі академічного рівня (Навчальна програма з інформатики для 10-11 класів загальноосвітніх навчальних закладів. Академічний рівень. (авт.: І. О. Завадський, Ж. В. Потапова, Ю. О. Дорошенко [98]) матеріал з навчання програмування поданий більш широко. На вивчення розділу «Основи алгоритмізації та програмування» відводиться 28 навчальних годин. Згідно вимог програми, протягом перших уроків на вивчення виносяться такі питання: Поняття моделі. Типи моделей. Моделювання як метод дослідження об'єктів. Поняття програми як автоматизованої системи. Складові програми: дані, логіка, інтерфейс. Поняття об'єкта у програмуванні. Властивості об'єкта. Елементи інтерфейсу користувача як об'єкти. Поняття події та обробника події. Поняття про методи об'єкта.

Програма з інформатики академічного рівня передбачає такі результати навчання перших уроків теми «Основи алгоритмізації та програмування»:

Учень описує:

– поняття моделі, об'єкта, предметної галузі;

– типи моделей, їх характеристики;

Учень пояснює:

– поняття програми;

– поняття даних, їх роль та способи зберігання;

– поняття програмної логіки та інтерфейсу;

– поняття об’єкта, властивостей і методів об’єкта;

– поняття події й обробника події;

Учень наводить приклади:

– подій для таких елементів управління, як кнопка, поле, поле зі списком;

Учень вміє:

– будувати інформаційну модель задачі [98, с. 39].

У тексті програми зміст навчального матеріалу цієї теми записано в наступному порядку:

1. Загальні поняття про моделювання як спосіб пізнання та діяльності.

2. Комп’ютерна програма як різновид інформаційної моделі.

3. Поняття алгоритму та програми.

4. Мова програмування та середовище програмування.

5. Елементи програми, логіка інтерфейс.

6. Об’єкти у програмуванні на прикладі елементів інтерфейсу як програмних об’єктів.

7. Властивості та методи об’єктів, події, обробка подій.


Рис 2.3. Традиційний порядок навчання програмування

Такий порядок типовий для більшості сучасних навчальних програм для ЗНЗ з інформатики. Більш наочно послідовність такого навчання вказано на рисунку 2.3. Така послідовність питань відповідає історичному розвиту галузі у минулому, найбільше підходить для навчання на основі парадигми структурного програмування і мало торкається сучасних підходів до розробки програмного забезпечення.

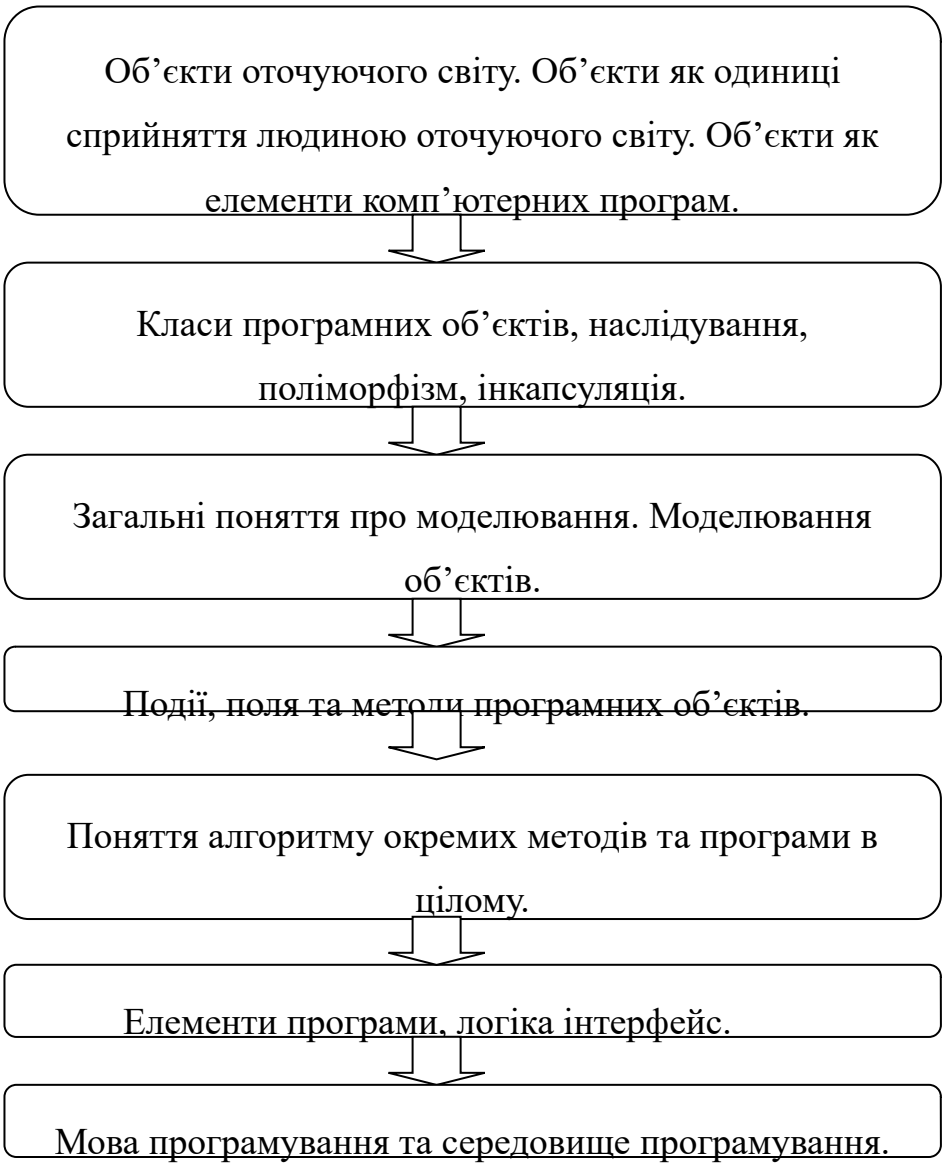



Рис 2.4. Порядок навчання програмування на основі об’єктного підходу

Якщо в основу навчання вчитель збирається покласти парадигму об’єктно-орієнтованого програмування, то подавати матеріал доцільно дещо в іншій послідовності (див. рис. 24.). Вказаний перелік понять дещо ширший за ті, що пропонує програма та, поряд із цим, дозволяє розглянути практично усі

поняття процедурного програмування, хоча і під дещо іншим кутом. Такий підхід не заперечує парадигму процедурного програмування, а розглядає її як складову парадигми програмування об’єктно-орієнтованого. Саме таке доповнення перших уроків розділу надає можливість здійснювати навчання програмування на основі сучасних об'єктно-орієнтованих мов програмування, до яких належить мова C#. У додатку наводяться поурочні плани вивчення перших уроків з теми «Основи алгоритмізації та програмування», побудовані на основі інтерактивних технологій навчання. (див. додаток Є.)

**2.2.2.4. Засоби навчання**

Тлумачення поняття «засіб навчання» має деякі особливості. Навіть одні і ті ж дослідники в різних своїх роботах звертають увагу на дещо різні аспекти цього поняття: «Під засобами навчання ми розуміємо спеціально утворені об'єкти, які формують навчальне середовище та беруть участь у навчальній діяльності, виконуючи при цьому навчальну, виховну та розвивальну функції [89]».; «Під засобами навчання ми розуміємо предмети, які формують матеріальну складову навчального середовища та приймають участь у навчальній діяльності [90]». Поряд із цим варто звернути увагу на англомовний синонім поняття «засіб навчання» – «educational media». В зарубіжній літературі широко розповсюджене таке його пояснення: «Під цим терміном розуміють будь-які засоби, агенти або інструменти, що використовуються для передачі навчальної інформації [146]».

Дослідники наводять переліки засобів навчання у контексті тієї чи іншої методичної системи. Зокрема, Н. В. Морзе включає до методичної системи навчання інформатики такі засоби, як навчально-методичний комплекс, прикладні програмні засоби для комп'ютерної підтримки навчально-пізнавальної діяльності при вивченні основних розділів інформатики, прикладні програмні засоби для контролю знань, умінь і навичок, комп'ютер, засоби телекомунікацій, відеотехніка, підручники і навчальні посібники [182,с. 12].

А. П. Забарна називає домінуючі апаратні та програмні засоби навчання інформатики у профільній школі: комп’ютерно-орієнтований навчально-методичний комплекс професійного спрямування, прикладне програмне забезпечення для розв’язання дослідницьких задач, засоби телекомунікації, відеотехніка, інтерактивні дошки, підручники, навчальні посібники, практикуми, зокрема і електронні [94].

Методична система навчання програмування на основі використання мови програмування C# спирається на комп’ютерну підтримку практичної роботи учня над розв’язанням навчальних задач з об'єктно-орієнтованого програмування. Під час навчання програмування як засоби навчання, в їх самому широкому розумінні, виступають мови та середовища програмування. Поряд із цим мова та середовище програмування, що використовуються під час навчання, одночасно постають не лише як засіб навчання, а здебільшого саме як об'єкт (предмет) вивчення. Визначальними засобами навчання у межах методичної системи, що розглядається, є мова програмування C# та певне середовище розробки програмного забезпечення, що підтримує написання програм цією мовою.

Проблема добору середовища програмування розглядалася нами в першому розділі дослідження. Окремо доцільно розглянути таку його характеристику як особливості інтерфейсу. Інтерфейс повинен відповідати вимогам зручності та зрозумілості. Ті чи інші особливості інтерфейсу можуть бути по-різному використані педагогом під час навчання. Простий, традиційний інтерфейс дуже часто виявляється більш методично доцільним, а відсутність деяких елементів управління може навіть сприяти кращому опануванню тих чи інших особливостей програмування. Прикладом може бути окремий запуск на компіляцію коду та на виконання вже відкомпільованої програми. Поряд із цим складний багатофункціональний інтерфейс може відволікати учня від виконання основних навчальних завдань. Незрозумілі пункти меню, панелі інструментів, інші елементи інтерфейсу потребують часу,

якщо не на освоєння, то хоча б на загальне пояснення їх призначення. Замість того, щоб зосередитись на мові програмування, учень розпорошує свою увагу. В окремих випадках складний багатофункціональний інтерфейс навіює учням думку про непосильність і без того не простого навчання програмування.

У більшості візуальних середовищ програмування реалізовано функції автоматичної генерації коду. Традиційно автоматична генерація коду використовується при створенні форм, кнопок, перемикачів та інших візуальних елементів. Деякі сучасні середовища можуть автоматично генерувати мало не будь-які фрагменти програм, а то навіть і цілі програми. Автоматична генерація коду дозволяє, з одного боку, прискорити виконання учнем завдань, продемонструвати учням ефективність та потужність сучасних засобів програмування, з іншого боку, використання готового коду, до якого варто віднести автоматично генерований код, недостатньо сприяє розумінню учнями процесу розробки програми та принципів її роботи.

Для деяких мов програмування існують спеціальні, спрощені навчальні версії. Особливо зручні такі мови і розроблені для них середовища програмування для знайомства з програмуванням школярів молодшого та середнього шкільного віку. Прикладом такого середовищ є Small Basic. Як зазначено на офіційній сторінці Small Basic: «Microsoft Small Basic – система, яка намагається зробити програмування зрозумілим для початківців [354]». Розробники, окрім простоти та доступності, зазначають головну перевагу Small Basic – його сучасність. Варто зазначити, що нам невідомо про існування подібного середовища для початківців, яке б підтримувало хоча б якусь мову програмування, що має С- подібний синтаксис. Швидше за все, доцільно створити та впровадити таке спеціальне навчальне середовище, яке б спростило навчання програмування мовою C# учнів молодшого віку. Корисно передбачити для такого середовища можливість унаочнити об’єктно-орієнтоване програмування.

Наведені вище та інші результати дослідження проблеми добору мови програмування, як засобу навчання, опубліковано нами в електронному фаховому виданні «Інформаційні технології і засоби навчання» [302]. Характеристики існуючих середовищ програмування мовою C# описано там же, та в першому розділі даного дослідження. Найкращою є та ситуація, коли вчитель має змогу самостійно вибрати середовище програмування відповідно до визначених навчальних завдань та наявного апаратного і програмного забезпечення. Допомогти в цьому може таблиця назв та деяких відомостей про найбільш поширені середовищ розробки мовою C#.

*Таблиця 2.1.*

**Найбільш поширені середовища розробки мовою C#**

| Назва | Розробник | Варіант безоплатного використання | Офіційний сайт | Особливості, додаткові умови |
|---|---|---|---|---|
| Microsoft Visual Studio Professional | Microsoft | Безкоштовно доступні пробні 90 денні версії | http://msdn.microsoft.com | Дещо громіздке |
| Microsoft Visual C# Express Edition | Microsoft | Безкоштовне | http://msdn.microsoft.com | Має незначні обмеження функцій та неповні бібліотеки класів |
| Sharp Develop | Незалежне середовище розробки | Безкоштовне | http://www.icsharpcode.net | Незалежне середовище розробки |
| Mono Develop | Xamarin (Novell) | Безкоштовне | http://monodevelop.com | Кросплатформне для Mono |
| Borland C# Editor | Borland | Нині не підтримується | | |
| Antechinus C# Editor | C-point | Безкоштовне ознайомлення впродовж 30 днів | http://www.c-point.com/ | працює з усіма версіями .Net Framework та операційних систем Windows |

### 2.2.2.5. Методи навчання

Український педагогічний словник дає визначення поняттю «метод» у його націленні на освоєння дійсності, як процесу її дослідження: «Метод (грец. μεθοδος – шлях дослідження чи пізнання) – спосіб організації практичного й теоретичного освоєння дійсності, зумовлений властивостями об'єкту дослідження. З розвитком науки відбувається розвиток і диференціація методів, що приводить до виникнення вчення про методи – методології [286, с. 205]». У більш широкому розумінні метод – спосіб досягнення мети, сукупність прийомів та операцій теоретичного або практичного освоєння дійсності, а також людської діяльності, організованої певним чином [44, с. 633]. Для досягнення освітніх цілей застосовуються *методи навчання*. Метод навчання – це спосіб упорядкованої взаємозв'язаної діяльності вчителя й учнів, спрямованої на досягнення завдань освіти, розвитку і виховання в процесі навчання [25, с. 57].

Дослідники вказують на важливі загальні функції, що виконують методи навчання в його процесі та наводять переліки методів навчання, що найбільш широко використовуються у контексті тієї чи іншої методичної системи. Зокрема С. П. Бондар, виходячи з особливостей і завдань профільного навчання, провідними функціями методів навчання у профільній школі вважає спонукальну, мотиваційну, освітню (навчальну), пошуково-розвивальну, дослідницько-розвивальну та комунікативну функції [25, с. 57]. Н. В. Морзе включає до методичної системи навчання інформатики такі методи навчання, як-от: метод проектів, самостійна робота учнів з ППЗ, тренажери, пошук інформації в мережі Інтернет, проблемний, дослідницький, частково-пошуковий [182, с. 12]. Стосовно навчання інформатики у профільній школі А. П. Забарна наводить такі домінуючі методи навчання інформатики: метод активного навчання, інтерактивні методи і технології навчання, контекстне навчання, проблемне навчання, ситуаційне навчання, кооперативне навчання, метод проектів, зокрема робота над телекомунікаційним проектами [94].

Під час навчання програмування мовою C# передбачається використання найбільш широкого діапазону навчальних методів. Проте застосування деяких методів навчання має певні особливості.

**Метод проектів** поряд з освітньою (навчальною) виконує пошуково-розвивальну та дослідницько-розвивальну функції. Проте цей метод варто використовувати лише за умови, коли учні володіють певним багажем знань, умінь та навичок для виконання цілісних проектних завдань. Використання цього методу на перших уроках теми пов'язано з певними труднощами.

**Самостійна робота учнів** із **середовищем програмування** повинна активно використовуватись на всіх етапах навчання мовою C#. Лише свідома практична реалізація вивченого у вигляді діючих програм може вважатися певним підсумком навчання програмування.

**Пошук інформації в глобальній мережі Інтернет** доцільно доручати учням як на початку даного курсу (коли виникає проблема обґрунтувати причини добору мови програмування), так і на решті його етапах. Такі завдання надзвичайно корисні для загального оволодіння учнів ІКТ.

**Проблемний, дослідницький** та **частково-пошуковий методи** корисно реалізовувати для навчання на етапі знайомства з новою темою, та під час розв'язання задач, що базуються на раніше вивченому матеріалі.

**Інтерактивні методи і технології навчання** надають можливість урізноманітнити навчальний процес та значно підвищити його ефективність. Використання цих методів вимагає ретельної підготовки вчителя та забирає багато часу. Особливим у використанні інтерактивних методів для навчання програмування є те, що таке використання дуже відрізняється від застосування інтерактивного навчання на уроках інших шкільних дисциплін. Ці особливості зумовлені інтерактивним характером взаємодії учня з комп'ютером у процесі практичного виконання навчальних завдань. Для навчання загальнотеоретичних питань курсу інтерактивні методи є надзвичайно ефективними, зокрема нами детально розроблено методику використання

інтерактивних методі для знайомства із загальними поняттями основ програмування на основі об’єктно-орієнтованої парадигми (див. додаток Є).

**Робота над телекомунікаційними проектами –** досить новий метод навчання, який поки що не можна віднести до основних. Завдяки швидкому розвитку Інтернеттелекомунікацій у найближчий час такі проекти тісно увійдуть до процесу навчання програмування. Нині, як певна нерозробленість цього методу навчання, так і недостатній розвиток телекомунікацій роблять його засобом для сміливих експериментів та новаторства педагогів.

Враховуючи особливості навчання програмування та його прикладну спрямованість, ми вважаємо, що важливу роль відіграють усі методи навчання, які передбачають елементи творчої діяльності.

### 2.2.2.6. Організаційні форми навчальної діяльності

Досягнення цілей навчання програмування обумовлюється не лише застосуванням тих чи інших методів, але й організаційними формами процесу навчання. Форми визначаються у взаємозв’язку з іншими компонентами методичної системи (цілі, зміст, методи, засоби) та конкретними умовами навчання: місце і час навчання, кількість учнів, забезпеченістю навчального закладу засобами навчання, тощо.

Форма (лат. forma — зовнішність, устрій, образ) — 1) зовнішній вигляд, обрис предмета; 2) будь-яке зовнішнє вираження змісту; 3) вид, устрій, тип, структура чогось; 4) спосіб здійснення, виявлення дії, процесу тощо [232, с. 693]. Загальне значення філософської категорії «форма» розглядається переважно у тісному зв’язку з категорією «зміст» [44, с. 963].

Визначаючи поняття «форма навчання», дослідники дають цьому терміну багатогранні трактування. С. О. Семеріков, наводячи означення форми навчання, наголошує на її системності та змістовому наповненні: «Форма навчання – цілеспрямована, чітко організована, змістовно насичена й методично забезпечена система пізнавального та виховного спілкування, взаємодії, співпраці викладачів та студентів [259]» Н. Є. Мойсеюк зазначає, що

форма організації навчання – це обмежена в часі та просторі взаємозумовлена діяльність педагога й учня [179, с. 241]. Ю. В. Триус визначає форму навчання як спосіб існування навчального процесу, оболонку для його внутрішньої сутності, логіки і змісту, а також вказує, що форма навчання, передусім, пов'язана з кількістю осіб, які навчаються, роллю учасників навчального процесу, часом і місцем навчання, порядком його здійснення і т.д. [283, с. 231].

У системі освіти впроваджувались та розвивались різноманітні форми навчання. Так за кількістю учасників спільної діяльності виділяють індивідуальну, парну, групову, фронтальну, колективну форми навчання. [283, с. 231]. В системі шкільної освіти основною формою навчання вважається класно-урочна. Класно-урочна форма найчастіше доповнюється індивідуальною та груповою формами навчання.

**Індивідуальна форма**. Елементи індивідуального навчання (індивідуальні консультації, репетиторство та ін.) широко використовуються для доповнення інших навчальних форм. Навчаючи індивідуально, вчитель має змогу найкраще пристосувати навчальний процес до умов навчання та особистості учня. Висока результативність такого навчання не завжди виправдовує значну його затратність. Проте в тих галузях, де учень повинен виявляти особливі фізичні та розумові здібності (навчання музики, спортивні тренування, наукова робота) така форма організації навчання надзвичайно актуальна. Сюди ж варто віднести і навчання програмування, адже багато програмістів передають учням свої знання саме в індивідуальному порядку.

**Групова форма навчання** передбачає об'єднання в групу учнів різного віку та рівня розвитку. Хоча така форма надає можливість одночасно навчати більшу кількість учнів, вона теж недостатньо ефективна.

І все ж під час проведення гуртків та організації інших позаурочних заходів різновікові групи, зазвичай, міцно об'єднують учнів за інтересами. Такі групи згуртовуються за рахунок активного спілкування її членів за межами навчального закладу у неформальній обстановці. Учні об'єднані спільними

пізнавальними цілями активно демонструють свої здібності, проявляють ініціативу у навчанні, допомагають один одному та водночас змагаються. У різновікових групах приклад та наставництво досвідчених учнів може значно сприяти залученню нових членів охочих до навчання. Хоча склад таких груп з часом змінюється, загалом вони можуть існувати десятки років. Окрім спільних інтересів, в учнів з'являються спільні цінності, традиції та навіть історія. Тоді, певною мірою, групова форма навчання переростає в колективну.

Формування різновікових груп для навчання програмування досить часто є виправданим, і таку форму роботи доцільно використовувати на позакласних заняттях з інформатики, для підготовки до олімпіад, конкурсів та турнірів

**Класно-урочна форма навчання**. Ще у сімнадцятому столітті у фундаментальній праці "Велика дидактика" [128] Я. А. Коменський розробив засади та теоретичні обґрунтування класно-урочної форми навчання. Зокрема, він визначив дидактичні основи, наповнюваність класів, розклад, регламент, структуру уроку, діяльність учителя і т. ін. Це стало основою подальшої розбудови теорії і практики уроку. Після появи цієї системи урок став провідною формою навчання у всіх школах світу.

Цікавими та, певною мірою, продуктивними були й інші форми організації навчання: **белл-ланкастерська, дальтон-план, бригадно-лабораторна,** інші [286, с. 348]. Проте ці форми не витримали випробування часом і нині практично не знаходять використання. Урок залишається основною формою навчання у всіх типах навчальних закладів.

У другій половині минулого століття з'явилася плеяда вчителів-новаторів, які творчо розвивали елементи класно-урочної форми навчання: Ш. О. Амонашвілі, І. П. Волков, М. П. Гузик, Є. М. Ільїн, С. М. Лисенкова, В. Ф. Шаталов та багато інших [286]. На цей період припадає чимало цікавих наукових досліджень, спрямованих на удосконалення уроку: Ю. К. Бабанського [15], М. І. Махмутова [168], В. О. Онищука, І. П. Підласого, А. В. Фурмана, М. М. Яковлева та ін.. Викристалізувалися головні ознаки

уроку: спільна діяльність педагога й учня, спрямована на оволодіння знаннями, уміннями й навичками, виховання; керівна роль педагога; постійний склад вихованців одного віку; проведення навчання у спеціальному приміщенні – класній кімнаті; у відповідності до встановленого регламенту часового перебігу навчальної роботи (розкладу занять). Дослідники вивчали урок відносно процесу навчання, як загальну форму його організації та як процес навчання даної сукупності учнів – систему, що творить учнівський колектив. Зокрема М. І. Махмутов дає два різних визначення поняття уроку у відповідності до того чи іншого підходу: 1. «…урок є основна форма руху навчання, обумовлена змістом, принципами та методами навчання, планована і регульована вчителем у певних просторово-часових межах і здійснювана сукупним суб'єктом - вчителем і учнями.»; 2. «Урок – це динамічна і варіативна форма організації процесу цілеспрямованого взаємодії (діяльностей та спілкування) певного складу вчителів (викладачів) і учнів, що включає зміст, форми, методи і засоби навчання та систематично застосовувана (в однакові відрізки часу) для вирішення завдань освіти, розвитку і виховання в процесі навчання» [168, с. 29]

Педагоги продовжують постійно працювати над удосконаленням класно-урочної системи навчання. Проте намагання вчених і педагогів-практиків знайти й змоделювати більш ефективні форми навчання практично не увінчались успіхом. Лише в останні роки, завдяки широкому поширенню та використанню в навчальних цілях інформаційно-комунікаційних технологій, широкого впровадження набуває дистанційна форма навчання.

Нині до форм організації навчання, зазвичай, відносять не лише класно-урочну, а й її доповнення та поєднання з іншими системами навчання: екскурсіями, семінарами, практичними заняттями, екзаменами, заліками, фронтальною, груповою, дистанційною, індивідуальною організацією навчання. З активним впровадженням ІКТ різноманітність використовуваних форм навчання значно збагатилась. Виникли нові підходи до їх класифікації. Зокрема Ю. В. Триус розрізняє традиційні й комп’ютерно-орієнтовані форми

організації навчання [283, с. 258]. Стосовно навчання у профільній школі дослідники наводять такі домінуючі організаційні форми та форми організації навчання інформатики: урок; практикум; екскурсії; групові форми роботи над проектом (навчальним, дослідницьким, інформаційним, творчим, підсумковим тощо), групові форми для участі в інтерактивному уроці; самостійна робота; індивідуальні форми роботи [94]. На нашу думку, для навчання програмування в класах технологічного профілю доцільно використовувати усі існуючі форми організації навчання з врахуванням специфіки такого навчання. По причині виключної новітності та загальної Інтернет-спрямованості навчання програмування мовою C#, на більш детальний розгляд заслуговує використання елементів дистанційної форми організації навчальної діяльності.

### 2.2.3. Використання елементів дистанційного навчання програмування мовою C#

Поняття дистанційної освіти досить широке та включає в себе цілу низку різних форм та способів організації систем навчання. Дистанційне навчання – форма організації і реалізації навчально-виховного процесу, за якою його учасники здійснюють навчальну взаємодію принципово і переважно на відстані, яка не дозволяє і не передбачає безпосередню навчальну взаємодію учасників віч-на-віч [21, с. 9]. Для такої взаємодії на відстані повинен існувати, в якості посередника, засіб зв’язку. Зокрема, означення дистанційного навчання, дане у педагогічному словнику за редакцією С. У. Гончаренка, містить наступний перелік таких засобів: «листування, магнітофонні, аудіо- та відеокасети, комп’ютерні мережі, кабельне чи супутникове телебачення, телефон чи телефакс [286, с. 92]».

За іншим визначенням, дистанційне навчання – новий засіб реалізації процесу навчання, в основу якого покладено використання сучасних інформаційних та телекомунікаційних технологій, що дозволяють навчатись на відстані без безпосереднього, особистого контакту між викладачем і учнем [72].

Отже, завдяки ІКТ дистанційна освіта – це, передусім, окрема система різноманітних сервісів всесвітньої глобальної мережі Інтернет.

Серед усіх можливих форм організації дистанційного навчання будемо розглядати лише ті, що є вільно доступними в мережі Internet і не потребують для їх використання якихось особливих передумов, як-от: соціальної, навчальної, професійної чи іншої приналежності, фінансування. В такому випадку поза нашою увагою залишаться різного роду комерційні, відомчі та інші дистанційні навчальні системи.

Вкажемо на такі складові системи дистанційного навчання:

– Виклад теоретичного матеріалу (у вигляді тексту та графічних ілюстрацій).

– Доповнення теоретичного матеріалу аудіо та відео повідомленнями. Аудіо- та відеозаписи лекцій, відеоілюстрації використання систем та сервісів.

– Системи тестування та інші системи перевірки знань.

– Контекстна та систематизована структура довідково-консультаційних матеріалів.

– Системи вільного обговорення та консультування. Форуми, чати.

– On-line виконавці та on-line системи тестування програмного коду.

Зазвичай матеріали навчання програмування мовою C# в Інтернеті подаються:

– розробниками стандартів мови та середовищ програмування;

– освітніми організаціями та їх представниками;

– загально інформаційними та енциклопедичними ресурсами;

– аматорами та представництвами аматорських груп;

– благодійними організаціями.

**Розробники стандартів мови та середовищ програмування.** Засновник та основний розробник стандартів мови C# один – фірма Microsoft Corporation. Основним розробником середовища з відкритим кодом Sharp Develop є група ентузіастів. Програмний продукт Mono Develop активно підтримується

компанією Novell [343]. Існують і інші, дещо менш відомі середовища, розробки мовою C#.

Хоча розробники масово не впроваджують системи дистанційної освіти, проте на їхніх сайтах можна знайти багато корисних матеріалів, що можна використати для навчання програмування мовою C#. Принаймні, учням доцільно повідомити про основне джерело стандартів мови програмування – C# Language Specification [333]. Тобто, про офіційну специфікацію мови програмування. Також насиченими джерелами відомостей про мову C# є центр розробки на C# [341], блог «Microsoft для преси» [332] та інші.

Для швидкого пошуку практично будь-яких наукових матеріалів зручно користуватися різного роду **загально інформаційними та енциклопедичними ресурсами**. Ці системи також не належать до дистанційної освіти, проте подані там матеріали можна успішно використовувати для отримання потрібних відомостей у мережі Інтернет. Прикладом може слугувати «Вільна енциклопедія – «Вікіпедія [329]».

**Дистанційні навчальні ресурси** представлені освітніми організаціями, зазвичай, розробляються для їхніх власних потреб та з метою популяризації. Щоб привабити потенційних студентів, учнів та інших користувачів, навчальні заклади викладають у вільному доступі певні навчальні матеріали чи взагалі надають змогу користуватися системами дистанційної освіти усім бажаючим. Приблизно за таким принципом діють інші системи дистанційного навчання, створені з благодійною, а не прибутковою метою.

Наприклад, ресурс «Портал знань», що за мотивами свого створення швидше за все є благодійним, містить розгорнуті матеріали дистанційного курсу: «Мова програмування C# [177]». Це, напевно, єдиний україномовний ресурс навчання програмування мовою C#. Вичерпний теоретичний матеріал чітко структурований та взаємоузгоджений. Проте система позбавлена елементів інтерактивності. Тут відсутній зворотний зв'язок з учнями, окрім системи добровільних відгуків, в яких майже ніхто з користувачів, станом на 11

листопада 2011 року, все ще не залишив коментарів. Значно додали б функціональності ресурсу хоч би самі прості тести та опитування для перевірки знань. Стосовно використання «Порталу знань» для навчання програмування учнів загальноосвітніх навчальних закладів, можна сказати, що викладений у ресурсі матеріал не досить придатний для вивчення школярами і більше підходить професіоналам та студентам.

Значний обсяг матеріалів з навчання програмування мовою C# містять російськомовні ресурси дистанційного навчання «Школа программирования [320]» та Інтернет-університет «ИНТУИТ-РУ [22]». Обидві системи раціонально побудовані, добре наповнені теоретичними та практичними матеріалами. До їх складу входять системи on-line тестування та організації зворотного зв'язку. В обох системах проводяться навчання як на безкоштовних, так і на платних курсах вивчення мов програмування. Проте лише відкритий Інтернет-університет «ИНТУИТ-РУ» має безкоштовні курси для початківців, у першу чергу, для учнів середніх шкіл: «Основы программирования на C#».

Швидше за все, існує ще багато систем дистанційного навчання, представлених іншими, відмінними від української та російської мовами. Наприклад, англомовний ресурс «C# station [331]». Він абсолютно відкритий і містить статті та книги на різні теми, щоб допомогти всім бажаючим оволодіти мовою C#. Різноманітність статей надзвичайно широка. Враховуючи певний дефіцит матеріалів для школярів з програмування мовою C#, доцільно давати учням переклад окремих відомостей з ресурсу «C# station», призначених початківцям, наприклад: «The C# Station Tutorial».

**On-line виконавці та on-line системи** тестування програмного коду є надзвичайно перспективним різновидом систем дистанційного навчання програмування. Вже розроблено багато різного роду on-line виконавців (компіляторів, інтерпретаторів), що поєднуються з автоматизованими

системами перевірки виконання завдань. Вони завойовують увагу вчителів та учнів завдяки поєднанню цілої низки можливостей:

– постійний доступ до таких систем, зазвичай можливий будь-де і будь-коли, з будь-якого комп’ютера за умови підключення його до мережі Internet;

– для роботи з цими системами непотрібно ніякого спеціального програмного забезпечення;

– такі системи можуть виконувати програми, що попередньо написані в текстовому редакторі чи середовищі розробки;

– перевірка результатів виконання програм здійснюється на спеціально підготовлених тестах;

– відбувається постійне ведення обліку діяльності користувача та збереження досягнутих ним результатів;

– в більшості таких систем організовано ведення загального рейтингу досягнень користувачів та їх рейтингу в складі певних груп.

Такі системи створюються ще й для проведення змагань зі спортивного програмування. Практично всі вони мають у своєму арсеналі набір простих за розв’язком задач, що робить їх придатними для навчання програмування. Втім далеко не всі такі системи підтримують написання та перевірку завдань мовою програмування C#. Як приклад зручної для навчання програмування можна назвати надзвичайно популярну російськомовну систему «Timus Online Judge, архив задач с проверяющей системой [357]». «Online Judge» перекладається з англійської як «он-лайн суддя». Ця система від початку свого створення приймає розв'язки задач об’єктно-орієнтованими мовами, такими як Java та C#.

Вітчизняною системою тестування програмного коду є популярна не лише на Україні, а й далеко за її межами система автоматичної перевірки задач та проведення олімпіад з інформатики в Інтернеті «E-Olimp [336]». Очолює колектив авторів системи Сергій Станіславович Жуковський [92; 93; 91]. Ресурс «E-Olimp» у травні 2011 року також почав підтримувати перевірку задач, написаних мовою програмування C#.

Для систематизації Інтернет-ресурсів підтримки навчання програмування мовою C# нами розроблено сайт під назвою «Мова C# в мережі Internet», який розміщено за адресою sites.google.com/site/sharpofnet. Матеріали цього сайту можна широко використовувати для організації та здійснення дистанційного навчання програмування мовою C#.

### 2.2.4. Методичне забезпечення навчання програмування мовою C#

Важливою умовою ефективного впровадження та функціонування тієї чи іншої методичної системи навчання є методична підтримка (методичне забезпечення) навчального процесу. Сюди варто віднести наявність, кількість, якість, а для деяких категорій і доступність:

– підручників та збірників задач;

– методик навчання та методичної літератури;

– навчального програмного забезпечення;

– ресурсів глобальної мережі Інтернет (як загально-інформаційних, так й інтерактивних сервісів та служб);

– прикладів такого впровадження в навчання.

Методичне забезпечення навчання програмування мовою C# в класах технологічного профілю включає:

– забезпечення календарним та поурочним плануванням навчального матеріалу, що здійснювалось у відповідності до вимог програми з інформатики для класів технологічного профілю [116, с.65-85]. (див. додаток В.);

– підготовка методичних рекомендацій щодо навчання програмування в класах технологічного профілю на основі мови C# (див. додаток А. та [308]);

– підготовка та тиражування довідника-посібника для учнів «C# коротко» (див. додаток Г.);

– добір та розповсюдження середовищ програмування мовою C#, а також надання рекомендацій щодо їх встановлення та використання;

– підготовка протоколів практичних робіт з теми навчання;

– розробка тестових та контрольних завдань для перевірки знань учнів та подання тестових завдань у форматі, що підтримують наявні в школах програмні системи тестового контролю знань;

– розробка, публікація в Інтернеті та супровід сайтів інформаційно-методичної підтримки.

Для методичної підтримки навчання програмування мовою C# в класах технологічного профілю нами створено сайт https://sites.google.com/site/c4plus/, який водночас використовувався для інформаційної підтримки педагогічного експерименту. Назва сайту: «C# у школі». Загальний девіз: «Протоптати ефективну стежину навчання у світ сучасного програмування». Створено сайт у вересні 2008 року, і з того часу він постійно розвивається та поповнюється новими матеріалами. Сайт має такі сторінки:

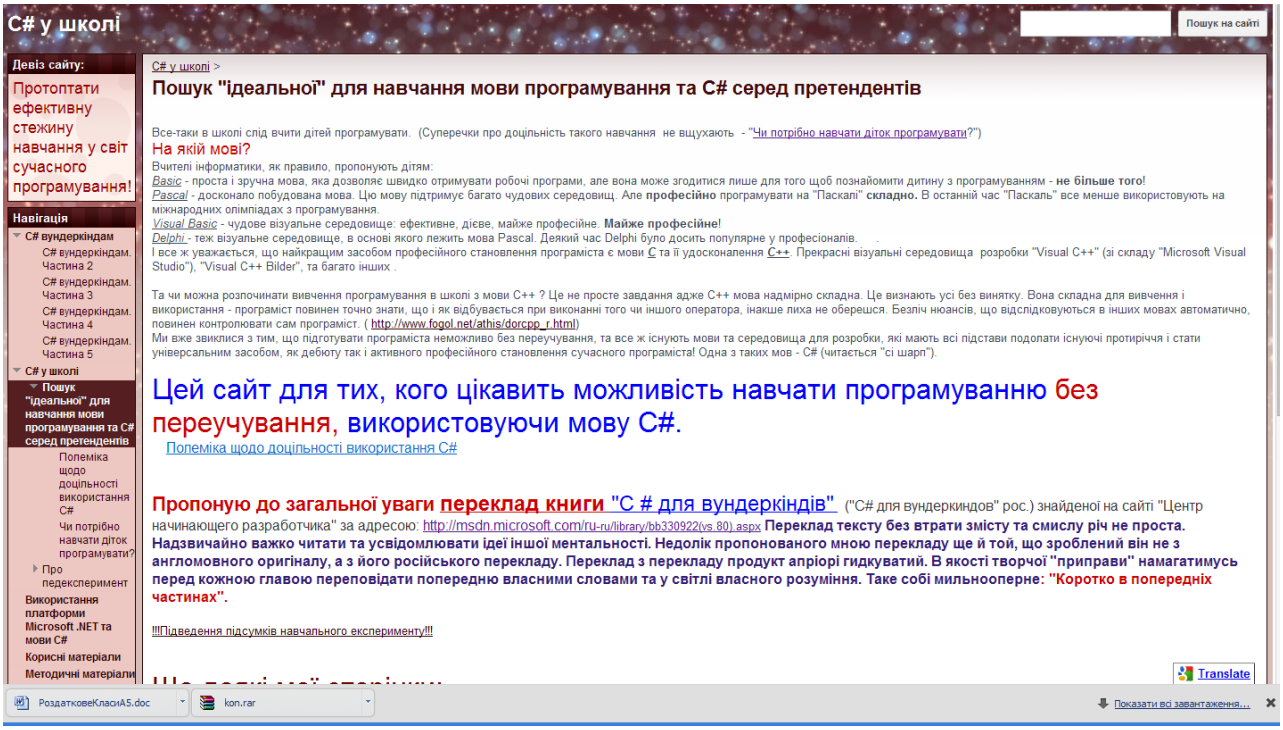



Рис. 2.5 Головна сторінка сайту «C# у школі»

– **Головна сторінка** під заголовком «Пошук «ідеальної» для навчання мови програмування та C# серед претендентів». Тут здійснено загальний опис проблеми добору мови для навчання програмування. Також на цій сторінці знаходяться повідомлення про актуальні матеріали на даному сайті та

посилання на інші сторінки мережі Інтернет, де можна ближче познайомитися з автором сайту та сферами його інтересів.

– «**Використання платформи Microsoft .NET та мови C#».** Тут розміщується стаття із вказаної проблеми.

– «**Корисні матеріали».** Тут можна завантажити програмне забезпечення та інші файли для забезпечення організації навчання програмування мовою C#.

– «**Методичні матеріали».** Ця сторінка містить різнопланові методичні та дидактичні матеріали: календарне планування, плани-конспекти уроків, тести та контрольні завдання, протоколи практичних робіт, зразки написання програм.

– **«C# вундеркіндам»**. Тут знаходяться сторінки, де розміщено авторський переклад на українську мову всіх чотирьох частин книги «C # для вундеркіндів» (C# для вундеркиндов. рос.)

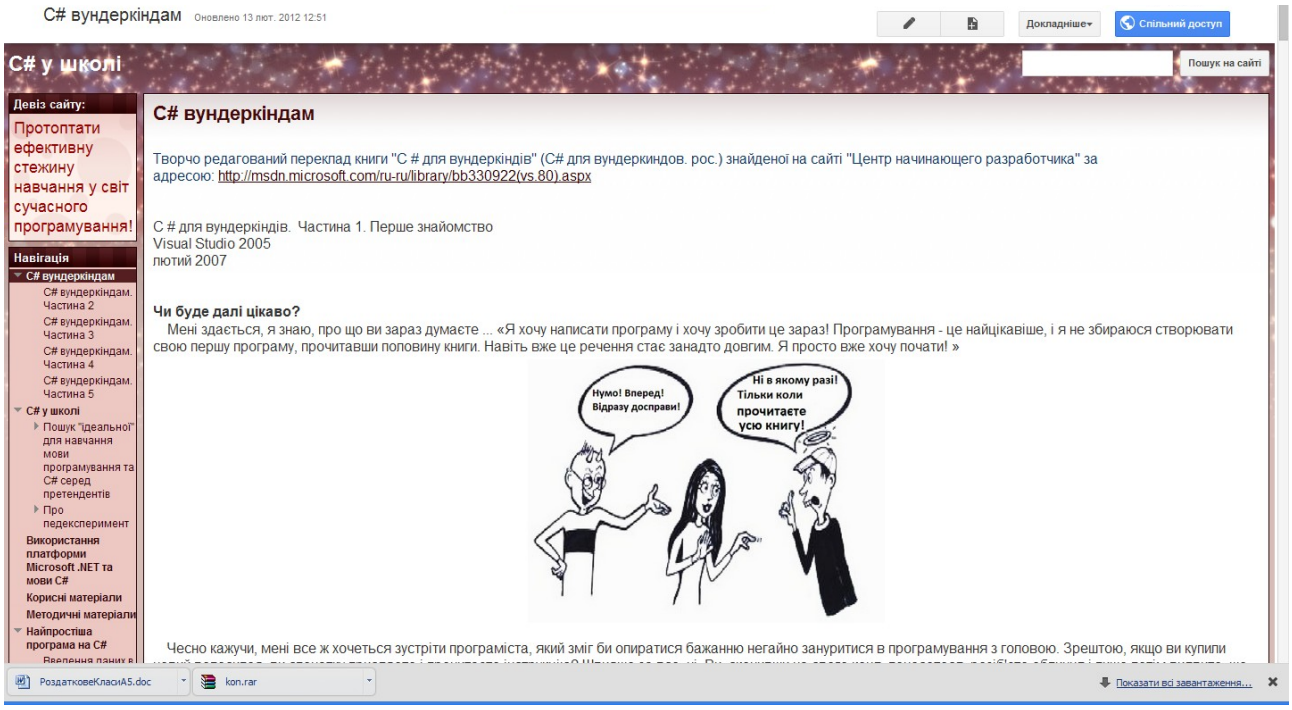



Рис. 2.6 Сторінка з перекладом книги «C # для вундеркіндів».

– «**Використання Notepad++ для програмування мовою C#».** На цій сторінці описано можливість використання одного з простих та досить поширених текстових редакторів Notepad++ для написання програм мовою C#. Також сторінка містить посилання на завантаження текстового редактора Notepad++ та програми nppCompiler. nppCompiler – це додаток до редактора

Notepad++, який дозволяє швидко компілювати код написаної в редакторі програми та запустити результат компіляції на виконання.

– «**Найпростіша програма на C#».** Матеріал присвячений першому практичному знайомству з мовою програмування.

– **«Введення даних у консоль C#».** Тут продовжується матеріал про написання найпростіших програм мовою C#.

– «**Підведення підсумків навчального експерименту**». Сторінка присвячена організації, підтримці та підведенню підсумків педагогічного експерименту, що проводився в рамках даного дослідження.

– «**Посилання на матеріали щодо C#».** Тут розміщено Інтернет-посилання для переходу на матеріали з інших джерел за тематикою сайту.

– «**Підтримка інших мов програмування»,** присвячена здебільшого мові програмування Small Basic.

– «**Про автора**». Сторінка надає можливість ближче познайомитися з автором сайту, зв'язатися з ним електронною поштою.

Крім цього для підтримки навчання програмування мовою C# розроблено сайт під назвою «Мова C# в мережі Internet», який розміщено за адресою sites.google.com/site/sharpofnet. Тут знаходиться достатньо велика колекція посилань, опис та аналіз Інтернет-сторінок, що стосуються C#.

Після завершення педагогічного експерименту обидва сайти продовжують функціонувати практично в тому ж тематичному руслі. Останні оновлення стосуються навчання програмування мовою C# в 11-х класах загальноосвітніх навчальних закладів за програмою академічного рівня [98].

**2.2.4.1. Підтримка діяльності вчителя**

Основною функцією методичного забезпечення є підтримка діяльності вчителя, яка, для забезпечення ефективного ведення навчального процесу на основі взаємодії вчителя і учня, передбачає вирішення трьох основних завдань:

– надання вчителю заздалегідь спланованої, відповідно до вимог програми, системи навчальних матеріалів для проведення всього циклу уроків з можливістю їх модифікації;

– автоматизацію тестування знань учнів, що створює умови для здійснення перевірки знань основних програмістських понять та навиків;

– засоби перевірки правильності ходу розв’язування учнями задач. Для вчителя цей вид підтримки полягає в тому, що за допомогою системи перевіряється правильність ходу розв’язування всієї задачі, здійсненої учнем.

## 2.3. Зміст і методичні особливості вивчення розділу «Основи алгоритмізації та програмування» з курсу інформатики в класах технологічного профілю

Зміст курсу інформатики включає сукупність двох взаємопов'язаних компонентів теоретичного і практичного [182]. Теоретична частина курсу навчання програмування мовою C# спрямована на формування в учнів основ теорії алгоритмів та програм, навичок аналізу і формалізації предметних задач, ознайомлення з такими поняттями, як-от: інформаційна модель, алгоритм, програма, виконавець, компіляція, величина (змінна), тип даних, присвоєння, розгалуження, повторення, масив, підпрограма, клас у програмуванні, об’єкт певного класу, метод об’єкта, рядкові дані, наслідування. Практичний аспект пов'язаний з виробленням навичок: роботи з середовищем програм, написанням програм мовою програмування C#. Вимоги, що ставляться до навчання інформатики в класах технологічного профілю, мають цілий ряд особливостей, націлених на практичну значимість знань та умінь учнів. Поряд із цим використання мови програмування C# накладає додаткові вимоги до організації навчання програмування.

Не можна обійти увагою проблеми використання мови C# на початковому етапі навчання учнів програмування з огляду на те, що ця мова є об’єктно-орієнтованою. Це суттєво ускладнює знайомство з програмуванням, де учень на простих прикладах має осягнути практичну реалізацію базових елементів алгоритмів (лінійні, циклічні, розгалуження), принципів

структурного та об’єктного програмування. Використання мови C# для навчання програмування учнів класів технологічного профілю ЗНЗ вимагає використання теоретично обґрунтованих та педагогічно виважених методик. Доцільно окремо розглянути особливості навчання ООП.

### 2.3.1. Шляхи організації навчання об’єктно-орієнтованого програмування

Детальний опис підходів до навчання програмування, що застосовуються у вищих навчальних закладах США, наводить Л. В. Гришко [64]. Зокрема Гришко називає шість стратегій навчання програмування, що існують за кордоном, а саме у США:

– імперативний підхід;

– об'єктний підхід;

– функціональний підхід;

– підхід з максимальним охопленням матеріалу;

– алгоритмічний підхід;

– апаратний підхід.

Автор детально характеризує підходи, вказуючи на їх особливості:

Імперативний підхід найтрадиційніший. «Якщо викладання ведеться з використанням об'єктно-орієнтованої мови, то на першому етапі увага повинна концентруватися на імперативних аспектах цієї мови, а саме: виразах, структурах управління, процедурах і функціях, а також інших ключових елементах традиційної процедурної моделі. Технології об'єктно-орієнтованого проектування вивчаються на наступному етапі [64].»

Об'єктний підхід. «… із самого початку робиться акцент на принципах об'єктно-орієнтованого проектування і програмування [64].»

«Підхід «з орієнтацією на **функціональне програмування**» характеризується використанням функціональних мов програмування (Lisp, Scheme), що вимагає від учнів достатньо високого рівня абстрактного мислення на більш ранній стадії навчання у порівнянні з використанням традиційних мов програмування [64].»

«Підхід «**з максимальним охопленням**» матеріалу дає найбільш цілісний погляд на програмування. Недоліком такого підходу є розгляд широкого спектру теоретичних питань з програмування на ранній стадії навчання, що ускладнює засвоєння навчального матеріалу [64].»

«Підхід з «**орієнтацією на алгоритми**» при викладанні основних концепцій програмування передбачає використання псевдокоду замість реальної мови програмування. [64].»

«Підхід «**з орієнтацією на апаратну частину**» передбачає навчання основ інформатики і, зокрема, основ програмування, починаючи з машинного рівня. [64].»

Розглянута система підходів до організації навчання ООП в США дещо нагадує ті шляхи, якими вирішується проблема впровадження об'єктно-орієнтованого програмування вітчизняними педагогами. Втім стосовно навчання програмування у загальноосвітніх навчальних закладах І-ІІІ ступенів, то, практично, не застосовуються функціональний підхід, апаратний підхід та підхід з максимальним охопленням матеріалу.

Алгоритмічний підхід донедавна широко використовувався у вітчизняних навчальних закладах. Він передбачав навчання програмування на основі шкільної алгоритмічної мови навіть за відсутності постійного доступу до обчислювальної техніки. Такий підхід отримав надзвичайно широку методичну підтримку науковців. Методика алгоритмічного підходу детально розглянута у працях А. П. Єршова, Н. В. Морзе, М. І. Жалдака, В. Д. Руденка та інших. Зокрема Н. В. Морзе детально описала методику навчання програмування алгоритмічними методами у посібнику для студентів педагогічних вузів за спеціальністю інформатика «Методика інформатики [185]». В останні роки через значне скорочення навчального часу, що відводиться на навчання програмування, алгоритмічна мова усе рідше вивчається в ЗНЗ.

Нині серед вітчизняних педагогів побутує усталена думка, що починати навчати програмування варто з ознайомлення з традиційним структурним та процедурним способом написання комп'ютерних програм. Автор багатьох книг

з навчання програмування Я. М. Глинський, активно підкреслюючи важливість якомога ранішого навчання об’єктно-орієнтованого програмування, пропонує так званий спіралевидний шлях, яким учень повинен дістатися до володіння сучасними методами та засобами написання комп’ютерних програм: «спочатку вивчаються основи мови програмування (типи даних і алгоритмічні конструкції, типові задачі), згодом елементи візуального програмування, а ще згодом елементи ООП [55, с. 38]».

Подібну думку розділяють й інші дослідники [27; 281]. Наприклад С. А. Вернигоренко зазначає: «Деякі вчителі пропонують спочатку ознайомити учнів із структурним та модульно-процедурним підходами у програмуванні (вивчити мову Сі) та лише після цього переходити до вивчення об’єктно-орієнтованого підходу (мова Сі++). Ця точка зору цілком виправдана, вона, зокрема, відображає історичний хід розвитку програмування [34, с. 61]». Даючи вчителю певну свободу у доборі мови та середовища для навчання програмування зміст навчального матеріалу в більшості шкільних програм також подається у вказаній послідовності.

Означуваний Я. М. Глинським «спіралевидний» підхід до навчання програмування є аналогом імперативного підходу [64]. Фактично йдучи таким шляхом, сучасна освіта займається освоєнням об’єктної моделі на досить пізній стадії, коли в учня вже сформувалися певні навички та власний стиль написання програм. У переважній більшості випадків навчання об‘єктно-орієнтованого програмування за такою схемою розпочинається лише у вищих навчальних закладах і навіть не на перших курсах цих установ. Випускники загальноосвітніх навчальних закладів дуже часто так і не отримують на шкільних уроках інформатики навіть найзагальніших уявлень про об’єктно-орієнтовану парадигму програмотворення. Одним із аргументів на користь такого підходу є твердження про те, що ООП є складним для розуміння. Проте ще основоположник ідеї ООП та автор першої повністю об’єктно-орієнтованої мови Алан Кей виявив, що об’єктно-орієнтованій мові Smalltalk легше навчати дітей, ніж професійних програмістів [355]. Пояснював він це тим, що стиль

вирішення завдань, втілений в об'єктно-орієнтованій парадигмі, моделює повсякденне життя. Початківці часто здатні сприйняти ідеї об'єктно-орієнтованого підходу порівняно легко, в той час як люди обізнані в програмуванні, через сформовані раніше уявлення потрапляють у глухий кут.

Інший шлях – шлях паралельного вивчення об’єктно-орієнтованого та традиційного методів програмування. Саме такий підхід реалізовано в посібнику А. С. Лєснєвського «Объектно-ориентированное программирование для начинающих [156]». Це нагадує американський підхід з максимальним охопленням матеріалу. Організувати навчання таким чином доцільно лише за умов досить значної частки інформатики порівняно з іншими предметами, що вивчаються в школі, адже учню доводиться паралельно розглядати досить різні за своєю суттю і способом реалізації методи програмування. Тому навіть у класах з поглибленим навчанням інформатики важко знайти час для паралельного вивчення двох різних технологій написання програм.

За теперішніх умов багато фахівців та ентузіастів прагнуть оволодіти об’єктно-орієнтованим програмуванням вже після того, як добре знають і використовують програмування структурне, процедурне чи, наприклад, функціональне. З огляду на це цікавим є приклад використання мови програмування C# для ознайомлення з об’єктно-орієнтованим програмуванням, який пропонує вчитель інформатики Гаврилівської загальноосвітньої школи Хмельницької області О. П. Пилипчук. Він демонструє приклад створення за допомогою C# нового класу для виконання операцій зі звичайними дробами. Мета його розробки – допомогти здійснити «перехід до об'єктно-орієнтованого програмування для тих, хто вихований на процедурному стилі [218]».

Напевно, окремим підходом до навчання ООП можна вважати оволодіння ним на основі прикладів готових програм. Це складний шлях, до якого, як правило, вдаються ті, хто прагне самостійного навчання програмування. Навіть деякі автори посібників з програмування, уникаючи теоретичних основ, описують лише порядок використання мови програмування до вирішення конкретних завдань. Звичайно, парадигма, стиль та методика розробки

знаходять відображення в текстах коду програм, і програміст призвичаюється до дій в руслі певних принципів та стратегій. За таким принципом написана популярна книга Бена Ватсона «C# 4.0 в примерах [32]». Варто віддати належне автору – текст книги побудовано таким чином, що складність пропонованих прикладів зростає поступово, всі програми логічно пов'язані та обумовлені попередньо розглянутими прикладами. Оскільки уважний читач, що поступово випробує на практиці усі запропоновані Ватсоном програми, може зрозуміти основні переваги дотримуваної автором парадигми програмування, то і такий підхід до навчання заслуговує на схвалення.

Найбільш раціонально розпочати навчання програмування з вивчення загальних особливостей об’єктної моделі (об'єктний підхід). Наприклад, науковці Кримського інженерно-педагогічного університету Ф. С. Ільясова та Ф. В. Шкарбан на самому початку курсу інформатики використовують для ознайомлення студентів з ООП спеціальну навчальну програму. «Курс використовує взаємодіючі натхнення 3D моделювання в дружньому для новачка інтерфейсі, для введення основних понять об'єктно-орієнтованого програмування. Дане середовище програмування можна з успіхом використовувати для реалізації моделей фізичних процесів і створення навчальних, контролюючих, демонстраційних програм [319]». Схожий шлях для ознайомлення з об’єктно-орієнтованою мовою програмування C# використовується в межах дистанційного курсу для початківців відкритого Інтернет-університету інформаційних технологій «ИНТУИТ-РУ [22]». Матеріал курсу «C# для школяра» розпочинається зі знайомства з основними поняттями об’єктної моделі побудови програм на прикладі фантастичної пригоди, в якій реальні предмети природи постають як приклади об’єктів аналогічні тим, що будуть використовуватися під час написання комп’ютерної програми.

Стосовно всіх можливих підходів до навчання програмування в загальноосвітніх закладах саме першочергове навчання принципам ООП найкраще розкриває загальні закономірності розробки сучасного програмного забезпечення. Таке навчання набуває якнайбільшого прикладного спрямування

тоді, коли здійснюється на основі широко використовуваної професійної мови програмування, наприклад мови C#. Та й для навчання програмування мовою C# найбільш раціональним з усіх названих є «об'єктний» підхід.

Попри значні відмінності, навчання програмування на основі різних підходів має багато подібного. Багато спільного, як за значенням так і за практичною реалізацією, є в дотриманні стилю програмування.

### 2.3.2. Стиль програмування у процесі його навчання

Велике значення має дотримання під час навчання єдиного стилю програмування. Під стилем програмування розуміють набір методів, прийомів або правил, що забезпечують створення програмістами правильних, ефективних та зрозумілих для інших програм [280]. Правила стилю програмування можна уподібнити до правил етикету, адже вони також є результатом колективного досвіду. Головне завдання програмного стилю: програма повинна бути зрозумілою тим, хто її використовує та супроводжує. Н. Вірт зазначає, що програма не несе користі, якщо користувач не має змоги розібратися в ній [37].

Протягом нетривалої історії становлення та розвитку програмування виробились певні ознаки загальноприйнятного стилю написання комп'ютерних програм. Зокрема Вірт [37], Тассел [280] та інші дослідники звертають увагу на такі риси якісного програмного стилю: прозора логіка, природні вирази, зрозумілі імена змінних, відповідне форматування, розгорнуті коментарі, відсутність нестандартних конструкцій. Доцільно використовувати певні засоби підвищення зрозумілості програмного коду, а саме: код повинен містити коментарі, зрозумілі ідентифікатори, візуальне форматування. Відомості про вказані засоби узагальнено нами в єдину таблицю (див. табл. 2.2).

Візуальне форматування коду надає йому певної вертикальної структури. У більшості випадків можна уявити вертикальну лінію, проведену через операторні дужки, щодо якої вкладений в дужки код буде дещо праворуч від іншого тексту програми (див рис. 2.7).

*Таблиця 2.2.*

**Окремі засоби забезпечення стилю програмування**

| Засіб та його різновиди | Суть засобу | Особливості використання |
|---|---|---|
| Коментарі:<br>– призначення програми або підпрограми;<br>– призначення змінних;<br>– опис вводу-виведення;<br>– опис методу;<br>– по обсягу пам'яті;<br>– відомості про авторів;<br>– дати внесення змін. | У коді програми певним чином виділяється текст, що не буде виконуватись під час запуску програми. | Коментарі варто залишати вже під час написання програми.<br>Коментарі повинні допомагати розуміти програмний код, а не нагромаджувати його.<br>«Некоментована програма – це найгірша помилка, яку може зробити програміст, а також ознака дилетантського підходу [24]». |
| Ідентифікатори<br>– змінних,<br>– констант,<br>– процедур,<br>– функцій,<br>– модулів,<br>– об'єктів,<br>– класів,<br>– методів,<br>– інші.<br>. | Слугують для звернення до відповідних частин коду чи ділянок розподіленої пам'яті.<br>Вдало поіменований ідентифікатор може замінити використання відповідного коментаря. | Зрозумілість призначення ідентифікаторів забезпечують легко зрозумілими чи загальноприйнятими позначеннями.<br>Існують домовленості написання окремих ідентифікаторів:<br>– розпочинати імена змінних з малої літери, імена класів, методів – з великої, імена констант повністю складаються з великих літер;<br>– використання в ідентифікаторі проміжків (пробілів) не допускається, тому багатослівні імена констант розділяються підкресленням, а для імен змінних і класів кожне слово в багатослівному імені може починатися з великої літери [22] |
| Візуальне форматування коду.<br>Виконується: відступами, пропусками, табуляцією | Певне розміщення операторів, дужок та відступів, що забезпечує загальний вигляд коду зручний для сприйняття. | – Для кожного окремого оператора відводиться окремий рядок.<br>– Бінарні оператори відокремлюються від операндів одним пропуском.<br>– Цілісні вирази не розриваються. Якщо частина виразу переноситься на наступний рядок, то знак бінарної операції повинен лишитися в |

| | | |
|---|---|---|
| | «Показує логічну структуру програми», – Дж. Макконелл [162]. | попередньому рядку.<br>– Операторні дужки та службові слова, якими починається та закінчується той чи інший оператор, варто записувати на одній вертикалі.<br>– Вміст складового оператора, вкладені структури (цикли, розгалуження, складовий оператор) мають розташовуватись з нового рядка і правіше відносно конструкцій, що їх використовують. |

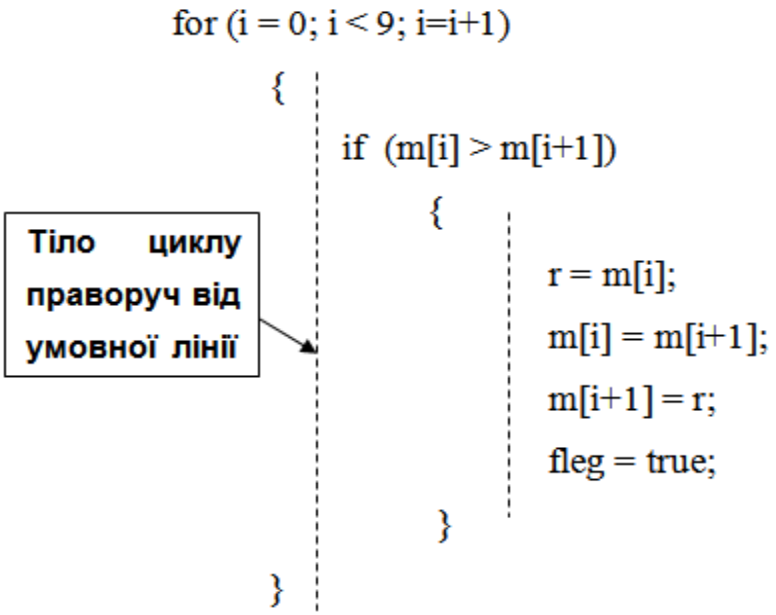



Рис 2.7. Форматування коду відступами

Якщо розробник залишив код без дотримання стилю та недокументованим, то це вказує, що такий код може мати інші серйозні вади, на усунення яких програмісту теж не вистачило терпіння. Крім того, зроблене важко супроводжувати, вносити необхідні зміни, забезпечити в подальшому коректність і стійкість коду. Навчання стилю, зокрема, сприяє навчанню програмування загалом. Враховуючи дотримання учнем стилю форматування, можна точніше оцінити його вміння описувати алгоритми та створювати програми. Навики дотримання єдиного стилю неодмінно допоможуть майбутньому фахівцю в колективній роботі над проектами.

### 2.3.3. Особливості навчання C# в консолі

Найпростішим різновидом комп'ютерних інтерфейсів є, швидше за все, інтерфейс командного рядка, також відомий під назвою консоль. Мова C# добре пристосована для розробки програм консольного виконання. Такий підхід детально описано нами в методичних рекомендаціях для вчителів інформатики щодо навчання програмування мовою C# [308]. Фрагменти цих рекомендацій наводяться в додатку А до даного дослідження.

Найпершою практичною реалізацією програми мовою C# є написання найпростішого коду, що його можна безпомилково відкомпілювати. Умовно назвемо такий програмний код **найпростішою програмою**. В мові C#, як і в багатьох інших мовах програмування, за допомогою найпростішої програми можна пояснити учням особливості використання **роздільників** в алфавіті мови програмування та **операторних дужок**. Загальні правила розміщення цих елементів власне і визначають стиль програмування. На цьому ж етапі відбувається навчання коментуванню тексту [308 с.13].

Наступним етапом є знайомство з **оператором виведення** та програмою типу «Hello World!» [308 с.14]. Даний матеріал унікальний тим, що дозволяє наочно пояснити учням основні **типи даних** мови C#. Для цього достатньо використати в якості аргументу оператора виведення різнотипові величини. А використання в якості аргументів тих чи інших виразів дозволяє ознайомити учнів практично з усіма операціями допустимими над даними того чи іншого типу. При цьому відмінності між операціями над цілочиселними даними та даними дійного типу чисел доцільно розглянути детально. Ці відмінності, фактично, є проявом однієї з визначальних рис ООП – **поліморфізму**. Той випадок, коли в якості аргументів виразів оператора виведення виступають різнотипові дані, також дозволяє розглянути деякі **перетворення типів**.

Перейшовши до **оголошенням змінних** [308 с.16], корисно вказати на те, що за умов об'єктного підходу усякий тип даних – це **клас**, а конкретні дані цього типу – це **об'єкти** даного класу. Тут таки є підстави ознайомити з

**наслідуванням** властивостей типів як прикладом наслідування класів. Також можна розглянути поняття **інкапсуляція,** навівши приклади модифікаторів доступу, що використовуються при оголошенні змінних. Отже, розглядаючи типи даних мови C#, можна продемонструвати основні характерні риси ООП: наслідування, поліморфізм та інкапсуляцію.

Після наступного вивчення **присвоєння** розглядається **оператор введення** та особливості його використання: зупинка програми для введення даних до натискання клавіші «Enter»; дві форми оператора введення («Read» та «ReadLine»); необхідність перетворення типів командою Convert для введення числових даних [308 с.16].

Навчання **розгалуження** та **циклів** доцільно супроводжувати розглядом блок-схем [308 с.17]. Зображення блок-схем основних видів розгалуження та циклів, що містять умову подано на малюнках рис. 2.8. – 2.11.

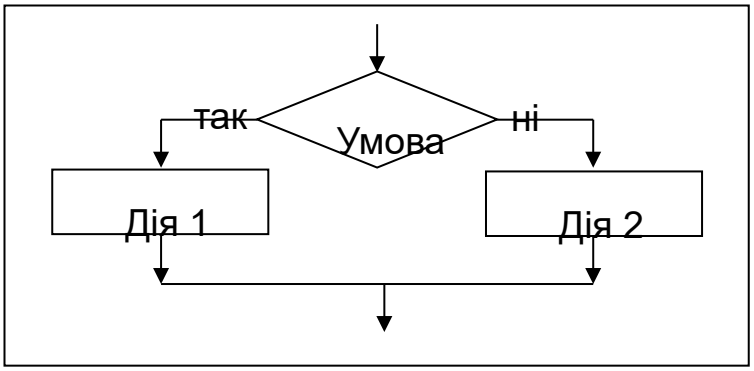



Рис 2.8. Схематичне зображення повного розгалуження

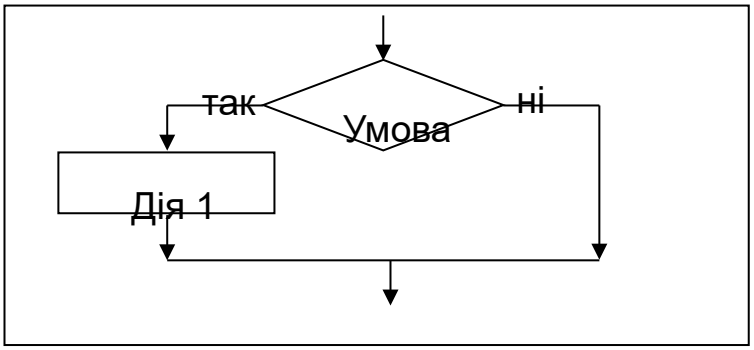



Рис 2.9. Схематичне зображення неповного розгалуження

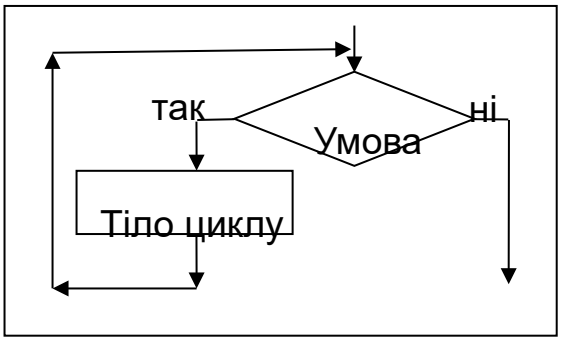



Рис 2.10. Схематичне зображення циклу з передумовою

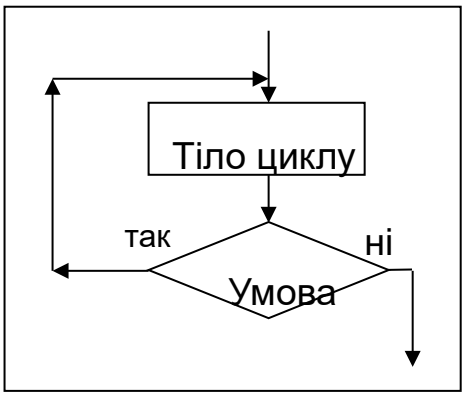



Рис 2.11. Схематичне зображення циклу з післяумовою

Важливо детально розглянути окремі складові розгалужень: умову; дію (серію команд), що виконується, якщо умова істинна; дію (серію команд), що виконується, якщо умова хибна. Складовими циклів є: тіло циклу (команди, що повторюються) та умова припинення повторення. Демонструючи відмінність у використанні циклів з перед- та післяумовою, доцільно вказати на можливість практично повної взаємозаміни цих алгоритмічних конструкцій.

Дещо окремо розглядаються цикли з параметром. Поняття параметру має цілий ряд аспектів: це обов'язково величина так званого перелічуваного типу; параметр має початкове та кінцеве значення, крок. Цикли з параметром широко використовуються для обробки табличних величин – **масивів** [308 с.19]**.**

Вивчаючи використання масивів, можна ознайомити учнів з існуванням більш складних, в даному випадку структурованих, типів даних. Важливими поняттями даного етапу є: **елемент масиву**, його **індекс**, **базовий тип даних.** Написання програм обробки масивів дозволяє закріпити на практиці правила використання операторних дужок, операторів введення та виведення, розгалужень та циклів, особливості використання цілочисельних та дійсних

числових даних, дотримання стилю програмування. Основними алгоритмами роботи з табличними величинами, що можна розглянути, є: введення даних до масиву, виведення значень масиву, визначення суми елементів масиву, знаходження найбільшого (найменшого) елементу масиву, пошуку елементів або їх кількості за певними умовами, сортування елементів масивів.

**Сортування масивів** є, напевно, найскладнішою категорією класичних задач теорії алгоритмів, що розглядаються у шкільному курсі інформатики. Навчаючи сортування масивів, можна продемонструвати подальший розвиток ідей основних алгоритмів їх обробки, використання вкладених циклів, обмін значень комірок пам'яті. Даний матеріал дозволяє розглянути існування великої кількості алгоритмів виконання одного і того ж завдання, пошук більш ефективних з них.

Обсяг часу, що відводиться на уроках інформатики для навчання програмування, постійно зменшується. За таких умов вчитель повинен подбати про підвищення доступності навчального матеріалу. Одним із способів підвищення доступності навчального матеріалу є його часткове спрощення та скорочення. Для цього нами пропонується свідомо уникати використання оператора інкременту, додаткових модулів, бібліотек функцій, класів [308 с.24].

### 2.3.4. Описове знайомство з основами об'єктно-орієнтованого програмування

Розпочати знайомство учнів з принципами об'єктно-орієнтованого програмування можна шляхом аналізу та уточнення значення поняття «об'єкт». Цей термін надзвичайно широко використовується у навчальному матеріалі різних дисциплін, у повідомленнях засобів масової інформації, різноманітній літературі. Це слово частково знайоме навіть учням молодших класів. Розпочати розгляд терміну «об'єкт» можна навівши учням цілий ряд тверджень, що стосуються цього поняття:

1. Об'єкти завжди можна чітко відмежувати один від одного.
2. Об'єкти існують незалежно від того, чи знає про них хтось, незалежно

від людської свідомості.

3. Об’єкти мають певні властивості.
4. Об’єкти можна описувати та розповідати про них.
5. Існують об’єкти зі схожими властивостями.
6. Об’єкти можуть взаємодіяти один з одним.
7. Деякі об’єкти можуть виконувати дії.
8. З об’єктами можуть відбуватися якісь події.

Більшість старшокласників можуть легко пояснити практично кожне з наведених висловів. Проте для деяких учнів середнього, особливо молодшого, шкільного віку це може бути доволі непростим завданням. Полегшити його можна, як уже зазначалось, використовуючи інтерактивні методи навчання, прийоми розвитку критичного мислення (див. додаток Е). Зокрема, учні, працюючи в групах, з цікавістю знаходять підтвердження або ж заперечення тих чи інших тверджень, висловлюють своє власне розуміння понять та явищ. Швидше за все вчитель не повинен ставити за мету дати дітям точне визначення поняття об’єкт. Врешті решт це не легко, адже таке визначення в різних галузях формулюється досить по-різному. Проте дуже добре, коли учні прийдуть до певного висновку стосовно того, для чого люди найчастіше використовують слово «об’єкт». Висновок може бути, наприклад, таким: «Об’єкт – це умовне позначення тієї частини оточуючого світу, яку людина розглядає, як одне ціле».

Надзвичайно корисно пояснити учням те, що усякий об’єкт має певні властивості та іноді проявляє певні «можливості». В даному випадку «властивість» розглядається, як аналогія поняття «поле» об’єкта у програмуванні. Так само «можливість» – це, практично, те ж, що й «метод» об’єкта. Скажімо, об’єкт «пес» може характеризуватися такими властивостями (полями), як колір шерсті, кількість та довжина ніг, кількість зубів. Для цього об’єкта притаманні можливості (методи): гавкати, втікати, кусати. Крім того з тими чи іншими об’єктами можуть відбуватися певні події. Наприклад, з

об’єктом «пес» може трапитися подія «зголодніти». Самі об’єкти також можуть вчинити певну дію. Всі події та дії можуть відбуватися з об’єктами в межах їх властивостей та можливостей (полів та методів). Скажімо, об’єкт «пес» у відповідь на подію «зголодніти» може проявити дію – метод «їсти», до складу якого входить метод «кусати».

Наступним завданням на початковому етапі знайомства з основними поняттями ООП є проілюструвати учням класифікацію об’єктів оточуючого світу. Така класифікація притаманна практично всім наукам. Біологи класифікують живі організми, хіміки класифікують сполуки, хімічні елементи, реакції та інше. Багато певних систем класифікації є у фізиці, географії, математиці, інформатиці. Швидше за все добре усвідомивши різного роду класифікацію об’єктів, учень краще зрозуміє, яку роль відіграє службове слово «class» в тексті конкретних комп’ютерних програми.

Ввівши поняття об’єкту та класу, можна дати загальне пояснення, що таке «наслідування», «поліморфізм», «інкапсуляція». Таке поняття об’єктно-орієнтованого програмування, як «наслідування», учні сприймають порівняно легко. Спираючись на добре відоме біологічне значення наслідування живих організмів, можна запропонувати учням визначити для вказаних об’єктів оточуючого світу ті об’єкти, що є спадкоємцями та пращурами певних властивостей та методів. Це можна робити як на основі прикладів з живої, так і прикладів з неживої природи. Слід пам’ятати про те, що в подальшому учні повинні сприймати наслідування, як раціональний спосіб повторного використання коду та забезпечення певної структурної організації об’єктів у комп’ютерних програмах.

Набагато складніше дати «безкомп’ютерне» пояснення таких фундаментальних понять ООП, як інкапсуляція та поліморфізм. Умовно кажучи, в живій природі та решті оточуючого світу об’єкти «інкапсулюють» від зовнішніх впливів та поліморфно проявляють свої «методи». Все ж іноді учитель повинен самостійно вирішувати потрібно чи ні негайно давати

пояснення деяких понять, якщо таке пояснення може переобтяжити навчальний матеріал. Наводячи конкретні приклади, варто прагнути не лише негайно з'ясувати на аналогіях значення нового для учнів терміну, а й дбати про загальну якість такого пояснення. До того ж не можна беззастережно розглядати досить складні терміни програмування виключно за їх порівнянням з властивостями оточуючого світу. Слід враховувати, що реальне значення «програмістських» термінів має певні відмінності від того, що можна з'ясувати, використовуючи лише аналогії та порівняння.

### 2.3.5. Пояснення принципів ООП на прикладах роботи програмного забезпечення в середовищі Windows

Принципи об'єктно-орієнтованого програмування також можна розглянути під час пояснення роботи операційної системи та прикладних програм, що в ній виконуються. Таке пояснення використовується нами в посібнику-довіднику для учнів (див. додаток Д) та в матеріалах до уроків (див. додаток Є). Наведемо основні його ідеї.

Якщо в операційній системі запустити на виконання ту чи іншу програму, то це можна розглянути як появу об'єкта певного класу. В більшості випадків сучасні багатозадачні операційні системи дозволяють запускати на виконання декілька екземплярів однієї і тієї ж програми. Наприклад на робочому столі Windows можна одночасно розмістити кілька вікон одного і того ж графічного чи текстового редактора. В такому разі всі одночасно запущені на виконання об'єкти будуть, фактично, екземплярами одного й того ж класу. Одночасно відкриті вікна різних програм ілюструють співіснування об'єктів різних класів. Об'єкти одних і тих же чи різних класів можна демонструвати учням в процесі навчання прикладних програмам, елементів їх інтерфейсу.

На прикладі функціонування операційної системи та програм в її середовищі вчитель має можливість проілюструвати учням практично всі основні положення ООП. Наприклад той факт, що клас характеризується певними *членами* (полями та методами) можна розглянути порівнюючи ті чи

інші властивості різних програм. Характеристики інтерфейсу (розмір вікна, його колір, інші) є прикладами *полів*. Виконувані програмами завдання (відтворення звуку, друк, сортування тексту за алфавітом і т.д.) – це *методи*. Для пояснення *наслідування* можна порівнювати елементи різного виду вікон Windows, адже в багатьох випадках вони є нащадками одного й того ж класу Form. *Поліморфізм* можна продемонструвати на прикладі того як одна і таж функція певної програми по-різному реалізовується за різних умов. Програвач по-різному відтворює звукові та відео медіа файли. В середовищі навчальної програми Gran 2D створення симетричної точки виконується по-різному в залежності від обраного типу симетрії. Прикладом певної *інкапсуляції* програмних об'єктів, що одночасно виконуються в середовищі операційної системи, є те, що вони не заважають один одному.

Загальні пояснення основних понять полегшують навчання того, як створюються класи та описуються їх екземпляри вже безпосередньо мовою C#.

**2.3.6. Особливості навчання ООП мовою C#**

Найбільш загальним правилом побудови програм на C# є те, що усяка програма є класом чи сукупністю окремих класів. Кожен окремий клас має своє ім'я (ще кажуть ідентифікатор) та, умовно кажучи, вміст. Точніший термін «члени класу» (замість «вміст») на початковому етапі потребує певного пояснення. При оголошенні (ще кажуть створенні, описуванні) класу ім'я задається після службового слова class, а члени класу задаються відразу після його імені у фігурних дужках. Схематично це позначають так: class *<Ім'я класу>* {*вміст класу*}. Ім'я класу задається за тими ж правилами, що й імена будь-яких ідентифікаторів мови C#. Ідентифікатори є важливими елементами алфавіту мови програмування і їм варто присвятити достатньо часу.

Навіть розглядаючи загальний приклад оголошення класу слід дотримуватись стилю програмування, тобто ім'я класу бажано починати з великої літери, а операторні дужки розміщувати в окремих рядках.

class *Dog*

```
{
}
```

Щоб правильно пояснити, як задати поля того чи іншого класу, слід використати поняття типів даних. Але C# типи даних в C# також можна розглядати як певні класи. Отже, існує замкнутий ланцюжок взаємозв’язаних понять. Для того, щоб пояснити учням порядок створення класів, доцільно, щоб вони знали, що таке типи даних. А для того, щоб пояснити, як мовою C# реалізовується типізація, корисно пояснити, як цією мовою описується класи та об’єкти в них. Загалом поняття типів у мові C# досить складне та багатогранне і аж ніяк не може бути повністю розкритим в процесі навчання основних курсів інформатики в ЗНЗ. Скажімо, самі ж таки класи є певними типами. Точніше посилальними типами. У змінній об'єкта класу зберігається посилання на адресу об'єкта в керованій купі. Навіть такий фрагмент теорії може дуже і дуже обтяжити навчальний матеріал уроку інформатики в загальноосвітньому навчальному закладі.

Щодо черговості розгляду вказаних понять на практиці дотримуються наступних підходів:

1. Традиційно до навчання ООП переходять після знайомства з процедурним програмуванням. Тобто до опису класів учні вже знають, що таке типи даних. Такий підхід має певні недоліки, про що вже говорилося в нашому дослідженні. При цьому деякі вчителі вдаються до значних спрощень у поясненні нових понять та термінів. Наприклад Я. М. Глинський пропонує розглядати поняття класу та об’єкту на основі понять тип даних та змінна: «Терміни клас і тип вважаються синонімами. Екземпляром певного класу є об'єкт (синонім змінна). Об'єкти характеризуються властивостями (атрибутами), що є аналогами класичних змінних стандартних (примітивних) типів. Конкретні стани об'єктів задаються значеннями атрибутів. Об'єкти можуть володіти певною функційністю (аналогом є класичні функції). Функційність класів зосереджена у методах (синонім функції)» [55].

2. Якщо пріоритет надається об’єктному підходу, то навчання програмування розпочинають зі знайомства з поняттям клас. Учням, як правило, повідомляють обмежені відомості про певні найбільш використовувані типи даних відразу, коли вперше описується вміст, члени якогось класу. Після цього розглядається оголошення екземплярів класу (об’єктів), в яких поля тих чи інших типів отримуватимуть конкретні значення.

```
class Dog
{
        int numberOfLegs; // кількість ніг.
}
```

У такому випадку для полегшення знайомства з типами даних та використання їх для побудови найпростіших класів застосовується якомога менша кількість різновидів таких типів. Як правило, розглядається лише по одному з цілочисельних та дійсних типів (можливо, *int* та *float),* а також логічний (*bool*) та рядковий (*string*) типи. Особливості запису, межі допустимих значень, інші характеристики типів даних детально не висвітлюються, а вивчаються пізніше чи пояснюються в процесі використання.

3. Та особливість мови програмування C#, що типи даних – це, з певним припущенням, також класи, дає можливість повноцінно реалізувати об’єктний підхід до навчання типів даних. Оскільки поняття класу та об’єкту є первинним то, на нашу думку, слід відразу знайомити учнів з типами даних як з класами. Можна повідомити, що типи даних – це класи, які описано в бібліотеці класів середовища виконання коду. До цієї бібліотеки, а отже до описаних в ній класів, можна завжди звертатися при написанні програм.

Щоб мати можливість для такого підходу, знайомство з типами даних проводять після того, коли розглянуто загальну схему оголошення екземплярів класів, тобто об’єктів. Нажаль пояснювати загальну схему створення об’єкту певного класу доводиться на прикладі фрагментів коду виконання (компіляція) яких, до наповнення даними та методами, неможлива. Але саме такий підхід і вважається нами найбільш доцільним.

### 2.3.6.1. Загальна схема опису об’єктів

Отже, наступним кроком після знайомства із загальною схемою оголошення класу буде доповнення її описом екземплярів, об’єктів оголошеного класу. А для цього слід розглянути оператор *new*. Загалом оператор *new* призначений для виділення динамічної пам’яті. За допомогою оператора *new* створюються екземпляри класів. Для оголошення об'єкта довільного типу використовується наступна конструкція: *<ім’я_класу> <ім’я_об’єкту>; <ім’я_об’єкту>* = new *<ім’я_класу ()>;*. Якщо ці лексеми ідуть в тексті програми поряд, то їх можна об’єднати в одну: *<ім’я_класу> <ім’я_об’єкту>* = new *<ім’я_класу ()>;*.Слід зазначити, що з виділенням пам’яті пов’язано оголошення змінних, але оператор *new* використовувати для цього не обов’язково. Отже, спочатку можна ознайомити учнів з порядком виділенням пам’яті та ідентифікації нових об’єктів, а вже потім навчати оголошувати величини певних типів даних за такою ж схемою.

Наприклад, об’єкт Sirko класу Dog створюється так:

*Dog sirko;*

*sirko* = new *Dog ();*.

Або так (дещо коротше):

*Dog sirko* = new *Dog ();*.

Для полегшення сприйняття коду програми прийнято ім'я класу писати з великої прописної літери. Імена об'єктів частіше починаються з маленької рядкової букви. В наведеному нижче прикладі певне протиріччя з українським правописом дозволить звернути на це увагу учнів: *Dog* пишеться нами з великої, бо це ім'я класу, а *sirko* – з маленької, бо це об'єкт.

```
class Dog
{
}
   {
      Dog sirco;
```

```
        sirco = new Dog();
    }
```

Дуже добре, якщо учні запам’ятають та усвідомлять структуру класів, схеми їх оголошення, наслідування, створення об’єктів того чи іншого класу ще до написання програм, що мають певне практичне значення.

Як уже зазначалось, обсяг початкових відомостей про типи даних може бути невеликим. Головне, щоб їх вистачило, аби правильно пояснити найпростіші приклади оголошення класів та їх екземплярів. Оголошувати поля класів та просто певні змінні можна так, як оголошуються об’єкти, екземпляри користувацьких класів. Наприклад, якщо зазвичай оголошення типу даних *int* виглядає так: *int a*, то оголошення об'єкту класу *Int32* буде таким: *Int32 a = new Int32().* Що в принципі одне і теж. Тобто поле певного типу вперше краще описати так:

```
class Dog
{
    Int32 numberOfLegs = new Int32(); // кількість ніг у собаки.
}
```

замість:

```
class Dog
{
    int numberOfLegs; // кількість ніг у собаки.
}
```

Після розгляду оголошення змінних вводиться поняття присвоєння. Для величини «*a*» типу *int* (а це те ж саме, що для об’єкту «*a*» класу *Int32*) присвоєння записується так: *a = 5*;

**2.3.6.2. Метод Main та модифікатори доступу**

Розгляд практичного розміщення певного класу та оголошуваних об’єктів в тексті програми неможливий без знайомства з використанням в програмі методу *Main*. Крім того, опис цього методу обов’язково супроводжується модифікаторами *static* та *void*. Постає ще одна дилема визначення послідовності подання навчального матеріалу. Традиційно перше використання методу *Main*

здійснюється в процесі знайомства з найпростішою програмою, що безпомилково компілюється. Призначення рядка *static void Main()* детально не пояснюється. Вчителі обмежуються загальними фразами на зразок таких: «Це точка входження в програму…», «Це основний метод, з якого розпочинається виконання програми і без якого воно розпочатися не може», та інших.

Модифікатори надають класу, його полю або методу певні властивості. Серед них є ті, що забезпечують інкапсуляцію. Такі модифікатори називають модифікаторами доступу. Для початку навчання мови C# доцільно ознайомлювати з трьома основними модифікаторами доступу: *public*, *protected* та *private*. Хоча в кожному конкретному випадку використовувані класи та їх члени взагалі можуть бути по відношенню до іншого класу лише в двох станах – або *закритими*, або *відкритими*.

Модифікатор *public* вказує, що до даного класу, поля чи методу можна звертатися в будь-якій частині програми. Призначення *protected* можна описати знаючи, як реалізовується мовою C# наслідування. *protected* розмежовує доступ для всіх «рідних» (тобто похідних або батьківських) класів. Найбільш захищеним є код, позначений модифікатором доступу *private*. Він позначає методи або поля доступні лише в межах даного класу. Відповідно клас *Dog* можна описувати:

*public* class *Dog*
{
},
*protected* class *Dog*
{
}
або ж
*private* class *Dog*
{
}

Якщо модифікатор не вказано, то клас, метод чи поле вважаються *private.*

Хоча службове слово *static* – це не модифікатор доступу, а *void* і взагалі важко назвати модифікатором, знайомити учнів з методом *Main* доцільно попередньо розглянувши загальну схему використання модифікаторів доступу. При цьому модифікатор *static* пояснюється як такий, що визначає метод статичним. Основною особливістю статичного методу є те, що звертатися до нього можна, не створюючи екземпляру класу, членом якого цей метод є. Службове слово *void* вказує, що метод не повертає ніяких значень.

Лише добре підготувавши учнів теоретично, можна впевнено переходити до написання програм, структура яких не буде суперечити об’єктному підходу до програмування. Ось зразок простої програми, на прикладі якої можна пояснити використання простого користувацького класу та його об’єкту.

```
using System;
   class Dog
    {
          public String name;
    }
   class MyProgram
   {
     static void Main()
     {
       Dog sirco;
       sirco = new Dog();
       sirco.name = "Сірко";
       Console.WriteLine(sirco.name);
       Console.ReadLine();
     }
   }
```

## Висновки до розділу 2

У даному розділі коротко обґрунтовано вибір напряму досліджень та подано загальну методику його проведення. Описуються провідна ідея, концепція та гіпотеза, що розглядаються. Подаються основні тенденції, закономірності, методологічна основа даного дослідження. Дається загальний опис етапів науково-педагогічного пошуку, огляд методів вирішення задач дослідження та їх порівняльні оцінки.

Також у даному розділі розглянуто методичні засади навчання програмування учнів у класах технологічного профілю з використанням мови C#. При детальному розгляді питання на передній план постала відповідність навчальної програми парадигмі написання комп'ютерних програм, якій відповідає обрана мова програмування. Імперативність мови програмування C# дозволяє повноцінно використовувати її для написання найпростіших програм консольного виконання, що відповідають парадигмі процедурного програмування. Додатково використання мови програмування C# забезпечує оволодіння учнями елементами об'єктно-орієнтованого програмування. Об'єктно-орієнтоване програмування, як найбільш розповсюджений стиль написання комп'ютерних програм, може задовольнити вимоги переважної більшості сучасних розробників програмного забезпечення.

Вдалося встановити основні компоненти методичної системи навчання програмування учнів класів технологічного профілю з використанням мови C#. В результаті дослідження визначено зміст і методичні особливості вивчення розділу «Основи алгоритмізації та програмування» з курсу інформатики в класах технологічного профілю. Зміст навчального матеріалу містить у своїй основі програму з інформатики для класів технологічного профілю.

Створена методична система навчання програмування мовою C# в класах технологічного профілю має такі особливості:

– підтримує мультипарадигмальний підхід до навчання програмування;

– сприяє впровадженню навчання об'єктно-орієнтованого програмування;

– більш якісно інтегрує тему про алгоритми та програми з іншими темами курсу інформатики;

– якісніше пов'язує навчання програмування з іншими шкільними предметами;

– є базою для розвитку дитячої науково-технічної творчості у сфері комп’ютерних інформаційних технологій.

Основні результати другого розділу опубліковані нами в роботах [302; 303; 304; 305; 307; 310; 311; 312; 314; 315].

# РОЗДІЛ 3.
# ОРГАНІЗАЦІЯ ТА РЕЗУЛЬТАТИ ПЕДАГОГІЧНОГО ЕКСПЕРИМЕНТУ

У третьому розділі теоретично обґрунтовано та описано проведення педагогічного експерименту. Педагогічний експеримент повинен підтвердити чи спростувати гіпотезу, що лежить в основі даного дослідження. Також у розділі вивчається проблема перевірки ефективності впровадження новітніх технологій та засобів розробки програмного забезпечення під час навчання програмування. З'ясовується необхідність застосування спеціальних методик такого дослідження, що виключають вплив особливостей мови та використовуваних парадигм програмування на результати дослідження. Розглянуто один з методів проективних технологій психологічних досліджень, а саме: метод часткового семантичного диференціалу для порівняння ефективності навчання програмування. З використанням розробленого різновиду методу семантичного диференціалу проведено порівняння ефективності навчання програмування мовою C# та іншими мовами програмування в класах технологічного профілю загальноосвітніх навчальних закладів.

Результати описаного експерименту склали основу висновків щодо справедливості гіпотези, покладеної в основу нашого дослідження.

## 3.1. Дослідно-експериментальне обґрунтування методики навчання програмування на основі використання мови C# в класах технологічного профілю ЗНЗ

Впровадження навчання програмування на основі використання мови C# у класах технологічного профілю загальноосвітніх навчальних закладів потребує експериментальної перевірки. Аналіз методичної літератури з організації педагогічного дослідження [114, 141, 195, 262, 294] свідчить, що дослідно-експериментальна робота відіграє значну роль у науково-педагогічних дослідженнях, тому що емпіричні дані, отримані в результаті її проведення, є підставою для підтвердження висунутої гіпотези і завдань дослідження. Багато

сучасних дослідників вважають, що розвиток методів дослідження є необхідною умовою вдосконалення педагогічної теорії [136; 294, с. 26]. Саме в цьому полягає велике загальнотеоретичне значення педагогічного експерименту.

Суть експерименту (від лат. Experimentum – проба, досвід), як методу дослідження, полягає в спеціальній організації педагогічної діяльності вчителів і учнів з метою перевірки й обґрунтування наперед розроблених теоретичних припущень або гіпотез. Коли гіпотеза чи припущення знаходить своє підтвердження на практиці, дослідник робить відповідні теоретичні узагальнення і висновки [294,с. 28].

Педагогічний експеримент повинен перевірити відповідність отриманих результатів гіпотезі, що лежать в основі даного дослідження. А саме: **Гіпотеза дослідження** ґрунтується на припущенні, що за умов використання науково-обґрунтованої та розробленої авторської методики навчання програмування учнів у класах технологічного профілю дозволить:

– забезпечити теоретичне та практичне знайомство з елементами ООП учнів класів технологічного профілю під час навчання програмування;

– покращити узгодженість, внутріпредметні зв’язки, змістову єдність розділу «Основи алгоритмізації та програмування» з іншими розділами шкільної інформатики;

– покращити узгодженість, міжпредметні зв’язки, змістову єдність інформатики з іншими предметами, що вивчаються в класах технологічного профілю загальноосвітніх навчальних закладів.

Для перевірки висунутої гіпотези дослідження був організований педагогічний експеримент, який проводився протягом 2008 – 2011 рр. На базі Миропільської гімназії, Миропільської загальноосвітньої школи І-ІІІ ступенів № 2, Камінської загальноосвітньої школи І-ІІІ ступенів, Гордіївської загальноосвітньої школи І-ІІІ ступенів, Биківської загальноосвітньої школи І-ІІІ ступенів, Романівського ліцею. А також на базі чотирьох навчальних закладів

інших районів Житомирської області: загальноосвітньої школи І-ІІІ ступенів № 7 м. Житомира, загальноосвітньої школи І-ІІІ ступенів № 21 м. Житомира, загальноосвітньої школи І-ІІІ ступенів № 3 м. Олевська та Озернянської загальноосвітньої школи І-ІІІ ступенів Житомирського району.

Мета педагогічного експерименту полягала в емпіричному підтвердженні концепції дослідження та у визначенні рівня ефективності розробленої методичної системи навчання програмування учнів класів технологічного профілю на основі використання мови С#.

Основними завданнями експерименту були:

1. З’ясування необхідності впровадження мови С# для навчання програмування учнів, що навчаються в класах технологічного профілю загальноосвітніх навчальних закладів.

2. З’ясування необхідності створення методичної системи навчання програмування в класах технологічного профілю на основі використання мови програмування С#.

3. Розробка та впровадження компонентів методичної системи навчання програмування в класах технологічного профілю на основі використання мови С#.;

4. Перевірка ефективності розробленої методики навчання програмування в класах технологічного профілю на основі мови програмування С#.

5. Аналіз результатів експериментальної перевірки методики навчання програмування на основі мови С# в класах технологічного профілю;

6. Корекція окремих теоретичних і практичних рекомендацій щодо методики навчання програмування учнів класів технологічного профілю на основі використання мови С#. Внесення необхідних змін до змісту навчального матеріалу та безпосередньо в організацію навчання.

У залежності від цілей педагогічного експерименту, об'єктів дослідження та інших характеристик експериментального дослідження, прийнято виділяти

такі види експерименту, які поряд із цим є відповідними етапами усього наукового дослідження [145]:

– констатуючий експеримент (вивчення існуючого стану досліджуваної проблеми та пошук шляхів вирішення досліджуваної проблеми);

– формуючий експеримент (створення і перевірка ефективності нових методів, прийомів, засобів, які, на думку дослідника, покращують існуючий стан досліджуваної проблеми після педагогічного впливу).

– контрольний експеримент (виявлення результатів формувального впливу на попередньому етапі експерименту.

Було поставлено такі завдання констатуючого етапу експерименту:

а) виявлення сучасного стану навчання програмування в курсі інформатики профільних класів ЗНЗ, актуальних проблем його розвитку, умов впровадження новітніх мов програмування до такого навчання;

б) визначення поняття «парадигма програмування» у застосуванні до навчання програмування в класах технологічного профілю;

в) виявлення доцільності навчання програмування в класах технологічного профілю на основі мови програмування C#;

г) виявлення умов ефективного навчання програмування в класах технологічного профілю на основі мови програмування C#;

д) з'ясування ставлення вчителів до даної проблеми та основних труднощів, що перешкоджають впровадженню навчання програмування в класах технологічного профілю на основі мови програмування C#;

е) виявлення дидактичних вимог до сучасних мов та середовищ програмування;

є) пошук різних організаційних форм, методичних прийомів навчання програмування в класах технологічного профілю на основі мови програмування C#;

ж) аналіз результатів практичного впровадження навчання програмування в класах технологічного профілю на основі мови C#;

з) пошук шляхів подальшого удосконалення системи навчання програмування в класах технологічного профілю на основі мови програмування C#.

На першому етапі (2008 – 2009 н. р.) (Констатуючий експеримент) вивчався теоретичний стан досліджуваної проблеми шляхом аналізу психолого-педагогічної, наукової та науково-методичної літератури, Інтернет-ресурсів; емпірично вивчалися і аналізувалися знання, вміння та навички учнів класів технологічного профілю; вивчалися наукові досягнення в галузі програмування, зарубіжний і вітчизняний досвід учителів, які здійснюють навчання програмування, працюють у ЗНЗ, мають відношення до профільного навчання; вивчалися педагогічні технології, визначалися напрямки і завдання експерименту; проводилося анкетування та експрес-опитування учнів, випускників; розроблялася варіативна частина навчального плану.

У результаті констатуючого експерименту було встановлено:

– більшість учнів класів технологічного профілю загальноосвітніх навчальних закладів вважає, що їхня професійна культура визначається знаннями, отриманими під час вивчення дисциплін, безпосередньо пов'язаних з теоретичними і практичними основами інформатики;

– у більшості учнів знання програмування сформовано на рівні досвіду вирішення стандартних алгоритмічних проблем, не пов'язаних з реальними професійними завданнями;

– більшість учнів класів технологічного профілю погано уявляє завдання, що стоять перед майбутніми програмістами, не має уявлення про взаємозв’язок навчання програмування з підготовкою до діяльності, непов’язаної з написанням комп’ютерних програм;

– відсутнє цілеспрямоване формування основних компонентів професійної культури під час навчання програмування;

– необхідна цілеспрямована робота по розробці методичної системи навчання програмування в класах технологічного профілю загальноосвітніх навчальних закладів.

Дані висновки знаходили багаторазове підтвердження в бесідах з учителями інформатики, що працюють у навчальних закладах різних регіонів України, а також з колегами під час міжнародних і всеукраїнських наукових та науково-методичних конференцій і семінарів.

Поряд із цим, теоретично обґрунтовувалися основні концептуальні підходи до створення методики навчання програмування учнів класів технологічного профілю на основі використання мови C#. Було розроблено методику навчання програмування на основі мови програмування C# в класах технологічного профілю. Підготовлено методичні рекомендації з навчання програмування мовою C#. Створено сайт для інформаційної підтримки такого навчання. Підготовлено матеріали для методичного забезпечення формуючого експерименту: календарне планування, плани-конспекти уроків, навчальні презентації, систему питань для тестування знань учнів, протоколи практичних робіт, завдання для контролю знань учнів, комплект інсталяції найпростішого середовища розробки мовою C# на основі програми Antechinus C# editor та платформи .Net Framework 1.015

Для забезпечення об’єктивності проведення формувального та контрольного експериментів виникла необхідність:

– визначити критерії формування контрольних та експериментальних груп;

– визначити методи визначення еквівалентності вхідних умов для здійснення навчання програмування в контрольних та експериментальних групах;

– передбачити методи проведення контролюючого експерименту.

Формування контрольних та експериментальних груп здійснювалося шляхом підбору рівноцінних вхідних умов та загального рівня розвитку учнів. До уваги бралися такі параметри:

– рівень підготовки вчителя, його бажання та можливості брати участь в експерименті;

– наявність відповідних програмно-технологічних умов у комп'ютерних класах школи;

– рівень навчальних досягнень учнів експериментальних та контрольних класів з предметів природничо-математичного циклу;

– стан сформованості початкових знань з інформатики та основ програмування, що визначався за результатами тестування.

Для визначення рівня навчальних досягнень учнів експериментальних та контрольних класів вивчалися середні бали оцінок з предметів природничо-математичного циклу. Стан сформованості початкових знань з інформатики та основ програмування визначався за результатами тестування, в основі якого лежали питання, що перевіряли знання апаратної складової комп'ютера, знання пристроїв введення та користування ними, загальні відомості про операційну систему, поняття про роботу з текстом та графікою; знання прикладного ПЗ: загальні поняття про Інтернет; загальні поняття з програмування (алгоритм, тип даних, розгалуження, цикл); уявлення про засоби та системи, що використовуються для програмування. Запитання для системи підбирались із наступної літератури: Я. Б. Кащеєв, Г. І. Кащеєва «Збірник практичних завдань з інформатики [119]»; М. К. Чоба, Л. В. Палюшок, Л. М. Кос та ін. «Завдання для тематичного оцінювання з інформатики (10-11 класи) [105]». Результати вказувались у відсотках правильних відповідей до загальної кількості запитань. Аналіз рівня навчальних досягнень учнів з предметів природничо-математичного циклу та стану сформованості початкових знань з інформатики дозволив підібрати експериментальні та контрольні класи так, щоб забезпечити приблизно однаковий рівень підготовки учнів. При цьому також вдалося

досягти незначну різницю у кількості учнів експериментальних та контрольних класів.

### 3.1.1. Проблема визначення ефективності навчання програмування

Для проведення контрольного експерименту першочерговим завданням було добір методів експериментальних педагогічних досліджень, які б дозволили перевірити ефективність формувального експерименту.

Традиційно результати навчання визначаються шляхом перевірки виконання учнями певних контрольних завдань. Досить складно об'єктивно оцінити ефективність навчання програмування шляхом порівняння результатів виконання одних і тих же завдань на основі різних мов та парадигм написання комп’ютерних програм. Це зумовлено тим, що різні мови програмування не лише надають різні засоби для розв’язання одних і тих самих завдань, а й вимагають дотримання різних підходів. Суттєво впливає на порядок написання тієї чи іншої програми парадигма, на основі якої це здійснюється. Наприклад, навіть обсяг найпростішої програми, що виводить повідомлення на екран типу «Hello world», може значно відрізнятися відповідно до мови програмування, якою вона написана [302, с. 49]. Отже, для порівняння результатів навчальної діяльності з програмування доцільно використовувати такі методи, які були б якомога більш віддаленими від розв’язання тих завдань, ефективність вирішення яких визначається технологією програмування. Для об’єктивного визначення загальної ефективності навчання програмування потрібно отримати кількісні характеристики розвитку особистості в його процесі. Для того, щоб зменшити вплив формальних деталей, окремих елементів змісту начального матеріалу, специфічних особливостей певних засобів та форм навчання на результати освітнього вимірювання, варто використовувати методики, дещо віддалені від конкретної діяльності учнів. У такому разі заслуговують на увагу так звані проективні методи психолого-педагогічних досліджень, що спираються на принципи побудови проективних методик дослідження особистості в психології.

Дослідження проективних технологій у психології проводили И. Ю. Агапова [2], Е. Ю. Артем’єва [5; 6], Н. С. Бурлакова [29], Л. Ф. Бурлачук [3030], Р. С. Немов [191], В. Ф. Петренко [213; 212], Ю. С. Савенко [247], В. П. Серкін [261], Е. Т. Соколова [269], И. Б. Ханіна [293] та інші. Соколова вказує на проективні методики у психології як такі, що слугують для дослідження прихованих особливостей особистості [269, с. 3]. Р. С. Немов відзначає високу валідність та надійність проективних методик [191, с. 41]. Однією з широко розповсюджених проективних технологій є метод семантичного диференціалу (скорочено СД), що знайшов широке застосування для психолого-соціальних досліджень у різних сферах [5; 192; 219; 348].

Існує позитивний приклад використання семантичного диференціалу у педагогічних дослідженнях. Зокрема, Н. К. Голубєв успішно використовував шкали особистісного диференціала, розроблені на основі шкал семантичного диференціала. Для оцінки досліджуваного явища використовується комплекс семантичного профілю. Отримані результати зображуються у вигляді семантичного профілю, який відзначається шляхом з'єднання точок на шкалах [58]. З використанням методу часткового семантичного диференціала Ю. О. Жук та О. П. Пінчук проводили оцінювання рівня сформованості предметних компетентностей учнів основної школи під час навчання фізики. Використання такої методики дослідження дозволило оцінити динаміку формування моделі предметної галузі в індивідуальній свідомості учнів по відношенню до моделі предметної галузі, яка сформована у компетентного експерта [219].

Постало завдання розробити та впровадити методику застосування семантичного диференціалу для визначення ефективності навчання програмування мовою C# в класах технологічного профілю загальноосвітніх навчальних закладів.

### 3.1.2. Метод семантичного диференціалу

Метод семантичного диференціалу (гр. sêmantikos – позначаючий і лат. differentia – різниця) розроблений в 1952 році групою американських вчених під керівництвом Чарльза Осгуда для вивчення емоційного ставлення людей до тих чи інших понять [348]. Цей метод психолінгвістики та експериментальної семантики є окремим різновидом способів побудови суб'єктивних семантичних просторів. Метод СД був розроблений в ході вивчення механізмів синестезії і отримав широке застосування в дослідженнях, пов'язаних зі сприйняттям і поведінкою людини, з аналізом соціальних установок і особистісних смислів. Висока гнучкість та об’єктивність методу додала йому чималої популярності. Його традиційно використовують у психології та соціології, теорії масових комунікацій і рекламі, а також в області естетики. Сфери використання семантичного диференціалу постійно розширюються. Як уже зазначалося, перспективним є використання семантичного диференціалу в педагогічній експериментальній роботі [58; 219].

Як вважає Осгуд [348], метод семантичного диференціалу надає можливість вимірювати конотативні значення – ті стани, які слідують за сприйняттям певного символа-подразника і обов’язково передують осмисленим операціям з цим символом [6]. Метод СД є комбінацією методу контрольованих асоціацій і процедур шкалювання. У методі семантичного диференціалу вимірювані об'єкти (поняття, зображення, окремі персонажі і т. п.) оцінюються по ряду біполярних градуальних (трьох-, п'яти-, семибальною) шкалах, полюси яких задані за допомогою вербальних антонімів. Оцінки понять за окремими шкалами корелюють одна з одною [6; 212; 348]. У роботах Осгуда було виділено три основні чинники («Оцінка», «Сила», «Активність»), що об'єднує багато різних шкал, і для диференціації конотативних значень використовувався декартовий тривимірний простір. Аналогічні результати були отримані В. Ф. Петренко на матеріалі російської лексики [212].

Поряд з універсальними СД, побудованими на базі лексики з різних семантичних класів, будуються і окремі семантичні диференціали для обмежених понятійних класів. Наприклад, дослідниками побудовано ряд окремих семантичних просторів: «особистий семантичний диференціал», «диференціал політичних термінів» і т. п. [212]. Побудова таких семантичних просторів дозволяє проводити більш тонкий семантичний аналіз. Окремі семантичні простори, побудовані для даної соціальної групи або окремої особи, не мають міжкультурної інваріантності і несуть диференціальні психологічні ознаки. Останнє робить можливим використання їх для об’єктивних досліджень самих різноманітних індивідуальних відмінностей. Поряд з вербальними семантичними диференціалами розроблені невербальні СД, що використовують як шкали графічні опозиції, живописні картини і фотографічні портрети.

На відміну від «класичного» семантичного диференціалу, де лексика відбирається випадковим чином з самих різних семантичних областей, нами застосовано частковий семантичний диференціал, який будується на основі вузького понятійного класу. Аналогічним чином метод СД використовують в багатьох різних галузях, наприклад, у маркетингових дослідженнях, політичної іміджелогії, візуальному дизайні.

### 3.1.3. Застосування методу СД для визначення ефективності навчання програмування

У нашому дослідженні мовою категорій, що задані певними термінами, описується семантичний простір учня. Відслідковується те, як відбулося зміщення семантичного диференціалу в експериментальних навчальних групах у порівнянні з контрольними, і чи відбулося воно взагалі. Також визначається умовний напрям зміщення по відношенню до семантичного диференціалу експерта. Семантичний диференціал експерта виступає певним еталоном, що вказує на ефективність виявленого зміщення. Було обрано таку різновидність СД, за умов якої учень повинен визначити ступінь зв'язку, що, на його думку, існує між різними поняттями та термінами з галузі програмування. Поряд з тим,

додатково ставиться завдання оцінити зв’язок термінів зі сфери програмування та деяких понять з інших галузей. Це дозволило визначити, як учень пов’язує програмування з іншими питаннями курсу шкільної інформатики, навчальним матеріалом інших шкільних дисциплін. Такий добір семантичних пар дозволяє відслідкувати змістову єдність розділу «Основи алгоритмізації та програмування», єдність цього розділу з іншими розділами шкільної інформатики, змістову єдність інформатики з іншими предметами, що вивчаються в класах технологічного профілю загальноосвітніх навчальних закладів.

Для опису семантичного простору було підібрано список 20-ти основних понять, що стосуються, як процедурного, так і об’єктно-орієнтованого програмування: інформаційна модель, алгоритм, програма, виконавець, компіляція, величина (змінна), тип даних, присвоєння, розгалуження, повторення, масив, підпрограма, клас у програмуванні, об’єкт певного класу, метод об’єкта, рядкові дані, наслідування, віконна форма, елемент управління, подія.

Ще 25 понять та термінів відповідають іншим розділам інформатики: інформатика, комп’ютер, інформація, повідомлення, носій інформації, байт, кодування, пристрої введення, процесор, операційна система, файл, типи файлів, віконний інтерфейс, вірус комп’ютерний, архів файловий, вікно на екрані комп’ютера, буфер обміну, редактор тексту, слайд презентації, база даних, адреса файлу, мережа комп’ютерна, Інтернет, World Wide Web, браузер Інтернет.

Поряд з тим було додано 12 понять, що не стосуються або мало стосуються сфери інформаційних технологій: математика, мистецтво, кулінарія, моделювання, виробництво, спорт, мовознавство, фізика, культура, біологія, музика, техніка.

Таким чином, проводилось порівняння 20 понять зі сфери програмування з шістдесятьма двома поняттями програмування, інформатики та інших

шкільних дисциплін. Варто зазначити, що усі поняття з програмування та інших розділів інформатики вибиралися зі змісту навчального матеріалу, який передбачений навчальними програми з інформатики для ЗНЗ [116]. Дванадцять термінів – це назви різних сфер людської діяльності: кулінарія, спорт, мовознавство, виробництво. Серед цих термінів деякі є назвами шкільних навчальних дисциплін: математика, фізика, біологія, музика.

Було розроблено бланки анкет та інструкцію щодо їх заповнення. Бланки анкет пропонувалися у двох різних за оформленням варіантах (формах). Перша, більш наочна форма анкети, (див. рис 3.1.) розміщується на п’яти сторінках формату А4 та передбачає заповнення окремих, для кожного терміну, таблиць оцінювання зв’язків цього терміну з іншими. Оцінка визначається шляхом встановлення позначки напроти відповідного поняття в одну з шести комірок. Комірки за значенням оцінки розміщуються у порядку спадання зліва направо. Таким чином, досягається певна візуалізація процесу оцінювання. На анкетах такої форми зручно пояснювати порядок оцінювання понять.

Прізвище, ім’я _____ Клас _____ Школа _____

| 1.Інформаційна модель<br>Як пов’язані «інформаційна модель» та: | Тісно пов’язані | Досить пов’язані | Більше пов’язані | Частково’язані | Мало пов’язані | Зв’язку нема |
|---|---|---|---|---|---|---|
| | 5 | 4 | 3 | 2 | 1 | 0 |
| математика | ✓ | | | | | |
| мистецтво | | ✓ | | | | |
| кулінарія | | | | | ✓ | |
| моделювання | ✓ | | | | | |
| виробництво | | | ✓ | | | |
| спорт | | | | ✓ | | |
| мовознавство | | | | | | |
| фізика | | | ✓ | | | |
| культура | | | | | | ✓ |

| 2. Алгоритм<br>Як пов’язані «алгоритм» та: | Тісно пов’язані | Досить пов’язані | Більше пов’язані | Частково’язані | Мало пов’язані | Зв’язку нема |
|---|---|---|---|---|---|---|
| | 5 | 4 | 3 | 2 | 1 | 0 |
| математика | ✓ | | | | | |
| мистецтво | | | | ✓ | | |
| кулінарія | | | | | | |
| моделювання | | | | | | |
| виробництво | | | | | | |
| спорт | | | | | | |
| мовознавство | | | | | | |
| фізика | | | | | | |
| культура | | | | | | |

| 3. Програма<br>Як пов’язані «програма» та: | Тісно пов’язані | Досить пов’язані | Більше пов’язані | Частково’язані | Мало пов’язані | Зв’язку нема |
|---|---|---|---|---|---|---|
| | 5 | 4 | 3 | 2 | 1 | 0 |
| математика | | | | | | |
| мистецтво | | | | | | |
| кулінарія | | | | | | |
| моделювання | | | | | | |
| виробництво | | | | | | |
| спорт | | | | | | |
| мовознавство | | | | | | |
| фізика | | | | | | |
| культура | | | | | | |

Рис. 3.1. Частина першої, більш наочної форми для проведення анкетування.

Друга, узагальнена форма, (див. рис 3.2.) виконана у вигляді прямокутної таблиці, заголовками стовпців та рядочків якої є поняття, зв’язок між якими потрібно оцінити. У комірки таблиці другої форми виставляється цифра від 0 до 5, яка й показує наскільки тісно учень чи експерт пов’язують між собою поняття, що оцінюються. Така таблиця займає на папері значно менше місця і з її використанням суттєво полегшується перенесення даних анкетування до комп’ютера.

| Як пов'язані між собою поняття? | інформаційна модель | алгоритм | програма | виконавець | компіляція | величина (змінна) | тип даних | присвоєння | розгалуження | повторення | масив | підпрограма | клас в програмуванні | об'єкт певного класу | метод об'єкту (класу) | рядкові дані (величини) | наслідування | віконна форма | елемент управління | подія |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| математика | | | | | | | | | | | | | | | | | | | | |
| мистецтво | | | | | | | | | | | | | | | | | | | | |
| кулінарія | | | | | | | | | | | | | | | | | | | | |
| моделювання | | | | | | | | | | | | | | | | | | | | |
| виробництво | | | | | | | | | | | | | | | | | | | | |
| спорт | | | | | | | | | | | | | | | | | | | | |
| мовознавство | | | | | | | | | | | | | | | | | | | | |
| фізика | | | | | | | | | | | | | | | | | | | | |
| культура | | | | | | | | | | | | | | | | | | | | |
| біологія | | | | | | | | | | | | | | | | | | | | |
| музика | | | | | | | | | | | | | | | | | | | | |
| техніка | | | | | | | | | | | | | | | | | | | | |
| інформатика | | | | | | | | | | | | | | | | | | | | |
| комп'ютер | | | | | | | | | | | | | | | | | | | | |

Рис. 3.2. Частина другої, узагальненої форми для анкетування.

Враховуючи великі обсяги та значну трудомісткість, як анкетування, так і наступної оброки анкет, нами розроблено інтерактивний сайт для організації тестування, збору та узагальнення даних. Сайт розміщено за адресою: http://sd.ms1.org.ua/.

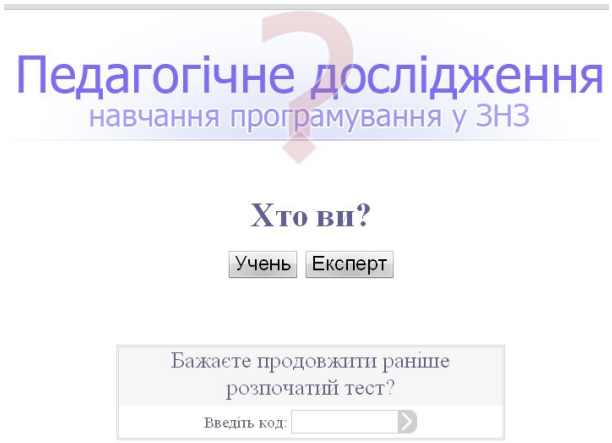



Рис. 3.3. Стартова сторінка сайту для проведення анкетування.

Щоб розпочати анкетування, потрібно надати про себе деякі відомості (див. рис 3.4.). Незакінчене опитування можна продовжити, увівши на стартовому вікні сайту персональний код доступу (див. рис 3.3.). Код доступу рекомендується учням записати на папері ще на початку сеансу роботи з сайтом та зберігати до завершення опитування. Код надає можливість в будь-який

момент призупинити та в зручний час продовжити анкетування. Персональний код доступу відображається у верхньому лівому кутку інтерактивної таблиці анкетування на сайті (див. рис 3.5.).

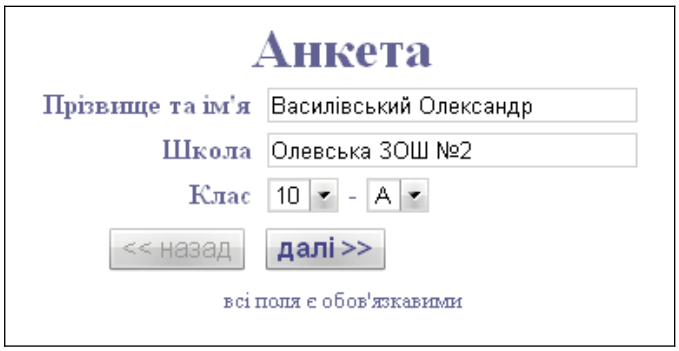



Рис. 3.4. Анкета для початкової реєстрації на сайті учня, як учасника анкетування.

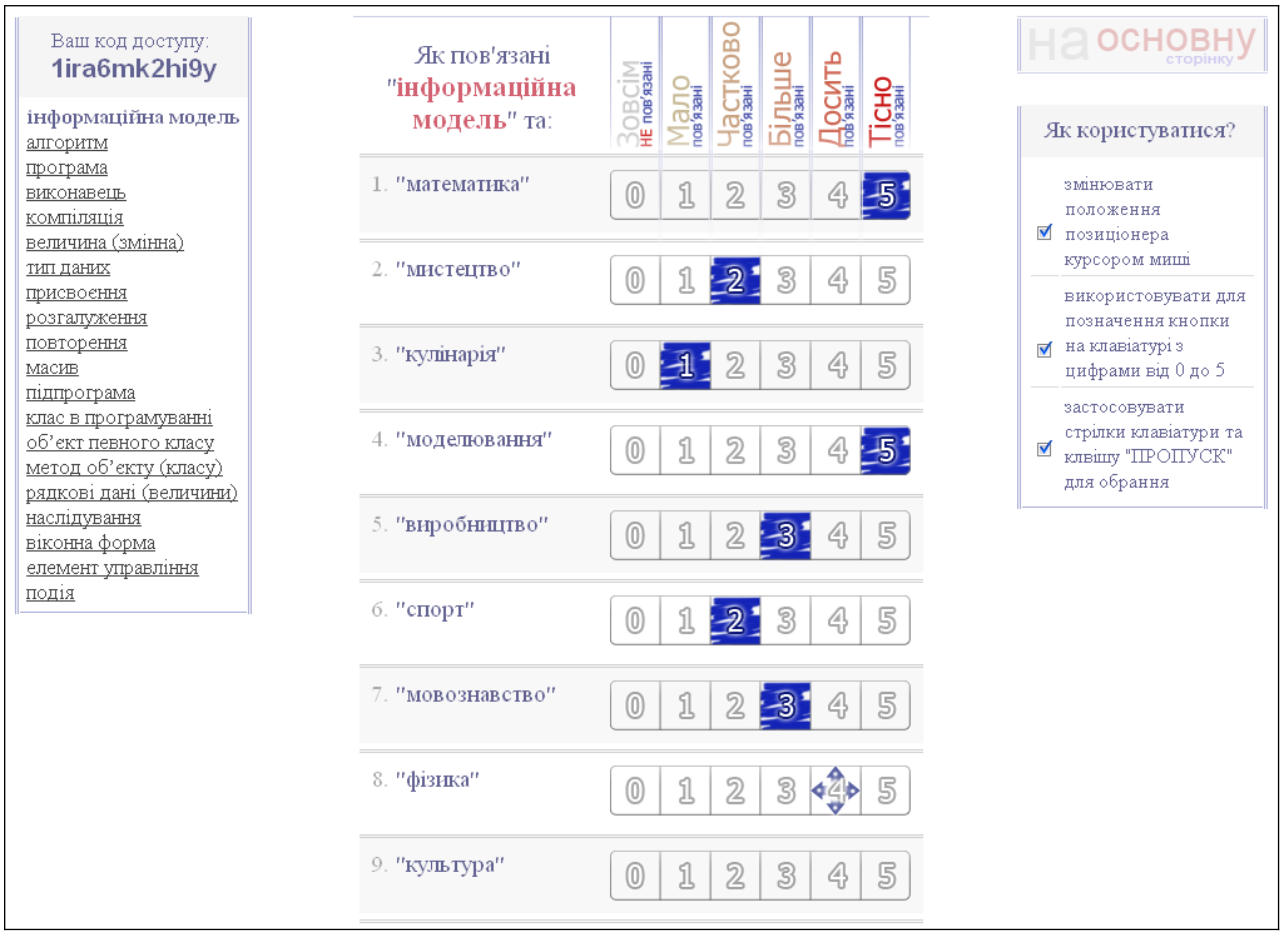



Рис. 3.5. Вигляд інтерактивної таблиці на сайті анкетування.

Алгоритм роботи з сайтом можна описати у вигляді наступної інструкції:

Учасник анкетування, зайшовши на сайт http://sd.ms1.org.ua, натискає кнопку з написом «Учень» або «Експерт» відповідно до того, ким він являється. Після цього проходить його реєстрація, що полягає у заповненні полів реєстраційної форми.

Наступним кроком є оцінювання того, як тісно пов'язані пари запропонованих понять. Для оцінювання використовуються оцінки в діапазоні від 0 до 5, де:

0 – поняття зовсім не пов'язані;

1 – мало пов'язані;

2 – частково пов'язані;

3 – більш пов'язані ніж непов'язані;

4 – досить достатньо пов'язані;

5 – тісно пов'язані між собою;

Порівняння відбувається між двома окремими списками понять. Кожне поняття з першого списку порівнюється з кожним елементом другого. Оскільки частина понять у списках повторюється, тому система виключає повторне оцінювання.

В меню, що знаходиться зліва, користувач може сам вибрати поняття для порівняння.

Користувачеві пропонується оцінити всі комбінації пар, тому процес опитування триває досить довго, потребує зосередженості та уваги. Щоб надати користувачу можливість робити перерву у тестуванні передбачений механізм ідентифікації у вигляді персонального коду доступу.

Для зручності роботи з сайтом на сторінці присутня панель налаштувань, яка дозволяє вибирати відповідний спосіб заповнення анкет. Найоптимальніший спосіб передбачає використання клавіатури. З допомогою клавіатури можна задати оцінку від 0 до 5, натискаючи на відповідні клавіші. При цьому перехід до наступної пари відбувається автоматично.

### 3.1.4. Аналіз даних

Ще одна функція сайту – це автоматизація початкового аналізу даних.

На відповідній сторінці сайту http://sd.ms1.org.ua/view можна переглянути результати анкетування його учасників у вигляді прямокутних таблиць та середні значення порівняння для кожної пари понять окремо всіх учнів експериментальних класів, усіх учнів контрольних класів та всіх залучених до анкетування експертів.

Учні

| id | Ім'я | Школа | Клас | hash |
|---|---|---|---|---|
| 12 | Кухарський Євген | Понінківський НВК №1 | 11-а | 1it335ogsr8y |
| 13 | Лановюк Діана | Миропільська гімназія | 11-а | 1it5kroaotgy |
| 285 | Зайнчківський Володимир | Загальноосвітня школа I-III ступенів № 7, м. Житомира | 10-в | 4ool2ef1y |
| 286 | Кохан Денис | Загальноосвітня школа I-III ступенів № 7, м. Житомира | 10-в | 4ool2ef2y |
| 287 | Комаренко Аліна | Загальноосвітня школа I-III ступенів № 7, м. Житомира | 10-в | 4ool2ef3y |
| 288 | Лук'янчук Богдан | Загальноосвітня школа I-III ступенів № 7, м. Житомира | 10-в | 4ool2ef4y |
| 289 | Мала Анастасія | Загальноосвітня школа I-III ступенів № 7, м. Житомира | 10-в | 4ool2ef5y |

Рис. 3.6. Сторінка перегляду даних на сайті педагогічного дослідження

На сторінці, що відображає таблицю середніх значень експертів поряд з середнім значенням вказується діапазон їх розходження. Розходження в значеннях оцінок експертів визначається як середнє квадратичне відхилення.

Середньоквадратичне відхилення — дорівнює кореню квадратному з дисперсії випадкової величини [167, с. 344] :

$$\sigma = \sqrt{\sigma^2} \quad (3.1)$$

Відповідно до формул з обчислення дисперсії:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad (3.2)$$

при невеликій вибірці (n<=40 – 50) вводиться поправка Бесселя:

$$s = \sqrt{\frac{n}{n-1}\sigma^2} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad (3.3)$$

де:

$s$ – стандартне відхилення, незміщена оцінка середньоквадратичного відхилення випадкової величини X відносно її математичного сподівання;

$\sigma^2$ – дисперсія;

$x_i$ – і-й елемент вибірки;

$\bar{x}$ — середнє арифметичне вибірки.

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i. \qquad (3.4)$$

$n$ — обсяг (розмір) вибірки.

Автоматично визначається двадцять пар, що мають найменше розходження в межах даного поняття з галузі програмування. Користувач має можливість вибрати іншу пару понять в межах того ж таки певного поняття з галузі програмування. Можливість самостійно вибрати пари понять дозволяє проводити аналіз результатів оцінювання зв’язку термінів галузі програмування з термінами, вибраними з тієї чи іншої групи понять: об’єктно-орієнтованого програмування, процедурного програмування, інших розділів інформатики, термінів мало причетних до ІКТ.

Натисканням відповідної кнопки на сторінці, що відображає таблицю результатів анкетування експертів, формується прямокутна матриця, у якій вказано кількість кожної з шести можливих оцінок для тієї чи іншої з 20-ти вибраних пар понять.

На наступному етапі за аналогічними таблицями для учнів експериментальних та контрольних класів визначається результуюче відхилення.

Окрім загальної оцінки пов’язаності двадцяти пар, дібраних за найменшим квадратичним розходженням, з усієї шкали, яка містить аж 62 терміни різних галузей, аналогічно здійснюється оцінка понять галузі програмування з окремими групами понять. Усього здійснюється ще чотири порівняння:

– усіх 20-ти понять галузі програмування та лише понять об’єктно-орієнтованого програмування;

– усіх понять галузі програмування та лише понять процедурного програмування;

– усіх понять галузі програмування та понять інших розділів інформатики;

– усіх понять галузі програмування та понять мало причетних до ІКТ.

На основі отриманих результатів розбіжностей можна проводити аналіз та робити висновки щодо результатів педагогічного впливу.

### 3.2. Практична реалізація та підсумки експериментального дослідження

Протягом другого етапу (формуючий експеримент) здійснювалось навчання програмування учнів контрольних та експериментальних класів у відповідності до вимог програми з інформатики для класів технологічного профілю [116, с.65-85]. За основу було взяте рекомендоване календарне планування згідно цієї програми розроблене Т. Г. Проценко, опубліковане в журналі «Інформатика та інформаційні технології в навчальних закладах» № 4-5 за 2006 р. [231]. (див. додаток В.) Розподіл годин на вивчення окремих розділів та кількість практичних робіт визначалася за відповідними рекомендаціями до обраної навчальної програми [122, с.88]. Відмінність у навчанні програмування між контрольними та експериментальними групами полягала у використанні мови програмування C# та елементів авторської методики навчання програмування. Навчання контрольних класів здійснювалось на основі тієї ж програми та практично такого ж календарного планування, але переважно на основі мови програмування Pascal з використанням традиційних методик такого навчання.

Дослідно-експериментальною роботою було охоплено 255 учнів. З них 126 учнів в експериментальних та 128 учнів у контрольних групах класів технологічного профілю. Також участь в організації та проведенні експериментального дослідження взяли 12 учителів інформатики.

### 3.2.1. Формування експериментальних та контрольних груп

Перед початком другого етапу (формуючий експеримент), який відбувався в 2009-2011 н. р., з метою з'ясування стану сформованості знань із програмування протягом другого этапу, виконувались констатуючі зрізи. Результати опитування сприяли такому формуванню експериментальних та контрольних груп, який би міг максимально забезпечити однакові вхідні умови за рівнем розвитку учнів обох груп. Першу (експериментальну) групу складали учні класів технологічного профілю загальноосвітніх навчальних закладів, які будуть вивчати програмування на основі мови C#. Другу групу (контрольну) склали учні класів технологічного профілю, які мають вивчати програмування на основі інших мов, здебільшого мови Pascal. З респондентами груп опитування проводилось у формі анкетування. Результати опитування наведені у табл. 3.1

*Таблиця 3.1.*

**Стан сформованості початкових знань з інформатики та основ програмування**

| № п/п | Розглядувані поняття та питання | Правильні відповіді, у % | |
|---|---|---|---|
| | | Експ. Група | Контр. група |
| 1 | Знання апаратної складової комп'ютера | 91 | 93 |
| 2 | Знання пристроїв введення та користування ними | 85 | 85 |
| 3 | Загальні відомості про операційну систему | 83 | 79 |
| 4 | Поняття про роботу з текстом та графікою | 93 | 92 |
| 5 | Знання прикладного ПЗ | 73 | 74 |
| 6 | Загальні поняття про Інтернет | 91 | 90 |
| 7 | Поняття алгоритму | 52 | 54 |
| 8 | Поняття типу даних | 72 | 71 |
| 9 | Поняття розгалуження | 67 | 68 |

| | | | |
|---|---|---|---|
| 10 | Поняття циклу | 67 | 69 |
| 11 | Уявлення про засоби та системи, що використовуються для програмування | 60 | 65 |

Узагальнити та проаналізувати результати анкетування дозволяє діаграма подана на Рис. 3.7.

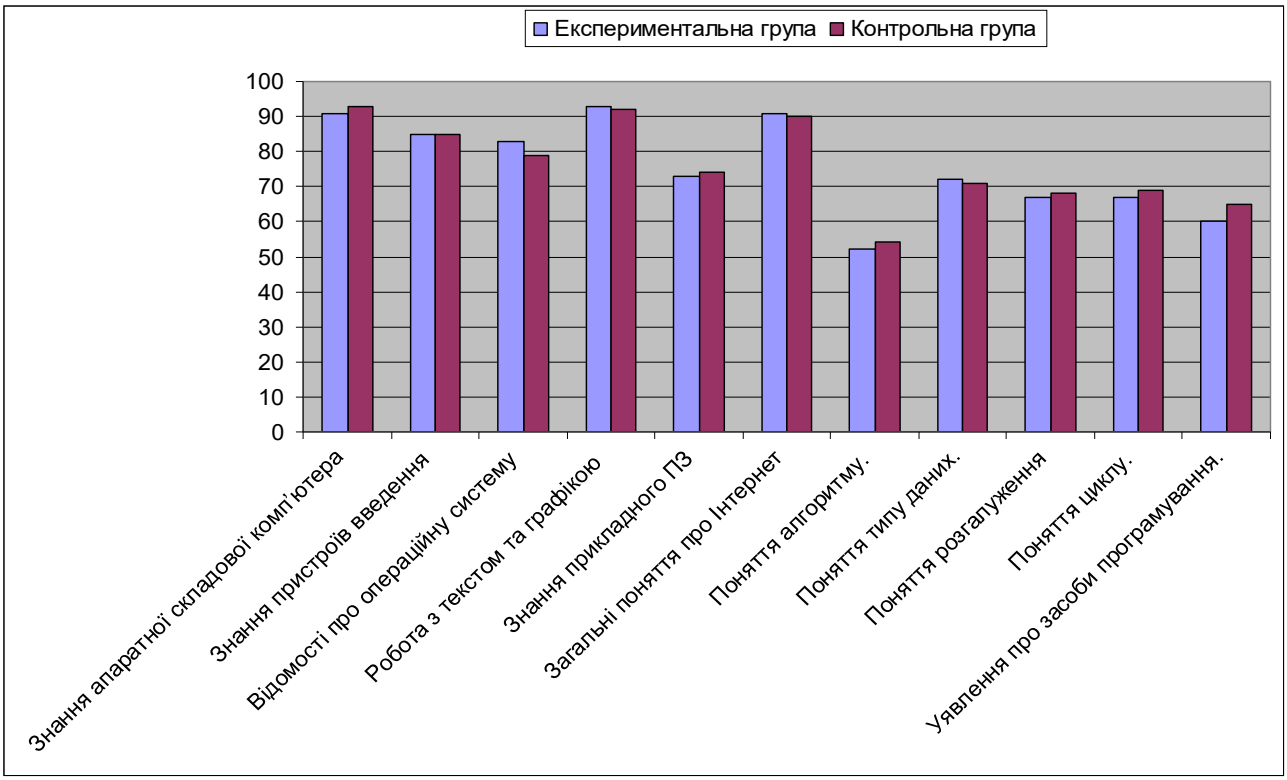



Рис. 3.7. Стан сформованості початкових знань з інформатики та основ програмування

Результати проведеного анкетування показали приблизно однаковий рівень усвідомлення учнів у галузі інформаційних технологій. Відповіді на питання, пов’язані з основними поняттями програмування (пункти 7-11 табл. 3.1), учні обох груп показали розходження, в середньому, менші 4%. В загальному, показники контрольних груп вищі на 0,5 % від результатів, що продемонстрували експериментальні групи.

Додатково було проаналізовано бал успішності з профільних предметів. Результати наведені у табл. 3.2.

Аналіз даних зібраних під час добору експериментальних та контрольних класів засвідчив характерні для всіх опитуваних особливість:

– учні міських шкіл мають загалом більш високі бали з профільних дисциплін, ніж учні сільських шкіл;

*Таблиця 3.2*

**Бал успішності з профільних дисциплін**

| № п/п | Навчальна дисципліна | Середній бал | |
|---|---|---|---|
| | | Експериментальна група | Контрольна група |
| 1 | Математика | 9,3 | 9,7 |
| 2 | Фізика | 8,9 | 9,5 |

– учні контрольної групи мали дещо кращі показники з профільних дисциплін, ніж експериментальної групи.

– незважаючи на середній бал з профільних предметів (фізика, математика), базові знання ІКТ в обох групах знаходяться приблизно на однаковому рівні.

### 3.2.2. Загальний порядок проведення контрольного експерименту

Формуючий етап експерименту проводився в 2010-2011 н. р. на базі наступних загальноосвітніх навчальних закладів: Миропільської гімназії, Миропільської загальноосвітньої школи І-ІІІ ступенів № 2, Камінської загальноосвітньої школи І-ІІІ ступенів, Гордіївської загальноосвітньої школи І-ІІІ ступенів, Биківської загальноосвітньої школи І-ІІІ ступенів, Романівського ліцею. А також на базі чотирьох навчальних закладів інших районів Житомирської області: загальноосвітньої школи І-ІІІ ступенів № 7 м. Житомира, загальноосвітньої школи І-ІІІ ступенів № 21 м. Житомира, загальноосвітньої школи І-ІІІ ступенів № 3 м. Олевська та Озернянської загальноосвітньої школи І-ІІІ ступенів Житомирського району.

Формуючий експеримент проводився за допомогою методики семантичного диференціалу. Це один із методів побудови суб'єктивних семантичних просторів і застосовується в дослідженнях, пов'язаних зі сприйняттям людини і її поведінкою.

Загалом, під час дослідження анкетування проводилось, як шляхом заповнення анкет, так і на сторінках інтерактивного сайту. За основу оцінки ефективності навчання програмування було взято результати анкетування експертів з галузі розробки програмного забезпечення. Як експерти виступили провідні вчителі інформатики, науковці та фахівці з галузі програмування різних організацій. До формування експертних оцінок вдалося залучити 24 фахівці. Серед них є співробітники Інституту інформаційних технологій і засобів навчання НАПН України, студенти старших курсів та викладачі деяких вищих навчальних закладів, учасники обласних та районних семінарів учителів інформатики Житомирської та Хмельницької областей, професійні програмісти.

Усього в анкетуванні взяло участь 126 учнів експериментальних класів, що вивчали програмування на основі мови C#, 128 учнів контрольних класів, де проводилося навчання програмування на основі інших мов програмування, здебільшого мови Pascal. Експеримент проводився в десяти школах Житомирської та Хмельницької областей на базі класів технологічного профілю, що вивчали програмування в 10 або 11 класах.

Під час анкетування було виявлено окремі складнощі. Для заповнення однієї анкети учню потрібно було витратити більше тридцяти хвилин часу. Респондент швидко стомлювався і це ставило під загрозу об’єктивність дослідження. Нами було прийнято рішення рекомендувати учителям розбивати анкетування не менше ніж на чотири етапи, відводячи на нього не більше ніж по 15 хв. Використання інтерактивного сайту значно спростило організацію та проведення анкетування. Учень чи експерт завжди міг повернутися до заповнення анкет on-line у зручний для нього час.

Ще однією проблемою була надзвичайно громіздка процедура переведення даних анкетування, що здійснювалось на папері, до електронного вигляду. Хоча практично усі школи, що брали участь у педагогічному експерименті, мають у комп’ютерних класах доступ до мережі Інтернет, використання сайту для анкетування не дозволило повністю уникнути

заповнення анкет на папері. Щоб забезпечити однорідність та зручність обробки результатів такі дані також вносилися до відповідних таблиць сайту.

### 3.2.3. Аналіз даних контрольного експерименту

Статистичний аналіз даних, отриманих під час експерименту, проводився з використанням методики визначення достовірності збігів і відмінностей для експериментальних даних, виміряних у порядковій шкалі [196, с. 55].

В нашому випадку порядкова шкала з L=6. L вказує кількість можливих варіантів відповідей учасників анкетування. Цілі числа від 0 до 5 усього 6 варіантів. Характеристикою групи є число її членів, що виставили той або інший бал. Для експериментальної групи вектор балів є $n = (n_1, n_2, ., n_L)$, де $n_k$ – число членів експериментальної групи, що отримали, k-й бал, k = 1, 2,.., L. Для контрольної групи вектор балів є $m =(m_1, m_2, ., m_L)$, де $m_k$ – число членів контрольної групи, k, що отримали, k-й бал, k = 1, 2,.., L.

Для даних, отриманих в порядковій шкалі, доцільне використання критерію однорідності $\chi^2$ («хі» – буква грецького алфавіту, назва критерії читається: «хі-квадрат»), емпіричне значення $\chi^2_{емп}$ обчислюється за формулою

$$\chi^2_{емп} = NM\sum_{i=1}^{l}\frac{(\frac{n_i}{N}-\frac{m_i}{M})^2}{n_i+m_i} \quad (3.5)$$

Критичні значення $\chi^2_{0.05}$ критерію $\chi^2$ для рівня значущості 0.05 і градацій шкали оцінок L=6 дорівнює 11,07.

Для аналізу отриманих оцінок використовуємо наступний алгоритм визначення достовірності збігів і відмінностей для експериментальних даних, виміряних в порядковій шкалі:

1. Обчислення для порівнюваних вибірок $\chi^2_{емп}$ емпіричного значення $\chi^2$ по формулі (3.5).

2. Порівняння цього значення з критичним значенням $\chi^2_{0.05}$=11,07 отриманим з відповідної таблиці значень. Якщо $\chi^2_{емп} \leq \chi^2_{0.05}$, то таке порівняння дозволяє зробити висновок, що «достовірність відмінностей характеристик порівнюваних вибірок збігаються з рівнем значущості 0.05». В іншому разі,

якщо $\chi^2_{емп} > \chi^2_{0.05}$, то можна зробити висновок, що «достовірність відмінностей характеристик порівнюваних вибірок складає 95%»;

*Таблиця 3.3.*

Кількісні показники анкетування групи експертів

| № п/п | | Σ | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|---|
| 0 | ВСЬОГО | 24 | 72 | 70 | 62 | 88 | 77 | 111 |
| 1 | інформаційна модель | 24 | 4 | 4 | 3 | 6 | 7 | 0 |
| 2 | Алгоритм | 24 | 2 | 2 | 1 | 3 | 7 | 9 |
| 3 | Програма | 24 | 0 | 5 | 4 | 6 | 6 | 3 |
| 4 | Виконавець | 24 | 5 | 0 | 6 | 2 | 6 | 5 |
| 5 | Компіляція | 24 | 6 | 6 | 6 | 5 | 1 | 0 |
| 6 | величина (змінна) | 24 | 5 | 1 | 5 | 5 | 1 | 7 |
| 7 | тип даних | 24 | 5 | 5 | 7 | 5 | 2 | 0 |
| 8 | Присвоєння | 24 | 0 | 2 | 0 | 4 | 2 | 16 |
| 9 | Розгалуження | 24 | 4 | 6 | 1 | 5 | 3 | 5 |
| 10 | Повторення | 24 | 5 | 7 | 2 | 6 | 2 | 2 |
| 11 | Масив | 24 | 5 | 4 | 5 | 7 | 3 | 0 |
| 12 | Підпрограма | 24 | 3 | 3 | 5 | 4 | 2 | 7 |
| 13 | клас у програмуванні | 24 | 7 | 3 | 1 | 4 | 6 | 3 |
| 14 | Об’єкт певного класу | 24 | 2 | 4 | 3 | 0 | 0 | 15 |
| 15 | метод об’єкта | 24 | 6 | 1 | 3 | 3 | 7 | 4 |
| 16 | Рядкові дані | 24 | 0 | 2 | 0 | 6 | 4 | 12 |
| 17 | Наслідування | 24 | 6 | 0 | 0 | 6 | 6 | 6 |
| 18 | Віконна форма | 24 | 5 | 6 | 0 | 6 | 7 | 0 |
| 19 | Елемент управління | 24 | 2 | 7 | 4 | 1 | 5 | 5 |
| 20 | Подія | 24 | 0 | 2 | 6 | 4 | 0 | 12 |

*Таблиця 3.4.*

Кількісні показники анкетування експериментальної групи

| № п/п | | Σ | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|---|
| 0 | ВСЬОГО | 126 | 302 | 457 | 320 | 418 | 309 | 714 |
| 1 | інформаційна модель | 126 | 6 | 15 | 19 | 20 | 6 | 60 |
| 2 | Алгоритм | 126 | 1 | 7 | 2 | 20 | 6 | 90 |
| 3 | Програма | 126 | 38 | 24 | 9 | 30 | 25 | 0 |
| 4 | Виконавець | 126 | 2 | 21 | 27 | 24 | 37 | 15 |
| 5 | Компіляція | 126 | 15 | 31 | 33 | 15 | 9 | 23 |
| 6 | величина (змінна) | 126 | 6 | 31 | 40 | 13 | 23 | 13 |
| 7 | тип даних | 126 | 0 | 38 | 36 | 18 | 6 | 28 |
| 8 | Присвоєння | 126 | 21 | 18 | 24 | 10 | 4 | 49 |
| 9 | Розгалуження | 126 | 0 | 40 | 13 | 1 | 6 | 66 |
| 10 | Повторення | 126 | 34 | 12 | 20 | 38 | 22 | 0 |
| 11 | Масив | 126 | 4 | 6 | 4 | 15 | 40 | 57 |
| 12 | Підпрограма | 126 | 37 | 33 | 15 | 28 | 13 | 0 |
| 13 | клас у програмуванні | 126 | 28 | 31 | 4 | 24 | 5 | 34 |
| 14 | об’єкт певного класу | 126 | 16 | 2 | 10 | 40 | 5 | 53 |
| 15 | метод об’єкта | 126 | 5 | 39 | 21 | 11 | 9 | 41 |
| 16 | Рядкові дані | 126 | 29 | 29 | 12 | 25 | 20 | 11 |
| 17 | Наслідування | 126 | 23 | 0 | 4 | 36 | 7 | 56 |
| 18 | Віконна форма | 126 | 12 | 12 | 0 | 27 | 27 | 48 |
| 19 | елемент управління | 126 | 19 | 31 | 7 | 18 | 33 | 18 |
| 20 | Подія | 126 | 6 | 37 | 20 | 5 | 6 | 52 |

*Таблиця 3.5.*

Кількісні показники анкетування контрольної групи

| № п/п | | Σ | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|---|
| 0 | ВСЬОГО | 128 | 459 | 490 | 455 | 387 | 315 | 454 |
| 1 | інформаційна модель | 128 | 7 | 24 | 21 | 25 | 17 | 34 |
| 2 | Алгоритм | 128 | 31 | 10 | 35 | 25 | 27 | 0 |
| 3 | Програма | 128 | 35 | 17 | 9 | 30 | 3 | 34 |
| 4 | Виконавець | 128 | 41 | 31 | 16 | 22 | 3 | 15 |
| 5 | Компіляція | 128 | 5 | 39 | 17 | 28 | 18 | 21 |
| 6 | величина (змінна) | 128 | 35 | 40 | 5 | 40 | 8 | 0 |
| 7 | тип даних | 128 | 22 | 39 | 15 | 20 | 28 | 4 |
| 8 | Присвоєння | 128 | 6 | 14 | 35 | 4 | 11 | 58 |
| 9 | Розгалуження | 128 | 7 | 26 | 12 | 24 | 28 | 31 |
| 10 | Повторення | 128 | 17 | 13 | 18 | 19 | 14 | 47 |
| 11 | Масив | 128 | 0 | 19 | 17 | 4 | 8 | 80 |
| 12 | Підпрограма | 128 | 33 | 11 | 39 | 22 | 7 | 16 |
| 13 | клас у програмуванні | 128 | 24 | 16 | 24 | 18 | 9 | 37 |
| 14 | об’єкт певного класу | 128 | 16 | 37 | 23 | 6 | 30 | 16 |
| 15 | метод об’єкта | 128 | 29 | 10 | 41 | 4 | 21 | 23 |
| 16 | Рядкові дані | 128 | 29 | 27 | 39 | 30 | 3 | 0 |
| 17 | Наслідування | 128 | 31 | 19 | 2 | 25 | 23 | 28 |
| 18 | Віконна форма | 128 | 13 | 28 | 39 | 5 | 33 | 10 |
| 19 | елемент управління | 128 | 38 | 38 | 32 | 20 | 0 | 0 |
| 20 | Подія | 128 | 40 | 32 | 16 | 16 | 24 | 0 |

Попередньо результати експерименту для груп «Експерти», «Експеримент» і «Контроль» записуємо до таблиць, нормалізованих для

застосування описаного вище методу визначення достовірності збігів і відмінностей для експериментальних даних, виміряних в порядковій шкалі (див. табл. 3.3, табл. 3.4, табл. 3.5)

Кожна із наведених таблиць містить відомості про загальну кількість опитаних та кількість респондентів, що оцінили зв'язок даної пари понять. Як уже зазначалося усього таких пар визначається двадцять, по одній для кожного поняттями галузі програмування. Дані про оцінювання однієї пари розміщуються у відповідному рядку. Поля таблиці містять, дані про кількість респондентів, що вказали ту чи іншу степінь зв'язку для даної пари понять.

Використовуючи вище описаний метод визначення достовірності збігів і відмінностей для експериментальних даних, виміряних в порядковій шкалі, отримуємо таблицю емпіричних значень $\chi^2$ для заданих результатів досліджень.

*Таблиця 3.6.*

| $\chi^2$ | Експерти | Експеримент | Контроль |
|---|---|---|---|
| Експерти | 0,00 | 9,90 | 13,65 |
| Експеримент | 9,90 | 0,00 | 12,11 |
| Контроль | 13,65 | 12,11 | 0,00 |

Отже, у випадку порівняння кількісних результатів анкетування групи експертів – стовпчики та рядки «Експерти» з кількісними результатами анкетування експериментальної групи – стовпчики та рядки «Експеримент» критерій однорідності складає $\chi^2_{емп} \leq \mathbf{11,07}$, тобто «характеристики цих вибірок збігаються з рівнем значущості 0,05». А у випадку порівняння вибірок «Експертів» з анкетування контрольної групи – стовпчики та рядки «Контроль» маємо $\chi^2 = \mathbf{13,65} > \mathbf{11,07} = \chi^2_{0.05}$, тобто «достовірність відмінностей характеристик порівнюваних вибірок «Експертів» з «Контрольними групами» складає 95%». Отже, дані результати вказують на значно менші відмінності в оцінці понять між експериментальною групою та групою експертів, ніж вони є між контрольною та експертною групами. Такими є загальні оцінки ступеню зв'язку, між різними поняттями та термінами з галузі програмування з цими ж

поняттями, деякими поняттями з інших розділів шкільної інформатики, з термінами галузей віддалених від інформаційних технологій.

### 3.2.4. Визначення достовірних збігів оцінки пар понять окремих груп

Аналогічним чином проводилось визначення достовірності збігів оцінки пар термінів галузі програмування з термінами, вибраними окремо з тієї чи іншої групи понять: об'єктно-орієнтованого програмування, процедурного програмування, інших розділів інформатики, термінів мало причетних до ІКТ.

Аналіз оцінок пар, що містять поняття о**б'єктно-орієнтованого програмування** у випадку порівняння кількісних результатів анкетування групи експертів з кількісними результатами анкетування експериментальної групи складає $\chi^2_{емп}$**= 8,13 ≤ 11,07**. У випадку порівняння вибірок «Експертів» з анкетування контрольної групи $\chi^2$ **= 13,42 > 11,07**, тобто «достовірність відмінностей характеристик порівнюваних вибірок «Експертів» з «Контрольними групами» складає не менше ніж 95%». Дані результати також вказують на значно менші відмінності в оцінці понять об'єктно-орієнтованого програмування між експериментальною групою та групою експертів, ніж вони є між контрольною та експертною групами.

Емпіричні значень $\chi^2$ для заданих результатів подано у табл. 3.7.

*Таблиця 3.7.*

| $\chi^2$ | Експерти | Експеримент | Контроль |
|---|---|---|---|
| Експерти | 0,00 | 8,13 | 14,42 |
| Експеримент | 8,13 | 0,00 | 11,12 |
| Контроль | 14,42 | 11,12 | 0,00 |

Аналіз оцінок пар, що містять поняття **процедурного програмування** у випадку порівняння кількісних результатів анкетування групи експертів з кількісними результатами анкетування експериментальної групи складає $\chi^2_{емп}$**= 6,6 ≤ 11,07**. І у випадку порівняння вибірок «Експертів» з анкетування контрольної групи також $\chi^2_{емп}$**= 6,21 ≤ 11,07**, тобто «достовірність відмінностей характеристик порівнюваних вибірок «Експертів» з «Контрольними групами»

менше ніж 95%». Дані результати також вказують на значні відмінності в оцінці понять процедурного програмування між контрольною та експертною групами та групою експертів.

Емпіричні значень $\chi^2$ для заданих результатів подано у табл. 3.8.

*Таблиця 3.8.*

| $\chi^2$ | Експерти | Експеримент | Контроль |
|---|---|---|---|
| Експерти | 0,00 | 6,6 | 6,21 |
| Експеримент | 6,6 | 0,00 | 2,2 |
| Контроль | 6,21 | 2,2 | 0,00 |

Аналіз оцінок пар, що містять поняття **інших розділів інформатики** у випадку порівняння кількісних результатів анкетування групи експертів з кількісними результатами анкетування експериментальної групи складає $\chi^2_{емп}$= **10,2 ≤ 11,07**. У випадку порівняння вибірок «Експертів» з анкетування контрольної групи $\chi^2$ = **12,92 > 11,07**. Дані результати також вказують на значно менші відмінності в оцінці понять інших розділів інформатики між експериментальною групою та групою експертів, ніж вони є між контрольною та експертною групами.

Емпіричні значень $\chi^2$ для заданих результатів подано у табл. 3.9.

*Таблиця 3.9.*

| $\chi^2$ | Експерти | Експеримент | Контроль |
|---|---|---|---|
| Експерти | 0,00 | 10,2 | 12,92 |
| Експеримент | 10,2 | 0,00 | 8,72 |
| Контроль | 12,92 | 8,72 | 0,00 |

Аналіз оцінок пар, що містять **мало причетні до ІКТ** у випадку порівняння кількісних результатів анкетування групи експертів з кількісними результатами анкетування експериментальної групи, складає $\chi^2_{емп}$= **9,73 ≤ 11,07**. У випадку порівняння вибірок «Експертів» з анкетування контрольної групи $\chi^2$ = **12,57.** Дані результати виявили менші відмінності в оцінці понять, не

причетних до ІКТ між експериментальною групою та групою експертів, ніж вони є між контрольною та експертною групами.

Емпіричні значень $\chi^2$ для заданих результатів подано у табл. 3.10.

*Таблиця 3.10.*

| $X^2$ | Експерти | Експеримент | Контроль |
|---|---|---|---|
| Експерти | 0,00 | 9,73 | 12,57 |
| Експеримент | 9,73 | 0,00 | 8,34 |
| Контроль | 12,57 | 8,34 | 0,00 |

### 3.2.5. Остаточні результати контрольного експерименту

Узагальнити та проаналізувати результати контрольного експерименту дозволяє діаграма, подана на рис. 3.8. На даній діаграмі вертикальна вісь відображає значення критерію однорідності $\chi^2$ для всіх п'яти вибірок пар понять з найменшим середнім квадратичним значенням відхилення по групі «експертів».

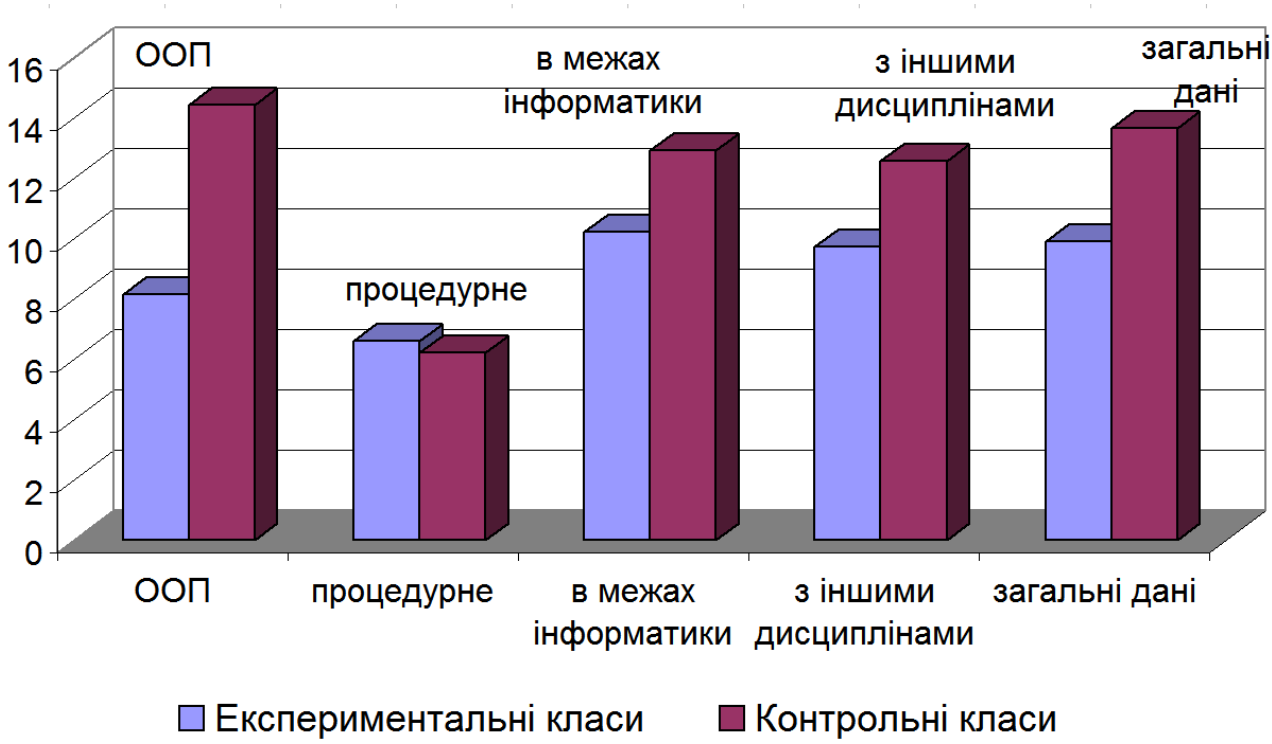



Рис. 3.8. Значення критерію однорідності для порівняння оцінок експериментальних та контрольних груп з оцінками експертів

Згідно даних діаграми значні відмінності в оцінці між експериментальною групою та групою експертів виявлено лише стосовно понять з галузі процедурного програмування. Між контрольною групою та групою експертів розбіжності виявлено в усіх п’яти вибірках. Це підтверджує гіпотезу дослідження, а також дозволяє зробити висновки про загальні переваги навчання програмування мовою C# в класах технологічного профілю.

Результати опрацювання статистичних даних, зібраних у ході проведення експериментального навчання, свідчать про його ефективність. Дані анкетування учнів експериментальних класів відносно даних, отриманих під час анкетування у контрольних класах, значно менше відрізняються від тих, що дали у своїх анкетах експерти. Це дозволяє зробити висновки про загальні переваги навчання програмування мовою C#. Поряд з тим, результати експериментального визначення ефективності впровадження до навчання мови програмування C# вказують на переваги використання цієї мови для навчання учнів на основі об’єктно-орієнтованої парадигми програмування.

## Висновки до розділу 3

У процесі педексперименту вивчено проблему перевірки ефективності впровадження новітніх технологій розробки програмного забезпечення у процес навчання програмування. З'ясовано необхідність застосування особливих методик такого дослідження, що виключають вплив особливостей мови та парадигми програмування на результати дослідження.

Вдалося адаптувати один з методів проективних технологій психологічних досліджень, а саме: метод часткового семантичного диференціалу для визначення ефективності навчання програмування. З використанням розробленого різновиду методу СД проведено порівняння якості навчання програмування мовою C# та іншими мовами програмування в класах технологічного профілю загальноосвітніх навчальних закладів.

Результати, що виявлено під час анкетування учнів експериментальних класів, значно менше відрізняються від тих, що дали в своїх анкетах експерти. Це дозволяє зробити висновки про загальні переваги навчання програмування мовою C#. Поряд з тим, результати експериментального визначення ефективності впровадження до навчання мови програмування C# вказують на переваги використання цієї мови для навчання учнів на основі об'єктно-орієнтованої парадигми програмування.

Запропонований варіант використання семантичного диференціалу для перевірки ефективності впровадження навчання програмування на основі мови C# все ще має окремі недоліки: анкети громіздкі та потребують багато часу на їх заповнення, надзвичайно трудомісткою є процедура переведення даних анкетування, що здійснювалось на окремих аркушах до електронного вигляду.

Проблема експериментального визначення доцільності та ефективності впровадження інновацій до процесу навчання залишається надзвичайно актуальною. Існує потреба подальшої розробки та удосконалення проективних технологій для перевірки ефективності впровадження до навчання перспективних мов та парадигм програмування.

Основні результати розділу 3 опубліковані в роботах [300; 301; 311; 313].

# ВИСНОВКИ

У процесі проведення дисертаційного дослідження було вирішено усі поставлені завдання і відповідно до мети та висунутої гіпотези отримано такі **результати**:

1. Проведено аналіз, систематизацію, узагальнення наукової вітчизняної та зарубіжної фахової, педагогічної та навчально-методичної літератури, науково узагальнено досвід вчителів інформатики загальноосвітніх навчальних закладів та з'ясовано:

– психолого-педагогічні особливості організації шкільного навчально-виховного процесу за умов профілізації освіти та концептуальні потреби алгоритмічної підготовки учнів у класах технологічного профілю;

– особливості організації та здійснення навчання інформатики в цілому та навчання програмування, зокрема у профільних класах загальноосвітніх навчальних закладів;

– стан дослідження проблеми використання середовища .NET та мови C# в навчанні програмування, особливості їх застосування для навчання;

– програмно-технологічні можливості використання мови C# для навчання програмування у ЗНЗ.

2. Визначено сутність понять, якими мають оволодівати учні під час навчання програмування мовою C# та педагогічні умови впровадження мови C# у шкільний курс інформатики.

3. Визначено і створено основні компоненти методичної системи навчання програмування учнів класів технологічного профілю на основі використання мови C#, в межах якої забезпечується впровадження об'єктно-орієнтованої парадигми програмування.

4. У ході педагогічного експерименту перевірено ефективність методики навчання програмування на основі використання мови C# в класах технологічного профілю.

5. Розроблено та апробовано навчально-методичні матеріали для курсу інформатики загальноосвітніх навчальних закладів, а саме:

– розроблено методичні рекомендації щодо навчання програмування на основі використання мови C# в класах технологічного профілю;

– удосконалено та адаптовано до навчання програмування мовою C# в класах технологічного профілю загальноосвітніх навчальних закладів рекомендоване календарне планування згідно програми інформатики для класів технологічного профілю;

– розроблено практичні завдання, перелік тестових питань, завдання для перевірки знань учнів;

– для методичної підтримки навчання програмування мовою C# в класах технологічного профілю створено сайт https://sites.google.com/site/c4plus/ який, водночас, використовувався для інформаційної підтримки педагогічного експерименту.

Отримані результати дисертаційного дослідження дозволяють зробити такі **висновки**:

1. В умовах випереджаючого розвитку ІКТ, що здійснюється на основі новітніх технологій програмування, а також задля забезпечення якісного впровадження профільного навчання особливого значення набуває проблема удосконалення педагогічних технологій навчання інформатики в загальноосвітніх навчальних закладах. Навчання програмування є особливою складовою оволодіння учнями інформаційно-комунікаційними технологіями. Разом з тим виявлено суттєві проблеми в галузі навчання програмування на уроках інформатики загальноосвітніх навчальних закладів. Зокрема, спостерігається значне відставання галузі навчання програмування з огляду на впровадження новітніх технологій розробки програмного забезпечення, недостатньо розробленими залишаються технології навчання програмування на основі новітніх мов та середовищ розробки програмного забезпечення.

2. На основі аналізу сучасних тенденцій в галузі розробки програмного забезпечення визначено пріоритетність об’єктно-орієнтованої парадигми програмування для використання під час навчання. ООП дозволяє знайомити учнів з сучасними принципами розробки програмного забезпечення; покращує змістовні зв’язки різних розділів інформатики з розділом «Алгоритми та програми»; підсилює міжпредметні зв’язки інформатики з іншими шкільними дисциплінами. Однією з широко розповсюджених та перспективних мов, на основі якої доцільно здійснювати навчання об’єктно-орієнтованого програмування, є мова C#.

3. Обґрунтована й апробована під час дослідно-експериментальної роботи модель навчання програмування учнів класів технологічного профілю загальноосвітніх навчальних закладів на основі використання мови C# являє собою структурно-функціональну конструкцію, у якій фігурують взаємопов’язані та взаємозумовлені між собою компоненти. Ці компоненти визначають сутність процесу навчання програмування, як результат взаємодії професійної, алгоритмічної й особистісної складових, підкріплених відповідними програмно-технологічними можливостями навчальних закладів та необхідною підготовкою вчителів інформатики.

4. Розроблена й експериментально перевірена методика навчання програмування на основі використання мови C# в класах технологічного профілю загальноосвітніх навчальних закладів розглядається як інтегративний освітній процес, спрямований на досягнення запланованих результатів на основі діагностики поточного стану, моніторингу загальних і спеціальних навчальних умінь і навичок учнів. Технологія навчання програмування на основі використання мови C# учнів класів технологічного профілю загальноосвітніх навчальних закладів представлена послідовністю етапів: визначення цілей навчання з урахуванням вікових особливостей і базової підготовки учнів; створення і вирішення проблемних навчальних ситуацій, які вимагають навчальної активності, самоосвіти, розвиваючого мислення та інших

інтелектуальних здібностей учнів; організація різних видів творчої діяльності; забезпечення стійкої мотивації до засвоєння знань, умінь і навичок; створення емоційного фону в оволодінні учнями ІКТ; моніторинг навчальної діяльності учнів.

5. У ході дослідження з урахуванням вітчизняного і зарубіжного досвіду було виявлено й експериментально перевірено педагогічні умови, що впливають на ефективність навчання програмування. Виконане дослідження підтвердило висунуту гіпотезу про ефективність методики навчання програмування на основі використання мови C# учнів класів технологічного профілю загальноосвітніх навчальних закладів. Зокрема, впровадження мови C# до навчання в класах технологічного профілю дозволило: забезпечити теоретичне та практичне знайомство з елементами ООП учнів класів технологічного профілю під час навчання програмування; покращити узгодженість, внутріпредметні зв’язки, змістову єдність розділу алгоритми та програми з іншими розділами шкільної інформатики; покращити узгодженість, міжпредметні зв’язки, змістову єдність інформатики з іншими предметами, що вивчаються в класах технологічного профілю загальноосвітніх навчальних закладів.

6. Проведене дослідження не вичерпує всіх аспектів проблеми навчання програмування на основі використання мови C# учнів класів технологічного профілю загальноосвітніх навчальних закладів. До напрямків, що потребують подальшого дослідження, віднесено: необхідність конкретизації та наповнення змісту всіх структурних елементів методики навчання програмування, вивчення механізму впливу навчання програмування на особистісну самореалізацію учня у професійній діяльності, визначення можливостей впровадження програмування на основі мови C# до науково-технічної творчості учнів.

# СПИСОК ВИКОРИСТВНИХ ДЖЕРЕЛ


1. Авраменко М. М. Профільне навчання в середній школі Федеративної Республіки Німеччини: дис. ... кандидата пед. наук : 13.00.01 / Мирослава Миколаївна. Авраменко; Ін-т педагогіки АПН України. . – К., 2007. – 221 с.
2. Агапова И. Ю. Восприятие рекламы: методика использования репертуарных решеток для формирования биполярных шкал семантического дифференциала / И. Ю. Агапова // Социология: методология, методы и математическое моделирование. – М., 1999. – № 11. – С. 73–100.
3. Анисимов А. В. Информатика. Творчество. Рекурсия. / А. В. Анисимов – К.: Наукова думка, 1988. – 224 с.
4. Анисимов А. В. Компьютерная лингвистика для всех: Мифы. Алгоритмы. Язык / А. В. Анисимов – К.: Наукова думка, 1991. – 208 с.: ил.
5. Артемьева Е. Ю. Вероятностные методы в психологии. / Е. М. Мартынов, Е. Ю. Артемьева – М.: Изд-во Моск. ун-та, 1975. – 206 с.
6. Артемьева Е. Ю. Основы психологии субъективной семантики / Е. Ю. Артемьева; под ред. И. Б. Ханиной. М.: Наука; Смысл, 1999. – 350 с.
7. Арчер Т. Основы C#. Новейшие технологии. / Арчер Том ; [пер. с англ.]. – М.: Издательско-торговый дом «Русская Редакция», 2001. – 448 с.
8. Асмус В. Ф. Платон. Серия «Мыслители прошлого». / В. Ф. Асмус / – М.: Мысль.2-е изд. 1975. – 223 с.
9. Ахмечет В. Функциональное программирование для всех [Электронный ресурс] / В. Ахмечет; [пер. с англ.] // RSDN Mag. – 2006. – № 2. – С. 11–22. — Режим доступа к журн. : http://www.rsdn.ru/article/funcprog/fp.xml. – Назва з екрана.

10. Ахо А. В. Построение и анализ вычислительных алгоритмов / А. В. Ахо, Д. Э. Хопкрофт, Д. Д. Ульман; [пер. с англ.]. – М.: Издательство «Мир», 1978. – 535 с.
11. Ахо А. В. Структуры данных и алгоритмы / А. В. Ахо , Д. Э. Хопкрофт, Д. Д. Ульман; [пер. с англ.]. – М. : Издательский дом «Вильямс», 2001. – 384 с.
12. Бабанский Ю. К. Дидактические проблемы совершенствования учебных комплексов / Ю. К. Бабанский. – М.: Просвещение, 1980. – С. 17–33.
13. Бабанский Ю. К. Методы обучения в современной общеобразовательной школе / Ю. К. Бабанский. – М.: Просвещение, 1985. – 208 с.
14. Бабанский Ю. К. Оптимизация процесса обучения. Общедидактический аспект / Бабанский Ю. К. – М. : Педагогика, 1977. – 256 с.
15. Бабанский Ю. К. Рациональная организация учебной деятельности. / Ю. К. Бабанский. – М. : Знание, 1981. – 96 с.
16. Бадд Т. Объектно-ориентированное программирование в действии / Тимоти Бадд ; [пер. с англ.]. – СПб.: «Питер», 1997. – 464 с.
17. Бен-Ари М. Языки программирования. Сравнительный анализ / М. Бен-Ари ; [пер. с англ.]. – М. : Мир, 2000. – 366 с.
18. Бентли Дж. Жемчужины программирования / Дж. Бентли ; [пер. с англ.]. – СПб. : Питер, 2002. – 272 с.
19. Бешенков С. А. Дидактические основы профильного обучения информатике [Текст]: автореф. дис. д-ра пед. наук / С. А. Бешенков – М.: ИОШ РАО, 1993. – 90 с.
20. Биков В. Ю. Навчальна програма з інформатики для 8-11 класів загальноосвітніх навчальних закладів універсального та фізико-математичного профілю./ В. Ю. Биков, В. Д. Руденко // Комп’ютер у школі та сім'ї – 2005, №1. С. 17–33.
21. Биков В. Ю. Технологія розробки дистанційного курсу: Навчальний посібник. / В. Ю. Биков, В. М. Кухаренко, Н. Г. Сиротенко, О. В. Рибалко,

Ю. М. Богачков – [За ред. В. Ю. Бикова та В. М. Кухаренка]. – К:, Міленіум 2008. – 324 с.

22. Биллиг В. А. Основы программирования на C#. [Електронний ресурс] / В. А. Биллиг // Интернет-университет информационных технологий. «ИНТУИТ-РУ». – режим доступу: URL : http://www.intuit.ru/department/pl/csharp/. – Назва з екрана.
23. Блонский П. П. Память и мышление / П. П. Блонский. – СПб.: Питер, 2001. – 357 с.
24. Бобров Е. Г. Карл Линней. 1707–1778. / Е. Г. Бобров – Л.: Наука. 1970. – 285 с.
25. Бондар С. П. Модернізація методів навчання у профільній школі [Текст] / С. П. Бондар // Вісник Житомирського державного університету імені Івана Франка. – 2010. – N 53. – С. 56–61.
26. Бондарев В. М. Основы программирования / В. М. Бондарев, В. И. Рублинецкий, Е. Г. Качко – Харьков: Фолио, 1998. – 368 с.
27. Бондаренко М. А. Особливості навчання об’єктно-орієнтованого програмування майбутніх інженерів-педагогів / М. А. Бондаренко // Інформатика та інформаційні технології в навчальних закладах. – 2009. – №1. – С. 46–50.
28. Брукс Ф. Мифический человеко-месяц или Как создаются программные системы / Ф. Брукс. – М.: Символ-Плюс, 2010. – 304 с.
29. Бурлакова Н. С. Проективные методы: теория, практика применения к исследованию личности ребенка / Н. С. Бурлакова, В. И. Олешкевич. – М.: Институт общегуманитарных исследований, 2001. – Ч. 1. – 352 с.
30. Бурлачук Л. Ф. Психодиагностика / Л. Ф. Бурлачук // – СПб.: Питер, 2002. – 352 с.
31. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++ / Гради Буч ; [пер. с англ.]. – М. : Издательство БИНОМ, СПб. : Невский диалект, 1999. – 560 с.

32. Ватсон Б. C# 4.0 на примерах. / Бен Ватсон – СПб. : БХВ-Петербург, 2011. – 608 с.: ил. – ISBN: 978-5-9775-0608-3.
33. Ващук Б. В. Модульне програмування [Електронний ресурс] : Основи програмування. Електронний підручник для вищих навчальних закладів I-II рівня акредитації / Б. В. Ващук – Режим доступу : URL : http://vvpc.com.ua/tests_informatiks/inform/osnovy_program/ku.htm.
34. Вернигоренко С. А. Вивчення основ об’єктно-орієнтованого програмування у класах фізико-математичного профілю / С. А. Вернигоренко // Комп'ютерно-орієнтовані системи навчання: збірник наукових праць. М-во освіти і науки України, НПУ ім. М. П. Драгоманова; Відп. ред. М. І. Жалдак. – Київ, 2011. – Вип. 12 – С. 61–74.
35. Вернигоренко С. А. Використання засобів об'єктно-орієнтованого програмування для розвитку пізнавальної активності учнів / С. А. Вернигоренко // Комп'ютерно-орієнтовані системи навчання: збірник наукових праць. М-во освіти і науки України, НПУ ім. М. П. Драгоманова; Відп. ред. М. І. Жалдак. – Київ, 2005. – Вип. 9 – С. 182–197.
36. Вирт Н. Алгоритмы + структуры данных = программы / Н. Вирт ; [пер. с англ.]. – М. : Мир, 1985. – 406 с.
37. Вирт Н. Программирование на языке Модула-2 / Н. Вирт ; [пер. с англ.] // – М. : Мир, 1987. – 224 с.
38. Волкова Н. П. Педагогіка: [посібник для студентів вищих навчальних закладів] / Н. П. Волкова – К. : Видавничий центр «Академія» , 2001. – 576 с.
39. Володіна І. Л. Інформатика: підруч. для 10 кл. загальноосвіт. навч. закладів: рівень стандарту: у 2 ч. / І. Л. Володіна, В. В. Володін. – Х. : Гімназія, 2010. – Ч.1. – 352 с. : іл. ISBN 978-966-474-110-8.
40. Володіна І. Л. Основи інформатики, 7 клас: навчальний посібник / І. Л. Володіна, В. В. Володін, Ю. О. Дорошенко, Ю. О. Столяров. – Х. : Гімназія, 2007. – 230 с.

41. Волошинов С. А. Алгоритмічна підготовка судноводіїв в умовах інформаційно-комунікаційного педагогічного середовища / С. А. Волошинов // Інформаційні технології в освіті. – 2010. – № 8. – С. 103–108.
42. Вольянська С. Є. Організація профільного навчання в загальноосвітній школі в умовах регіону: дис. … кандидата пед. наук : 13.00.01 / Світлана Євгенівна Вольянська ; ХНПУ імені Г. С. Сковороди. – Харків., 2006. – 164 с.
43. Вольянська С. Є. Організація профільного навчання в загальноосвітній школі в умовах регіону: автореферат дис. канд. пед. наук. 13.00.01 / С. Є. Вольянська ; ХНПУ імені Г. С. Сковороди, Харків., 2006. – 15 с.
44. Всемирная энциклопедия: Философия / [Главн. науч. ред. и сост. А. А. Грицанов] М.: АСТ, Мн.: Харвест, Современный литератор, 2001. – 1312 с.
45. Выготский Л. С. Избранные психологические произведения / Л. С. Выготский – М. : Учпедгиз, 1956. – 426 с.
46. Выготский Л. С. Мышление и речь. Изд. 5, испр. / Л. С. Выготский – М.: Лабиринт, 1999. – 352 с.
47. Выготский Л. С. Педагогическая психология / Л. С. Выготский ; под ред. В. В. Давыдова. – М.: Педагогика, 1999. – 536 с.
48. Габрусєв В. Ю. Зміст і методика вивчення шкільного курсу інформатики на основі вільно поширюваної операційної системи Linux: дис. … кандидата пед. наук : 13.00.02 / Валерій Юрійович Габрусєв / НПУ імені М. П. Драгоманова, – К., 2003. – 221 с.
49. Гаевский А. Ю. Информатика: 7-11 кл.: Учеб. пособие. / А. Ю. Гаевский – [2-е изд. доп.]. – К.: А. С. К., 2006. – 536 с., ил.
50. Галыгина Л. В. Изучение информационных и коммуникационных технологий в профильных курсах информатики: дис. … кандидата пед. наук : 13.00.02 / Лилия Владимировна Галыгина – М., 2001. – 190 с.

51. Гальперин П. Я. Развитие исследований по формированию умственных действий / П. Я. Гальперин // Психологическая наука в СССР. – М.: Изд-во АПН РСФСР, 1959. – Т. 1. – С. 441–469.
52. Герович В. А. Интер-Нет! Почему в Советском Союзе не была создана общенациональная компьютерная сеть / В. А. Герович «Неприкосновенный запас» 2011, №1(75). [Електронний ресурс]. – режим доступу: URL : http://www.intelros.ru/readroom/nz/nz-75-1-2011/8691-inter-net-pochemu-v-sovetskom-soyuze-ne-byla-sozdana-obshhenacionalnaya-kompyuternaya-set.html. – Назва з екрана.
53. Гиппенрейтер Ю. Б. Введение в общую психологию. Курс лекций. / Ю. Б. Гиппенрейтер – М. : АСТ, – 1988. – 352с.
54. Глинський Я. М. Інформатика: 10-11 класи: Навч. посіб.: У 2 ч. – ч. 1.: Алгоритмізація й програмування. / Я. М. Глинський / 7-ме вид. – Львів: СПД Глинський, 2007. – 56 с.
55. Глинський Я. М. Переваги застосування мови програмування JAVA в навчальному процесі / Я. М. Глинський, В. Є. Анохін, В. А. Ряжська. // Інформатика та інформаційні технології в навчальних закладах : Науково-методичний журнал. – К. : Освіта України, – 2005. – № 4 – С. 34–38.
56. Глушков В. М. Теория автоматов и формальные преобразования микропрограмм / В. М. Глушков // Кибернетика, 1965. – № 5. – С. 1–10.
57. Голуб Ален И. С и С++. Правила программирования / Ален И. Голуб // – М.: БИНОМ, 1996. – 272 с.
58. Голубев Н. К. Диагностика и прогнозирование воспитательного процесса / Н. К. Голубев. – Л.: ЛГПИ им. А. И. Герцена, 1988. – 86 с.
59. Гончарова О. Н. Теоретико-методические основы личностно-ориентированной системы формирования информатических компетентностей студентов экономических специальностей : дис. … доктора пед. наук. 13.00.02 / Оксана Николаевна Гончарова –

Симферополь: Таврический Национальний Университет им. В. И. Вернадского, 2007. – 471с.

60. Горошко Ю. В. Методика навчання інформатики - історія і перспективи / Ю. В. Горошко, Г. Ю. Цибко // Науковий часопис Національного педагогічного університету імені М. П. Драгоманова. Серія 2, Комп'ютерно-орієнтовані системи навчання / М-во освіти і науки України, Нац. пед. ун-т ім. М. П. Драгоманова. – К. : Вид-во НПУ ім. М. П. Драгоманова, 2010. – Вип. 9 (16). – С. 98–102.
61. Горошко Ю. В. Міжпредметні зв’язки інформатики з математикою та фізикою у навчанні майбутнього інженера / Ю. В. Горошко, Д. А. Покришень // Інформаційні технології і засоби навчання: електронне наукове фахове видання [Електронний ресурс] – 2009. – №1(9). – Режим доступу до журналу http://www.nbuv.gov.ua/e-journals/ITZN/em9/emg.html.
62. Горошко Ю. В. Проблеми та особливості впровадження вільного програмного забезпечення в навчальний процес / Ю. В. Горошко, А. О. Костюченко, М. І. Шкардибарда // Комп'ютер у школі та сім'ї. – 2010. – № 7. – С. 8–10.
63. Гришко Л. В. Вимоги до професійних якостей програміста / Л. В. Гришко // Вісник Черкаського університету, серія «Прикладна математика. Інформатика». Випуск 143 – Черкаси, 2010. – С. 116–120.
64. Гришко Л. В. Концептуальні підходи до навчання основ програмування у вищій школі / Л. В. Гришко // Комп'ютерно-орієнтовані системи навчання: збірник наукових праць. М-во освіти і науки України, НПУ ім. М. П. Драгоманова; Відп. ред. М. І. Жалдак. – Київ, 2004. – Вип. 8 – С. 134–47.
65. Гришко Л. В. Навчання стилю програмування, як складова формування професійної культури майбутнього інженера-програміста / Л. В. Гришко // Вісник Черкаського університету, серія «Педагогічні науки». Випуск 143. – Черкаси, 2009. – С. 37–43.

66. Грэм Пол. Языки программирования через сто лет / Пол Грэм / Компьютерра online 03 августа 2004 года. [Електронний ресурс]. – режим доступу: URL : http://www.computerra.ru/hitech/35042/. – Назва з екрана.
67. Давыдова Н. А. Технология формирования содержания образования по информатике в профильных классах общеобразовательных школ: дис. … кандидата пед. наук / Надежда Алексеевна Давыдова. – Челябинск., 2001. – 239 с.
68. Дал У. Структурное программирование: Пер. с англ. / У. Дал, Э. Дейкстра, К. Хоор – М.: Издательство «Мир», 1975. – 245 с.
69. Дейкстра Э. Заметки по структурному программированию // У. Дал, Э. Дейкстра, К. Хоор / Структурное программирование. – М.: Мир, 1975. – 247 с.
70. Дейкстра Э. Дисциплина программирования: Пер. с англ. / Э. Дейкстра – М.: Издательство «Мир», 1978. – 274 с.
71. Державний стандарт загальної середньої освіти в Україні. Інформатика. Освітня галузь «Технології» – К., Освіта України, 2003. – 123с.
72. Дистанційне навчання — теорія і практика. [Електронний ресурс] // Веб-сайт «Портал знань». – режим доступу: URL : http://www.znannya.org/?view=e-learning. – Назва з екрана.
73. Дорошенко Ю. О. Програма курсу за вибором «Основи Інтернету» / Ю. О. Дорошенко, І. О. Завадський, Н. С. Прокопенко // Інформатика та інформаційні технології в навчальних закладах. – 2006. – № 4/5. – С. 41–48.
74. Дорошенко Ю. О. Програма курсу за вибором «Основи комп’ютерної графіки» / Ю. О. Дорошенко, І. О. Завадський // Інформатика та інформаційні технології в навчальних закладах. – 2006. – № 4/5. – С. 27–34.
75. Дорошенко Ю. О., Навчання інформатики у структурі 12-річної загальної середньої освіти / Ю. О. Дорошенко, Н. С. Прокопенко // Інформатика та інформаційні технології в навчальних закладах. – 2006. – № 1. – С. 55–72.

76. Ершов А. П. Школьная информатика: концепции, состояния, перспективы // А. П. Ершов, Г. А. Звенигородский, Ю. А. Первин [препринт №152 ВЦ СО АН СССР, Новосибирск, 1979] Архив академика А. П. Ершова [Электронный ресурс]. – 12.09.2011. – Режим доступа: http://ershov.iis.nsk.su/archive/eaindex.asp?lang=1&did=41919.
77. Ершов А. П. Программирование – вторая грамотность // Архив академика А. П. Ершова [Электронный ресурс]. – 12.09.2011. – Режим доступа: http://ershov.iis.nsk.su/russian/second_literacy/article.html. – Назва з екрана.
78. Ершов А. П. Компьютеризация школы и математическое образование./ А. П. Ершов // Программирование, № 1. – 1990. – С. 3–25.
79. Ершов А. П. О человеческом и эстетическом факторах в программировании // История информатики в России: ученые и их школы / [сост.: В. И. Захаров, Р. И. Подловченко, Я. И. Фет.] – М.: Наука, 2003. – С. 291–298.
80. Жалдак М. І. «Основи інформатики» як одна з вагомих складових системи навчальних предметів загальноосвітньої школи / М. І. Жалдак, Н. В. Морзе, Ю. С. Рамський // Сучасні інформаційні технології в навчальному процесі: [зб. наук. праць] – К: НПУ, 1997. – С. 3–21.
81. Жалдак М. І. Про деякі методичні аспекти навчання інформатики в школі і педагогічному університеті / М. І. Жалдак. // Комп’ютерно-орієнтовані системи навчання / НПУ ім. М. П. Драгоманова. – 2005. – Вип. 9. – С. 3–4.
82. Жалдак М. И. Система подготовки учителя к использованию информационной технологии в учебном процессе : дис. в форме науч. доклада доктора пед. наук : 13.00.02 / Мирослав Иванович Жалдак; АПН СССР; НИИ содержания и методов обучения. – М., 1989. – 48 с.
83. Жалдак М. І . Профільне навчання інформатики / М. І. Жалдак, Н. В. Морзе, О. Г. Кузьмінська / Комп'ютерно-орієнтовані системи навчання: збірник наукових праць. М-во освіти і науки України, НПУ

ім. М. П. Драгоманова; Відп. редактор М. І. Жалдак. – Київ, 2004. – Вип. 8 – С. 3–14.

84. Жалдак М. І. Изучение языков программирования в школе. / М. І. Шкіль, М. І. Жалдак, Н. В. Морзе, Ю. С. Рамський. // – Київ.: «Радянська школа». 1988. – 272 с.
85. Жалдак М. І. Інформатика: Навчальний посібник / М. І. Жалдак, Ю. С. Рамський / [за ред. М. І. Шкіля] – К.: Вища школа, 1991. – 319 с.
86. Жалдак М. І. Комп’ютерно орієнтовані засоби навчання математики, фізики, інформатики : [посібник для вчителів] / М. І. Жалдак, В. В. Лапінський, М. І. Шут – К. : НПУ імені М. П. Драгоманова, 2004. – 182 с.
87. Жалдак М. І. Яким бути шкільному курсу інформатики / М. І. Жалдак. // Комп’ютер в школі та сім’ї. – 1988. – №1. – С. 3–8.
88. Жук Ю. О. Засоби навчання як параметр освітнього простору / Ю. О. Жук // Фізика та астрономія в школі. – 2003. – № 4 – С. 13–18.
89. Жук Ю. О. Концепція створення засобів навчання нового покоління для середніх закладів освіти України / О. Я. Савченко, А. М. Гуржій, В. М. Доній, В. П. Волинський, Ю. О. Жук, В. В. Самсонов, М. І. Шут, // Проблеми освіти : наук.-метод. зб. – Київ, 1997. – Вип. 10. – С. 207–218.
90. Жук Ю. О. Роль засобів навчання у формуванні навчального середовища / Ю. О. Жук // Нові технології навчання. – 1998. – № 22. – С. 106–112.
91. Жуковський С. С. «E-Olimp» – система автоматичної перевірки задач та проведення олімпіад з інформатики в Інтернеті [Текст] / С. С. Жуковський // Комп'ютер у школі та сім'ї : Науково-методичний журнал. – 2008. – N1. – С. 48–50.
92. Жуковський С. С. Аналіз, дослідження та розв’язування конкурсних задач під час учнівської олімпіади з інформатики [Текст] / С. С. Жуковський // Вісник Житомирського державного університету імені Івана Франка. – 2010. – N 52. – С. 153–157.

93. Жуковський С. С. Використання Інтернет-порталу організаційно-методичного забезпечення «E-OLIMP» для підготовки обдарованих школярів до олімпіади з інформатики [Текст] / С. С. Жуковський // Комп'ютер у школі та сім'ї : Науково-методичний журнал. – 2010. – N 8. – С. 47–48.
94. Забарна А. П. Базова модель методичної системи навчання інформатики в загальноосвітніх навчальних закладах природничо-математичного напрямку профілізації / А. П. Забарна // Інформатика та інформаційні технології в навчальних закладах. – 2008. – №6. – С. 27–31.
95. Забарна А. П. Компетентнісний підхід як основа організації профільного навчання інформатики. / А. П. Забарна // Комп'ютер у школі та сім'ї. – 2009. – №2(74). – С. 21–25.
96. Завадський І. О Курси за вибором з інформатики в системі профільної освіти / І. О. Завадський // Комп'ютер у школі та сім'ї. – 2008. – №7(71). – С. 3–4.
97. Завадський І. О. Інформатика. Навчальна програма для 10-11 класів загальноосвітніх навчальних закладів. Рівень стандарту / І. О. Завадський, Ю. О. Дорошенко, Ж. В. Потапова // Інформатика та інформаційні технології в навчальних закладах. – 2010. – №5-6. – С. 20–33.
98. Завадський І. О. Інформатика. Навчальна програма для 10-11 класів загальноосвітніх навчальних закладів. Академічний рівень / І. О. Завадський, Ю. О. Дорошенко, Ж. В. Потапова // Інформатика та інформаційні технології в навчальних закладах. – 2010. – № 5-6. – С. 34–50.
99. Завадський І. О. Навчальна програма з інформатики для 9-12 класів загальноосвітніх навчальних закладів. Рівень стандарту / І. О. Завадський, Ж. В. Потапова, Ю. О. Дорошенко // Інформатика та інформаційні технології в навчальних закладах. – 2008. – № 2-3. – С. 8–35.

100. Завадський І. О. Навчальна програма з інформатики для 9-12 класів загальноосвітніх навчальних закладів. Академічний рівень / І. О. Завадський, Ж. В. Потапова, Ю. О. Дорошенко // Інформатика та інформаційні технології в навчальних закладах. – 2008. – № 2-3. – С.37–67.
101. Завадський І. О. Основи візуального програмування / І. О. Завадський, Р. І. Заболотний : [Навч. Посіб.]. – К.: Вид. група BHV. – 2008. – 272 с.: іл.
102. Завадський І. О. Програма курсу за вибором «Основи веб-дизайну» / І. О. Завадський, Н. С. Прокопенко, Т. Г. Проценко // Інформатика та інформаційні технології в навчальних закладах. – 2006. – № 4/5. – С.48–55.
103. Завадський І. О. Програма курсу за вибором «Основи візуального програмування / І. О. Завадський // Комп'ютер у школі та сім'ї. – 2008. – №7(71). – С. 4–10.
104. Завадський І. О. Програма курсу за вибором «Основи створення комп’ютерних презентацій» / І. О. Завадський, Н. С. Прокопенко, Т. Г. Проценко // Інформатика та інформаційні технології в навчальних закладах. – 2006. – № 4/5. – С. 35–40.
105. Завдання для тематичного оцінювання з інформатики (10-11 класи) [Збірник] / М. К. Чоба, Л. В. Палюшок, Л. М. Кос, Г. Ю. Сабор та ін. – Львів: Астон, 2003. – 80с.
106. Закон України «Про загальну середню освіту» // Інформаційний збірник Міністерства освіти України. – 1999. – № 15. – С. 6–31.
107. Закон України «Про основні засади розвитку інформаційного суспільства в Україні на 2007–2015 роки» від 9 січня 2007 року № 537-V.// Відомості Верховної Ради України (ВВР) – 2007. – №12. – С.102. [Електронний ресурс]. – режим доступу: URL : http://zakon.rada.gov.ua/cgi-bin/laws/main.cgi. – Назва з екрана.
108. Закон України про авторське право і суміжні права. К.: Парламентське видавництво, 1998. – 31 с.

109. Зарецька І. Т. Інформатика: Навч. посібн. для 10–11 кл. середн. загальноосвітн. шкіл / І. Т. Зарецька, Б. Г. Колодяжний, А. М. Гуржій, О. Ю. Соколов. – К.: Навчальна книга, 2002. – 496 с.: іл.
110. Захарова Т. Б. Профильная дифференциация обучения информатике на старшей ступени школы: дис. … доктора пед. наук : 13.00.02 / Татьяна Борисовна Захарова – М.: ИОСО, 1997. – 456 с.
111. Зеленяк О. П. Користувацький ухил затягнувся недозволено довго [Текст] / О. П. Зеленяк // Комп'ютер у школі та сім'ї : Науково-методичний журнал. – 2010. – N 2. – С. 7–8.
112. Зеленяк О. П. Практикум программирования на Turbo Pascal. Задачи, алгоритмы и решения. / О. П. Зеленяк – К.: Издательство «Диасофт», 2001. – 320 с.
113. Зубов В. С. Программирование на языке TURBO PASCAL (версии 6.0 и 7.0). / В. С. Зубов Издание 3-е, исправленное – М.: Информационно-издательский Дом «Филинъ», 1997. – 320 с.
114. Интерактивная система бально-рейтингового контроля знаний: монография / Под ред. Б. Г. Бобылева, П. И. Образцова. – Орел: ОрелГТУ, 2007. – 182 с.
115. Інструктивно-методичні рекомендації щодо вивчення Інформатики в загальноосвітніх навчальних закладах у 2008/09 навчальному році. // Комп’ютер у школі та сім'ї – 2008. – №3. – С. 6–12.
116. Інформатика. Програми для загальноосвітніх навчальних закладів / За ред. акад. М. І. Жалдака. – Запоріжжя: Прем'єр, 2003. – 304 с.
117. Караванова Т. П. Навчальна програма поглибленого вивчення інформатики для учнів 8-12 класів ЗНЗ (напрям: технологічний, профіль: інформаційно-технологічний) / Т. П. Караванова, В. П. Костюков // Інформатика та інформаційні технології в навчальних закладах. – 2008. – №2. – С. 5–11.

118. Караванова Т. П. Основи алгоритмізації та програмування. 750 задач з рекомендаціями та прикладами. Посібник. – К.: ТОВ. «Форум». – 2001. – 286 с.
119. Кащеєв Я. Б. Збірник практичних завдань з інформатики. Pbrush, Word. / Я. Б. Кащеєв , Г. І. Кащеєва – Х.: Торсінг, 2003. – 40 с.
120. Кирютенко Ю. А. Объектно-ориентированное программирование. Язык Smalltalk / Ю. А. Кирютенко, В. А. Савельев – М. : Вузовская книга, 2007. – 328 с.
121. Клаус Г. Введение в дифференциальную психологию учения / Г. Клаус : [пер. с нем.] [под ред. И. В. Равич-Щерба] – М. Педагогика, 1987. – 241 с.
122. Книга вчителя інформатики: Довідково-методичне видання / Упоряд. Н. С. Прокопенко, Т. Г. Проценко – Харків: ТОРСІНГ ПЛЮС, 2005. – 256 с.
123. Кнут Д. Э. Алгоритмическое мышление и математическое мышление / Д. Э. Кнут, пер. И. В. Лебедева. – М.: Изд. иностр. лит-ры, 1999. – 110 с.
124. Кнут Д. Э. Искусство программирования, том 1. Основные алгоритмы [3-е изд.] : [пер. с англ.] : [уч. пос.] Д. Э. Кнут – М.: Издательский дом «Вильямс», 2000. – 720 с.
125. Кнут Д.Э. Искусство программирования, том 2. Получисленные алгоритмы [3-е изд.] : [пер. с англ.] : [уч. пос.] Д. Э. Кнут – М.: Издательский дом «Вильямс», 2000. – 832 с.
126. Кнут Д. Э. Искусство программирования, том 3. Сортировка и поиск [3-е изд.] : [пер. с англ.] : [уч. пос.] Д. Э. Кнут – М.: Издательский дом «Вильямс», 2000. – 832 с.
127. Колесов А. А. Прыжок от Java к .NET [текст] /А. А. Колесов // PC Week/RE – 2001. – № 08. – С. 45.
128. Коменский Я. А. Великая дидактика / Я. А. Коменский. // Избр. пед. соч. Пер. с лат. В. И. Ивановского, Д. Н. Королькова, Н. С. Терновского. - Т. 1. - М.: Педагогика, 1982. – С. 242–476.

129. Концепція загальної середньої освіти (12-річна школа) // Інформаційний збірник Міністерства освіти і науки України. Січень 2002. – № 2. – К., Педагогічна преса, 2002. – 23с.
130. Концепція профільного навчання в старшій школі. // Трудове навчання. – 2010. – № 4(28). – С. 3–7.
131. Копаєв О. В. Алгоритм як модель алгоритмічного процесу / О. В. Копаєв. // Комп'ютерно-орієнтовані системи навчання: Зб. наук. праць / Редкол. – К.: НПУ ім. М. П. Драгоманова. – Випуск 6. – 2003 – С. 206–213.
132. Копаєв О. В. Модельна сутність алгоритму / О. В. Копаєв // Комп'ютерно-орієнтовані системи навчання: зб. наук. праць. – К.: НПУ імені М. П. Драгоманова, 2007. – Вип. 5 (12). – С. 189–193.
133. Копаєв О. В. Алгоритмічна діяльність та рівень розвитку теоретичного мислення // Вісник Черкаського Національного університету. Серія «Педагогічні науки». Випуск 55. – Черкаси, 2004. – С. 77–81.
134. Копотій В. В. Розвиток мислення учнів на уроках інформатики / В. В. Копотій // Комп'ютер у школі та сім'ї – 2006. – №5. – С. 8–11.
135. Краевский В. В Основы обучения. Дидактика и современная методика. / В. В. Краевский, А. В. Хуторской. – М. : Академія, 2007. – 352 с.
136. Краевский В. В. Методология педагогики [текст]: Пособие для педагогов-исследователей / В. В. Краевский. – Чебоксары: Изд-во Чуваш. ун-та, 2001. – 244 с.
137. Краевский В. В. Методология педагогического исследования / В. В. Краевский. – Самара : Издательство Сам ГПИ, 1994. – 165 с.
138. Крапивка С. В. Проблемы и особенности профильной дифференциации обучения информатике в общеобразовательной школе / Педагогический поиск. – 1998. – № 11. – С. 9–12.
139. Криницкий Н. А. Алгоритмы вокруг нас / Н. А. Криницкий – М.: Наука, 1977. – 224 с.

140. Кронгауз М. А. Жизнь и судьба гипотезы лингвистической относительности / М. М. Бурас, М. А. Кронгауз / Наука и жизнь. – 2011. – №8. – С. 66–72.
141. Кузьмина Н. В. Методы системного педагогического исследования: [учебное пособие] / Н. В. Кузьмина. – Л.: ЛГУ, 1980. – 172 с.
142. Кузьминський А. І. Педагогіка: Підручник / А. І. Кузьминський, В. Л. Омеляненко – К.: Знання-Прес, 2003. – 418 с. – (Навчально-методичний комплекс з педагогіки).
143. Кун Т. С. Структура научных революций [Електронний ресурс] / Т. С. Кун; [пер. с англ. И. З. Налетова] / Psylib. Самопознание и саморазвитие. Психологическая библиотека Киевского Фонда содействия развитию психической культуры. – режим доступу: URL : http://www.psylib.ukrweb.net/books/kunts01/index.htm. – Назва з екрана.
144. Кыверялг А. А. Методы исследования в профессиональной педагогике. / А. А. Кыверялг – Талин: Валгус, 1980. – 334 с.
145. Лаврентьєва Г. П. Методичні рекомендації з організації та проведення науково-педагогічного експерименту / Г. П. Лаврентьєва, М. П. Шишкіна – Київ: ІІТЗН, 2007. – 72 с.
146. Лаврентьєва Г. П. Психолого-ергономічні вимоги до застосування електронних засобів навчання [Електронний ресурс] / Г. П. Лаврентьєва // Інформаційні технології і засоби навчання. – 2009. – Випуск 4.– Режим доступу до журн. : http://www.ime.edu-ua.net/em.html. – Назва з екрана.
147. Лаврищева К. М. Аспектно–орієнтоване програмування. // К. М. Лаврищева / Програмна інженерія: Підручник [Електронний ресурс] – Режим доступу: http://www.programsfactory.univ.kiev.ua/ru/content/books/2/66.
148. Лавров С. С. Программирование. Математические основы, средства, теория. / С. С. Лавров – СПб.: БХВ-Петербург, 2001. – 320 с.

149. Левченко Т. М. Концепція неперервної освіти у світовому контексті / Т. М. Левченко // Неперервна професійна освіта : теорія і практика. – 2001. – Випуск ІІІ. – С. 7–12.
150. Лекции лауреатов премии Тьюринга за первые двадцать лет, 1966-1985 / ACM Turing Award Lectures: The First Twenty Years./ [пер. с англ.] ; [под ред. Эшенхёрста Р.] – М.: Мир, 1993. – 560 с., ил.
151. Леонтьев А. Н. Деятельность. Сознание. Личность. / А. Н. Леонтьев – М.: Политиздат, 1975. – 304 с.
152. Леонтьев А. Н. Проблемы развития психики. / А. Н. Леонтьев – 3-е изд. – М.: Изд-во МГУ, 1972. – 575 с.
153. Лернер И. Я. Проблемное обучение. / И. Я. Лернер.– М. : Знания, 1984. – 63 с.
154. Лернер И. Я. Дидактические основы методов обучения. – М.: Педагогика, 1981. – 186 с.
155. Лесневский А. С. Об основных понятиях школьного курса информатики. / А. С. Лесневский. / – Информатика и образование. – 1994. – №2. – С. 41-44.
156. Лесневский А. С. Объектно-ориентированное программирование для начинающих. / А. С. Лесневский. – М.: «БИНОМ. Лаборатория знаний», 2005. – 232 с.: ил..
157. Линней К. Философия ботаники = Philosophia botanica / [пер. с лат. Н. Н. Забинковой, С. В. Сапожникова, под ред. М. Э. Кирпичникова] – М.: Наука, 1989. – 456 с. – (Классики науки. Подсерия Памятники истории науки). – ISBN 5-02-003943-8.
158. Лисенко Т. І. Інформатика 11 : підруч. [для заг. навч. закл. ; академ.] / Й. Я. Ривкінд, В. В. Шакотько, Т. І. Лисенко, Л. А. Чернікова – К.: Генеза – 2011 – 304 с. : іл.
159. Лященко М. Я. Програмування на ЕКОМ. Посібник для факультативних занять у 9 класі / М. Я. Лященко, І. Ф. Следзінський – К.: «Радянська школа» – 1987, – 128 с. : іл.

160. Мадзигон В. Н. Продуктивная педагогика : политехн. основы соединения обучения с производит. трудом: [монография] / В. Н. Мадзигон ; АПН Украины, Институт педагогики. – К. : Педагогічна думка, 2007. – 358 с.
161. Майерс Д. Социальная психология / Д. Майерс: [пер. с англ.] – СПб.: Питер, 1997. – 688 с: ил.
162. Макконелл Дж. Анализ алгоритмов. Вводный курс / Дж. Макконелл: [пер. с анг.] – М.: Техносфера, 2002. – 304 с.
163. Макконнелл С. Совершенный код. / С. Макконнелл / Мастер-класс: [пер. с анг.] – М.: Издательско-торговый дом «Русская редакция»; СПб.: Питер, 2005. – 896 с.
164. Малафіїк І. В. Дидактика: Навчальний посібник /І. В. Малафіїк// – К.: Кондор, 2005. – 397с.
165. Мануйлов В. Г. Разработка программного обеспечения на Паскале: учеб. пособие / В. Г. Мануйлов; [под ред. А. И. Китова] – М.: ПРИОР, 1996. – 240 с., ил.
166. Марков А. А. Теория алгорифмов / А. А. Марков/ Тр. Мат. ин-та АН СССР им. В. А. Стеклова. – Т.42. – М.: Изд-во АН СССР, 1954. – С. 3–375.
167. Математический энциклопедический словарь / Гл. ред. Ю. В. Прохоров; Ред. кол.: С. И. Адян, Н. С. Бахвалов, В. И. Битюцков, А. П. Ершов, Л. Д. Кудрявцев, А. Л. Онищик, А. П. Юшкевич. – М.: Сов. энциклопедия, 1988. – 847 с., ил.
168. Махмутов М. И. Современный урок: Монографии [Текст] / М. И. Махмутов : [2-е изд., исп. и доп.] – М.: Педагогика, 1985. – 184 с.
169. Машбиц Е. И. Психолого-педагогические проблемы компьютеризации обучения. / Е. И. Машбиц – М.: Педагогика, 1988. – 192с.
170. Мейер Б. Методы программирования: В 2-х томах. Т.1. / Б. Мейер, К. Бодуэн [пер. с франц. Ю. А. Первина] : [под ред. и с предисловием А. П. Ершова.] – М.: Мир, 1982. – 356 с.

171. Мейер Б. Методы программирования: В 2-х томах. Т.2. / Б. Мейер, К. Бодуэн [пер. с франц. Ю. А. Первина] : [под ред. и с предисловием А. П. Ершова.] – М.: Мир, 1982. – 368 с.
172. Методичні рекомендації щодо вивчення інформатики у 2009/10 навчальному році // Комп'ютер у школі та сім'ї. – 2009. – №5. – С. 47–55.
173. Методологические и теоретические проблемы психологии : [под ред. Е. В. Шороховой. – М.: Наука, 1969. – 376 с.
174. Милитарев В. Ю. Информационная культура эпохи НТР / В. Ю. Милитарев, И. М. Яглом // Информатика и культура. – Новосибирск: Наука. Сиб. отд-ние, 1990. – С. 94–108.
175. Михалін Г. О. З історії становлення Київської наукової школи у галузі інформатико-математичної освіти / Г. О. Михалін // Науковий часопис НПУ ім. М. П. Драгоманова. Серія №2. Комп’ютерно-орієнтовані системи навчання. Зб. наукових праць / Ред.рада. – К.: НПУ ім. М. П. Драгоманова, 2010. № 9(16). – С. 267 – 338.
176. Мова C# в мережі Internet [Електронний ресурс] Веб-сайт «Колекція посилань, опис та аналіз Інтернет-сторінок, що стосуються мови програмування C#» . – режим доступу: URL : sites.google.com/site/sharpofnet. – Назва з екрана.
177. Мова програмування C#. [Електронний ресурс] Веб-сайт «Портал знань» . – режим доступу: URL : http://www.znannya.org/?view=csharp. – Назва з екрана.
178. Могилев А. В. и др. Информатика: Учеб. пособие для пед. вузов / А. В. Могилев, Н. И. Пак, Е. К. Хённер; [под ред. У. К. Хённера] – М.:, 1999. – 816 с.
179. Мойсеюк Н. Є. Педагогіка. Навчальний посібник / Н. Є. Мойсеюк / – 5-е видання, доп. і перероб. – К. 2007 – 656 с.

180. Монахов В. М. Дифференциация обучения в средней школе / В. М. Монахов, В. А. Орлов, В. В. Фирсов// Советская педагогика. – 1990. – №8. – С. 42–47.
181. Морзе Н. В. Інформатика : підруч. для 11 кл. загальноосвіт. навч. закл.: рівень стандарту / Н. В. Морзе, О. В. Барна, В. П. Вембер, О. Г. Кузьмінська, – К.: Школяр 2011. – 304 с.: іл.
182. Морзе Н. В. Методика навчання інформатики: Навч. посіб.: У 4 ч. / Н. В. Морзе // [за ред. акад. М. І. Жалдака] – К.: Навчальна Книга, 2003. – Ч. 1: Загальна методика навчання інформатики. – 254 с.
183. Морзе Н. В. Методика навчання інформатики: Навч. посіб.: У 4 ч. / [за ред. акад. М. І. Жалдака] – К.: Навчальна Книга, 2003. – Ч. ІІ: Методика навчання інформаційних технологій. – 287 с.
184. Морзе Н. В. Методика навчання інформатики: Навч. посіб.: У 4 ч. / [за ред. акад. М. І. Жалдака] – К.: Навчальна Книга, 2004. – Ч. ІІІ: Методика навчання основним послугам глобальної мережі Інтернет. – 200 с.
185. Морзе Н. В. Методика навчання інформатики: Навч. посіб.: У 4 ч. / [за ред. акад. М. І. Жалдака] – К.: Навчальна Книга, 2004. – Ч. ІV: Методика навчання основ алгоритмізації та програмування. – 368 с.
186. Морзе Н. В. Основи методичної підготовки вчителя інформатики. Монографія / Н. В. Морзе – К.: Курс, 2003. – 372 с.
187. Морзе Н. В. Система методичної підготовки майбутніх вчителів інформатики в педагогічних університетах: дис. ... доктора пед. наук: 13.00.02 / Наталія Вікторівна Морзе. – К.: НПУ імені М. П. Драгоманова., 2003. – 600 с.: іл.
188. Морозов А. А. С чего начинался ОГАС / А. А. Морозов, В. В. Глушкова, Э. П. Карпец // Труды конференции «Системы поддержки принятия решений. Теория и практика» К.: Институт проблем математических машин и систем НАН Украины и АТН Украины, 2010, – 230 с.

189. Мухортов В. В. Объектно-ориентированное программирование, анализ и дизайн: Методическое пособие / В. В. Мухортов, В. Ю. Рылов – Новосибирск : Новософт, 2002. – 108 с.
190. Наследов А. Д. Математические методы психологического исследования. Анализ и интерпретация данных. Учебное пособие / А. Д. Наследов – СПб.: Речь, 2004. – 392 с.
191. Немов Р. С. Психология : учебник: В 3-х кн. Кн. 3 : Психодиагностика / Р. С. Немов. – 4-е изд. – М. : Владос, 2005. – 631 с.
192. Немов Р. С. Психология : учебник: В 3-х кн. Кн. 1 : Общие основы психологии / Р. С. Немов. – 5-е изд. – М. : Владос, 2006. – 687 с.
193. Немов Р. С. Психология : учебник: В 3-х кн. Кн. 2 : Психология образования / Р. С. Немов. – 4-е изд. – М. : Владос, 2005. – 606 с.
194. Ніколаєнко О. Ю. Вивчення елементів візуального програмування в педагогічному вузі / О. Ю. Ніколаєнко // Комп'ютерно-орієнтовані системи навчання: зб. наук. пр. НПУ ім. М. П. Драгоманова. – К.: НПУ, 2000. – Вип. 2. – С. 195 – 202.
195. Новиков А. М. Научно-экспериментальная работа в образовательном учреждении / А. М. Новиков. – М. : Издательство АПО, 1998. – 132 с.
196. Новиков Д. А. Статистические методы в педагогических исследованиях (типовые случаи) / Д. А. Новиков. – М.: МЗ-Пресс, 2004. – 67 с.
197. Новое в лингвистике. Вып. 1: Метод глоттохронологии. Гипотеза Сепира Уорфа. Глоссематика. / [под ред. В. А. Звегинцева] – М.: Издательство иностранной литературы, 1960. – 464 с.
198. Новые педагогические и информационные технологии в системе образования / Е. С. Полат, М. Ю. Бухаркина, М. В. Моисеева и др. [под ред. Е. С. Полат] – М.: Педагогика. – 2007. – 272 с.
199. О системе программирования PascalABC.NET. [Електронний ресурс] // Веб-сайт «PascalABC.NET». – режим доступу: URL : http://pascalabc.mmcs.rsu.ru/. – Назва з екрана.

200. Одинцов И. О. Профессиональное программирование. Системный подход. / И. О. Одинцов – СПб.: БХВ·Петербург, 2002. – 512 с.
201. Окулов С. М. Основы программирования / С. М. Окулов – М.: ЮНИМЕДИАСТАЙЛ, 2002. – 424 с.
202. Онищенко С. М. Програмування мовою Паскаль: лабораторний практикум: навч. посібник / С. М. Онищенко. – К.: Логос, 2004. – 122 с.
203. Онищенко С. М. Вивчення основ програмування на базі систем програмування Turbo Pascal та Delphi в середній загальноосвітній школі / С. М. Онищенко // Комп'ютерно-орієнтовані системи навчання: зб. наук. пр. НПУ ім. М. П. Драгоманова. – К.: НПУ, 2000. – Вип.2. – С. 163–167.
204. Онищенко С. М. Використання Delphi (Kylix) при вивченні основ програмування / С. М. Онищенко, В. М. Франчук // Вісник: Зб. наук. статей НПУ ім. М. П. Драгоманова. – К.: НПУ ім. М. П. Драгоманова, 2004. Вип.6
205. Основи нових інформаційних технологій навчання: Посібник для вчителів / Авт. кол.; [за ред. Ю. І. Машбиця] / Інститут психології ім. Г. С. Костюка АПН України. – К.: ІЗМН, 1997. – 264 с.
206. Павлов И. П. Избранные произведения / И. П. Павлов / под общ. ред. Х. С. Коштоянца. – М., 1949. – 569 с.
207. Парадигма програмування. [Електронний ресурс] Матеріал з Вікіпедії – вільної енциклопедії. – режим доступу: URL : http://uk.wikipedia.org/wiki/Парадигма_програмування. – Назва з екрана.
208. Пасько В. П. Програма курсу за вибором «Основи інформаційної безпеки»/ В. П. Пасько, Н. С. Прокопенко // Інформатика та інформаційні технології в навчальних закладах. – 2006. – №4/5. – С. 56–60.
209. Педагогика : Учебное пособие для студентов пед. вузов / [Ю. К. Бабанский, В. А. Мластенин, Н. А. Сорокин и др.] ; под ред. Ю. К. Бабанского. – [2-е изд., доп. и перераб.]. – М. : Просвещение, 1988. – 479 с.

210. Пекшева А. Г. Методика подготовки учителя информатики к обеспечению предпрофильного обучения. дис. … кандидата пед. наук : 13.00.02 / Анна Георгиевна Пекшева. – Ростов-на-Дону, 2008. – 180с.
211. Перегудов Ф. И. Введение в системный анализ. / Ф. И. Перегудов, Ф. П. Тарасенко – М.: Высшая школа, 1989. – 320 с.
212. Петренко В. Ф. Основы психосемантики: Учеб. Пособие. / В. Ф. Петренко – М.: Изд-во Моск. ун-та, 1997. – 400 с.
213. Петренко В. Ф. Психосемантика сознания. / В. Ф. Петренко – М.: Изд-во Моск. ун-та, 1988. – 208 с.
214. Петриковский А. Субъектное программирование / А. Петриковский // «Компьютерра». — 05.04.2006. — № 13 [Електронний ресурс] // Веб-сайт «Компьютерры-Онлайн» – 22.05.2011. – Режим доступу: http://offline.computerra.ru/2006/633/262140/. – Назва з екрана.
215. Петров А. Н. Совершенствование методики обучения объектно-ориентированному программированию на основе объектно-ориентированного проектирования : на примере дисциплины «Программирование» для будущих учителей информатики : автореф. дис. на соискание уч. степени канд. пед. наук : спец. 13.00.02 – теория и методика обучения и воспитания (информатика, уровень высшего профессионального образования)/ А. Н. Петров – Москва – 2009. – 20 с.
216. Петров А. Н. Совершенствование методики обучения объектно-ориентированному программированию на основе объектно-ориентированного проектирования : на примере дисциплины «Программирование» для будущих учителей информатики : дис. … кандидата пед. наук : 13.00.02 / Петров Алексей Николаевич. – М., 2009. – 151 с.: ил.
217. Пилипчук О. П. Моделювання звичайних дробів засобами об'єктно-орієнтованого програмування (C#). / О. П. Пилипчук // Творча лабораторія

вчителя інформатики. [Електронний ресурс] – 14.04.2011. – Режим доступу: http://teachlab.ucoz.ua/publ/10-1-0-15. – Назва з екрана.

218. Пилипчук О. П. Програмування на C#... без середовища. Як підключити компілятор до текстового редактора. / О. П. Пилипчук // Творча лабораторія вчителя інформатики. [Електронний ресурс] – 24.10.2010. – Режим доступу: http://teachlab.ucoz.ua/publ/6-1-0-35. – Назва з екрана.
219. Пінчук О. П. Оцінювання результатів педагогічного впливу в контексті компетентнісно зорієнтованого навчання фізики [Електронний ресурс] / О. П. Пінчук // Інформаційні технології і засоби навчання. – 2009. – № 3(11). – Режим доступу до журн. : http://www.nbuv.gov.ua/e-journals/ITZN/em11/emg.html.
220. Пінчук О. П. Оцінювання рівня сформованості предметних компетентностей учнів основної школи методом семантичного диференціала в процесі навчання фізики / Ю. О. Жук, О. П. Пінчук // Науковий часопис НПУ імені М. П. Драгоманова. Серія № 5. Педагогічні науки: реалії та перспективи. – Випуск 12: Зб. наук. пр. – К.: Вид-во НПУ імені М. П. Драгоманова, 2008. – С. 120–127.
221. Про Національну доктрину розвитку освіти: Указ Президента України від17 квітня 2002 р. № 347, // Офіційний вісник України. – 2002. – № 16. – ст. 860. Національна доктрина розвитку освіти [текст] // Освіта. – 2002. – № 26. – С. 2–12.
222. Бермус А. Г. Проблемы и перспективы реализации компетентностного подхода в образовании : (Проблемы компетентностного подхода) [Електронний ресурс] / А. Г. Бермус //. – режим доступу: URL : http://www.eidos.ru/journal/2005/0910-12.htm. – Назва з екрана.
223. Програма для загальноосвітніх навчальних закладів 10-11 класи. Інформатика. Академічний рівень. Профільний рівень. Рівень стандарту. – К.: Міністерство освіти і науки України, 2010. – 110 с.

224. Производительность .Net миф или фантастика? [Електронний ресурс] // Веб-сайт «Got Dot Net» – сообщества .NET разработчиков . – режим доступу: URL : http://www.gotdotnet.ru/LearnDotNet/CSharp/513899.aspx. – Назва з екрана.
225. Прокопенко І. Ф. Педагогічна технологія. / І. Ф. Прокопенко, В. І. Євдокимов – Харків, 1995. – 374 с.
226. Прокопенко Н. С. Інструктивно-методичний лист про вивчення Інформатики у 2010/2011 навчальному році. / Н. С. Прокопенко, // Інформатика та інформаційні технології в навчальних закладах. – 2010. – №5-6. – С. 34–50.
227. Профільне навчання в старшій школі: шляхи розвитку : наук.-допом. бібліогр. покажч. Вип. 2 / НАПН України, ДНПБ України ім. В. О. Сухомлинського [та ін. ; упоряд.: Л. О. Пономаренко, Н. А. Стельмах, Л. М. Айвазова; наук. консультант В. І. Кизенко; авт. вступ. ст.: Н. М. Бібік, В. І. .Кизенко; наук. ред. П. І. Рогова]. – К. : [б. в.], 2010. – 362 с.
228. Профільне навчання в старшій школі: шляхи розвитку: Наук.-доп. бібліогр. покажч. Вип.1 / АПН України. ДНПБ України ім. В. О. Сухомлинського; Уклад.: Л. О. Пономаренко, Л. І. Ніколюк, Л. І. Самчук, І. М. Каневська; Наук. консультант і автор вступ. ст. В. І. Кизенко, канд. пед. наук; Наук. ред. і відп. за вип. Т. Ф. Букшина; Бібліогр. ред. Є. К. Бабич; Рецензент Н. М. Бібік, д-р пед. наук. – К., 2004. – 163 с.
229. Профільне навчання: досвід упровадження, інноваційні технології. Упор. Л. Ф. Пашко, О. П. Коваленко, Л. І. Симоненко – Полтава: ПОІППО, 2008. – 196 с.
230. Проценко В. С. та ін. Техніка програмування мовою Сі: Навч. Посібник / В. С. Проценко, П. П. Чаленко, А. Б. Ставровский. – К.: Либідь, 1993. – 224 с.

231. Проценко Т. Г. Календарно-тематичне планування навчання інформатики за програмою 10-11 класів загальноосвітніх навчальних закладів фізико-математичного, природничого і технологічного профілів / Т. Г. Проценко // Інформатика та інформаційні технології в навчальних закладах. – 2006. – №4/5. – С. 107–114.
232. Пустовіт Л. О. Словник іншомовних слів. / Л. О. Пустовіт, О. І. Скопенко, Г. М. Сюта, Т. В. Цимбалюк – К., 2000. – 1018 с.
233. Пышкало А. М. Методическая система обучения геометрии в начальной школе: Авторский доклад по монографии «Методика обучения геометрии в начальных классах», предст. на соиск. уч. степ. докт. пед. наук. / А. М. Пышкало – М., 1975. – 60 с.
234. Пышкало А. М. Методическая система обучения геометрии в начальной школе: автореф. дис. … док. пед. наук. / А .М. Пышкало – М.:1975. – 60 с.
235. Рамський Ю. С. Вивчення Web-програмування в школі: Навчальний посібник. / Ю. С. Рамський, І. С. Іваськів, О. Ю. Ніколаєнко – Тернопіль: Навчальна книга – Богдан , 2004. – 200 с.
236. Рамський Ю. С. Методика навчання основ об’єктно-орієнтованого програмування/ Ю. С. Рамський, І. М. Лукаш//Комп’ютер у школі та сім’ї. – 2002. – №1. – С. 3–7; №2. – С. 3–8 ; №3. – С. 7–13; №4. – С. 17–22; №5. – С. 10–17; №6. – С.16–21.
237. Рамський Ю. С. Методика навчання основ об’єктно-орієнтованого програмування. Об’єктно-орієнтований аналіз / Ю. С. Рамський, І. М. Лукаш// Комп’ютер у школі та сім’ї. – 2003. – №1. – С.3–9.
238. Ребрина В. А. Факультативний курс з програмування мовою С++ 7-9 класи / С. Д. Вапнічний, В. В. Зубик, В. А. Ребрина – Хмельницький: видавничий відділ Хмельницького ОІППО, 2010. – 128 с.
239. Решетова З. А. Психологические основы профессионального обучения. / З. А. Решетова – М., 1985. – 202 с.

240. Ривкінд Й. Я. Інформатика : 9 кл. : підруч. [для заг. навч. закл.] /Й. Я. Ривкінд, Т. І. Лисенко, Л. А. Чернікова, В. В. Шакотько; за заг. ред. М. З. Згуровського. – К.: Генеза, 2009. – 296 с. : іл.
241. Ривкінд Й. Я. Інформатика : 10 кл. : підруч. [для заг. навч. закл. ; станд] / Й. Я. Ривкінд, Т. І. Лисенко, Л. А. Чернікова, В. В. Шакотько; за заг. ред. М. З. Згуровського. – К.: Генеза, 2010. – 296 с. : іл.
242. Ривкінд Й. Я. Інформатика : 11 кл. : підруч. [для заг. навч. закл. ; профільний] / Й. Я. Ривкінд, Т. І. Лисенко, Л. А. Чернікова, В. В. Шакотько; за заг. ред. М. З. Згуровського. – К.: Генеза, 2011. – 304 с. : іл.
243. Росс С. И. Элементы системного анализа. Понятия и основные свойства систем : Конспект по исследованию систем управления. / С. И. Росс / [Электронный ресурс] // Веб-сайт «StudeFiles. Всё для учёбы» . – режим доступу: URL : http://www.studfiles.ru/dir/cat38/subj347/file1298/view2121.html. – Назва з екрана.
244. Рубинштейн С. Л. Основы общей психологии / С. Л. Рубинштейн – 2-е изд. (1946г.) – СПб.: Питер, 2002 – 720 с.
245. Руденко В. Д., Базовий курс інформатики / В. Д. Руденко, О. М. Макарчук, М. О. Патланжоглу; [за заг. ред. В. Ю. Бикова] : [навч. посіб.] – К.: Вид. група BHV. – Кн. 2: Інформаційні технології, 2006. – 368 с. : іл.
246. Руденко В. Д. Практичний курс інформатики / В. Д. Руденко, О. М. Макарчук, М. О. Патланжоглу; за ред. В. М. Мадзігона – К.: Фенікс, 2001. – 370 с.
247. Савенко Ю. С. Проективные методы в изучении бессознательного / Ю. С. Савенко // Бессознательное. Природа; Функции. Методы исследования: — Тбилиси, 1978. Т.3. - С. 632–637.
248. Сазонов А. Д. Профессиональная ориентация молодежи / А. Д. Сазонов, Н. И. Калугин, А. П. Менщиков и др.– М.: Высш. шк., 1992. – 334 с.

249. Сазонов А. Д. Профессиональная ориентация молодёжи. / А. Д. Сазонов, Н. И. Калугин, В. Д. Симоненко – М., Просвещение, 1983. – 284 с.
250. Самордін А. Професійна диференціація – проблема української школи / А. Самордін // Директор школи, ліцею, гімназії. – 2001. – №1 – С. 7–13.
251. Свердлов С. З. Язык программирования Си #: критическая оценка. / С. З. Свердлов // PC Week/RE – 2001. – № 20 – С. 7–10 ; № 22. – С. 7–9.
252. Себеста Р. Основные концепции языков программирования / Р. Себеста / [5-е издание] : [пер. с англ.] – М.: Издательский дом «Вильямс», 2001. – 672 с.
253. Сейдаметова З. С. Сценарний підхід в навчанні інформатики: об'єктність, наочність, креативність./ З. С. Сейдаметова, Ф. В. Шкарбан // Науковий часопис НПУ ім. Драгоманова, серія №2. Комп'ютерно-орієнтовані системи навчання: зб. наук. праць / Редрада. – К.: НПУ ім. Драгоманова, – №11 (16) – 2011. – С. 3–11.
254. Сейдаметова З. С. Методическая система уровневой подготовки будущих инженеров-программистов по специальности «Информатика» : дис. ... доктора пед. наук : 13.00.02 / Зарема Сейдалиевна Сейдаметова; НПУ им. М. П. Драгоманова. – К., 2007. – 559 с.
255. Сейдаметова З. С. Подготовка инженеров-программистов по специальности «Информатика» [Текст] / З. С. Сейдаметова ; Национальный педагогический ун-т им. М. П. Драгоманова. – К. ; Симф. : Крымучпедгиз, 2007. – 480 с.
256. Сейдаметова З. С., Моделі навчання основ програмування на молодших курсах комп'ютерних спеціальностей університетів // З. С. Сейдаметова, Л. М. Меджитова // Науковий часопис НПУ ім. М. П. Драгоманова. Серія №2. Комп'ютерно-орієнтовані системи навчання. Зб. наукових праць / Ред. рада. – К.: НПУ ім. М. П. Драгоманова, 2009. № 7(14). – С. 103–107.
257. Сейтешев А. П. Пути профессионального становления учащейся молодежи. / А. П. Сейтешев – М., Высшая школа, 1988. – 336 с.

258. Секунов Н. Ю. Самоучитель C#. / Н. Ю. Секунов – СПб.: БХВ-Петербург, 2001. – 576 с. : ил.
259. Семеріков С. О. Активізація пізнавальної діяльності студентів при вивченні чисельних методів у об’єктно-орієнтованій технології програмування. [Текст] : Дис. ... канд. пед. наук : 13.00.02 / Сергій Олексійович Семеріков ; Криворізький держ. педагогічний ун-т. – Кривий Ріг, 2000. – 255 с.
260. Семеріков С. О. , Роль, місце та зміст комп’ютерного моделювання в системі шкільної освіти // С. О. Семеріков, І. О. Теплицький // Науковий часопис НПУ ім. М. П. Драгоманова. Серія №2. Комп’ютерно-орієнтовані системи навчання. Зб.наукових праць / Ред.рада. – К.: НПУ ім. М. П. Драгоманова, 2010. № 9(16). – С. 3–11.
261. Серкин В. П. Методы психологии субъективной семантики и психосемантики: Учебное пособие для вузов / В. П. Серкин. – М.: «Пчела», 2008. – 382 с.
262. Сиденко А. Вы начали эксперимент / А. Сиденко, В. Чернушевич // Народное образование. – 1997. – № 7–8. – С. 65–68.
263. Сидоренко Е. В. Методы математической обработки в психологи / Е. В. Сидоренко. – СПб.: ООО «Речь», 2003. – 350 с. – ISBN 5-9268-0010-2
264. Система. [Електронний ресурс] Матеріал з Вікіпедії – вільної енциклопедії – режим доступу: URL : http://uk.wikipedia.org/wiki/Система. – Назва з екрана.
265. Скляр І. В. Розвиток алгоритмічного мислення – основна задача курсу інформатики [Текст] / І. В. Скляр // Комп'ютер у школі та сім'ї : Науково-методичний журнал. – 2010. – N 2. – С. 11–13.
266. Словари и энциклопедии на «Академике» .NET Framework – толкования. [Электронный ресурс] // Веб-сайт «Словари и энциклопедии на «Академике» . – режим доступу: URL : http://dic.academic.ru/dic.nsf/ruwiki/56724. – Назва з екрана.

267. Смолкин А. М. Методы активного обучения: Научн.-метод. пособие. / А. М. Смолкин – М.: Высшая школа, 1991. – 176 с.
268. Смульсон М. Л. Психологія розвитку інтелекту: Монографія. / М. Л. Смульсон – К., 2001. – 276 с.
269. Соколова Е. Т. Проективные методы исследования личности / Е. Т. Соколова /– М.: Изд-во МГУ, 1987. – 102 с.
270. Співаковський О. В. Основи програмування. Навчальний посібник. / А. М. Гуржій, М. С. Львов , О. В. Співаковський – К.: Наукова думка, 2004. – 355 с.
271. Спірін О. М. Початки штучного інтелекту : навч. посіб. для студ. фіз.-мат. спец. вищ. пед. навч. закл. / О. М. Спірін – Житомир : Вид-во ЖДУ, 2004. – 172 с.
272. Спірін О. М. Диференційований підхід у вивченні основ штучного інтелекту в курсі інформатики фізико-математичного факультету вищого педагогічного закладу: дис. … кандидата пед. наук : 13.00.02 / Олег Михайлович Спірін; НПУ ім. Драгоманова. – К., 2001. – 223 с.
273. Спірін О. М. Практична інформатика: 3-те видання, перероблена і доповнене: Методичний посібник для природничих спеціальностей. / О. М. Спірін – Житомир: Поліграфічний центр ЖДПУ, 2001. – 176 с., іл.
274. Структурное программирование. Из истории программирования. [Электронный ресурс] // Веб-сайт «Информационные технологии для начинающих и не только...» . – режим доступу: URL : http://manuilov.narod.ru/structura/2_1.htm – Назва з екрана.
275. Структурне програмування / UA5.ORG. Методичні матеріали з інформатики. [Електронний ресурс]. – Режим доступу : URL : http://www.ua5.org/osnprog/200-strukturne-programuvannja.html. – Назва з екрана.

276. Сукин И. А. Haskell: серебряная пуля современного программирования? / И. А. Сукин. // Информатика : метод. газ. для учителей информатики / гл. ред. С. Л. Островский. – М. : Первое сент. 2011. – N 14 – С. 36–46.
277. Сччтрауструп Б. Язык программирования С++ / Б. Сччтрауструп // [3-е изд.] : [пер. с англ.] – СПб.; М.: «Невский диалект» – «Издательство БИНОМ», 1999. – 991 с.
278. Талызина Н. Ф. Управление процессом усвоения знаний / Н. Ф. Талызина – М.: Изд-во МГУ, 1975. – 343 с.
279. Талызина Н. Ф. Пути использования теории планомерного формирования умственных действий в практике образования / Н. Ф. Талызина // Вестник московского университета. Сер. 14. Психология. – 1992. – №4. – С. 18–26.
280. Тассел Д. Ван. Стиль, разработка, эффективность, отладка и испытание программ / Д. Ван Тассел // – М.: Мир, 1985. – 332 с.
281. Теплицький О. І. Динамічне графічне об’єктно-орієнтоване моделювання в мультимедіа-середовищі мобільного навчання Squeak / О. І. Теплицький, І. О. Теплицький, С. О. Семеріков // Комп'ютерно-орієнтовані системи навчання: збірник наукових праць. М-во освіти і науки України, НПУ ім. М. П. Драгоманова; Відп. редактор М. І. Жалдак. – Київ, 2009. – Вип. 14 – С. 72–76.
282. Теслер Г. С. Новая кибернетика / Г. С. Теслер – К.: Логос, 2004. – 404 с.
283. Триус Ю. В. Комп'ютерно-орієнтовані методичні системи навчання математичних дисциплін у вищих навчальних закладах. : дис. ... доктора пед. наук : 13.00.02 / Юрій Васильович Триус; НПУ им. М. П. Драгоманова. – К., 2005. – 649 с.
284. Триус Ю. В. Особливості створення методичної системи навчання основам програмування для підготовки майбутніх інженерів-програмістів // Ю. В. Триус, О. О. Богатирьов, Л. В. Гришко / Вісник Черкаського університету, серія «Педагогічні науки». Випуск 35. – Черкаси, 2002. – С. 135–141.

285. Троелсен Эндрю. Язык программирования C# 2005 (Си Шарп) и платформа .NET 2.0 = Troelsen, Andrew. Pro C# 2005 and the .NET 2.0 Platform. / Andrew Troelsen = Эндрю Троелсен. – 3-е изд. – М.: «Вильямс», 2007. – 1168 с. – ISBN 5-8459-1124-9, 978-5-8459-1124-7.
286. Український педагогічний словник / [авт.-уклад. С. У. Гончаренко]. – Київ : Либідь, 1997. – 376 с.
287. Унт И. Э. Индивидуализация и дифференциация обучения / И. Э. Унт – М.: Педагогика, 1990. – 192 с.
288. Уфимцева М. А. Формы организации обучения в современной общеобразовательной школе. / М. А. Уфимцева – М.: Просвещение, 1986. – 80 с.
289. Фаннінгер Л. П. Особливості профільного навчання в основній школі Австрії / Людмила Павлівна Фаннінгер : дис. канд. пед. наук. 13.00.01; ТНПУ ім. В. Гнатюка. – Тернопіль, 2008. – 236 с.
290. Философская энциклопедия. – Т. 4. – М., 1970. – С. 383.
291. Фіцула М. М. Педагогіка: Навчальний посібник для студентів вищих педагогічних закладів освіти. / М. М. Фіцула – К.: Видавничий центр «Академія», 2000. – 544 с.
292. Форми навчання в школі: Книга для вчителя / [за ред. Ю. І. Мальованого] – К.: Освіта, 1992. – 160 с.
293. Ханина И. Б. Семантические факторы обучающего общения : автореф. дис. на соискание науч. степени канд. психол. наук : спец. 19.00.01 «Общая психология, история психологии, психология личности» / И. Б. Ханина – М., – 1986. – 19 с.
294. Харламов И. Ф. Педагогика / И. Ф. Харламов – М.: Гардарики, 1999. – 520 с.
295. Хейлсберг А. Язык программирования C#. Классика Computers Science / А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд; [пер. с англ.]. – 4-е изд. – СПб.: Питер, 2012. – 773 с.: ил.

296. Черняк Л. Алан Кей: из гитаристов в пророки. / Леонид Черняк // PC Week, корпоративные информационные технологии и решения. – М, 1998, – №39 (163) – С. 12–19.
297. Шавир П. А. Психология профессионального самоопределения в ранней юности. / П. А. Шавир – М., 1981. – 95 с.
298. Шаповаленко И. В. Возрастная психология (Психология развития и возрастная психология). / И. В. Шаповаленко – М.: Гардарики, 2005. – 349 с.
299. Шахмаев Н. М. Дифференциация обучения в средней общеобразовательной школе // Дидактика средней школы / Н. М. Шахмаев [под ред. М. Н. Скаткина]. – [2-ое изд., перера. И доп.] – М.: Просвещение, 1982. – Гл. 8. – С. 269–296.
300. Шевчук П. Г. Визначення результатів педагогічного впливу в процесі навчання програмування / П. Г. Шевчук // Звітна наукова конференція Інституту інформаційних технологій і засобів навчання НАПН: Матеріали наукової конференції. – Київ: ІІТЗН НАПН України, 2012. – С. 117–118.
301. Шевчук П. Г. Використання Інтернет для дослідження результатів навчання програмування учнів класів технологічного профілю загальноосвітніх навчальних закладів / П. Г. Шевчук // Матеріали міжнародного форуму фахівців у галузі освітніх вимірювань (Київ, 1 червня 2012 р.) – К. : НПУ, 2012. – С. 61–62.
302. Шевчук П. Г. Використання платформи Microsoft .NET для навчання програмування в середніх загальноосвітніх навчальних закладах / П. Г. Шевчук // Нові інформаційні технології в освіті для всіх: інноваційні методи та моделі / Збірник праць IV Міжнародної конференції «Нові інформаційні технології в освіті для всіх: інноваційні методи та моделі», що відбулася 24-26 листопаду 2009 р. на базі Міжнародного науково-навчального центру інформаційних технологій та систем НАН і МОН України. – С. 378–387.

303. Шевчук П. Г. Від Pascal до C# [Текст] / П. Г. Шевчук // Комп'ютер у школі та сім'ї : Науково-методичний журнал. – 2011 – № 4 – С. 47–52; № 5 – С. 40–45.
304. Шевчук П. Г. Візуальне програмування на уроках інформатики загальноосвітніх навчальних закладів з використанням мови програмування C#/ П. Г. Шевчук // Комп’ютерне моделювання та інформаційні технології в науці, економіці і освіті / Збірник праць VIII Всеукраїнської науково-практичної конференції; Черкаси-Одеса, 25-27 травня 2011 р / Редкол.: В. М. Соловйов (відп. За випуск) та ін.. Черкаси: Брама., 2011 – С. 204–206.
305. Шевчук П. Г. Значення стилю програмування в процесі навчання учнів та студентів / О. М. Кривонос, П. Г. Шевчук // Комп'ютерно-інтегровані технології: освіта, наука, виробництво : Науковий журнал; Відп. ред. В. Д. Рудь. – Луцьк, 2011. – № 5 – С. 148–150.
306. Шевчук П. Г. Критерії добору мови та середовища програмування як засобів навчання / П. Г. Шевчук // Звітна наукова конференція Інституту інформаційних технологій і засобів навчання НАПН: Матеріали наукової конференції. – Київ: ІІТЗН НАПН України, 2010. – С. 89–92.
307. Шевчук П. Г. Навчання об’єктно-орієнтованому програмуванню в загальноосвітніх навчальних закладах засобами мови C# / П. Г. Шевчук // Теорія та методика навчання математики, фізики, інформатики: збірник наукових праць. Випуск IX. – Кривий ріг: Видавничий відділ НМетАУ, 2011 – С. 595–601.
308. Шевчук П. Г. Навчання програмування в класах технологічного профілю загальноосвітніх навчальних закладів на основі використання мови C# : Методичні рекомендації для вчителів інформатики / П. Г. Шевчук – К., 2012. – 32 с. [Електронний ресурс] – режим доступу: URL : http://lib.iitta.gov.ua/896/4/Metodrekom_Shevchuk_.pdf.

309. Шевчук П. Г. Навчання програмування мовою C# у ЗНЗ: аспекти дистанційного навчання / П. Г. Шевчук // Звітна наукова конференція Інституту інформаційних технологій і засобів навчання НАПН: Матеріали наукової конференції. – Київ: ІІТЗН НАПН України, 2011. – С. 83–84.
310. Шевчук П. Г. Огляд ресурсів Internet корисних для навчання програмування мовою C# / П. Г. Шевчук, О. М. Шимон // Інформаційні технології в навчальному процесі: праці науково-методичного семінару 16-23 травня 2011 р., ННПУ імені Ушинського, Одеса / наук. ред. М. І. Жалдак . – Одеса: Вид. «ВМВ»., С. 159–162.
311. Шевчук П. Г. Основні підходи добору мови та середовища програмування як засобів навчання // П. Г. Шевчук / Інформаційні технології і засоби навчання: електронне наукове фахове видання [Електронний ресурс] / Ін-т інформ. технологій і засобів навчання АПН України, Ун-т менеджменту освіти АПН України; гол. ред.: В. Ю. Биков. – 2010. – № 3(17). – Режим доступу: http://journal.iitta.gov.ua/index.php/itlt/article/view/251.
312. Шевчук П. Г. Особливості навчання інформатики в класах технологічного профілю загальноосвітніх навчальних закладів / П. Г. Шевчук // Сучасні тенденції розвитку інформаційних технологій в науці, освіті та економіці : матеріали VI Всеукраїнської науково-практичної конференції (Луганськ, 31 травня – 1 червня 2012 р.) – Луганськ : Phoenix, 2012. – С. 255–258.
313. Шевчук П. Г. Оцінювання ефективності навчання програмування на основі різних мов та парадигм написання комп’ютерних програм // П. Г. Шевчук / Інженерія програмного забезпечення: Науковий журнал: – 2011 – № 4(6) – С. 78–83.
314. Шевчук П. Г. Платформа Microsoft .NET в змісті навчального матеріалу курсу інформатики середньої загальноосвітньої школи // П. Г. Шевчук / Інформаційні технології і засоби навчання: електронне наукове фахове видання [Електронний ресурс] / Ін-т інформ. технологій і засобів навчання АПН України, Ун-т менеджменту освіти АПН України; гол. ред.:

В. Ю. Биков. – 2009. – № 3(11). – Режим доступу: http://journal.iitta.gov.ua/index.php/itlt/article/view/60.

315. Шевчук П. Г. Програмно-технологічні можливості використання мови C# для навчання програмування / П. Г. Шевчук // Всеукраїнська науково-практична конференція «Освіта в інформаційному суспільстві: до 25-річчя шкільної інформатики» : Матеріали наукової конференції. – Київ: університет ім. Бориса Грінченка, 2010. – С. 45–47.
316. Шевчук П. Г. Програмно-технологічні умови використання мови C# для навчання програмування в загальноосвітніх навчальних закладах / П. Г. Шевчук // Комп'ютерно-орієнтовані системи навчання: збірник наукових праць. М-во освіти і науки України, НПУ ім. М. П. Драгоманова; Відп. ред. М. І. Жалдак. – Київ, 2011. – Вип. 17 – С. 80–83.
317. Шестаков А. П. Профильное обучение информатике в старших классах средней школы на примере курса «Компьютерное математическое моделирование» : дисс. ... кандидата пед. наук : 13.00.02 / Александр Петрович Шестаков. – Омск, 1999. – 153 с.
318. Шишкіна М. П. Засоби навчання: проблеми термінології / М. П. Шишкіна // Проблеми освіти. – 1998. – №14. – С. 205 – 208.
319. Шкарабан Ф. В. Формування об'єктно-орієнтованного мислення у студентів комп'ютерних спеціальностей. / Ф. С. Ильясова, Ф. В. Шкарбан // Сучасні інформаційні технології та інноваційні методики навчання у підготовці фахівців: методологія, теорія, досвід, проблеми. – Вінниця, 2010. – Вип. 24. [Електронний ресурс] . – режим доступу: URL : http://www.nbuv.gov.ua/portal/soc_gum/Sitimn/2010_24/Formuvanna_obektno_orientovanogo_muslenna_u_studentiv.pdf.
320. Школа программирования. Обучение разработке ПО и интернет-проектов [Електронний ресурс] // Веб-сайт «Школа программирования» . – режим доступу: URL : http://prog-school.ru/. – Назва з екрана.

321. Шнейдерман Б. Психология программирования: Человеческие факторы в вычислительных и информационных системах. / Б. Шнейдерман [пер. с англ.] – М.: Радио и связь, 1984. – 304 с.
322. Шост Д. М. 10-11 класи: Тематичне оцінювання. 36. завдань. Д. М. Шост – Тернопіль: Навчальна книга – Богдан, 2006. – 36 с.
323. Эккель Б. Философия Java. Библиотека программиста. 4-е издание / Эккель Брюс [пер. с англ.] – СПб.: Питер, 2009. – 640 с.: ил.
324. Ягупов В. В. Педагогіка: [Навч. посібник.] / В. В. Ягупов – К.: Либідь, 2002. – 560 с.
325. .NET Framework [Електронний ресурс] // Материал из Википедии – свободной энциклопедии. – режим доступу: URL : http://ru.wikipedia.org/wiki/.NET. – Назва з екрана.
326. .NET Framework Developer Center. Главная страница. [Електронний ресурс] // Веб-сайт «Центр разработчиков .NET» . – режим доступу: URL : http://msdn.microsoft.com/ru-ru/netframework/default.aspx. – Назва з екрана.
327. Antechinus® C# Editor: visually design .Net applications with the best programming language today. [Electronic Resource] // Web site «Video Marketing. Software and services.» – Mode of access : URL : http://msdn.microsoft.com/ru-ru/vcsharp/default.aspx. Title from the screen.
328. Bloom, B. S. Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook 1, Cognitive domain / Bloom B. S., Engelhart M. D., Furst E. J. and oth. ; ed. by B. S. Bloom. – New York : David McKay, 1956. – 201 p.
329. C Sharp. [Електронний ресурс] Матеріал з Вікіпедії – вільної енциклопедії. – режим доступу: URL : http://uk.wikipedia.org/wiki/C_Sharp. – Назва з екрана.
330. C# – в школу. [Електронний ресурс] // Веб-сайт «Microsoft для прессы». – режим доступу: URL : http://www.ms4press.ru/post/2009/11/03/C-d0b2-d188d0bad0bed0bbd183.aspx. – Назва з екрана.

331. C# station. [Electronic Resource] // This is a community site for people interested in applying .NET using the C# programming language. – Mode of access : URL : http://www.csharp-station.com/ Title from the screen.
332. C# у школі [Електронний ресурс] // Веб-сайт «Для тих хто хоче дізнатися яке ж місце за шкільною партою дістанеться новій мові програмування C#.». – режим доступу: URL : https://sites.google.com/site/c4plus. – Назва з екрана.
333. C#. Language Specification. Version 3.0 / Copyright. Microsoft Corporation 1999-2007. All Rights Reserved. [Електронний ресурс] // Центр загрузки Microsoft. – режим доступу: URL : http://download.microsoft.com/download/3/8/8/388e7205-bc10-4226-b2a8-75351c669b09/CSharp%20Language%20Specification.doc. – Назва з екрана.
334. C-подобный синтаксис [Електронний ресурс] // Материал из Википедии – свободной энциклопедии. – режим доступу: URL : http://ru.wikipedia.org/wiki/C-подобный_синтаксис. – Назва з екрана.
335. Dotnetpowered Language List. [Electronic Resource] // Web site «Dotnetpowered». – Mode of access : URL : http://www.dotnetpowered.com/languages.aspx. Title from the screen.
336. E-olimp. On-line контролююча система. [Електронний ресурс] // Веб-сайт «E-olimp» . – режим доступу: URL : http://www.e-olimp.com.ua/. – Назва з екрана.
337. Greeno J. G. The structure of memory and the process of problem solving. / J. G. Greeno – University of Michigan: Human Perfomance Center Technical Report 37, 1972. – p. 128.
338. Java. Материал из Википедии – свободной энциклопедии. [Електронний ресурс] // Веб-сайт «Википедии – свободная энциклопедия». – режим доступу: URL : http://ru.wikipedia.org/wiki/Java. – Назва з екрана.
339. Kuhn T. S. The Structure of Scientific Revolutions. / T. S. Kuhn. – Chicago: Univ. of Chicago Pr., 1962, – 168 c.

340. Microsoft Visual Studio. Visual Studio 2010 Professional. Требования к программному обеспечению. Требования к оборудованию. [Електронний ресурс] // Веб-сайт фірми Microsoft. – режим доступу: URL : http://www.microsoft.com/visualstudio/ru-ru/products/2010-editions/ professional/system-requirements. – Назва з екрана.
341. Microsoft Windows XP Professional Edition. Системные требования. [Електронний ресурс] // Веб-сайт фірми Microsoft . – режим доступу: URL : http://www.microsoft.com/rus/smb/products/os/winxp/requirements.mspx. – Назва з екрана.
342. Microsoft. Разработка приложений C#. [Електронний ресурс] // Веб-сайт фірми Microsoft. – режим доступу: URL : http://msdn.microsoft.com/ru-ru/vcsharp/aa336794.aspx. – Назва з екрана.
343. Microsoft. Центр загрузки. Microsoft Visual Studio Learning Pack 2.0 [Електронний ресурс] // Веб-сайт фірми Microsoft . – режим доступу: URL : http://www.microsoft.com/ru-ru/download/details.aspx?id=22347. – Назва з екрана.
344. Mono is a cross platform, open source .NET development framework. [Electronic Resource] // Web site «Mono». – Mode of access : URL : http://www.mono-project.com/Main_Page Title from the screen.
345. MonoDevelop 2.2 Released. [Electronic Resource] // Web site «MonoDevelop». – Mode of access : URL : http://monodevelop.com/Download/MonoDevelop_2.2_Released Title from the screen.
346. Moore G. E. Cramming more components onto integrated circuits / G. E. Moore // Electronics Magazine. – 1965. – N 8. – p. 114–117.
347. MSSOFT.RU. Программное обеспечение. Системные требования: Visual Studio 2008 Professional. [Електронний ресурс] // Веб-сайт «Mssoft.ru программное обеспечение». – режим доступу: URL :

http://www.mssoft.ru/Makers/Microsoft/Visual_Studio_2008_Professional/SysReq/. – Назва з екрана.

348. Osgood C. E. The Measurement of Meaning / C. E. Osgood, G. J. Suci, P. H. Tannenbaum – Urbana.: – 1957. – pp 290–304.
349. Paint.NET – Home [Electronic Resource] // Web site «Getpaint.net». – Mode of access : URL : http://www.getpaint.net/ Title from the screen.
350. Paint.NET. From Wikipedia, the free encyclopedia. [Electronic Resource]. – Mode of access : URL : http://en.wikipedia.org/wiki/Paint.NET Title from the screen.
351. Paint.NET. Український сайт підтримки Paint.NET [Електронний ресурс] . – режим доступу: URL : http://paintnet.org.ua/. – Назва з екрана.
352. Shankar, K. Data Design: Types, Structures, and Abstractions. / K. Shankar / Handbook of Software Engineering. – New York, NY: Van Nostrand Reinhold, – 1984. p. 253.
353. Simple C# – Программирование с нуля. Введение в C Sharp и .Net. [Електронний ресурс]. – режим доступу: URL : http://simple-cs.ru/store/csharp/1/ . – Назва з екрана.
354. Small Basic [Електронний ресурс] // Центр начинающего разработчика. – режим доступу: URL : http://msdn.microsoft.com/ru-ru/beginner/cc950524.aspx. – Назва з екрана.
355. Smalltalk?! Объектно-ориентированное программирование. [Електронний ресурс] // Веб-сайт «Smalltalk по-русски» . – режим доступу: URL : http://www.smalltalk.ru/articles/smalltalk.html. – Назва з екрана.
356. Software Engineering 2004: Curriculum Guide Lines for Undergraduate Degree Programs in Software Engineering [Electronic Resource] // Web site «EEE Computer Society». – Mode of access : URL : http:/www.computer.org/portal/cms_docs_ieeecs/education/cc2001/SE2004Volume.pdf Title from the screen.

357. Timus Online Judge. Архив задач с проверяющей системой. [Електронний ресурс] . – режим доступу: URL : http://acm.timus.ru/. – Назва з екрана.
358. Visual C# Home. Язык C#. Центр разработки на Visual C#. [Електронний ресурс]. – режим доступу: URL : http://msdn.microsoft.com/ru-ru/vcsharp/default.aspx . – Назва з екрана.
359. Weinberg G. M. The Psychology of Computer Programming. / G. M. Weinberg. – New York: Van Nostrand Comp., 1971. – 279 p.

# ДОДАТКИ

## Додаток А. Методичні рекомендації з навчання програмування мовою C# в класах технологічного профілю ЗНЗ

### 1. Особливості об’єктно-орієнтованої парадигми програмування

Традиційно ми звикли, що в основі програмування лежить поняття алгоритму. Втім, так само як комп’ютер «мертвий» без програм, що ним виконуються, алгоритм не має жодної користі без даних, які він обробляє. Сукупність алгоритмів та даних і являє собою те, що у програмуванні називають об’єктом. ООП (об’єктно-орієнтоване програмування) – це технологія розробки програмного забезпечення. Структура даних у програмі, розробленій за принципами ООП, дуже нагадує те, як людина сприймає відомості про оточуючий світ. Адже людина із нерозривно пов’язаної в одне ціле природи виокремлює у своїй уяві певні об’єкти, поіменовує їх, досліджує та класифікує. Навіть можна сказати, що об’єктно-орієнтовані програми нагадують за своєю будовою організацію світу, який нас оточує. Є окрема наукова теорія ООП, але початківцям найкраще пояснювати загальні поняття на конкретних життєвих прикладах.

#### 1.1 Приклади, що дозволяють пояснити на аналогіях взаємодії об’єктів у комп’ютерній програмі

Існує аналогія між особливостями взаємодії об’єктів у комп’ютерній програмі та порядком виконання доручень, що люди дають один одному в повсякденному житті. Розглянемо конкретний приклад.

##### 1.1.1 Передача завдання

Одного разу у мене виникло бажання подарувати своїй матусі пральну машину-автомат. Нині це нескладно. Я дуже швидко вибрав у магазині найкращу, на мій погляд, модель. Але це ще не все. Подарунок потрібно

привезти у квартиру моєї мами, змонтувати у належному місці та запустити в дію. Але ж я такого ніколи не робив! За аналогією з традиційним підходом до написання комп'ютерних програм мені доведеться не лише навчитися слюсарній та автомобільній справі, а й придбати необхідне обладнання, включаючи вантажний транспортний засіб.

Об'єктний підхід дозволяє передоручити певне завдання іншому об'єкту. Отже, я телефоную знайомому слюсарю, аби він відвіз і встановив пральну машину у квартирі моєї мами. Я можу бути впевнений, що машина буде належним чином відтранспортована і введена в дію. Механізм, що я використав, – це пошук відповідного агента (слюсаря) і передача йому завдання. Обов'язком агента є необхідність виконати моє завдання. Слюсар реалізує певний алгоритм – метод, щоб виконати моє завдання. Мені необов'язково знати, якими методами він виконає завдання. Методи агента навіть можуть бути мені невідомі. Агент вільний у виборі методу вирішення завдання.

Можливо, якби я поцікавився методами агента-слюсаря, то виявив би, що слюсар не буде виконувати таке просте завдання сам, а доручить його помічнику. Передача завдання помічнику буде обумовлена ще й тим, що у помічника є власний автомобіль. Помічник залучить ще когось, щоб той допоміг йому завантажити покупку. Моє завдання все рівно буде виконане через послідовність завдань, які агенти надішлють один одному.

### 1.1.2 Деякі принципи ООП

Першим принципом об'єктно-орієнтованого програмування є спосіб **задавання завдання**. Хоча це і звучить дещо тавтологічно, але завдання мало сформулювати, його важливо комусь задати.

Завдання в об'єктно-орієнтованому програмуванні ініціюється за допомогою передачі повідомлень агенту (об'єкту), що є відповідальним за завдання. Повідомлення містить вказівку на виконання завдання і супроводжується додатковими даними (аргументами), необхідними для його

виконання. Одержувач – це агент, якому надсилається повідомлення. Якщо він приймає повідомлення, то на нього автоматично покладається відповідальність за виконання зазначеної дії. Як реакція на повідомлення, одержувач запустить деякий метод, щоб задовольнити отриману вказівку.

Незалежність агента забезпечує принцип маскування даних (ще цей принцип називають **інкапсуляцією**) щодо пересилання повідомлень, згідно з якими клієнтові не потрібно знати про засоби, за допомогою яких його запит буде виконано.

Спочатку клієнт, якщо в нього є завдання, повинен знайти, кому можна було б його доручити. Це також певний принцип. Якщо програміст довгий час розробляє код традиційними методами, то цілком нормальне вміння «перекласти свої завдання на чужі плечі» в нього майже повністю атрофувалося. Практично всяка мова програмування дозволяє йому самому розробити кожен окремий елемент програми від самого початку, не звертаючись до послуг ще когось.

У традиційних мовах програмування приховування даних також є важливим принципом. У підпрограмах (процедурах чи функціях) внутрішня структура відділена від зовнішньої певними рівнями доступу. Але є відмінності між викликом процедури і передачею «доручення». Перша полягає в тому, що в «доручення» є цілком конкретний одержувач – агент, якому надіслано повідомлення. Цей агент повністю самостійний. При виклику процедури немає настільки явно виділеного одержувача. Точніше процедура не володіє повною незалежністю, а виконується наче під наглядом основної програми. Ще одна відмінність полягає в тому, що інтерпретація повідомлення (тобто метод, що викликається після прийому повідомлення) залежить від одержувача і є різною для різних одержувачів. Це так званий **поліморфізм**. Я можу передати моє повідомлення, наприклад, моїй дружині, і вона його зрозуміє, і як результат завдання буде виконано (а саме пральна машина буде встановлена у квартирі моєї матусі). Однак метод, який використовує моя

дружина для виконання запиту, буде іншим, ніж той, який застосує слюсар у відповідь на той же самий запит. Я більш ніж впевнений, що моя дружина просто переадресує завдання встановити пральну машину своїй подрузі – дружині слюсаря. Можливо, в такому випадку завдання буде виконано найкраще, але такі нюанси вже далеко виходять за межі ООП.

З наведених прикладів не слід робити висновок, що одне і теж завдання можна доручати кому завгодно. Якщо я попрошу потурбуватися про встановлення пральної машини мого сина, в нього може не виявитися відповідного методу для розв’язання поставленого завдання. Але якщо мій син не сприйме цей запит, то він видасть певне діагностичне повідомлення про помилку.

### 1.1.3 Відповідальність та пізнє зв’язування

У традиційному (процедурному) програмуванні використовують так зване раннє зв’язування. Раннє зв’язування між повідомленням (ім'ям процедури або функції) і фрагментом коду (методом) відбувається при традиційному виклику процедури. Лише дана підпрограма і лише вказаним методом має виконати завдання. В ООП має місце пізнє зв’язування. Інтерпретація (вибір відповідного методу, який запускається у відповідь на повідомлення) може бути різною для різних одержувачів. Хоча у повідомлення існує певний одержувач, цей конкретний одержувач фактично лишається невідомим аж до виконання програми. Отже, визначити, який метод буде викликаний, заздалегідь неможливо.

Важливою особливістю ООП є **відповідальність** за виконання завдання. Слюсар вільний у виборі способів виконання завдання, втім саме завдання для нього обов’язкове. Це і задає більш вищий рівень абстракції, незалежність агентів – критичний фактор у складних програмах.

Об'єкти не можуть постійно реагувати на завдання тільки таким способом, що звертаються до інших з вказівкою виконати деяку дію. Це призведе до нескінченного циклу запитів. Пригадуєте, як Собакевич з Чичиковим

пропонували один одному пройти в двері першим? Так-от, у випадку складної комп’ютерної програми ланцюжок (кількість «Собакевичів-Чичикових», що надсилають одному повідомлення пройти першим) може виявитись значно довшим, але суть залишиться та ж сама: програма не зможе виконуватись. Якщо не всі, то хоча б деякі об'єкти повинні виконувати певну роботу, перш ніж звертатися з вказівкою до інших.

### 1.1.4 Наслідування

Ще варто розглянути таке поняття, як наслідування. Сфера моїх життєвих потреб не обмежується вирішенням однієї проблеми – як подбати про побут рідної мами. Я реалізовую багато методів у різних сферах. У мене підростає син, і він у своєму житті, можливо, скористається тими ж методами, що й я. Але я не можу бути впевненим у цьому. Майже повсякчас діти не зважають на досвід своїх батьків. У ООП все багато простіше: методи, що наслідують об’єкти від своїх «батьків» завжди чітко визначені. Як і в реальному житті наслідування допомагає зберегти зроблене «старшими поколіннями», тобто багаторазово використовувати один і той же код.

## 2. Мова програмування C#

Щоб оволодіти об’єктно-орієнтованим програмуванням на практиці, корисно знати та вміти використовувати конкретну мову програмування. Серед когорти мов програмування чільне місце займає мова, що унаслідувала практично усе найкраще від своїх попередників, молода і красива мова програмування C#. Ця мова набуває все більшої популярності. Її освоюють не лише початківці, а й ті програмісти, що мають навики використання інших мов програмування. Ця мова програмування повністю підтримує три основних принципи, що лежать в основі об’єктно-орієнтованого програмування: інкапсуляцію, поліморфізм та успадкування – і широко використовується для навчання об’єктно-орієнтованого програмування.

Поряд із цим, мова програмування C# має цілий ряд інших переваг, що роблять її зручною у використанні для навчання програмування:

–мова програмування C# – мультипарадигмальна (така, що підтримує декілька парадигм). Окрім об’єктно-орієнтованої вона підтримує імперативну та функціональну парадигми;

–мова C# надає можливість створювати програми як для консольного, так і для віконного виконання;

–«C-подібний» синтаксис мови C#, як найбільш поширений, дозволяє фахівцю відносно легко перейти до використання багатьох інших мов програмування;

–існує багато доступних для використання візуальних середовищ програмування мовою C#, (Microsoft Visual Studio Professional, Microsoft Visual C# Express Edition, Sharp Develop, Mono Develop, Borland C# Editor).

–програми, написані мовою C#, можуть виконуватися під управлінням багатьох операційних систем, що підтримують програмні платформи Microsoft .Net Framework чи Mono;

–фірма Microsoft (розробник мови програмування C# та середовища програмування Visual Studio Professional), а також інші розробники середовищ програмування мовою C# надають потужну і різносторонню підтримку своїх впроваджень.

## 2.1 Ресурси Інтернету для підтримки навчання мовою програмування C#

Мова програмування C# все ще мало використовується у навчальних цілях. Існують значні труднощі не тільки щодо навчально-методичної, а й загальнотехнічної підтримки впровадження. Для вирішенням таких проблем зручно скористатися багаточисельними ресурсами глобальної мережі Інтернет. Існує ціла низка навчальних, комерційних, благодійних, аматорських та інших колективів, організацій, що надають потужну підтримку навчання програмування мовою C# в мережі Internet.

Найбільш повно мова програмування C# представлена в мережі Інтернет її розробником, фірмою Microsoft. Ця фірма фактично одноосібно встановлює та розвиває стандарти мови C#, розробляє та супроводжує власний продукт, середовище розробки Visual Studio. Офіційна сторінка продукту доступна за адресою http://www.microsoft.com/visualstudio/en-us/home. Від початку виходу програмної платформи .Net до складу Microsoft Visual Studio обов'язково входить мова програмування C#. Фірма Microsoft здійснює в Інтернеті потужну та різносторонню підтримку як мови програмування, так і середовища розробки. Головний сайт підтримки проекту «Visual C# Developer Center» доступний за адресою http://msdn.microsoft.com/en-us/vcsharp/. Також насиченим джерелом інформації про мову C# є російськомовний блог «Microsoft для преси».

Знайти останню версію та основні матеріали з написання програм мовою C# з використанням середовища розробки Sharp Develop можна на сайті цього продукту http://www.sharpdevelop.com/OpenSource/SD/ . Сайт розробника Mono Develop знаходиться за адресою http://mono-project.com.

В Internet публікуються приклади представлення мови C# загальноосвітніми навчальними закладами. Два сайти сільських шкіл з Хмельницької області містять практичні матеріали з навчання програмування мовою C#. На сайті Малиницького НВК вчитель В. А. Ребрина розмістив матеріали факультативного курсу для 8 класу з програмування на C# (http://malnvk.ucoz.ua/load/1-1-0-1). Також активно розвивається сайт «Творча лабораторія вчителя інформатики Гаврилівської ЗОШ Хмельницької області О. П. Пилипчука», на якому представлені корисні в навчанні матеріали для ознайомлення з мовою програмування C#. Зокрема О. П. Пилипчук демонструє на своєму сайті приклади написання класів C# для виконання обчислень із звичайними дробами, які спеціально розроблені для демонстрації переваг об'єктно-орієнтованого програмування (http://teachlab.ucoz.ua/publ/6-1-0-35).

Ще одним прикладом розміщення матеріалів з навчання програмування мовою C# може слугувати «Автоматизована система навчання «Уроки в Інтернеті» на сайті Миропільської гімназії Романівського району Житомирської області (http://www.ms1.org.ua/workbooks/). Доступ до матеріалів цієї системи мають лише зареєстровані користувачі: учні, вчителі та «гості гімназії». В якості «гостя» може зареєструватись практично будь-який відвідувач сторінки. В старій версії системи «Уроки в Інтернеті» (http://www.ms1.org.ua/lessonsold/) без обмеження доступу можна проглянути уроки, серед яких є розробки з навчання програмування різними мовами та в різних середовищах. Для інформаційної підтримки навчання програмування мовою C# використовується форум цього сайту.

Окремої уваги заслуговують on-line виконавці (компілятори, інтерпретатори), автоматизовані системи перевірки виконання завдань, зокрема проект «E-olimp» (http://www.e-olimp.com.ua/). Такі системи є ефективними Інтернет-ресурсами для навчання програмування.

## 3. Методичні особливості навчання C# в консолі

### 3.1 Найпростіші програми мовою C#

У мові C# та ще багатьох інших мовах програмування за допомогою найпростішої програми можна пояснити учням призначення роздільників в алфавіті мови програмування, розтлумачити значення операторних дужок, продемонструвати можливість розділяти текст коду будь-якою кількістю проміжків, нових рядків, можливість додавати в певних місцях програми будь-яку кількість крапок з комою. При написанні програм будь-якою мовою для зручності перегляду її код розбивають на кілька рядків так, щоб у кожному містилася окрема, відносно незалежна частина програми, іноді доречно сказати, команда або ж оператор. Крім поділу на рядки, для команд створюють відступи від початку рядка. Відступи роблять різні за ієрархічним принципом.

У деяких середовищах програмування мовою C# консольне вікно, де виводяться результати виконання програми, автоматично закривається після завершення її виконання. Отже, щоб переглянути результат виконання, потрібно виконати певні додаткові дії. Щось подібне зустрічається і в окремих середовищах розробки мовою програмування C#. Початківцям навіть нескладна маніпуляція створює певні труднощі. Уникнути передчасного закриття вікна з результатами роботи програми можна, додавши в кінці програми рядок, що містить оператор введення. Оператор введення зупинить програму до натискання клавіші «Enter», і на екрані можна буде бачити останні повідомлення середовища виконання. Застосовувати оператор введення лише для зупинки виконання програми можна без будь-яких параметрів.

### 3.2 Коментарі у програмі

Навіть програми для перших комп’ютерів вже містили досить багато команд. Щоб розібратися чи просто знайти помилку в громіздкій програмі, доводиться докладати значних зусиль. Саме тому практично в усіх мовах програмування передбачено можливість створення так званих коментарів. Коментар – це частина тексту програми, що ігнорується під час виконання і містить певні пояснення та підказки. Коментар завжди відокремлюється від основного тексту програми спеціальними вказівками. Мова C# надає можливість створювати кілька видів коментарів. Знак «//» перетворює в коментар увесь текст після «//» і до кінця цього рядка. Якщо коментар потрібно реалізувати в кількох рядках, то можна користуватися вказівками «/*» та «*/». Все, що знаходиться між цими знаками, незалежно від кількості рядків, при виконанні програми буде ігноруватися. Доцільно додавати коментарі навіть до найпростішої програми.

Коментування можна використовувати для того, щоб тимчасово зробити певну частину програми невиконуваною. Частину тексту програми, яку позначають як коментар, для того щоб вона тимчасово не виконувалась, називають закоментованою. Процес перетворення частини програмного коду в

коментар називають закоментовуванням. Закоментовування використовують для налагодження програми: пошуку помилок, внесення інших тимчасових змін. Наприклад, у програмі, що не виконується через синтаксичні помилки, легко впевнитися, де вони знаходяться, закоментувавши вірогідно неправильні команди.

### 3.3 Виведення на екран даних різних типів

З допомогою оператора виведення також можна проілюструвати особливості основних типів даних. Для цього достатньо внести незначні зміни до найпростішої програми типу «Hello World!». Наприклад, не важко замінити у програмі текст, що виводиться на екран, на якесь невелике ціле число. На перший погляд, виведення на екран цифр відбувається так само, як і текстових даних. Втім, оператор виведення багатьох мов програмування може виконувати над даними практично всі допустимі для цього типу даних операції. Над даними цілочисельного типу можна виконувати додавання, віднімання, множення, ділення націло та визначення залишку від ділення. Тобто, якщо в операторі виведення вказати, наприклад, 2+2 – на екрані з’явиться результат його обчислення – 4. Відмінність між числовими і рядковим типами даних можна продемонструвати, подавши цей же вираз як текст, тобто в лапках. Тепер вираз обчислюватися не буде, і на екрані з’явиться запис аналогічний тому, що записаний як аргумент вказівки виведення «2+2». Крім того, над цілочисельними даними можна виконувати ділення націло та визначення залишку від ділення. Для мови C# ділення націло та визначення залишку від ділення позначаються відповідно «/» та «%». Ділення націло числа 5 на число 2 та залишок від такого ділення на мові C# можна подати відповідно командами System.Console.WriteLine(5/2) та System.Console.WriteLine(5%2). Варто зазначити, що залишок від ділення ще називають «діленням по модулю».

Подавши як аргумент оператора виведення числа у вигляді десяткових дробів, можна зумовити виконання дій над ними, як над дійсними числами. Мова C# автоматично не перетворює значення результатів обчислення дійсних

чисел у форму з плаваючою комою. Тому вказівка System.Console.WriteLine(5.0 * 2.0) видасть на екран 10.0.

Пояснивши існування цілочисельних та дійсних числових типів, варто звернути увагу на відмінність у виконанні операції ділення над цими даними. Для дійсних числових даних ділення позначається однаково – «/». Якщо потрібно отримати точний результат ділення цілих чисел у вигляді дійсного числа, тобто, не як ділення чисел націло і не як залишок від ділення, то виконати таку операцію можна, лише попередньо перетворивши типи даних аргументів з цілочисельного в дійсний. Можна застосовувати як явне, так і неявне перетворення типів і доцільно це розглядати при більш поглибленому вивченні матеріалу.

Дуже часто в різних мовах програмування типи даних з однією і тією ж назвою можуть набувати інших діапазонів можливих значень. Невідповідність типів даних різних мов програмування створює суттєву проблему трансляції програм з однієї мови на іншу. Значною мірою така проблема вирішується використанням програмної платформи Microsoft .Net Framework, що підтримує багато мов програмування. Платформа .Net передбачає єдину систему типів даних для різних мов програмування. Мова програмування C# не лише базується на платформі .Net Framework, а є в рамках цієї платформи основною мовою програмування.

### 3.4 Оголошення даних, присвоєння

Нові змінні в мові програмування C# можна оголошувати в межах певного класу. Існують спеціальні вказівки, що визначають «зону видимості» змінної. Можна вказати, чи буде змінна доступною лише в межах даного класу, чи можна використовувати оголошену змінну в інших класах програми. Фактично оголошувати змінні в мові C# можна майже в будь-якому місці програми.

### 3.5 Оператори введення

Дуже часто користувач сам має вводити дані до програми під час її виконання. Для цього в мовах програмування використовуються оператори

введення. Зазвичай, оператор введення зупиняє програму, надає можливість користувачу ввести дані з клавіатури і, після натискання клавіші «Enter», переносить введені значення у відповідні області пам’яті та продовжує виконання програми.

Оператором «Read» у мові C# передбачено введення лише символьних значень. Отримати з введених користувачем символів числові чи інші дані можна лише використавши явне або неявне перетворення типів.

Оператори введення, аналогічно до оператора виведення, теж буває в двох модифікаціях. Оператори введення мови C# – Read та ReadLine. Обидва різновиди оператора введення мови C# після введення даних обов’язково переводять курсор на новий рядок. Відмінність між операторами Read та ReadLine мови C# полягає в тому, що, якщо перший дозволяє здійснювати введення лише даних символьного типу, то другий можна використовувати для введення рядкової величини. Тобто, якщо оператор Read дозволяє ввести з клавіатури лише один символ, то оператор ReadLine забезпечує введення рядка символів.

Варто зазначити, що оператор введення в мові C# потрібно використовувати в поєднанні з оператором присвоєння. Якщо ми хочемо ввести з клавіатури дані в змінну «a», то потрібно писати a = System.Console.Read().

Забезпечити введення з клавіатури даних типу відмінного від рядкового (символьного) відносно нескладно. Для цього доцільно використовувати перетворення типів командою Convert. Наприклад, щоб ввести до цілочисельної змінної дані з клавіатури, можна використати оператор введення як аргумент оператора перетворення типу даних:

```
int a = System.Convert.ToInt32(System.Console.ReadLine());
```

### 3.6 Розгалуження

Лінійний хід виконання програми можна змінити, використавши оператор вибору, який ще називають «розгалуження». Розгалуження складається з трьох елементів:

1) умови;

2) дії (серії команд), що виконується, якщо умова істинна;

3) дії (серії команд), що виконується, якщо умова хибна.

Якщо розгалуження не містить третього елементу, то його називають неповним. Тобто, відмінність повної і неповної форми розгалуження полягає в тому, що неповне розгалуження не містить ніяких дій (команд) для виконання в тому разі, коли умова хибна.

У мові програмування C# розгалуження записується: if (<умова>) <дії, що виконується, якщо умова істинна> else <дії, що виконується, якщо умова хибна>;. Неповна форма розгалуження записується без «else»: if (<умова>) <дії, що виконується, якщо умова істинна> ;.

### 3.7 Цикли

Цикл – це послідовність команд, що повторюються скінченну кількість разів. У більшості мов програмування використовуються три основних типи циклів: з передумовою, з післяумовою та цикли з параметром. Мова програмування C# теж дозволяє створювати такі типи циклів. Втім вона має особливий тип циклів – «foreach». Цей тип циклів C# не унаслідувала від інших мов програмування [6]. Швидше за все, розгляд циклів «foreach» варто здійснювати лише під час поглибленого вивчення програмування.

Цикли всіх типів обов’язково мають «тіло циклу». Тіло циклу – це, власне, і є ті команди, що повторюються. Одне окреме повторення тіла циклу називають ітерацією. Кількість повторень, відповідно, звуть кількістю ітерацій циклу. Цикли з після та передумовою, окрім «тіла», містять ще «умову». Умова – це логічна величина. Якщо в циклах з передумовою вона перевіряється до виконання тіла циклу, то в циклах з післяумовою умова перевіряється після кожного виконання тіла циклу. Тобто цикл з післяумовою передбачає хоча б одне обов’язкове виконання тіла циклу.

мові програмування C# цикл з передумовою записується: while (<умова>) тіло циклу;.

У мові програмування C# цикл з післяумовою: do <тіло циклу> while(<умова>);.

Демонструючи відмінність у використанні циклів з перед- та післяумовою, доцільно вказати на можливість практично повної взаємозаміни цих алгоритмічних конструкцій. Наприклад, програму знаходження суми введених з клавіатури чисел, де використовувався цикл з передумовою, можна трохи змінити, щоб команда введення не з'являлась в ній двічі.

Цикли з параметром, окрім «тіла», містять вказівки надання додатковій змінній (параметру) початкового значення та зміни його під час ітерацій. За час виконання циклу з параметром додаткова змінна пробігає певний діапазон значень від свого початкового до кінцевого значення.

У мові програмування C# цикл з параметром характеризується, окрім початкового та кінцевого значень параметру, ще й «кроком». Крок вказує, на яку величину змінюється параметр при кожному виконанні тіла циклу. Поряд із цим у мові C#, на відміну від Pascal, кінцеве значення параметру вказується у вигляді логічної умови. Для того, щоб цикл виконувався, умова повинна бути істинною. Ось так схематично записується цикл з параметром мовою програмування C#: for (<вираз визначення параметра та його початкового значення>, <умова набуття параметром кінцевого значення (припинення циклу)>, <вираз зміни параметра на величину кроку>) тіло циклу;.

### 3.8 Масиви

Окрім простих типів даних, мова програмування має цілу низку так званих структурованих типів. У шкільному курсі програмування детально вивчається лише структуровані типи даних «масиви». Їх ще звуть табличними величинами та матрицями. Масив – це скінченна пронумерована сукупність даних одного типу. Масив складається зі значень, що називаються елементами масиву. Тип даних елементів називають базовим типом даних масиву. Кожен елемент характеризується індексом. Індекс вказує на порядок розміщення елементу в масиві і набуває значень певного перелічуваного типу даних. Для кожного

елементу масиву, крім першого і останнього, обов'язково є наступний та попередній. Індекс попереднього елементу масиву на одиницю менший, а індекс наступного елементу на одиницю більший.

Масиви можна оголошувати подібно до оголошення інших типів даних. У мові C# стосовно масивів діють ті ж самі правила видимості, що й для інших величин.

У мові програмування C# на те, що нова змінна буде масивом, вказують прямокутні дужки після зазначення її базового типу даних. Наприклад: фрагмент коду int[] m – вказує на те, що оголошується цілочисельний масив з іменем m. Специфікація мови C# передбачає два основних способи визначення розміру масиву. Найпростіше вказати розмір масиву, задавши на етапі оголошення усю множину значень його елементів. У такому разі кількість вказаних елементів масиву і визначить його розмір. Наприклад, запис int[] m = {1,0,2,9,3,8,3,7,4,6,5}; вказує на те, що масив m складається з десяти елементів. Схематично оголошення масиву, в поєднанні із зазначенням усіх його елементів, записується так: <тип даних>[] <ім'я масиву> = {<значення першого елементу>,< значення другого елементу >,…,< значення останнього елементу >};. Якщо на етапі оголошення масиву не потрібно вказувати його значення, тобто доцільно залишити масив невизначеним, запис про оголошення масиву набуде наступного схематичного вигляду <тип даних>[] <ім'я масиву > = new <тип даних>[<кількість елементів>];. Наприклад, запис int[] m = new int[10] оголошує масив цілочисельних даних з іменем m, що має розмір 10 елементів. У наведених прикладах запису масиви мови C# індексуються цілими числами починаючи з нуля.

Розв'язання задач з використанням масивів спирається на деякі основні алгоритми роботи з табличними величинами: введення даних до масиву, виведення значень масиву, визначення суми елементів масиву, знаходження найбільшого (найменшого) елементу масиву, пошуку елементів або їх кількості за певними умовами, сортування елементів масивів.

Можна організувати введення даних до масиву з клавіатури та виведення їх на екран, використавши цикли. Найчастіше для роботи з масивами використовують цикли з параметром. Параметр циклу вказує на індекс елементу масиву, до якого відбувається звернення.

Для визначення суми елементів масиву потрібно оголосити змінну, що цю суму буде накопичувати.

Знаходження найбільшого елементу масиву потребує використання розгалуження:

Велику кількість задач можна запропонувати учням на пошук елементів або їх кількості за певними умовами. Наприклад, розглянемо програму, що визначає кількість від'ємних елементів масиву:

```
using System;
class My Class
{
 static void Main()
 {
 int[] m = new int[10]; //оголошення масиву з 10-ти значень
 int i; // «i» буде використовуватись як параметр циклу
 for (i = 0; i < 10; i=i+1)
          //  цикл в якому «i» змінюється від о до 9
         m[i] = Convert.ToInt32(Console.ReadLine());
 // тіло циклу – введення значень масиву з клавіатури
 int n = 0; // n – міститиме кількість від'ємних елементів
 for (i = 1; i < 10; i=i+1)
         // цикл організований як попередній
         if (m[i] <0) n=n+1; // тілом циклу є розгалуження
 Console.WriteLine(n); // результат виводиться на екран
 // програма зупиняється до натискання «Enter»
```

```
        Console.Read();
    }
}
```

Однією з класичних задач теорії алгоритмів є сортування (впорядкування) елементів масивів за їх значенням. У шкільному курсі найчастіше вивчаються два загальновідомі алгоритми сортування масивів – метод вибору та метод «бульбашки». Кожен із цих алгоритмів передбачає обмін місцями елементів масиву. Для цього зручно використовувати додаткову змінну базового типу.

Програма сортування масиву методом вибору полягає у визначенні найбільшого елементу масиву та обміну його з першим. Алгоритм повторюється щоразу з меншим числом даних. Елемент, що перед цим був визначений як найбільший, не бере участі в наступних обмінах:

```
using System;
class MyClass
{
    static void Main()
    {
    int[] m = new int[10]; // оголошення масиву
    int i, j, imax, r; // оголошення допоміжних величин
    for (i = 0; i < 10; i=i+1) // цикл в якому і змінюється від о до 9
            m[i] = Convert.ToInt32(Console.ReadLine());
            // введення значень масиву
    for (i = 0; i < 9; i=i+1)//цикл в якому і змінюється від о до 8
        {
        imax = i;
        for (j = i+1; j < 10; j++)
          // цикл в якому і змінюється від і+1 до 9
            if (m[j] > m[imax])
```

```
            // умова обміну  значень комірок масиву
            {
            imax=j;
            r = m[i]; m[i] = m[imax];
            m[imax] = r;
            // обмін комірок масиву
            }
      for (i = 0; i < 10; i=i+1)
            Console.WriteLine(m[i]); //виведення результату
      Console.Read();
      // програма зупиняється до натискання «Enter»
    }
}
```

Сортування масиву методом «бульбашки» ще носить назву «обмінного сортування». Під час обмінного сортування відбувається обмін місцями сусідніх елементів масиву. Сортування припиняється тоді, коли не вдається знайти жодних двох сусідніх елементів, що потребують обміну. В такому разі виконується ітерація циклу, в якій не відбувається жодного обміну елементів масиву, що й слугує сигналом для припинення сортування:

```
using System;
class MyClass
{
    static void Main()
    {
    int[] m = new int[10]; // оголошення масиву
    int i; int r; bool fleg; // оголошення допоміжних величин
    for (i = 0; i < 10; i=i+1)//цикл в якому і змінюється від о до9
     m[i] = Convert.ToInt32(Console.ReadLine()); // введення масиву
```

```
    do // початок циклу з післяумовою
          {
           fleg = false; // покажчик припинення сортування
           for (i = 0; i < 9; i=i+1) // цикл проходження масиву
           if (m[i] > m[i+1])   // порівнняння значень сусідніх комірок
              {
               r = m[i];
               m[i] = m[i+1];
               m[i+1] = r;
               // обмін комірок масиву
               fleg = true;
               // вказує, що сортування не закінчено
              }
          }
    while (fleg); // умова циклу з післяумовою
    for (i = 0; i < 10; i=i+1)
             Console.WriteLine(m[i]);  // виведення результату
    Console.Read();
             // програма зупиняється до натискання «Enter»
    }
}
```

### 3.9 Спрощення навчального матеріалу

Обсяг часу, що відводиться на уроках інформатики для навчання програмування, постійно зменшується. Нова навчальна програма з інформатики для учнів 10-11 класів ЗНЗ, академічний рівень, передбачає всього-на-всього 26 годин навчального часу на вивчення розділу «Основи програмування [7]». Програма з інформатики (рівень стандарту) для знайомства учнів з програмуванням взагалі відводить всього 5 годин теми «Базові поняття програмування [8]». За таких умов вчитель повинен подбати про підвищення

доступності навчального матеріалу. Одним із способів підвищення доступності навчального матеріалу є його часткове спрощення та скорочення.

У наведених вище прикладах використання мови C# свідомо уникнуто використання оператора інкременту. Наприклад, у задачах обробки масивів замість безпосереднього інкременту змінної «i» використовується вираз, що реалізовує інкремент через присвоєння та додавання – «i <присвоїти> i+1».

Таблиця 1. Реалізація інкременту мовою C#.

| Операція | C# |
|---|---|
| Інкремент | i++ |
| Заміна присвоєнням | i = i+1 |

Інкремент (з англійської increment *збільшення)* – операція збільшення аргументу на деяку фіксовану величину або ж, у деяких випадках, на змінну. Зворотну операцію називають декремент (зменшення). Збільшення певної величини на одиницю надзвичайно широко використовується при написанні комп'ютерних програм. Практично в усіх мовах програмування використання спеціальних функцій інкременту прискорює виконання алгоритму, а отже, є одним із засобів його оптимізації. Поряд із цим, саме позначення операції інкременту по впливало на появу назви мови C#. Одна з попередниць C# – мова програмування C++ отримала свою назву від позначення цієї операції. Значок # – став образним втіленням чотирьох плюсів. Такий цікавий історичний факт можна повідомити учням, щоб пробудити інтерес до предмету.

Ще один аспект написання програм – це використання додаткових модулів, бібліотек процедур, функцій, а для мови C# – бібліотек класів. Історично використання комп'ютера пов'язано з виконанням значних математичних обчислень. Дуже важливо продемонструвати учням обчислювальні можливості комп'ютера вже на етапі написання перших комп'ютерних програм. Проте в мові C# для виконання обчислень з використанням тригонометричних функцій, логарифмів та інших математичних

засобів виникає необхідність використання додаткового класу. Платформа .NET Framework містить бібліотеку із звичайними математичними функціями, відому як бібліотека «Math». Розглянемо програму, що за відомими катетами обчислює гіпотенузу прямокутного трикутника.

```
using System;
class Example
{
 static void Main()
 {
 Console.WriteLine(«довжину першого катета 'a'»);
 double a = Convert.ToDouble(Console.ReadLine());
 Console.WriteLine(«довжину другого катета 'b'»);
 double b = Convert.ToDouble(Console.ReadLine());
 double c = (Math.Sqrt (a * a + b * b ));
 Console.Write(«довжина гіпотенузи складає»);
 Console.Write(c);
 Console.Read();
 }
}
```

Навіть такі прості приклади варто відкласти на пізніше адже це потребує додаткового часу на пояснення призначення та використання бібліотек.

## Додаток Б. Інструкції та вказівки вчителям інформатики, що беруть участь в організації педагогічного експерименту навчання програмування в класах технологічного профілю ЗНЗ на основі мови C#

Оскільки питання використання мови C# в навчанні програмування учнів середніх загальноосвітніх навчальних закладів являється практично невивченим, було вирішено провести експериментальну апробацію такого нововведення.

У 2009-2010 та 2010-2011 навчальних роках проводиться педагогічний експеримент з вивчення програмування в класах технологічного профілю з використанням мови програмування C#. До участі в експерименті можуть приєднатися всі бажаючі вчителі інформатики, які викладають її в 10-11 класах загальноосвітніх навчальних закладів за програмою технологічного профілю.

Педексперимент стосується вивчення інформатики в 10 або 11 класі в той період, коли календарним плануванням передбачається вивчення розділу «Основи алгоритмізації та програмування». Навчальною програмою вивчення цього розділу передбачено в 11-му класі. Проте більшість учителів, користуючись правом змінювати порядок вивчення навчального матеріалу, переносять цю тему до 10-го класу.

Створено сайт з питань використання мови C# у школі – http://sites.google.com/site/c4plus/. На сайті організовано методичну підтримку експерименту. До дослідження залучаються школи, де працюють досвідчені педагоги, які не один рік читають програмування в 10-11 класах технологічного профілю. Також до самостійного оволодіння та впровадження мови програмування C# на своїх уроках запрошуються всі бажаючі педагоги середніх загальноосвітніх шкіл, ліцеїв, гімназій, колегіумів.

### Як офіційно оформити свою участь в експерименті?

1. Про це потрібно повідомити педагогічну раду вашої школи. На одній із педрад у вашому начальному закладі можна виступити з повідомленням про те,

що Ви берете участь в експерименті. Оскільки Ви як вчитель, згідно положень шкільної програми, маєте право самостійно обирати мову програмування, то Вам ніхто не може заборонити такий експеримент проводити. Але, якщо в колег будуть запитання, на них варто коротко відповісти, не погано заслухати думки інших учителів, можна й проголосувати.

2. Ваш навчальний заклад повинен укласти договір з ІІТЗН (інститутом інформаційних технологій та засобів навчання) про співпрацю, узгодити програму дослідження і його технічне завдання. Учасниками домовленостей виступають керівники установ, що співпрацюють, а також безпосередні виконавці дослідження: вчитель інформатики школи та організатор дослідження зі сторони інституту.

Зразки документів (договору, технічного завдання, програми дослідження) необхідні для участі школи в педексперименті знайдете на сторінці «Корисні матеріали» у файлі «Зразки.rar» – архів захищено паролем, який я повідомлю особисто!

**Як провести експеримент?**

1. Потрібно визначити класи, в яких проводитиметься дослідження.

2. Потрібно внести зміни до календарного планування у відповідності до вимог програми, за якою вивчатиметься програмування під час дослідження. Рекомендоване календарне планування та програма з інформатики для старших класів технологічного профілю знаходиться на сторінці «Корисні матеріали» у файлі «Календарне планування.doc»

Для календарного планування та організації навчального процесу, за бажанням, можна використати «Електронний журнал вчителя інформатики до вивчення розділу «Основи алгоритмізації та програмування» який теж можна завантажити на сторінці сайту «Корисні матеріали».

3. У комп'ютерному класі школи, на учнівських комп'ютерах потрібно інсталювати середовища програмування мовою C#, попередньо встановивши

відповідну версію Microsoft_.NET. Це може бути Microsoft Visual Studio (безкоштовна версія для C# – Microsoft Visual C# Express Edition) або безкоштовний пакет SharpDevelop (остання версія третя).

Напевно, найпростіша програма зручна для вивчення C# – SharpDevelop перших версій (працює навіть під управлінням Windows 98). Для роботи SharpDevelop 1.1 потрібна Microsoft .NET теж першої версії.

SharpDevelop версії 1.1.0.2124 (має російськомовний інтерфейс) та Microsoft .NET версії 1.1 на російській мові (архів з двох частин) можна завантажити на сторінці «Корисні матеріали» нашого сайту. Файли SharpDevelop_1.1.0.2124_Setup.rar, dotnetfx.part1.rar, dotnetfx.part2.rar відповідно.

З сайту Microsoft (http://www.microsoft.com/exPress/download/) можна встановити безкоштовну версія Microsoft Visual C# 2008 Express Edition – http://www.microsoft.com/exPress/download/#webInstall або завантажити весь Microsoft Visual Studio Express Edition.

4. Проводити уроки інформатики згідно календарного планування, ретельно з'ясовуючи переваги та недоліки використання для навчання мови програмування C#.

Для підтримки навчання програмування мовою C# будуть публікуватися методичні матеріали.

5. По завершенню навчального року зібрати та узагальнити результати успішності в експериментальних класах та інші дані, згідно вказівок до підведення підсумків навчального експерименту.

Про збір та оформлення результатів експерименту детально буде описано на сторінці «Підведення підсумків навчального експерименту» у вигляді вказівок та рекомендацій.

Стосовно участі в експерименті виникли і продовжуються дискусії. Деякі з них намагатимусь розміщувати на сторінці: «Думки щодо педагогічного дослідження»

**Порівняння мов програмування C# та Pascal щодо доцільності їх використання як засобів навчання**

Перше, що розрізняє ці мови, це те, що мова програмування Pascal створювалась як процедурна, а C# – як об’єктно-орієнтована. Для мови Pascal існує її діалект Object Pascal, що підтримує технологію об'єктного програмування. Однак, використання цієї мови значно ускладнює підхід до навчання, що розпочинається знайомством з об’єктами в програмуванні. До того ж синтаксис Pascal мало використовується при професійному написанні програм. Нинішня популярність мови Pascal в освіті обумовлена тим, що вона була спеціально розроблена і широко використовується як навчальна. Широкому застосуванню цієї мови у навчанні також сприяє потужне методичне забезпеченням, яке накопичилося протягом довгих років її використання в школах та вузах різних країн.

Певна проблема співтовариства об'єктно-орієнтованих програмістів полягає в поширеності різних термінів для позначення подібних ідей. Так, у мові Object Pascal клас називається «об'єктом» (тип даних object), а «надкласом» називають батьківський клас, клас-предок і т. д. Ці розбіжності іноді переносяться на процес навчання програмування. Під час навчання ООП на основі різних мов програмування доцільно дотримуватись єдиних угод не лише в термінології, а й у стилях оформлення та запису тексту програм.

Створюючи нову мову C#, її розробники значно вдосконалили саме синтаксис мови C++. Побутує думка, що мова C# досить проста в сприйнятті та надзвичайно зручна, як для професійного програмування, так і для навчання його основ. Мова повністю об’єктно-орієнтована. Втім методична підтримка для C# поки що є недостатньою. Так, на Україні все ще не існує жодного рекомендованого МОНмолодьспорту підручника для її вивчення в школі. Насамперед, відсутність належної методичної підтримки, при всіх перевагах мови C#, не дозволяє стверджувати про беззаперечну доцільність її використання як засобу навчання.

Розглянемо можливості добору середовища програмування для мов Pascal та C#. Мову програмування Pascal нині здебільшого вивчають з використанням таких середовищ як Turbo Pascal версій 5.0 та 7.0, Free Pascal різних версій, Pascal ABC, Pascal ABC.Net, Delphi. Нині найбільш широко використовуються середовища Turbo та Free Pascal, незважаючи на те, що вони розроблені для морально застарілих операційних систем типу DOS. Ці середовища нормально працюють під управлінням Windows XP, хоча і не підтримують багатьох його можливостей. Середовище Delphi не має вільно розповсюджуваних версій, середовище ABC Pascal спирається на специфічний діалект мови Pascal та не має, можливо, поки що засобів візуального програмування.

В той же час мова C# забезпечена достатньою кількістю середовищ розробок, орієнтованих на сучасні і популярні операційні системи про що вже зазначалося в даному дослідженні.

Отже, за результатом порівняння мов програмування, а також середовищ розробки для них, Pascal має переваги за простотою синтаксису та наявності значної методичної підтримки. Перевагами мови C# є зручність вивчення об’єктно-орієнтованого програмування, наявність значної кількості вільно доступних середовищ розробки, переважна більшість з яких підтримує візуальне програмування та повна придатність для забезпечення професійної діяльності фахівців-програмістів.

## Додаток В. Календарно-тематичне планування теми «Основи алгоритмізації та програмування» навчання інформатики за програмою для 10-11 класів загальноосвітніх навчальних закладів фізико-математичного, природничого і технологічного профілів

Автор Т. Г. Проценко, з деякими змінами П. Г. Шевчука на основі матеріалів: Т. Г. Проценко Календарно-тематичне планування навчання інформатики за програмою 10-11 класів загальноосвітніх навчальних закладів фізико-математичного, природничого і технологічного профілів / Т. Г. Проценко // Інформатика та інформаційні технології в навчальних закладах. – 2006. – №4/5. – С. 107 – 114.

| № по р. | № теми і № уроку з теми | Тема уроку | Дата проведення уроку | |
|---|---|---|---|---|
| | | | | |
| 9. **Основи алгоритмізації та програмування (48 год.)** | | | | |
| 9.1. **Інформаційна модель (2 год.)** | | | | |
| 19 | 9.1.1 | Об'єкт. Класифікація та наслідування об’єктів. Модель. Моделювання об’єктів. Поняття інформаційної моделі | | |
| 20 | 9.1.2 | Основні етапи розв'язування прикладної задачі за допомогою комп'ютера. Побудова моделі | | |
| 9.2. **Алгоритми (5 год.)** | | | | |
| 21 | 9.2.1 | Методи об’єктів. Алгоритмічна сутність методів. Поняття алгоритму. Властивості алгоритмів | | |
| 22 | 9.2.2 | Способи опису об’єктів та алгоритмів. Виконавець алгоритму. Основні алгоритмічні структури. Аналіз алгоритму | | |
| 23 | 9.2.3 | Порядок складання алгоритмів. Конструювання алгоритмів «зверху донизу» | | |

| | | | | |
|---|---|---|---|---|
| 24 | 9.2.4 | Структурний підхід до побудови алгоритмів. Сучасна галузь розробки програмного забезпечення. Об'єктно-орієнтоване програмування | | |
| 25 | 9.2.5 | ***Тематична атестація з теми:*** «Інформаційна модель. Алгоритми» | | |
| 9.3. **Програма. Мова програмування** ***(10 год.)*** | | | | |
| 26 | 9.3.1 | Поняття програми. Мови програмування. Поняття середовища програмування | | |
| 27 | 9.3.2 | Класифікація мов програмування. Поняття редактора, транслятора, налагоджувача. Інтерпретація та компіляція. Компілятор. Поняття про системи програмування. Інтегровані середовища програмування | | |
| 28 | 9.3.3 | Алфавіт мови програмування. Основні поняття мови: службові слова, ідентифікатори | | |
| 29 | 9.3.4 | **Практична робота № 6.** Основні прийоми роботи в середовищі програмування | | |
| 30 | 9.3.5 | Величини та їх опис мовою програмування. Дані, типи даних. Стандартні типи даних. Змінні та константи | | |
| 31 | 9.3.6 | Арифметичні вирази. Стандартні функції. Бібліотеки підпрограм чи класів. | | |
| 32 | 9.3.7 | Структура програми. Уведення даних з клавіатури. Виведення даних на екран дисплею. Оператор присвоєння. | | |
| 33 | 9.3.8 | Створення лінійних програм | | |
| 34 | 9.3.9 | **Практична робота № 7.** Створення лінійних програм | | |
| 35 | 9.3.10 | **Тематична атестація з теми**: «Програма. Мова програмування» | | |
| 9.4. **Звернення до алгоритмів і функцій** ***(5 год.)*** | | | | |
| 36 | 9.4.1 | Поняття підпрограми (метод як підпрограма). | | |

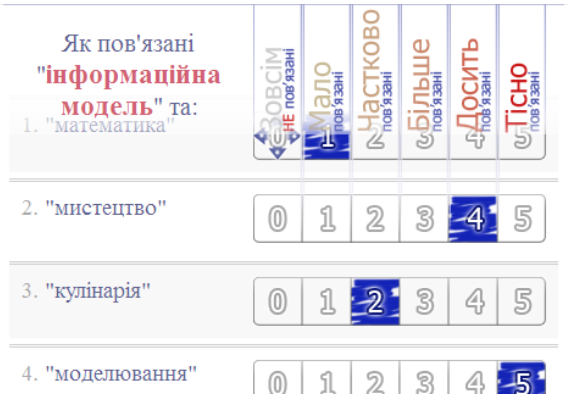
| | | | | |
|---|---|---|---|---|
| | | Процедури. Функції | | |
| 37 | 9.4.2 | Формальні та фактичні параметри | | |
| 38 | 9.4.3 | Область видимості змінних | | |
| 39 | 9.4.4 | **Практична робота № 8.** Створення програм з процедурами та функціями | | |
| 40 | 9.4.5 | ***Тематична атестація з теми: «Використання допоміжних алгоритмів»*** | | |
| 9.5. ***Вказівки повторення та розгалуження (10 год.)*** | | | | |
| 41 | 9.5.1 | Оператори умовного переходу та вибору | | |
| 42 | 9.5.2 | Розв'язання задач з використанням операторів умовного переходу та вибору | | |
| 43 | 9.5.3 | **Практична робота № 9.** Створення програм з використанням операторів умовного переходу та вибору | | |
| 44 | 9.5.4 | Поняття циклу. Цикли з передумовою та з післяумовою. Вкладені цикли | | |
| 45 | 9.5.5 | Створення програм з використанням циклів з передумовою та післяумовою | | |
| 46 | 9.5.6 | **Практична робота № 10.** Створення програм з використанням циклів | | |
| 47 | 9.5.7 | Цикли з параметром | | |
| 48 | 9.5.8 | Створення програм з використанням циклів з параметром | | |
| 49 | 9.5.9 | **Практична робота № 11.** Створення програм з використанням циклів з параметром | | |
| 50 | 9.5.10 | ***Тематична атестація з теми: «Вказівки повторення та розгалуження»*** | | |
| 9.6**.** ***Масиви (10 год.)*** | | | | |
| 51 | 9.6.1 | Обробка структурованих типів даних. Масиви | | |
| 52 | 9.6.2 | Алгоритм роботи з таблицями. Введення та виведення значень масиву з використанням циклів | | |

| | | | | |
|---|---|---|---|---|
| 53 | 9.6.3 | Знаходження суми елементів масиву. Типові завдання обробки масивів | | |
| 54 | 9.6.4 | Алгоритм пошуку елементів масиву із деякою властивістю | | |
| 55 | 9.6.5 | **Практична робота № 12.** Складання та реалізація програм опрацювання одновимірних масивів | | |
| 56 | 9.6.6 | Поняття сортування. Сортування одновимірних масивів методом вибору | | |
| 57 | 9.6.7 | Сортування одновимірних масивів методом обміну | | |
| 58 | 9.6.8 | **Практична робота № 13.** Складання та реалізація програм сортування одновимірних масивів | | |
| 59 | 9.6.9 | Поняття двовимірного масиву. Обробка такого масиву | | |
| 60 | 9.6.10 | **Тематична атестація з теми**: «Масиви» | | |
| 9.7. **Рядкові величини** ***(5 год.)*** | | | | |
| 61 | 9.7.1 | Поняття рядка. Операції над рядковими величинами. Стандартні процедури та функції для обробки рядкових величин | | |
| 62 | 9.7.2 | Найпростіші алгоритми роботи з рядками | | |
| 63 | 9.7.3 | Створення алгоритмів для обробки рядкових величин | | |
| 64 | 9.7.4 | **Практична робота № 14**. Створення алгоритмів для роботи з рядками у середовищі програмування | | |
| 65 | 9.7.5 | **Тематична атестація з теми: «Робота з рядками. Графічні операції»** | | |
| 66 | | **Підсумковий урок** | | |

## Додаток Г. Інструкція щодо роботи з сайтом для анкетування методом семантичного диференціалу

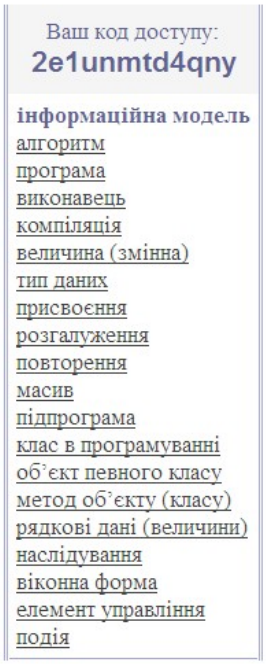
Учасник анкетування зайшовши на сайт http://sd.ms1.org.ua, натискає кнопку з написом «Учень» або «експерт» відповідно до того ким він являється. Після цього проходить його реєстрація, що полягає в заповненні полів реєстраційної форми.

Наступним кроком є оцінювання того як тісно пов’язані пари запропонованих понять. Для оцінювання використовуються оцінки в діапазоні від 0 до 5, де




0 – поняття зовсім не пов’язані
1 – мало пов’язані
2 – частково пов’язані
3 – більш пов’язані ніж непов’язані
4 – досить пов’язані
5 – тісно пов’язані між собою

Порівняння відбувається між двома окремими списками понять. Кожне поняття із першого списку порівнюється із кожним елементом другого. Оскільки частина понять у списках повторюється, то робота системи виключає повторне оцінювання.




В меню, що знаходиться зліва, користувач може сам вибрати поняття для порівняння.

Користувачеві пропонується оцінити всі комбінації пар, тому процес опитування триває досить довго і потребує розумових зусиль, через це тест можна перервати на деякий час і продовжити його, ввівши свій код доступу на головній сторінці.

Для зручності роботи з сайтом на сторінці присутня панель налаштувань, яка дозволяє вибирати відповідний спосіб заповнення анкет. Самий оптимальний спосіб передбачає використання клавіатури з допомогою якої можна задати оцінку від 0 до 5, натискаючи на відповідні клавіші, при цьому перехід до наступної пари відбувається автоматично.

## Додаток Д. Посібник-довідник для учнів «C# коротко»

**Сайт підтримки: https://sites.google.com/site/c4plus/**

**Середовища програмування**

Microsoft Visual C# Express Edition;

Sharp Develop;

Mono Develop;

Antechinus C# Editor – http://www.c-point.com/c_sharp_editor.php

**Що ж таке програма?**

Сучасні програми найчастіше будують так, щоб вони якомога повніше відображали явища оточуючого світу. Найкраще це можна реалізувати, використовуючи об'єктне моделювання та **об'єктно-орієнтоване програмування**.

Комп'ютерна програма – це інформаційна модель реальної системи чи предмету. Щоб побудувати модель, варто визначити об'єкти, класи об'єктів, що складають систему та описати їх мовою програмування. Програма складається з елементів, що моделюють реальні об'єкти та взаємодію між ними.

Нехай нам треба створити програму яка б допомагала керувати тваринницькою фермою. Найперше ми повинні визначити, з яких об'єктів складається модельована нами система – ферма. Далі визначити властивості (характеристики) цих об'єктів та те, як вони між собою взаємодіють: як одні об'єкти породжують інші, як вони впливають на інші об'єкти, які є канали передачі інформації.

Конкретними об'єктами на фермі може бути, наприклад, власне сама «ферма пп. Кіндрат», «корівник №1», у корівнику: «корова Зірка», «корова Муня», «теля Орлик», «доярка Марія Миколаївна». Ферму охороняє об'єкт: «сторож Вася», і т.д. Наша програма може моделювати й іншу ферму, де конкретні об'єкти будуть в інших кількостях, носитимуть не такі імена, матимуть інакші характеристики.

Програма повинна забезпечити створення всіх можливих об'єктів у необхідних кількостях. Конкретні об'єкти з'являються лише під час її виконання. Коли ми будемо групувати об'єкти та визначати як вони пов'язані використовується поняття «**класи»**. Сама по собі програма містить певні класи,

на основі яких, створюються конкретні об’єкти. **Класи** – це сукупності об’єктів, що мають спільні властивості. Стосовно спільних властивостей тварин, що утримуються на фермі, то вони, наприклад, характеризуються: масою, розмірами, споживають певні корми в конкретних кількостях і т.д. Тобто, є підстави об’єднати усі тварини в один клас.

Варто передбачити список усіх необхідних класів. Наприклад методом «мозкового штурму» можна назвати максимальну кількість можливих класів і залишити ті, що не повторюються та не пересікаються:

| | | | |
|---|---|---|---|
| Ферма | Вівця | Свинарка | Комбікорм |
| Корова | Обора | Вівчар | Силос |
| Теля | Стійло | Миша | Вода |
| Свиня | Свинарник | Сіно | Склад |
| Порося | Доярка | Буряки | Сторож. |

Іноді класи доцільно об’єднати в більш загальні: «худоба», «персонал», «корми», «будівлі». Також може виникнути завдання розділити деякі класи на підкласи. Наприклад, худобу можна поділити на «ВРХ», «свині», «вівці». У такому випадку доцільно використовувати так зване ***наслідування*** – важливий механізм об’єктно-орієнтованого програмування. Описати необхідність споживати корм можна для усього класу тварин. При цьому клас «ВРХ» буде споживати сіно, а клас «свині» – комбікорм.

Класифікація об’єктів на фермі може здійснюватись по-різному, наприклад за розміщенням на території ферми: «корівник» об’єднує «корів», «телят» та «доярок», а клас «свинарник» буде мати підкласи: «свиней», «поросят», «свинарок». Класи можна структурувати за будь-якими ознаками: датою появи на фермі, вартістю, масою, т. і.. В будь-якому випадку утворюється певна чітка ієрархія класів програми. Яким чином формувати класи, визначає програміст, і від цього багато в чому залежить механізм роботи моделі (програми), її ефективність. Значно спростити написання програми може така класифікація, в якій якомога більше особливостей загальних класів у**наслідуються** їх підкласами.

Кожен клас характеризується певними властивостями (**полями**) та можливостями (**методами**). Наприклад, властивостями корови є кличка, колір шерсті, маса, кількість споживання корму. Можливостями корови є можливість ревіти, споживати корм, віддавати молоко. Конкретний об’єкт класу корови, наприклад корова «Зірка», має чорнорябу шерсть, масу 470 кг., споживає на добу 40 кг сіна і т.д.. Задаючи поля та методи класу, ми визначаємо властивості

та можливості які повинні бути описані в кожного з об’єктів даного класу. У кожного конкретного об’єкту те чи інше поле має своє певне значення.

**Події** – це зміни, що відбуваються з об'єктом. У реальному світі події відбуваються навколо нас безперервно. Наприклад, коли корова «Зірка» зголодніла, ця подія відбувається з нею. Події, зазвичай, являють собою дії, що впливають на певний об'єкт, але сам об'єкт вплинути на них не може. У комп'ютерному світі подіями можуть бути «натискання кнопки» або «переміщення миші».

Дії виконуються об'єктом. Наприклад, корова «Зірка» лиже свою шерсть – це дія. Користувач бажає, щоб комп’ютер теж міг виконати дію, наприклад, перегорнув сторінку тексту при натисканні на відповідну кнопку.

Хочете вірте, хочете ні, але комп'ютер не має уявлення про те, як гортати сторінки. Тому необхідно дати опис, який буде зрозумілим комп'ютеру – метод зміни зображення сторінки на наступну. Метод являє собою набір покрокових інструкцій, що визначають порядок дій, подібно до рецепта.

**Завдання та запитання:**

1. *Запишіть в зошит та коротко обґрунтуйте перелік усіх класів, які, на вашу думку, повинна містити програма, що моделює роботу зоопарку.*

**Класи та об’єкти в роботі комп’ютера**

Продемонструвати принцип об’єктної розробки програм можна на прикладі функціонування операційної системи Windows. Операційна система призначена для узгодження роботи багатьох прикладних програм. Наприклад ви бажаєте малювати в редакторі «Paint» і при цьому слухати музику з допомогою програми «Windows Media Player». Кожна готова до запуску програма – це практично окремий клас. Якщо запустити програму на виконання, наприклад відкрити Paint, то в операційній системі утвориться конкретний об’єкт цього класу, об’єкт класу «Paint». До того ж таких об’єктів у системі може існувати декілька – ніхто не забороняє запустити кілька екземплярів програми Paint і в кожному з них виконувати дещо інше завдання. Вони всі належать одному й тому ж класу Paint, але кожен з них має свої власні значення властивостей цього класу та дещо інше застосування його методів. Програма «Media Player», під час відтворення музики – це також об’єкт операційної системи, але зовсім іншого класу ніж редактор «Paint».

Подібним чином відбувається взаємодія об’єктів і в самій прикладній програмі. Наприклад користуючись Paint, ви хочете зобразити еліпс. Вибравши відповідний пункт панелі інструментів на екрані, з’являється об’єкт класу

«еліпс». Під час побудови користувач задає властивості еліпса: розмір, колір, товщину лінії.

**Завдання та запитання:**

2. *Запустіть чи пригадайте гру «Тетріс». У клітинках зошита замалюйте типові об'єкти з яких будуються шари фігур у цій грі. Запишіть у зошит перелік усіх класів, що, на вашу думку, містить ця програма. Назви та пояснення класам можете давати на свій розсуд.*

**Класи у програмі**

Мова програмування C# має певний алфавіт (набір елементів, лексем, з яких можна будувати програму) та синтаксис (домовленості про взаємне розміщення лексем у програмі). Найбільш загальним правилом побудови програм на C# є те, що усяка програма є класом або сукупністю окремих класів. Кожен окремий клас має своє ім'я (ще кажуть ідентифікатор) та вміст класу. Ім'я пишеться після слова class, а вміст класу пишеться відразу після його імені та обов'язково вміщується у фігурних дужках. Схематично це позначають так: class <Ім'я класу> {вміст класу}. Наведемо приклад оголошення класу з іменем MyClass

```
class MyClass
    {

    }
```

Вміст класу – це певні поля та методи. Поки їх не задати програма виконуватись не може.

**Найпростіша програма**

Сучасна прикладна програма, написана мовою C# може містити багато класів з великим переліком полів та методів, механізмами унаслідування цих властивостей, алгоритмами обробки подій. Навіть найпростіша програма – це вже клас, у якому повинен виконуватись хоча б один метод. Ось приклад такої програми:

```
class FirstClass {static void Main() {}}
```

Метод, з якого розпочинається виконання програми, обов'язково носить назву «Main», англійською перекладається «основний». Саме такий метод обов'язково присутній в усякому проекті написаному мовою C#.

Найпростіша програма – це клас, ім'я якому ми даємо самі, та метод *Main* у цьому класі. Така програма, хоч і нічого не виконує, не містить помилок і без проблем компілюється.

**Порядок написання та виконання програми у Antechinus C# Editor:**

Зберегти програму в файл (бажано у власну папку):  або Ctrl+S.

Відкомпілювати:  або F8. Запустити на виконання:  або Ctrl+F5.

Увага, компілювання не відбулося, якщо є повідомлень про помилку – «Error».

**Завдання та запитання:**

*3. Напишіть, збережіть, відкомпілюйте та запустіть на виконання найпростішу програму.*

**Програма виведення тексту на екран:**

Програмістам часто доводиться вирішувати досить схожі завдання. Однією з переваг мови програмування C# є наявність вже готових класів, що дозволяє значно спростити написання будь-яких програм на основі готових рішень. Готові класи зберігаються в певних бібліотеках. Для зручності доступу бібліотеки об'єднані у так звані простори імен. Простори імен, класи, методи класів, мають ідентифікатори (назви), за якими їх можна використовувати. Щоб отримати доступ до методу певного класу, що належить до бібліотеки, з певного простору імен у програмі створюється оператор за наступною конструкцією:

<Назва простору імен>.<Назва класу>.<Назва методу> (параметри методу);

Наприклад, метод виведення повідомлення на екран в консольному режимі повідомлення «Hello, world!» записується так: System.Console.Write(«Hello, world!»);

Якщо у програмі дуже часто доводиться звертатися до певної бібліотеки класів, то на початку програми службовим словом using вказується ідентифікатор простору імен і це дозволяє не використовувати назву простору імен в інших частинах програми, а звертатися безпосередньо до класів та їх методів.

Ось приклад програми, де таким чином використовується метод виведення повідомлення на екран «*Write*» класу «*Console*», що належить бібліотеці «*System*»:

```
using System;
  class FirstClass
  {
  static void Main()
```

```
    {
        Console.Write(«Hello, world!»);
        Console.Read();
    }
}
```

Існують певні правила розміщення операторів у тексті програми – візуальне форматування коду. Воно виконується відступами, пропусками та табуляцією.

Окремий вираз, оператор бажано повністю розміщувати в одному рядку. (**оператор** – логічно завершена і нероздільна лексема)

{} – операторні дужки, що об'єднують кілька операторів в один, так званий «складовий оператор». Їх виносять на наступний рядок відносно конструкції, щодо якої ці дужки використовуються. Вміст складового оператора варто розташувати правіше відносно операторних дужок що його формують.

**Завдання та запитання:**

*4. Напишіть, збережіть, відкомпілюйте та запустіть на виконання програму виведення на екран повідомлення «I like C#!».*

*5. Доповніть програму кількома операторами виведення на екран текстових повідомлень довільного змісту.*

*6. Використовуючи буфер обміну спробуйте експериментально з'ясувати повідомлення якого найбільшого розміру можна вивести на екран одним оператором.*

**Коментарі у програмі**

Щоб не заплутатися у тексті програми, щоб полегшити її розуміння, використовують так звані *коментарі*. Коментар – це частина тексту програми, що «непомітна» для компілятора. Тобто те, що пишеться у коментарі, на хід виконання програми зовсім не впливає. Перетворивши у коментар частину програмного коду, можна тимчасово уникнути його виконання.

Знак «//» перетворює в коментар увесь текст після «//» і до кінця цього рядка. Якщо коментар потрібно реалізувати в кількох рядках, то можна користуватися вказівками «/*» та «*/». Все, що знаходиться між цими знаками, незалежно від кількості рядків, при виконанні програми буде ігноруватися.

Текст найпростішої програми C# доповнених коментарями:

```
class FirstClass // FirstClass – ім'я класу.
// Це ім'я задає програміст на свій розсуд.
/*Існують певні правила та домовленості, що визначають як потрібно надавати імена класам та методам */
```

```
{
   static void Main()
   {
      System.Console.WriteLine(«Hello, world!»);
      System.Console.ReadLine();
   }
}
```

**Завдання та запитання:**

*7. Відкрийте будь-яку раніше створену програму. Додайте до програми довільні коментарі обох типів. Перевірте чи не вплинуло це на виконання програми.*

**Оператор виведення. Знайомство з типами даних**

Аргумент оператору виведення може бути різного типу. Наприклад, надпис «24» можна вивести на екран різними способами:

System.Console.WriteLine(24); – виведення цілого числа.

System.Console.WriteLine(4+20); – також виведення цілого числа, але уже як результату обчислення. Оператор виведення попередньо виконує обчислення виразу, що є його аргументом.

System.Console.WriteLine(«24») виведення рядкової величини.

Можна навести ще чимало способів, як отримати на екрані одне і те ж зображення числа.

Ось приклади оператора виведення, де схожий аргумент має різний тип.

| Оператор виведення на екран | Текст, що буде виведено на екран | Тип даних аргументу |
|---|---|---|
| System.Console.WriteLine(2+2); | 4 | Цілочисельний |
| System.Console.WriteLine(«2+2»); | 2+2 | Рядковий |
| System.Console.WriteLine(2,0+2,0); | 4,0 | Дійсний |
| System.Console.WriteLine(2+2==4); | True | Логічний |

Оператор виведення WriteLine визначає тип даних аргументу, виконує дії над даним, якщо як аргумент використовується вираз, що містить допустимі операції.

**Завдання та запитання:**

*8. Напишіть програму, яка б містила не менше чотирьох операторів виведення на екран маси вашого тіла у вигляді даних різних типів. Наприклад: 54; 54.0; 5.40000e01, «п'ятдесят чотири».*

**Основні типи даних мови програмування C#**

Будь-яка програма оперує з даними, а дані необхідно десь зберігати.

Оператор виведення зберігає свої аргументи прямо у файлі програми, а на час виконання вони поміщаються в оперативну пам'ять. В залежності від типу цих даних, обсяг відведеної пам'яті буде різним.

Коли у програмі створюються класи, їх поля та методи, а також, коли створюються об'єкти певного класу, в оперативній пам'яті комп'ютера знову ж таки відводяться під них певні області. І в цьому випадку обсяг пам'яті відведений під дані також залежить від їх типу.

Мова C# має тринадцять простих типів даних: Sbyte, Byte, Short, Ushort, Int, Uint, Long, Ulong, Float, Double, Char, Bool. Ці типи даних ще звуть базовими.

**Цілочисельні типи даних**

Цілочисельні типи даних поділяються на знакові – можуть набувати додатних та від'ємних значень. Та без знакові – набувають лише додатних значень.

| Тип | Область значень | Розмір |
|---|---|---|
| Знакові цілочисельні типи | | |
| sbyte | Від -128 до 127 | 8-біт |
| short | Від -32768 до 32767 | 16-біт |
| Int | Від -2147483648 до 2147483647 | 32-біт |
| Long | від -9223372036854775808 до 9223372036854775807 | 64-біт |
| Без знакові цілочисельні типи | | |
| Byte | Від 0 до 255 | 8-біт |
| ushort | Від 0 до 65535 | 16-біт |
| Uint | Від 0 до 4294967295 | 32-біт |
| ulong | Від 0 до 18446744073709551615 | 64-біт |

Над цілочисельними даними крім дій: **додавання, віднімання, множення** ще допустимі **ділення націло** та **визначення залишку від ділення.** Останні позначаються відповідно «/» та «%». Результат ділення націло числа 5 на число 2 та залишок від такого ділення можна вивести на екран відповідно командами

*System.Console.WriteLine(5/2)* та *System.Console.WriteLine(5%2)*. Варто зазначити, що залишок від ділення ще називають «діленням по модулю».

Дійсні дані мають дещо іншу форму запису. Команда (оператор): *System.Console.WriteLine(5.0 * 2.0)* видасть на екран 10.0.

**Дійсні типи даних**

| Тип | Область значень | Розмір, точність |
|---|---|---|
| Float | Від ±1,5*10-45 до ±3,4*1033 | 4 байти, точність – 7 розрядів |
| Double | Від ±5*10-324 до ±1,7*10306 | 8 байт, точність –16 розрядів |

Для дійсних числових даних ділення позначається – «/» і на відміну від цілих значень виводить точний результат. Якщо потрібно отримати точний результат ділення цілих чисел у вигляді дійсного числа, тобто не як ділення чисел націло і не як залишок від ділення, то виконати таку операцію можна лише попередньо перетворивши типи даних аргументів з цілочисельного в дійсний. Можна застосовувати як явне, так і неявне перетворення типів але детально способи перетворення типів ми розглядати не будемо.

**Не числові типи даних**

| Назва типу українською | Назва типу в C# | Область значень | Розмір |
|---|---|---|---|
| Символьний | Char | Від U+0000 до U+ffff | 16-бітовий символ Unicode |
| Логічний | Bool | True або false | 1 байт |

Для логічного типу повинно вистачити одного біта, але реально відводиться один байт. Це тому, що найменший обсяг будь-якого об’єкту в C# – один байт.

Тип даних «String» (рядковий тип даних) – не основний, а похідний. String – це сукупність даних типу Char.

**Оголошення даних, присвоєння**

Щоб оперувати даним, їх потрібно ідентифікувати або ще кажуть, пойменувати. Коли дані ставляться у відповідності до імені (ідентифікатора) змінної, говорять, що змінній присвоїли певне значення. Присвоєння має наступну структуру: <ім’я змінної> <знак присвоєння> <присвоюване значення змінної>. У мові програмування C# знак присвоєння має вигляд «=», тобто його роль виконує звичайний знак «дорівнює». Але присвоєння – це зовсім не те, що порівняння, і про це ще буде сказано.

Присвоїти значення можна лише уже «оголошеній» змінній. Оголошення змінної – це відведення в пам’яті комп’ютера місця для зберігання даних певного типу та призначення певного імені (ідентифікатора) змінної у відповідність цим даним. Оголошення змінної має наступну структуру: <тип даних> <ім’я змінної>. Імена змінних програміст призначає сам.

Обов’язкові вимоги до імені (ідентифікатору) змінної:

– ім’я змінної не повинно співпадати з іншими іменами (ідентифікаторами) даної програми та зі службовими словами мови C#;
– ім’я змінної повинно містити лише латинські літери, арабські цифри та ще деякі додаткові символи;
– ім’я змінної не повинно починатися з цифри;
– ім’я змінної повинно бути скінченним;
– ім’я змінної не повинно містити проміжків (пробілів).

Рекомендовані вимоги до імені (ідентифікатору) змінної:

– ім’я змінної повинно сприяти розумінню програми, розкривати призначення змінної;
– ім’я повинно починатися з малої літери і не повинно містити літер верхнього регістру (прописних) взагалі.

Приклад програми, що містить оголошення та присвоєння:

```
class MyClass
{
   static void Main()
   {
      int a;
      a = 10;
      System.Console.WriteLine(a);
      System.Console.ReadLine();
   }
}
```

У мові програмування C# присвоєння можна записувати відразу після оголошення змінної. Тобто запис команд *int a;* та *a = 10;* можна спростити, об'єднавши в один – *int a = 10*;.

У наведеній програмі оголошується змінна іменована «*a*» цілочисельного типу «integer». У мові C# тип даних – це практично теж саме, що й клас. Точніше, кожен тип даних це клас, а змінна певного типу, це об’єкт даного класу. Між іншим, практично будь-який клас, створений у програмі

користувачем, теж може виступати як тип даних. І створення об’єкту певного класу – це не що інше, як оголошення змінної певного типу.

Змінні є прикладами можливих полів тих класів, в яких вони оголошуються. Різні операції над даними є ніщо інше, як методи певних класів. Коли під час виконання програми поля якогось класу набувають значень – з’являється об’єкт даного класу.

**Деякі нюанси:** Деякі числові типи «більшого обсягу» можна приводити до типів «меншого обсягу» явно. Структура явного приведення даних у поєднання з їх присвоєнням така: <ім’я величини якій треба присвоїти значення> = (<тип даних до якого потрібно привести значення>) <ім'я величини значення якої буде приводитись>. Наприклад є деяка величина «a» типу integer a її значення потрібно зберегти в іншу змінну b типу sbyte.

*int a = 122121;*

*sbyte b = (sbyte) a.*

На жаль, таке присвоєння дещо спотворить початкове значення даних, адже явне приведення типів допускає втрату даних.
Але приведення буде повністю коректним, якщо значення змінної a не перевищуватиме максимального допустимого значення типу змінної b.

Присвоєння значень змінним різних типів дещо відрізняється між собою. Скажімо, для деяких дійсних типів даних. float f = 3.14F

Уже зазначалося, що коли текстовій змінній присвоюється текстове значення, його попередньо беруть в подвійні лапки. Якщо ж змінна символьна, то лапки беруть одинарні.

d = 0.314E1;

decimal g = 3.14M**8//*******-/841

**Завдання та запитання:**

*9. Внесіть зміни до програми з попереднього завдання так, щоб дані попередньо зберігалися у змінних відповідного типу. (Програма передбачала не менше чотирьох операторів виведення на екран числа у вигляді даних різних типів.)*

*10. Зареєструйтесь на сайті **http://www.e-olimp.com/ua/**. На сторінці «Допомога» клацніть надпис: «Розв`язок мовою C#». На сторінці , що відкрилась скопіюйте до буферу обміну текст під словами: «Приклад програми з використанням консолі для введення/виведення». На сторінці «Список задач» знайдіть «Проста задача?». Перейдіть на вкладку «Відправити розв’язок».*

*Виберіть компілятор «Visual C# 2010». Вставте скопійований текст до вікна «Вихідний код розв'язку». Натисніть кнопку «Відправити».*

**Оператори введення «ReadLine»**

Дуже часто користувач сам має вводити дані до програми під час її виконання. Для цього в мовах програмування використовуються оператори введення. Порядок виконання оператора введення:

1. оператор введення зупиняє програму;
2. надає можливість користувачу ввести дані з клавіатури, введені дані відображаються на екрані;
3. після натискання клавіші «Enter» введені дані переносяться у відповідні області пам’яті;
4. продовжується виконання програми.

Те, що оператор введення зупиняє програму до натискання клавіші «Enter», часто використовують, щоб тимчасово залишити на екрані «чорне» консольне вікно. Для цього в кінці програми пишуть оператор введення без жодних аргументів: System.Console.Read()**;.**

Оператор введення в мові C# потрібно використовувати в поєднанні з оператором присвоєння. Якщо ми хочемо ввести з клавіатури дані в змінну «a», то потрібно писати a = System.Console.Read().

Текст програми мовою C#, що використовує введення даних.

```
class MyClass
{
   static void Main()
   {
      string a = System.Console.ReadLine();
      System.Console.WriteLine(a);
      System.Console.Read();
   }
}
```

Оператором «ReadLine» в мові C# передбачено введення лише символьних (рядкових) значень. Отримати з введених користувачем символів числові чи інші дані можна лише використавши перетворення типів наприклад командою Convert. Convert –це окремий клас зі стандартної бібліотеки класів C#. Клас Convert має набір методів для перетворення рядкових даних у дані інших типів. Так, метод Convert.ToInt32 перетворює рядкові величини в цілочисельні типу Int. Такий метод некоректно працює з рядком, що містить символи, відмінні від

цифр. Метод Convert.Double перетворює рядкові величини в дійсні типу Double. Рядок може містити крім цифр кому та букву «e» для запису даних у стандартному вигляді (формі з плаваючою комою).

Щоб ввести до цілочисельної змінної дані з клавіатури, можна використати оператор введення як аргумент оператора перетворення типу даних:

```
int a = System.Convert.ToInt32(System.Console.ReadLine());
```

Програма обчислення суми двох цілих чисел введених з клавіатури, буде мати такий вигляд:

```
class MyClass
{
   static void Main()
   {
      int a = System.Convert.ToInt32(System.Console.ReadLine());
      int b = System.Convert.ToInt32(System.Console.ReadLine());
      System.Console.WriteLine(a+b);
      System.Console.Read();
   }
}
```

Якщо командою using підключити перед текстом програми простір імен «System», то текст програми стане помітно коротшим:

```
using System;
class MyClass
   {
   static void Main()
   {
   int a = Convert.ToInt32(Console.ReadLine());
   int b = Convert.ToInt32(Console.ReadLine());
   Console.WriteLine(a+b);
   Console.Read();
   }
}
```

**Завдання та запитання:**

*11. Відкрийте програму створену під час виконання попередніх завданнь, яка передбачає виведення на екран маси вашого тіла у вигляді даних різних типів. Знову внесіть зміни до програми, щоб вона просила ввести масу тіла користувача та перетворювала введені символи в дані інших типів.*

12. *На сайті **e-olimp.com/ua/** розв'яжіть задачу № 1000 «Задача A + B»*
13. *На **e-olimp** розв'яжіть № 937 «Добуток цифр трицифрового числа»*

**Класи та об'єкти**

Насправді в C# усі типи даних є класами. Взагалі, в C# всі дані об'єкти, а всі їх різновиди – це класи. Як створити свій клас ми вже знаємо. Але порожній клас нікому не потрібний. У межах одного класу можна використовувати об'єкти іншого класу, якщо це не заборонено певними директивами доступу.

```
using System;
class Animal
{
   public int Weight; // { get; set; }   // вейт
}
class MyFirst
{
 public static void Main(string[] args)
 {
/* int a = 12;
 int i ;
 int [] m = new int[a];
  for (i = 0 ; i < 12 ; i++) m[i]=i;
  for (i = 0 ; i < 12 ; i++) Console.WriteLine(m[i]);

  m = new int[4];
  for (i = 0 ; i < 4 ; i++) m[i]=i;
  for (i = 0 ; i < 4 ; i++) Console.WriteLine(m[i]);*/

   Animal a = new Animal();
   a.Weight = 12;
   Console.WriteLine(a.Weight);
   Console.ReadLine();
 }
```

**Принципи ООП**

Методи є частиною певних класів, ще кажуть *членами* класів. Саме в методах і здійснюються розміщення алгоритму у вигляді програми, що може виконати комп'ютер.

**Методи як різновид підпрограм**

Методи є частиною певних класів, ще кажуть *членами* класів. Саме в методах і здійснюється обробка даних. У програмах, написаних мовою C#, можна використовувати як самостійно розроблені методи тих чи інших класів, так і готові методи, які містять класи бібліотеки .Net Framework. Як уже зазначалось, не використавши жодного метода, неможливо створити навіть найпростішу програму. Виконання усякої програми розпочинається з виконання основного методу ***Main.*** Цей метод ще звуть «точкою входу».

Загалом у методи доцільно об'єднувати ті частини програми, що багаторазово використовуються. Наприклад, для обчислення площі поверхні трикутної піраміди потрібно кілька разів обчислювати площу поверхні трикутника.

Усякий метод, що створюється у програмі, має вказівку на тип, власне ім'я (зазвичай, пишеться з великої літери), параметри та модифікатори доступу. Ось схематичний запис методу:

<модифікатор доступу> <тип методу> <Ім'я методу> (<тип даних першого аргументу> <ім'я аргументу>, <тип даних другого аргументу> <ім'я аргументу>, … <тип даних останнього аргументу> <ім'я аргументу>)

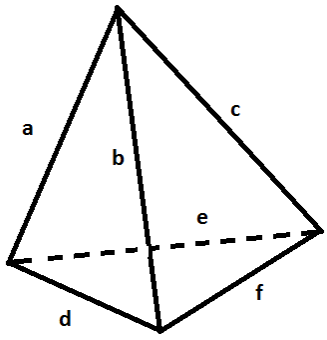
{

<власне код методу>;

/* якщо метод повертає значення то воно вказується після службового слова return*/

return <ім'я величини значення якої видасть метод після виконання> ;

}




Як і у випадку класів та інших даних модифікатори доступу визначають в яких частинах програми метод буде доступним. Параметри – вхідні величини на основі яких і здійснюється виконання методу. Тип методу це, зазвичай, тип даних, що будуть повертатися методом у програму. Якщо метод не повертає жодних значень, то його тип ***void.*** Метод ***Main*** саме такого типу. У методах типу ***void*** службове слово ***return*** не пишуть взагалі.

Розглянемо приклад використання методу у програмі обчислення площі поверхні трикутної піраміди за величинами її граней.

Спочатку напишемо програму, що обчислює площу s трикутника зі сторонами x, y, z за формулою Герона. У цій програмі для обчислення кореня числа ми використовуватимемо метод з простору імен System бібліотеки класів .Net Fremwork – Math.Sqrt().

```
using System;
class MyClass
{
 static void Main()
 {
  double x = Convert.ToDouble(Console.ReadLine());
  double y = Convert.ToDouble(Console.ReadLine());
  double z = Convert.ToDouble(Console.ReadLine());
  double p = (x + y + z) / 2;
  double s = Math.Sqrt(p * (p – x) * (p – y) * (p – z));
  Console.WriteLine(s);
  Console.Read();
 }
}
```

Фрагмент програми, де обчислюється площа трикутника, перетворимо у метод який назвемо Triangle.

```
 static double Triangle(double a, double b, double c)
 {
  double p = (x + y + z) / 2;
  double s = Math.Sqrt(p * (p – x) * (p – y) * (p – z));
  return s;
 }
```

Доповнимо програму кодом, що буде отримувати з клавіатури дані про значення довжин граней піраміди та викликати метод Triangle для кожної її грані.

```
using System;
class MyClass
{
 static void Main()
 {
  double a = Convert.ToDouble(Console.ReadLine());
  double b = Convert.ToDouble(Console.ReadLine());
  double c = Convert.ToDouble(Console.ReadLine());
```

```
    double d = Convert.ToDouble(Console.ReadLine());
    double e = Convert.ToDouble(Console.ReadLine());
    double f = Convert.ToDouble(Console.ReadLine());
    Console.WriteLine(Triangle(a, b, d)+ Triangle(a, c, e)+ Triangle(f, b, c)+
  Triangle(d, e, f));
    Console.Read();
   }
   static double Triangle(double a, double b, double c)
   {
    double p = (a + b + c) / 2;
    double s = Math.Sqrt(p * (p – a) * (p – b) * (p – c));
    return s;
   }
  }
```

Якби нам довелося використовувати цей же метод в іншому класі то ми повинні були б надати йому модифікатор доступу ***public.*** Для виклику методу в іншому класі перед його іменем потрібно вказувати ім’я об’єкту, що містить цей метод.

**Завдання та запитання:**

*14. Напишіть програму обчислення периметра многокутника за координатами його вершин у площині X,Y. (Важливо домовитись, що координати вершин вводяться послідовно. Остання введена координата та перша введена координата також належать одній стороні многокутника). Програма повинна містити метод обчислення довжини відрізка за координатами початку та кінця.*

**Розгалуження**

Лінійний хід виконання програми можна змінити, використавши оператор вибору, який ще називають «розгалуження». Розгалуження складається з трьох елементів:

1. умови;
2. дії (серії команд), що виконується, якщо умова істинна;
3. дії (серії команд), що виконується, якщо умова хибна.

Якщо розгалуження не містить третього елементу, то його називають неповним. Тобто, відмінність повної і неповної форми розгалуження полягає в тому, що неповне розгалуження не містить ніяких дій (команд) для виконання у тому разі, коли умова хибна.

У мові програмування C# розгалуження записується з використанням дещо меншої кількості службових слів: *if (<умова>) <дія за істинної умови> else <дія коли умова хибна >;.*

Спрощена форма розгалуження:

if (<умова>) <дія за істинної умови>;

Проілюструвати використання оператора вибору можна на прикладі виведення більшого з двох чисел.

```
using System;
class MyClass
{
 static void Main()
 {
  int a = Convert.ToInt32(Console.ReadLine());
  int b = Convert.ToInt32(Console.ReadLine());
     if (a > b)
       Console.WriteLine(a);
     else
      Console.WriteLine(b);
  Console.Read();
 }
}
```

Неповну форму оператора вибору можна застосувати, наприклад, у програмі для визначення модуля цілого числа.

```
using System;
class MyClass
   {
    static void Main()
    {
     int a = Convert.ToInt32(Console.ReadLine());
     if (a<0) a = – a;
     Console.WriteLine(a);
     Console.Read();
    }
   }
```

**Деякі нюанси:** Оператор розгалуження дозволяє вибирати один варіант з двох. Досить часто зустрічаються умови, що мають багато більше варіантів.

Наприклад, придбавши залізничний квиток, доводиться вибирати свій вагон з багатьох інших. Звичайно, такий вибір можна реалізувати через багаторазове використання послідовних розгалужень. На щастя, в більшості мов програмування для цього існують зручні засоби. Мова C# має оператор «***swich***», з англійської перекладається як *перемикач.* Структура використання такого оператора наступна

***swich*** (<ім'я величини, що може набувати різних варіантів>)
***{***
***}***

Код у фігурних дужках складається з багатьох однотипних частин, кожна з яких починається словом «***case***» – з англ. *випадок* і закінчується словом ***break***.

***case*** <можливе значення>
***{***
<те, що виконається у даному випадку >
***}***
***break;***

На випадок, якщо після жодного ***case*** не вказано правильний варіант, даний оператор може містити фрагмент ***default*** наступної структури:

***default***
***{***
<те, що виконається у даному випадку >
***}***
***break;***

Ось повна схема вибору:

***swich*** (<ім'я величини, що може набувати різних варіантів>)
***{***
***case*** <перше можливе значення>
***{***
<те, що виконається у даному випадку >
***}***
***break;***
***case*** <ще одне можливе значення>
***{***
<те, що виконається у даному випадку >
***}***
***break;***
***default***

```
        {
            <те, що виконається у даному випадку >
        }
    break;
    }
```

***break*** кожного разу вказує на те, що виконання ***swich*** припиняється і будуть виконуватись наступні команди.

**Завдання та запитання:**

*15. Відкрийте програму створену під час виконання попередніх завдання, яка просить ввести масу тіла користувача. (завдавання 8 ст.6), доповнене завданнями 9 та 11 (ст.9 та ст.11) Внесіть такі зміни до програми щоб вона визначала до якої вагової категорії користувач належить. (вище 81 кг –важка вага, до 81 кг – напівважка вага, до 70 кг – середня вага; до 61 кг – легка вага; до 52 кг – легша вага.)*

*16. На сайті **e-olimp** розв'яжіть задачу № 918 «Яка чверть?»*

*17. На сайті **e-olimp** розв'яжіть задачу № 1154 «Гурток хорового співу»*

*18. На сайті **e-olimp** розв'яжіть задачу № 76 «Нова шафа»*

**Цикли**

Цикл – це послідовність команд, що повторюються скінченну кількість разів. У більшості мов програмування використовуються три основних типи циклів: з передумовою, з післяумовою та цикли з параметром. Мова програмування C# теж дозволяє створювати такі типи циклів. Втім мова C# має особливий тип циклів – «foreach». Цей тип циклів мова C# не унаслідувала від інших мов програмування. Цикл «foreach» виконується для послідовностей певних даних, це зручно, наприклад, для обробки масивів. Цей цикл коротко описано далі, в кінці теми «Сортування масивів»

Цикли всіх типів обов'язково мають «тіло циклу». Тіло циклу – це, власне, і є ті команди, що повторюються. Одне окреме повторення тіла циклу називають ітерацією. Кількість повторень ще звуть кількістю ітерацій циклу. Цикли з після та передумовою, окрім «тіла», містять ще «умову». Умова – це логічна величина. Якщо в циклах з передумовою вона перевіряється до виконання тіла циклу, то в циклах з післяумовою умова перевіряється після кожного виконання тіла циклу. Тобто, цикл з післяумовою передбачає хоча б одне обов'язкове виконання тіла циклу.

У мові програмування C# цикл з передумовою записується: while (<умова>) тіло циклу;.

Для прикладу покажемо як з допомогою циклу з передумовою можна створити програму, що обчислює суму введених з клавіатури чисел доти, доки не введене число «0».

```
using System;
class MyClass
{
 static void Main()
 {
  int a = Convert.ToInt32(Console.ReadLine());
  int s = 0;
     while (a != 0)
     {
    s= s + a;
      a = Convert.ToInt32(Console.ReadLine());
     };
  Console.WriteLine(s);
  Console.Read();
   }
  }
```

У мові програмування C# цикл з післяумовою: do <тіло циклу> while(<умова>);.

За допомогою циклу з післяумовою задачу знаходження суми введених з клавіатури чисел доти, доки не введене число «0» можна вирішити, використавши дещо меншу кількість команд:

```
using System;
class MyClass
{
 static void Main()
 {
     int a; int s = 0;
     do
    {
      a = Convert.ToInt32(Console.ReadLine());
     s= s+a;
      }
```

```
    while (a != 0);
  Console.WriteLine(s);
  Console.Read();     }  }
```

Демонструючи відмінність у використанні циклів з перед- та післяумовою, доцільно вказати на можливість практично повної взаємозаміни цих алгоритмічних конструкцій. Наприклад, попередню програму знаходження суми введених з клавіатури чисел, де використовувався цикл з передумовою, можна не значно змінити так, щоб команда введення не з'являлась в ній двічі. Для цього доцільно надати змінній «а» будь-якого значення відмінного від нуля до початку циклу і в тілі циклу поміняти команди введення та присвоєння місцями:

```
using System;
class MyClass
{
 static void Main()
 {
  int a = -1;
  int s = 0;
  while (a !=0)
     {
     a = Convert.ToInt32(Console.ReadLine());
      s= s + a;
    };
  Console.WriteLine(s);
  Console.Read();
 }
}
```

Цикли ***з параметром,*** окрім «тіла», містять вказівки надання додатковій змінній (параметру) початкового значення та зміни його під час ітерацій. За час виконання циклу з параметром додаткова змінна пробігає певний діапазон значень від свого початкового до кінцевого значення.

У мові програмування C# цикл з параметром характеризується, окрім початкового та кінцевого значень параметру, ще й «кроком». Крок вказує, на яку величину змінюється параметр при кожному виконанні тіла циклу. Поряд із цим у мові C#, на відміну від Pascal, кінцеве значення параметру вказується у вигляді логічної умови. Для того, щоб цикл виконувався, умова повинна бути істинною. Ось так схематично записується цикл з параметром мовою

програмування C#: for (<вираз визначення параметра та його початкового значення>, <умова набуття параметром кінцевого значення (припинення циклу)>, <вираз зміни параметра на величину кроку>) тіло циклу;.

Приклад використання циклу з параметром. Програма знаходить суму п'яти, введених з клавіатури, чисел.

```
using System;
class MyClass
{
  static void Main()
  {
   int s = 0;
   for (int i = 0; i<5; i = i+1 )
      {
         int a = Convert.ToInt32(Console.ReadLine());
          s= s + a;
      };
   Console.WriteLine(s);
   Console.Read();
  }
}
```

**Деякі нюанси:** Досить часто трапляються випадки, коли в певний момент відпадає необхідність продовжувати виконання усього циклу або окремої його ітерації. В мові C# існує можливість розірвати виконання циклу практично будь-якого типу. Цикл припиняється після ключового слова ***break***. Перейти до наступної ітерації в циклі, не завершуючи дану, можна за допомогою ключового слова ***continue***.

**Завдання та запитання:**

*19. Створити програму виведення на екран усіх парних чисел від 1 до вказаного, використовуючи: а – цикл з передумовою, б – цикл з післяумовою, в – цикл з параметром.*

*20. На сайті **e-olimp** розв'яжіть задачу № 916 «Цікавий добуток»*

**Масиви**

Окрім простих типів даних мова програмування C# має цілу низку так званих структурованих типів. У шкільному курсі програмування детально вивчається лише структуровані типи даних «масиви». Їх ще звуть табличними величинами та матрицями. Масив – це скінченна пронумерована сукупність

даних одного типу. Масив складається зі значень, що називаються елементами масиву. Тип даних елементів називають базовим типом даних масиву. Кожен елемент характеризується індексом. Індекс вказує на порядок розміщення елементу в масиві і набуває значень певного перелічуваного типу даних. Для кожного елементу масиву, крім першого і останнього, обов'язково є наступний та попередній. Індекс попереднього елементу масиву на одиницю менший, а індекс наступного елементу на одиницю більший.

Масиви можна оголошувати подібно до оголошення інших типів даних. У мові C# стосовно масивів діють ті ж самі правила видимості, що й для інших величин.

На те, що нова змінна буде масивом, вказують прямокутні дужки після зазначення її базового типу даних. Наприклад, фрагмент коду int[] m – вказує на те, що оголошується цілочисельний масив з іменем m. Специфікація мови C# передбачає два основних способи визначення розміру масиву. Найпростіше вказати розмір масиву, задавши на етапі оголошення усю множину значень його елементів. У такому разі кількість вказаних елементів масиву і визначить його розмір. Наприклад, запис int[] m = {1,0,2,9,3,8,3,7,4,6,5}; вказує на те, що масив m складається з десяти елементів. Схематично оголошення масиву, в поєднанні із зазначенням усіх його елементів, записується так: <тип даних>[] <ім'я масиву> = {<значення першого елементу>,< значення другого елементу >,…,< значення останнього елементу >};. Якщо на етапі оголошення масиву не потрібно вказувати його значення, тобто доцільно залишити масив невизначеним, запис про оголошення масиву набуде наступного схематичного вигляду <тип даних>[] <ім'я масиву > = new <тип даних>[<кількість елементів>];. Наприклад, запис int[] m = new int[10] оголошує масив цілочисельних даних з іменем m, що має розмір 10 елементів. У наведених прикладах запису масиви мови C# індексуються цілими числами починаючи з нуля.

Організувати введення даних до масиву з клавіатури та виведення їх на екран можна використавши цикли. Найчастіше для роботи з масивами використовують цикли з параметром. Параметр циклу вказує на індекс елементу масиву, до якого відбувається звернення.

Приклад введення та виведення даних масиву з використанням циклів:

```
using System;
class MyClass
{
  static void Main()
```

```
  {
   int[] m = new int[10];   int i;
   for (i = 0; i < 10; i=i+1)
       m[i] = Convert.ToInt32(Console.ReadLine());
   for (i = 0; i < 10; i=i+1) Console.WriteLine(m[i]);
   Console.Read();
  }
}
```

**Завдання та запитання:**

*21. Створити програму, що оголошує масив п'яти дійсних чисел та просить ввести до масиву значення середньої добової температури протягом 5-ти днів.*

**Основні алгоритми роботи з масивами**

Розв'язання задач з використанням масивів спирається на деякі основні алгоритми роботи з табличними величинами: введення даних до масиву, виведення значень масиву, визначення суми елементів масиву, знаходження найбільшого (найменшого) елементу масиву, пошуку елементів або їх кількості за певними умовами, сортування елементів масивів.

Для визначення суми елементів масиву потрібно оголосити змінну, що цю суму буде накопичувати.

```
using System;
class MyClass
{
 static void Main()
  {
   int[] m = new int[10];   int i;  int s =0;
   for (i = 0; i < 10; i=i+1)
   m[i] = Convert.ToInt32(Console.ReadLine());
    for (i = 0; i < 10; i=i+1) s=s+m[i];
   Console.WriteLine(s);
   Console.Read();
  }
}
```

Знаходження найбільшого елементу масиву потребує використання розгалуження:

```
using System;
class MyClass
```

```
{
 static void Main()
 {
  int[] m = new int[10];
   int i;
   for (i = 0; i < 10; i=i+1)
      m[i] = Convert.ToInt32(Console.ReadLine());
       int max = m[0];
     for (i = 1; i < 10; i=i+1)
      if (m[i]>max) max = m[i];
    Console.WriteLine(max);
   Console.Read();
  }
}
```

Варто змінити знак в умові розгалуження на протилежний, тобто, замість if (m[i] > max) max = m[i]; написати if (m[i] < max) max = m[i]; як програма почне визначати не максимальний, а мінімальний елемент.

Велику кількість задач можна запропонувати учням на пошук елементів або їх кількості за певними умовами. Наприклад, розглянемо програму, що визначає кількість від'ємних елементів масиву:

```
using System;
class MyClass
{
 static void Main()
 {
  int[] m = new int[10];
   int i;
   for (i = 0; i < 10; i=i+1)
      m[i] = Convert.ToInt32(Console.ReadLine());
    int n = 0;
   for (i = 1; i < 10; i=i+1) if (m[i] <0) n=n+1;
   Console.WriteLine(n);
   Console.Read();
  }
}
```

**Завдання та запитання:**

*22. Відкрити програму, що зберігає значення температури протягом 5-ти днів. Доповнити програму обчисленням та виведенням на екран даних про: а – середнє арифметичне значення температури; б – максимальне значення температури, в – різницю між максимальним та мінімальним значеннями температури; г – кількість днів, протягом яких температура була більше +20-ти градусів.*

*23. На сайті **e-olimp** розв'яжіть задачу № 927 «Кількість іграшок»*

*24. На **e-olimp** розв'яжіть № 928 «Сума найбільшого та найменшого»*

**Сортування масивів**

Однією з класичних задач теорії алгоритмів є сортування (впорядкування) елементів масивів за їх значенням. У шкільному курсі найчастіше вивчаються два загальновідомі алгоритми сортування масивів – метод вибору та метод «бульбашки». Кожен із цих алгоритмів передбачає обмін місцями елементів масиву. Для цього зручно використовувати додаткову змінну базового типу.

Програма сортування масиву методом вибору полягає у визначенні найбільшого елементу масиву та обміну його з першим. Алгоритм повторюється щоразу з меншим числом даних. Елемент, що перед цим був визначений як найбільший, не бере участі в наступних обмінах:

```
using System;
class MyClass
{
  static void Main()
  {
   int[] m = new int[10];  int i, j, imax, r;
   for (i = 0; i < 10; i=i+1)
    m[i] = Convert.ToInt32(Console.ReadLine());
   for (i = 0; i < 9; i=i+1)
       {
       imax = i;
            for (j = i+1; j < 10; j++)
               {
                  if (m[j] > m[imax])  imax=j;
               r = m[i];    m[i] = m[imax];
                 m[imax] = r;
               }
```

```
    }
for (i = 0; i < 10; i=i+1) Console.WriteLine(m[i]);
Console.Read();
 }
}
```

Сортування масиву методом «бульбашки» ще носить назву «обмінного сортування». Під час обмінного сортування відбувається обмін місцями сусідніх елементів масиву. Сортування припиняється тоді, коли не вдається знайти жодних двох сусідніх елементів, що потребують обміну. В такому разі виконується ітерація циклу, в якій не відбувається жодного обміну елементів масиву, що й слугує сигналом для припинення сортування:

```
using System;
class MyClass
{
 static void Main()
 {
  int[] m = new int[10];  int i;
  for (i = 0; i < 10; i=i+1)
  m[i] = Convert.ToInt32(Console.ReadLine());
    int r; bool fleg;
  do
   {
    fleg = false;
    for (i = 0; i < 9; i=i+1)
    if (m[i] > m[i+1])
      {
     r = m[i];   m[i] = m[i+1];
     m[i+1] = r;    fleg = true;
     }
     }
 while (fleg);
  for (i = 0; i < 10; i=i+1) Console.WriteLine(m[i]);
  Console.Read();
 }
}
```

**Деякі нюанси:** В усіх описаних алгоритмах обробки масивів використовуються цикли з параметром for. Хоча ці ж програми можна створити

використовуючи цикли з перед- та післяумовою. Поряд із цим, як зазначалося вище, мова C# має у своєму розпорядженні спеціальний тип циклу для обробки послідовностей певних значень – цикл ***foreach***. Цикл ***foreach*** зручно використовувати для обробки масивів. Структура цього циклу наступна:

***foreach (****<тип даних елементу послідовності>* *<ім'я під яким окремий елемент буде використовуватись в тілі циклу>* *<ім'я послідовності над якою виконуватиметься цикл>****)***

***{***

***<тіло циклу>***

***};***

Недоліком циклу ***foreach*** у мові C# є те, що він працює лише на зчитування даних, а не на їх формування. Тому в алгоритмах обробки масивів неможливо обійтись виключно використанням цього циклу. У програмі сортування масиву з його допомогою доцільно організовувати виведення даних на екран.

```
using System;
class MyClass
{
 static void Main()
 {
  int[] m = new int[10];  int i;
                for (i = 0; i < 10; i=i+1)
  m[i] = Convert.ToInt32(Console.ReadLine());
    int r; bool fleg;
  do
   {
    fleg = false;
    for (i = 0; i < 9; i=i+1)
    if (m[i] > m[i+1])
      {
     r = m[i];   m[i] = m[i+1];
     m[i+1] = r;    fleg = true;
     }
     }
      while (fleg);
 foreach (int j in m) Console.WriteLine(j);
  Console.Read();
}
```

}

**Завдання та запитання:**

*25. Зобразити в зошиті блок-схеми обох програм сортування масиву.*

*26. Спираючись на відповідну блок-схему, створити програму сортування масиву одним зі способів.*

## Масиви як об'єкти класу *Array*. Використання довідки про бібліотеки класів .NET Framework

Всі величини мови C# є об'єктами певних класів. Масиви – це об'єкти класу ***array***. Створюючи програми обробки масивів від пошуку максимального елементу до сортування, ми створювали методи подібні до яких уже існують в класі ***array*** програмної платформи ***.NET Framework*** (на цій платформі базується мова C#). Розглянемо як можна скористатись методами класу ***Array*** з цієї бібліотеки та аналогічно можливостями інших класів C#, а отже, і ***.Net***.

Для того, щоб використати методи, потрібно знати про їх існування та про синтаксис звернення до них. Більшість середовищ програмування містять довідкову систему (зазвичай викликається клавішею ***F1***). Для пошуку в довідковій системі найчастіше використовують структурований зміст або ж пошук за ключовими словами у спеціальному рядку пошуку.

Ще простіше скористатися контекстною допомогою довідкової системи. Якщо розмістити курсор на відповідне службове слово C#, то натисканням певної комбінації клавіш (найчастіше це *Ctrl+F1*) можна відразу перейти до розділу допомоги пов'язаного з даним словом. Контекстна допомога по надрукованому у програмі слові «*Array*» містить дані про методи цього класу.

Якщо середовище програмування не включає в себе довідкової служби, або ж якщо довідкова служба з якихось причин не функціонує, існує досить багато ресурсів Internet, де можна отримати потрібні відомості. Ось деякі з них: http://msdn.microsoft.com/ru-ru/library/d11h6832(v=vs.90).aspx довідник по усій бібліотеці класів .NET Framework; http://msdn.microsoft.com/ru-ru/library/system.array_members(v=vs.90).aspx сторінка присвячена членам класу ***Array.***

Для прикладу продемонструємо використання готового методу обробки масивів.

Використанням методу ***sort*** класу ***array*** значно зменшує обсяг програми сортування масиву:

```
using System;
class MyClass
```

```
        {
         static void Main()
         {
          int[] m = new int[10];  int i;
          for (i = 0; i < 10; i=i+1) m[i] = Convert.ToInt32(Console.ReadLine());
             Array.Sort(m);
          for (i = 0; i < 10; i=i+1) Console.WriteLine(m[i]);
         Console.Read();
        }
}
```

**Завдання та запитання:**

*27. Створити програму, що з використанням методу **CopyTo** копіює значення одного масиву в інший. Відомості про цей метод почерпнути в довідковій системі середовища програмування чи на сайті http://msdn.microsoft.com/ru-ru/library/system.array.copyto(v=vs.90).aspx*

**Деякі використані спрощення**

У наведених вище прикладах використання мови C# свідомо уникнуто використання оператора інкременту. Наприклад, у задачах обробки масивів замість безпосереднього інкременту змінної «i» використовується вираз, що реалізовує інкремент через присвоєння та додавання – «i <присвоїти> i+1».

| Операція | Вираз |
|---|---|
| Інкремент | i++ |
| Заміна присвоєнням | i = i+1 |

Інкремент (з англійської increment збільшення) – операція збільшення аргументу на деяку фіксовану величину або ж, у деяких випадках, на змінну. Зворотну операцію називають декремент (зменшення). Збільшення певної величини на одиницю надзвичайно широко використовується при написанні комп'ютерних програм. Практично в усіх мовах програмування використання спеціальних функцій інкременту прискорює виконання алгоритму, а отже: є одним із засобів його оптимізації. Поряд із цим, саме позначення операції інкременту по впливало на появу назви мови C#. Одна з попередниць C# – мова програмування C++ отримала свою назву від позначення цієї операції. Значок # – став образним втіленням чотирьох плюсів. Такий цікавий історичний факт можна повідомити учням, щоб пробудити інтерес до предмету.

Історично використання комп'ютера пов'язано з виконанням значних математичних обчислень. Проте, у мові C# для виконання обчислень з використанням тригонометричних функцій, логарифмів та інших математичних засобів виникає необхідність використання додаткового класу – бібліотеку зі звичайними математичними функціями, відому як бібліотека «Math». Розглянемо програму, що за відомими катетами обчислює гіпотенузу прямокутного трикутника.

```
using System;
class Example
{
 static void Main()
 {
 Console.WriteLine(«довжину першого катета 'a'»);
 double a = Convert.ToDouble(Console.ReadLine());
 Console.WriteLine(«довжину другого катета 'b'»);
 double b = Convert.ToDouble(Console.ReadLine());
 double c = (Math.Sqrt (a * a + b * b ));
 Console.Write(«довжина гіпотенузи складає»);
 Console.Write(c);
 Console.Read();
 }
}
```

**Завдання та запитання:**

*28. Відкрити програму сортування масиву. Замінити у програмі, скрізь де це допустимо, вирази зміни значення даних на операцію інкременту.*

*29. Створити програму розв'язання квадратного рівняння з використанням розгалуження або на сайті **e-olimp** розв'яжіть задачу № 932 «Висота трикутника».*

*30. На сайті **e-olimp** розв'яжіть задачу № 926 (Формула Герона)*

*31. На e-olimp розв'яжіть задачу № 948 (Площа та об`єм піраміди)*

## Додаток Е. Тести для перевірки знань учнів під час навчання програмування в класах технологічного профілю ЗНЗ на основі мови С#

### Тест 1. Класи та об'єкти (С#) файл: class_objekt.tst

1. *Як зберегти програму в інший файл (під іншим іменем)?*

Натиснути Alt + F4
Натиснути Ctrl+s
Меню «Debug» пункт «Run»
Меню «Fail» пункт «Save»
**Меню** «Fail» пункт «Save As»

2. *Як зберігають програми написані мовою С#?*

**В файли** розширенням *.cs
В файли розширенням *.csharp
В файли розширенням *.c#
В Інтернеті на спеціальних сторінках
Їх ніхто ніколи не зберігає

3. *Як зберегти програму перед виконанням?*

Натиснути Ctrl+F5
Натиснути на «зелений трикутничок»
**Натиснути** Ctrl+S
Попередньо слід зберегти програму
Натиснути F8

4. *Як перевірити роботу програми?*

**Попередньо** слід зберегти програму
**Далі** запустити на виконання
Слід показати сусіду по парті
Слід покликати вчителя
Натиснути F12

5. *Як копіювати текст до буферу обміну?*

Стати на початок тексту утримуючи Shift натискати стрілку вправо ->
Стати в кінець тексту утримуючи Shift натискати стрілку вліво <-
**Спочатку** текст треба виділити.
Натиснути Control(Ctrl) та не відпускаючи натиснути Delete(Del),(Ctrl+Del)
Натиснути Shift та не відпускаючи натиснути Insert, (Shift+Ins)

6. *Як виділити текст для копіювання до буферу обміну?*

**Стати** на початок тексту утримуючи Shift і натискати стрілку вправо ->
Стати в кінець тексту утримуючи Shift і натискати стрілку вліво <-

Натиснути Control(Ctrl) та не відпускаючи натиснути Insert(Ins),(Ctrl+Ins)
Натиснути Control(Ctrl) та не відпускаючи натиснути Delete(Del),(Ctrl+Del)
Натиснути Shift та не відпускаючи натиснути Insert, (Shift+Ins)

*7. Як копіювати виділений текст до буферу обміну?*
Стати на початок тексту утримуючи Shift натискати стрілку вправо ->
Стати в кінець тексту утримуючи Shift натискати стрілку вліво <-
**Натиснути** Control(Ctrl) та не відпускаючи натиснути Insert(Ins),(Ctrl+Ins)
Натиснути Control(Ctrl) та не відпускаючи натиснути Delete(Del),(Ctrl+Del)
Натиснути Shift та не відпускаючи натиснути Insert, (Shift+Ins)

*8. Як вставити туди де розміщено курсор вміст буферу обміну?*
Стати на початок тексту утримуючи Shift натискати стрілку вправо ->
Стати в кінець тексту утримуючи Shift натискати стрілку вліво <-
Натиснути Control(Ctrl) та не відпускаючи натиснути Insert(Ins),(Ctrl+Ins)
Натиснути Control(Ctrl) та не відпускаючи натиснути Delete(Del),(Ctrl+Del)
**Натиснути** Shift та не відпускаючи натиснути Insert, (Shift+Ins)

*9. Як перенести (вирізати) виділений текст до буферу обміну?*
Стати на початок тексту утримуючи Shift натискати стрілку вправо ->
Стати в кінець тексту утримуючи Shift натискати стрілку вліво <-
Натиснути Control(Ctrl) та не відпускаючи натиснути Insert(Ins),(Ctrl+Ins)
**Натиснути** Control(Ctrl) та не відпускаючи натиснути Delete(Del),(Ctrl+Del)
Натиснути Shift та не відпускаючи натиснути Insert, (Shift+Ins)

*10.Що повинна містити найпростіша програма?*
**{**
**}**.
**метод** static void Main()
Main(class)
Write class

*11.Без яких записів (слів, знаків) у перших рядках програми на C# не можна обійтись?*
**{**
}
**pablic static void Main()**
**class**
Write

*12.Як у програмі на C# можна створювати коментарі?*
**/*- на початку коментаря** */- в кінці
}.
**від** // і до кінця рядка

class
Write

*13.Які рядки може містити програма в C# але без них вона буде працювати так самісінько?*

**Коментарі**: /*- на початку коментаря */- в кінці
}.
**Коментарі**: від // і до кінця рядка
class
Write

*14.Як створити обов'язковий метод Main на мові C#?*

в **оголошеному** класі написати static void Main()
**static** void Main() {інші поля та методи програми}
class {<поля та методи класу>} MyFirstClass
MyFirstClass {<поля та методи класу>}
static void MyFirstClass {<поля та методи класу>}

*15.Як створити клас з іменем MyFirstClass на мові C#?*

**class** MyFirstClass{}
MyFirstClass clas{}
class {} MyFirstClass
MyFirstClass {}
static void MyFirstClass {}

*16.Як створити клас з іменем MyFirstClass на мові C#?*

**class** MyFirstClass{<поля та методи класу>}
MyFirstClass clas{<поля та методи класу>}
class {<поля та методи класу>} MyFirstClass
MyFirstClass {<поля та методи класу>}
static void MyFirstClass {<поля та методи класу>}

*17.Як створити найпростішу програму на мові C#?*

Слід створити клас з довільним іменем і без жодних методів
Слід оголосити метод «Main» без створення жодних класів
Слід зберегти будь яке повідомлення в файл *.cs
Слід оголосити метод «Main» в якому створити клас
**Слід** створити клас в якому оголосити метод «Main»

*18.Об'єкти в програмуванні це?*

**Елементи** класу, що утворюються на час виконання програми
Дані без алгоритмів їх обробки
Алгоритми обробки даних без самих даних
Породжують нові класи під час роботи програми

**З'являються** на основі класів під час роботи програми

*19.Класи в програмуванні це?*
**Алгоритмами** в поєднанні з можливими даними
Алгоритми без даних
Об'єднання даних в одне ціле
Об'єднання програм в одне ціле
**Сукупність** можливих чи існуючих об'єктів

*20.Об'єкти в програмуванні це?*
**Дані** в сукупності з алгоритмами
Дані відокремлені від алгоритмів
Об'єднання даних в одне ціле
Об'єднання програм в одне ціле
**Об'єднання** даних та програм в одне ціле

*21.Де тут вказано на проблеми розмежування доступу до даних при виконанні сучасних комп'ютерних програм?*

**одночасне** виконання різних алгоритмів (програм)
висока ціна (вартість) комп'ютерів
повільне мислення людини, в порівнянні з комп'ютером
неможливість одночасного виконання різних алгоритмів (програм)
**одночасний** доступ різних програм до одних і тих же даних

*22.Яка особливість сучасних комп'ютерних програм з огляду на можливості їх одночасної роботи?*
**Вони** можуть виконуватися незалежно одна від одної.
**Вони** можуть завершуватися незалежно одна від одної.
Вони не можуть виконуватися незалежно одна від одної.
Програми не можуть функціонувати одночасно.
Не можна запустити одну програму не завершивши іншу.

*23.Як будують нові, складніші програми (алгоритми)?*
**вони** складаються з простіших
вони не можуть складатися з простіших
шляхом вилучення старих фрагментів
тільки не об'єднанням простіших алгоритмів
**об'єднанням** існуючих програм в єдину

*24.Що таке комп'ютерна програма?*
**алгоритм** виконавцем якого є комп'ютер
алгоритм виконавцем якого є людина
Послідовність дій, яка призводить до несподіваного результату.

Будь-яка послідовність дій.
алгоритм виконавцем якого є тварина

*25. Що таке алгоритм?*
**Послідовність** дій, яка обов'язково призводить до потрібного результату.
Послідовність дій, яка не обов'язково призводить до потрібного результату.
Послідовність дій, яка призводить до несподіваного результату.
Будь-яка послідовність дій.
Результати роботи користувача
Натиснути Alt + F4 і змінити ім'я файлу

## Тест 2. Перші кроки в Antechinus C# Editor файл: BeginC#Editor.tst

*1. Як зберегти програму в інший файл (під іншим іменем)?*
Натиснути Alt + F4 і змінити ім'я файлу
Натиснути Ctrl+s і не змінювати ім'я файлу
меню «Build» пункт «Run» і не змінювати ім'я файлу
Меню «File» пункт «Save» і змінити ім'я файлу
** **Меню** «File» пункт «Save As» і змінити ім'я файлу*

*2. Як зберегти зміни в програмі (в тому ж файлі)?*
Натиснути на панелі інструментів «прапорець»
Натиснути на панелі інструментів «білий аркуш»
Натиснути на панелі «стрілочку над листочками»
Натиснути на панелі інструментів «синенький знак оклику»
* **Натиснути** на панелі інструментів «дискету»

*3. Як зберегти зміни в програмі (в тому ж файлі)?*
Натиснути Alt + F4 і не змінювати ім'я файлу
Натиснути Ctrl+Весь
меню «Build» пункт «Run» і не змінювати ім'я файлу
Натиснути Alt + S
* **Меню** «File» пункт «Save As» і не змінювати ім'я файлу

*4. Як зберегти зміни в програмі (в тому ж файлі)?*
Натиснути Alt + F4 і не змінювати ім'я файлу
* **Натиснути** Ctrl+s
меню «Build» пункт «Run» і не змінювати ім'я файлу
* **Меню** «File» пункт «Save»
Меню «File» пункт «Save As» і змінити ім'я файлу

*5. Навіщо часто зберігати зміни в програмі?*
Antechinus C# Editor дуже рідко самозакривається

* **Antechinus C# Editor дуже часто самозакривається**
У випадку збою C# Editor відновлює не збережені зміни
* **Щоб** не зник текст програми
Щоб зник текст програми

*6. Навіщо часто зберігати зміни в програмі?*
Так вимагає вчитель (покарає поганою оцінкою)
* У **випадку** збою не збережені зміни назавжди зникнуть
У випадку збою всякий файл завжди можна відновити
* **Щоб** не доводилося знову набирати текст програми
Щоб доводилося знову набирати текст програми

*7. Що відбувається коли програму запускаємо на виконання?*
програма збережеться в оперативній пам'яті
програма перетвориться в машинні коди і в такому вигляді збережеться
машинні коди перетворяться в текст програми і збережуться
в папці, де збережено текст програми, утвориться виконуваний файл
* **виконуваний** файл буде запущено в дію

*8. Як запустити програму на виконання?*
Натиснути Alt + F4
**Натиснути** Ctrl + F5
Меню «File» пункт «Save»
**меню** «Build» пункт «Run»
Натиснути F8

*9. Як запустити програму на виконання? (за допомогою миші)*
Натиснути на панелі інструментів значок «прапорець»
Натиснути на панелі інструментів значок «білий аркуш»
Натиснути на панелі інструментів значок «стрілочку над листочками»
**Натиснути** на панелі інструментів значок «синенький знак оклику»
Натиснути на панелі інструментів значок «дискету»

*10.Що станеться, якщо програму відкомпілювати?*
машинні коди перетворяться в текст програми
**програма** стане придатною для багаторазового виконання
утвориться файл, який не можна виконувати самостійно
**утвориться** файл, який можна виконувати самостійно
Виконуваний файл буде запущено на виконання

*11.Що станеться якщо програму відкомпілювати?*
програма збережеться в оперативній пам'яті
* **програма** перетвориться в машинні коди і в такому вигляді збережеться
машинні коди перетвориться в текст програми і збережуться

* в **папці** де збережено текст програми утвориться виконуваний файл
Виконуваний файл буде запущено в дію

*12.Як відкомпілювати програму? (з допомогою миші)*
Натиснути на панелі інструментів значок «прапорець»
Натиснути на панелі інструментів значок «білий аркуш»
* **Натиснути** на панелі інструментів значок «стрілочку над листочками»
Натиснути на панелі інструментів значок «синенький знак оклику»
Натиснути на панелі інструментів значок «дискету»

*13.Як відкомпілювати програму?*
Натиснути Alt + F8
Натиснути Ctrl + F5
* **Меню** «Build» пункт «Compile»
меню «Build» пункт «Run»
Натиснути F7

*14.Як відкомпілювати програму?*
Натиснути Alt + F4
Натиснути Ctrl + F5
Меню «File» пункт «Save»
меню «Build» пункт «Run»
* **Натиснути** F8

*15.Що повинна містити найпростіша програма?*
* **оголошення** класу та метод Main в ньому
* **class**<якесь ім'я>{static void Main(){ }}
Console.Write();
коментарі
оголошення змінних

*16.Які роздільники команд можуть використовуватися в C#?*
* **Крапка** з комою – ;
Крапка з комою – .,
Крапка
* **проміжок**
* **початок** нового рядка

*17.Без якого з елементів програми не можна обійтись навіть в будь-якій програмі на C# ?*
* **функції** (методу) Main
Console.Read();
* **static** void Main()
пари лапок « »

оголошення змінних

*18.Як у програмі на C# можна створювати коментарі?*
* /*- на **початку** коментаря */- в кінці
{коментар}
* **від** // і до кінця рядка
class <коментар>
Write(«коментар»);

*19.Які рядки може містити програма в C# але без них вона буде працювати так самісінько?*

* /*- на **початку** коментаря */- в кінці
{
* від // і до **кінця** рядка
class
Console.ReadLine()

*20.Які рядки часто зустрічаються програмах C# але без них вони будуть працювати так само?*
* **Коментарі**: /*- на початку коментаря */- в кінці
}
* **Коментарі**: від // і до кінця рядка
class (після якого йде назва класу)
Write – для виведення повідомлення на екран

*21.Як перейти на інше вікно редактора C# Editor?(з допомогою миші)*
Клацнути на меню Fail вибери пункт Open
Клацнути на меню Fail вибери пункт New
Клацнути на меню Fail вибери пункт Exit
Клацнути на значку іншого вікна панелі задач
* **Клацнути** на значку(вкладку) іншого вікна під редактором тексту

*22.Як перейти на інше вікно редактора C# Editor? (з допомогою клавіатури)*
Натиснути Alt + F4
в меню Fail вибери пункт New
в меню Fail вибери пункт Open
Натиснути Alt + Tab
* **Натиснути** Ctrl + Tab

*23.Як перейти на інше вікно в операційній системі? (з допомогою миші)*
Клацнути на меню Fail вибери пункт Open
Клацнути на меню Fail вибери пункт New
Клацнути на меню Fail вибери пункт Exit

* **Клацнути** на значку іншого вікна панелі задач
Клацнути на значку іншого вікна під редактором тексту

*24.Як перейти на інше вікно в операційній системі? (з допомогою клавіатури)*
Натиснути Alt + F4
в меню Fail вибери пункт New
в меню Fail вибери пункт Open
* **Натиснути** Alt + Tab
Натиснути Ctrl + Tab

*25.Як закрити Antechinus C# Editor?*
Натиснути Alt + BakeSpase
На робочому столі знайти його ярлик
В меню «Пуск» у списку «всіх програм» знайти Antechinus C# Editor
* **Меню** «Fail» пункт «Exit»
* **Натиснути** Alt + F4

*26.Як запустити Antechinus C# Editor?*
Натиснути Alt + F4
* **Знайти** ярлик Antechinus C# Editor та двічі клацнути мишею на ньому.
Зайти на диск «C» у папку «BP» та підпапку «Bin» ввести «Turbo»
Натиснути F3
* В **меню** «Пуск» серед списку «всіх програм» знайти Antechinus C# Editor

## Тест 3. Присвоення в C#

*1. Де вірно поєднано оголошення рядкової величини та присвоєння їй значення введеного з клавіатури?*
**string Name1** = **System**.Console.ReadLine();
int Name1 = System.Convert.ToInt32(System.Console.ReadLine());
int Name1 = System.Convert.ToInt32;
int Name1 = System.Console.ReadLine();
string Name1 = System.Convert.ToInt32(System.Console.ReadLine());

*2. Де вірно поєднано оголошення цілочисельної величини та присвоєння їй значення введеного з клавіатури?*
string Name1 = System.Console.ReadLine();
int **Name1** = System.Convert.ToInt32(System.Console.ReadLine());
int Name1 = System.Convert.ToInt32;
int Name1 = System.Console.ReadLine();
string Name1 = System.Convert.ToInt32(System.Console.ReadLine());

*3. Де вірно поєднано оголошення рядкової величини та присвоєння їй значення введеного з клавіатури?*

string b = **System**.Console.ReadLine();
int b = System.Convert.ToInt32(System.Console.ReadLine());
int b = System.Convert.ToInt32;
int b = System.Console.ReadLine();
string b = System.Convert.ToInt32(System.Console.ReadLine());

*4. Де вірно поєднано оголошення цілочисельної величини та присвоєння їй значення введеного з клавіатури?*

string b = System.Console.ReadLine();
int b = **System**.Convert.ToInt32(System.Console.ReadLine());
int b = System.Convert.ToInt32;
int b = System.Console.ReadLine();
string b = System.Convert.ToInt32(System.Console.ReadLine());

*5. Де вірно поєднано оголошення величини дійсного типу та присвоєння їй значення введеного з клавіатури?*

string b = System.Convert.ToDouble(System.Console.ReadLine());
**double** b = System.Convert.ToDouble(System.Console.ReadLine());
int b = System.Convert.ToDouble(System.Console.ReadLine());
int b = System.Console.ReadLine();
double b = System.Convert.ToInt32(System.Console.ReadLine());

*6. Яке призначення оператора виведення? System.Console.WriteLine (величина яку слід вивести на екран)*

**Вивести** повідомлення на екран
Зупинити програму для введення даних
Присвоїти змінній її значення
Відвести в пам'яті комп'ютера поіменоване місце для збереження даних
Продовжити виконання в одному з напрямків у залежності від умови

*7. Яке призначення оператора введення? Величина(яка набуде введеного значення)= System.Console.ReadLine()*

Вивести повідомлення на екран
Зупинити програму для введення даних
Присвоїти змінній її значення
Відвести в пам'яті комп'ютера поіменоване місце для збереження даних
**Продовжити** виконання програми після натискання «Enter»

*8. Яке призначення команди? <Ім'я змінної> =<значення змінної>;*

Вивести повідомлення на екран
Зупинити програму для введення даних
**Присвоїти** змінній її значення

Відвести в пам'яті комп'ютера поіменоване місце для збереження даних
Продовжити виконання в одному з напрямків у залежності від умови

*9. Яке призначення команди? <тип даних> <ім'я змінної>;*
Вивести повідомлення на екран
Зупинити програму для введення даних
Присвоїти змінній її значення
**Відвести** в пам'яті комп'ютера поіменоване місце для збереження даних
Продовжити виконання в одному з напрямків у залежності від умови

*10.Яке призначення присвоєння?*
Вивести повідомлення на екран
Зупинити програму для введення даних
**Присвоїти** змінній її значення
Відвести в пам'яті комп'ютера поіменоване місце для збереження даних
Продовжити виконання в одному з напрямків у залежності від умови

*11.Яке призначення оголошення змінних?*
Вивести повідомлення на екран
Зупинити програму для введення даних
Присвоїти змінній її значення
**Відвести** в пам'яті комп'ютера поіменоване місце для збереження даних
Продовжити виконання в одному з напрямків у залежності від умови

*12.Яке призначення команди? System.Console.WriteLine(величина яку слід вивести на екран);*
**Вивести** повідомлення на екран
Зупинити програму для введення даних
Присвоїти змінній її значення
Відвести в пам'яті комп'ютера поіменоване місце для збереження даних
Продовжити виконання в одному з напрямків у залежності від умови

*13.Яке призначення команди? System.Console.ReadLine();*
Вивести повідомлення на екран
**Зупинити** програму для введення даних
Присвоїти змінній її значення
Відвести в пам'яті комп'ютера поіменоване місце для збереження даних
Продовжити виконання в одному з напрямків у залежності від умови

*14.Яке призначення команди? <Ім'я змінної> =<значення змінної>;*
Вивести повідомлення на екран
Зупинити програму для введення даних
**Присвоїти** змінній її значення
Відвести в пам'яті комп'ютера поіменоване місце для збереження даних

Продовжити виконання в одному з напрямків у залежності від умови

*15.Яке призначення вказівки? <тип даних> <ім'я змінної>;*
Вивести повідомлення на екран
Зупинити програму для введення даних
Присвоїти змінній її значення
**Відвести** в пам'яті комп'ютера поіменоване місце для збереження даних
Продовжити виконання в одному з напрямків у залежності від умови

*16.Яке призначення команди? System.Console.WriteLine(a);*
**Вивести** повідомлення на екран
Зупинити програму для введення даних
Присвоїти змінній її значення
Відвести в пам'яті комп'ютера поіменоване місце для збереження даних
Продовжити виконання в одному з напрямків у залежності від умови

*17.Яке призначення команди? b = System.Console.ReadLine();*
Вивести повідомлення на екран
**Зупинити** програму для введення даних
**Присвоїти** змінній її значення
Відвести в пам'яті комп'ютера поіменоване місце для збереження даних
Продовжити виконання в одному з напрямків у залежності від умови

*18.Яке призначення команди? a = 10;*
Вивести повідомлення на екран
Зупинити програму для введення даних
**Присвоїти** змінній її значення
Відвести в пам'яті комп'ютера поіменоване місце для збереження даних
Продовжити виконання в одному з напрямків у залежності від умови

*19.Яке призначення вказівки? int a;*
Вивести повідомлення на екран
Зупинити програму для введення даних
Присвоїти змінній її значення
**Відвести** в пам'яті комп'ютера поіменоване місце для збереження даних
Продовжити виконання в одному з напрямків у залежності від умови

*20.Яке призначення команди? System.Console.WriteLine(a)*
Вивести повідомлення на екран про призначення програми
**Вивести** на екран значення змінної (a)
Зупинити програму для введення значення (a)
Відвести в пам'яті комп'ютера місце під іменем «a»
Зупинити програму для введення значення (b)

*21.Яке призначення команди? b =System.Console.ReadLine();*

Вивести повідомлення на екран про призначення програми
Вивести на екран значення змінної (a)
Зупинити програму для введення значення (a)
Відвести в пам'яті комп'ютера місце під іменем «a»
**Зупинити** програму для введення значення (b)

*22.Яке призначення команди? a == 10;*

**перевірити** чи справді a та 10 рівні
Вивести на екран значення змінної (a)
Зупинити програму для введення значення (a)
Відвести в пам'яті комп'ютера місце під іменем «a»
Зупинити програму для введення значення (a)

*23.Яке призначення вказівки? int a;*

Вивести повідомлення на екран про призначення програми
Вивести на екран значення змінної (a)
Зупинити програму для введення значення (a)
**Відвести** в пам'яті комп'ютера місце під іменем «a»
Зупинити програму для введення значення (b)

*24.Яка помилка у фрагменті програми?*

```
class MyFirstClass{
int a.b;
}
```

Ім'я класу або методу містить проміжки (пробіли)
В кінці команди немає крапки з комою
**Оголошувані** змінні відокремлені недопустимим знаком
Після { крапка з комою викликає помилку
Використовується невідповідний тип даних.

*25.Яка помилка у фрагменті програми?*

```
class My FirstClascc{
int a,b;
}
```

**Ім'я** класу або методу містить проміжки (пробіли)
В кінці команди немає крапки з комою
Оголошувані змінні відокремлені недопустимим знаком
Після слова {крапка з комою викликає помилку
Використовується невідповідний тип даних.

*26.Яка помилка в фрагменті програми?*

```
{ int a,b; b=12;
```

*a=b/(2.12);*
*System.Console.WriteLine(a)}*

Ім'я класу або методу містить проміжки (пробіли)

В кінці команди немає крапки з комою

Оголошувані змінні відокремлені недопустимим знаком

Фрагмент не містить помилок

**Використовується** невідповідний тип даних.

*27.Яка помилка в фрагменті програми?*

*class NameClass{*
*static void Main() {*
*int a=System.Console.ReadLine(); } }*

Ім'я класу або методу містить проміжки (пробіли)

В кінці команди немає крапки з комою

Оголошувані змінні відокремлені недопустимим знаком

Компілятор C# самостійно виправляє помилки в програмі

**Використовується** невідповідний тип даних.

## Додаток Є. Зразки планів-конспектів уроків з розділу «Основи алгоритмізації та програмування» з інформатики в 11 класі за програмою академічного рівня

### Урок 1. Тема: Об'єкти оточуючого світу та комп'ютерних програм. Об'єктний підхід.

***Очікувані результати***

***Після цього заняття учні зможуть:***

- *Пояснювати, що таке об'єкти в оточуючому світі та в комп'ютерних програмах;*
- *Пояснювати, що таке наслідування, інкапсуляція, поліморфізм.*
- *Визначати та розрізняти окремі об'єкти живої та неживої природи;*
- *Групувати об'єкти в певні класи за спільними властивостями;*
- *Дати визначення понять: клас, наслідування, інкапсуляція, поліморфізм;*
- *Оцінювати власне уміння розрізняти об'єкти та їх класи в уже існуючих комп'ютерних програмах.*

***Основні питання теми:***

- Об'єкт у програмуванні.
- Класи об'єктів.
- Наслідування, інкапсуляція.
- Поліморфізм.
- Властивості об'єкта.
- Поняття про методи об'єкта.
- Елементи інтерфейсу користувача як об'єкти.

***Обладнання:***

1. Картки з різними, спеціально підібраними словами згідно додатку 1.
2. Картки з істинними та хибними твердженнями або їх номерами згідно списку поданого в додатку 2.
3. Тексти для самостійної та групової роботи в додатку 3 та додатку 5.
4. Шаблон ієрархії класів у додатку 4
5. Текст для домашньої роботи в додатку 6.

***Регламент уроку***

1. Вступна частина – 6 хв. (виклик, групова робота з аналізу)
2. Основна частина – 32 хв.

   Дія 1 (робота з оцінювання в основі якої проводиться пошук критеріїв поняття об'єкт) до 10 хв.

   Дія 2 (робота з текстом «Об'єкти та класи» на основі графічного організатора) до 10 хв.

   Дія 3 (синтез) резерв на випадок швидкої роботи учнів.

   Дія 4 (робота з текстом «Властивості класів» на основі методу опорних слів) до 12 хв.
3. Заключна частина – 7 хв. (рефлексія у процесі підбору синонімів)

***Організація пізнавальної діяльності учнів***

1. *Вступна частина*.

**Метод роботи на даному етапі**: групова робота з аналізу. (Ця робота має зайняти не більш ніж 6 хв.)

**Основне завдання етапу**: Обговорити поняття «об'єкт», визначити загальні критерії цього поняття.

**Організація роботи:** Об'єднуємо клас у чотири групи. Першим трьом групам роздаємо картки з окремими словами та словосполученнями (додаток 1). Доручаємо групам такі завдання:

1 група. Виберіть картки, де названо об'єкти, що є в нашому класі.

2 група. Виберіть картки ,де названо об'єкти, що є в нашій державі.

3 група. Відкиньте картки, де вказано слова, що позначають об'єкти.

4 група. Зробіть власний список об'єктів, що зустрічаються в редакторі Word.

Через 2 хв. вчитель просить кожну групу зачитати по 5 тих слів, при визначенні приналежності яких виникли трудності. Наступна група коментує зроблений вибір. Зазвичай, виникає суперечка. Причина суперечки – проблема визначення, що ж таке об'єкт.

**Підсумок вступної частини**

По завершенні роботи вчитель пропонує обговорити: *Що називають об'єктом взагалі? Що таке об'єкт у програмуванні?*

Потім вчитель представляє тему уроку та його очікувані результати: «Ми розглянемо сьогодні, що ж таке об'єкти у комп'ютерних програмах та у процесі їх розробки. Ви дізнаєтесь про властивості об'єктів, їх класифікацію,

особливості створення. Тема уроку: «Об’єкти оточуючого світу та комп’ютерних програм»

2. *Основна частина*

**Дія 1**

**Метод роботи на даному етапі**: Робота з оцінювання (пошук критеріїв поняття об’єкт та застосування цих критеріїв). **(**Ця робота має зайняти до 10 хв.)

**Основне завдання етапу**: розглянути основні поняття об’єктно-орієнтованого програмування: наслідування, поліморфізм, інкапсуляцію та поняття: поле класу, метод класу, подія.

**Організація роботи:** Заздалегідь записується на дошці чи іншим чином демонструється класу список тверджень, що стосуються об’єктів. У додатку 2 наведено приклад 20-ти таких висловлювань. Картки з цими твердженнями або їх номерами вчитель пропонує витягнути навмання кожному з учнів класу (виготовляється така кількість карток, щоб вистачило усім учням). Далі кожен учень розмірковує над тим, чи є вказане карткою твердження істинним.

Щоб з’ясувати, що ж можна вважати об’єктом а що ні, вчитель пропонує класу тимчасово об’єднатися в три групи:

1 група. Ті кому дісталося істинне твердження.

2 група. Ті кому дісталося хибне твердження.

3 група. Ті хто сумнівається, хибне твердження чи істинне.

Спочатку розглядаються твердження **3-ї** групи. Учні групи відправляються, після «з’ясування істинності», до групи 1 та 2 відповідно. Далі аналізуються вислови на картках учнів другої групи та першої. Якщо хибність твердження підтверджується, його з дошки витирають (вилучають). Отже, на дошці повинні залишитися лише істинні твердження. Корисно звернути увагу на твердження: «об’єкти існують незалежно від того чи знає про них хтось». Можна внести деякі уточнення: «об’єкти існують незалежно від усвідомлення людьми, людської свідомості». Після обговорення інших тверджень на дошці може залишитися таке:

1. Об’єкти завжди можна чітко відмежувати один від одного.
2. Об’єкти існують незалежно від ~~того чи знає про них хтось~~ людської свідомості.
3. Об’єкти мають певні властивості.
4. Об’єкти можна описувати та розповідати про них.
5. Існують об’єкти зі схожими властивостями.

6. Об’єкти можуть взаємодіяти один з одним.
7. Деякі об’єкти можуть виконувати дії.
8. З об’єктами можуть відбуватися якісь події.
9. Об’єкт це умовне позначення частини оточуючого світу, яку людина розглядає цілісно.

На завершення вчитель може дати деякі додаткові пояснення терміну: «Об’єкт» – це багатозначне слово. Ми будемо використовувати термін «об’єкт» як певну умовність. Об’єкти – фрагменти, на які людина «розчленовує» оточуючий світ. щоб, таким чином, означити та пізнати його. Об’єкт у нашому розумінні – це, певною мірою, одиниця людського мислення та навіть мови».

Учні повертаються до попередніх груп. Якщо у вступній частині уроку учні дуже погано визнали, що ж є об’єктом, то тепер можна ще раз розглянути це завдання. Нехай кожна група знову зачитає по 5 слів, а наступна прокоментує вибір істинними твердженнями, що збереглися на дошці.

**Дія 2**

**Метод роботи на даному етапі**: Робота з текстом на основі графічного організатора. **(**Ця робота має зайняти до 10 хв.)

**Основне завдання етапу**: розглянути ієрархічну структуру об’єднання об’єктів у класи а також елементи наслідування класів.

**Організація роботи:** Вчитель запрошує учнів до роботи з текстом «Об’єкти та класи» (див. додаток 3) та дає шаблон ієрархічної структури (додаток 4). Учні, залишаючись у групах, мають після читання тексту заповнити ієрархічне дерево конкретними об’єктами класів: ферма, будівлі, харчі, персонал, свині, ВРХ, вівці, порося, свинарник, миша, силос, корова, вівця, доярка, сіно, вода, теля, обора, свинарка, буряки, склад, свиня, стійло, вівчар, комбікорм, сторож. (Не обов’язково вимагати заповнення усього дерева.) Слова вчителя: «Пропоную прочитати текст про структуру об’єктів тваринницької ферми та побудувати ієрархічне дерево. Вище мають стояти ті класи, що включають у себе інші».

**Дія 3**

***Можливий варіант діяльності на уроці***

Якщо клас працює у швидкому темпі, вчитель може запропонувати виконати наступне завдання:

**Метод роботи на даному етапі**: продовження роботи з аналізу. **(**Ця робота має зайняти до 5 хв.)

**Основне завдання етапу**: з’ясувати деякі особливості організації класів.

**Організація роботи:** Ті ж чотири групи отримують завдання підібрати на картках зі словами, що отримали ще на початку уроку, об’єкти зі схожими властивостями та об’єднати їх за спільними рисами в певні класи.

Підчас обговорення ставляться запитання:

Що дає таке групування? (Варіанти правильної відповіді: впорядкування, спрощення, систематизацію, класифікацію, структуру.)

Чи можна було б вказані об’єкти згрупувати інакше? Як? (Варіант правильної відповіді: «У більшості випадків групувати об’єкти можна по-різному, навколо інших спільних властивостей»).

Чому ви обрали саме такий спосіб згрупувати об’єкти? (Варіанти правильної відповіді: більше спільних рис; одні об’єкти сформовані на основі інших.)

Чи зустрічаються на картках об’єкти, що сформовані на основі інших? (Варіант правильної відповіді: «Так, у них схожі риси.»).

**Дія 4.**

**Метод роботи на даному етапі**: Робота з текстом на основі методу опорних слів. **(**Ця робота має зайняти до 12 хв.)

**Основне завдання етапу**: розглянути основні поняття об’єктно-орієнтованого програмування: наслідування, поліморфізм, інкапсуляцію та поняття: поле класу, метод класу, подія.

**Організація роботи:** Вчитель запрошує учнів до роботи з текстом «Властивості класів» (додаток 5) з використанням методу опорних слів. Учні мають під час читання тексту виписати ті слова та словосполучення, які вони вважають ключовим для кожної ідеї тексту. Далі учні об’єднуються в пари та переказують одне одному зміст, спираючись на ключові слова. Також учні оцінюють, на скільки точно партнер переказує текст.

Зразок можливого переліку опорних слів: програма, наслідування (успадкування), батьківський клас, поліморфізм, інкапсуляція, властивості (поля), можливості (методи), значення поля, події.

*3. Заключна частина*

**Метод роботи на даному етапі**: Підбір списку синонімів. (Ця робота має зайняти не більш ніж 7 хв.)

**Основне завдання етапу**: повторити, узагальнити та підвести підсумок вивченого на уроці.

**Організація роботи:** Вчитель запрошує учнів повторити вивчене на уроці. Вони мають підібрати синоніми та уточнення до розглянутих на уроці термінів. Це можна проводити заповнюючи таблицю:

| Термін | Синоніми, пояснення. |
|---|---|
| Об’єкт | |
| Клас | |
| Наслідування | |
| Інкапсуляція | |
| Поліморфізм | |
| Властивості об’єкта | |
| Методи об’єкта. | |
| Інтерфейс об’єкта | |

Коротке обговорення зробленого.

Вчитель оголошує домашнє завдання: «Використовуючи читання з маркування, опрацювати текст «Класи та об’єкти в роботі комп’ютера» (див. додаток 6).

+ – позначте, що вже відомо;

- – позначте, що суперечить вашим уявленням;

! – позначте, що цікаве та неочікуване;

? – позначте, що незрозуміло та викликає бажання дізнатись більше

Результати роботи записати в таблицю:»

| + | – | ! | ? |
|---|---|---|---|
| | | | |

Вчитель має прокоментувати домашнє завдання, вказавши, що учням необхідно звернути увагу на відмінності між об’єктами в реальному житті та програмними об’єктами.

***Додатки (матеріали до уроку 1)***

***Додаток 1***

***Вказівки*** щодо завдань для груп

1 група. Виберіть картки де названо об’єкти, що є в нашому класі.

2 група. Виберіть картки де названо об’єкти, що є в нашій державі.

3 група. Відкиньте картки де вказано слова, що не позначають об’єкти.

4 група. Зробіть власний список об’єктів, що зустрічаються в редакторі Word.

***Приклади*** спеціально підібраних слів для розміщення на картках. Слова роздрукувати великим шрифтом розрізати так, щоб на картці розміщувалось лише одне слово чи словосполучення. Усі картки перемішати.

Молоко, кефір, чай, компот, сік.
Світанок, сутінки, зима, атака, велика перерва.
Львів, Київ, Вишгород, Миргород, Диканька.
Хвиля, приплив, вітер, захід сонця.
Посмішка, привітання, жест, танцювальне па, фігура вищого пілотажу.
Чотирикутник, квадрат, ромб, прямокутник, паралелограм.
Імпульс, сила, енергія, довжина, середина.
Інфляція, дефолт, монополізм, альтруїзм, неологізм.
Комунізм, фашизм, фетишизм, садизм, популізм.
Клаустрофобія, грип, апендицит, короткозорість, довгоногість.
Стіна, стеля, підлога, вікно, двері, плінтус.
Парта, дошка, стіл, стілець, шафа, люстра.
Вулиця, стежка, річка, водогін, залізниця.
П’єса, вистава, кінопрем’єра, танець, звук.
Червоний, довгий, значний, практичний, винятковий.
Бігти, ловити, шукати, писати, мріяти, міряти, марити.
Дванадцять, сорок чотири, багато, достатня кількість, трішки.
Любов, ненависть, дружба, ворожнеча, байдужість.
Обід, вечеря, полудень, швецький стіл, сніданок.
Прибирання класу, шкільний вечір, лінійка, змагання, урок.
Опитування, читання, писання, фотографування.
Зображення, тінь, луна, дзеркальне відображення.

***<u>Додаток 2</u>***

Приклади тверджень що стосуються об’єктів з яких потрібно вибрати істинні.

**Об’єкти завжди можна чітко відмежувати один від одного.**
Об’єкти завжди великі.
Об’єкти описують якийсь процес.
**Об’єкти можуть взаємодіяти один з одним.**
Об’єкти характеризують щось.
**Об’єкти існують незалежно від того чи знає про них хтось.**
Всі об’єкти хочуть їсти.
Об’єкти завжди зникають коли ми просинаємось.
Кожен об’єкт можна нагріти.
Всякий об’єкт є окремим побутовим предметом.
Всякий об’єкти можна брати до рук.
**Об’єкти можна описувати та розповідати про них.**
**Об’єкти мають певні властивості.**
**Існують об’єкти зі схожими властивостями.**
Різні об’єкти не можуть мати жодної однакової властивості.

**Деякі об’єкти можуть виконувати дії.**
Усякий об’єкт не може нічого виконувати самостійно.
**З об’єктами можуть відбуватися якісь події.**
Трапляються події, що не стосуються ніяких об’єктів.
**Об’єкт це умовне позначення частини оточуючого світу яку людина розглядає цілісно**.

***Додаток 3***

**Текст для роботи на уроці**

**Об’єкти та класи**

Сучасні програми найчастіше будують так, щоб вони якомога повніше повторювали особливості оточуючого світу.

Нехай нам треба створити програму яка б допомагала керувати тваринницькою фермою. Найперше ми повинні визначити з яких об’єктів складається модельована нами система – ферма.

Наприклад об’єктами на фермі може бути: власне сама «ферма п. п. Кіндрат», «корівник №1», у корівнику: «корова Зірка», «корова Муня», «телятко Сокіл», «доярка Марія Карпівна». Ферму охороняє об’єкт: «сторож Василь Семенович», і т.д. Наша програма може моделювати й іншу ферму де конкретні об’єкти будуть в інших кількостях, носитимуть не такі імена, матимуть інакші характеристики.

Коли ми будемо групувати об’єкти та визначати, як вони пов’язані використовується поняття «клас». Клас описує правила за якими діють об'єкти; ці об'єкти називаються «екземплярами" цього класу. Класи – це сукупності об’єктів, що мають спільні властивості.

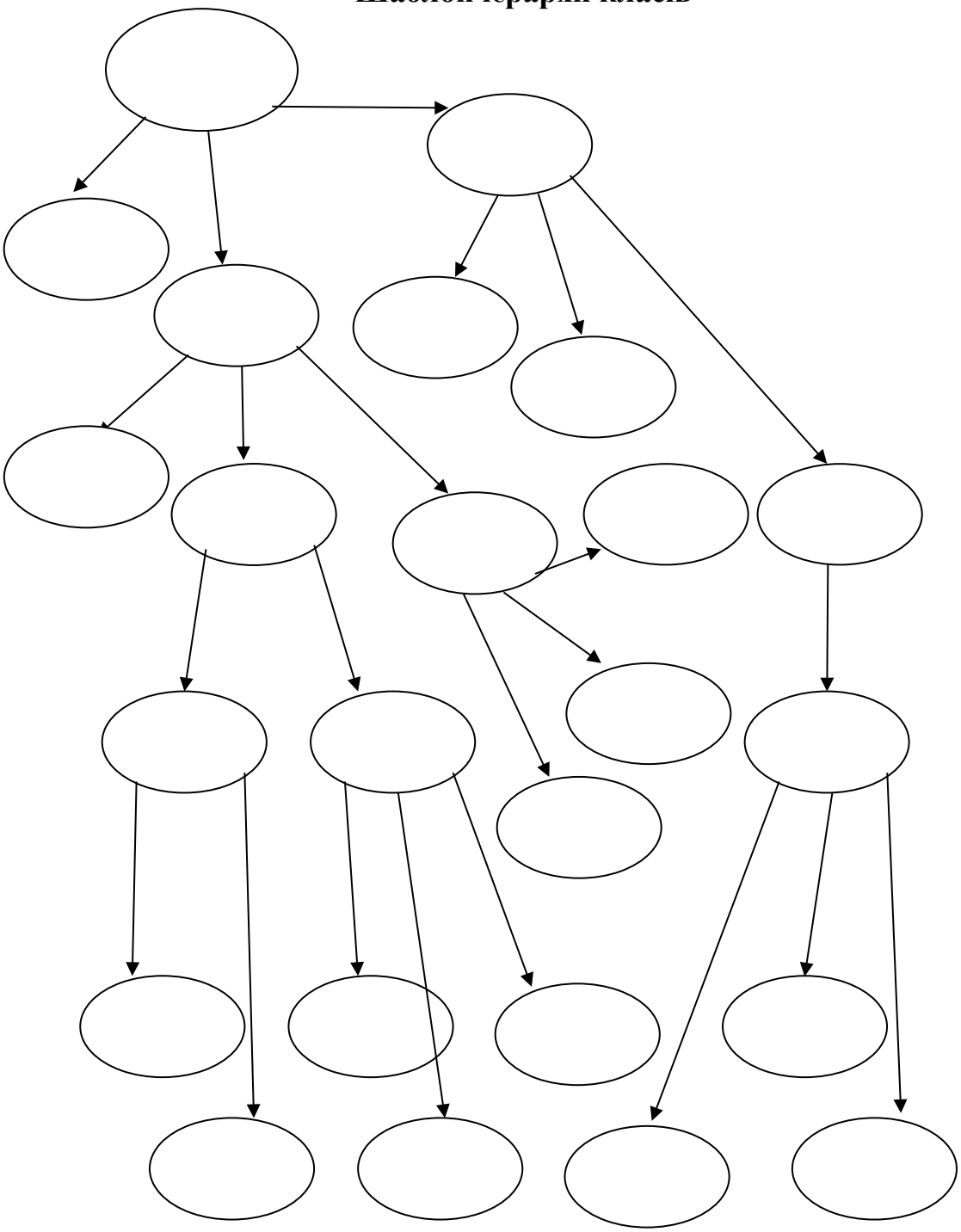
Стосовно спільних властивостей тварин, що утримуються на фермі, то вони, наприклад, характеризуються: масою, розмірами, споживають певні корми в конкретних кількостях і т.д. Тобто є підстави об’єднати усі тварини в один клас.

Класифікація об’єктів на фермі може здійснюватись по-різному, наприклад за розміщенням на території ферми: «корівник» об’єднує «корів», «телят» та «доярок» а клас «свинарник» буде мати підкласи: «свиней», «поросят», «свинарок». Класи можна структурувати за будь-якими ознаками: датою появи на фермі, вартістю, масою, т. і.. В будь-якому випадку утворюється певна чітка ієрархія класів програми. Яким чином формувати класи визначає програміст, і від цього багато в чому залежить механізм роботи моделі (програми), її ефективність. Значно спростити написання програми може така класифікація, в якій якомога більше особливостей загальних класів унаслідуються їх підкласами.

***Додаток 4***

**Графічний організатор для роботи на уроці**

**Шаблон ієрархії класів**



***Додаток 5***

**Текст для роботи на уроці**

## Властивості класів

Кожен клас характеризується певними властивостями (**полями**) та можливостями (**методами**). Наприклад властивостями корови є кличка, колір шерсті, маса, кількість споживання корму. Можливостями корови є можливість пастись, споживати корм, віддавати молоко, втікати. Конкретний об’єкт класу корови, наприклад корова «Зірка», має чорнорябу шерсть, масу 470 кг., споживає на добу 40 кг сіна і т.д.. Задаючи поля та методи класу ми визначаємо властивості та можливості які повинні бути описані в кожного з об’єктів даного класу. В кожного конкретного об’єкту те чи інше поле має своє певне значення.

**Події** - це зміни, що відбуваються з об'єктом. У реальному світі події відбуваються навколо нас безперервно. Наприклад, коли корова «Зірка» зголодніла, ця подія відбувається з нею. Події, як правило, являють собою дії, що впливають на певний об'єкт, але сам об'єкт вплинути на них не може. Дії виконуються об'єктом. Наприклад, корова «Зірка» жує жуйку - це дія.

При користуванні комп'ютером подіями можуть бути «натискання кнопки» або «переміщення миші». Іноді, коли відбуваються події, потрібно, щоб виконувалися певні дії, наприклад «перевірка правопису тексту при натисканні на кнопку F7». Сам по собі комп'ютер не має уявлення про те, яке слово написано вірно а яке ні. Комп’ютеру необхідно дати зрозумілий опис – метод. У нашому випадку це буде метод перевірки тексту. **Метод** являє собою набір покрокових інструкцій, які визначають порядок дій тобто алгоритм.

Іноді класи доцільно об’єднати в більш загальні: «худоба», «персонал», «корми», «будівлі». Також може виникнути завдання розділити деякі класи на підкласи. Наприклад худобу можна поділити на «ВРХ», «свині», «вівці». В такому випадку доцільно використовувати так зване **наслідування** (успадкування). Успадкованих клас буде містити в собі все, що зазначено для всіх його батьківських класів. Наприклад описати споживання корму можна для усього класу тварин. Тут проявляється наступна важлива риса – поліморфізм. **Поліморфізм** це коли один і той самий метод по-різному буде виконуватись стосовно різних умов. У нашому випадку метод «споживати корм» батьківського класу «худоба» для наслідуваного класу «ВРХ» передбачатиме споживання сіна, а класу «свині» – комбікорму.

При розробці програм слід забезпечити цілісність і взаємонезалежності об’єктів з допомогою інкапсуляції. Інкапсуляція це буквально розміщення в капсулу. **Інкапсуляція** класу передбачає його незалежність та захист від інших частин програми. Внесення зміни до окремого класу не повинне вплинути на інші частини програми.

***Додаток 6***

**Текст для домашньої роботи**

## Класи та об’єкти в роботі комп’ютера

Продемонструвати принцип об’єктної розробки програм можна на прикладі функціонування операційної системи Windows. Операційна система призначена для узгодження роботи багатьох прикладних програм.

Наприклад ви бажаєте малювати в редакторі «Paint» і при цьому слухати музику з допомогою програми «Windows Media Player». Кожна готова до запуску програма – це практично окремий клас.

Якщо запустити програму на виконання, наприклад відкрити Paint, то в операційній системі утвориться конкретний об’єкт цього класу, об’єкт класу «Paint». До того ж таких об’єктів у системі може існувати декілька – ніхто не забороняє запустити кілка екземплярів програми Paint і в кожному з них виконувати дещо інше завдання. Вони всі належать одному й тому ж класу Paint, але кожен з них має свої власні значення властивостей цього класу та дещо інше застосування його методів.

Програма «Media Player», у процесі відтворення музики, це також об’єкт операційної системи, але зовсім іншого класу ніж редактор «Paint».

Подібним чином відбувається взаємодія об’єктів і в самій прикладній програмі. Нехай користуючись Paint ви хочете зобразити еліпс. Вибравши відповідний пункт панелі інструментів на екрані з’являється об’єкт класу «еліпс». У процесі побудови користувач задає властивості еліпса: розмір, колір, товщину лінії.

На уроці ми розглядали, що кожен клас характеризується певними властивостями (**полями**) та можливостями (**методами**). Прикладом полів програми «Windows Media Player» є вигляд програми: розмір вікна, його колір та інше. Один з методів програми – перекодовувати дані звукового файлу в формат, що може відтворити звукова карта комп’ютера.

Запущений до виконання Media Player є об’єктом з яким можуть відбуватися **події**. Наприклад клацання лівої клавіші миші коли її курсор розміщено на кнопці «пауза». Також об'єкт Media Player виконує **дії** до того ж певними методами. Наприклад отримання відомостей про мелодію для її відтворення відбувається медом зчитування даних зі звукового файлу.

При створенні програм використовують **наслідування**. Програма Windows Media Player має багато полів та методів притаманних іншим комп’ютерним програмам. Підчас роботи програми Windows Media Player деякі методи проявляють **поліморфізм**. Наприклад один і той самий метод відтворення звуку по-різному буде реалізовуватись стосовно різних за форматом звукових файлів.

**Інкапсуляція** класу програми «Windows Media Player» передбачає його незалежність та захист від інших програми що виконуються в середовищі

Windows. Наприклад коли доводиться двом різним програмам зчитувати дані з файлів, вони не заважають одна одній.

**Урок 2. Тема: Моделювання та програмування.**

***Очікувані результати***

***Після цього заняття учні зможуть:***

- *Описувати поняття моделі, предметної галузі;*
- *Описувати типи моделей, їх характеристики;*
- *Пояснювати, що таке програма, дані, програмна логіка, інтерфейс;*
- *Визначати та розрізняти типи моделей;*
- *Дати визначення понять модель, інтерфейс;*
- *Оцінювати власний рівень уміння будувати інформаційну модель задачі.*

***Основні питання теми:***

Поняття програми як автоматизованої системи.
Складові програми: дані, логіка, інтерфейс.
Поняття моделі.
Типи моделей.
Моделювання як метод дослідження об'єктів.

***Обладнання:***

1. Тексти для самостійної та групової роботи в додатках.
2. Комп'ютери у класі з операційною системою, системою тестування та офісним додатком.
3. Файли з тестовими завданнями записані на усі учнівські комп'ютери. Зміст тестів наведено в додатку 3 .

***Регламент уроку***

1. Вступна частина – 6 хв. (виклик, групова робота з повторення)
2. Основна частина – 34 хв.

   Дія 1 (мозковий штурм) до 5 хв.

   Дія 2 (групова робота з текстами про дві сторони моделювання під час написання комп'ютерних програм) до 10 хв.

   Дія 3 (класифікація моделей на основі відомостей з тексту) до 10 хв.

   Дія 4 (тестування) до 9 хв.
3. Заключна частина – 5 хв. (побудова дерева асоціацій для узагальнення вивченого та остаточного означення моделювання).

***<u>Організація пізнавальної діяльності учнів</u>***

1. *Вступна частина*.

**Метод роботи на даному етапі**: групова робота з повторення. (Ця робота має зайняти не більш ніж 6 хв.)

**Основне завдання етапу**: З метою повторення матеріалу вивченого на попередньому уроці та опрацьованого дома, а також задля створення мотивації в учнів до вивчення нової теми, а також, щоб повторити матеріал інших розділів інформатики – вчитель пропонує розглянути різні комп'ютерні програми та дати відповіді на питання щодо їх функціонування.

**Організація роботи:** Об'єднуємо клас у групи по чотири учні, що сидять поряд. Найперше, учні мають обговорити прочитаний та промаркований у дома текст «Класи та об'єкти в роботі комп'ютера».

Наступне завдання є логічним продовженням обговорення. Потрібно пригадати певну комп'ютерну програму. (На дошці або на картках можна нагадати порядок запуску програм через головне меню.) В залежності від рівня підготовки класу слід обрати одну для всіх груп або кожній групі окрему програму зі стандартних програм операційної системи.

- **Блокнот,** «Пуск» – «Всі програми» – «Стандартні» – «Блокнот»;
- **Paint,** «Пуск» – «Всі програми» – «Стандартні» – «Paint»;
- **Калькулятор,** «Пуск» – «Всі програми» – «Стандартні» – «Калькулятор»;
- **WordPad,** «Пуск» – «Всі програми» – «Стандартні» – «WordPad»;
- **Звукозапис,** «Пуск» – «Всі програми» – «Стандартні» – «Розваги» – «Звукозапис».

Доручаємо кожній групі дати свої відповіді на питання, сформульовані за першими чотирма рівнями мислення з систематики Блума:

1. Які об'єкти містить інтерфейс програми? (знання)
2. Як, на вашу думку, зародилася ідея розробити таку програму? (розуміння)
3. Які «докомп'ютерні пристрої (об'єкти)» дана програма замінює? (застосування)
4. В чому полягає робота програми? (аналіз)

Відповідь на питання учні шукають колективно, але озвучують по одному. розподіливши питання у групі.

Приблизно через 3 хв. вчитель просить кожну групу презентувати програму яку вона досліджувала, давши відповіді на вказані питання. Корисно

наголосити, що робота на даному етапі отримає своє продовження при виконанні наступного домашнього завдання.

**Підсумок вступної частини**

По завершенні виступів вчитель представляє тему уроку та його очікувані результати: Ми сьогодні поглибимо знання про те, *що таке комп'ютерна програма* та дізнаємось дещо про те *як програми розробляють,* ***точніше, як вони зароджуються***. Тема уроку: «Комп'ютерні програми та моделювання».

*2. Основна частина*

**Дія 1**.

**Метод роботи на даному етапі**: Мозковий штурм. **(**Ця робота має зайняти до 5 хв.)

**Основне завдання етапу**: Ознайомити учнів з тим, що основними складовими усякої комп'ютерної програми є: дані, логіка та інтерфейс. А також, формувати поняття програми як автоматизованої системи.

**Організація роботи:** Вчитель звертається до усіх учнів з питанням: «А які об'єкти обов'язково є в будь-якій програмі?» Цим самим закликає до мозкового штурму. Далі вчитель просить уникнути повторень та узагальнити написане. Це можна зробити забравши (закресливши) ті елементи, що повторюються та ті, що можна віднести до одного і того ж типу. Далі узагальнення написаного продовжують групуючи об'єкти за схожим їх призначенням у програмі. В результаті узагальнення всі названі об'єкти потрапляють до однієї з трьох складових програми: **даних, логіки** чи **інтерфейсу**. У ході дискусії, не обов'язково явно, вказується на те, що у всякій комп'ютерній програмі реалізована певна автоматизація. Отже, формується поняття програми як автоматизованої системи.

**Дія 2**.

**Метод роботи на даному етапі**: Групова робота з текстами. (Ця робота має зайняти не більш ніж 10 хв.)

**Основне завдання етапу**: З'ясувати дві сторони моделювання під час написання комп'ютерних програм.

**Організація роботи.** Слова вчителя: «Ми детально розглянули складові самих різних комп'ютерних програм. Та все ще мало, що знаємо про те як вони створюються. З написанням програм тісно пов'язана така людська діяльність як моделювання». Моделювання стосовно програмування дає два взаємодоповнюючі підходи. Познайомитися з ними можна використовуючи текст 1 та текст 2 (див. додаток 1). Для цього об'єднаємо зусилля. Ті учні, що в своїх групах сидять лівіше читають текст 1, ті що правіше – текст 2. Завдання: дайте назву для тексту та коротко подайте його ідею. Прочитане корисно обговорити у своїх групах адже кожна група записує своє бачення ідеї тексту на дошці. Перемагає краще формулювання. Вчитель ділить дошку навпіл і підписує частини: «текст 1», «текст 2».

Можливі назви текстів:

Текст 1. Моделювання як шлях створення нового.

Текст 2. Моделювання як спосіб пізнання існуючого.

Можливі формулювання ідей:

Текст 1. Моделювання передує написанню програм.

Текст 2. Програма це моделювання чогось.

**Дія 3.**

**Метод роботи на даному етапі**: Класифікація моделей на основі відомостей з тексту. **(**Ця робота має зайняти до 10 хв.)

**Основне завдання етапу**: Ознайомитись з підходами до класифікації моделей. Усвідомити прокладі моделювання з власного досвіду та класифікувати їх за типами моделей.

**Організація роботи.** Слова вчителя: «Як зрозуміло з ваших слів кожна людина час від часу займається моделюванням. Безумовно кожен із вас також десь та щось моделював. Прошу написати в зошит назву одного зі своїх самих захоплюючих моделювань». Далі вчитель пропонує кільком учням зачитати записане і продовжує: «Моделі досить різні. Існує певна класифікація моделей. Щоб знати до якого типу належать ваші моделі прошу це визначити на основі тексту: «Класифікація моделей» (див. додаток 2). А ще кожна група повинна підібрати приклади чотирьох моделей які б належали до різного класу.»

**Дія 4**.

**Метод роботи на даному етапі**: Тестування. **(**Ця робота має зайняти до 9 хв.)

**Основне завдання етапу**:  Закріпити навчальний матеріал та оцінити рівень навчальних досягнень учнів.

**Організація роботи:** Вчитель наголошує, що настав час перевірити те, як учні працювали на сьогоднішньому уроці та те, як до нього готувались. На тестування виносяться питання за матеріалом попереднього уроку, домашнього завдання та щойно розглянутий матеріал. Учні виконують на комп'ютерах тестування (зразок змісту питань та відповідей тестів наведено в додатку 3)

*3. Заключна частина.*

**Метод роботи на даному етапі**: Побудова дерева асоціацій для узагальнення вивченого та остаточного означення моделювання[1]. (Ця робота має зайняти не більш ніж 5 хв.)

**Основне завдання етапу**: Дати визначення моделюванню та узагальнити багатогранність цього поняття. Також повторити, узагальнити та підвести підсумок вивченого на уроці.

**Організація роботи:** Вчитель запрошує учнів назвати ті асоціації до слова модель, що з'явилися в них на сьогоднішньому уроці та записати їх на дошці. Будується дерево асоціацій (воно може нагадувати дерево класифікації моделей). За результатами обговорення вчитель пропонує кожній групі запропонувати визначення моделі об'єкта та процесу її створення. В результаті обговорення добирається найкращі варіанти означень та наводяться їх приклад зі шкільного підручника:

***Модель об'єкта* – це новий об'єкт, який відображає властивості об'єкта суттєві для даного дослідження.**

**Процес створення та дослідження об'єктів на основі їх моделей називається *моделюванням*.**

**Оголошення домашнього завдання.** Вчитель просить пригадати, що на початку уроку учні презентували комп'ютерні програми даючи відповіді на певні питання. Домашнє завдання: «Запропонувати модель удосконалення тієї програми, що розглядала ваша група на початку уроку та сформулювати критерії за якими можна оцінити таку модель». Таким чином вчитель продовжує процес осмислення учням комп'ютерних програм на двох вищих рівнях мислення за систематикою Блума. Тобто шляхом синтезу та оцінювання.

---

[1] Метод побудови дерева асоціацій не прийнято використовувати для рефлексії але рідко який навчальний матеріал має таку багатогранність проявів як поняття «модель» та «моделювання». Вважаю, що підбираючи асоціації можна невимушено узагальнити величезну різноманітність та багатогранність розглянутих понять.

***Додатки (матеріали до уроку 2)***

***Додаток 1***

**Текст 1**

Якщо ви хочете спорудити собачу буду, то можете розпочати роботу маючи кілька дощок, жменю цвяхів та молоток. Після невеликого планування, ви, змайструєте цілком придатну конуру. Якщо вам треба побудувати будинок для своєї родини, ви, звичайно, можете скористатися тим же набором, але часу на це піде значно більше, і ваші домочадці виявляться більш вимогливими, ніж пес. Краще ретельно все продумати перед тим, як забити перший цвях, зробити хоча б кілька ескізів зовнішнього вигляду майбутньої споруди. А ще краще підготувати детальні креслення. На підставі цих планів ви зможете правильно розрахувати час, вибрати будматеріали. Поставивши собі за мету побудувати хмарочос, було б зовсім нерозумно братися за справу без детального планування як усієї споруди так і кожної деталі. Корисно навіть виготовити зменшену копію задуманого.

Хоча це і здається комічним, багато компаній, що розробляють програмне забезпечення, хочуть створити хмарочос, у той час як їх підхід до справи дуже нагадує будівництво собачої буди. А ще часто проекти задумані як маленька споруда швидко виростали до розмірів хмарочоса, стаючи жертвою власного успіху. Адже якщо розлетиться на друзки буда, це лише роздратує вашу собаку, а якщо звалиться хмарочос – це катастрофа.

Якщо ви дійсно бажаєте створити програмний продукт, який можна порівняти з житловим будинком або хмарочосом, то завдання не зводиться до написання великого обсягу коду. Хоча успіх програмного проекту забезпечується безліччю різних складових, одним із загальних є застосування моделювання.

Моделювання – це усталена і повсюдно прийнята інженерна методика. Ми будуємо архітектурні моделі будівель, щоб допомогти їх майбутнім мешканцям у всіх подробицях уявити собі готовий продукт. Іноді вдаються навіть до математичного моделювання будівель, щоб врахувати вплив сильного вітру або землетрусу.

Моделювання застосовується не тільки в будівництві. Навряд чи ви зумієте налагодити випуск нових літаків або автомобілів, не зазнавши свій проект на моделях: від комп'ютерних моделей і креслень до фізичних моделей в аеродинамічній трубі, а потім і повномасштабних прототипів. Навіть у кінематографії успіх фільму неможливий без попередньо написаного сценарію (теж своєрідна форма моделі). В соціології, економіці чи менеджменті ми також звертаємося до моделювання, яке дозволяє перевірити наші теорії і випробувати нові ідеї з мінімальним ризиком і витратами.

(За мотивами глави з книги: Г. Буч Язык UML. Руководство пользователя: Пер. с англ. / Г. Буч, Д. Рамбо, И. Якобсон / – М.: ДМК-прес, 2000. 432 с.: ил. ст..4-7)

**Текст 2**

Для вивчення властивостей і взаємозв'язків об'єктів (предметів, процесів або явищ) люди проводять різноманітні дослідження. Але не завжди можна або доцільно досліджувати самі предмети, процеси або явища безпосередньо. У

таких випадках створюють і досліджують не самі об'єкти, а їхні моделі. Термін модель походить від латинського слова *modulus* – зразок, аналог.

Моделі створюють для дослідження об'єктів тоді, коли сам об'єкт недоступний і його неможливо дослідити безпосередньо (наприклад, зірка сузір'я Великої Ведмедиці або виверження вулкана), або коли дослідження об'єкта можуть призвести до його руйнування (наприклад, мостовий перехід), або коли його виготовлення потребує значних коштів (наприклад, забудова нового мікрорайону) тощо.

Моделі об'єктів створюють не тільки тоді, коли вони недоступні або дорого коштують, а й тоді, коли потрібно дослідити конкретну властивість або групу властивостей об'єкта. У таких випадках створюють модель об'єкта, яка обов'язково має ті властивості, що досліджуються, а інші властивості, що є несуттєвими для даного дослідження, можуть бути в моделі відсутні.

Зазначимо, що в різних науках досліджують різні властивості об'єктів. І тому для кожного об'єкта можуть існувати різні моделі. Це залежить від того, які саме властивості досліджуються. Так, різними будуть моделі людини у дослідженнях фізика, хіміка, біолога, лікаря, модельєра.

З іншого боку, різні об'єкти можуть мати одну й ту саму модель. Так, прямокутний паралелепіпед може бути моделлю книжки, шафи для одягу, будинку та багатьох інших об'єктів. А функція $y = kx$ може слугувати моделлю прямолінійного руху матеріальної точки з постійною швидкістю, змінення напруги електричної мережі залежно від сили струму при постійному опорі, вартості покупки картоплі залежно від маси покупки та ін.

Можна стверджувати, що будь-яка розумова діяльність людини являє собою оперування образами предметів, процесів і явищ, які є, по суті, їхніми моделями. Дійсно, міркуючи про конкретний об'єкт, людина виділяє з усіх його властивостей лише окремі, які стосуються мети його розумової діяльності про предмети або явища, що досліджуються.

Очевидно, що правильна побудова моделей об'єктів та їхнє дослідження сприяють точності й правильності наукових та інженерних висновків, пропозицій, рішень. У сучасній науці та техніці побудова моделей, а також їхнє дослідження проводяться з використанням комп'ютерів, спеціальних комп'ютерних програм.

Крім того, створюють спеціальні комп'ютерні програми, які реалізують модель об'єкта. Такі програми називають **комп'ютерною моделлю об'єкта**. У наш час комп'ютерні моделі широко використовуються для дослідження об'єктів, проведення **обчислювальних експериментів** у тих випадках, коли проведення реальних експериментів неможливе, або потребує багато коштів, або має непередбачувані наслідки.

(Зі ст. 20-21 підручника Й. Я. Ривкінд, Т. І. Лисенко, Л. А. Чернікова, В. В. Шакотько Інформатика. 11 клас. Академічний рівень, профільний рівень.)

***Додаток 2***

***Текст для роботи на уроці***

**Класифікація моделей**

Моделі класифікують за різними ознаками (рис. 2.1):

за способом подання;

за галузями використання;

за фактором часу

та ін.

**За способом подання** моделі розподіляють на **матеріальні (предметні)** та **інформаційні**.

Іграшки, опудала тварин, манекени, муляжі, глобус, модель водяного млина – все це приклади **матеріальних моделей**. Матеріальні моделі призначені для проведення **практичних досліджень**.

Фізична карта України, рівняння хімічної реакції, математична функція, розповідь про береги Дніпра - це приклади **інформаційних моделей.** Інформаційні моделі призначені для проведення **теоретичних досліджень**.

***Матеріальна модель об'єкта – це модель об'єкта, подана у вигляді матеріального об'єкта (предмета).***

***Інформаційна модель об'єкта — це модель об'єкта, подана у вигляді його опису.***

Як і матеріальні моделі, інформаційні моделі одного й того ж об'єкта будуть різні, залежно від мети дослідження. Так, наприклад, інформаційна модель об'єкта «помідор» для постачальника міститиме дані про розміри, умови зберігання, фактори і терміни дозрівання, максимальні терміни зберігання тощо. А для фермера інформаційна модель цього самого об'єкта міститиме дані про час сіяння, регулярність прополювання і поливання, раціональне використання добрив тощо.

Інформаційні моделі у свою чергу розподіляють на:

**словесні** (усні та письмові описи);

**графічні** (рисунки, креслення, піктограми, карти та ін.);

**структурні** (таблиці, графіки залежностей, діаграми, схеми та ін.);

**алгоритмічні** (правила, плани дій та ін.);

**математичні** (формули, рівняння, нерівності, функції та ін.);

**спеціальні** (хімічні формули і рівняння, нотні записи, записи шахових партій та ін.).

Для створення інформаційної моделі об'єкта потрібно:

1. Визначити об'єкт дослідження, для якого створюється модель, і мету дослідження.

2. Виділити ті властивості об'єкта, які є суттєвими для вказаного дослідження.

3.Установити взаємозв'язки між вибраними властивостями та виразити їх, використавши одну з форм представлення.

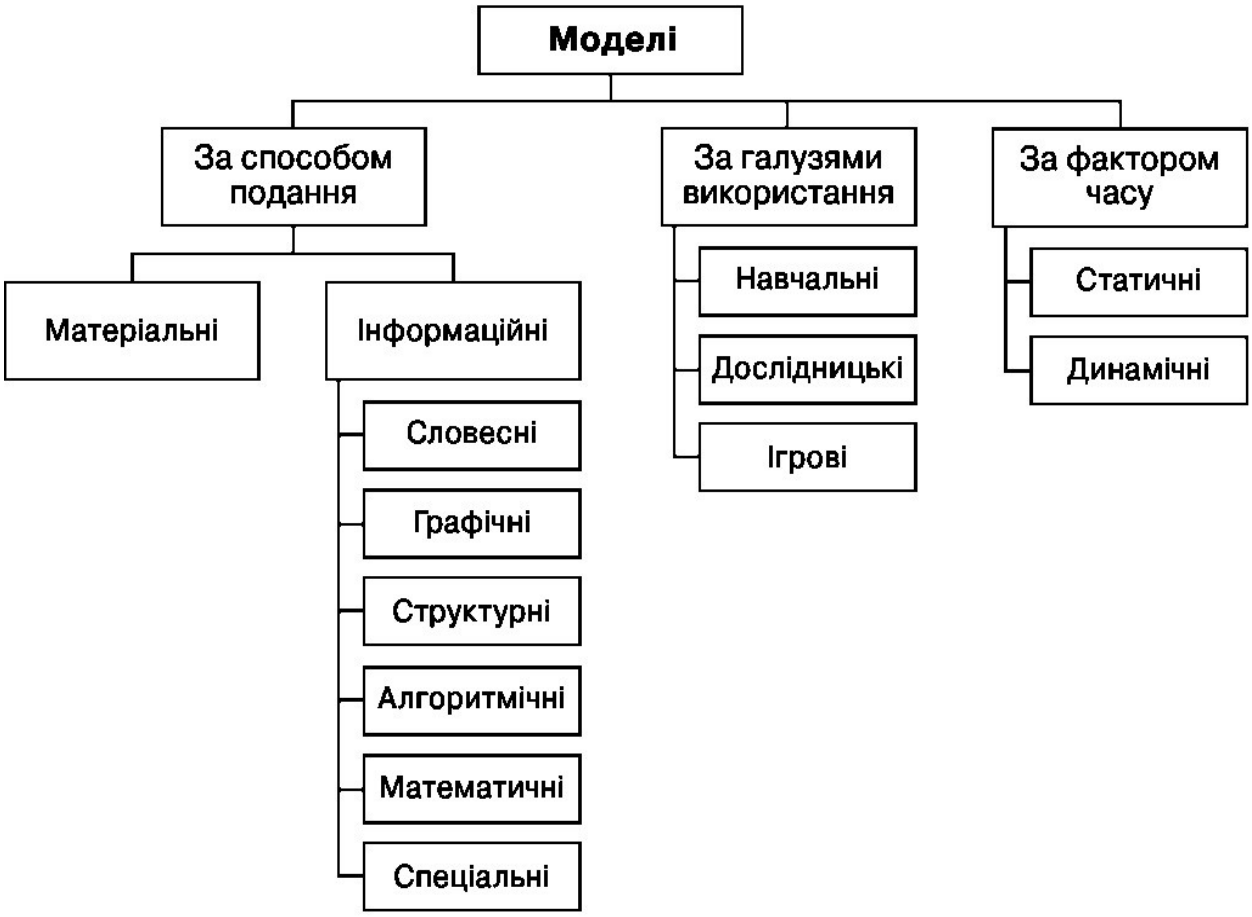



Рис. 2.1. Типи моделей

(Зі ст.22-23 підручника Й. Я. Ривкінд, Т. І. Лисенко, Л. А. Чернікова, В. В. Шакотько Інформатика. 11 клас. Академічний рівень, профільний рівень.)

***Додаток 3***

**Тест: Об'єкти та їх моделі**

*1. Об'єкти завжди*

**можна** чітко відмежувати один від одного
завжди великі

хочуть їсти
**мають** певні властивості
**існують** незалежно від того чи знає про них та чи інша людина

2. *Об'єкти ніколи не можуть*

нічого виконувати самостійно
взаємодіяти один з одним
**безпідставно** з’являтись та зникати
мати певні властивості
виконувати якісь дії

3. *Усі об'єкти можуть*

**взаємодіяти** один з одним
нагріватись та охолоджуватись
потрапити в чужу кишеню
**мати** певні властивості
використовуватись у програмуванні

4. *Класи це*

методи конкретних об’єктів
це сукупності об’єктів, що мають спільні властивості
об'єднання програм в одне ціле
**сукупність** можливих чи існуючих об'єктів
засіб інкапсуляції

5. *Об’єднання в класи дозволяє*

**реалізовувати** успадкування
надавати об’єктам нові властивості
**зберігати** властивості існуючих класів для використання в нових
перетворювати програму в машинний код
уникати інкапсуляції

6. *Наслідування це?*

**утворення** нових класів на основі існуючих
різне виконання одного й того ж методу в залежності від умов
захист внутрішньої структури об’єктів
дослівно «розміщення в капсулу»
**засіб**, що дозволяє багаторазово використовувати фрагменти коду програми

7. *Поліморфізм це?*

утворення нових класів на основі існуючих
**різне** виконання одного й того ж методу в залежності від умов
захист внутрішньої структури об’єктів
дослівно «розміщення в капсулу»
засіб, що дозволяє багаторазово використовувати фрагменти коду програми

*8. Інкапсуляція це?*

утворення нових класів на основі існуючих
різне виконання одного й того ж методу в залежності від умов
**захист** внутрішньої структури об’єктів
**дослівно** «розміщення в капсулу»
засіб, що дозволяє багаторазово використовувати фрагменти коду програми

*9. Які об’єкти в роботі програми Paint?*

**панель** інструментів
**засіб** збережена малюнку в файл
**зображувані** елементи малюнку
електронний словник
процесор та пам’ять

*10. До якого типу об’єктів програми Paint відноситься засіб збережена малюнку у файл?*

інтерфейс
дані
**логіка**
апаратні складові комп’ютера
власність користувача

*11. До якого типу об’єктів програми Paint відносяться зображувані елементи малюнку?*

інтерфейс
**дані**
логіка
апаратні складові комп’ютера
власність користувача

*12. До якого типу об’єктів програми Paint відноситься панель інструментів?*

**інтерфейс**
дані
логіка
апаратні складові комп’ютера
власність користувача

*13. Модель це*

**спрощене** подання якогось процесу чи об'єкту
узагальнення теоретичних положень розрізнених галузей знань
виключно високотехнологічне виробництво
незалежна експертиза
усякий об’єкт

*14. Моделювання використовують для*

**проектування** чогось нового
заборони використання пластикових пакетів
абсолютно для будь-чого
**перевірки** гіпотез та припущень
**навчання**

*15. Моделі доречно класифікувати за*

**способом** подання
напрямком розвитку
засобами знищення
**галузями** використання
**фактором** часу

*16. За способом подання моделі поділяються на*

**матеріальні** та інформаційні
статичні та динамічні
навчальні, дослідницькі та ігрові
**вербальні** та знакові
незалежні та автономні

*17. За галузями використання моделі поділяються на*

матеріальні та інформаційні
статичні та динамічні
**навчальні**, дослідницькі та ігрові

вербальні  та знакові
**залізничні,** аерокосмічні, геологічні, автотранспортні і так далі.

*18. За фактором часу моделі поділяються на*

матеріальні та інформаційні
**статичні** та динамічні
навчальні, дослідницькі та ігрові
вербальні  та знакові
незалежні та автономні

*19. Якого типу моделі є комп'ютерна програма*

**інформаційна**
вербальна
уявна
**знакова**
матеріальна