МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Т.А. ВАКАЛЮК, Л.Д.ШЕВЧУК, С.А.ПОСТОВА

Структурне та візуальне програмування

навчальний посібник
для студентів фізико-математичного факультету

Переяслав-Хмельницький 2019

УДК 004.42+004.432.2
ББК 73р
В14

Затверджено Вченою радою ДВНЗ "Переяслав-Хмельницький державний педагогічний університет імені Григорія Сковороди", протокол № 10 від 18.06.2019 р.

Рецензенти:

Медведєва М.О. – кандидат педагогічних наук, доцент, завідувач кафедри інформатики і інформаційно-комунікаційних технологій Уманського державного педагогічного університету імені Павла Тичини;

Почтовюк С.І. – кандидат педагогічних наук, доцент кафедри інформатики і вищої математики Кременчуцького національного університету імені Михайла Остроградського;

Хомич В.Ф. – кандидат педагогічних наук, доцент, перший проректор з навчально-виховної роботи ДВНЗ «Переяслав – Хмельницький державний педагогічний університет імені Григорія Сковороди».

Вакалюк Т.А., Шевчук Л.Д., Постова С.А.

В14 Структурне та візуальне програмування. Навчальний посібник для студентів фізико-математичного факультету. – Переяслав-Хмельницький: вид-во ПХДПУ, 2019. – 318 с.

Посібник призначений для використання студентами під керівництвом викладача на лекціях, практичних та лабораторних заняттях. Посібник містить лекційний та практичний курс з структурного та візуального програмування. Викладений матеріал відповідає діючій програмі з програмування для майбутніх учителів інформатики. Також даний посібник може бути використаний учителями для навчання учнів програмування за поглибленою програмою.

Для студентів фізико-математичних спеціальностей вищих педагогічних закладів, вчителів інформатики загальноосвітніх шкіл.

УДК 004.42+004.432.2
ББК 73р

Зміст

Вступ

Суспільство вступило в важливий період свого розвитку – еру інформатизації. Використання електронних обчислювальних машин перейшло в сферу безпосереднього виробництва.

Для вирішення теоретичних і практичних задач, що виникають при діяльності людини у різних галузях науки, техніки та виробництва з метою звільнення людини від надмірного інтелектуального навантаження великий ефект дає використання обчислювальної техніки при умові достатнього програмного забезпечення й ефективного його використання.

Дисципліна "Програмування" орієнтована на майбутніх учителів інформатики. Базується на засвоєнні студентами основних понять про мови та системи програмування, а також основних принципів об’єктної методології. Вивчення курсу дозволить студентам оволодіти методами структурного програмування, теоретичними знаннями щодо принципів структурного підходу при розробці програм, вміння користуватись сучасним програмним забезпеченням.

Курс "Програмування" має своєю метою:

- надання основних знань з структурного програмування, а також практичних навичок використання методів та засобів сучасних інформаційних технологій у повсякденній практичній діяльності;
- підготувати студентів до ефективного використання сучасних комп’ютерних технологій при розв’язуванні фахових завдань.

Рівень підготовки по курсу повинен дозволити студентам працювати з різними середовищами програмування, створювати власні програми та проекти.

Навчальна дисципліна "Програмування" входить до вибіркової частини циклу дисциплін за вибором студентів та доповнює знання з фундаментальних дисциплін для використання їх на практиці.

Предмет "Програмування" є логічним продовженням курсів "Інформатика" та "Математична логіка" і змістовно пов’язана з базовими дисциплінами. Засвоєння студентами основних положень цієї дисципліни поряд з освітньо-пізнавальним має і науково-прикладне значення на початковому етапі навчання і формування фахівця загалом.

Навчальним планом передбачається: вивчення дисципліни на лекційних та лабораторних заняттях, самостійна робота студентів; перевірка основних теоретичних знань та практичних умінь студентів за допомогою тестових завдань та контрольної роботи; складання іспиту.

Основними труднощами при вивченні даної дисципліни є багатоплановість матеріалу, який розглядається, і його великий об'єм. Тому успішне засвоєння курсу не можливе без регулярної самостійної роботи з літературою і творчого відношення до виконання практичних і лабораторних робіт.

Основними завданнями, що мають бути вирішені у процесі викладання дисципліни, є надання студентам необхідних знань з теорії і практики використання алгоритмічних мов програмування, сформувати уявлення у студентів про основні етапи розв'язування задачі ЕОМ, послідовність дій, вміння та навички роботи з сучасним програмним забезпеченням, налагодження програми.

У результаті вивчення курсу студент повинен знати: основні етапи розв'язування задач з використанням ЕОМ; поняття інформаційної моделі задачі; властивості алгоритмів, форми подання алгоритму; основні базові структури алгоритмів; сутність методу послідовного уточнення алгоритму; основні принципи структурного та об'єктно-орієнтованого програмування; порядок складання алгоритмів і програм; правила запису алгоритмів і програм; основні елементи однієї з мов програмування; алфавіт, основні поняття мови: числа, рядки, описи, ідентифікатори, оператори, величини, вказівки; типи даних у мові програмування, набір функцій і операцій, допустимих для кожного з типів даних; принципи побудови опису програми; сутність операції присвоювання; призначення та правила описування вказівок розгалуження й повторення; звернення до алгоритмів; поняття по алгоритми-процедури та алгоритми-функції; особливості використання табличних та рядкових величин.

У результаті вивчення курсу студент повинен вміти: користуватись програмами FreePascal, ABCPascal; створювати програму на мові Pascal, ABCPascal, а також переконатися, що всі її дії виконуються коректно, при необхідності налагодити програму; опрацьовувати масиви; працювати з файлами тощо.

Розділ "Візуальне програмування" є логічним продовженням у вивченні мов та систем програмування, але вже на якісно новому, сучасному рівні, з врахуванням тенденцій широкого застосування інформаційних технологій в навчальному процесі.

Проблемний характер лекції може бути реалізований за допомогою проблемно- символічних сигналів, що лежать в основі модельно-символічної технології організації розвивального навчання. Проблемна символіка може використовуватися як при викладанні нового матеріалу, так і при узагальненні матеріалу лекції. У першому випадку лектор записує на дошці чи то проектує проблемно-символічний сигнал (студенти отримують його запис разом із конспектом лекції), в основі якого лежить пара термінів (понять, принципів, положень), що характеризують тематику лекції. Подальше викладання матеріалу лекції супроводжується поступовим заповненням сигналу лектором. За такої організації роботи наприкінці лекції перед студентами буде "побудований" логічний конструкт теоретичного матеріалу із основними характеристиками предмету лекції. Слід зазначити, що можливий варіант, коли один із термінів опорної пари вже знайомий студентам. В такому випадку відбувається ознайомлення з новим матеріалом через призму вже відомого. На наступному етапі використання проблемної символіки можливе і при узагальненні матеріалу лекції. В такому випадку

лектор, підсумовуючи сказане, пропонує аудиторії самостійно виконати проблемно-диференційоване завдання. Така робота на лекційному занятті сприятиме не просто репродуктивному засвоєнню нового програмного матеріалу, а й більш виразнішому, логічному зв'язку його з матеріалом попередніх лекцій, що в свою чергу позитивно впливає на рівень усвідомлення навчального матеріалу. Під час вивчення курсу передбачено обов'язкове виконання індивідуальних завдань. Завдання можуть бути як реферативного, так і практичного характеру (конкретні програмні реалізації).

ОБ'ЄКТ КУРСУ - технологія візуального програмування.

ПРЕДМЕТ - система візуального об'єктно-орієнтованого програмування Delphi.

МЕТА КУРСУ:

• ознайомлення із основними методологіями програмування та висвітлення процесу їх еволюції;

• загальний огляд основних технологій програмування;

• розгляд основних положень технології візуального програмування.

ЗАВДАННЯ КУРСУ:

• набути навичок практичної роботи із системами, що підтримують принципи візуального програмування;

• детально ознайомлення із системою візуального програмування Delphi.

• навчитися використовувати систему візуального програмування Delphi у власній професійній діяльності.

Частина І. Структурне програмування

Теоретичний матеріал

З історії програмування

Будь-яка мова програмування – це формалізована система позначень і правил для точного та зрозумілого записування алгоритмів і їх виконання.

Алгоритм, записаний певною мовою програмування, називають комп'ютерною програмою.

Паскаль - одна з найпоширеніших мов програмування 80-90-х років XX ст., що підтримує найсучасніші методології проектування програм (спадне, модульне проектування, структурне програмування).

Августа Ада Кінг (уроджена Байрон), графиня Лавлейс (англ. Augusta Ada King Byron, Countess of Lovelace, зазвичай згадується просто ***Ада Лавлейс*** (10 грудня 1815, Лондон, Великобританія - 27 листопада 1852, там же) - англійка-математик. Відома перш за все створенням опису обчислювальної машини, проект якої був розроблений Чарльзом Беббіджем. Склала першу в світі програму (для цієї машини). Ввела у вживання термін «цикл» і «робоча комірка», вважається першим програмістом

Чарльз Беббідж (англ. Charles Babbage; 26 грудня 1791, Лондон, Англія - 18 жовтня 1871, там же) – англійський математик, винахідник першої аналітичної обчислювальної машини. Іноземний член-кореспондент Імператорської академії наук у Санкт-Петербурзі (1832). Праці з теорії функцій, механізації рахунку в економіці. Сконструював і побудував (1820-22) машину для табулювання. З 1822 працював над будівництвом різницевої машини. У 1833 розробив проект універсальної цифрової обчислювальної машини - прообразу сучасної ЕОМ.

Власне перші мови програмування з'явилися задовго до появи перших комп'ютерів. Ще в 19-му столітті існували "програмовані" ткацькі верстати та піаніно-програвачі, спосіб програмування нагадує так звані предметно-орієнтовані мови програмування. На початку 20-го століття починають використовуватись перфокарти, та механічна обробка даних. В 1930 -1940 рр. виникає лямбда-числення та машина Тюринга, які застосовували математичну абстракцію для опису алгоритмів. Лямбда-числення згодом здійснило вплив на проектування мов програмування.

В 1940 роках створюються перші електричні, двійкові комп'ютери. Вважається, що першу мову програмування високого рівня —Планкалькюль (нім. *Plankalkül*) розробив німець Конрад Цузе в період 1943 -1945 році, але в той час вона не була реалізована і не одержала уваги. Реалізацією мови зайнялися і здійснили лише в 1998-2000 роках.

У кінці 40-их — початку 50-их застосовувалися інтерпретовані системи кодування, коли певні команди мови програмування кодувалися числами, які уже інтерпретувалися машинним кодом. Ці системи називалися «автоматичним програмуванням», були простішими для програмування, ніж

машинні коди, але могли мати значно меншу (до 50 разів) швидкодію, через що часто надавали перевагу машинним кодам. До таких систем належали — Short Code для BINAC (1949) і UNIVAC I (1952), Speedcodingдля IBM 701, розроблена Джоном Бекусом у 1954.

Першою широковживаною компільованою мовою став розроблений групою *Джона Бекуса* Фортран, анонсований у 1954 році і випущений у 1957 для IBM 704. Основним призначенням Фортрану були швидкі наукові обчислення, оголошувалося що швидкодія згенерованого компілятором коду майже не відрізнятиметься від машинного коду написаного вручну. Уже у квітні1958 близько половини програм для IBM 704 були написані на Фортрані. Випущений у 1958 році Фортран II дозволяв незалежну компіляцію підпрограм, що дозволило створювати більші програми, оскільки низька надійність IBM 704 не дозволяла скомпілювати без збоїв велику програму (понад 300—400 рядків) одразу. Розроблений у 1960—1962 роках Фортран IV був однією з найпоширеніших мов того часу і лишався стандартною версією Фортрану до появи у 1978 році Фортран 77.

У 1958 році у MIT розробили LISP — першу функційну мову, яка понад чверть століття домінувала у програмуванні задачштучного інтелекту.

У кінці 1950-их почали розроблятися різні мови програмування. У 1958 році декілька значних груп комп'ютерних користувачів уСША, включаючи SHARE — групу науковців-користувачів IBM і USE (UNIVAC Scientific Exchange, група науковців-користувачівUNIVAC) запропонували ACM заснувати робочу групу зі створення універсальної мови програмування. Також ще у 1955 році німецьке Товариство прикладної математики і механіки (GAMM) заснувало комітет зі створення універсальної мови програмування. У кінці травня 1958 року було проведено зустріч у Цюриху між ACM і GAMM, на матеріалах якої у грудні опубліковано "ALGOL 58 Report". На його основі було створено 3 значні реалізації — MAD (1961), NELIAC (1963), JOVIAL (1963). З них лише JOVIAL отримав поширення, ставши на чверть століття офіційною мовою програмування у Військово-морських силах США. SHARE і IBM почали створення власної реалізації ALGOL, але припинили, врахувавши витрати на створення і просування Фортрану.

Впродовж 1959 року ALGOL 58 широко обговорювався, була запропонована нотація для опису синтаксису мов програмування — форма Бекуса-Наура. У 1960 проведено чергову зустріч і опубліковано ALGOL 60 Report. ALGOL вплинув на багато мов програмування і став стандарною мовою для публікації алгоритмів, але через ряд причин не одержав широкого поширення — він був заскладним, і не було реалізацій, які підтримували його повністю, відсутність стандартного введення-виведення привела до появи різних несумісних реалізацій, деякі неоднозначності опису мови так і не були розв'язані. Також широкого вжитку уже набув Фортран, і IBM не підтримала ALGOL.

У 1959 році була проведена зустріч у Пентагоні для створення мови CBL (Common Business Language), засновано комітет з його створення, і у

1960 опубліковано початкову специфікацію COBOL 60, який невдовзі став першою мовою прийнятою уМіністерстві оборони США. У 1968 році COBOL було стандартизовано ANSI.

У 1964 році було створено спрощену мову BASIC (Beginners All-purpose Symbolic Instruction Code) для навчання програмуванню студентів, які переважно спеціалізувалися у вільних мистецтвах, а не технічних науках.

Тоді як науковці переважно використовували Фортран, а бізнес — COBOL, у 1963 році в IBM вирішили створити універсальні платформу IBM/360 і мову програмування. У стислі терміни до 1965 року було розроблено мову PL/I, яка поєднувала можливості Фортран, ALGOL і COBOL, і виявилась заскладною, хоча і була у широкому вжитку у 1970-их у наукових і бізнес задачах, також її підмножини (PL/C, PL/CS) використовувалися для навчання програмуванню.

На початку 1960-их було створено перші мови із динамічною типізацією — APL і SNOBOL.

SIMULA 67 була першою об'єктно-орієнтованою мовою програмування.

Мова Паскаль має свою досить багату історію розвитку, початок який поклало оголошення в 1965 р. конкурсу зі створення нової мови програмування - спадкоємця АЛГОЛА-60. Участь у конкурсі взяв швейцарський учений ***Ніклаус Вірт***, що працював доцентом на факультеті інформатики Стенфордського університету.

Ніклаус Вірт, професор, директор Інституту інформатики Швейцарської вищої політехнічної школи, лауреат Тьюрингівської премії, автор багатьох широко відомих праць з програмування. Н. Вірт також є автором таких мов програмування, як Ейлер, Модула, Модула-2. Окрім цього, ним запропонована методика покрокової розробки програм - від глобального до локального, від загального до часткового, - тобто занурення в алгоритм зверху донизу. Ця методика була визнана найсильнішою ідеологією програмування 70-х рр. XX ст.

Проект, що був запропонований ним, був відхилений комісією в 1967 р. Але Вірт не припинив роботу над створенням нової мови. Повернувшись до Швейцарії, разом зі співробітниками Швейцарського федерального інституту технології в Цюріху, він уже в 1968 р. розробив першу версію мови Паскаль.

Це не абревіатура (як Фортран, Алгол), - мова названа на честь великого французького математика й механіка Блеза Паскаля, який в 1642 р. створив першу рахункову машину.

Вірт вирішив узятися за створення нової мови програмування з методичною метою: дати своїм студентам інструмент для вивчення програмування як систематичної, логічної дисципліни, що базується на фундаментальних поняттях.

У своїй роботі "Систематичне програмування" Ніклаус Вірт пише: "...мова, якою студента навчають висловлювати свої думки, має глибокий вплив на його навички мислення та винахідницькі здібності...".

В 1971 р. Н.Вірт випустив опис своєї мови, а в 1975 р. було розроблене керівництво для користувачів версії Паскаля, що практично лягла в основу стандарту ISO на мову Паскаль, який з’явився у 1982 р.

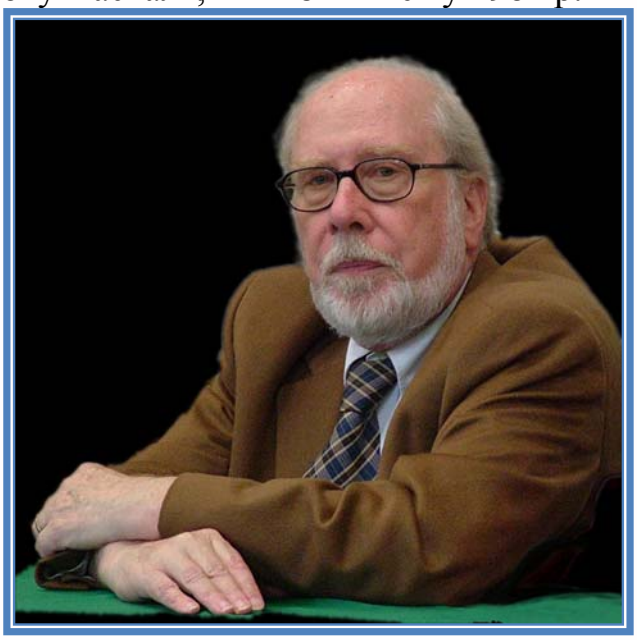


Рис. 1 Ніклаус Вірт

Пройшло багато часу з моменту появи Паскаля на ринку програмних продуктів, перш ніж він одержав загальне визнання. Визнання програмістів і простих користувачів прийшло внаслідок появи мови програмування Турбо Паскаль (ТП) - діалекту мови, створеного американською фірмою Борланд. Ця фірма об'єднала дуже швидкий компілятор з редактором тексту й додала до стандартного Паскалю потужне розширення, що сприяло успіху першої версії цієї мови.

В 1985 році на ринку ПЕОМ з'явилася мова програмування Турбо Паскаль (версія 3.0) з компілятором стандартного Паскаля. З тих пір Паскаль став застосовуватися в загальноосвітніх, професійно-технічних школах й у сфері вищої освіти в якості "першої" мови програмування. Завдяки простоті використання мова Турбо Паскаль одержала широке поширення й в аматорських колах. Підвищенню популярності Турбо Паскаля сприяв набір невеликих супутніх програм, що дозволяють одержувати надзвичайно компактну, швидку й легку у читанні програму. Ці якості Турбо Паскаля були високо оцінені й у середовищі професійних програмістів. Вбудований редактор тексту використовує досить широко розповсюджену систему команд, що бере початок від пакета WordStar і добре знайому кожному, хто інтенсивно використовує ПЕОМ.

У пакеті, що з'явився згодом, Турбо Паскаль 4.0 була усунута більшість обмежень компілятора, що піддавалися критиці, й була підвищена продуктивність системи. Крім того, новий компілятор версії 4.0 мав істотні відмінності від попередньої версії. Найбільш важливим нововведенням була UNIT-концепція, запозичена з мови Модула-2. Це дало можливість реалізувати в рамках мови розробку великих програмних продуктів.

З виходом у світ версії 5.0 Турбо Паскаль одержав ще більші шанси на позитивну реакцію з боку професійних користувачів завдяки вбудованому в середовище програмування інтегрованому відладчику, що дозволив підвищити продуктивність роботи.

Істотно поліпшила технічні характеристики Турбо Паскаля реалізація апарата перекриттів (overlays), що дозволяє будувати потужні програмні комплекси, розраховані на експлуатацію в малих за обсягом областях пам'яті. Суть механізму перекриттів зводиться до розподілу програми на частині, по черзі, що завантажують по мірі необхідності з магнітного диску й ту саму область пам'яті, заміняючи при цьому частину, що перебувала там, програми.

Крім того, у Турбо Паскалі 5.0 були розширені можливості налагодження (debugging) програм і забезпечена можливість підтримки розширеної пам'яті в стандарті Lotus-Intel-Microsoft (LIMS/EMS 4.0). Скорочення EMS позначає Expanded Memory Specification (специфікація розширеної пам'яті).

У цій версії були також виправлені й поліпшені бібліотеки графічних процедур, що поставляють разом з пакетом Турбо Паскаль. При цьому забезпечувалася повна сумісність із графічними адаптерами класу VGA (Video Graphics Array).

У рамках версії Турбо Паскаль 5.5 були здійснені подальші перетворення в напрямку поліпшення технічних характеристик пакета. Поряд із внутрішніми поліпшеннями й новими можливостями вбудованої довідкової системи Help і більшим набором навчальних прикладів, важливим нововведенням виявилася реалізація в мові концепції об’єктно-орієнтованого програмування (ООП).

Через деякий час на ринку з'явилася версія 6.0, у якій чисто теоретична концепція об’єктно-орієнтованого програмування була реалізована практично з повним набором об'єктів, які могли використатися для рішення прикладних задач користувача. Крім того, реалізація системи меню, приведена у відповідність зі стандартом SAA (Turbo Vision). Як практичний приклад використання нових можливостей був реалізований текстовий редактор, вбудований в IDE - Integrated Development Environment - інтегровану інструментальну оболонку. При цьому прихильники програмування на Турбо Паскаль 6.0 отримали можливість не тільки працювати з вбудованим багатовіконним текстовим редактором, але й використовувати мишу, що значно полегшує роботу користувача.

В 1992 році фірма Borland International представила користувачам чергову версію мови програмування Паскаль - Турбо Паскаль 7.0. Поряд з усіма перевагами, які Турбо Паскаль 7.0 успадкував від попередньої версії Турбо Паскаль (багатовіконний режим роботи, можливість використання миші, можливість використання при написанні програм мови програмування низького рівня Асемблер, можливість створювати об’єктно-орієнтовані програми), у ньому були зроблені зміни й поліпшення.

По-перше: з'явилася можливість виділяти певним кольорами різні елементи вихідного тексту (зарезервовані слова, ідентифікатори, числа й т.і.),

що дозволяє навіть недосвідченим користувачам усувати помилки на етапі введення вихідного тексту. По-друге: мова програмування Турбо Паскаль 7.0 була розширена (з'явилася можливість використовувати типізований адресний оператор, відкриті масиви й рядки й т.і.), що надало користувачеві додаткових можливостей при розв'язанні повсякденних завдань. По-третє: був поліпшений компілятор, внаслідок чого "коди програм" стали більш ефективними. По-четверте: був поліпшений інтерфейс користувача. Крім того, у Турбо Паскале 7.0 розширені можливості об'єктно-орієнтованого програмування (зокрема, розширені й поліпшені можливості Turbo Vision).

Мова програмування Паскаль, на відміну від усіх попередніх популярних мов, пропонує програмістові прості конструкції команд, які виглядають дуже природно. Набір команд мови Паскаль зовсім невеликий і складає її базову частину. Усі додаткові можливості Паскаля дають змогу програмістові користуватися типами змінних, властивих лише цій мові програмування, а також створювати свої власні типи.

Таким чином, мова Паскаль дуже швидко набула популярності серед програмістів і була реалізована для різних типів комп'ютерів.

Середовище програмування FREE Pascal

У системі Free Pascal об'єднано **текстовий редактор** для набирання текстів програм, **компілятор** для виявлення помилок у програмах, та запуску програм на виконання в разі відсутності помилок, **налагоджувача** для покрокового виконання програми, відслідковування значень змінних для виявлення складних помилок.

Інтегроване середовище запускається файлом fp.exe. Після запуску середовища програмування ви побачите (див. малюнок) у верхній частині екрана меню, а в нижній рядок повідомлень.

Середовище Free Pascal дозволяє користувачу працювати як за допомогою клавіатури, так і за допомогою мишки. Для ефективності роботи в даному середовищі рекомендуємо засвоїти ряд гарячих клавіш.

F10 – активізація меню

Alt+F – активізація меню файл.

Розглянемо пункти головного меню.

File – меню для роботи файлами, яке містить наступні команди:

- **New (Alt+F -> N)–** відкрити нове вікно для створення програми.
- **Open(F3)** – відкрити з диска програму, яку було раніше збережено
- **Save(F2)** – зберегти вміст активного вікна на диску.
- **Save as..** – зберегти вміст активного вікна під новим вказаним іменем.
- **Save all** – зберегти вміст всіх активних вікон.
- **Change dir**… – зміна поточного каталога для читання або запису файлів.
- **Print** виведення на пристрій друку вміст активного вікна

- **Command shell** – тимчасовий виклик командного процесора операційної системи для виконання його команд (повернення і середовище програмування за допомогою команди EXIT).
- **Exit(Alt-X)** – Завершення сеансу роботи в інтегрованому середовищі.

Edit – редагування програми в активному вікні.

Для виділення фрагменту тексту необхідно тримаючи клавішу **Shift** переміщати курсор клавішами керування курсора. Зняти відмітку виділеного тексту за допомогою натиснення клавіш Ctrl+K+H або натисненням лівої кнопки миші.

- **Undo (Alt+BkSp)** – відмінити останню операцію.
- **Cat (Shift+Del)** – знищення виділеного фрагменту з занесенням його в буфер обміну.
- **Copy (Ctrl+Ins)** – копіювання виділеного фрагменту в буфер обміну.
- **Paste (Shift+Ins)** – вставлення фрагменту з буферу обміну в активне вікно починаючи з позиції курсора.
- **Clear (Ctrl+Del) -** знищення виділеного фрагменту без занесення його в буфер обміну.
- **Show clipboard** – поміщення вмісту буферу обміну в окреме вікно з можливістю редагування.

Рис. 2. Екран інтегрованого середовища програмування Free Pascal

Run:

- **Run** (Ctrl+F9) – запуск програми на виконання з одночасним пошуком синтаксичних помилок.
- **Program reset (Ctrl+F2)** - закриття всіх відкритих програмою файлів і встановлення лічильника на початок програми.
- **Go to cursor (F4)** – запуск програми на виконання до позиції курсора.
- **Trance into (F7)** – покрокове виконання програми з входженням у всі функції.

- **Step over (F8)** - покрокове виконання програми без входженням у всі функції (виконання функції як одна операція).

Compile:

- **Compile (Alt + F9**) Компіляція активної програми.
- **Make (F9)** - перевірка синтаксичних помилок в програмі.

Debug:

- **Add watch..(Ctrl + F7)** – добавлення змінної до вікна watch для відслідковування її під час покрокового виконання програми.
- **Output** – відкриття вікна виведення.
- **User Screen (Alt+F5)** перехід на екран виведення

Window

- **Size/Move (Ctrl+F5)** – входження в режим зміни положення і розміру активного вікна. Клавші керування курсора міняють положення вікна, Shift+ «стрілка» змінює розмір вікна.
- **Zoom (F5)** – розгорнути активне вікно на весь екран.
- **Next (F6)** – перехід до наступного вікна.
- **Previous (Shift+F6)** – перехід до попереднього вікна.
- **Close(Alt + F3)** – закрити активне вікно.
- **Close all** – закрити всі відкриті вікна.

Help:

- **Index (Shift+F1)** – відкриття алфавітний перелік ключових слів довідки або відкриття довідки по ключовому слову на якому знаходиться курсор.
- **Topic search (Ctrl+F1)** - відкриття довідки по ключовому слову на якому знаходиться курсор.
- **Previos topic (Alt+F1)** – відкриття останнього відкритого вікна довідки

Додаткові гарячі клавіши

Ctrl+Y – знищити стрічку, де знаходиться курсор.

Базові елементи мови програмування Pascal. Лінійні програми.

Основні поняття мови Паскаль.

Будь-яка мова програмування характеризується трьома основними складовими: алфавіт, синтаксис і семантика.

Сукупність символів, які дозволяється використовувати при побудові опису програм мовою програмування, називають ***алфавітом мови***.

Сукупність правил (опису) побудови вказівок алгоритмів деякою мовою програмування називають ***синтаксисом мови програмування***.

В будь-якій мові програмування можна виділити чотири типи елементів, що використовуються при побудові описів програм:

- символи,
- слова,
- вирази,
- команди (оператори).

Символи мови - це основні нероздільні знаки, за допомогою яких описуються програми і дані.

Слова мови - структури, що утворені із символів алфавіту мови програмування і мають певний зміст. Слова - це імена змінних та констант, числа, службові слова та ін.

Вираз **-** це текст, що задає правило обчислення одного значення величини. Якщо одержуване значення числове, то вираз називають арифметичним, якщо значення логічне, то вираз називають логічним або бульовим, якщо одержуване значення - текст, то вираз називають літерним.

Команда (оператор) - це вказівка про виконання деякої дії. При написанні програм команди називають операторами, а величини, що використані в команді - операндами.

Алфавіт мови. Літери латинського алфавіту, цифри, спеціальні символи, зарезервовані слова.

Зарезервовані слова є складовою частиною мови, мають фіксоване написання і назавжди визначений зміст. Наприклад: **begin**, **else**, **function**, **goto**, **end**, **program** тощо.

Стандартні слова призначені для заздалегідь визначених розробником мови типів даних, констант, процедур і функцій (наприклад, **sin**, **cos**, **Pi**).

Ідентифікатори – імена об'єктів: констант, типів даних, змінних, функцій, програм.

Правила запису ідентифікаторів:

- всі ідентифікатори складаються з літер латинського алфавіту, цифр, починаються з літери або знаку "_"
- великі і малі літери не розрізняються
- ідентифікатори можуть мати різну довжину, але використовуються тільки перші 63 символи.

Елементи даних

Поняття величини

За допомогою величин передаються всі значення, виконуються різні обчислення.

Величиною називають таку характеристику предмета або явища, значення якої можна виміряти або обчислити.

Види величин

- ***сталі (константи)***
- ***змінні***

Константа – комірка пам'яті або величина, значення якої на протязі виконання програми залишається незмінним.

CONST *< Ідентифікатор> = < значення константи>;*

Змінна – комірка пам'яті або величина, значення якої на протязі виконання програми може змінюватись.

Для опису змінних використовується службове слово VAR.

VAR <список ідентифікаторів>: < тип>;

Наприклад:

VAR i,x,r1,d : real;

t,ca : integer;

Кожна змінна повинна бути описана тільки один раз на початку програми після слова VAR

Основні характеристики величин

1) Іменем (ідентифікатором) змінної величини може бути будь-яка послідовність латинських літер (*a-z* та *A-Z*), цифр та знака підкреслення, яка повинна починатись з літери. Вони застосовуються для позначення констант, типів, змінних, процедур і функцій. У стандарті мови імена розрізняються за першими 8-ма символами.

2) Тип величини - це набір даних (констант, змінних, значень функцій і т.д.), які мають спільні характеристики її значень (формат подання в пам'яті ПК, множина допустимих значень, множина допустимих операцій, що можна використовувати для даного типу).

3) Значення – динамічна характеристика величини, яка може багаторазово змінюватись в процесі обробки інформації.

Коментар – призначений для внесення до тексту програми пояснень. Обмежується символами {} та (* *).

Рис. 3. Мухамед ібн Муса аль-Хорезмі

Слово "**алгоритм**" є перефразуванням географічної назви місцевості Хорезм через праці відомого узбецького математика Мухамеда ібн Муса аль-Хорезмі.

У IX ст. великий узбецький математик Мухаммед, уродженець Хорезма (арабською "аль-Хорезмі"), розробив правила виконання чотирьох арифметичних дій над числами. Множину цих правил навали алгоритмом (algorithmi – від латинського написання імені аль-Хорезмі), а потім словом "алгоритм" почали позначати сукупність правил певного виду, а не тільки правил виконання арифметичних дій.

Алгоритм – це скінченна та впорядкована послідовність вказівок (команд), формальне виконання яких дозволяє за обмежений час отримати розв'язок задачі.

Властивості алгоритму

- **дискретність –** процес, що визначається алгоритмом, можна (розділити) на окремі елементарні етапи (кроки), кожен з яких називається кроком алгоритмічного процесу чи кроком алгоритму;
- **результативність (скінченність)** – вказує на наявність таких варіантів вхідних даних, для яких обчислювальний процес, що реалізується за наданим алгоритмом, повинен через скінчену кількість етапів (кроків) зупинитись і дати шуканий результат або сигнал про те, що наданий алгоритм непридатний для розв'язання поставленої задачі.
- **детермінованість (визначеність)** – через повну однозначність правил, встановлених в алгоритмі, застосування алгоритму до однакових вхідних даних повинно приводити до однакового результату;

- **масовість** - алгоритм повинен бути придатним для розв’язування широкого класу задач певного типу.

Способи опису алгоритмів:

- природною мовою (зрозумілою для людини);
- засобами мови програмування;
- графічним способом (у вигляді блок-схем).

Програма – це алгоритм, команди якого “зрозумілі” комп'ютеру і можуть бути ним виконані.

Існують різні класифікації мов програмування.

Відповідно до найпоширенішої з них мови програмування поділяються на:

• мови низького рівня (мова мікрокоманд, машинна мова, асемблер);

• мови високого рівня (Basic, Pascal, Сі).

Існує ще одна ***поширена класифікація*** мов програмування:

Рис. 4 Існуючі мови програмування

Однак існують класифікації і за іншими ознаками. Так відповідно до “***прикладної***” класифікації мови програмування поділяють за ***галузями застосування***.

Процедурні мови програмування – це мови програмування, в яких реалізована можливість покрокової реалізації алгоритму, низхідного та висхідного програмування

Логічне програмування – це запис задачі мовою математичної логіки у вигляді набору висловлювань, відношень між висловлюваннями та правил виведення одних висловлювань з інших.

Об’єктно-орієнтоване програмування – це нова ідеологія програмування, що базується на використанні сукупності об'єкта та подій, на які він може реагувати.

Візуальне програмування – це практичне застосування об'єктно-орієнтованого програмування при використанні готових бібліотек компонентів, передбачених середовищем програмування.

Транслятор – це програма, що призначена для перекладу тексту програми з однієї мови програмування на іншу. Процес перекладання називається трансляцією.

Розрізняють два типи трансляторів:

- компілятори;
- інтерпретатори.

Компілятор – це програма, призначена для перекладу в машинні коди програми, що написана мовою високого рівня. Процес такого перекладання називається компіляцією.

Інтерпретатор – це програма, що призначена для трансляції та виконання вихідної програми по командах (на відміну від транслятора, який цей процес виконує в цілому). Такий процес називається інтерпретацією.

Основні етапи розв'язання задач

Загальний алгоритм розв'язування задач з використанням ПК має вигляд (див. рис. 5):

Рис. 5. Загальний алгоритм розв'язування

Розглянемо більш детально IV етап – тестування і налагодження програми: перевірка правильності роботи програми за допомогою тестів і виправлення виявлених помилок. Для цього спочатку дамо визначення ключових слів даного етапу.

Налагодження (англ. debugging) – це пошук і виправлення помилок у розроблюваній програмі.

Програмні помилки, як правило поділяються на три види: синтаксична, семантична, логічна.

Синтаксична помилка – невірне використання синтаксичних конструкцій, або помилка в написанні зарезервованих слів. Ці помилки виявляти найпростіше, адже компілятор см виявить їх і вкаже вам на них.

Семантична помилка – помилка у програмі, яка пов'язана з неправильним змістом дій та використанням недопустимих значень величин (наприклад, помилки даних: символьні замість числових, ділення на 0, корінь з від'ємного числа тощо).

Логічна помилка – порушення логіки програми, яке призводить до невірного результату. Подібні помилки ховаються в алгоритмах і потребують ретельного аналізу та всебічного тестування.

Для налагодження найчастіше використовують покрокове виконання програми, що забезпечує слідкування за значеннями змінних на різних етапах виконання програми.

Тому для зменшення ймовірності виникнення помилок, використовується *захисне програмування*. Найпростіший метод використання захисного програмування полягає у тому, що при написанні програми потрібно передбачити обробку ситуацій, які не можуть статись ні за яких обставин:

1. Перевіряйте тип вхідних даних. Контролюйте літерні поля, щоб переконатися, що вони не містять цифрових даних. Перевіряйте цифрові поля на відсутність в них літерних даних.
2. Робіть перевірку області значень змінних, щоб упевнитися, наприклад, що додатні величини завжди додатні.
3. Виконуйте контроль правдоподібності значень змінних, які не повинні перевищувати деяких значень.
4. Контролюйте підсумки обчислень шляхом введення всюди, де це можливо, перехресних підсумків та контрольних сум.

Тестування (англ. testing) – це випробування, перевірка.

Якщо ж вживати більш ширше поняття, то *тестування* – це виконання комплексу завдань для перевірки на правильну працездатність програми.

Наголосимо на основних принципах тестування програм:

1. Використовуйте принцип *захисного програмування* (див. вище).
2. *Тестуйте граничні умови (*так звані контрольні тести*)*:
   - в умовному виразі необхідно переконатися, що розгалуження виконується вірно;
   - при написанні циклу потрібно передбачити перевірку того, чи тіло циклу буде виконано потрібну кількість разів;

   Основною ідеєю є те, що коли трапляється помилка, то можна сказати з досить великою ймовірністю, що вона пов'язана саме з виходом за граничні значення. І навпаки, якщо програма працює вірно при усіх граничних значеннях тестових даних, то більш за все вона буде поводитись коректно і у звичайних умовах.
3. *Аналізуйте результати тестування*. Це можна зробити декількома способами: для порівняння обчисліть результат іншим способом

(наприклад на калькуляторі, або іншою програмою), використовуйте табличні дані тощо.

4. *Тестуйте окремі блоки незалежно один від одного*. Враховуйте проміжні результати.

Описані принципи тестування програми у спільному використанні здатні забезпечити правильне виконання програми. При цьому сам процес тестування програми можна розділити на три етапи (див. рис. 6):

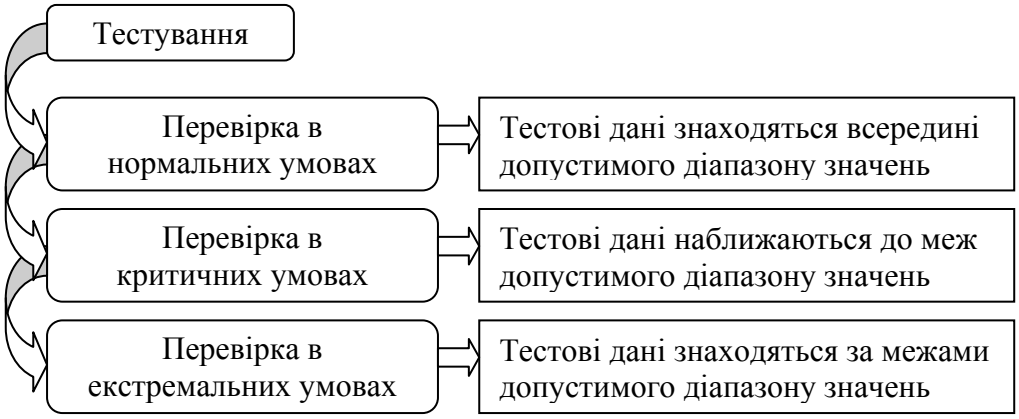
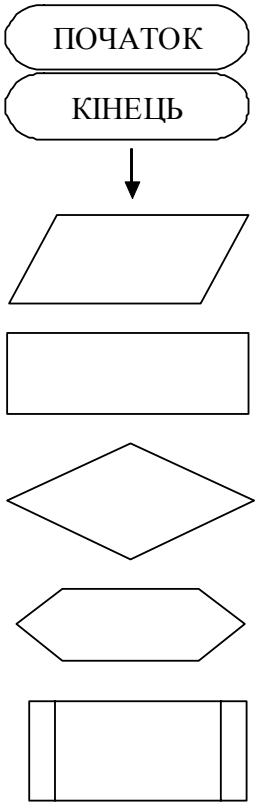


Рис. 6. Етапи процесу тестування

Тест – це набір вхідних даних, для яких заздалегідь відомий результат. Приклад створення та перевірки тестів наведемо трішки пізніше.

Умовні позначення блок-схем



Початок алгоритму

Кінець алгоритму

Лінії потоку

Введення, виведення даних

Блок обчислень

Розгалуження, початок циклу з передумовою, кінець циклу з післяумовою

Початок циклу з лічильником

Виклик підпрограми

Структура програми мовою програмування Pascal

Заголовок програми

PROGRAM – Заголовок програми

Описова частина

USES – Бібліотеки, що підключаються
LABEL – Оголошення глобальних міток
CONST – Оголошення глобальних констант
TYPE – Оголошення глобальних типів
VAR – Оголошення глобальних змінних
Частина опису процедур та функцій
PROCEDURE/FUNCTION – Заголовок процедури/функції
LABEL – Оголошення локальних міток
CONST – Оголошення локальних констант
TYPE – Оголошення локальних типів
VAR – Оголошення локальних змінних
BEGIN – Початок блоку процедури/функції
END; – Кінець блоку процедури/функції
Основний блок програми (розділ операторів)
BEGIN – Початок основного блоку програми
END. – Кінець основного блоку програми

Розглянемо найбільш важливі частини вищеописаних блоків. Під заголовком програми розуміється ім'я, що допомагає визначити її призначення. Після імені програми варто поставити «;», цей знак служить у Паскалі для поділу послідовних інструкцій. Помітимо, що ім'я програми може не збігатися з ім'ям відповідного файлу на диску.

Глобальні директиви компілятора

У цьому розділі програми компілятору можна дати вказівки, що визначають режими його роботи при трансляції програми. Ці вказівки оформляються в тексті програми як коментарі, що починаються парою символів ({$) і закінчуються символом (}). Такі вказівки можуть містити "замовлення" на включення в текст програми фрагментів інших програм (з відповідних файлів), інформацію чи відомості про необхідності використання арифметичного сопроцесора.

Оператор USES

Оператор USES відіграє важливу роль у підключенні до тексту програми системних модулів з бібліотек. У цьому операторі Ви вказуєте компілятору, з якої бібліотеки використовує модулі програма. Поняття "бібліотека", "модуль", "блок" складають основу термінології програмування на Паскалі. Бібліотека включає набір модулів, кожен з яких, має власне ім'я, компілюється окремо і до нашої програми підключається вже як "чорний ящик" з відомим інтерфейсом. Кожен модуль (блок (UNIT), як його називають на Паскалі) являє собою програму, що включає декларації типів і змінних, процедури і функції.

Опис міток

Опис міток уводить сукупність ідентифікаторів і/або цілих чисел, призначених для організації послідовності обчислень. За допомогою міток можна відзначити (указати) ті оператори, на які буде передане керування з інших точок програми. Передача керування на позначений оператор реалізується за допомогою спеціального оператора переходу.

Опис констант

Після слова *const* містяться описи сталих, котрі будуть використані в програмі, наприклад:

const *Zero = 0;*
pi = 3.1415926;
my_const = -1.5;

Визначення типів

Визначення типів призначене для завдання конкретних множин значень. Зазначені множини позначаються іменами (ідентифікаторами) і надалі можуть служити для опису змінних.

Оголошення глобальних змінних

За словом var розташовують оголошення змінних, котрі знадобляться нам при написанні програми. Змінні Паскаля можуть зберігати дані різної природи: числа, рядки тексту, окремі символи і т.п. Нижче приводиться частина типів перемінних, котрі можна застосовувати.

Оголошення змінних записуються в наступній формі:

var <змінна> : <тип>;

Якщо описуються декілька змінних одного типу, то досить записати їхні імена через кому, а після двокрапки поставити загальний тип.

Приклади оголошення:

var *Number: integer;*
d,l: real;
Name: ***string****[20];*
Line: ***string****;*
Key1,Key2: char;

Опис процедури або функції

Опис процедури або функції визначає частину програми як окрему синтаксичну одиницю і зіставляє з нею ім'я. Згодом дії, зосереджені в процедурі (функції), можуть бути виконані ("викликані") за допомогою вказівки її імені. Крім дій, опис процедури (функції) може містити сукупність описів локальних об'єктів, що утворять власний контекст імен. Процедури і функції є основним засобом структурування програми.

Блок основної програми.

Тут, між словами begin і end. розташовуються команди (точніше, оператори), що будуть виконуватися один за іншим при запуску програми. Розглянемо найпростіші типи операторів на наступному прикладі:

Зауваження про імена. Для позначення змінних забороняється використання ряду слів, називаних зарезервованими, вони грають у мові особливу роль. Нам уже зустрічався ряд зарезервованих слів: program, begin, end, string, const, var, і т.п.

Тип даних **–** це діапазон значень, що можуть приймати об'єкти програми, і сукупність операцій, які дозволяється виконувати над цими значеннями.

Усі типи даних у мові програмування Паскаль розділяються на дві групи:

- скалярні (прості),
  - стандартні типи
    - цілі
    - дійсні
    - літерний
    - логічний
  - типи користувача
- структуровані (складені)
  - масиви
  - множини
  - файли
  - записи
  - об'єкти
  - посилання

Таблиця 1

Типи змінних

Тип	Можливі значення	Примітка
Byte	0..255	Цілочисельний тип данних
Word	0..65535	
Integer	-32768..32767	
Longint	2147483648..2147483647	
Single	$1.5x10^{-45}$ до $3.4x10^{38}$	Числовий тип даних з певною кількістю знаків після коми
Real	$2.9x10^{-39}$ до $1.7x10^{38}$	
Double	$5.0x10^{-324}$ до $1.7x10^{308}$	
Extended	$3.4x10^{-4932}$ до $1.7x10^{4932}$	
Char	1 символ	Символьний тип даних Береться з обох сторін в апостроф
String	група символів	Рядковий тип даних Береться з обох сторін в апостроф
Boolean	True або False	Логічний тип

Цілі типи даних являють собою значення, що можуть використовуватися в арифметичних виразах.

При виході значень даних цілого типу за вказаний діапазон помилки виконання програми не буде, але результат виводиться неправильний.

Дійсні числа – це десяткові дроби і, в окремому випадку, цілі числа, записані у вигляді десяткового дробу.

Наприклад:

Числа 5; 0 – цілі числа,

Числа 5.0;0.0 – дійсні числа.

Дійсні числа можуть бути записані двома способами:

- з фіксованою крапкою (5.45; 9.23);
- з плаваючою крапкою (3E+5; -8.1E-4);

Запис числа з фіксованою крапкою

Дійсні десяткові числа з фіксованою крапкою записуються за звичайними правилами арифметики.

Ціла і дробова чистини дійсного числа розділяються десятковою крапкою, а не комою!

Наприклад: 0.39; -37.4

Якщо десяткова крапка відсутня, число вважається цілим. Перед числом може знаходитися знак "+" або "-". Якщо знак відсутній, за замовчуванням число вважається додатнім. Число не може починатися з крапки і не може нею закінчуватися. Записи 0. і .89 недопустимі.

Запис числа з плаваючою крапкою

Дійсні ***десяткові числа у форматі з плаваючою крапкою*** подаються в наступному (експоненціальному) виді:

mE+p, де *m* – мантиса (ціле або дробове число з фіксованою десятковою крапкою), *E* – означає "десять в степені", *p* – порядок степеню (ціле число).

Наприклад:

Таблиця 2

Запис числа з плаваючою крапкою

Число у форматі з плаваючою крапкою	**Значення числа**
0.4500E+02	0.45*102=45
-2.600E05	-2.6*105=-260000
+0.45670E-02	0.4567*10-2=0.004567

Типи користувача

Перелічувальний – задається безпосереднім переліком імен всіх значень, які можуть приймати змінні даного типу. Окремі значення записується через кому, а весь список береться в круглі дужки.

Type *<ім'я типу>=<значення1, значення2, …, значенняN>;*

Var *<ідентифікатор,…>:< ім'я типу>;*

Наприклад:

Type *Фрукти= (яблуко, груша, вишня, черешня);*

Var *Фрукти;*

Обмежений (інтервальний) – задається двома константами, які визначають границі діапазону значень для даної змінної

Type *< ім'я типу>=<константа1.. константа2>;*

Var *< ідентифікатор,…>:< ім'я типу >;*

Наприклад:

Const Min=1; Max=100;

Type *temperatyre=min..max;*

Організація введення та виведення даних

Оператор введення

read (список імен); або readln (список імен);

Оператор виведення

write (список значень); write(s:n:m); {s – змінна типу real, a n,m – integer} де n – кількість цифр у всьому числі; m – кількість значущих цифр у дробовій частині змінної.

Приставка ln до кожного з операторів означає, що після виконання даного оператору (введення або виведення) курсор переводиться на нову стрічку. Якщо дана приставка ln до будь-якого з операторів відсутня, то після

виконання даного оператору (введення або виведення) курсор залишається у тій самій стрічці.

Оператор присвоєння

Присвоєння відбувається командою :=.

Загальний вигляд: ідентифікатор:=вираз.

Присвоює змінній конкретне значення, заповнюючи комірку пам'яті, відведену для змінної, новим значенням, водночас знищуючи старе.

Стандартні операцій та функції мови Pascal

Оператори – це деякі неподільні елементи програми, що дозволяють виконувати певні алгоритмічні дії у програмі, тобто виконувати в програмі певні команди. Фактично, ***оператор*** – це окрема команда в алгоритмі програми, тобто окремий крок виконання програми.

Операнди – це спеціальні символи або послідовності символів, які виконують над даними певні операції (математичні, логічні і т.д.). Операнди математичних операцій: “+”, “-”, “*”, “/”, “=”

Таблиця 3

Операції відношення

Операція відношення	***Значення***
>	більше
> =	більше або дорівнює
<	менше
<=	менше або дорівнює
< >	не дорівнює

Таблиця 4

Математичні операції

Математична операція	***Значення***
+	додавання
–	віднімання
*	множення
/	ділення
div	ділення націло (наприклад, 2 div 2 =1 означає, що при діленні 2 на 2 ціла частина= 1)
mod	остача при діленні націло (наприклад, 2 mod 2 =0 означає, що при діленні 2 на 2 дробова частина=0)

Таблиця 5

Математичні функції

Математична функція	***Значення***
sqrt (x)	корінь квадратний з деякого числа x
sqr (x)	квадрат деякого числа x

sin (x)	синус кута x (x – у радіанах)
cos (x)	косинус кута x
arctan (x)	арктангенс числа x
abs (x)	модуль числа x
exp (x)	експонента числа x
ln (x)	натуральний логарифм числа x
round (x)	округлення числа до цілого, згідно математичних правил округлення
trunc (x)	ціла частина числа: від числа відкидається дробова частина (результат дії даної функції набуває типу Integer або Longint)
frac (x)	дробова частина числа: відкидається ціла частина, а дробова частина записується як ціле число
odd (x)	визначає, до парних чи непарних чисел відноситься дане число. Результат дії функції Odd набуває типу Boolean – true, якщо число парне або false в іншому випадку
int (x)	ціла частина числа, тобто, те ж саме, що і функція trunc, але на відміну від дії функції trunc, результат дії функції Int залишається типу real

Це важливо! Математичні операції виконуються в такому порядку – спочатку виконуються дії в дужках, потім обчислюються функції, виконується множення і ділення (зліва направо), і лише потім додавання і віднімання (якщо вони не виконались раніше в дужках).

Таблиця 6

Математичні процедури

Математична процедура	***Значення***
inc (x,y)	Збільшує x на y
inc (x)	Збільшує x на 1
dec(x,y)	Зменшує x на y
dec (x)	Зменшує x на 1

Базові алгоритмічні конструкції.

Базові алгоритмічні конструкції (управляючі структури) – це способи управління процесами обробки даних. Комбінуючи керуючі структури, можна складати програми для розв'язання різноманітних завдань.

Виділяють три базові алгоритмічні конструкції:

- лінійні алгоритми (послідовне виконання);
- розгалуження (виконання при певних умовах);
- цикли.

Лінійна структура – передбачає, що тіло програми являє собою послідовність операторів, що виконуються підряд один за одним.

Правила написання Pascal-програм

1. Основний текст будь-якої програми починається службовим словом **begin** і закінчується словом **end,** після чого слідує крапка. Без крапки

програма вважається незакінченою. І навпаки, якщо в програмі знаходиться крапка, то всі команди, що слідують за нею ігноруються (оскільки програма завершена).

2. В кінці кожної команди ставиться крапка з комою (“;”) - символ, що розділяє команди між собою.
3. При введенні в програму виразів, що містять будь-які дужки потрібно пам’ятати, що кількість закритих та відкритих дужок повинна бути однаковою.
4. Якщо потрібно використати декілька операторів у якості одного складеного оператора, то їх слід взяти в операторні дужки, що починаються словом **begin** і закінчуються **end.** При цьому кількість слів **begin** у програмі повинна співпадати з кількістю слів **end.**
5. Усі змінні, константи та типи даних, що використовуються в програмі, повинні бути описані в розділах **const, type** та **var.**

"Правила хорошого тону"

1. Після команди **begin** всі наступні команди, аж до відповідного йому **end,** бажано записувати з відступом (наприклад, на величину слова **begin).**

Розгалуження

Структурні оператори являють собою угрупування, побудовані з інших операторів за строго визначеними правилами. Усі структурні оператори підрозділяються на три групи:

- складені;
- умовні;
- повтору.

Складений оператор являє собою групу з довільного числа операторів, відділених один від одного крапкою з комою, і обмежену операторними дужками *begin* і *end*.

Формат опису:

begin
< оператор >;
...;
< оператор >;
end;

Складений оператор сприймається, як єдине ціле, і може знаходитися в будь-якому місці програми.

Умовні оператори забезпечують виконання або невиконання деякого оператора, групи операторів або блока в залежності від заданих умов. Використовуються умовні оператори в таких алгоритмах, де можливі декілька варіантів розв'язання задачі в залежності від початкових умов.

Паскаль допускає використання двох умовних операторів: *if* та *case*.

Умовний оператор (команда розгалуження) є одним із самих поширених засобів, що змінюють лінійний порядок виконання операторів програми. Він може приймати одну з наступних форм:

Повна форма:
if *< умова >* ***then***
begin
<серія 1>;
end
else
begin
<серія 2>;
end
Важливо відмітити, що **перед else не ставиться крапка з комою**.
Мовою блок-схем ця команда записується у наступному вигляді:

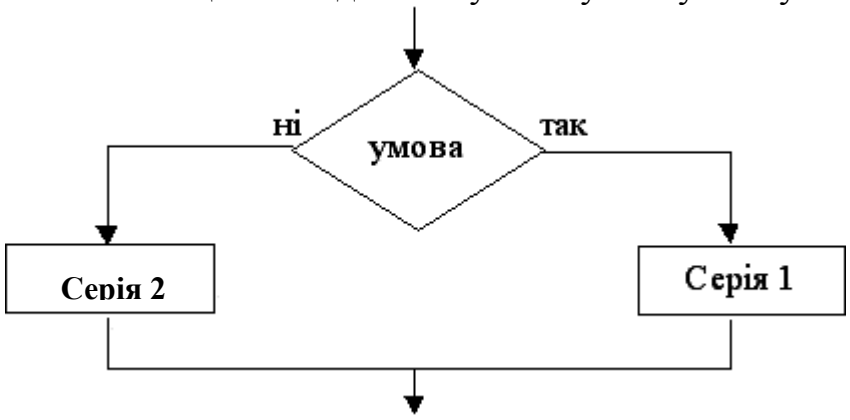


Рис. 7. Повна форма команди розгалуження

Скорочена форма команди розгалуження:
if < умова > **then**
begin
<серія>;
end;
Мовою блок-схем вона записується так:

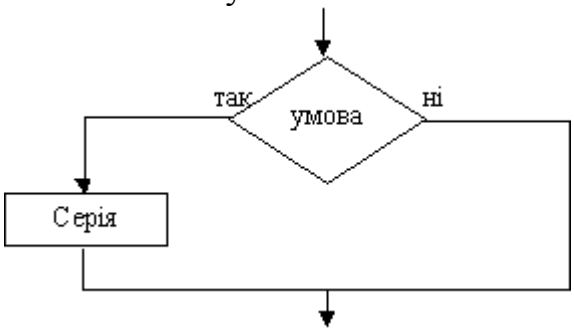


Рис. 8. Скорочена форма команди розгалуження

У повній формі команди умовного переходу виконується *<серія 1>*, якщо умова істинна, і *<серія 2>*, якщо вона хибна. В скороченій формі - серія виконується тільки у випадку, якщо умова істинна, у протилежному випадку команда розгалуження закінчується і виконується наступний за нею оператор.

Один оператор *if* може входити до складу іншого. У цьому випадку говорять про вкладеність операторів. При вкладеності операторів кожне *else* відповідає *then*, що йому передує.

Умова - вираз булівського типу, тобто це речення, на яке можна відповісти "так" або "ні".

Умова може бути простою або складеною. Проста умова – це два арифметичних вирази поєднані між собою операціями відношення. Складені умови утворюються з простих за допомогою логічних операцій *and*, *or*, *not*, причому вочевидь, що службове слово ***or*** використовується, якщо необхідно виконання хоча б однієї умови, а службове слово ***and*** - якщо необхідно одночасне виконання всіх умов.

Наприклад, мама дозволила мені піти погуляти, якщо я "*виконав уроки*" та (англійською ***and***) "*помив посуд*". Тільки при виконанні обох умов мені дозволено піти до друзів.

Другий приклад: якщо сьогодні "*вихідний*" ***або*** (англійською ***or***) "*святковий день*" ***або*** "*канікули*" ***або*** "*я хворий*", я не піду до школи. Виконання хоча б однієї з умов приведе до того, що я залишусь вдома і не піду до школи. Зовсім не обов'язково, щоб одночасно було свято, вихідний, канікули та ще й хвороба, щоб не треба було йти до школи.

Службове слово ***not*** виконує заперечення умови, тобто, якщо вона була істиною, то стає хибною і, навпаки, якщо була хибною - стає істиною. Наприклад, якщо на вулиці ***не*** (англійською ***not***) йде дощ, то я залишу вдома парасольку. В цьому випадку "*йде дощ*" - умова, яка може бути істиною чи хибною в залежності від погоди на вулиці. А слово ***не*** заперечує цій умові, тобто робить її хибною, якщо дощ дійсно йде, і, навпаки, істиною, якщо дощу немає.

При написанні програм на мові Паскаль в записі простих умов можуть використовуватися всі можливі операції відношення (дорівнює, не дорівнює, менше, більше, не більше, не менше тощо). Результат виразу має булівський тип.

Оператор вибору

Цей оператор є узагальненням оператора ***if*** і дозволяє зробити вибір із довільного числа наявних варіантів. Він складається з виразу, що називається селектором, і списку параметрів, кожному з яких передує список констант вибору (список може складатися і з однієї константи). Як і в операторі ***if***, тут може бути присутнім слово ***else***, що має той же зміст.

Формат опису:

case < вираз-селектор > of
< список констант вибору1 > : < оператор 1 >;
< список констант вибору 2 > : < оператор 2 >;
...
< список констант вибору n > : < оператор n >;
else < оператор >
end;

Оператор ***case*** працює наступним чином. Спочатку обчислюється значення виразу-селектора, потім забезпечується реалізація того оператора, константа вибору якого дорівнює поточному значенню селектора. Якщо

жодна з констант не дорівнює поточному значенню селектора, виконується оператор, що знаходиться за словом ***else***. Якщо слово ***else*** відсутнє, активізується оператор, що знаходиться за словом ***end***, тобто перший оператор за межею дії ***case***. Селектор повинен відноситися до одного з перелічувальних типів (цілого, булівського або літерного). Дійсні та рядкові типи використовувати в якості селектора заборонено. Список констант вибору складається з довільної кількості значень або діапазонів, відділених один від одного комами. Межі діапазону записуються двома константами через складений символ діапазону "..". Тип констант у будь-якому випадку повинен збігатися з типом селектора.

Цикл з параметром

Цикл - це послідовність операторів, що виконується деяку мпевну кількість разів. Він використовується при розв'язуванні таких задач, де необхідно повторити деяку послідовність команд більше одного разу.

Відомі два типи команди повторення, що суттєво розрізняються:

- цикл з відомою заздалегідь кількістю повторів;
- цикл з невідомою кількістю повторів.

Щоб це пояснити, розглянемо приклад. Вчителька в першому класі дає дітям завдання: "Діти, напишіть, будь ласка, десять букв "А" та рядочок букв "Б"". Чим відрізняються ці два завдання? В першому випадку зразу ж відомо, скільки разів необхідно повторювати виконання команди "напишіть букву А", а в другому - кількість літер "Б" буде залежати від великої кількості різних факторів: почерку дитини, розміру букв, відстані, що залишається між буквами тощо. Ясно, що в першому випадку ми можемо чітко обумовити кількість повторів, а в другому - необхідно знайти таку умову, перевіряючи яку, дитина зможе закінчити свою роботу.

В мові програмування Паскаль існує три типи циклів: *for*, *repeat* та *while*. Якщо кількість повторів відома заздалегідь, використовується оператор for, якщо кількість повторів невідома, застосовуються оператори *repeat* або *while*.

Цикл з параметром *for*.

Цей оператор повторення інакше називається циклом з лічильником. Він складається із заголовка та тіла циклу та може бути поданий у двох форматах.

1. ***for*** <параметр циклу> := <S1> ***to*** <S2> ***do***
begin <серія>; *end*;
2. ***for*** <параметр циклу> := <S1> ***downto*** <S2> ***do***
begin <серія>; *end*;

де S1 і S2 - вирази, що визначають відповідно початкове і кінцеве значення параметру циклу;

for.. do - заголовок циклу;

<серія> - тіло циклу. Тіло циклу може бути простим або складеним.

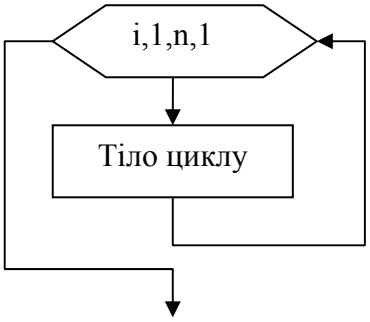


Рис. 9. Цикл з параметром мовою блок-схем

Цей оператор забезпечує виконання тіла циклу доти, поки не будуть перебрані всі значення параметра циклу від початкового до кінцевого.

Параметр циклу, його початкове і кінцеве значення повинні належати до одного типу даних. При цьому можливо використання будь-якого скалярного типу, крім дійсного, тобто цілого (*integer*,*byte*, *word*, *longint*), булівського (*boolean*) та символьного (*char*). Значення параметра циклу послідовно збільшується при *for... to* або зменшується при *for... downto* на одиницю при кожному повторі.

Кількість повторів тіла циклу в операторі for можна визначити за наступною таблицею:

Таблиця 7

Оператор	*S1 < S2*	*S1 = S2*	*S1 > S2*
for...to	S2-S1+1 разів	1 раз	Не виконується
for...downto	Не виконується	1 раз	S1-S2+1 разів

В операторі *for* після *do* може знаходитися складений оператор, в тілі якого ***заборонені оператори, що змінюють значення параметра циклу***. Після нормального завершення циклу значення параметра циклу дорівнює кінцевому значенню.

Цикл може не виконатися зовсім (дивись таблицю), але ніколи не може зациклитись на відміну від наступних двох операторів повторення.

Цикл з передумовою

Цикл з передумовою *while*

У операторі *while* перевірка умови виконання тіла циклу робиться на самому початку оператора.

Формат опису:

while <умова> ***do***

begin <серія>; *end*;

Умова - булівський вираз, а серія - простий або складений оператор. Перед кожним виконанням тіла циклу обчислюється значення виразу умови. Якщо результат являється істинним (*true*), тіло циклу виконується, у протилежному випадку відбувається вихід із циклу і перехід до першого після *while* оператору. Якщо перед першим виконанням циклу значення виразу було хибним (*false*), то тіло циклу взагалі не виконується і відбувається перехід на наступний оператор.

Мовою блок-схем цей цикл описується наступним чином:

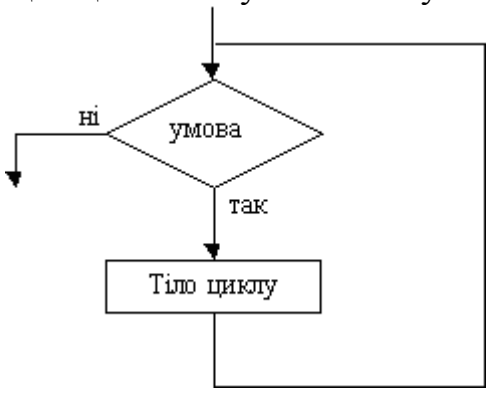


Рис. 10. Цикл з передумовою мовою блок-схем

Програміст сам повинен подбати про переприсвоєння значення змінної, що визначає умову виходу з циклу, інакше він буде нескінченним.

Цикл з післяумовою

Цикл з післяумовою *repeat..until.*

Наступний оператор циклу складається з заголовка (*repeat*), тіла та умови закінчення (*until*).

Формат опису:

repeat

<серія>

until <умова>;

В цьому циклі спочатку виконується серія (тіло циклу), а потім перевіряється умова виходу з циклу. Саме тому ця команда повторення інакше називається циклом *із післяумовою*. Якщо умова виходу з циклу хибна (*false*), цикл активізується ще раз, якщо результат істинний (*true*), відбувається вихід із циклу.

Мовою блок-схем цей цикл описується наступним чином:

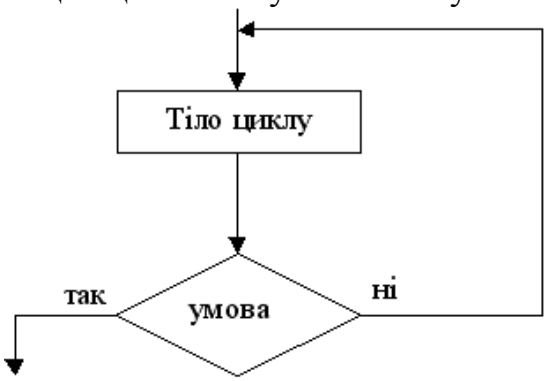


Рис. 11. Цикл з передумовою мовою блок-схем

Цей оператор повтору має наступні характерні риси:

- використовується у випадках, коди користувачу не відома заздалегідь кількість повторень;
- тіло циклу виконується хоча б один раз;
- тіло циклу виконується, поки умова хибна *false*;
- у тілі може знаходитися будь-яка кількість операторів без операторних дужок *begin... end*;

- принаймні один оператор у тілі циклу повинен змінювати значення умови, інакше цикл буде виконуватися нескінченно.

Для передчасного виходу з циклу можна присвоїти параметру циклу значення, що виходить за діапазон дозволених значень.

Порівняння циклу з передумовою, циклу з післяумовою та циклу з лічильником

- В операторі **WHILE** цикл виконується при істинності умови, а цикл **REPEAT** – поки умова хибна. Ще одна відмінна риса – для циклу **REPEAT** при записі в тілі циклу декількох операторів немає необхідності брати їх в операторні дужки begin…end, оскільки цей оператор сам виступає в цій ролі: repeat…until їх замінює.
- В операторі **REPEAT** тіло циклу виконується хоча б 1 раз.
- При використанні як оператору **WHILE**, так і оператору **REPEAT**, потрібно мати на увазі те, що змінні чи вирази, які записані в умові циклу – між словами **WHILE** і **DO** та після **UNTIL**, в тілі циклу повинні змінювати своє значення, в іншому випадку отримаємо **"вічний" цикл**.
- При використанні **циклу з лічильником** значення параметру змінюється автоматично через 1 (або в сторону збільшення, або в сторону зменшення). Отже цикл з лічильником **не може бути вічним**.

Одновимірні масиви

Поняття таблиці виникло тоді, коли програмістам знадобилося запам'ятати та обробити великий набір однотипних даних. Наприклад, якщо ми хочемо знайти середній бал кожного учня класу з інформатики за чверть, нам необхідно знайти суму дуже великої кількості оцінок. Як зберігати всі ці оцінки? Зарезервувати для цього 40 (а може і більше змінних? Це дуже незручно. Ось тут і приходить на допомогу такий структурований тип даних, як таблиця або інакше масив.

Масив - це структура даних, що являє собою однорідну (за типом), фіксовану (за розміром і конфігурацією) сукупність елементів, упорядкованих за номерами.

Таблиця визначається ім'ям (ідентифікатором) і кількістю індексів (номерів), що потрібні для визначення місцезнаходження необхідного елементу масиву. Ім'я масиву є єдиним для всіх його елементів.

В програмуванні кількість індексів таблиці називають його розмірністю, кількість дозволених значень кожного індексу - його діапазоном, а сукупність розмірності та діапазону - формою масиву.

Оскільки конфігурація елементів масиву фіксована, то до окремого елементу можна звертатися за допомогою одного або кількох індексів. У якості індексів можуть використовуватися константи або змінні порядкових типів. Елементами можуть бути як прості змінні будь-яких типів, так і змінні складених типів (масивів, рядків, тощо), тобто може існувати масив масивів, масив рядків і т.і. Глибина вкладеності цих типів - довільна, тому кількість індексів не обмежена, але сумарна довжина структури не повинна

перевищувати дозволені Паскалем 64 Кбайти. В пам'яті ПК всі елементи масиву зберігаються послідовно, тому при переході від молодших до старших адрес першим змінюється самий правий індекс.

Порядок роботи з масивом:

1) оголосити про масив у розділі описів, вказавши його розмір і тип елементів, що в нього входять (тобто приготувати місце в пам'яті, де будуть зберігатися значення елементів);

2) заповнити необхідними значеннями масив для розв'язування задачі;

3) якщо треба, вивести масив на екран для зорової перевірки роботи з ним;

4) робота з даним масивом;

5) виведення отриманих результатів.

Під час розв'язування задач, як правило, використовуються одновимірні та двовимірні масиви. Масиви більшої розмірності на практиці майже не зустрічаються.

Одновимірний масив

Одновимірний масив інакше ще називають лінійним масивом або вектором. Кожному його елементу ставиться у відповідність один індекс.

Масив А

Рис. 12. Одновимірний масив

А[6] або А[i], якщо i=6

Для початку роботи з масивом готуємо місце в пам'яті, для чого описуємо його в розділі оголошень.

Формат опису (1 варіант):

Var <ім'я масиву> : *array* [<розмірність>] *of* <базовий тип елементів>;

Для опису масиву можна використовувати попередньо визначену константу:

Const G1=40;

Var *Mas:array[1..G1] of real;*

Приклади опису:

Const n = 100;

Var A: array[1..n] of real;

B: array[1..100] of integer;

Першим описаний масив, до складу якого входить 100 дійсних чисел, а другим - масив з 100 цілих чисел.

Крім того масив можна описати за допомогою опису відповідного типу.

Формат опису (2 варіант):

Type < ім'я типу > = *array* [<розмірність>] *of* < базовий тип елементів>;

Var <ім'я масиву> : <ім'я типу>;

Приклад:

Type Massiv = array [1..20] of longint;
Var M : Massiv;

Зверніть увагу на те, що значень елементів у масиві не обов'язково буде стільки, скільки ми їх оголосили, але не може бути більше.

Звертання до елементу масиву:

<ІМ'Я_масиву>[<його_індекс>]

Приклад:

M[6] - шостий елемент масиву ***M***;

A[10] - десятий елемент масиву ***A***;

B[i] - і-тий елемент масиву ***B***.

Для роботи з масивом необхідно використовувати будь-який оператор повторення, тому що кожна дія з його елементами виконується однаково.

Паскаль не має засобів введення-виведення усіх елементів масиву водночас, тому введення і виведення значень робиться послідовно окремо кожен елемент.

Методи заповнення одновимірного масиву:

1) за формулою:

```
for i:=1 to n do
M[i]:=i*i-10 {або будь-яка формула};
```

2) з клавіатури:

```
for i:=1 to n do
begin
write('Введіть M[',i,']: ');
readln(M[i]);
end;
```

3) випадковим чином (генератором випадкових чисел) із проміжку [A, B]:

```
for i:=1 to n do
M[i]:=random(B-A)+A;
```

Методи виведення елементів одновимірного масиву на екран

1) виведення у стовпчик:

```
for i:=1 to n do
writeln(M[i]);
```

2) виведення у рядок:

```
for i:=1 to n do write(M[i]:5);
```

При виведенні елементів масиву у рядок бажано зазначити формат виведення, наприклад, *write(M[i]:10:3)* - для дійсних чисел або *write (M[i]:5)* - для цілих.

Дозволяється об'єднувати в одному циклі кілька етапів розв'язування задачі. Наприклад, очищення, заповнення масиву та виведення елементів масиву для контролю на екран.

Двовимірні масиви

Двовимірний масив - це масив, де кожному елементу ставиться у відповідність два індекси.

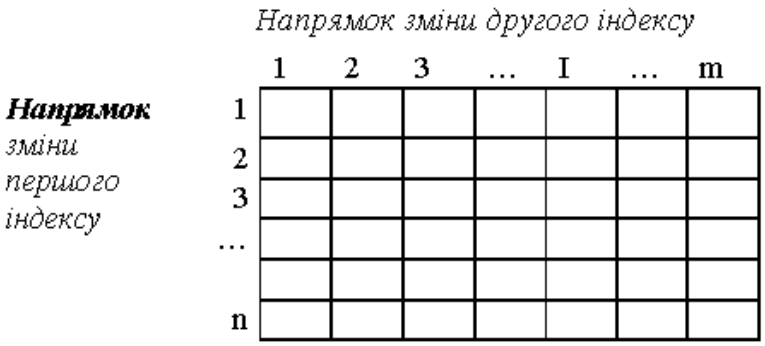


Рис. 13. Двовимірний масив

Для початку роботи з масивом готуємо місце в пам'яті у вигляді прямокутника, що має задану кількість рядків і стовпчиків. Для цього описуємо його в розділі оголошень, використовуючи зарезервоване слово *Array*, після якого в квадратних дужках вказуємо розмірність масиву, причому враховуємо, що на першому місці вказуються індекси рядків, а на другому - стовпчиків, і обов'язково тип елементів.

Опис двовимірного масиву

<Ім'я масиву> : array[<поч інд рядків>..<кін інд рядків>,
<поч інд стовп>. .<кін_інд_стовп>] of <базовий_тип_елементів> ;

Приклад опису:

```
Const n:=100; m:=100;
Var A:array[1..n,1..m] of real;
D:array[1..10,1.100] of integer;
```

Зверніть увагу на те, що значень у рядках або стовпчиках масиву не обов'язково буде стільки, скільки ми оголосили, але не більше.

Звертання до елементу двовимірного масиву:

Ім'я_масиву[<індекс_рядка>, <інд_стовпчика>]

Заповнення масиву:

- з клавіатури:

```
for i:=1 to n do
for j:=1 to m do
begin
write ('введіть A[',i,',',j,']: ');
readln (A[i,j]);
end;
```

- за формулою:

```
for i:=1 to n do
for j:=1 to m do
A[i,j]:=i*i-10 {або будь-яка формула};
```

- випадковим чином із проміжку [K,L]:

```
for i:=1 to n do
for j:=1 to m do
A[i,j]:=random(L-K)+K;
```

Виведення двовимірного масиву на екран

```
for i:=1 to n do
begin
```

```
for j:=1 to m do
write(A[i,j]:8); {виведення в рядок}
writeln; {перехід на новий рядок}
end;
```

Виведення в рядку необхідно обов'язково форматувати, щоб не трапилось "злипання" елементів (дивись приклад вище).

Як було зазначено вище, для роботи з масивом потрібен будь-який оператор повторення. Вочевидь, що у двовимірному масиві необхідно використовувати їх два: один цикл, внутрішній, потрібен для переходу між елементами рядка (тобто, по стовпчиках), а другий, зовнішній, - для переміщення між рядками.

Якщо в матриці кількість рядків і стовпчиків однакова, то таку матрицю називають ***квадратною*** (на відміну від звичайної прямокутної таблиці). Тільки в квадратних матрицях існують головна та бічна діагоналі.

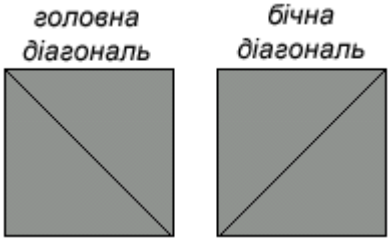


Рис. 14. Головна та бічна діагоналі в квадратній матриці

Елементи, що стоять на головній діагоналі, мають індекси (1, 1), (2, 2), (3, 3), ... (i, i). ..., (n, n), тобто *номер рядка дорівнює номеру стовпчика!* Елементи, що стоять на бічній діагоналі, мають такі індекси (1, n), (2, n-1), (3, n-2), ..., (i, n+1-i), (n,1), тобто *індекси елементів взаємозалежні за формулою* $j= n+1 - i$.

Пошукові алгоритми

Лінійний пошук (послідовний)

```
flag:=false;
for i:=1 to n do
If a[i]=x then flag:=true;
if flag then writeln ('шуканий елемент  знайдено')
        else writeln ('шуканий елемент  не знайдено');
```

Ефективніше використовувати цикл з передумовою

- не всі переглянуті
- не збігання шуканого з поточним a[i]≠ x

```
i:=1;
while (a[i]<>x) and (i<=N) do i:=i+1;
if i<=N then writeln ('шуканий елемент  знайдено')
          else writeln ('шуканий елемент  не знайдено');

a[N+1]:=x;
i:=1;
```

```
while (a[i]<>x) do i:=i+1;
if i<N+1 then writeln ('шуканий елемент  знайдено')
        else writeln ('шуканий елемент  не знайдено');
```

Бінарний пошук

Пошук діленням навпіл

a_i, $i=1,2,...n$

$a_i \le a_i+1$, $i=1,2,...n-1$

```
x, a[m]
[1,N]
a[N+1div2]
L:= (N+1div2)+1,
R:= (N+1div2)-1

L:=1;
R:==N;
flag:=false;
while(L<=R) and not flag do
begin
        m:=(L+R) div 2;
                if a[m]=x
                then flag:=true
                        else if a[m]<x
                        then L:=m+1;
                        else R:=m-1
        end;
if flag then writeln ('шуканий елемент  знайдено')
        else writeln ('шуканий елемент  не знайдено');
if a[m]<x
then L:=m+1
else if a[m]>x
        then R:=m-1
        else flag:=true;

L:=1;
R:==N;
repeat
m:=(L+R) div 2;
        if x>a[m]
        then L:=m+1
                        else if x<a[m]
                        then R:=m-1
until (a[m]=x) or (L=R);
if a[m]=x then writeln ('шуканий елемент  знайдено')
                else writeln ('шуканий елемент  не знайдено');
```

```
L:=1;
R:==N;
while(L<=R) do
begin
        m:=(L+R) div 2;
                if a[m]<x
                        then L:=m+1
                        else R:=m
        end;
if a[R]=x then writeln ('шуканий елемент  знайдено')
        else writeln ('шуканий елемент  не знайдено');
```

Упорядкування масивів

Дуже часто при розв'язуванні задач, пов'язаних з обробкою масивів, необхідно виконувати сортування його елементів за зростанням або спаданням. Такі задачі мають велике практичне значення. Розглянемо деякі з методів, що дозволяють впорядкувати елементи таблиць.

Всі існуючі методи сортування можна поділити на три групи:

обмінні сортування - виконується обмін між собою двох деяких (найчастіше сусідніх) елементів масивів, якщо відповідні елементи розташовані у вихідному масиві невірно; процес повторюється або певну кількість разів, або доки при черговому проході елементи в масиві не будуть переставлятися;

· методи прямого вибору - в масиві вибирається елемент з певними властивостями (наприклад, мінімум або максимум), а потім вибраний елемент становиться на своє місце;

методи прямої вставки - послідовно вибирається елемент з масиву і після визначення його місця у впорядкованому наборі даних вставляється безпосередньо на своє місце.

Найбільш відомим обмінним сортуванням являється метод ***"бульбашки"***. В ньому при послідовному проході по масиву порівнюються два сусідніх елементи. Якщо їх розміщення являється неправильним (наприклад, при впорядкуванні за зростанням лівий елемент більший за правий), виконується взаємообмін елементів. Процес повторюється щонайменше *N-1* разів, де *N* - кількість елементів в масиві.

Простіший алгоритм "бульбашки" має наступний вигляд :

```
Program Bubble; {Сортування за зростанням}
Const N=20;
Var Mas:array[1..N] of integer;
    i,j:integer;
    Rez:integer;
Begin
    For i:=1 to N do
        For j:=1 to N-1 do
            If Mas[j]>Mas[j+1] then
```

```
        Begin {Обмін елементів масиву через третю змінну}
            Rez:=Mas[j];
            Mas[j]:=Mas[j+1];
            Mas[j+1]:=Rez;
        End;
End.
```

Метод можна модифікувати, зменшуючи діапазон сортування після кожного проходу, адже ясно, що після кожного проходу максимальний елемент масиву буде "спливати наверх", тобто займати спочатку останню позицію таблиці, потім передостанню і таке інше (дивись таблицю).

Таблиця 8

Почат. масив	***Прохід 1***	***Прохід 2***	***Прохід 3***	***Прохід 4***	***Прохід 5***	***Прохід 6***	***Прохід 7***	***Прохід 8***
703	**908**	908	908	908	908	908	908	908
765	703	**897**	897	897	897	897	897	897
677	765	703	**765**	765	765	765	765	765
612	677	765	703	703	703	703	703	703
509	612	677	677	677	677	677	677	677
154	509	612	653	653	653	653	653	653
426	154	509	612	612	612	612	612	612
653	426	154	509	**512**	512	512	512	512
275	653	426	154	509	509	509	509	509
897	275	653	426	154	**503**	503	503	503
170	897	275	512	426	154	**426**	426	426
908	170	512	275	503	426	154	**275**	275
061	512	170	503	275	275	275	154	**170**
512	061	503	170	170	170	170	170	154
087	503	061	087	087	087	087	087	087
503	087	087	061	061	061	061	061	061

Програма, що реалізує описаний алгоритм має наступний вигляд:

```
Const N=20; {Сортування за зростанням}
Var Mas:array[1..N] of integer;
    i,j:integer;
    Rez:integer;
Begin
    For i:=1 to N do
        For j:=1 to N-i do
          If Mas[j]>Mas[j+1] then
          Begin {Обмін елементів масиву через третю змінну}
              Rez:=Mas[j];
              Mas[j]:=Mas[j+1];
              Mas[j+1]:=Rez;
        End;
 End.
```

Зверніть увагу, що в цьому алгоритмі у вкладеному циклі, що безпосередньо здійснює порівняння елементів, змінна циклу змінюється за іншим законом, ніж в попередньому випадку: від 1 до *N-i*, де *i* - змінна циклу зовнішньої команди повторення.

Другим методом сортування є ***метод прямого вибору***. Один з його різновидів полягає в тому, що вибирається мінімальний елемент масиву, а потім виконується його обмін з першим елементом таблиці. Після цього перший елемент вважається впорядкованим і процес повторюється для підмасиву, що містить на один елемент менше за початковий, тобто елементи з 2-го до останнього. Процес повторюється кожен раз для масиву, зменшеного на один елемент. Закінчується він тоді, коли невпорядкований підмасив стає довжиною один елемент. Таким чином, загальна кількість повторень дорівнює, як і в попередньому випадку, *N-1* (*N* - кількість елементів масиву).

Програмна реалізація запропонованого методу наведена нижче:

```
Const N=20;
Var Mas:array[1..N] of integer;
    i,j,Min,N_Min:integer;
Begin
    For i:=1 to N-1 do
    Begin
    Min:=Mas[i]; {Зберігання еталону мінімуму}
    N  Min:=i; {Зберігання номера мінімуму}
    For j:=i+1 to N do
    If Mas[j]then
    Begin
    Min:=Mas[j]; {Перевизначення еталону}
    N_Min:=j; {Зберігання номеру еталону}
    End;
    {Обмін місцями мінімуму та першого елементу підмасиву}
    Mas[N_Min]:=Mas[i];
    Mas[i]:=Min;
    End;
End.
```

Зверніть увагу на те, що пошук мінімуму в програмі організований стандартно, тобто перший елемент береться за еталон, а потім порівнюється з усіма останніми *i*, якщо новий елемент являється меншим за еталон, то еталон переприсвоюється. Крім цього в алгоритмі запам'ятовується місце знаходження цього мінімального елемента для того, щоб після виходу з циклу можна було обміняти місцями знайдений мінімум і перший елемент підмасиву. Але так як підмасив весь час змінює свій розмір, за еталон береться перший елемент саме того підмасиву, який розглядається на наступному кроці, тобто *i*-ий елемент початкового масиву (*i* - змінна зовнішнього циклу, що вказує на початок нового підмасиву на кожному кроці).

Метод ***прямої вставки*** забезпечує вставку кожного елементу невпорядкованого масиву на своє місце у вже впорядкований масив. На початку сортування масив розбивається на два підмасиви, лівий з яких повинен бути впорядкованим, а правий - ні. У невідомому масиві тільки один елемент можна вважати впорядкованим, тому спочатку ліва відсортована частина складається всього з одного елементу. Потім по черзі беруться елементи з другої невпорядкованої частини і для них знаходиться місце вставки в першу частину таке, щоб впорядкованість не порушувалась. Це означає, що при сортуванні за зростанням необхідно знайти таке місце в масиві, де лівий елемент буде меншим або рівним за той, що вставляється, а правий - більшим за той, що вставляється. Після цього в масиві необхідно зробити зсув елементів, щоб звільнити місце, на яке і вставити черговий елемент.

Щоб оптимізувати розглянутий алгоритм, можна поєднати зсув елементів з пошуком місця вставляння. Для цього перевірки виконуються в зворотному напрямку від елемента, що потрібно вставити до місця вставки Так як елемент, що вставляється, береться першим з невпорядкованої частини масиву, то ліворуч від нього всі елементи вже впорядковані. Тому фактично необхідно порівнювати даний елемент з усіма лівішими від нього і, якщо даний елемент менший за той, з яким порівнюється, то виконується обмін елементів. Елемент наче "пливе" ліворуч від свого початкового місця розташування і процес цей припиняється, якщо знайдений елемент, не більший за даний, або ми досягли початку масиву. Наприклад, даний такий масив:

12 -8 0 30 5 100

Розбиваємо його на дві частини. До першої входить єдиний впорядкований елемент *{12}*, а до другої - всі останні *{-8 0 30 5 100}*. Запишемо тепер процес впорядкування по етапах:

I етап: елемент, що впорядковується = -8.

1) -8 < 12, тому виконується обмін, тобто після першого кроку масив має наступний вигляд:

-8 12 0 30 5 100

На цьому цикл припиняє свою роботу, тому що досягнуто початок масиву (i=1).

II етап: елемент, що впорядковується = 0.

1) 0 < 12, значить виконується обмін, тобто після першого кроку масив має вигляд:

-8 0 12 30 5 100

2) 0 > -8 , значить обмін не виконується, здійснюється вихід з циклу, масив залишається без змін.

III етап: елемент, що впорядковується = 30.

1) 30 > 12, вхід до циклу не відбувається, масив залишається без змін.

IV етап: елемент, що впорядковується = 5.

1) 5 < 30, виконується обмін, після перестановки масив має вигляд:

-8 0 12 5 30 100

2) 5 < 12, здійснюється обмін, масив набуває вигляду:

-8 0 5 12 30 100

3) 5 > 0, цикл припиняє свою роботу, масив залишається без змін.

V eman: елемент, що впорядковується = 100.

1) 100 > 30, вхід до циклу не відбувається, тому що умова відразу хибна і масив залишається без змін.

Програму, що виконує запропонований алгоритм, наведено далі:

```
Const N=20;
Var Mas:array[1..N] of integer;
    i,j,Rez:integer;
Begin
    For i:=2 to N do
    Begin
        j:=i;
        {Цикл працює доки, лівий елемент більший за
        правий та не досягнуто початок масиву}
        while (j>1) and (Mas[j]<Mas[j-1])  do
        Begin
            Rez:=Mas[j];
            Mas[j]:=Mas[j-1];
            Mas[j-1]:=Rez;
            j:=j-1;
        End;
    End;
End.
```

Графіка

Як Вам вже відомо, будь-який монітор ПЕОМ може працювати в одному з двох режимів:

- текстовому;
- графічному.

Перший з них дозволяє виводити на екран будь-який символ ASCII-таблиці (дивись вище) у визначене знакомісце екрану, що знаходиться на перетині рядка та стовпчика. Кількість знакомісць залежить від текстового режиму, але найчастіше дорівнює 25 рядкам по 80 колонок в кожному. В цьому режимі можна керувати кольором та яскравістю символів, забезпечуючи за бажанням їх миготіння, та кольором тла.

В графічному режимі будь-яке зображення отримується як сукупність різнокольорових точок - пікселів. Кількість елементів зображення теж задається відповідним режимом, але стандартно дорівнює 640 на 480 відповідно по горизонталі та вертикалі.

Для роботи в графічному режимі в Паскалі використовується модуль *Graph*, який складається з більш ніж 90 графічних процедур і функцій широкого профілю. Усі стандартні засоби модуля Graph стають доступними після його підключення до програми в розділі *Uses*:

Uses Graph;

Зверніть увагу на те, що для того, щоб графічна бібліотека стала доступною, необхідно прописати до неї шлях доступу в пункті меню *Options ->Directories ->Unit directories* оболонки Паскаль. За замовчанням цей шлях доступу наступний: BP\UNITS відповідного диску.

Екран у графічному режимі може адресуватися за допомогою системи координат, причому значення *X* (номера стовпчика) збільшується зліва праворуч, а значення *Y* (номера рядка) збільшується зверху до низу. За замовчанням координати екрана мають такий вигляд:

(0,0) - лівий верхній кут;

(639,0) - правий верхній кут;

(319,239) - центр;

(0,479) - лівий нижній кут екрана;

(639,479) - правий нижній кут.

Графічна система підтримує поточний покажчик CP, який виконує ті ж функції, що і курсор, але не виводиться на екран. Для переміщення покажчика використовуються процедури *MoveTo*,*LineTo*, *InitGraph*, *LineRel*, *MoveRel* і деякі інші.

У графічному режимі можна виводити текст, причому є можливість масштабування і вибору типу шрифту, виконання вирівнювання виведеного тексту тощо.

Наявні програми підтримують різні засоби малювання і заповнення фігур, зокрема, крапку, лінії, окружності, еліпси, прямокутники, багатокутники.

Для всіх команд виведення можна встановити графічне вікно (прямокутну область на екрані заданого розміру). Уся графічна інформація виводиться у поточне вікно, поки не активізується інше. При установці віконного режиму всі графічні зображення, якщо вони виходять за межі вікна, усікаються.

При виконанні графічної операції може виникнути помилка, код якої можна одержати за допомогою функції *GraphResult*. Код помилки може приймати одне з наступних значень:

Таблиця 9

0:	Помилки немає
-1:	Режим BGI не встановлений
-2:	Графічні апаратні засоби не виявлені
-3:	Файл драйвера пристрою не знайдений
-4:	Неправильно визначений файл драйвера пристрою
-5:	Не вистачає пам'яті для завантаження драйвера
-6:	Вихід за межі пам'яті при заповненні
-7:	Вихід за межі пам'яті при заливанні
-8:	Файл із шрифтом не знайдений

-9:	Не вистачає пам'яті для завантаження шрифту
-10:	Неправильний графічний режим для обраного драйвера.

Для забезпечення переходу екрану монітора в графічний режим програма повинна починатися викликом процедури *InitGraph*, що автоматично виявляє апаратні засоби і завантажує відповідний графічний драйвер. Стандартний драйвер *EGAVGA.BGI* розміщується у каталозі BP\BGI на відповідному диску. Якщо апаратні засоби не виявлені або в процесі ініціалізації відбулася помилка, то на екран виводиться повідомлення про помилку і програма зупиняється.

До основних функцій для роботи з графікою, за допомогою яких можна виконати майже всі завдання шкільного курсу, відносяться наступні:

Arc(X,Y:integer; поч_кут, кін_кут, радіус:word) - програмна процедура, результатом роботи якої є дуга окружності з центром в точці (*X*,*Y*) і радіусом "радіус". Дуга креслиться від початкового кута ("поч_кут") до кінцевого кута ("кін_кут"). При цьому використовується поточний колір малювання.

Bar(X1,Y1,X2,Y2) - процедура малює зафарбований прямокутник, використовуючи колір зафарбування, що встановлюється процедурою *SetFillStyle*. Контур прямокутника креслиться кольором і типом лінії, що встановлені процедурами *SetLineStyle* і *SetColor*. Точки з координатами (*X1*,*Y1*) та (*X2*, *Y2*) задають дві діагонально протилежні вершини прямокутника.

Bar3D(Xl,Yl,X2,Y2:integer; глибина : word, вершина : boolean) - процедура малює зафарбований тривимірний паралелепіпед. Контур паралелепіпеда креслиться кольором і типом лінії, що встановлені процедурами *SetLineStyle* і *SetColor*, тип і колір зафарбування встановлюється за допомогою процедур *SetFillStyle*. "Глибина" являє собою число елементів зображення, що задають третій вимір тривимірного контуру. Якщо змінна, зазначена як параметр "вершина", приймає істинне значення (*True*), то для паралелепіпеда малюється тривимірна вершина, у протилежному випадку вершина не малюється (що дозволяє малювати кілька паралелепіпедів, розташованих один на одному).

Circle(X,Y:integer; радіус : word) - процедура малює окружність поточним кольором. Точка (*X*,*Y*) - центр окружності, а "радіус" - її радіус.

ClearDevise - процедура очищує поточний графічний екран і підготовлює його для виведення даних.

ClearViewport - процедура очищує поточне вікно.

CloseGraph - процедура припиняє роботу графічної системи (закриття графіки) і повертає монітор до текстового режиму.

DetectGraph(Var драйвер, режим : integer) - процедура перевіряє наявність відповідних апаратних засобів і визначає, який графічний режим і драйвер варто використовувати.

Ellipse(X,Y:integer; поч_кут, кін_кут:word; радX,радY:word) - процедура малює еліптичну дугу, використовуючи (*X*,*Y*), як точку центра і

"*радX*", "*радY*" - як радіуси на горизонтальній і вертикальній осях. Дуга еліпса малюється від початкового кута (параметр "*поч_кут*") до кінцевого кута (параметр "*кін_кут*") поточним кольором.

FillEllipse(X,Y:integer;Xрадіус,Yрадіус:word) - процедура вичерчує зафарбований еліпс, використовуючи точку з координатами (*X*,*Y*), як центр, а "*Xрадіус*" і "*Yрадіус*" - у якості радіусів на горизонтальній та вертикальній осях. Контур еліпса креслиться кольором і типом лінії, що встановлені процедурами *SetLineStyle* і *SetColor*, тип і колір зафарбування встановлюється за допомогою процедури *SetFillStyle*.

FloodFill(X,Y,границя:word) - процедура заповнює замкнену область, використовуючи поточний заповнювач, заданий процедурою *SetFillStyle*. Точка (*X*,*Y*) є внутрішньою точкою області, що зафарбовується. Заповнюється область, обмежена лінією, що має колір, визначений параметром "*границя*". Якщо точка (*X*,*Y*) знаходиться усередині замкненої області, то заповнюється внутрішня область. Якщо ця точка знаходиться поза замкненої області, то заповнюється її зовнішня частина (поле екрана, що не належить області).

GetBkColor : word - функція повертає поточне значення кольору тла (у діапазоні 0 - 15), встановлене процедурою *SetBkColor*.

GetColor : word - функція повертає поточне значення основного кольору малювання (у діапазоні 0 - 15), встановлене раніше процедурою *SetColor*.

GetMaxColor:word; - функція повертає найбільше значення кольору, яке можна передати процедурі *SetColor*.

GetMaxX:integer; - функція повертає максимальний для поточного графічного режиму і драйвера номер правого стовпчика (дозвіл по *X*).

GetMaxY:integer; - функція повертає максимальний для поточного графічного режиму і драйвера номер нижнього рядка екрана (дозвіл по *Y*).

GetPixel(X,Y:mteger): word; - функція повертає значення (колір) елемента зображення в точці (*X*, *Y*).

GetX: integer; - функція повертає *X*-координату поточного покажчика СР.

GetY: integer; - функція повертає *Y*-координату поточного покажчика СР.

GraphResult : integer; - функція повертає код помилки (у діапазоні 0-10) для останньої графічної операції.

InitGraph (Var гр_драйв:integer; Var гр_режим:integer; путь_драйв:string); - процедура ініціалізує графічну систему і переводить апаратну частину в графічний режим.

Line(X1, Y1, X2, Y2 : integer); - процедура вичерчує пряму лінію (товщина і тип якої встановлений процедурою *SetLineStyle*, колір - процедурою *SetColor*) із крапки (*X1*,*Y1*) у крапку (*X2*,*Y2*).

LineRel(Dx,Dy); - процедура вичерчує пряму лінію з точки поточного покажчика СР в точку, задану відносною відстанню (*Dx*, *Dy*) від поточного покажчика.

LineTo(X,Y : integer); - процедура малює пряму лінію з точки, у якій знаходиться поточний покажчик СР, у точку з координатами (*X*,*Y*).

MoveRel(Dx, Dy : integer); - процедура переміщує покажчик СР з поточної точки у точку, задану відносною відстанню (*Dx*, *Dy*).

MoveTo(X, Y : integer); - процедура переміщує поточний покажчик СР у точку з координатами (*X*,*Y*).

OutText (рядок : string); - процедура виводить текст "рядка" на монітор, починаючи з точки розташування покажчика СР.

OutTextXY(X, Y: integer; текст_рядок : string); - процедура виводить текст, що міститься у "текст_рядок", починаючи з точки, заданої координатами (*X*, *Y*). Якщо рядок занадто довгий і виходить за межі екрана чи поточної області перегляду, то він усікається.

PieSlice (X, Y: integer; нач_кут, кін_кут R : word); - процедура вичерчує і заповнює поточним кольором сектор кола радіусом *R*. Точка (*X*,*Y*) - центр окружності, а сектор малюється від початкового до кінцевого кута. Тип і колір зафарбування попередньо задається процедурою *SetFillStyle*.

PutPixel(X, Y: integer; ел_зображ : word); - процедура зафарбовує точку з координатами (*X*,*Y*) у колір, що визначається параметром "ел_зображ".

Rectangle(Xl, Yl, X2, Y2 : integer); - процедура вичерчує прямокутник, використовуючи поточний колір і тип лінії. (*X1*,*Y1*) та (*X2*,*Y2*) - координати діагонально протилежних вершин прямокутника.

Sector (X, У : integer; поч__кут, кін_кут, X_Радіус, Y_Радіус : word); - процедура вичерчує і заповнює еліптичний сектор. (*X*,*Y*) - центр окружності, "*X_Радіус*", "*Y_Радіус*" - горизонтальний і вертикальний радіуси. Сектор креслиться від початкового кута "*поч_кут*" до кінцевого кута "*кін_кут*". Сектор малюється поточним кольором і зафарбовується з використанням зразка зафарбування і кольорів, заданих за допомогою процедури *SetFillStyle*.

SetActivePage (сторінка: word); - процедура встановлює для графічного виведення активну сторінку (їх всього чотири від 0 до 3). Графічне виведення буде тепер здійснюватися в цю сторінку.

SetBkColor (колір : word); - процедура встановлює поточний колір для тла.

SelColor(колір : word); - процедура встановлює поточний колір малювання .

SetFillStyle(зразок : word; колір :р word); - процедура встановлює зразок і колір зафарбування для всіх типів зафарбування, виконуваних процедурами *Bar*, *Bar3D*, *FillEllipse*, *FloodFill*та *PieSlice*. Можна використовувати декілька типів зафарбування (наприклад, 1 - суцільне фарбування, 2 - штрихування лініями, 7 - штрихування символом "+", 11 - штрихування крапками тощо). За замовчуванням вибирається суцільне зафарбування білого кольору.

SetLineStyle(тип_рядка :word; зразок: word; товщина : word); - процедура встановлює поточну товщину і тип лінії.

SetTextJustify(гориз, верт : word); - процедура встановлює значення вирівнювання тексту, що використовуються процедурами *OutText* і *OutTextXY*.

SetTextStyle (шрифт: word; направл: word; розм_символу: CharSize-type); - процедура встановлює поточний шрифт, тип і коефіцієнт розміру символу.

SetViewport(XI, Y1, X1, X2 word, clip: boolean); - процедура встановлює для графічного поточного виведення чи перегляду вікно, де (*X1*,*Y1*) - верхній лівий кут області перегляду, (*X2*,*Y2*) - нижній правий кут. Процедура переміщує поточний покажчик у точку з координатами (0,0). *Clip* - булівська змінна.

SetVisualPage (сторінка : word); - процедура задає номер відображуваної графічної сторінки.

TextHeight(текст_рядок: string) : word; - функція повертає висоту рядка в елементах зображення.

TextWidth(текст_рядок : string) : word; - функція повертає ширину рядка в елементах зображення.

Літерні величини

Крім чисел, мови програмування можуть оперувати з окремими словами, наборами слів (реченнями, текстами). При цьому слова та їх набори необов'язково повинні мати смисл з точки зору людини – це можуть бути будь-які послідовності символів.

Основні означення

Під *рядковою величиною* розуміють значення константи (змінної), яке складається з будь-якої послідовності символів кодової таблиці ЕОМ. Для такої величини кількість символів може бути від 0 до 255.

Під *символьною величиною* розуміють значення константи (змінної), яке складається з будь-якого, але тільки одного символу кодової таблиці комп'ютера.

Над рядковими та символьними величинами можна виконувати такі операції як склеювання, визначення довжини величини, виділення (вирізка) частини символів з цілої послідовності тощо.

Опис змінних

Символьні величини описуються як тип даних Char. У програмі значення змінних і констант цього типу мають бути поміщені у апострофи, наприклад:

```
Var s: char;
... S:='A';  ...
```

Рядкові величини описуються як тип даних String. Рядок по іншому можна розглянути як масив, компоненти якого мають тип char і тип індексу має нижню межу рівну 1, тобто тип String описується як array [1..255] of char. При використанні у програмі значення рядкових констант та змінних поміщаються у апострофи, наприклад:

```
Var n : string;
```

... n:='Шевченко Тарас Григорович'.

Серед усіх можливих значень рядків є порожній рядок. Він зображується двома одинарними лапками, між якими нічого немає.

Для того, щоб символ "апостроф" входи до складу рядка цей символ повторюють двічі. Наприклад,

Write('ім''я'); виведе на екран рядок ім'я.

Основні операції для роботи з літерними величинами

Для роботи з літерними величинами у мові Паскаль передбачено наступні операції, процедури та функції :

- ***Length(st)*** – визначає довжину літерної величини st. Іншими слова, просто підраховує кількість символів в літерній величині. Цю ж інформацію можна отримати, зчитавши значення st[0], але будьте уважні, необхідно на друк виводити значення не st[0], а ord(st[0])!
- ***Copy(st, m, n)*** – виділяє з літерної величини st n символів, починаючи з символу під номером m. Дана функція використовується для копіювання.
- ***Concat(st1, st2)*** – об'єднує дві літерні величини. Дана функція виконує ту ж саму дію, що й операція '+', введена вона для сумісності з більш ранніми версіями мови Паскаль.
- ***Pos(s1:string, s:string)*** : byte – повертає позицію, з якої рядок S1 перший раз зустрічається у рядку S.
- ***Delete (st : string, poz : integer, n : integer)*** – з рядка st вилучає n символів, починаючи з позиції poz.
- ***Insert (s1 : string, s : string, poz : integer)*** – вставляє рядок s1 у рядок s, починаючи з позиції poz
- ***Str (x,st)*** – перетворює числове значення величини x у рядок st. Після x може записуватись формат, аналогічний формату виводу оператора write
- ***Val(st, m, kod)*** – перетворює літерну величину st в число m. Якщо kod <> 0, то введена літерна величина не є числом.

Процедури

Підпрограмою називається іменована, логічно закінчена група операторів мови, яку можна викликати для виконання будь-яку кількість разів із різних місць програми.

У мові Паскаль для організації підпрограм використовуються процедури і функції.

Процедура - це незалежна поіменована частина програми, призначена для виконання визначених дій. Вона складається з тіла і заголовка. За структурою її можна розглядати як програму в мініатюрі. Після однократного опису процедуру дозволяється викликати за іменем з наступних частин програми. Використання імені процедури в програмі називається викликом процедури. Ім'я процедури не може знаходитися у виразі у якості операнду.

Усі процедури і функції мови Паскаль підрозділяються на дві групи:

вбудовані;

визначені користувачем.

Вбудовані (стандартні) процедури є частиною мови і можуть викликатися за іменем без попереднього опису в розділі описового блока. З багатьма з них ви вже знайомилися в попередніх розділах.

Процедури користувача організовуються самим програмістом відповідно до синтаксису мови і являють собою локальні блоки. Попередній опис процедур і функцій користувача є обов'язковим.

Опис **процедури** включає заголовок (ім'я) і тіло процедури. Заголовок складається з зарезервованого слова **Procedure**, ідентифікатора (імені) процедури і необов'язкового списку формальних параметрів із вказівкою їх типу, який укладається в круглі дужки .

Формат опису:

Procedure <ім'я> [(формальні параметри)];

Приклад:

Procedure Korrekt;

Procedure Sort (A:byte);

Ім'я процедури - ідентифікатор, унікальний у межах програми. Тіло процедури являє собою локальний блок, за структурою аналогічний програмі:

Procedure <ім'я> [(формальні параметри)];

[<розділи описів>;]

begin

<розділи операторів>

end;

Зверніть увагу, що як формальні параметри, так і розділ описів у процедурі може бути відсутній.

Щоб звернутися до процедури, треба використати оператор виклику процедури. Він складається з ідентифікатора (імені) процедури і списку фактичних параметрів, що відділені один від одного комами і знаходяться у круглих дужках. Якщо процедурі не передається ніяких параметрів, то фактичні параметри не вказуються.

Формат виклику процедури:

<ідентифікатор> [(фактичні параметри)];

Параметри забезпечують механізм заміни, який дозволяє виконувати процедуру з різними початковими даними. Між фактичними параметрами в операторі виклику процедури і формальними параметрами у заголовку опису процедури встановлюється взаємо-однозначна відповідність у результаті їхнього перебору зліва направо. Фактичні параметри за кількістю і типами повинні дорівнювати кількості і типам формальних параметрів.

Параметри, за допомогою яких здійснюється обмін значеннями змінних між підпрограмами та програмою, що їх викликає, можуть мати будь-який тип, в тому числі структурований. Існують два типи параметрів:

параметр-значення;

параметр-змінна.

Група параметрів, перед якими відсутнє зарезервоване слово *Var*, називається **параметрами-значеннями**.

Наприклад, в описі *Procedure Korrect(S,K:real)* *S* і *K* - параметри-значення. Формальний параметр-значення обробляється, як локальна стосовно процедури або функції, змінна. Зміни формальних параметрів-значень не впливають на відповідні значення фактичних параметрів.

Група параметрів, перед якими знаходиться ключове слово *Var*, називається **параметрами-змінними.** Наприклад, в описі *Procedure Obr(Var A,B:integer);* *A* та *B* - параметри-змінні. Параметр-змінна використовується в тому випадку, якщо значення повинно бути передане з процедури в блок, що її викликає. При активізації процедури або функції формальний параметр-змінна заміщується фактичною змінною, а тому будь-які зміни в значенні формального параметра-змінної відбиваються на фактичному параметрі.

І в тому, і в іншому випадку тип фактичного параметра повинен збігатися з типом формального. Якщо формальний параметр має рядковий тип, йому надається атрибут довжини, рівний 255, а тому і фактичний параметр в цьому випадку повинен також мати рядковий тип з атрибутом довжини, що дорівнює 255. У якості параметра-змінної може використовуватися будь-який тип, в тому числі файловий.

Область дії ідентифікаторів.

Для правильного визначення області дії ідентифікаторів при використанні в програмі процедур і функцій необхідно притримуватися наступних правил:

1. Кожний ідентифікатор повинен бути описаний перед тим, як він буде використаний.

2. Ідентифікатор діє у межах блоку, в якому він описаний.

3. Всі ідентифікатори в одному блоці повинні бути унікальними, тобто не повторюватися.

4. Однакові ідентифікатори можуть бути по-різному визначені у кожному окремому блоці, але це вважається поганим стилем програмування і тому не рекомендується в різних блоках програми використовувати змінні з однаковими іменами..

5. Якщо ідентифікатор підпрограми користувача збігається з ім'ям стандартної процедури або функції, то вони стають недоступними в межах області дії підпрограми, оголошеної користувачем, тобто стандартна функція ігнорується, а виконується програма користувача.

Функції

Функція відрізняється від процедури тим, що, по-перше, передає в точку виклику скалярне значення (результат своєї роботи), а по-друге, ім'я функції може входити у вирази, як операнд. Функція, якщо вона зустрічається у виразі, називається покажчиком функції або звертанням до функції.

Усі процедури і функції мови Паскаль підрозділяються на дві групи:

- вбудовані;
- визначені користувачем.

Вбудовані (стандартні) процедури і функції є частиною мови і можуть викликатися за іменем без попереднього опису в розділі описового блока. З багатьма з них ви вже знайомилися в попередніх розділах.

Процедури і функції користувача організовуються самим програмістом відповідно до синтаксису мови і являють собою локальні блоки. Попередній опис процедур і функцій користувача є обов'язковим.

Функція, визначена користувачем, складається з заголовка і тіла функції. Заголовок містить зарезервоване слово **Function**, ідентифікатор (ім'я) функції та, укладений у круглі дужки, необов'язковий список формальних параметрів і тип значення, що повертається функцією.

Формат опису:

Function <ім'я> [(формальні параметри)]:<тип результату>;

Ім'я функції - унікальний у межах блока ідентифікатор. Результат, що повертається, може мати будь-який простий тип і тип *string*.

Тіло функції являє собою локальний блок, за структурою аналогічний програмі:

Function <ім'я> [(формальні параметри)]:<тип результату>;
[<розділи описів>;]
begin
<розділ операторів>
end;

У розділі операторів повинен перебувати хоча б один оператор, що присвоює ідентифікатору функції значення. Якщо таких операторів декілька, то результатом виконання функції буде значення останнього оператора присвоювання.

Звертання до функції здійснюється за іменем з необов'язковою вказівкою списку аргументів. Кожен аргумент повинен відповідати формальним параметрам, зазначеним у заголовку, і мати той же тип.

Формат звертання:

Y:=<ідентифікатор функції > [(фактичні параметри)];

Функції можуть повертати значення цілих, дійсних, булівских, літерних і рядкових типів.

Параметри, за допомогою яких здійснюється обмін значеннями змінних між підпрограмами та програмою, що їх викликає, можуть мати будь-який тип, в тому числі структурований. Існують два типи параметрів: параметр-значення; параметр-змінна.

Група параметрів, перед якими відсутнє зарезервоване слово *Var*, називається **параметрами-значеннями**.

Наприклад, в описі *Procedure Korrect(S,K:real)* *S* і *K* - параметри-значення. Формальний параметр-значення обробляється, як локальна стосовно процедури або функції, змінна. Зміни формальних параметрів-значень не впливають на відповідні значення фактичних параметрів.

Група параметрів, перед якими знаходиться ключове слово *Var*, називається **параметрами-змінними.** Наприклад, в описі *Procedure Obr(Var A,B:integer); A* та *B* - параметри-змінні. Параметр-змінна використовується в тому випадку, якщо значення повинно бути передане з процедури в блок, що її викликає. При активізації процедури або функції формальний параметр-змінна заміщується фактичною змінною, а тому будь-які зміни в значенні формального параметра-змінної відбиваються на фактичному параметрі.

І в тому, і в іншому випадку тип фактичного параметра повинен збігатися з типом формального. Якщо формальний параметр має рядковий тип, йому надається атрибут довжини, рівний 255, а тому і фактичний параметр в цьому випадку повинен також мати рядковий тип з атрибутом довжини, що дорівнює 255. У якості параметра-змінної може використовуватися будь-який тип, в тому числі файловий.

Область дії ідентифікаторів.

Для правильного визначення області дії ідентифікаторів при використанні в програмі процедур і функцій необхідно притримуватися наступних правил:

1. Кожний ідентифікатор повинен бути описаний перед тим, як він буде використаний.

2. Ідентифікатор діє у межах блоку, в якому він описаний.

3. Всі ідентифікатори в одному блоці повинні бути унікальними, тобто не повторюватися.

4. Однакові ідентифікатори можуть бути по-різному визначені у кожному окремому блоці, але це вважається поганим стилем програмування і тому не рекомендується в різних блоках програми використовувати змінні з однаковими іменами..

5. Якщо ідентифікатор підпрограми користувача збігається з ім'ям стандартної процедури або функції, то вони стають недоступними в межах області дії підпрограми, оголошеної користувачем, тобто стандартна функція ігнорується, а виконується програма користувача.

Рекурсія

Рекурсією називається така ситуація, коли підпрограма (процедура або функція) викликає сама себе.

Типова конструкція рекурсивної процедури повинна мати наступний вигляд:

Procedure *Rec (t:integer);*
Begin
<Дії на початку рекурсії>
If *<Перевірка умови>* ***Then*** *Rec(t+1);*
<Дії на виході з рекурсії>
End*;*

Саме головне при написанні рекурсії усвідомити, як правильно задати умову, яка буде заглиблювати нас у рекурсію або, навпаки, виводити з неї.

Розглянемо приклад рекурсивного виклику на обчисленні степеня числа x^n, де x - будь-яке дійсне число, а n - ціле, додатне число. Очевидно, що

$$x^n = x^{n-1} \cdot x$$

$$x^{n-1} = x^{n-2} \cdot x$$

…

Постає питання, коли і як припинити цей процес? В даному випадку зупинкою для обчислень буде обчислення x0, так як відомо, що нульова степінь будь-якого числа дорівнює 1. Тобто рекурсивна функція для обчислення степеня числа буде мати наступний вигляд:

```
Function Step(x:real; n:integer):real:
Begin
If n = 0
then Step:=1
else Step:=Step(x,n-1)*x;
End;
```

Проаналізуємо роботу цієї функції на прикладі знаходження значення 23. При першому виклику функції значення змінних буде дорівнювати відповідно:

x = 2, n = 3.

Так як значення n не дорівнює 0 спрацює гілка else, тобто почне виконуватись такий оператор *Step:=Step(x,n-1)*x;* де x буде дорівнювати 2, і n - теж 2.

Цей оператор містить виклик функції step, тому виконання підпрограми перерветься і виконається виклик тієї ж самої функції *step*, але з параметрами *x=2 та n=1*. При виконанні повторного виклику функції ситуація повториться, так як *n* поки ще не дорівнює 0 і тільки на четвертому виклику функції *step* параметр *n* досягне нульового значення, після чого спрацює гілка *then,* яка присвоїть значення функції 1.

В даній найпростішій рекурсії ніякі дії при вході в рекурсію не відбуваються, але при виході з рекурсії необхідно знати точки, з яких здійснювався вхід в чергову функцію, щоб мати можливість повернутися в них. Ці точки виклику запам'ятовуються у спеціалізованій пам'яті, що зветься стек, і потім по черзі являються тими точками повернення, які використовує система при зворотному русі з рекурсії.

Очевидно, що виконання рекурсії є досить складним процесом, який крім того вимагає суттєвих витрат додаткових ресурсів (що найменше пам'яті). Тому використання рекурсії не рекомендується в тих випадках, коли вона просто замінюється ітеративним процесом (як в наведеному прикладі). Однак, існує велика кількість досить складних алгоритмів, які без використання рекурсії мають дуже заплутану логіку роботи.

Рекурсивні пошукові алгоритми

1 2 2 3 4 4 4 5 6 7

a[k+1]-a[k]=1

(k<l) a[l] – a[k]=l-k

```
        L:=1;
        R:=N;
        while(L<=R) do
        begin
                m:=(L+R) div 2;
                        if a[m]-a[L]=m-L
                                then L:=m+1
                                else R:=m
                end;
        if a[R]=x then writeln ('шуканий елемент  знайдено')
                else writeln ('шуканий елемент  не знайдено');

procedure Bin_Recur (L,R:byte);
        var m:byte;
        begin
                if L+1<>R
                        then
                        begin
                        m:=(L+R) div 2;
                                if a[m]-a[L]<>m-L
                                        then Bin_Recur (L,m);
                                if a[R]-a[m]<>R-m
                                        then Bin_Recur (m,R);
end
```

Робота з файлами

Файл – деяка інформація, що міститься у певному місці, та яка може мати певне ім’я. У випадку файлу, який розміщено на диску, це поняття можна було б подати у дещо іншому вигляді: **файл** – частина дискового простору (зовнішнього запам’ятовуючого пристрою), яка заповнена даними і має ім’я, що складене за певними правилами. У мові Паскаль розрізняють наступні ***види*** файлів: типізовані, текстові і нетипізовані. Слід відмітити, що у мові Паскаль файлова система найбільш повно використовує можливості операційної системи по передачі та обробці даних.

***Примітка*:** ми будемо розглядати і використовувати текстові файли (тип **text** у розділі оголошення змінних).

Команди для роботи з файлами:

Assign(f, ‘Name’); – файлу ставиться у відповідність файлова змінна певного типу, тому перед початком роботи з файлом необхідно обов’язково встановити дану відповідність; де f – змінна довільного файлового типу (ми будемо розглядати і використовувати тип **text** – текстовий файл), а Name –

повне ім’я файлу, що задовольняє вимогам операційної системи. Якщо файл розміщено не в поточному каталозі, то слід вказувати ім’я файлу разом з повним шляхом до нього.

Для роботи з файлом перед усе його потрібно відкрити.

Reset(f); – відкриває існуючий файл для зчитування даних.

Rewrite(f);– створює і відкриває новий файл для запису.

***Примітка**:* У описі обох процедур параметр f означає файлову змінну довільного типу. Зауважимо, що відкриття неіснуючого файлу може призвести до помилки при виконанні програми.

Close(f); – закриває файл; операція закриття файлу є логічним закінченням роботи з довільним відкритим файлом.

***Примітка*:** Використання вказаної процедури призводить до закриття файлу, якому відповідає файлова змінна f і далі для того, щоб знову можна було працювати з цим файлом, над ним потрібно знову провести описані вище дії: поставити у відповідність файловій змінній ім’я файлу і відкрити його.

Erase(f) – видаляє невідкритий зовнішній файл, якому поставлено у відповідність файлову змінну f.

***Примітка*:** Наголошуємо, що файл, який видаляється, не повинен бути перед цим відкритий жодною з відповідних процедур відкриття файла.

Eof(f) – дана функція повертає булівське значення True, якщо вказівник кінця файлу знаходиться відразу ж за останнім компонентом і False у противному випадку. Іншими словами, дана функція повідомляє, чи досягли ми кінця файлу, чи ні.

Eoln(f) - дана функція повідомляє про досягнення кінця рядка.

Append(f); - відкриває вже існуючий файл і встановлює вказівник у кінець файлу, тобто, використавши дану процедуру ми маємо змогу дописувати інформацію у кінець файлу.

Множини. Записи

У мові Паскаль множина – це довільна сукупність значень перерахованого типу. Тип множини описується наступним чином:

type <ім’я типу> = set of <тип елементів>;

Операції над множинами у мові програмування Pascal

- **in** – належність елементу множині (результат – логічний).

Звичайно використовують при перевірці умови належності множині, наприклад, нехай задано множину цифр:

cifra : set of char;

а в самій програмі на початку цю множину конкретно визначено:

cifra := [‘0’..‘9’];

Тоді перевірку належності введеного символу ch типу char на предмет належності множині цифр можна оформити як:

...*Readln(ch);*

*if ch in cifra then write(‘ Належить цифрам ’);****...***

- + – об’єднання множин (результат – множина). Якщо, наприклад множина A={1,2,3,4,5}, а B={4,5,6,7}, то в результаті виконання операції C=A+B ми отримаємо C={1,2,3,4,5,6,7}.
- – – різниця множин (результат – множина). Якщо, наприклад множина A={1,2,3,4,5}, а B={4,5,6,7}, то в результаті виконання операції C=A–B ми отримаємо C={1,2,3}.
- * – переріз множин (результат – множина). Якщо, наприклад множина A={1,2,3,4,5}, а B={4,5,6,7}, то в результаті виконання операції C=A*B ми отримаємо C={4,5}.
- = – рівність множин (результат – логічний).
- <> – не рівність множин (результат – логічний).
- <= – є підмножиною (результат – логічний).
- >= – включає підмножину (результат – логічний).
- [] – порожняя множина.

Запис – це структурований тип даних, який об’єднує в собі різнотипні елементи, які називають полями запису. Запис описується спеціальною конструкцією, яка розпочинається словом Record.

Доступ до компоненти:

в програмі слідом за іменем запису ставиться крапка, а потім ім’я відповідного поля.

Запис може:

- входити у склад даних більш складної структури
- бути полем іншого запису.

Наприклад, для ведення власного бібліотечного каталогу можна було б використати таку конструкцію:

```
Type                    { розділ опису типів}
    Book = Record           { початок запису }
        AuthorFamile : string[20];      { прізвище автора }
        AuthorName : string[15];            { ім’я автора }
        Title : string[90];         { назва книги }
        Date : word;                { рік видання }
        Page : integer;             { кількість сторінок }
        Klass : byte;         {для визначення тематики книги }
    End;        { кінець запису }
```

Структури даних

Програма=алгоритм+структури даних (Н.Вірт)

В програмуванні та комп'ютерних науках ***структури даних*** – це способи організації даних в комп'ютерах. Часто разом зі структурою даних пов'язується і специфічний перелік операцій, які можуть бути виконаними над даними, організованими в таку структуру.

Структура даних – це спосіб подання інформації.

Під ***СТРУКТУРОЮ ДАНИХ*** в загальному випадку розуміють безліч елементів даних і безліч зв'язків між ними.

Правильний підбір структур даних є надзвичайно важливим для ефективного функціонування відповідних алгоритмів їх обробки. Добре побудовані структури даних дозволяють оптимізувати використання машинного часу та пам'яті комп'ютера для виконання найбільш критичних операцій.

Відома формула "Програма = Алгоритми + Структури даних" дуже точно виражає необхідність відповідального ставлення до такого підбору. Тому іноді навіть не обраний алгоритм для обробки масиву даних визначає вибір тої чи іншої структури даних для їх збереження, а навпаки.

Підтримка базових структури даних, які використовуються в програмуванні, включена в комплекти стандартних бібліотек найбільш розповсюджених мов програмування, такиї як Standart Template Library для C++, Java API, Microsoft .NET.

Класифікація структур даних відбувається за різними ознаками:

1. Простими структурами даних називаються такі, які не можуть бути розчленовані на складові частини, більші, ніж біт.

   Структурованими називаються такі структури даних, складовими частинами яких є інші структури – прості або, у свою чергу, структуровані.
2. Залежно від відсутності або наявності явно заданих зв'язків між елементами даних прийнято розрізняти незв'язні і зв'язні структури.
3. Мінливість структури даних – зміна числа елементів і (або) зв'язків між елементами структури. По ознаці мінливості розрізняють структури статичні, напівстатичні і динамічні.
4. Важлива ознака структури даних - характер впорядкованості її елементів. По цій ознаці структури можна ділити на лінійні і нелінійні структури.

Існує така класифікація структур даних:

1. Прості
- Числові
- Символьні
- Логічні
- Перелічувальні
- Інтервальні
- Показчики

2. Статичні
- Масиви
- Множини
- Записи
- Таблиці

3. Напівстатичні
- Стеки
- Черги
- Деки
- Рядки

4. Динамічні

- Зв‘язні списки
- Графи
- Дерева

5. Файлові

- Послідовного доступу
- Прямого доступу
- Комбінованого доступу

Розглянемо найпоширеніші структури даних:

1. Елементарні структури даних

1.1. Масив

В програмуванні масив (англ. array) — одна з найпростіших структур даних, сукупність елементів переважно одного типу даних, впорядкованих за індексами, які зазвичай репрезентовані натуральними числами, що визначають положення елемента в масиві.

Масив може бути одновимірним (вектором), та багатовимірним (наприклад, двовимірною таблицею), тобто таким, де індексом є не одне число, а кортеж (сукупність) з декількох чисел, кількість яких співпадає з розмірністю масива.

В переважній більшості мов програмування масив є стандартною вбудованою структурою даних.

Переваги та недоліки

- **Ефективність операцій**

Масиви ефективні при звертанні до довільного елементу, яке відбувається за постійний час (O(1)), однак такі операції як додавання та видалення елементу, потребують часу O(n), де n — розмір масиву. Тому масиви переважно використовуються для зберігання даних, до елементів яких відбувається довільний доступ без додавання або видалення нових елементів, тоді як для алгоритмів с інтенсивними операціями додавання та видалення, ефективнішими є зв'язані списки.

- **Збереження в пам'яті**

Інша перевага масивів, яка є досить важливою — це можливість компактного збереження послідовності їх елементів в локальній області пам'яті (що не завжди вдається, наприклад, для зв'язаних списків), що дозволяє ефективно виконувати операції з послідовного обходу елементів таких масивів.

Масиви є дуже економною щодо пам'яті структурою даних. Для збереження 100 цілих чисел в масиві необхідно рівно в 100 разів більше пам'яті, ніж для збереження одного числа (плюс, можливо, ще декілька байтів). В той же час, усі структури даних, які базуються на вказівниках, потребують додаткової пам'яті для збереження самих вказівників разом з даними. Однак, операції з фіксованими масивами ускладнюються тоді, коли виникає необхідність додавання нових елементів у вже заповнений масив. Тоді його необхідно розширювати, що не завжди можливо і для таких задач слід використовувати зв'язані списки, або динамічні масиви.

- **Індекси в масивах**

У випадках, коли розмір масиву є досить великий та використання звичайного звертання за індексом стає проблематичним, або великий відсоток його комірок не використовується, слід звертатись до асоціативних масивів, де проблема індексування великих об'ємів інформації вирішується більш оптимально.

З тої причини, що масиви мають фіксовану довжину, слід дуже обережно ставитись до процедури звертання до елементів за їхнім індексом, тому що намагання звернутись до елементу, індекс якого перевищує розмір такого масива (наприклад, до елементу з індексом 6 в масиві з 5 елементів), може призвести до непередбачуваних наслідків.

Слід також бути уважним щодо принципів нумерації елементів масиву, яка в одних мовах програмування може починатись з 0, а в інших — з 1.

1.2 Лінійний список.

Лінійний список в інформатиці та програмуванні визначається як екземпляр абстрактного типу даних, що формалізує концепцію впорядкованої множини елементів. Наприклад, абстрактний тип даних для безтипових, змінних списків можна визначити із допомогою конструктора та чотирьох операцій:

- конструктор для створення порожнього списка;
- операція визначення порожності списка;
- операція для додавання елемента в початок списка (cons в Лісп);
- операція отримання першого елемента списка (або «голови») списка (car в Лісп);
- операція для визначення списка, що складається із всіх елементів списка окрім першого (або його «хвоста») (cdr в Лісп).

Характеристики

За визначенням, список — це послідовність з $n \geq 0$ елементів X[1],X[2], … X[n], для якої виконується наступна умова: якщо $n>0$ та X[1] — перший елемент у списку, а X[n] — останній, то k-й елемент розташований між X[k-1] та X[k+1] елементами для усіх $1<k<n$.

З такими структурами даних виконуються наступні операції:

- отримання k-го елемента списку для читання чи запису в нього нового значення;
- додавання нового елемента в будь-яку позицію в списку;
- видалення елемента списку;
- об'єднання в одному списку двох або більше лінійних списків;
- розбиття списку на два або більше фрагментів;
- створення копії списку;
- визначення кількості елементів в списку;
- сортування елементів списку;
- пошук елемента, що задовільняє певним критеріям.

Важливими окремими випадками лінійних списків, в яких операції додавання та вилучення певних значень можуть бути виконані лише для першого чи останнього елементу, є:

- ***стек –*** лінійний список, в якому операції додавання та вилучення виконуються лише на одному кінці списку (верхівці стеку). Принцип побудови стеку називають LIFO (англ. last in, last out).
- ***черга*** (однобічна черга) – лінійний список, в якому усі операції додавання виконується на одному кінці (в голові черги), а операції вилучення — на іншому кінці списку (в хвості черги). Принцип побудови черги називають FIFO (англ. first in, first out).
- ***дек або двобічна черга*** (англ. double-ended queue, deq) – лінійний список, в якому всі операції вставки та видалення виконуються на обох кінцях списку.

Не менш важливим окремим випадком лінійного списку є ***зв'язаний список,*** в якому кожний елемент окрім поля даних зберігає також вказівник на наступний. Така структура дозволяє зняти обмеження на зберігання лінійного списку в безперервній області пам'яті.

Важливим узагальненням лінійного списку є багатовимірний масив.

2. Більш складні структури даних

2.1 Граф

Граф — пара множин V і E, елементи множини V називають вершинами (англ. vertex), множина E містить впорядковані та невпорядковані пари вершин. Невпорядкована пара вершин називається ребром, впорядкована — дугою.

2.2 Дерево

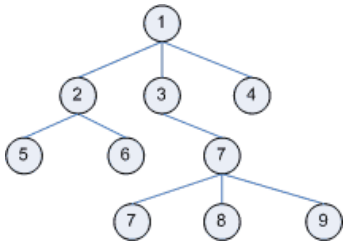


Рис. 15. Графічне зображеня дерева

Дерево - в інформатиці та програмуванні одна з найпоширеніших структур даних. Формально дерево визначається як скінченна множина T з однієї або більше вершин (вузлів, nodes), яке задовольняє наступним вимогам:

існує один виокремлений вузол - корень (root) дерева

інші вузли (за виключенням кореня) розподілені серед $m \geq 0$ непересічних множин T1 .Tm і кожна з цих множин в свою чергу є деревом. Дерева T1 .Tm мають назву піддерев (subtrees) даного кореня.

А) Бінарне дерево:

В програмуванні бінарне дерево -- дерево структура даних, в якому кожна вершина має не більше двох дітей. Зазвичай такі діти називаються

правим та лівим. На базі бінарних дерев будуються такі структури, як бінарні дерева пошуку та бінарні купи.

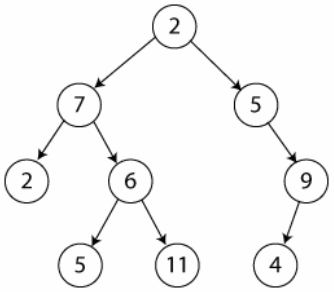


Рис. 16. Графічне зображеня бінарного дерева

Б) Бінарне дерево пошуку:

Бінáрне дéрево пóшуку (англ. binary search tree, BST) в інформатиці - бінарне дерево, в якому кожній вершині x співставлене певне значення val[x]. При цьому такі значення повинні задовольняти умові впорядкованості:

нехай x -- довільна вершина бінарного дерева пошуку. Якщо вершина y знаходиться в лівомі піддереві вершини x, то val[y] ≤ val[x]. Якщо y знаходиться у правому піддереві x, то val[y] ≥ val[x].

2.3 Купа

КУПА або піраміда (англ. heap) в інформатиці -- спеціалізована деревовидна структура даних, в якій існують певні властивості впорядкованості. Така структура даних повинна задовільняти основній властивості купи:

нехай A та B -- елементи купи, такі що B підпорядковане A (B - дитина A). Тоді значення B не повинно перевищувати A, тобто val[B] ≤ val[A]

Найбільш уживаним класом куп є бінарні купи.

Базові операції з купою такі:

- підтримка основної властивості купи
- побудова купи з невпорядкованого масиву
- сортування купи
- видалення найменшого елемента
- отримання найбільшого елемента
- додавання елемента

А) Бінарна купа

Бінарна купа (англ. binary heap) — це структура даних, що є масивом, який можна розглядати як майже повне бінарне дерево. Кожен вузол цього дерева відповідає певному елементу масиву. На всіх рівнях, крім, можливо останнього, дерево повністю заповнене (заповнений рівень — такий, що містить максимально можливу кількість вузлів). Останній рівень заповнюється послідовно зліва направо до тих пір, доки в масиві не закінчатся елементи.

Б) Біноменальна купа

Біноміальна купа (англ. binomial heap) — це множина біноміальних дерев, що задовільняє властивостям біноміальної купи:

Кожне біноміальне дерево у купі підпорядковується властивості неспадної купи (англ. min-heap property): ключ вузла не менший за ключ його батьківського вузла.

Над всіма структурами даних можуть виконуватися **чотири операції**:

1. **Створення**. Операція створення полягає у виділенні пам'яті для структури даних. Пам'ять може виділятися в процесі виконання програми при першій появі імені змінної в початковій програмі або на етапі компіляції.
2. **Знищення.** Операція знищення структур даних протилежна по своїй дії операції створення.
3. **Вибір (доступ).** Операція вибору використовується програмістами для доступу до даних всередині самої структури. Форма операції доступу залежить від типу структури даних, до якої здійснюється звернення.
4. **Оновлення.** Операція оновлення дозволяє змінити значення даних в структурі даних.

Зв’язок простих та структурованих типів
З поняттям **простого типу** пов’язані:

- ім’я типу;
- множина допустимих значень типу;
- набір операцій, відношень та функцій, що визначені для типу.

З поняттям **структурованого типу** пов’язані:

- ім’я типу;
- спосіб об’єднання даних у певну структуру;
- спосіб доступу до даних, що утворюють структурований тип даних.

Динамічні структури даних

Повернемося тепер до питання про економію пам'яті при збереженні табличних даних. З використанням покажчиків можна відмовитися від масиву і використовувати динамічні структури. Найпростіший з них є *список*, що схематично зображується так:

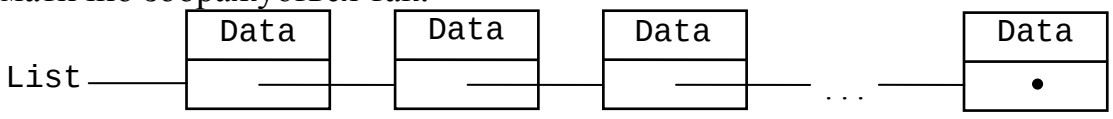


Рис. 17. Графічне зображення списку

Прямокутники на цій схемі — динамічні змінні типу запис, Data (поле (чи поля), що містять корисну інформацію (наприклад прізвища і номери телефонів), поле, що зображене нижче Data — це покажчик на наступну запис. Змінна List також є покажчиком на запис. Жирна крапка в поле «наступний елемент» у самому останньому записі означає, що там лежить значення nil, щоб показати, що цей запис останній в списку.

Для опису списку на Паскалs досить описати тип покажчика на запис і тип самого запису. Виглядає все це так:

type *tItemPtr = ^tItem; {покажчик на елемент}*

```
tItem = record
  Data: tData;    {корисні дані}
  Next: tItemPtr; {покажчик на наступний елемент списку}
end;
```

Оголосити сам список можна як покажчик на елемент:

var List : tItemPtr;

поки наш список порожній, List варто пприсвоїти значення nil. При створенні першого елемента будемо виконувати дії New(List); List^.Next:=nil.

У списках завжди зберігається рівно стільки елементів, скільки потрібно; якщо який-небудь елемент даних утратив свою цінність, то його завжди можна видалити зі списку; якщо з'явилися нові дані, то можна додати новий елемент.

Динамічні змінні: інші види списків, стек і черга.

1. Інші види списків

Крім розглянутих списків можливі більш складні варіанти, зв'язані з наявністю двох додаткових властивостей:

1. Двонаправленність списку. У кожному елементі таких списків є не тільки покажчик на наступний елемент, але і на попередній. Така організація може виявитися корисною при додаванні чи видаленні елемента, що передує зазначеному.

2. Замкнутість списку. Поле next в останньому елементі вказує на перший елемент. Інакше такі списки називаються кільцевими. Цей вид дозволяє спростити процедуру видалення елемента списку й інші операції.

З урахуванням цих властивостей можливі чотири різних типи списків.

2. Стек і черга

Стеком називається такий спосіб збереження даних, при якому елемент, записаний у сховище даних, останнім завжди витягається першим (дисципліна LIFO – «last in - first out»). При витягу елемента відбувається його видалення зі стека.

Розглянемо найпростіший приклад використання стека. Припустимо, що мається рядок, що складається з одних лише відкриваючих і закриваючих дужок. Потрібно визначити, чи є віна правильним дужковим виразом (тобто для кожної відкриваючої дужки повинна знайтися закриваюча). Заведемо масив і змінну для збереження номера останнього значимого елемента в масиві (тобто вершини стека), у який при проході по рядку будемо складати всі дужки, що відкриваються, (зі збільшенням номера вершини на 1), а при зустрічі з закриваючої будемо видаляти відповідну відкриваючу (попросту зменшувати номер вершини стека). Якщо виявиться, що «прийшла» закриваюча дужка, а стік порожній (тобто номер вершини дорівнює 0), то вираз помилковий. Це ж можна сказати й у випадку, коли рядок закінчився, а стек не порожній.

Очевидно, що для реалізації такого стека масив використовувати не обов'язково, досить зберігати в деякій змінній лише число відкриваючих дужок. При надходженні закриваючої дужки з цієї змінної віднімається 1.

Помилка виникає, якщо значення змінної стало негативним, або при досягненні кінця рядка воно не дорівнює нулю.

Для даних більш складного виду стек можна організувати за допомогою односпрямованого некільцевого списку. Щоб вставити елемент у стек, потрібно додати його в початок списку, щоб витягти зі стека — одержати дані першого елемента, після чого видалити його зі списку.

Будь-яка реалізація стека повинна містити наступні процедури і функції:

procedure InitStack — ініціалізація стека;
procedure Push(d: tData) — вставити елемент у стек;
procedure Pop(var d: tData) – витягти елемент із вершини стека;
function NotEmpty: boolean – перевірка чи стек порожній ;

Черга відрізняється від стека тим, що останній елемент, що прийшов у неї, буде витягнутий останнім, а перший (першим («FIFO»). За допомогою списків її можна організувати так: будемо зберігати не тільки покажчик на «голову» списку, але і на «хвіст»; додавати будемо в «хвіст», а витягати з «голови».

Будь-яка реалізація черги (не обов'язково за допомогою списків) повинна «уміти» виконувати такі дії:

procedure InitQueue — ініціалізація черги;
procedure AddQueue(d: tData) — поставити елемент у чергу;
procedure SubQueue(var d: tData) – витягти елемент із черги;
function NotEmpty: boolean – перевірка чи черга порожня;

Дерева і пошук у деревах

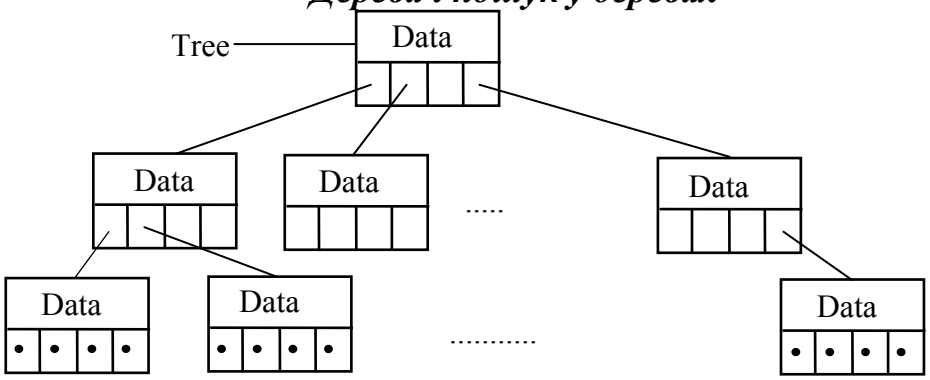


Рис. 18. Графічне зображеня дерева

Деревами називаються структури даних наступного виду:

Елементи дерева називаються *вершинами*. Вершина Tree^ називається *коренем* дерева, а вся множина вершин, зв'язаних з деякою вершиною за допомогою одного з покажчиків називається *піддеревом*. Вершини, у яких усі покажчики рівні nil, іноді називають *листами*.

Докладніше ми розглянемо варіант *двійкового* дерева, тобто такого, у якому кожна вершина має два піддерева (кожне з них може бути порожнім). Такі дерева виявляються дуже зручними для рішення задачі пошуку, коли ключі для наших даних (наприклад прізвища при пошуку телефонних номерів) можна порівнювати на "=", "<" і ">". У кожну вершину дерева

заноситься елемент даних, причому робиться це таким чином, щоб для будь-якої вершини всі ключі даних (чи самі дані в найпростішому випадку) з лівого піддерева були менші ключа цієї вершини, а всі ключі з правого — більші. Виконання такої вимоги можна досягти при послідовному додаванні елементів (тобто побудові дерева, починаючи з «нуля», точніше з nil).

При описаній побудові дерева пошук виявляється досить простою справою: спочатку ми порівнюємо шуканий ключ із ключем кореня дерева. Якщо ці два ключі збігаються, то елемент знайдений, у противному випадку виконуємо пошук у левом піддереві, інакше — у правом, далі в обраному піддереві знову виконуємо порівняння його кореня із шуканим ключем, і т.д. Цей процес закінчиться або коли ми знайшли ключ, або коли чергове піддерево виявилося порожнім (це означає, що такий ключ у дереві відсутній).

Для реалізації двійкового дерева спочатку розглянемо його опис на Паскалі:

```
type tNodePtr = ^tNode; {покажчик на вершину}
    tNode = record
      data: tMyData;
      left,right: tNodePtr;
    end;
    tTree = tNodePtr; {для доступу до дерева досить зберігати покажчик на його корінь}
```

Під даними (tMyData) будемо розуміти запис, що складається з ключа, необхідного для порівнянь, і власне даних:

```
type tMyData = record
key: tKey;
data: tData;
end;
```

Для того щоб реалізувати дії з двійковим деревом, нам знадобляться так називані рекурсивні процедури.

Додавання елемента є рекурсивною процедурою:

```
procedure InsertNode(t: tTree; key: tKey; data: tData);
begin
  if t=nil then begin
    new(t);
    t^.key:=key;
    t^.data:=data;
  end
    else if key<t^.key then InsertNode(t^.left,key,data)
      else InsertNode(t^.right,key,data);
end;
```

Після того як дерево побудоване, можна виконувати пошук (також рекурсивний):

```
function Search(t: tree; key: tKey; var data: tData): boolean;
{повертає значення знайдене / не знайдений}
```

```
begin
  if t=nil then Search:=false
    else if key = t^.key then begin
      data:=t^.data;
      Search:=true;
    end
      else if key<t^.key then Search:=Search(t^.left,key,data)
        else Search:=Search(t^.right,key,data);
end;
```

Легко помітити, що елементи даних, «покладені» у двійкове дерево можна виводити у відсортованому порядку:

```
procedure Traversal(t: tTree); {обхід дерева}
begin
  if t<>nil then begin
    Traversal(t^.left);
    writeln('Key:',t^.key,' Data:',t^.data);
    Traversal(t^.right);
  end;
end;
```

Таблиці і найпростіші алгоритми пошуку.

1. Визначення й описи структур даних

Таблицею будемо називати структуру даних, придатну для збереження набору даних, що мають однакові типи. Найпростішим прикладом такої структури може служити масив, оскільки тип усіх його елементів той самий. Найчастіше елемент таблиці складається з декількох частин, одна з яких має найбільше значення (називається ***ключем***), а інші містять інформацію, зв'язану з цим ключем, чи власне ***дані***. Якщо все це зобразити графічно, то вийде те, що називається таблицею в звичайному змісті:

Таблиця 10

Ф. И. О.	Адреса	Телефон	Рік народження
Петров Ф. М.	Північна 99-88	29-29-29	1962
Іванов П. С.	Світу 111-222	77-88-99	1976
Козлов Н. В.	Жовтнева 135-246	45-67-89	1970
.....			

Тут ключем є прізвище, а всі інші елементи — корисна інформація про людину з таким прізвищем. У випадку, коли наша таблиця стає досить великий, знайти дані про потрібному нам людині стає досить складно. Алгоритми, призначені для пошуку в таблиці даних із зазначеним ключем, називаються ***алгоритмами пошуку***. Пошук може бути ***вдалим*** (якщо елемент із шуканим ключем мається в масиві) і ***невдалим*** (у противному випадку).

При використанні мови Паскаль для роботи з табличними даними досить зручно використовувати запису як елементи даних. У нашому прикладі таблиця буде складатися з елементів наступного типу:

```
type tItem {елемент} = record
    surname: string[30]; {прізвище, ключове поле}
    address: string;     {адреса}
    phone:   longint;    {телефон}
    birth:   word;       {рік народження}
  end;
```

При розгляді алгоритмів пошуку ми будемо користатися більш загальною формою для запису типу елемента:

```
type tItem = record
    key:  tKey;  {ключ}
    data: tData; {дані}
  end;
```

Типи tKey і tData залежать від конкретної задачі, яку потрібно вирішувати. У нашому прикладі tKey — рядок до 30 символів довжиною, а tData можна зробити записом із трьох полів (address, phone і birth).

Розглянемо тепер деякі способи реалізації всієї таблиці:

1. Масив

Це найбільш простий варіант і досить зручний, оскільки положення елемента в таблиці однозначно визначається номером елемента в масиві. Якщо розмір таблиці міняється в ході роботи з нею (дані час від часу додаються чи видаляються), то масив виявляється не дуже економічним: оскільки точна кількість елементів заздалегідь невідома, приходиться заводити масив з великої кількості елементів, частина з яких не буде використовуватися, але буде займати місце в пам'яті.

Для того щоб зберігати таблицю, нам буде потрібно запис із двох полів: сам масив і цілочисльне поле, що позначає поточний розмір масиву:

```
const maxsize = 2000; {максимальний розмір таблиці}
type tTable = record
    a: array[1..maxsize] of tItem; {це сам масив}
    n: integer;  {а це - реальне число елементів}
  end;
var Table: tTable;
```

Передбачається, що в будь-який момент часу дані таблиці зберігаються в перших n елементах масиву, а інші вважаються порожніми.

2. Список

Цей варіант більш економічний у плані витрати пам'яті, тому що завжди буде зайнято рівно стільки місця, скільки потрібно під дані. На відміну від масиву, ми не можемо легко переглядати дані довільного елемента, для переходу від одного елемента до іншого потрібно довго рухатися по ланцюжку покажчиків; це є недоліком списку.

Як виглядає така таблиця на Паскалі нам уже відомо:

```
type tItemPtr = ^tItem; {покажчик на елемент списку}
```

```
    tItem = record    {елемент списку}
      key:  tKey;
      data: tData;
      next: tItemPtr;
    end;
    tList: tItemPtr; {задається покажчиком на перший елемент}
var Table: tList     {таблиця є списком}
```

3. Дерево

Як зберігати і шукати дані в двійковому дереві, ми вже знаємо, а таблицю можна задати так:

```
type tItemPtr = ^tItem;     {покажчик на елемент}
    tItem = record        {елемент}
      key:       tKey;
      data:      tData;
      left, right: tItemPtr;
    end;
    tTree = tItemPtr;
var Table: tTree;          {таблиця є деревом}
```

2. Алгоритми

1. Лінійний пошук у масиві

Нехай таблиця представлена у виді масиву. Тоді перше, що приходить у голову з приводу пошуку елемента — це обхід всіх елементів, починаючи з першого, доти, поки не буде знайдений елемент із шуканим ключем, чи поки масив не скінчиться. Такий спосіб називається ***лінійним пошуком у неупорядкованому масиві***. Оформимо його на Паскалі у виді процедури:

```
procedure LinearSearch(var T:tTable; k:tKey;var index:integer);
var i: integer;
begin
  i:=1; index:=0;
  while (i<=T.n)and(index=0) do begin
    if T.a[i].key=k then index:=i;
    i:=i+1;
  end;
end;
```

Розглянемо докладніше частини цієї процедури. Параметрами процедури є таблиця (T), у якій потрібно шукати елемент, шукане значення ключа (k) і вихідний параметр (index), у якому процедура повинна вказати номер елемента, якщо він знайдений, і 0 у противному випадку. У списку параметрів таблиця T описана як параметр змінна, хоча процедура і не повинна змінювати які-небудь дані з таблиці. Це потрібно для того, щоб не створювати копію таблиці в стеці при передачі параметра процедурі, оскільки таблиця може мати великий розмір.

Можливий більш раціональний варіант: замість того щоб усякий раз перевіряти, чи не закінчився масив, можна використовувати масив з фіктивним елементом номер 0, перед пошуком записувати в нього шукане

значення ключа, і рухатися по масиву від останнього елемента до першого. Такий спосіб називається ***лінійним пошуком з бар'єром*** (бар'єр — нульовий елемент):

```
procedure LinearSearch2(var T:tTable; k:tKey; var index:integer);
var i: integer;
begin
  T.a[0]:=k;  index:=T.n; index:=0;
  while T.a[index]<>k do index:=index-1;
end;
```

У такому варіанті стає значно менше порівнянь, отже, алгоритм працює швидше попереднього.

2. Двійковий пошук

Наступний алгоритм також застосовується для таблиці, представленої у виді масиву, крім того, масив повинний бути відсортованим за значеннями ключа (для визначеності — по зростанню). Тоді при пошуку можна використовувати наступні розуміння: візьмемо елемент, що знаходиться в середині масиву, якщо його ключ дорівнює шуканому ключу, то ми знайшли потрібний елемент, якщо менший — продовжуємо пошук у першій половині масиву, якщо більший — то в другий. Під продовженням розуміємо аналогічний процес: знову беремо середній елемент з обраної половини масиву, порівнюємо його ключ із шуканим ключем, і т.д. Цей цикл закінчиться, коли частина масиву, у якій виконується пошук, не буде містити жодного елемента. Тому що цей алгоритм багаторазово розбиває масив на дві частин, його називають ***алгоритмом двійкового пошуку***. Нижче приведена відповідна процедура на Паскале.

```
procedure BinarySearch(var T:tTable; k:tKey; var index:integer);
var l,c,r: integer;
begin
  index:=0;
  l:=1; r:=T.n;
  while (index=0)and(l<=r) do begin
    c:=(l+r) div 2;
    if T.a[c].key=k then index:=c
    else if T.a[c].key>k then r:=c-1
    else l:=c+1;
  end;
end;
```

Змінні l, r і c позначають відповідно номер лівого краю, центра і правого краю частини масиву, у якій ми шукаємо елемент із заданим ключем. Пошук припиняється або якщо елемент знайдений (index <> 0), або якщо частина масиву, у якій потрібно шукати, була вичерпана (тобто номер лівого краю перевищив номер правого). Усередині циклу знаходимо номер середини частини масиву (c), потім порівнюємо ключ цього середнього елемента із шуканим ключем. Якщо виконалася рівність, то елемент

знайдений, якщо середній більше шуканого, то встановлюємо праву границю частини масиву рівною c-1, якщо більше — змінюємо ліву границю на c+1.

3. Лінійний пошук у списку

Нехай тепер дані таблиці містяться в списку. У цьому випадку можна використовувати для пошуку алгоритм, дуже схожий на алгоритм лінійного пошуку в масиві. Відмінність лише в тім, що зміна номера поточного елемента заміняється переходом до покажчика на наступний елемент списку:

```
procedure SearchInList(T: tTable; k: tKey; var p: tItemPtr);
var notfound: boolean;
begin
  notfound:=true;
  p:=T;
  while (p<>nil) and (notfound) do begin
    if p^.key=k then notfound:=false;
    p:=p^.next;
  end;
end;
```

Параметр T у цьому випадку не потрібно робити параметром-змінною, оскільки він є тільки покажчиком на початок таблиці, а сама таблиця лежить у динамічній пам'яті. Замість номера знайденого елемента будемо повертати користувачу нашої процедури покажчик на нього (якщо пошук був удалим) чи nil, якщо пошук невдалий.

Скорочувати число перевірок у циклі за допомогою бар'єра було б нерозумно: щораз бар'єр прийдеться ставити в «хвіст» списку, а для цього потрібно спочатку обійти весь список, починаючи з голови, затрачаючи при цьому багато часу.

Змішані таблиці

Найбільш очевидним способом збереження таблиці є масив, у якому трохи перших елементів являють собою корисні дані, а інші елементи вважаються порожніми. Нижче буде розглянутий інший варіант застосування масиву для реалізації таблиці.

Дозволимо даним розташовуватися в будь-яких елементах масиву, а не тільки в перших n. Щоб відрізняти порожні елементи від зайнятих нам знадобиться спеціальне значення ключа, яке ми будемо заносити в ключове поле всіх порожніх комірок. Якщо ключ — число, а всі корисні ключі позитивні, то можна як ключ порожньої комірки використовувати 0, якщо ключі — рядки, що містять прізвища, то ключем порожньої комірки можна зробити порожній рядок і т.п. Нехай ключами є рядки, тоді для таблиці будуть потрібні такі оголошення:

```
const empty = '';
      nmax  = 1000;
type tKey   = string;
     tData  = .....;
     tItem  = record
                key:  tKey;
```

```
    data: tData;
  end;
tTable = array[0..nmax-1] of tItem;
```

Перед тим як поміщати дані в масив заповнимо ключові поля всіх його елементів «порожніми» значеннями. Заносити в таблицю будемо не всі дані відразу, а один за другим, по черзі. Для того, щоб визначити номер комірки масиву, у яку потрібно помістити елемент даних, що додається, напишемо функцію, значення якої залежить тільки від ключа елемента, що додається. У такій ситуації пошук можна буде здійснювати досить просто: знаходимо значення функції на шуканому ключі, і дивимося на елемент масиву з отриманим номером. Якщо ключ елемента дорівнює шуканому ключу, ми знайшли те, що шукали, інакше — пошук виявився невдалим.

Реалізована описаним способом таблиця називається **змішаною** (чи **hash-таблицею**), а функція — **функцією розміщення** ключів (**hash-функцією**). Такі назви зв'язані з тим, що дані безладно розкидано по масиву.

Тепер покажемо, як усе сказане втілити в програму на Паскалі. Як ключі в наших прикладах використовуються рядки, тому можна запропонувати такий варіант хэш-функции: скласти коди всіх символів рядка, і, щоб отримане число не виходило за максимально можливий індекс масиву, візьмемо залишок від ділення на розмір масиву:

```
function hash(key: tKey): integer;
var i: integer;
begin
  sum:=0;
  for i:=1 to length(key) do sum:=sum+ord(key[i]);
  hash := sum mod nmax;
end;
```

Процедура додавання елемента в таблицю в попередньому варіанті буде виглядати так:

```
procedure AddItem(var t: tTable; item: tItem);
var h: integer;
begin
  h:=hash(item.key);
  t[h]:=item.key;
end;
```

У написаної процедури є один істотний недолік: якщо елемент, номер якого вказала нам хэш-функція був зайнятий, то нові дані будуть записані на місце старих, а старі безповоротно зникнуть. Щоб вирішити цю проблему будемо намагатися знайти який-небудь інший вільний елемент для даних, що додаються. Тут знадобиться ще одна функція (вторинна хэш-функція), яка за номером зайнятого елемента і за значенням ключа даних, що додаються, укаже номер іншої комірки, якщо і там буде зайнято, викличемо вторинну функцію ще раз, і т.д. доти поки вільна комірка не знайдеться.

Найбільш проста хэш-функція буде додавати до номера зайнятої комірки яке-небудь постійне число:

```
const HC = 7;
function hash2(n: integer, key: tKey): integer;
begin
  hash2 := (n + HC) mod nmax;
end;
```

Залишок від ділення на nmax знадобилося обчислювати по тій же причині, що й у первинній хэш-функції.

Зараз можна написати остаточний варіант процедури додавання елемента даних у таблицю:

```
procedure AddItem(var t: tTable; item: tItem);
var h: integer;
begin
  h:=hash(item.key);
  while t[h].key<>empty do h:=hash2(h,item.key);
  t[h].key:=item.key;
  t[h].data:=item.data;
end;
```

Нехай у хэш-таблицю занесені всі необхідні дані і потрібно відшукати дані з деяким ключем. Для цього будемо діяти за такою схемою: обчислюємо значення хэш-функції на даному ключі, якщо комірка з отриманим номером вільна, то елемент не знайдений, якщо зайнята, то порівнюємо її ключ із шуканим. У випадку збігу ми знайшли потрібний елемент, інакше — знаходимо значення вторинної функції і дивимося на ключ в комірці з отриманим номером. Якщо він дорівнює «порожньому» значенню, то пошук невдалий, якщо дорівнює шуканому ключу — то вдалий, інакше — знову знаходимо значення вторинної хэш-функції і т.д. На Паскалі все це виглядає так:

```
const notfound = -1;   continue = -2;
procedure Search(var t: tTable; key: tKey; var index: integer);
var h: integer;
begin
  h:=hash(key);  index:=continue;
  repeat
    if t[h].key = key then index:=h
    else if t[h].key = empty then index:= notfound  else h:=hash2(h,key);
  until index<>continue;
end;
```

Процедура видає відповідь про результати пошуку через параметр-змінну index. При вдалому пошуку там буде лежати номер знайденого елемента, при невдалому — константа notfound. Константа continue означає «поки не знайдений» і використовується тільки усередині процедури. При пошуку спочатку обчислюється значення первинної хэш-функції, а в index заноситься значення continue. Потім виконується перевірка на рівність ключів, якщо вона виконується, то комірка масиву знайдена, і ми записуємо її номер у index, інакше, якщо комірка порожнія, то елемент не знайдений (у

index записуємо notfound), у третьому випадку знаходимо значення вторинної функції. Ці дії продовжуються доти, доки index не перестане бути рівним continue.

Метод покрокової деталізації

1-й крок. Розділити складну задачу на декілька простих задач (якщо виконавець не може виконати дану операцію, то виникає необхідність розкласти її на деяку сукупність більш простих операцій).

2-й крок. Якщо отримаємо сукупність операцій, яка містить в собі тільки операції системи команд виконавця, то потрібно завершити процес деталізації.

3-й крок. Скласти для кожної допоміжної задачі свій допоміжний алгоритм, при необхідності об'єднати дані в структури.

4-й крок. Скомпонувати результати проектування простих задач.

5-й крок. Проаналізувати роботу алгоритму.

Структурний підхід забезпечує:

- легкість читання алгоритму;
- простоту перевірки правильності виконання алгоритму;
- зручність в його модифікації.

Структурний підхід передбачає:

- конструювання алгоритму з використанням трьох базових алгоритмічних структур;
- використання методу покрокової деталізації;
- використання допоміжних алгоритмів (підпрограм);
- об'єднання даних, які зв'язані за значенням, у складні структури даних;
- аналіз алгоритму (контроль правильності кожної структури алгоритму і взаємозв'язків структур).

Пошук оптимального алгоритму розв'язування

Критерії оптимальності алгоритму

По-перше, це може бути *компактність* текстового коду програми, що реалізує розроблений алгоритм.

По-друге, економне використання пам'яті комп'ютера алгоритмом, що реалізує поставлену задачу. Цей фактор оцінки ефективності роботи алгоритму називається *ємкісною складністю* алгоритму.

По-третє, розв'язування алгоритмом поставленої задачі за найменшу кількість операцій, що називається *часовою складністю.*

На жаль, розробники деяких сучасних комерційних продуктів не завжди звертають увагу на часову ефективність програм, на раціональне використання пам'яті комп'ютера. Подекуди вважається, що користувач завжди зможе докупити додаткову пам'ять, однак швидкість роботи комп'ютерів не може збільшуватися безмежно.

Тому в першу чергу при розробці алгоритму треба намагатися досягти найменшої кількості виконуваних ним операцій, записуючи алгоритм при

цьому максимально компактно і зрозуміло та використовуючи мінімальну кількість додаткових змінних.

Оскільки оцінка оптимальності алгоритму, що розробляється, є досить суб'єктивною, то можна сказати, що алгоритм вважається оптимальним, якщо не існує алгоритму, який працює краще. Але все ж таки як дізнатися, чи наш алгоритм є оптимальним?

Оскільки кожний алгоритм розв'язує конкретну задачу, то для того, щоб дізнатися чи є алгоритм оптимальним, потрібно визначити, яку найменшу кількість операцій необхідно виконати, щоб отримати очікуваний результат.

На прикладі конкретної задачі спробуємо з'ясувати, як можна оцінити оптимальність розробленого алгоритму.

Практичне застосування теоретичних основ програмування мовою Pascal

Лінійні програми

Задача 1. Обчислити площу трикутника за відомими трьома сторонами (вважається, що усі введені дані відповідають умові існування трикутника та є додатними).

І. Постановка задачі. Для того, щоб розв'язати задачу з використанням ПК, необхідно згідно до поставленої умови задач ***визначити вхідні та вихідні параметри***, і, при потребі, деталізувати умову задачі: визначити, які дані допустимі; за яких умов можливе отримання допустимі результатів, а за яких – ні; які результати будуть вважатися правильними.

В нашому випадку вхідними параметрами будуть сторони трикутника *a, b, c,* вихідними – площа *S*. Причому, усі (вхідні та вихідні) дані мають бути додатними, і обов'язково має виконуватись нерівність трикутника для кожної сторони: будь-яка сторона трикутника менша за суму двох: $a<b+c;\ b<a+c;c<a+b$. (в умові вважається, що дані усі введені завідомо вірно!!!)

ІІ. Опис алгоритму, який в свою чергу поділяється на два етапи:

1) ***розв'язання поставленої задачі за допомогою відомих математичних та логічних законів, властивостей*** тощо:

Як відомо з курсу математики, площа трикутника за відомими трьома сторонами обчислюється за формулою Герона:

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

де *a, b, c* – сторони трикутника,

p – півпериметр, який обчислюється за формулою: $p = (a+b+c)/2$.

2) ***опис алгоритму за допомогою блок-схеми***:

Опишемо словесно алгоритм розв'язання даної задачі.

1. Початок програми.
2. Введення вхідних даних.
3. Обчислюємо півпериметр за відомою формулою, та площу за формулою Герона.

4. Виводимо площу на екран.
5. Кінець програми.

Складаємо блок-схему (рис. 19) за описаним словесним алгоритмом.

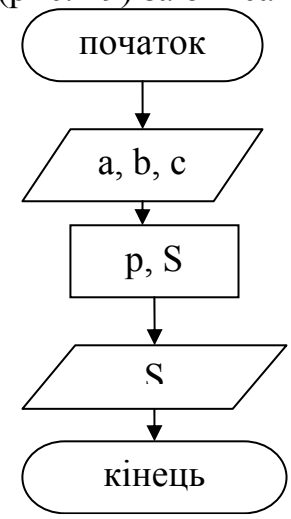


Рис. 19. Блок-схема розв'язку задачі 1.

Після складання блок-схеми, переходимо до наступного етапу.

III. Складання програми: написання програми мовою програмування Pascal.

```
var a,b,c, S, p : real;          {оголошення змінних дійсного типу}
begin                            {початок програми}
    readln(a,b,c);               {зчитуємо вхідні дані}
    p:=(a+b+c)/2;                {обчислюємо півпериметр}
    S:=sqrt(p*(p-a)*(p-b)*(p-c)); {обчислюємо площу}
    writeln('S=',S:0:2);         {виводимо площу на екран}
end.                             {кінець програми}
```

IV. Тестування і налагодження програми: перевірка правильності роботи програми за допомогою тестів і виправлення виявлених помилок.

Таблиця 11

Приклади тестів до задачі 1.

Вхідні дані	**Вихідні дані**
3 4 5	6,00
2 4 4	3,87
2 3 3	2,83

V. Експлуатація програми: подальше використання розробленої програми.

Розглянемо ще декілька прикладів задач лише з математичним розв'язком і прикладом розв'язання на мові програмування з детальними поясненнями.

Завдання: скласти блок-схему до кожної наступної задачі.

Задача 2.

Знайти площу круга, якщо відомий його радіус.

Розв'язання

Як відомо з шкільного курсу математики, площа круга обчислюється за формулою $s=\pi r^2$, де $\pi=3.1415926\ldots$, а r – радіус круга.

Програма розв'язку задачі

```
var r,s:real;                                          {опис змінних}
begin                                                  {початок тіла програми}
  write('Введіть радіус круга ');          {виведення повідомлення користувачу}
  readln(r);                                           {введення змінної}
  s:=pi*sqr(r);                    {В мові Паскаль константа Пі відноситься до
                                   вбудованих констант і її не потрібно описувати}
  write('Площа круга з радіусом ',r:5:2,' = ',s:5:2);      {виведення результату}
end.                                                   {кінець програми}
```

Задача 3.

Знайти суму цифр трицифрового цілого десяткового числа.

Розв'язання

Запишемо трицифрове число n у вигляді abc, де у числі a – кількість сотень, b – кількість десятків і c – кількість одиниць.

Таким чином, вхідними даними буде лише саме число n, а вихідними – сума цифр, позначимо її s.

Щоб отримати кількість сотень, потрібно саме число розділити націло на 100, тобто a = n div 100; відповідно кількість одиниць c = n mod 10, а кількість десятків b = (n div 10) mod 10.

Кількість десятків можна виразити і по іншому. Спробуйте самостійно знайти іншу формулу для знаходження кількості десятків.

Отже, s=a+b+c.

```
var n,s:integer;                                       {опис змінних}
a,b,c:byte;
begin                                                  {початок тіла програми}
  write('Введіть трицифрове число'); {виведення повідомлення користувачу}
  readln(n);                                           { введення змінної}
  a:=n div 100;
  b:=(n div 10) mod 10;
  c:= n mod 10;
  s:=a+b+c;
  write('Сума цифр даного числа = ',s);                {виведення результату}
end.                                                   {кінець програми}
```

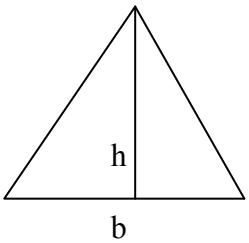


Рис. 20. Малюнок до задачі 4.

Задача 4.

Визначити висоту трикутника площею *S*, якщо його основа більша за висоту на величину *a*. Результат вивести з точністю до сотих.

Математичний розв'язок.

За умовою задачі $b-h=a$, тоді звідси $b=h+a$, але ми знаємо,

що $S = \frac{1}{2}hb$, звідси підставивши значення *b* отримаємо квадратне рівняння: $x^2 + ax - 2s = 0$, яке розв'язується через дискримінант формула якого відома ще зі школи.

Код програми

```
var s, a, h, d : real;                          {оголошуємо змінні}
begin                                           {початок програми}
   write('Введіть площу та величину a ');
   readln(s,a);                                 {зчитуємо дані}
  d:=sqr(a)+8*s;                                {обчислюємо d}
  h:=(-a+sqrt(d))/2;                            {обчислюємо h}
  writeln('h= ',h:5:2);                 {виводимо результат на екран}
  readln                                 {своєрідна затримка на екрані}
end.                                            {кінець програми}
```

Розгалуження

Задача 5.

Обчислити значення логічних виразів.

1) $x < y$ при $x = -2.5$, $y = 0.1$.

Ця умова буде істиною.

2) *a and not* (*b* = *c*) при *a* = *false*, *b* = *false*, *c* = *true* .

Умова буде хибною (*false*), тому що при з'єднанні двох умов службовим словом ***and***, якщо одна з умов хибна, то й вся умова хибна. Дійсно, якщо хоч одна з умов, поєднаних словом "***ma***", не являється істиною, то й весь логічний вираз буде хибним.

3) *not* (*a and b*) *or b* = *a n*при *a* = *true*, *b* = *false*

Ця умова буде істиною, тому що (*a and b*) - хибна (одна з умов хибна); *not* (*a and b*) - істина (заперечність хибності); *not* (*a and b*) *or b* = *a* - істина (бо, якщо з двох, поєднаних службовим словом ***or***, умов хоч одна істина, то весь вираз істинний).

4) *not* (*a and* (*x* < *y*)) *or* (*x* < 0) при *a* = *true*, *x* = -0.1, *y* = 0.7

Умова буде істиною, тому що друга з умов (*x* < 0) буде істиною, а при з'єднанні двох умов службовим словом ***or*** достатньо одній з них бути істиною, щоб весь логічний вираз був істинним.

Задача 6.

Записати у вигляді логічних виразів висловлювання, наведені нижче:

1) значення *x* **не** належить інтервалу (0;1)

(*x* <= 0) *or* (*x* >= 1)

2) точка *M*(*x*, *y*) лежить в другій чверті координатної площини

(*x* < 0) *and* (*y* > 0)

3) точка *M*(*x*, *y*)лежить всередині або на межі одиничного круга з центром у початку координат

(*sqr*(*x*) + *sqr*(*y*) <= 1)

4) координати дійсного вектора *x*(*x*1, *x*2, *x*3) утворюються неспадну послідовність і всі вони невід'ємні

(*x*1 <= *x*2) and (*x*2 <= *x*3) *and* (*x*1 >= 0) *and* (*x*2 >= 0) *and* (*x*3 >= 0)

Задача 7.

Записати за допомогою умовного оператора виконання дій:

1) дійсне значення *x* замінити його абсолютною величиною

if x < 0 *then x* := -*x*;

2) менше з двох дійсних значень *x* та *y* (або будь-яке з них, якщо вони рівні) замінити нулем *if x* < *y then x* := 0 *else y* := 0;

3) присвоїти змінній *x* значення 0, якщо її початкове значення належало інтервалу (0,2)

if (0 <= *x*) *and* (*x* <= 2) *then x* := 0;

Задача 8.

Дано значення дійсної величини *X*. Визначити:

$$\frac{x-5}{x^3+x-2}$$

Для розв'язання цієї задачі необхідно пам'ятати, що ділити на нуль не можна.

```
Uses crt;
Var X,Rezultat:real;
Begin
  Clrscr;                                   {Очищення екрану}
  Write('Введіть значення X: ');
  Readln(X);
  If X*X*X+X-2<>0
  Then
    begin
      Rezultat:=(X-5)/(X*X*X+X-2);
      Writeln('Rezultat=',Rezultat:8:2);
    end
  Else
    Writeln('Обчислення неможливі – ділення на нуль!');
  Readkey;       {Затримка на екрані до натискання будь якої клавіші}
End.
```

Задача 9.

При даному значенні *X* обчислити:

$$\sqrt{X^3-\sqrt{X-1}}$$

Для розв'язання цієї задачі необхідно пам'ятати, що не можна знайти квадратний корінь з від'ємного числа.

```
Var X,Rezultat:real;
Begin
```

```
  Write('Введіть значення X: ');
  Readln(X);
  If (X>=1) and (X*X*X-sqrt(X-1)>=0)
  Then
    begin
      Rezultat:=sqrt(X*X*X-sqrt(X-1));
      Writeln('Rezultat=',Rezultat:8:2);
    end
  Else
    Writeln('Обчислення неможливі – від'ємний підкореневий вираз!');
End.
```

Задача 10.

Дано дійсні значення *x* та *y*. Обчислити:

$$\frac{\sqrt{X^3 - Y + 0{,}5}}{X^2 - Y^2}$$

Для розв'язання цієї задачі необхідно виконання умов обох попередній прикладів.

```
Var X,Y,Rezultat:real;
Begin
  Write('Введіть значення X та Y: ');
  Readln(X);
  If (sqr(X)-sqr(Y)<>0) and (X*X*X-Y+0.5>=0)
  Then
    begin
      Rezultat:=sqrt(X*X*X-Y+0.5)/(sqr(X)-sqr(Y));
      Writeln('Rezultat=',Rezultat:8:2);
    end
  Else
    Writeln('Обчислення неможливі!');
End.
```

Задача 11.

Розв'язати квадратне рівняння.

```
var a,b,c,d: real;
begin
  writeln;
  write('Уведіть коефіцієнти a,b,c квадратного рівняння : ');
  readln(a,b,c);
  d:=sqr(b)-4*a*c;
  if d>=0 then
    if d=0 then writeln('Єдиний корінь: x=',-b/(2*a):8:3)
       else writeln('Два корені : x1=',(-b+sqrt(d))/(2*a):8:3,, x2=',(-b-
sqrt(d))/(2*a):8:3)
```

```
else {d<0} writeln('Коренів немає');
readln;
end.
```

Задача 12.

Обчислити значення функції:

$$y=\begin{cases}\log_7(8-x)+3x-1, & x<8\\ \dfrac{\sqrt{x-8}+x}{x^2-17x+70}, & 8\le x<10\\ \sin 2x-\cos 2x+2\sin^2 x, & x\ge 10\end{cases}$$

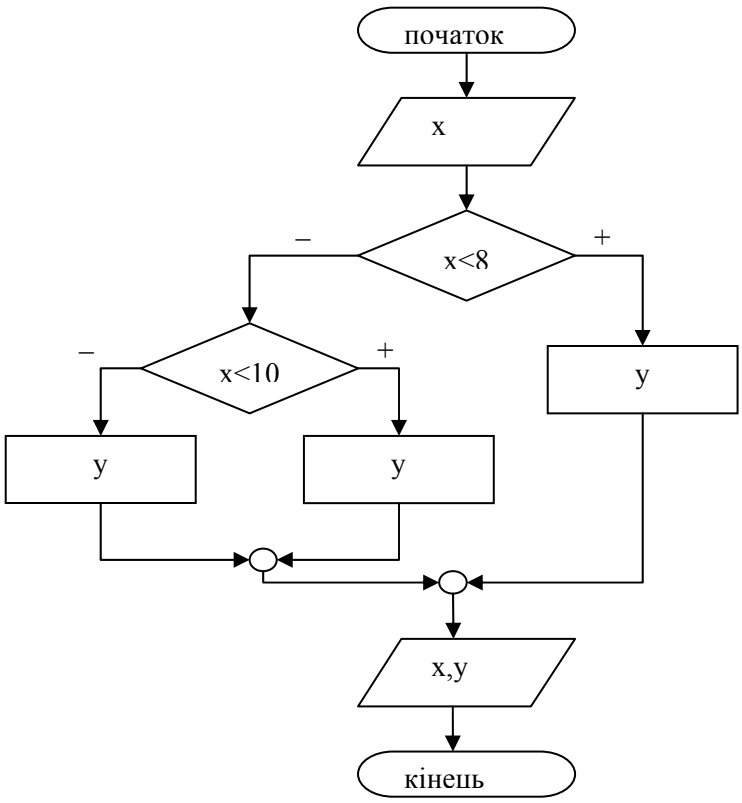


Рис. 21. Блок-схема розв'язку задачі 12.

```
var x, y : real;
begin
   write('x=');   readln(x);
   if x<8 then y:=ln(8-x)/ln(7)+3*x-1
   else if x<10 then y:=(sqrt(x-8)+x)/(sqr(x)-17*x+70)
   else y:=sin(2*x)-cos(2*x)+2*sqr(sin(x));
   writeln('x=',x:6:2,' y=',y:6:2);
   readln;
end.
```

Задача 13.

На площині задано чотирикутник координатами своїх вершин. Визначити найбільшу з його сторін (її довжину).

```
var x1,x2,x3,x4,y1,y2,y3,y4,a,b,c,d,max : real;
begin
  write('Vvedit kkordunati poparno x ta y');
  readln(x1,y1,x2,y2,x3,y3,x4,y4);
```

```
  a:=sqrt(sqr(x1-x2)+sqr(y1-y2));  b:=sqrt(sqr(x3-x2)+sqr(y3-y2));
  c:=sqrt(sqr(x1-x4)+sqr(y1-y4));  d:=sqrt(sqr(x3-x4)+sqr(y3-y4));
  max:=a;
  if b>max then max:=b;   if c>max then max:=c;   if d>max then max:=d;
  writeln('max=',max);
  readln;
end.
```

Задача 14.

Скласти програму, яка визначає вид паралелограма (ромб, прямокутник, квадрат, паралелограм) за відомими двома сторонами, a, b, та кутом між ними U.

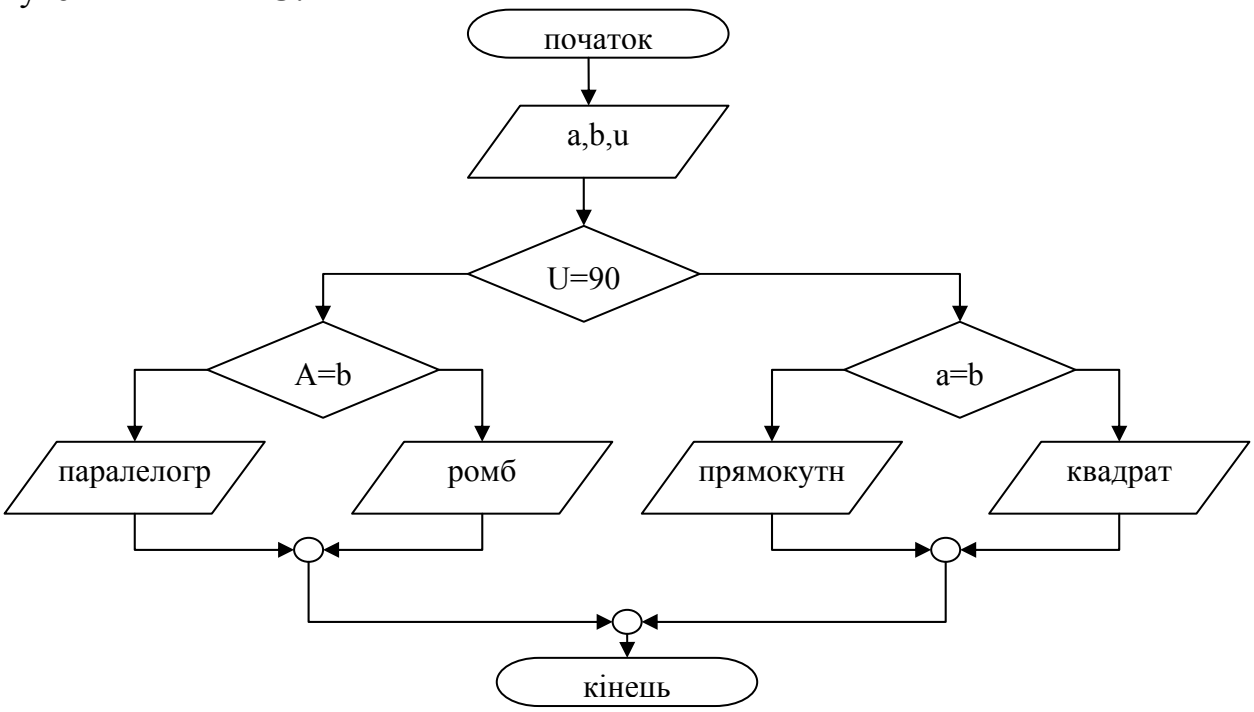


Рис. 22. Блок-схема розв'язку задачі 14.

```
var a, b, u : real;
begin
   write('vvedit storonu a,b, ta kut miw numu ');
   readln(a,b,u);
   if u=90 then
      if a=b then writeln('kvadrat')
      else writeln('prjamokytnuk')
   else if a=b then writeln('romb')
        else writeln('paralelogram');
   readln;
end.
```

Оператор вибору

Задача 15.

Розробити діалогову програму, яка запитує вік користувача і визначає, до якої вікової категорії він належить:

1) від 1 до 10 років - дитина;
2) від 11 до 15 років - підліток;
3) від16 до 20 років - юнак (юнка);
4) від 21 до 30 років - молода людина;
5)після 31 року - доросла людина.

Особливих пояснень ця задача не потребує, адже її можна розв'язати і за допомогою команди розгалуження. Однак зробимо її за допомогою команди вибору, причому, щоб скористатися гілкою***Else***, будемо вважати, що людина може мати вік не більше 150 років (навіть за всіма відомими рекордами, людина не може жити більше 150 років). Якщо ж користувач введе число, що не входить в дозволений діапазон, будемо вважати, що він пожартував.

```
Var Years:byte;
Begin
  Write('Введіть Ваш вік: ');   Readln(Years);
  Write('Ви ');
  Case Years of
   0..10: Writeln('- дитина.');
   11..15: Writeln('- підліток.');
   16..20: Writeln('- юнак (юнка).');
   21..30: Writeln('- молода людина.');
   31..150: Writeln('- доросла людина.')
   Else writeln(' Мабуть, пожартували? Людина стільки не живе!');
  End;
End.
```

Задача 16.

Розробити програму видачі номеру кварталу, до якого відноситься місяць, заданий числом від 1 до 12.

```
Var Month:byte;
Begin
  Write('Введіть номер місяця: ');
  Readln(Month);
  Case Month of
   1..3: Writeln('Перший квартал.');
   4..6: Writeln('Другий квартал.');
   7..9: Writeln('Третій квартал.');
   10..12: Writeln('Четвертий квартал.');
   Else writeln('Помилка вхідних даних.');
  End;
End.
```

Задача 17.

Розробити програму виведення інформації про день тижня, - вихідний він чи робочий, якщо задано його номер від 1 до 7 (1 - понеділок).

```
Var Day:byte;
Begin
  Write('Введіть номер дня тижня: ');
  Readln(Day);
  Case Day of
   1..5: Write('Це робочий день ');
   6,7: Write('Це вихідний день ');
   Else write('Це не день ');
  End;
  Writeln('тижня.');
End.
```

Задача 18.

Дано ціле число *N* (1 <= *N* <= 3) та дійсне число *X*. За даним значенням змінної *N*, яка є номером функції, визначити:

1) sin *X*; 2) cos *X*; 3) tg *X*.

```
Var N:byte;
    X,Y:real;
Begin
  Write('Введіть значення X: ');
  Readln(X);
  Write('Введіть номер функції, що обчислюється: ');
  Writeln('1 - sin');
  Writeln('2 - cos');
  Writeln('3 - tg');
  Readln(N);
  Writeln('Результат обчислень:')
  Case N of
   1: begin Y:=sin(X); writeln('sin(x)=',Y:8:2); end;
   2: begin Y:=cos(X); writeln('cos(x)=',Y:8:2); end;
   3: begin Y:=sin(X)/cos(X); writeln('tg(x)=',Y:8:2); end;
   Else writeln('Помилка вхідних даних.');
  End;
End.
```

Задача 19.

Розробити алгоритм-"лотерею", який, використовуючи генератор випадкових чисел, визначатиме призи:

1) комп'ютер; 2) принтер; 3) сканер; 4) компакт-диск; 5) набір дискет.

Якщо ми хочемо зробити безпрограшну лотерею, необхідно примусити генератор випадкових чисел генерувати числа в діапазоні від 1 до 5. Для цього можна скористатися наступним виразом:

Random(4) + 1.

Нагадуємо, що генератор генерує цілі числа в діапазоні від 0 до числа, що вказано в дужках після слова ***random***, але додаванням до цього виразу одиниці ми позбавимось числа 0. Таким чином програма для розв'язання цієї задачі має наступний вигляд:

```
Var N:byte;
Begin
  Randomize; {Процедура, що дозволяє генерувати при кожному новому
                                                запуску програми нові числа}
  N:=random(4)+1;
  Write('Вітаємо! Ви виграли  ')
  Case N of
   1: writeln('комп'ютер!!!');
   2: writeln('принтер!!!');
   3: writeln('сканер!!!');
   4: writeln('компакт-диск!!!');
   5: writeln('набір дискет!!!');
  End;
End.
```

Задача 20.

Дано натуральне число *N* (*N* <= 100), яке позначає вік людини. Додати до цього числа відповідно слова: "рік", "роки", "років", наприклад: 1 рік, 12 років, але 43 роки.

Очевидно, що для того, щоб правильно дописати відповідне слово, необхідно виділити останню цифру числа, що позначає вік людини. Тоді, якщо це цифра "1", то дописується слово "рік", якщо цифри "2", "3" або "4" - дописується слово "роки", а в усіх останніх випадках - дописується слово "років". Виключенням являється діапазон між 10 та 20 роками: в цих випадках завжди пишеться слово "років".

```
Var Years:byte;
Begin
  Write('Введіть Ваш вік: ');  Readln(Years);  Write('Вам ',Years);
  If (Years>=10) and (Years<=20)
  Then writeln('років')
  Else
    Case Years mod 10 of
      1: writeln('рік.');
      2..4: writeln('роки.');
      0,5..9: writeln('років.');
  End;
End.
```

Цикл з параметром

Задача 21.

Знайти суму всіх натуральних чисел від 1 до 100.

Розв'язок:

```
Var Sum, i : integer;
Begin
    Sum := 0;
      For i := 1 to 100 do
            Sum := Sum + i;
    Writeln('Sum = ', Sum);
End.
```

Задача 22.

Дано ціле *n*. Визначити *n*!

Відомо, що *n*! (вимовляється, як н-факторіал) - це добуток всіх натуральних чисел від 1 до *n*. Тому вихідна програма має вигляд:

```
Var I,n:word;
   Factorial:longint;
Begin
  Factorial:=1;
  Write('Введіть значення n: ');
  Readln(n);
  For I:=1 to n do  Factorial:=Factorial*I;
  Writeln('Factorial= ',Factorial);
End.
```

Задача 23.

Дано ціле *n*. Визначити 1*3*5*7*…*(2*n*+1).

```
Var I,n:word;          Rez:longint;
Begin
  Rez:=1;   Write('Введіть значення n: ');   Readln(n);
  For I:=0 to n do  Rez:=rez*(2*I+1);
  Writeln('Rez= ',Rez);
End.
```

Задача 24.

Вивести в 9 стовпчиків всі чотиризначні числа паліндроми (числа, які читаються зліва на право і справа на ліво однаково 1221, 3553).

```
var i, a, b, c, d, k : integer;
begin
  k:=0;
  for i:=1000 to 9999 do          {Задаємо цикл від 1000 до 9999, т.б. пробігаємо
  begin                                                {всі чотиризначні числа}
    a:=i div 1000;                                     {відділяємо першу цифру}
    b:=(i div 100) mod 10;                             {відділяємо другу цифру}
    c:=(i div 10) mod 10;                              {відділяємо третю цифру}
    d:=i mod 10;                                       {відділяємо четверту цифру}
```

```
    if (a=d) and (b=c) then          {перевіряємо чи 1 і 4 та 2 і 3 цифри рівні}
     begin
       write(i:5);                   {якщо число паліндром, то виводимо на екран}
        k:=k+1;                      {встановлюємо лічильник скільки вивели чисел}
        if k mod 9 =0 then writeln;  {якщо вивели 9 шт. перех. на наст. рядок}
    end;
  end;
end.
```

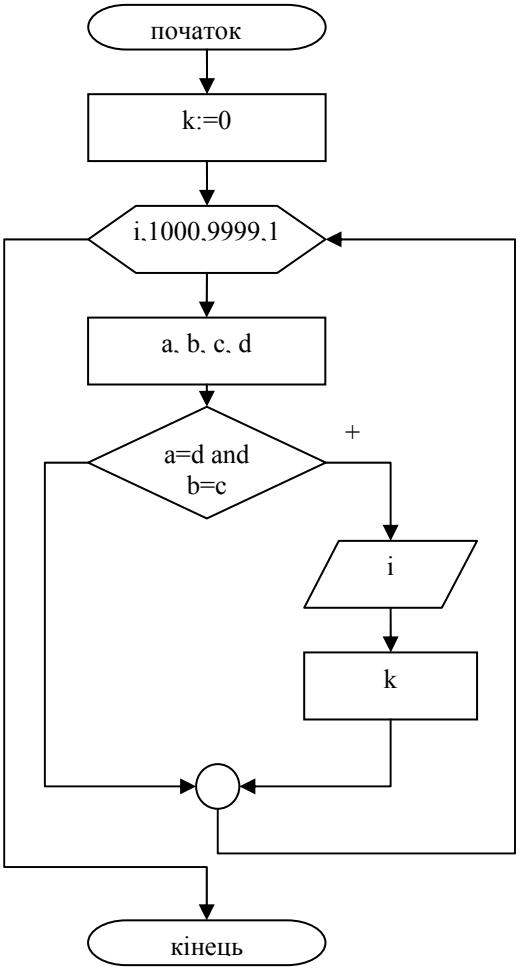


Рис. 23. Блок-схема розв'язку задачі 24.

Задача 25.

N-кутник задано координатами своїх вершин, які вводяться по черзі одна за одною по часовій стрілці. Обчислити його площу, використовуючи векторний добуток.

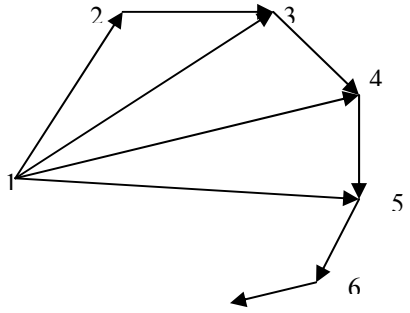


Рис. 24. Многокутник, що розбитий на трикутники

Розіб’ємо многокутник на трикутники, як показано на малюнку. Площа кожного такого трикутника буде дорівнювати векторному добутку, поділеному на 2. Знайшовши всі такі площі і додавши їх отримаємо площу многокутника.

Розглянемо площу одного такого трикутника для наочності.

Нехай вершини мають попарно координати x_1, y_1, x_2, y_2, x_3, y_3. Представимо наш трикутник як трикутник, утворений двома векторами, які виходять з однієї точки: $a=(a_x;a_y;a_z)$, $b=(b_x,b_y,b_z)$, де $a_x=x_2-x_1$; $a_y=y_2-y_1$; $a_z=0$; $b_x=x_3-x_1$; $b_y=y_3-y_1$; $b_z=0$. Як відомо, площа трикутника за векторним добутком буде обчислюватись за формулою:

$S = \frac{1}{2}\sqrt{\begin{vmatrix} a_y & a_z \\ b_y & b_z \end{vmatrix}^2 + \begin{vmatrix} a_x & a_z \\ b_x & b_z \end{vmatrix}^2 + \begin{vmatrix} a_x & a_y \\ b_x & b_y \end{vmatrix}^2}$. А підставивши відповідні значення, отримаємо формулу: $S = \frac{1}{2}\sqrt{\begin{vmatrix} a_x & a_y \\ b_x & b_y \end{vmatrix}^2} = \frac{1}{2}\left|a_x b_y - b_x a_y\right|$.

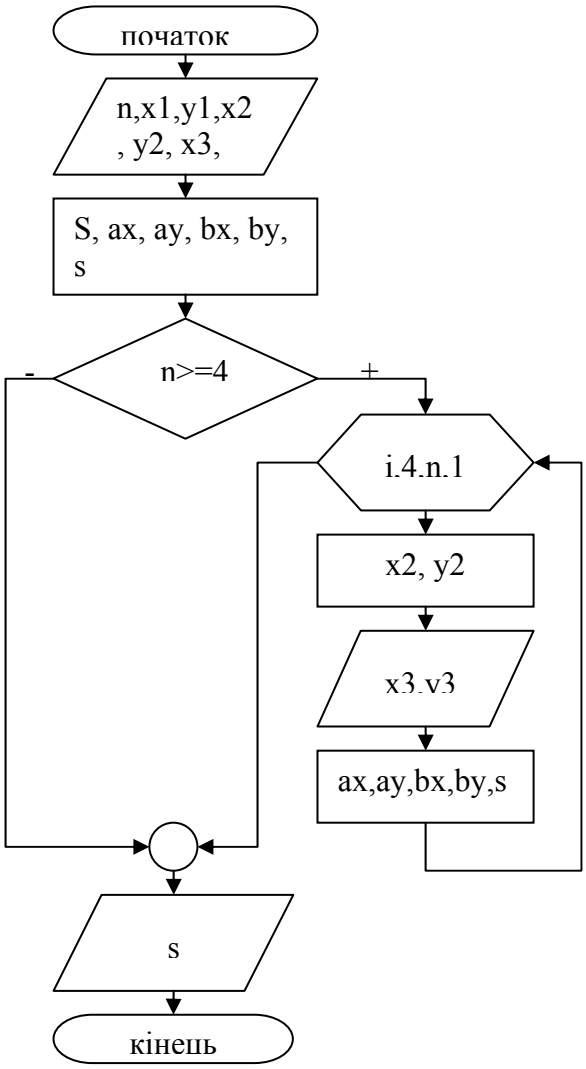


Рис. 25. Блок-схема розв’язку задачі 25.

```
var n, i : integer;
   ax,ay,bx,by,x1,y1,x2,y2,x3,y3, s : real;
begin
  write('Vvedit n '); readln(n);
  write('Vvedit poparno koordunaty');
  readln(x1,y1);  readln(x2,y2); readln(x3,y3);
  s:=0;                                  {знаходимо площу першого трикутника}
   ax:=x2-x1;  ay:=y2-y1;  bx:=x3-x1; by:=y3-y1;
   s:=s+abs(ax*by-ay*bx)/2;
  if n>=4 then
  for i:=4 to n do       {починаючи з четвертої вершини: першу вершину}
  begin                          {залишаємо без змін, 2–>3, а кожну наступну}
    x2:=x3; y2:=y3;   {зчитуємо як третю, і для кожної шукаємо свою}
    readln(x3,y3);                {площу, і додаємо до попередньої}
     ax:=x2-x1;   ay:=y2-y1;
     bx:=x3-x1;   by:=y3-y1;
     s:=s+abs(ax*by-ay*bx)/2;
  end;
  writeln('s=',s:5:2);
end.
```

Задача 26.

Обчислити методом трапецій площу фігури, обмеженої кривими: $Y=X^2$ и $Y= X^4$

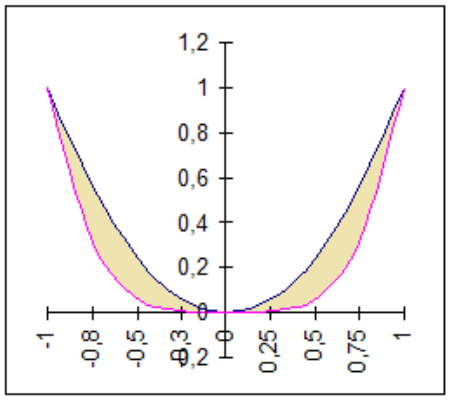


Рис. 26. Геометричне зображення задачі 26.

Нагадаємо з математичного аналізу формулу для знаходження площі фігури методом трапеції: $\int_a^b f(x)dx \approx \frac{b-a}{2n} \cdot \left(f(a) + f(b) + 2\sum_{k=1}^{n-1} x_k \right)$.

У нашому випадку графік симетричний відносно осі ОУ, тому можемо знайти площу з однієї сторони а потім помножити результат на 2. А з правої сторони, як бачимо, графіки перетинаються в точках 0 та 1, тому інтеграл шукаємо від 0 до 1. І як відомо з курсу алгебри та початків аналізу, площа фігури, обмеженої лініями, буде обчислюватись як інтеграл від функції $f(x)=x^2-x^4$. Отже, маємо формулу: $S = 2\int_0^1 \left(x^2 - x^4\right)dx$

```
    var a, b, s,x : real;
   i : integer;
begin
  a:=0; b:=1; s:=0;
  for i:=1 to 9 do
  begin
     x:=a+i/10;
     s:=s+(sqr(x)-sqr(sqr(x)));
  end;
s:=(b-a)/20*(sqr(a)-sqr(sqr(a))+sqr(b)-sqr(sqr(b))+2*s);
  s:=s*2; writeln('s=',s:10:3);
end.
```

Цикл з передумовою

Задача 27.

Дано натуральне число N. Визначити кількість цифр в числі.

```
  Var N : longint; k : integer;
Begin
  Write('Введіть число: '); Readln(N); k := 0;
  if N=0 then k:=1 else
  While N > 0 do
   Begin
    k:=k+1;                          {Підрахунок кількості цифр}
    N:=N div 10;                     {Відкидання останньої цифри}
   End;
  Writeln('Кількість цифр у заданому числі дорівнює', k);
End.
```

Задача 28.

Коли Василині Премудрій виповнилося 18 років, Чахлик Невмирущий вирішив взяти її заміж. Василина запитала Чахлика, скільки у нього скринь із золотом. Чахлик сказав, що в нього зараз *n* скринь і щороку додається ще по *m* скринь. Василина пообіцяла, що вийде заміж тоді, коли у Чохлика буде *k* повних скринь із золотом. Скільки років буде тоді нареченій?

```
Var m,n,k:word;  Sum,Years:word;
Begin
  Write('Введіть початкову кількість скринь з золотом:  '); Readln(n);
  Write('Введіть щорічний прибуток Чахлика:  ');  Readln(m);
  Write('Введіть "потреби" Василини Премудрої:  ');  Readln(k);
  Sum:=n;              Years:=18;
  While Sum<=k do
   Begin
     Sum:=Sum+m; Years:=Years+1;
   End;
  Writeln('Василині вже виповнилося ',Years,' років.');
End.
```

Задача 29.

Дано натуральне число *n*. Визначити суму цифр в числі.

Для розв'язку цієї задачі використаємо такий штучний прийом: Щоб знайти суму цифр, ми повинні "брати" цифри по одній і додавати їх одна до одної, а потім використану цифру "відкидати". На мові Паскаль це нам дозволять зробити операції ділення націло та знаходження залишку від цілочисельного ділення. Так, при діленні числа націло на 10 остання цифра числа буде "відкидатися", а при знаходженні залишку від ділення націло ми виділяємо останню цифру числа. Тобто:

123 div 10 = 12; 3928 mod 10 = 8.

Процес буде повторюватись, доки від числа "нічого не залишиться", тобто, доки воно не перетвориться на нуль. Програма, що реалізує описаний алгоритм має наступний вигляд:

```
Var n:longint;  Sum:byte;
Begin
 Sum:=0;
 Write('Введіть ціле число:  ');  Readln(N);  N:=abs(N);
 While N>0 do
  Begin
   Sum:=Sum+N mod 10;                    {Знаходження суми цифр}
   N:=N div 10;                {"Відкидання" останньої цифри числа }
  End;
 Writeln('Sum= ',Sum);
End.
```

Задача 30.

Визначити число, яке отримується записуванням у зворотному порядку цифр заданого натурального числа. Дане число у записі містить не більше 6 цифр.

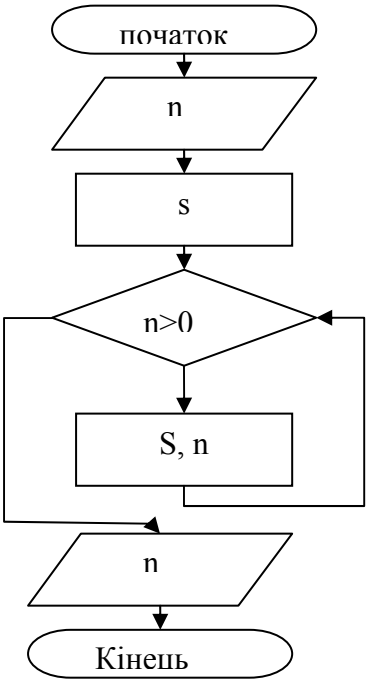


Рис. 27. Блок-схема розв'язку задачі 30.

```
var i, s, n : longint;
begin
  write('Введіть натуральне число: ');  readln(n);
  s := 0;
while n > 0 do                    {виконуємо цикл доки число більше 0}
  begin                           {новоутворене число *10 і додаємо}
    s := 10*s + n mod 10;         {останню цифру початкового числа}
   n := n div 10;                 {«відкидаємо» останню цифру}
  end;
  writeln('Число записане навпаки = ',s);
end.
```

Цикл з післяумовою

Задача 31.

Перевірка коректності введення. Дани три числа, що задають величини кутів трикутника. Визначити, чи можна побудувати трикутник, що має задані кути. Якщо ні, примусити користувача ввести інші дані.

```
  Var a,b,c : real;
Begin
  Repeat
    Write('Введіть довжини сторін трикутника: '); Readln(a,b,c);
  Until (a>0)and(b>0)and(c>0)and(a+b+c)=180;
End.
```

Задача 32.

Дано натуральне число *n*. Визначити кількість цифр у цьому числі.

Для розв'язання цієї задачі можна використати як цикл з передумовою, так і цикл с післяумовою. Однак, на наш погляд, другий варіант кращий, тому що навіть число "0" має у своєму складі одну цифру, а цикл з передумовою цей випадок пропустить. Справа в тому, що умовою виходу з циклу і в тому, і в другому випадку буде "зникнення" числа, тобто перетворення його на нуль після відкидання чергової цифри, а, якщо число с самого початку дорівнює "0", то цикл з передумовою не виконається ні разу, а цикл с післяумовою виконається обов'язково і підрахує одну цифру. Програма для розв'язання цієї задачі має наступний вигляд:

```
Var N:longint;    Count:byte;
Begin
  Write('Введіть натуральне число: ');
  Readln(N);   N:=abs(N);   Count:=0;
  Repeat
    Count:=Count+1;
    N:=N div 10;
  Until N = 0;
  Writeln('Кількість цифр в числі = ',Count);
End.
```

Задача 33.

На скільки років необхідно покласти в банк суму *X* грошових одиниць, щоб одержати суму *N* грошових одиниць (*N* > *X*), якщо банк нараховує 200% річних?

Очевидно, що умовою виходу з цього циклу буде отримання заданої суми грошей. Якщо за умовою задачі *N* > *X*, то кожну перевірку ми будемо виконувати після того, як до вкладеної суми додамо щорічний банківський процент.

```
Var X,N:real;  Rez:real;  Years:longint;
Begin
 Write('Введіть початкову суму вкладу: ');   Readln(X);
 Write('Введіть бажану суму вкладу: ');   Readln(N);
 If N<=X
 Then writeln('Ви вже маєте бажану суму!')
 Else
  Begin
   Rez:=X;  Years:=0;
   Repeat
    Rez:=3*Rez;  {200% річних збільшують за рік вклад втричі}
    Years:=Years+1;
   Until Rez>=N;
   Writeln('Ви отримаєте бажану суму через ',years,' років.');
  End;
End.
```

Одновимірні масиви

Задача 34.

Дано одновимірний масив цілих чисел A[i], де i = 1,2,…n. Вивести елементи масиву у зворотному порядку.

```
 Var N,i:word;  A:array [1..100] of longint;
Begin
Write ('Введіть кількість елементів масиву (<100): ');  Readln(N);
 For i:=1 to N do Readln(A[i]);
Writeln;                    {Переведення курсору на наступний рядок}
 For i:=N downto 1 do Write (A[i]:5);
End.
```

Задача 35.

Дано одновимірний масив цілих чисел *A[i],* де *i = 1,2,…n.* Вивести елементи масиву з парними індексами.

В даному випадку незручно користуватися для виведення на екран елементів з парними індексами циклом з параметром, тому що він дозволяє зміну індексу тільки на одиницю. Тому пропонуємо скористатися циклом з перед або післяумовою.

```
   Var N,i : word;   A : array [1..100] of longint;
Begin
  Write ('Введіть кількість елементів масиву (<100): ');  Readln (N);
 For i:=1 to N do Readln(A[i]);
   Writeln;   i:=2;
  while i<=N do
   Begin
     Write(A[i]:5);
     i:=i+2; {Змінна циклу змін. на 2, щоб вибр. тільки парні елементи}
    End;
  End.
```

Задача 36.

Дано натуральне число А. Складіть програму, що представляє його у вигляді многочлена. Наприклад, 123 ==>1 * 10^2 + 2 * 10^1 + 3*10^0.

Ця задача фактично зводиться до пошуку окремих цифр числа. Так як ми не знаємо на початку роботи, скільки цифр має число, для їх зберігання можна використати масив цілих чисел, причому розмірність цього масиву можна задати не більше 10 елементів, тому що навіть найбільше ціле число типу *longint* має в своєму складі не більше 10 цифр.

Щоб вивести на екран отриманий многочлен, ми спочатку знаходимо кількість цифр в числі, а потім організовуємо цикл від найстаршої значущої (ненульової) цифри числа до молодшої з виведенням на екран самої цифри, помноженої на 10 в степені номер розряду -1 (тобто i-1).

```
Var N,i,k:longint; Cifra:array [1..10] of byte;
Begin
  k:=0;  Write ('Введіть ціле число: '); Readln (N);
 While N>0 do
 Begin
    Cifra[i]:=N mod 10;
    N:=N div 10;   k:=k+1;
 End;
 Write('N = ');
 For i:=k downto 1 do
 Begin
   Write(Cifra[i],'*10^',i-1);       If i>1 Then write(' + ');
{Якщо доданок не останній, то до нього дописується знак "+"}
 End;
End.
```

Задача 37.

В даному масиві з n дійсних чисел знайти добуток перших п‘яти від‘ємних елементів, які мають непарні індекси.

```
var a:array [1..1000] of real;    s:real;  i,n,k:integer;
begin
    s:= 1;   writeln('vvedit k-st chisel');  readln(n);
   for i:=1 to n do                          {Вводимо елементи масиву у циклі}
```

```
  begin    writeln('vvedit ',i,' element');  readln(a[i]);  end;
 for i:=1 to n do                          {Для кожного елементу масиву}
 begin
    if (a[i]<0) and (i mod 2=1) then                  {перевіряємо необхідні
умови}
    begin
        s:=s*a[i];   inc(k);
    end;
     if k=5 then break                    {Якщо  5 елем. то виходимо з циклу}
    end;
    writeln('Dobutok 5 elementiv = ',s:6:0) ;
end.
```

Двовимірні масиви

Задача 38.

Дано натуральні числа n, m та випадкові дійсні числа, що утворюють таблицю *A[i,j], де i = 1,2,…, n; j = 1, 2, …, m.* Роздрукувати у рядок елементи, що розташовані в першому стовпчику.

В даній задачі, хоча таблиця задана двовимірна, другий індекс у всіх елементів, що будуть друкуватися, дорівнює 1, тому достатньо одного циклу по рядках для виконання задачі. Зверніть увагу, що для заповнення масиву повністю необхідні два цикли по рядках та стовпчиках.

```
Const n = 10; m = 8;
Var A: array[1..n,1..m] of real;   i,j: integer;
Begin
   Randomize;
   For i:=1 to n do
   Begin
     For j:=1 to m do
     begin
       A[i,j]:=random*50-random*30;  {Запов. масиву випадк. числами}
          Write(A[i,j]:8:2);
      end;   writeln;
   End;
   Writeln ('Перший стовпчик масиву:');
   For i:=1 to n do  Write (A[i,1]:8:2);
End.
```

Задача 39.

Дано натуральні числа *n, m* та випадкові дійсні числа, що утворюють таблицю *A[i,j], де i = 1,2,…, n; j = 1, 2, …, m.* Роздрукувати у рядок елементи, що розташовані на бічній діагоналі.

Нагадуємо, що на бічній діагоналі елементи мають таку властивість: сума номера рядка та номера стовпчика дорівнюють розмірності масиву *+1*, тобто номер стовпчика обчислюється за наступною формулою: *j = n-1+i.*

```
Const n = 10;
Var A: array[1..n,1..n] of real;   i,j: integer;
Begin
    Randomize;
   For i:=1 to n do
   Begin
      For j:=1 to m do
      begin
         A[i,j]:=random(500)/11-random*30;
         Write (A[i,j]:8:2);
      end;  writeln;
   End;
   Writeln ('Бічна діагональ масиву:');
   For i:=1 to n do
   Write (A[i,n-i+1]:8:2);
End.
```

Задача 40.

Дано квадратну матрицю розмірності n. Надрукувати суму елементів бічної діагоналі.

Розв'язок задачі являється тривіальним, якщо згадати, яку залежність мають індекси бічної діагоналі *(i+j=n+1)*. Перевіривши цю залежність в середині циклів, що організовують прохід по масиву, ми знайдемо бажану суму.

```
Const n = 10;
Var A: array[1..n,1..n] of real;
   i,j:integer; Sum:real;
Begin
   Randomize;
   For i:=1 to n do
   Begin
      For j:=1 to n do
      begin
         A[i,j]:=random*50-random(80)/3;
         Write(A[i,j]:8:3);
      end;  writeln;
   End;
   Sum:=0;                                  {Початкове значення суми}
   For i:=1 to n do
      For j:=1 to n do
         if i + j = n+1 then Sum:=Sum+A[i,j];
   Writeln ('Сума елементів бічної діагоналі =',Sum:8:2);
End.
```

Зверніть увагу на те, що для цієї задачі можна значно спростити цикл знаходження суми, адже фактично ми розглядаємо тільки лінійний масив (елементи на діагоналі дійсно складають одновимірний масив). Тому цикл

знаходження суми можна зробити таким чином (наведений фрагмент програми):

```
Sum:=0;
For i:=1 to n do Sum:=Sum+A[i,n+1-i];
```

Задача 41.

В масиві D[m][n] дійсних чисел знайти суму всіх натуральних чисел.

```
var a:array[1..100,1..100] of real;
b:real; i,j,n,m,s:integer;
begin
     writeln('vvedit rozmir matruci');  readln(n,m);
    for i:=1 to n do
      for j:=1 to m do
        begin
                  writeln('vvedit ',i,',',j,' element'); readln(a[i,j]);
                 b:=(frac(a[i,j])); {знаходимо дробову частину числа}
                if (b=0) and ((a[i,j])>0) then {якщо вона =0 і число більше 0}
                s:=s+trunc(a[i,j]);     {то додаємо до суми}
            end;
          writeln('suma naturalnyh chisel = ',s); readln;
end.
```

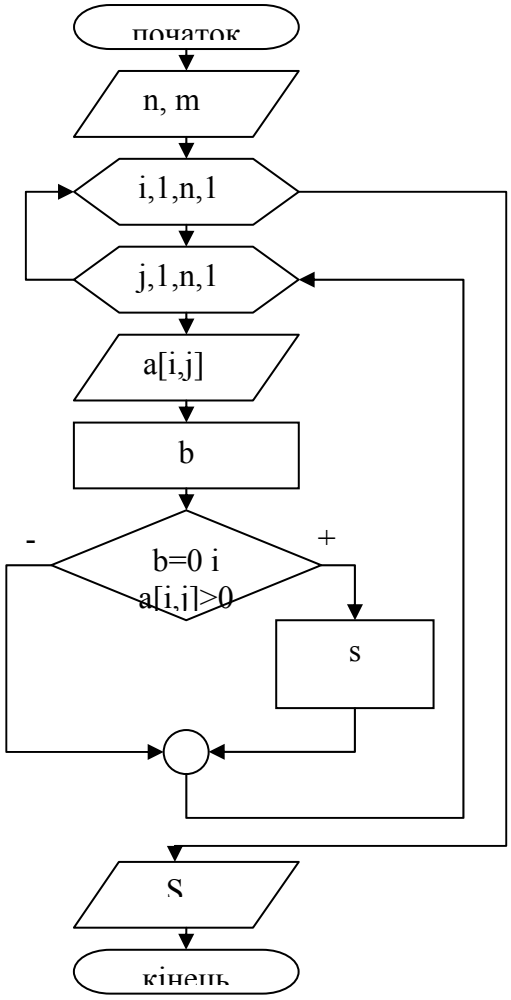


Рис. 28. Блок-схема розв'язку задачі 41.

Упорядкування масивів

Задача 42.

Упорядкувати масив методом бульбашки у порядку зростання.

```
uses crt;
var n, i, j, k:integer;                    {кількість та індекси елементів}
   a: array[1..10] of integer;
   tmp: integer;                           {допоміжний елемент для обміну}
begin
clrscr;
writeln('enter number of the elements <=10');
readln(n);
writeln('enter elements');
for i:=1 to n do                           {введення масиву}
 read(a[i]);
writeln('sort process');
for i:=1 to n do            {від першого елемента до останнього виконуємо}
   for j:=1 to n-1 do       {від першого ел. до передостаннього виконуємо}
      if a[j]>a[j+1] then {якщо один елемент більший за попередній, тоді}
      begin
         tmp:= a[j];                                  {міняємо їх місцями}
         a[j]:= a[j+1];
         a[j+1]:= tmp;
      end;
for k:=1 to n do                  {виведення проміжних результатів}
 write(a[k],' ');
 writeln;
end;
writeln('sorted array');
for i:=1 to n do                           {вивести відсортований масив}
 write(a[i],' ');
readkey
end.
```

Задача 43.

В даному масиві з 32 дійсних чисел суму всіх елементів, які знаходяться на парних місцях після сортування по зростанню.

```
  var a:array [1..100] of real;
        s,b:real;  i,j,n:integer;
    begin
       s:=0;   n:=32;
       for i:=1 to n do readln(a[i]);
       for j:=1 to n-1 do                  {Сортуємо елементи масиву
      метод.}
          for i:=1 to n-1 do               {бульбашки}
          begin
```

```
        if a[i]>a[i+1] then
       begin
         b:=a[i+1];  a[i+1]:=a[i];  a[i]:=b;
       end;
     end;
    for i:=1 to n do
     if (i mod 2)=0 then s:=s+a[i];        {якщо на парному місці то
сумуємо}
   writeln('suma elementiv = ',s:6:2) ;
  end.
```

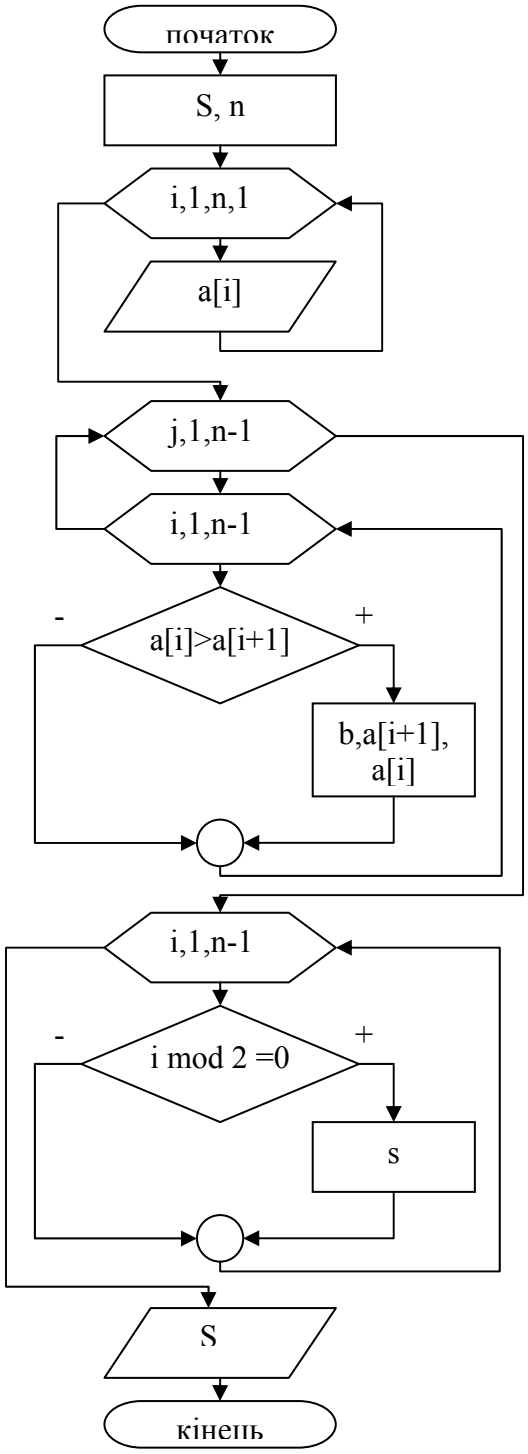


Рис. 29. Блок-схема розв'язку задачі 43.

Графіка

Задача 44.

Вивести на екран монітора наступний малюнок:

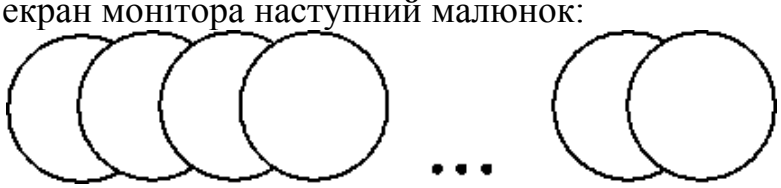


Рис. 30. Умова задачі 44.

Для побудови малюнку перш за все необхідно проініціалізувати графічний режим роботи монітора. Це робиться за допомогою процедури *initgraph*. Далі необхідно задати початкове положення першого кола, тобто координати його центра. Якщо рядок кіл ми хочемо отримати посередині екрану, то координата y повинна дорівнювати 240 (розмір всього екрану по вертикалі 480), а *x* - не менше радіусу кола. Кількість кружечків, що необхідно отримати на екрані, може бути розрахована з урахуванням радіусу кола та відстані між центрами кіл, а може бути задана константно.

В запропонованому розв'язку радіус кола та їх кількість задається з клавіатури, а відстань між колами вираховується в залежності від розмірів екрану (640 пікселів по горизонталі) та кількості кіл).

```
Uses crt,graph;                          {Підключення бібліотек}
Var GraphDriver,GraphMode:integer;
   x,y,R,i,N,S:integer;
Begin
  Clrscr;
  Write('Введіть радіус кола:  ');
  Readln(R);
  Write('Введіть кількість кіл:  ');
  Readln(N);
  S:=640 div N;                                {Відстань між колами}
  x:=R;   y:=240;       {Початкові координати центра першого кола}
  GraphDriver:=VGA;                {Ініціалізація графічного режиму}
  GraphMode:=VGAHi;
  InitGraph(GraphDriver,GraphMode,'');
  For i:=1 to N do
    Begin
      Circle(x,y,R);
      x := x + S;                        {Зсув центра кола по горизонталі}
    End;
  Readkey;
  Closegraph;                             {Закриття графічного режиму}
End.
```

Задача 45.

Вивести на екран монітора наступний малюнок:

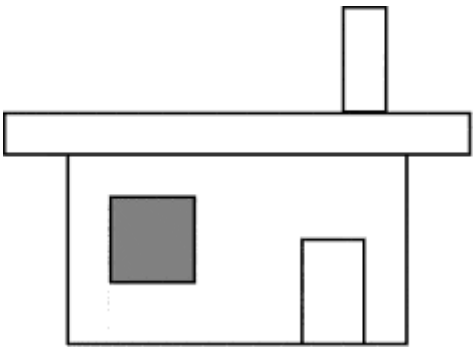


Рис. 31. Умова задачі 45.

Для побудови даного малюнку можна скористатися процедурами *Rectangle*, що малює на екрані незафарбований прямокутник, та *bar*, що малює зафарбований прямокутник. Сам малюнок буде зображуватись червоним кольором, який встановлений процедурою *setcolor*, а віконце буде зафарбовано жовтим кольором, що заданий процедурою *setfillstyle*. Фігури на малюнку не повторюються, тому вихідний алгоритм буде лінійним. Програма, що реалізує цей алгоритм, має наступний вигляд:

```
Uses graph;                          {Підключення графічної бібліотеки}
Var GraphDriver,GraphMode:integer;
Begin
  GraphDriver:=VGA;                  {Ініціалізація графічного режиму}
  GraphMode:=VGAHi;
  InitGraph(GraphDriver,GraphMode,'');
  setcolor(red);
  {Малювання будиночка, стелі, труби та дверей відповідно}
  rectangle(100,250,500,450);
  rectangle(50,200,550,250);
  rectangle(400,20,450,200);
  rectangle(360,300,470,450);
  setfillstyle(1,yellow);   {Зад. тип зафарб. (1-суцільне) та колір зафарб.}
                                {Малювання зафарбованого віконечка}
  bar(140,290,210,360);
  Readln;
  Closegraph;                        {Закриття графічного режиму}
End.
```

Задача 46.

Скласти програму, яка при натисканні клавіші Д (день) малює сонце, а при натисканні клавіші Н (ніч) малює місяць.

По-перше, для вибору малюнку (день чи ніч) введемо символьну змінну Ch, в залежності від значення якої і будемо малювати сонце чи місяць.

По-друге, зауважимо, що малювання сонця складається з малювання зафарбованого кола процедурою *FillEllipse* (ця процедура малює зафарбований еліпс, але, якщо еліпс має однакові радіуси по осям, то він перетворюється на коло) та кількох прямих (променів), а місяць можна отримати, якщо накласти одне на одне два кола різних кольорів (жовтого та чорного) з деяким зміщенням.

Програма, що реалізує запропонований алгоритм, має наступний вигляд:

```
Uses graph,crt;
Var GraphDriver,GraphMode:integer;   Ch:char;
Begin
 Clrscr;
 Writeln('Введіть Ваш вибір: Д – день, Н – ніч.');   Readln(ch);
 GraphDriver:=VGA;    GraphMode:=VGAHi;
 InitGraph(GraphDriver,GraphMode,'');
 if (Ch='Д') or (Ch='д')
 then
  begin
   setfillstyle(1,yellow);  setcolor(yellow);
   fillellipse(100,80,50,50);                    {Малювання сонця}
   line(100,80,250,80);                          {Малювання променів}
   line(100,80,240,30);
   line(100,80,200,250);
   line(100,80,230,180);
   line(100,80,150,250);
   line(100,80,100,300);
   line(100,80,50,380);
   line(100,80,20,280);
   line(100,80,0,150);
   line(100,80,0,80);
   line(100,80,0,30);
   line(100,80,10,0);
   line(100,80,50,0);
   line(100,80,100,0);
   line(100,80,150,0);
  end
 else
  if (Ch='Н') or (Ch='н')
  then
  begin
   setfillstyle(1,yellow);
   setcolor(yellow);
   fillellipse(100,80,50,50);
   setfillstyle(1,black);
   setcolor(black);
   fillellipse(130,80,50,50);
  end
  else writeln('Ви помилилися!');
 Readkey;
 Closegraph;
End.
```

Задача 47.

"Зоряне небо". Заповнити екран монітора різнокольоровими точками, кількість яких, колір та координати визначаються випадковим чином.

Для вибору випадковим чином вказаних величин скористуємось функцією *Random*, що вибирає числа з заданого діапазону, причому врахуємо, що, якщо в дужках після функції вказане ціле число, то будуть генеруватися цілі числа в діапазоні від 0 до вказаного числа. Зверніть увагу на те, що всього можливих кольорів 16 (від 0 до 15), але на чорному тлі чорний колір (з нульовим номером) не видимий, тому можна скористатися такою формулою для отримання ненульових цілих чисел в діапазоні від 1 до 15:

random(14)+1

Аналогічно можна вибрати координати та кількість "зірок" (точок) на екрані, причому відслідкувати, щоб кількість ніколи не була нульовою. Сама "зірка" (точка) на екрані може бути отримана процедурою *Putpixel*, що задає колір та координати точки виведення.

Програма, що реалізує запропонований алгоритм, має наступний вигляд:

```
Uses graph;
Var GraphDriver,GraphMode:integer;
   x,y,color,N:integer;
   i:integer;
Begin
 Randomize;
 GraphDriver:=VGA;
 GraphMode:=VGAHi;
 InitGraph(GraphDriver,GraphMode,'');
 N:=random(1000)+200; {Генерується к-сть точок в діап. [200;1200]}
 for i:=1 to N do
 begin
  x:=random(640);
  y:=random(480);
  color:=random(14)+1;
  putpixel(x,y,color);      {Виведення пікселя заданого кольору color у
задані координати екрану x та y}
 end;
 Readkey;
 Closegraph;
End.
```

Літерні величини

Задача 48.

У даному тексті всі послідовності крапок замінити на одну крапку.

Для розв'язку цієї задачі пропонується організувати цикл перегляду кожного елементу рядка (краще за допомогою циклу з передумовою),

причому, якщо буде знайдена група крапок, вилучити її повністю, а потім вставити на це ж місце тільки одну крапку. Для вилучення групи крапок можна скористатися кількома способами. Наприклад, підрахувати їх кількість, а потім всі вилучити.

Ми пропонуємо вилучати всі крапки, доки не зустрінеться символ, що не являється крапкою.

```
Var i:word;                                    {i - змінна циклу}
    St:string;                                 {St - даний текст}
Begin
    Write ('Введіть текст: '); Readln (St);
    i:=1;
    While i<=length(St) do
    Begin
        While St[i]='.' do Delete(St,i,1);
        Insert('.',St,i);
        i:=i+1;
    End;
    Writeln ('Результуючий рядок: '); Writeln (St);
End.
```

Задача 49.

Дано деякий текст, у якому є хоча б одна кома. Визначити порядковий номер останньої коми в тексті.

Для пошуку останньої коми в тексті необхідно організувати цикл перегляду рядка з кінця до початку, тобто від останнього символу рядка з номером *length(St)* до першого. В зв'язку з тим, що ми не знаємо, якою за номером буде кома, краще це зробити командою повторення з передумовою. Цикл завершить свою роботу при досягненні позиції, в якій знаходиться кома (за умовою вона обов'язково є в тексті). Значення змінної циклу і буде номером шуканої позиції.

Програма, що реалізує описаний алгоритм, має наступний вигляд:

```
Var i:byte;
   St:string;
Begin
   Write ('Введіть текст: '); Readln(St);
   i:=length(St);
   while St[i]<>',' do i:=i-1;
   Writeln ('Номер позиції останньої коми в тексті ',i);
End.
```

Задача 50.

Дано деякий текст. Створити новий текст, який утворено із даного читанням з кінця до початку.

Для реалізації даної задачі знов, як і в попередньому випадку, пройдемо рядок з кінця до початку, виконуючи конкатенацію кожного наступного символу до нового рядка.

Програма, що реалізує описаний алгоритм, має наступний вигляд:

```
Var i:byte; St,Rez:string;
Begin
    Write ('Введіть текст: '); Readln (St);
    Rez:='';                                    {Очищення рядка}
    For i:=length(St) downto 1 do
        Rez := Rez+St[i];
    Writeln ('Результуючий рядок: '); Writeln (St);
End.
```

Задача 51.

Перевірити, чи однаково читається дане слово зліва направо і навпаки.

Для розв'язку цієї задачі можна використовувати попередню задачу, як підзадачу, тобто спочатку отримати новий рядок, який являється оберненим відносно даного, а потім порівняти даний та отриманий рядки. Якщо вони співпадають, слово являється паліндромом (читається в обох напрямках однаково, наприклад, "дід", "потоп", "Пилип" тощо), в протилежному випадку - ні.

Програма, що реалізує описаний алгоритм, має наступний вигляд:

```
Var i:byte; St,Rez:string;
Begin
   Write ('Введіть текст: '); Readln (St); Rez:='';
   For i:=length(St) downto 1 do
      Rez := Rez+St[i];                         {Перегортання рядка}
   If Rez = St Then Writeln ('Слово являється паліндромом.')
   Else Writeln ('Слово не являється паліндромом.');
End.
```

Задача 52.

Нехай дано текст *S* та значення символьних змінних *x* та *y*. Із тексту вилучити всі символи, що збігаються з *x* і повторити двічі всі символи, що збігаються з *y*.

Для розв'язання запропонованої задачі необхідно переглянути кожен символ рядка і, якщо він співпадає з символом *x*, вилучити його процедурою ***delete***, а якщо збігається із символом *y* - вставити перед ним такий самий процедурою *insert*. Програма для реалізації цього алгоритму має наступний вигляд:

```
Var i:byte;  S:string; x,y:char;
Begin
    Write ('Введіть текст: '); Readln (S);
    Write ('Введіть шукані символи: '); Readln (x,y);
    For i:=1 to length(S) do
    Begin
       If S[i] = x Then delete(S,i,1);
       If S[i] = y Then insert(y,S,i);
    End;
    Writeln ('Результуючий рядок після зміни: ',S);
End.
```

Задача 53.

Скласти програму яка підрахує кількість заданих букв в даному тексті.

```
var St : string;       obrazec : char;      i, kol : byte;
begin
  write('Введіть речення: '); readln(St);
  write('Яку літеру підраховувати: ');readln(obrazec);
  kol := 0;
  for i := 1 to length(St) do
    if St[i] = obrazec then inc(kol);
  writeln('Шуканих літер в тексті -> ',kol);
end.
```

Задача 54.

Скласти програму, яка з даного тексту видалить всі зайві пропуски та підрахує кількість слів в даному тексті.

```
var St, St1 : string;
    i : integer;
begin
  write('Введіть текст: ');readln(St);
  St1 := St[1];
  for i := 2 to length(St) do
    if st[i] <> ' ' then st1:=st1+st[i]
    else if (st[i] = ' ') and (st[i+1] <> ' ') then
          st1 := St1 + st[i]
  St1 := St1 + ' ';
       {алгоритм підрахунку к-сть слів}
       for i:=1 to length(st1) do
         if st1[i]=' ' then k:=k+1;
  writeln(St1);
end.
```

Задача 55.

Скласти програму, що підраховує кількість слів та заносить всі слова у масив (вважається, що лишні пропуски вже видалені).

```
    VAR st, st1: string;    a : array [1..100] of string;    i, n,n1, k : integer;
    begin
      readln(st);                              {алгоритм підрахунку к-сть слів n1}
      for i:=1 to length(st) do  if st[i]=' ' then k:=k+1;
      {алгоритм занесення слів у масив}
      i:=1; j:=1;
      while i<=k do
      begin
         while st[j]<>' ' do
         begin
          a[i]:=a[i]+st[j];  j:=j+1;
         end;
        j:=j+1;  i:=i+1;
```

```
  end; {виведення масиву (занесених слів) з нового рядка кожне слово}
  for i:=1 to k do
   writeln(a[i]);
end.
```

Процедури

Задача 56.

Нехай в програмі описано такі процедури:

```
Procedure P(x,y:integer);
Begin
    y:=x+1;
End;
Procedure Q(x:integer; var y:integer);
Begin
      y:=x+1;
End;
Procedure R(var x,y:integer);
Begin
      y:=x+1;
End;
```

Визначити, що буде надруковано в результаті виконання таких операторів:

Таблиця 12

Оператори	***Очікувана відповідь***
c:=2; d:=1; P(c,d); writeln(d);	на екрані буде 1
c:=2; d:=1; Q(c,d); writeln(d);	на екрані буде 3
c:=2; d:=1; R(c,d); writeln(d);	на екрані буде 3

Чи припустиме таке звертання до вищезазначених процедур?

Таблиця 13

Оператори	***Очікувана відповідь***
P(sqrt(c),d)	так
P(c,2)	так
Q(sqrt(c),d)	так
R(1,d)	ні
R(c,2)	ні

Пояснення до відповідей наступні: фактичне значення параметра-значення може бути константою, змінною або виразом, а параметра-змінної - тільки іменем змінної.

Задача 57.

Баба-Яга записалася на курси водіїв літальних апаратів. Але справи в неї були кепські, бо вона ніяк не могла запам'ятати, яким чином визначається тривалість польоту, якщо відомі швидкість і відстань. Довелося їй звернутися по допомогу до Хлопчика-Мізинчика, який швиденько написав їй шпаргалку,

куди Бабі-Язі треба було лише підставити свої значення. Як виглядала послідовність дій у цій шпаргалці і як нею користувалася Баба-Яга?

Очевидно, що "шпаргалку" Хлопчика-Мізинчика можна оформити як допоміжний алгоритм. Параметрами, що передаються в цей алгоритм, будуть швидкість літального апарату та відстань, яку необхідно подолати, а вихідним параметром - шукана тривалість польоту. Вхідні параметри процедури повинні бути параметрами-значеннями, а вихідний параметр - параметром-змінною (дивись матеріал попереднього уроку).

Позначимо у підпрограмі формальні параметри наступним чином:

V - швидкість літального апарату;

S - відстань, що необхідно подолати;

T - тривалість польоту.

В основній програмі ті ж самі змінні будуть мати відповідно імена: *X, Y* та *M* (імена змінних у основній програмі бажано, щоб не співпадали з іменами локальних параметрів підпрограми, тому їх вибір являється випадковим).

```
Procedure Solution (V,S - real; var T - time);
Begin
      T:=S/V;
End;
Var X,Y,M:real;
Begin
      Write('Введіть швидкість літального апарату: '); Readln(X);
      Write('Введіть відстань між населеними пунктами: '); Readln(Y);
      If (X<=0) or (Y<0) then writeln('Некоректні вхідні дані.')
      else
      begin
         Solution(X,Y,M);                          {Виклик процедури}
         Writeln('Тривалість польоту -> ',M:6:2);
      end;
End.
```

Задача 58.

Скласти програму знаходження НСК для n чисел з використанням процедури.

```
var a, nsk : longint;       {нам будуть потрібні всього дві змінні: число і НСК}
procedure nsk_2;            { процедура для знаходження НСК двох чисел }
var x, y : longint;         {дод. змінні, значення яких програма не бачить}
begin                                        { початок процедури }
  x := nsk; y := a;                          { зберігаємо вхідні дані }
  while x <> y do                    { і працюємо з внутрішніми змінними }
    if x > y then x := x - y        {процедури, використовуючи вже відомий}
    else y := y - x;                         { нам алгоритм Евкліда для НСД }
  nsk := (nsk div x) * a;                    { і знаходимо НСК  }
end;                                          { кінець процедури }
```

```
begin                                   { головна програма }
  nsk := 1;                    {для першого входження в процедуру nsk_2 }
  a :=  1;                              { для входження в цикл }
  while a <> 0 do              {цикл завершується коли з клавіатури ввели 0 }
  begin
    write('Введіть число: '); readln(a);          {  зчитали число }
    if a<>0 then                        { і якщо воно не нуль }
    begin
      nsk_2;                            { то знаходимо НСК двох чисел }
      writeln('НСК = ',nsk);            { і виводимо значення на екран }
    end;
  end;                                  { кінець циклу, коли ввели 0 }
  writeln('Для закінчення нажміть <Enter>');    { повідомлення }
  readln;                               { чекаємо натиснення <Enter> }
end.                                    { кінець програми }
```

Функції

Задача 59.

Знайдіть і поясніть помилки в записі функцій:

```
Function max(n:integer):real:
    Var a,max:real;
Begin
      Read(max);
      For i:=1 to n-1 do
      Begin
      Read(a);
      If a>max then max:=a;
      End;
End;
```

Очікувана відповідь:

- У функції описана внутрішня локальна змінна max, ім'я якої співпадає з іменем функції, а це неприпустимо, тому що вони обидві являються локальними для даної функції і не можуть мати однакові імена.
- Якщо все ж таки ім'я функції max, то неможливо використання його у операторах read(max) та if a>max, тому що ми отримуємо самовиклик функції, а це може призвести до помилки.
- В операторах read(max) та if a>max помилка, якщо max - ім'я функції, тому що після імені функції в момент її виклику повинні знаходитись в дужках фактичні параметри, кількість та тип яких мають співпадати з кількістю та типом фактичних параметрів даної функції (в даному випадку фактичний параметр повинен бути один).
- Якщо в програмі не існує глобальна змінна і, то вона залишається неописаною в підпрограмі.

Задача 60.

Використовуючи функцію *max2(a,b)*, яка визначає максимальне з двох

даних чисел, записати функцію *max3(a,b,c)*, що визначає максимальне з трьох даних чисел, і організувати виклик цієї функції для обчислення суми найбільших значень трьох трійок довільних дійсних чисел.

Алгоритм пошуку максимуму з двох чисел являється стандартним і його написання, на наш погляд, не може викликати якихось непорозумінь. Написання алгоритму пошуку максимуму з трьох чисел також являється тривіальним і розв'язується за допомогою двох викликів функції пошуку максимуму з двох чисел в такім послідовності:

- знаходиться максимум з двох чисел (наприклад, a та b)<.li>
- знаходиться максимум з вже найденого максимуму та третього числа c.

За умовою необхідно знайти суму найбільших значень трьох трійок довільних дійсних чисел, тому результатом роботи програми буде значення змінної *Rezultat*, в якій ми будемо здійснювати накопичення суми.

Виклик функції виконаємо трьома способами. Перший раз для трьох змінних x, y, z, що будуть введені з клавіатури, другий - для трьох констант, що вибрані нами випадково, а третій - для деяких виразів (наприклад, знайти максимум модулів трьох чисел). Це робиться з навчальною метою, щоб показати, що у якості фактичних параметрів можуть бути не тільки змінні, а й константи або вирази (Зверніть увагу учнів на те, що константами або виразами фактичні параметри можуть бути тільки в тому випадку, коли відповідні формальні параметри являються параметрами-значеннями. Якщо ж формальний параметр є параметром-змінною, то відповідний фактичний параметр теж повинен бути змінною).

З урахуванням всього вище сказаного, програма, що виконує запропонований алгоритм, має наступний вигляд:

```
Function Max2 (a,b:real):real;
begin
    if a > b then Max2:=a else Max2:=b;
end;
Function Max3 (a,b,c:real):real;
Var Max:real;
Begin
    Max:=Max2(a,b);
    Max3:=Max2(Max,c);
End;
Var x,y,z,Rezultat:real;
Begin
    writeln  ('Введіть три довільні числа: ');
    readln (x,y,z);
    Rezultat:=Max3(x,y,z);
    Rezultat:=Rezultat+Max3(2,-4.5,12.54);
    Rezultat:=Rezultat+Max3(abs(x),abs(y),abs(z));
    writeln  ('Результат -> ',Rezultat:8:2);
End.
```

Задача 61.

Скласти програму знаходження НСК для n чисел з використанням функції.

```
var n, i, j, k :longint;
    d : longint;
    x : array[1..21] of byte;
function nod (a,b:longint):longint;      {функція знаходження НСД двох чисел}
var x,y :longint;
begin
  x:=a;
  y:=b;
  while x<>y do
    if x>y then x:=x-y else y:=y-x;
  nod:=x;
end;
begin
   readln(n);
  for i:=1 to n do
    read(x[i]);
    k:=x[1];
    d:=k;
  for i:=2 to n do
   begin
     k:=nod(x[i],d);
    d:=d*x[i];
    d:=d div k;                                  {НСК=доб.чис./НСД}
   end;
  writeln(d);
end.
```

Рекурсія

Задача 62.

Використовуючи підпрограму обчислення факторіалу, розробити програму обчислення суми факторіалів усіх цілих чисел від 1 до 10.

Функція обчислення факторіалу є чи не найстандартнішою, яку приводять в усіх підручниках для пояснення явища рекурсії. Зробимо це й ми.

Очевидно, що

$$n! = \begin{cases} 1 & \text{якщо } n = 0 \\ (n-1)! \cdot n & \text{якщо } n > 0 \end{cases}$$

Тоді вихідна програма буде мати наступний вигляд:

```
Function Factorial (n:integer):longint;
Begin
      if n=0 then Factorial:=1 else Factorial:=Factorial(n-1)*n;
```

```
End;
Var Rez:longint;
    i:byte;
Begin
    Rez:=0;
    For i:=1 to 10 do Rez:=Rez+Factorial(i);
    writeln('Rezultat -> ',Rez);
End.
```

Задача 63.

Знайти найбільший спільний дільник двох натуральних чисел n та m за алгоритмом Евкліда:

$$f(n,m)=\begin{cases} n & \text{якщо } n = m \\ f(n-m,m) & \text{якщо } n > m \\ f(n,m-n) & \text{якщо } n < m \end{cases}$$

В даному випадку умовою виходу з рекурсії буде рівність двох чисел *n=m*. Написання самої програми, на наш погляд, являється тривіальним:

```
Function Evklid (n,m:integer):integer;
begin
    if n=m then Evklid:=n else
    if n>m then Evklid:=Evklid(n-m,m) else Evklid:=Evklid(n,m-n);
end;
Var x,y:integer;
Begin
    writeln ('Введіть два числа: '); readln (x,y);
    writeln ('НОД -> ',Evklid(abs(x),abs(y)));
End.
```

Робота з файлами

Задача 64.

У текстовому файлі input.txt знаходиться число N – кількість цілих чисел, та N цілих чисел, які стоять після N. Вивести у файл output.txt зчитаний масив.

```
var f : text;
    n, i : integer;
   a : array [1..1000] of integer;
begin
  assign(f,'input.txt'); reset(f);  read(f,n);
  for i:=1 to n do
     read(f,a[i]);
   close(f);
   assign (f,'output.txt'); rewrite(f);
   for i:=1 to n do   write(f,a[i],' ');
  close(f);
end.
```

Задача 65.

В текстовому файлі input.txt знаходиться невідома кількість цілих чисел яка не перевищує 1000. Зчитати з текстового файла всі числа та записати у вихідний файл output.txt.

```
var f : text;
    n, i :  integer;
    a : array [1..1000] of integer;
begin
    assign(f,'input.txt'); reset(f);
    n:=0; i:=0;
    while not(eof(f)) do
    begin
       inc(n); inc(i);
      read(f,a[i]);
    end;
   close(f);
   assign(f,'output.txt'); rewrite(f);
  for i:=1 to n do
    writeln(f,a[i]);
  close(f);
end.
```

Множини. Записи

Задача 66. Скласти програму, яка знаходить всі числа, що діляться на 6, а також всі числа, що діляться на 2 або на 3.

```
uses crt;
Const N = 255;
var N2,N3,N6,N23 : set of byte;
   k : integer;
begin
   N2 := [];  N3 := [];
   for k :=1 to N do
   begin
       if k mod 2 = 0 then N2 := N2 + [k];
       if k mod 3 = 0 then N3 := N3 + [k];
   end;
   N6 := N2 * N3;
   N23 := N2 + N3;
   writeln(' На  6 діляться: ');
      for k := 1 to N do if k in N6 then write(k:4);
   writeln;
   writeln(' На 2 або на 3діляться: ');
      for k := 1 to N do if k in N23 then write(k:4);
   writeln; readln;
end.
```

Задача 67. Скласти програму заповнення та редагування бази даних про домашній електронний телефонний довідник.

```
const n    = 10;
     Name : array[1..n] of string[15] = ('Петренко','Козлюк',
'Василенко','Носенко','Деричуб','Вернигора','Котигорошко','Івасюк',
'Іваненко', 'Коваленко');
Type Abonent = Record
                FName : string[15];
                Phone : integer;
               end;
Var Klient : array[1..2*n] of abonent;
   i, kol : integer;   o, m : byte;
Procedure NewKlient;
begin
  inc(kol);
  write('Прізвище нового абонента: '); readln(Klient[kol].FName);
  write('Телефон нового абонента: '); readln(Klient[kol].Phone);
  writeln;
end;
Procedure EditKlient;
begin
  write('Номер по списку абонента: '); readln(m);
  if (m > kol) or (m<1) then writeln('Будьте уважні!')
  else begin
         write('Новий телефон абонента: ');
         readln(Klient[m].Phone);
       end;
  writeln;
end;
Begin
   for i := 1 to n do
    begin
      Klient[i].Fname := Name[i];
      Klient[i].Phone := Random(10001)+20001;
    end;
    kol := n;    writeln;
    repeat
     for i := 1 to kol do writeln(i:2,'.  ',Klient[i].Phone,' - ',Klient[i].Fname);
      writeln;
  write('1-Новий абонент 2-Редагувати телефон 3-Закінчити роботу');
      readln(o);
      if o = 1 then NewKlient;
      if o = 2 then EditKlient;
    until o > 2;
end.
```

Задача 68. Написати модуль для роботи зі списками.

У ньому мають міститись процедури первісної підготовки списку; додавання елемента в початок списку; видалення елемента, що стоїть за зазначеним; пошук елемента з заданим номером; підрахунку елементів і очищення списку.

```
unit Lists;
interface
type  tData = record
        Name: string[50];
        Phone: longint;
      end;
      tItemPtr = ^tItem;
      tItem = record
        Data: tData;
        Next: tItemPtr;
      end;
procedure InitList(var l: tItemPtr);
procedure AddItemToHead(var l: tItemPtr; d: tData);
function  DeleteItemAfter(var l: tItemPtr; num: word): boolean;
function  Count(l: tItemPtr): word;
function  GetItem(l: tItemPtr; num: word; var d: tData): boolean;
procedure ClearList(var l: tItemPtr);
{-------------------------------------------------------------}
implementation
procedure InitList(var l: tItemPtr);
begin l:=nil end;
procedure AddItemToHead(var l: tItemPtr; d: tData);
var p: tItemPtr;
begin
  new(p);
  p^.data:=d;
  p^.next:=l;
  l:=p;
end;
function DeleteItemAfter(var l: tItemPtr; num: word): boolean;
var p,q: tItemPtr;
    i: word;
begin
  i:=1;
  p:=l;
  while (i<>num)and(p<>nil) do begin
    i:=i+1;
    p:=p^.next;
```

```
  end;
  if p<>nil then begin
    if p^.next<>nil then begin
      q:=p^.next^.next;
      dispose(p^.next);
      p^.next:=q;
      DeleteItemAfter:=true;
    end
      else DeleteItemAfter:=false; {не вилучений}
  end
    else DeleteItemAfter:=false;
end;
function Count(l: tItemPtr): word;
var p: tItemPtr;
    i: word;
begin
  i:=0;
  p:=l;
  while p<>nil do begin
    i:=i+1;
    p:=p^.next;
  end;
  count:=i;
end;
function GetItem(l: tItemPtr; num: word; var d: tData): boolean;
var p: tItemPtr;
    i: word;
begin
  i:=1;
  p:=l;
  while (i<>num)and(p<>nil) do begin
    i:=i+1;
    p:=p^.next;
  end;
  if p<>nil then begin
    d:=p^.data;
    GetItem:=true;
  end
    else GetItem:=false;
end;
procedure ClearList(var l: tItemPtr);
var p: tItemPtr;
begin
  while (l<>nil) do begin
    p:=l^.next;
```

```
    dispose(l);
    l:=p;
  end;
end;
end.
```

Задача 69. Для приклада розглянемо опис і реалізацію кільцевого двонаправленого списку:

```
type tItemPtr = ^tItem
     tItem = record
       data: tData;
       next,prev: tItemPtr;
     end;
var List: tItemPtr; {список - покажчик на один з елементів}
.....
{Видалити після зазначеного:}
procedure DelAfter(p: tItemPtr);
var q: tItemPtr;
begin
  if (p<>nil)and(p^.next<>p) then begin
    q:=p^.next^.next;
    dispose(p^.next);
    p^.next:=q;
    q^.prev:=p;
  end;
end;
{Уставити перед зазначеним:}
procedure InsertBefore(p: tItemPtr; d: tData);
var q: tItemPtr;
begin
  if p<>nil then begin
    new(q);
    q^.data:=d;
    q^.next:=p;
    q^.prev:=p^.prev;
    p^.prev:=q;
    q^.prev^.next:=q;
  end;
end;
```

Лабораторний практикум

Загальні вимоги до виконання лабораторних робіт

1. Ознайомитись з темою, метою роботи та засвоїти матеріал теоретичної частини.
2. Користуючись прикладами, які наведені в другому розділі, ознайомитись з практичними аспектами.
3. Вибрати відповідний варіант (за вказівкою викладача) та ознайомитись із завданням.
4. Виконати завдання, що означає виконати наступні етапи:
    a. Опис математичного розв'язку задачі.
    b. Складання блок-схеми розв'язку задачі
    c. Складання алгоритму розв'язання задачі.
    d. Опис алгоритму мовою програмування.
    e. Не менше трьох тестів з відповідями до кожної програми.
5. Зберегти результати виконаної роботи в індивідуальній робочій папці.
6. Продемонструвати результати виконаної роботи викладачу.
7. Оформити звіт про виконання лабораторної роботи, який має включати: назву теми, завданння, хід роботи, короткий ***математичний опис*** програм, блок-схеми до всіх завдань, ***тексти програм***, по ***три тести (мінімум) з відповідями*** до кожної програми та відповіді, файли LabN_M_VAR.pas з робочою програмою (де N – номер лабораторної роботи, M – номер завдання в даній роботі, VAR – номер індивідуального варіванту).
8. Захистити звіт про виконання лабораторної роботи та відповісти на контрольні питання, які зазначені на початку лабораторної роботи.

Середовище програмування FREE Pascal

Лабораторна робота №1

Мета: Набути уміння та навички роботи в середовищі Free Pascal відкриття, виконання, введення, збереження, компіляція, налагодження програми.

Програмне забезпечення*:* Free Pascal, ABCPascal.

Контрольні питання.

1. Що таке компіляція програми? Що таке налагодження програми?
2. Що таке інтерпретатор?
3. Що називають компілятором? Що називають транслятором?
4. Що таке тест?
5. Які визнаєте види помилок?
6. Яка помилка називається логічною, семантичною, синтаксичною?

Практичне завдання.

1. Завантажити середовище програмування Free Pascal.
2. Набути уміння та навички роботи в середовищі Free Pascal: відкриття, виконання, введення, збереження, компіляції, налагодження програми на Pascal.

Лінійні програми

Лабораторна робота №2

Мета: засвоєння найпростішої структури програми мовою PASCAL; отримання навичок в організації введення (виведення) значень стандартних типів даних; вивчення порядку дій при обчисленні виразів; здобуття навичок в запису виразів мовою PASCAL, використанні стандартних функцій; отримання практичних навичок роботи в діалоговому режимі.

Програмне забезпечення*:* Free Pascal, ABCPascal.

Контрольні питання.

1. З чого складається алфавіт на мові Pascal?
2. Що таке ідентифікатори в мові Pascal?
3. Опишіть основні правила запису ідентифікаторів.
4. Що таке величини і які вони є?
5. Яка величина називається константою?
6. Описати константи на мові Pascal?
7. Яка величина називається змінною?
8. Назвіть основні характеристики величин.
9. Що таке символи мови?
10. Що є словами мови? Що називають зарезервованими словами?
11. Що таке коментар? Як записуються коментарі на мові Pascal?

12. Дайте означення поняття виразу.
13. Дайте означення поняття оператора.

Завдання по варіантам.

Варіант 1.

1. Обчислити значення виразу $y = \log_4 \left| x^3 + 2x - \sin x \right| + \cos x$, при умові, що x вводиться з клавіатури користувачем. Результат вивести з точністю до десятих.
2. Вказаний в секундах час виразити в годинах, хвилинах, секундах.

Варіант 2.

1. Обчислити значення виразу $y = e^{2x+3} \cdot \sin x + \cos x - \log_2 \left| 2x^2 - 3x \right|$, при умові, що x вводиться з клавіатури користувачем. Результат вивести з точністю до сотих.
2. За довжинами сторін трикутника знайти міри його кутів у градусах.

Варіант 3.

1. Обчислити значення виразу $z = \log_{|x|} \left| 2y - \sin x + tgy \right| + e^{2x+y}$, при умові, що x та y вводиться з клавіатури користувачем. Результат вивести з точністю до тисячних.
2. Знайти площу рівнобічної трапеції за довжинами основ та градусною мірою кута при більшій основі.

Варіант 4.

1. Обчислити значення виразу $y = \sin\left(x^2 + 5x - 1\right) + \sqrt{2x}$, при умові, що x вводиться з клавіатури користувачем. Результат вивести з точністю до десятих.
2. Знайти периметр та площу трикутника за координатами його вершин.

Варіант 5.

1. Обчислити значення виразу $y=cos^3(2x)-log_2(x^2+2)$, при умові, що x вводиться з клавіатури користувачем. Результат вивести з точністю до десятих.
2. Знайти периметр та площу паралелограма за координатами трьох його вершин.

Варіант 6.

1. Обчислити значення виразу $y=cos^2(2x)-sinx$, при умові, що x вводиться з клавіатури користувачем. Результат вивести з точністю до десятих.
2. Знайти об'єм та площу поверхні куба за довжиною діагоналі.

Варіант 7.

1. Обчислити значення виразу $y = \sqrt{x^3 + 3} - 2\log_2(x^3 + 1)$, при умові, що x вводиться з клавіатури користувачем. Результат вивести з точністю до десятих.
2. За довжинами сторін трикутника знайти радіуси вписаного та описаного кіл.

Варіант 8.

1. Обчислити значення виразу $y = \sqrt{x^3 + 3x} + 2\sin x$, при умові, що x вводиться з клавіатури користувачем. Результат вивести з точністю до десятих.
2. Знайти площу кільця за довжинами його радіусів.

Варіант 9.

1. Обчислити значення виразу $y = \sqrt[5]{x^4 + 2x^3} - 2\sin x$, при умові, що x вводиться з клавіатури користувачем. Результат вивести з точністю до десятих.
2. Знайти об'єм та площу поверхні кулі за довжиною ребра вписаного куба.

Варіант 10.

1. Обчислити значення виразу $y = (x^2 - 2x)^3 - 4\log_2(x^4 + 1)$, при умові, що x вводиться з клавіатури користувачем. Результат вивести з точністю до десятих.
2. Обчислити висоти трикутника зі сторонами a, b, c. Результат вивести з точністю до десятих.

Варіант 11.

1. Обчислити значення виразу $y = 2^{2x} - 4\cos x$, при умові, що x вводиться з клавіатури користувачем. Результат вивести з точністю до десятих.
2. Трикутник задано координатами його вершин $A(x_1;y_1)$, $B(x_2;y_2)$, $C(x_3;y_3)$.Визначити площу трикутника. Результат вивести з точністю до десятих.

Варіант 12.

1. Обчислити значення виразу $y = \sin(3x) + \cos x$, при умові, що x вводиться з клавіатури користувачем. Результат вивести з точністю до десятих.
2. Чотири точки $A(x_1;y_1)$, $B(x_2;y_2)$, $C(x_3;y_3)$, $D(x_4;y_4)$ є вершинами паралелограма. Визначити довжину діагоналей і знайти координати точки їх перетину. Результат вивести з точністю до тисячних

Варіант 13.

1. Обчислити значення виразу $y = \sqrt[3]{x^3 + 2x^2} + 2\sin x$, при умові, що x вводиться з клавіатури користувачем. Результат вивести з точністю до десятих.
2. Відрізок задано координатами його кінців $M(x_1;y_1)$, $N(x_2;y_2)$. Знайти координати точки $O(x,y)$, що ділить його у відношенні α. Результат вивести з точністю до сотих.

Варіант 14.

1. Обчислити значення виразу $y = \sqrt[3]{x^5 + 5x^4} - 4x^{\frac{2}{3}}$, при умові, що x вводиться з клавіатури користувачем. Результат вивести з точністю до десятих.
2. Розв'язати систему двох лінійних рівнянь з двома невідомими за формулами Крамера. Результат вивести з точністю до тисячних.

Лабораторна робота №3

Мета: Набути уміння та навички складання лінійних програм на мові Pascal.

***Програмне забезпечення**:* Free Pascal, ABCPascal.

Контрольні питання.

1. Що таке алгоритм? Назвіть його основні властивості та способи опису.
2. Яка структура програми на мові Pascal ?
3. Яка програма називається лінійною?
4. Які оператори мови використовуються для введення та виведення даних? Описати синтаксис таких операторів.
5. Які типи даних використовуються в мові Pascal, їх обмеження?
6. Які математичні операції та функції використовуються в мові Pascal?
7. Назвіть основні етапи розв'язання задачі на ПК.
8. У чому полягає різниця між константами та змінними? Які типи констант та змінних використовують у програмах?
9. Поняття змінної. Який синтаксис має ім’я змінної? Як описується змінна для використання у програмі? На прикладі описати виконання вказівки присвоєння.
10. Поняття виразу та операції. Які є види операцій та виразів?
11. Які арифметичні операції, операції відношення та логічні операції відношення можна реалізувати у програмі?
12. За яким пріоритетом виконуються операції у програмах?
13. Описати синтаксис стандартних числових функцій.

Завдання по варіантам.

Варіант 1.

1. Знайти суму, різницю, добуток і частку комплексних чисел в алгебраїчній формі □=*a*+*bi,* □=*c*+*di*, де *a, b , c, d* - дійсні числа.
2. Записати дане чотирицифрове натуральне число без середніх цифр. (*)

Варіант 2.

1. Визначити найменший кут між векторами $n(x_1;y_1),m(x_2;y_2)$.
2. Знайти суму крайніх цифр даного чотиризначного натурального числа.

Варіант 3.

1. Визначити найкоротшу віддаль від точки $A(x_1;y_1)$ до кола радіусом *R* з центром *C(x, y)*.
2. В даному натуральному чотиризначному числі поміняти місцями середні цифри. (*)

Варіант 4.

1. Визначити координати точок перетину прямої $ax+by+c=0$ з колом радіусом *R* і центром *C(x,y)*.
2. Записати дане трицифрове натуральне число без середньої цифри. (*)

Варіант 5.

1. Дано два кола $(x-a)^2+(y-b)^2=R_1^2$; $(x-c)^2+(y-d)^2=R_2^2$ що не перетинаються. Визначити найкоротшу віддаль між ними та віддаль між центрами цих кіл.

2. Знайти остачу від ділення першої цифри на останню в даному натуральному трицифровому числі.

Варіант 6.

1. Дано коло $(x-a)^2+(y-b)^2=R_1^2$ і пряма $ax+by+c=0$, що не перетинаються. Визначити віддаль між ними.
2. Записати дане трицифрове натуральне число в зворотному порядку (*)

Варіант 7.

1. Дано коло $(x-a)^2+(y-b)^2=R_1^2$ і пряма $ax+by+c=0$, що перетинає коло. Визначити довжину хорди.
2. Розкласти дане трицифрове число на цифри. Вивести кожну цифру з нової стрічки.

Варіант 8.

1. Пряма $y=a$ перетинає трикутник з вершинами $A(0;0)$, $B(0;y_2)$, $C(x_3;0)$. Знайти співвідношення площ двох фігур, утворених при цьому.
2. Дано трицифрове число знайти добуток його цифр.

Варіант 9.

1. Дано довжину ребра куба. Знайти об'єм куба та площу бічної повірхні.
2. Знайти квадрат суми цифр двоцифрового натурального числа.

Варіант 10.

1. Трикутна піраміда задана координатами своїх вершин $A(x_1;y_1;z_1)$, $B(x_2;y_2;z_2)$, $C(x_3;y_3;z_3)$, $D(x_4;y_4;z_4)$. Визначити площу повної поверхні піраміди. Результат вивести з точністю до десятих
2. В даному трицифровому натуральному числі поміняти першу і останню цифри місцями. (*)

Варіант 11.

1. Визначити площу чотирикутника з вершинами $A(x_1;y_1)$, $B(x_2;y_2)$, $C(x_3;y_3)$, $D(x_4;y_4)$. Результат вивести з точністю до цілих .
2. З даного чотиризначного натурального числа створити двозначне, що складається з його середніх цифр. (*)

Варіант 12.

1. Сторона основи правильної чотирикутної піраміди d, бічне ребро p. Визначити площу повної поверхні та об'єм піраміди. Результат вивести з точністю до сотих.
2. Знайти квадратний корінь суми цифр даного трицифрового натурального числа.

Варіант 13.

1. Прямі $x=a$ і $y=b$ ділять чотирикутник з вершинами $A(0;0)$, $B(0;y_1)$, $C(x_1;y_1)$, $D(x_1;0)$ на чотири частини. Визначити площі утворених фігур. Результат вивести з точністю до тисячних.
2. Знайти квадрат суми цифр даного чотирицифрового натурального числа.

Варіант 14.

1. Обчислити площу повної поверхні правильної чотирикутної піраміди з стороною основи a і висотою h. Результат вивести з точністю до десятих.

2. Знайти різницю між добутком і сумою цифр даного трицифрового числа.

Примітка: У 2 завданні лабораторної роботи, які помічені (*), результат виводити ЛИШЕ як ОДНЕ число!!!

Розгалуження

Лабораторна робота №4

Мета: Набути уміння та навички складання програм з використанням розгалуження на мові Pascal **.**

***Програмне забезпечення**:* Free Pascal, ABCPascal.

Контрольні питання.

1. Яка програма називається програмою з розгалуженням? Яка структура блок-схеми описує таку програму?
2. Яка форма вказівки розгалуження (умовного переходу)?
3. Які умови в програмах з розгалуженнями можна назвати простими? Які складними? Описати синтаксис складних умов.
4. Який тип можуть мати аргументи та значення стандартних числових функцій?
5. Описати синтаксис оператора вибору.

Завдання по варіантам.

Варіант 1.

1. Визначити четверть площини, в якій знаходиться точка з координатами (x,y).
2. Знайти найменше з двох заданих чисел a та b.

Варіант 2.

1. Визначити, чи існує трикутник зі сторонами a, b, c.
2. Дано ціле число x. Вивести на екран модуль числа x.

Варіант 3.

1. Визначити, чи належить точка з координатами (x,y) кругу радіуса R з цетром у точці з координатами (x_0,y_0).
2. Визначити, чи введене число з клавіатури трицифрове.

Варіант 4.

1. Розв’язати квадратне рівняння виду $ax^2+bx+c=0$ з заданими коефіцієнтами a, b, c.
2. Визначити, чи являється число, введене з клавіатури парним.

Варіант 5.

1. Якщо двозначне число N має однакові цифри, то вивести початкове число в квадраті, якщо ні, то вивести на екран квадрати двозначних чисел, перше з яких має цифри рівні першій цифрі числа N, друге число має цифри рівні другій цифрі числа N.

2. Дано два числа. Якщо хоча б одне з них парне, вивсети на екран "Так" в інакшому випадку вивести "Ні".

Варіант 6.

1. За довжинами сторін трикутника визначити тип трикутника – рівнобедренний, рівносторонній чи різносторонній.
2. Якщо число, введен з клавіатури, чотирицифрове і ділиться на 5, то вивести на екран "ТАК".

Варіант 7.

1. Вивести на екран три задані числа в порядку зростання.
2. Визначити, чи складається двоцифрове число, введене з клавіатури, з однакових цифр.

Варіант 8.

1. За відомими координатами центрами двох кіл та радіусами визначити скільки спільних точок вони мають.
2. Дано трицифрове число. Чи правильно, що введен число складається з різних цифр?

Варіант 9.

1. Утворити з цифр трьохзначного числа найбільше можливе трьохзначне число.
2. Данно чотирицифрове число. Чи правильно, що сума першої та останньої цифри дорівнює різниці другої та третьої цифри?

Варіант 10.

1. За довжинами сторін трикутника визначити вид трикутника: гострокутній, тупокутній та прямокутній.
2. Якщо дане трицифрове число міститбь хоча б одну 1, то пдвоїти дане число, в інакшому випадку пізнести до квадрату.

Варіант 11.

1. Відомо, що астрологи поділяють рік на 12 періодів, і кожному із них ставлять у відповідність один із знаків Зодіаку. Написати програму визначення знаку зодіаку за введеною датою.
2. Знайти найбільше з трьох заданих чисел.

Варіант 12.

1. Записати ціле число N (N<13) римською нумерацією.
2. Знайти середнє з трьох заданих чисел. Середнім буде вважатись число більше найменшого, але не більше найбільшого. Якщо такого немає вивсети відповідне повідомлення на екран.

Варіант 13.

1. Скласти програму, яка по заданому віку користувача повідомляє, до якої вікової категорії він відноситься: до 13 – дитинство, від 14 до 20 - юність, від 20 до 30 – молодість, від 30 до 50 – зрілість, після 60 – старість.
2. Скласти програму визначення вартості покупки з урахуванням знижки. Знижка в 3% надається тоді, коли сума покупки більше 500 грн., і в 5% - якщо сума покупки більша 1000 грн.

Варіант 14.

1. Скласти програму для визначення виду паралелограма (ромб, прямокутник, квадрат, паралелограм) за двома сторонами та кутом між ними.
2. Розв'язати рівняння ax+b=0 за заданими коефіцієнтами a та b.

Цикл з параметром

Лабораторна робота №5

Мета: Набути уміння та навички програмування циклічних процесів з відомою кількістю повторень.

***Програмне забезпечення**:* Free Pascal, ABCPascal.

Контрольні питання.

1. Що таке цикл?
2. Які види циклів є у мові Pascal?
3. Який цикл називається циклом з параметром?
4. Якого типу може бути параметр циклу for?
5. Що таке вкладені цикли? Для чого їх використовують?
6. За допомогою якої команди можна перейти до наступної ітерації (повторення) циклу?
7. Яка команда дає можливість вийти з циклу?

Завдання по варіантам.

Варіант 1.

1. Знайти на відрізку [a,b] натуральне число з найбільшою сумою дільників.

Варіант 2.

1. Побудувати таблицю множення для заданого числа n.

Варіант 3.

1. Досконалим називають натуральне число, що дорівнює сумі всіх його дільників, крім самого себе (наприклад, 6=1+2+3). Знайти на заданому проміжку [a,b] всі досконалі числа.

Варіант 4.

1. Знайти n перших простих чисел.

Варіант 5.

1. Дано натуральні числа q1, q2, … qn. Знайти суму подвоєних непарних чисел.

Варіант 6.

1. Знайти n перших чисел Фібоначчі.

Варіант 7.

1. Задано натуральне число n. Знайти суму його дільників.

Варіант 8.

1. Задано натуральне число n. Обчислити y=1!+2!+3!+…+n!.

Варіант 9.

1. На заданому проміжку знайти всі натуральні числа, що дорівнюють кубу суми своїх цифр

Варіант 10.

1. Знайти кількість “щасливих” автобусних квитків у рулоні (номером квитка є ціле число, яке містить шість цифр: 123123, 001001).

Варіант 11.

1. Задано деяке число N. Скласти програму пошуку простих чисел менших за N.

Варіант 12.

1. Задано деяке число N. Скласти програму для перевірки, чи є задане число простим.

Цикл з передумовою

Лабораторна робота №6

Мета: Набути уміння та навички застосовувати цикл з передумовою.

Програмне забезпечення*:* Free Pascal, ABCPascal.

Контрольні питання.

1. Як записується цикл з передумовою?
2. Чим відрізняється цикл з параметром від циклу з передумовою?

Завдання по варіантам.

Варіант 1.

1. Знайти найбільший спільний дільник двох цілих чисел.

Варіант 2.

1. Скоротити звичайний дріб m/n.

Варіант 3.

1. З’ясувати, чи є натуральне число паліндромом.

Варіант 4.

1. Розкласте задане натуральне число на прості множники.

Варіант 5.

1. Вивести N-тий член Фібоначі. F(1)=1; F(2)=1; F(i)=F(i-1)+F(i-2)

Варіант 6.

1. Досконалим називають натуральне число, що дорівнює сумі всіх його дільників, крім самого себе (наприклад, 6=1+2+3). Знайти на заданому проміжку [a,b] всі досконалі числа.

Варіант 7.

1. Знайти n перших простих чисел.

Варіант 8.

1. Дано натуральні числа q1, q2, … qn. Знайти суму подвоєних непарних чисел.

Варіант 9.

1. Задано натуральне число n. Знайти суму його дільників.

Варіант 10.

1. На заданому проміжку знайти всі натуральні числа, що дорівнюють кубу суми своїх цифр

Варіант 11.

1. Знайти кількість “щасливих” автобусних квитків у рулоні (номером квитка є ціле число, яке містить шість цифр: 123123, 001001).

Варіант 12.

1. Задано деяке число N. Скласти програму пошуку простих чисел менших за N.

Цикл з післяумовою

Лабораторна робота №7

Мета: Набути уміння та навички застосовувати цикл з післяумовою.

***Програмне забезпечення**:* Free Pascal, ABCPascal.

Контрольні питання.

1. Як записується цикл з післяумовою?
2. Чим відрізняється цикл з передумовою і цикл з післяумовою?
3. Чим відрізняється цикл з параметром від циклу з післяумовою?

Завдання по варіантам.

Варіант 1.

1. Побудувати таблицю множення для заданого натурального числа N, де 0<N<10.

Варіант 2.

1. Побудувати таблицю значень функцій Trunc і Round на проміжку чисел [n, m] з кроком h.

Варіант 3.

1. Знайти найбільшу цифру натурального числа.

Варіант 4.

1. Скласти програму переведення будь-якого натурального числа з десяткової системи числення в систему числення за основою a, де a належить проміжку (1;10).

Варіант 5.

1. Знайти всі дільники заданого натурального числа.

Варіант 6.

1. Дано натуральні числа a і b, що мають однакову кількість цифр. Визначити кількість цифр числа a, які входять до запису числа b і знаходяться на тих самих позиціях, що і в записі числа a.

Варіант 7.

1. Досконалим називають натуральне число, що дорівнює сумі всіх його дільників, крім самого себе (наприклад, 6=1+2+3). Знайти на заданому проміжку [a,b] всі досконалі числа.

Варіант 8.

1. Знайти n перших простих чисел.

Варіант 9.

1. Дано натуральні числа q1, q2, … qn. Знайти суму подвоєних непарних чисел.

Варіант 10.

1. Задано натуральне число n. Знайти суму його дільників.

Варіант 11.

1. Задано натуральне число n. Обчислити y=1!+2!+3!+…+n!.

Варіант 12.

1. На заданому проміжку знайти всі натуральні числа, що дорівнюють кубу суми своїх цифр

Одновимірні масиви

Лабораторна робота №8

Мета: Набути уміння та навички застосування одновимірних масивів при розв'яуванні задач з програмування.

***Програмне забезпечення**:* Free Pascal, ABCPascal.

Контрольні питання.

1. Що таке масив?
2. Як створити масив мовою Pascal?
3. Як описати масив в мові Free Pascal?
4. Як звернутись до комірки пам’яті масиву?
5. Якого типу можуть бути індекси в масиві? Чи можуть бути індекси в масиві від’ємні?

Завдання по варіантам.

Варіант 1.

1. Знайти суму елементів парних номерів масиву дійсних чисел, більших за 9.

Варіант 2.

1. Замінити числом 5 елементи, менші від 5, парних номерів масиву натуральних чисел.

Варіант 3.

1. З’ясувати, чи упорядковані елементи масиву за зростанням.

Варіант 4.

1. Переставити місцями найбільший і найменший елементи масиву.

Варіант 5.

1. Здійснити зсув елементів масиву на k позицій вправо за правилом: при зсуві на одну позицію право і-тий елемент стає і+1-м, n-тий елемент – першим.

Варіант 6.

1. Заповнити масив C(n+m) елементами масиву A(n), потім елементами

масиву B(m).

Варіант 7.

1. Знайти елемент масиву, що найчастіше зустрічається.

Варіант 8.

1. Визначити кількість різних елементів масиву.

Варіант 9.

1. Замінити кожний елемент масиву, крім двох останніх, сумою двох наступних.

Варіант 10.

1. Дано масив. Упорядкувати його за спаданням. Вставити задане число у впорядкований масив із збереженням упорядкованості.

Варіант 11.

1. Заповнити масив простими числами по зростанню.

Варіант 12.

1. В даному масиві з n цілих чисел знайти всі непарні числа.

Двовимірні масиви

Лабораторна робота №9

Мета: Набути уміння та навички роботи з двовимірними масивами даних.

***Програмне забезпечення**:* Free Pascal, ABCPascal.

Контрольні питання.

1. Як описати двовимірний масив в мові Free Pascal?
2. Як ввести двовимірний масив?
3. Якого типу можуть бути індекси в масиві? Чи можуть бути індекси в масиві від'ємні.

Завдання по варіантам.

Варіант 1.

1. Знайти суму найбільших елементів рядків масиву цілих чисел.

Варіант 2.

1. У масиві цілих чисел знайти номери рядків, у яких всі елементи: 1) нулі; 2) однакові; 3) парні числа.

Варіант 3.

1. У кожному рядку масиву цілих чисел із від'ємним елементом на головній діагоналі знайти: 1) суму елементів; 2) найбільший елемент.

Варіант 4.

1. Знайти найменший елемент заштрихованої частини масиву:

Варіант 5.

1. Обчислити визначник матриці дійсних чисел.

Варіант 6.

1. Знайти найбільший елемент заштрихованої частини масиву:

Варіант 7.

1. Знайти найменший елемент заштрихованої частини масиву:

Варіант 8.

1. Знайти найбільший елемент заштрихованої частини масиву:

Варіант 9.

1. Знайти найменший елемент заштрихованої частини масиву:

Варіант 10.

1. Знайти найбільший елемент заштрихованої частини масиву:

Варіант 11.

1. Знайти найменший елемент заштрихованої частини масиву:

Варіант 12.

1. Знайти найбільший елемент заштрихованої частини масиву:

Варіант 13.

1. Знайти найменший елемент заштрихованої частини масиву:

Варіант 14.

1. Знайти найбільший елемент заштрихованої частини масиву:

Упорядкування масивів

Лабораторна робота №10

Мета: Набути уміння та навички упорядкування масивів різними способами.

Програмне забезпечення*:* Free Pascal, ABCPascal.

Контрольні питання.

1. Як відбувається сортування масиву бульбашковим способом?
2. Як відбувається сортування масиву методом мінімальних елементів?
3. Як відбувається сортування масиву методом вставки?
4. Як знайти max, min елементи масиву?

Завдання по варіантам.

Варіант 1.

1. Упорядкувати вибором масив цілих чисел за зростанням.

Варіант 2.

1. Упорядкувати методом вставки масив цілих чисел за зростанням.

Варіант 3.

1. Упорядкувати методом обміну масив дійсних чисел за зростанням.

Варіант 4.

1. Упорядкувати вибором масив цілих чисел за спаданням.

Варіант 5.

1. Упорядкувати методом вставки масив цілих чисел за спаданням.

Варіант 6.

1. Упорядкувати методом обміну масив дійсних чисел за спаданням.

Варіант 7.

1. В масиві n дійсних чисел знайти суму квадратів восьми найменших елементів.

Варіант 8.

1. В масиві з n цілих чисел знайти всі числа, які більші за середнє арифметичне. Результат вивести в порядку спадання.

Варіант 9.

1. В масиві з n дійсних чисел знайти десять найменших додатних чисел.

Варіант 10.

1. В масиві з n цілих чисел кількість парних чисел, які розміщені між першим найбільшим і останнім найменшим.

Варіант 11.

1. В масиві з n цілих чисел вивести всі парні числа у відсортованому порядку.

Варіант 12.

1. В даному масиві з n цілих чисел знайти всі непарні числа, і вивести їх на екран у відсортованому порядку.

Варіант 13.

1. Дано масив на 23 дійсних чисел.. Вивести всі елементи, які стоять на непарних місцях у відсортованому порядку.

Варіант 14.

1. В даному масиві з 17 дійсних чисел вивести всі від'ємні числа відсортовані по спаданню.

Літерні величини

Лабораторна робота №11

Мета: Набути уміння та навички складання програм з використанням даних типу String.

***Програмне забезпечення**:* Free Pascal, ABCPascal.

Контрольні питання.

1. Поняття рядкової та символьної величин. Як описуються рядкові та символьні величини?
2. Яка вказівка програми дозволяє виконати операцію склеювання рядкових (символьних) величин? Навести приклад.
3. Яка вказівка програми дозволяє визначити кількість символів рядкової величин? Навести приклад.
4. Яка вказівка програми дозволяє виконати операцію вирізки? Навести приклад.
5. Описати алгоритм, що дозволяє підрахувати скільки разів певний символ зустрічається у даному тексті.
6. Описати алгоритм, що дозволяє підрахувати кількість слів у даному тексті.
7. Описати алгоритм, за яким перевіряється, чи дане слово є словом-"паліндромом" (слово однаково читається зліва направо і справа наліво).

Завдання по варіантам.

Варіант 1.

1. Знайти кількість входжень одного текстового рядка в інший.

Варіант 2.

1. Знайти кількість різних символів текстового рядка.

Варіант 3.

1. Дано текстовий рядок. Утворити рядок, який буде складатись з символів початкового рядка, але розміщених у зворотному порядку.

Варіант 4.

1. Знайти найдовше слово у текстовому рядку.

Варіант 5.

1. Текстовий рядок перетворити у «речення» за такими вимогами: перед першим словом не повинно бути пропусків; після останнього слова повинна бути крапка; речення починається з великої літери; слова відокремлюються одним пропуском; між словом і крапкою чи комою немає пропуску.

Варіант 6.

1. Знайти однакові слова двох текстових рядків.

Варіант 7.

1. У текстовому рядку знайти слова, що повторюються.

Варіант 8.

1. Утворити всі можливі пари символів із заданої множини символів (наприклад: для множини символів «абв» можна утворити такі пари символів: аб, бв, ав).

Варіант 9.

1. Обчислити значення виразу, який подано текстовим рядком abc: a, c – числа, b – знак арифметичної операції (наприклад, 3+2).

Варіант 10.

1. Утворити рядок із перших літер слів текстового рядка.

Варіант 11.

1. З’ясувати, чи утворюють два текстові рядки однакові множини символів.

Варіант 12.

1) У текстовому рядку замінити всі одноцифрові десяткові числа римськими числами.

Варіант 13.

1. Арифметичний вираз подано текстовим рядком, який складається тільки з цифр та знаків арифметичних операцій. З’ясувати, чи коректно задано арифметичний вираз (знак арифметичної дії повинен розмізуватися між двома числами).

Варіант 14.

1. З’ясувати, чи правильно розміщено дужки в арифметичному виразі.

Використання процедур і функцій

Лабораторна робота №12

Мета: Набути уміння та навички використання процедур та функцій.

***Програмне забезпечення**:* Free Pascal, ABCPascal.

Контрольні питання.

1) Як описати процедуру мовою Pascal?
2) Як передати дані у процедуру і отримати їх?
3) Де можна записувати процедури?
4) Як описати функцію мовою Pascal?
5) Як передати дані у функцію і отримати їх?
6) Де можна записувати функції?
7) Як функція повертає значення?

Завдання по варіантам.

Варіант 1.

1. Написати функцію для знаходження НСД двох чисел.

Варіант 2.

1. Написати процедуру для знаходження НСД двох чисел.

Варіант 3.

1. Написати програму з використанням функції для такої задачі: Дано n цілих чисел. Знайти серед них число, у якого сума цифр має найбільше значення.

Варіант 4.

1. Написати програму з використанням процедури для такої задачі: Дано n цілих чисел. Знайти серед них число, у якого сума цифр має найбільше значення.

Варіант 5.

1. Написати функцію для знаходження НСК двох чисел.

Варіант 6.

1. Написати процедуру для знаходження НСК двох чисел.

Варіант 7.

1. Написати програму з використанням функції для такої задачі: Дано n цілих чисел. Знайти серед них число, у якого перша цифра має максимальне значення.

Варіант 8.

1. Написати програму з використанням процедури для такої задачі: Дано n цілих чисел. Знайти серед них число, у якого перша цифра має максимальне значення.

Варіант 9.

1. Написати програму з використанням функції для такої задачі: Дано n цілих чисел. Знайти серед них число з найбільшою кількістю дільників.

Варіант 10.

1. Написати програму з використанням процедури для такої задачі: Дано n цілих чисел. Знайти серед них число з найбільшою кількістю дільників.

Варіант 11.

1. Написати програму з використанням функції для такої задачі: Дано n цілих чисел. Знайти серед них число, у якого сума всіх дільників має масмальне значення.

Варіант 12.

1. Написати програму з використанням процедури для такої задачі: Дано n цілих чисел. Знайти серед них число, у якого сума всіх дільників має масмальне значення.

Варіант 13.

1. Написати програму з використанням функції для такої задачі: Дано n цілих чисел. Підрахувати кількість чисел, в записі яких немає цифри 8.

Варіант 14.

1. Написати програму з використанням процедури для такої задачі: Дано n цілих чисел. Підрахувати кількість чисел, в записі яких немає цифри 8.

Робота з файлами

Лабораторна робота №13

Мета: Набути уміння та навички роботи з текстовими файлами.

***Програмне забезпечення**:* Free Pascal, ABCPascal.

Контрольні питання.

1) Як встановлюється зв'язок між файлом на диску і файловою змінною?
2) Як відкрити файл для читання, для запису, для читання і запису одночасно?
3) Як закрити відкритий файл?
4) За допомогою якої функції можна виявити кінець стрічки, кінець файла?
5) Як зчитати з файла і записати у файл текстовий файл?

Завдання по варіантам.

Варіант 1.

1) Побудувати таблицю значень функції $y = \sqrt{\sin^2 x + \cos x}$ на відрізку [a;b] з кроком h, знайти найменше та найбільше значення функції на заданому проміжку. Усі дані вводяться з файлу input.txt, і виводяться у файл output.txt.

Варіант 2.

1. Побудувати таблицю значень функції $y = \sin^3 x$ на відрізку [a;b] з кроком h, знайти найменше та найбільше значення функції на заданому

проміжку. Усі дані вводяться з файлу input.txt, і виводяться у файл output.txt.

Варіант 3.

1. Побудувати таблицю значень функції $y = \ln\sqrt{2 + 0.55x}$ на відрізку [a;b] з кроком h, знайти найменше та найбільше значення функції на заданому проміжку. Усі дані вводяться з файлу input.txt, і виводяться у файл output.txt.

Варіант 4.

1. Побудувати таблицю значень функції $y = e^x - \sqrt{1 + x}$ на відрізку [a;b] з кроком h, знайти найменше та найбільше значення функції на заданому проміжку. Усі дані вводяться з файлу input.txt, і виводяться у файл output.txt.

Варіант 5.

1. Побудувати таблицю значень функції $y = x^2 + \ln\left(x^2 - 4\right)$ на відрізку [a;b] з кроком h, знайти найменше та найбільше значення функції на заданому проміжку. Усі дані вводяться з файлу input.txt, і виводяться у файл output.txt.

Варіант 6.

1. Побудувати таблицю значень функції $y = \sqrt{x} + \sin^2 x$ на відрізку [a;b] з кроком h, знайти найменше та найбільше значення функції на заданому проміжку. Усі дані вводяться з файлу input.txt, і виводяться у файл output.txt.

Варіант 7.

1. Побудувати таблицю значень функції $y = \lg\left|x^2 - 2x + 3\right|$ на відрізку [a;b] з кроком h, знайти найменше та найбільше значення функції на заданому проміжку. Усі дані вводяться з файлу input.txt, і виводяться у файл output.txt.

Варіант 8.

1. Побудувати таблицю значень функції $y = \dfrac{1}{\ln\left(x^2 - 2.4\right)}$ на відрізку [a;b] з кроком h, знайти найменше та найбільше значення функції на заданому проміжку. Усі дані вводяться з файлу input.txt, і виводяться у файл output.txt.

Варіант 9.

1. Побудувати таблицю значень функції $y = \dfrac{x}{\sqrt{1 + \dfrac{x^2}{9}}}$ на відрізку [a;b] з кроком h, знайти найменше та найбільше значення функції на заданому проміжку. Усі дані вводяться з файлу input.txt, і виводяться у файл output.txt.

Варіант 10.

1. Побудувати таблицю значень функції $y = x^3 - 2x + 1$ на відрізку [a;b] з кроком h, знайти найменше та найбільше значення функції на заданому

проміжку. Усі дані вводяться з файлу input.txt, і виводяться у файл output.txt.

Варіант 11.

1. Побудувати таблицю значень функції $y=(x+2)^3$ на відрізку [a;b] з кроком h, знайти найменше та найбільше значення функції на заданому проміжку. Усі дані вводяться з файлу input.txt, і виводяться у файл output.txt.

Варіант 12.

1. Побудувати таблицю значень функції $y=\frac{2x+3}{3x-7}$ на відрізку [a;b] з кроком h, знайти найменше та найбільше значення функції на заданому проміжку. Усі дані вводяться з файлу input.txt, і виводяться у файл output.txt.

Варіант 13.

1) Побудувати таблицю значень функції $y=1+\sqrt{x}$ на відрізку [a;b] з кроком h, знайти найменше та найбільше значення функції на заданому проміжку. Усі дані вводяться з файлу input.txt, і виводяться у файл output.txt.

Варіант 14.

1) Побудувати таблицю значень функції $y=\sqrt[3]{\sin^2 x}$ на відрізку [a;b] з кроком h, знайти найменше та найбільше значення функції на заданому проміжку. Усі дані вводяться з файлу input.txt, і виводяться у файл output.txt.

Множини. Записи

Лабораторна робота №14

Мета: навчитися працювати з множинами даних.

***Програмне забезпечення**:* Free Pascal, ABCPascal.

Контрольні питання.

1) Що таке множина?
2) Які дії над множинами ви знаєте?
3) Опишіть основні дії над множинами з прикладами.

Завдання по варіантам.

Варіант 1.

1) Знайти кількість латинських літер і кількість цифр у текстовому рядку.

Варіант 2.

1. Знайти кількість різних символів у текстовому рядку.

Варіант 3.

1. Сформувати множину з N (N<21) натуральних чисел, випадковим

чином вибраних із діапазону від 1 до 200. Вивести елементи множини на екран.

Варіант 4.

1. Сформувати множину з N (N<21) натуральних чисел, менших від 200, введенних з клавіатури. Вивести елементи множини та їх середнє арифметичне значення на екран.

Варіант 5.

1. Вивести на екран всі різні символи текстового рядка у порядку їх слідування в рядку.

Варіант 6.

1. Вивести на екран всі різні символи текстового рядка у алфавітному порядку.

Варіант 7.

1. Знайти однакові символи двох текстових рядків.

Варіант 8.

1. Знайти в текстовому рядку слово з найбільшою кількістю різних символів.

Варіант 9.

1. Сформувати множини символів текстових рядків A, B і вивести елементи знайдених множин на екран. Знайти символи рядка A, які не входять у B, і символи рядка B, які не входять у рядок A.

Варіант 10.

1. З діапазону цілих чисел від 0 до 100 випадковим чином вибирають по 10 чисел: до множини A – числа кратні 2, до множини B – числа кратні 3. Отримати перетин множин, об'єднання множин. Вивести елементи отриманих множин на екран.

Варіант 11.

1. З діапазону цілих чисел від 0 до 100 випадковим чином вибирають по 10 чисел: до множини A – числа кратні 5, до множини B – числа кратні 2. Отримати множини: різниці A\B та B\A. Вивести елементи отриманих множин на екран.

Варіант 12.

1. Сформувати множину з N (N<30) натуральних чисел, менших від 100, введенних з клавіатури. Вивести ті елементи множини на екран, які кратні 3.

Варіант 13.

1. Сформувати множину з N (N<25) натуральних чисел, менших від 800, введенних з клавіатури. Вивести ті елементи множини на екран, які кратні 3 і мають у своєму записі цифру 5.

Варіант 14.

1. Сформувати множину з N (N<12) натуральних чисел, менших від 300, введенних з клавіатури. Вивести ті елементи множини на екран, які кратні 5 і мають у своєму записі цифру 2.

Самостійна робота

Приклад виконання задачі з самостійної роботи на порталі e-olimp

1. Проста задача.

Програма зчитує двоцифрове число і виводить через пробіл кожну цифру окремо.

Технічні умови

Вхідні дані

Натуральне число на проміжку від 10 до 99 включно.

Вихідні дані

Спочатку першу цифру числа і через пропуск другу.

Приклад

Приклад вхідних даних	Приклад вихідних даних
23	2 3

Посилання на задачу: http://www.e-olimp.com/ua/problems/1

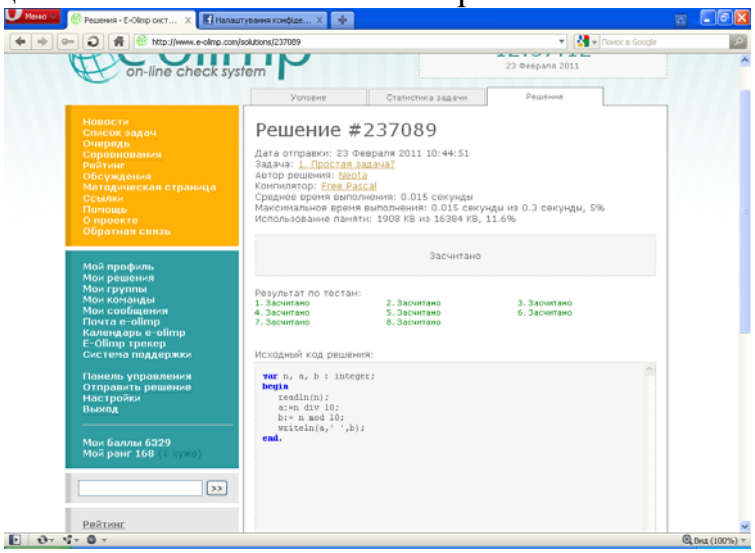
Розв'язок задачі:

```
var n, a, b : integer;
begin
  readln(n);                {зчитуємо число}
  a:=n div 10;              {відділяємо першу цифру}
  b:= n mod 10;             {відділяємо другу цифру}
  writeln(a,' ',b);         {виводимо їх на екран через пропуск}
end.
```

*{**УВАГА!** Останнє виведення має бути з переходом на новий рядок}*

У випадку, якщо задача розв'язана повністю вірно, екран буде мати вигляд:

Також буде показано скільки тестів зараховано.



Лінійні програми

1. Проста задача.

Програма зчитує двоцифрове число і виводить через пробіл кожну цифру окремо.

Технічні умови

Вхідні дані

Натуральне число на проміжку від 10 до 99 включно.

Вихідні дані

Спочатку першу цифру числа і через пропуск другу.

Приклад

Приклад вхідних даних	Приклад вихідних даних
23	2 3

Посилання на задачу: http://www.e-olimp.com/ua/problems/1

2. Метелик-санітар

Учні, йдучи з дому до школи або навпаки – зі школи додому, полюбляють їсти цукерки. Але, як завжди, ця приємна справа інколи має неприємні наслідки – дітки іноді викидають обгортки на шкільному подвір'ї.

Мурзик завжди слідкував за чистотою шкільного двору і йому у цьому з радістю допомагали метелики, вдячні за чудові фотографії, зроблені ним. Метелики могли використовувати власні крильця як лінзи, причому вони могли змінювати їх фокусну відстань. Помітивши обгортку від цукерки, що лежала на шкільному подвір'ї у точці з координатами X1, Y1, метелик перелітав у точку з координатами X2, Y2, Z2, розташовану на шляху сонячного проміння до обгортки і, змінюючи фокусну відстань своїх крилець-лінз, спалював обгортку від цукерки.

Яку оптичну силу D мали крильця-лінзи метелика у цей момент?

Технічні умови

Вхідні дані

У першому рядку 2 числа: координати X1, Y1 обгортки від цукерки. У другому – 3 числа: координати X2, Y2, Z2 метелика у момент спалювання обгортки.

Всі вхідні дані цілі числа, що не перевищують за модулем 1000.

Вихідні дані

Єдине число – оптична сила крилець-лінз D, обчислена з точністю до 3-х знаків після коми за правилами математичних округлень.

Приклад

Приклад вхідних даних	Приклад вихідних даних
10 20 10 20 100	0.010

Посилання на задачу: http://www.e-olimp.com/ua/problems/57

3. Анфіса і квіти

Мурзик одну з квіткових клумб зробив у вигляді шахової дошки розмірами M на N, у кожній клітинці якої росте якась квітка. Інколи на цю клумбу він виводить на прогулянку Анфісу (так, не дивуйтесь, вони дійсно друзі). Анфіса, починаючи завжди з верхнього лівого кута переміщується по клумбі до правого нижнього і збирає квіти, причому таким чином, щоб щоразу проходити по новому маршруту, а Мурзик на виході вручає їй шматочок сиру.

Порахувати, яка найбільша кількість шматочків сиру дістанеться Анфісі, якщо вона весь час намагатиметься зберегти якнайбільше квітів.

Технічні умови

Вхідні дані

У єдиному рядку через пропуск 2 числа: M та N
(0< M,N <= 2000000000)

Вихідні дані

Єдине число - найбільша кількість шматочків сиру, які може отримати Анфіса.

Приклад

Приклад вхідних даних	Приклад вихідних даних
2 3	3

Посилання на задачу: http://www.e-olimp.com/ua/problems/63

4. Сир для Анфіси - 2

Розрізаючи сир в задачі «Сир для Анфіси» у господаря залишалися куски сиру у вигляді прямокутного паралелепіпеда з різними цілими довжинами сторін. Готуючи нову страву із сиру для Анфіси господарю прийшлося розрізати дані куски сиру на кубики зі стороною 1. Яку найменшу кількість розрізів приходилось йому робити, щоб розрізати дані куски сиру, якщо він кожного разу розрізав один кусок сиру на дві частини.

Технічні умови

Вхідні дані

У єдиному рядку записано три числа A, B, C - довжини ребер куска сиру.
1 <= A, B, C <= 2000000000

Вихідні дані

Єдине число найменша кількість розрізів.

Приклад

Приклад вхідних даних	Приклад вихідних даних
2 3 4	23

Посилання на задачу: http://www.e-olimp.com/ua/problems/67

5. Квадрат і точки

Яку найбільшу кількість точок з цілочисельними координатами можна на аркуші в клітинку накрити квадратом зі стороною N клітинок?

Технічні умови

Вхідні дані:

Єдине число - сторона квадрату N (1 <= N <= 10000).
Вихідні дані:
Максимальна кількість накритих клітин K.
Приклад

Приклад вхідних даних	Приклад вихідних даних
1	4

Посилання на задачу: http://www.e-olimp.com/ua/problems/133

6. Білі кубики

Професор Самодєлкін задумав виготовити кубики з брусків білого кольору. Довжина кожного ребра дорівнює 1 дм. Після виготовлення кубиків професор вирішив зробити всі кубики також білого кольору. Скільки кубиків із стороною 1 дм зможе виготовити з одного бруска професор, та скільки сторін прийдеться йому пофарбувати, якщо відомо, що довжини сторін брусків - цілі числа і задані також в дециметрах.
Технічні умови
Вхідні дані
Один рядок містить три цілих числа – розміри бруска в дм, які не перевищують 1000000.
Вихідні дані
В єдиному рядку записати через пропуск два цілих числа: кількість отриманих кубиків та кількість граней кубиків, які необхідно пофарбувати.
Приклад

Приклад вхідних даних	Приклад вихідних даних
1 2 3	6 14

Посилання на задачу: http://www.e-olimp.com/ua/problems/478

7. Велика точність

Дано раціональний дріб m/n. Запишіть його у вигляді десяткового дробу з точністю до k знаків після крапки.
Технічні умови
Вхідні дані
В одному рядку записано 3 числа m, n, k. $0 < m, n \leq 100$, $0 \leq k \leq 1000$.
Вихідні дані
Вивести k точних значущих цифр після десяткової крапки шуканого числа.
Приклад

Приклад вхідних даних	Приклад вихідних даних
1 2 3	0.500

Посилання на задачу: http://www.e-olimp.com/ua/problems/11

8. Добуток цифр

Задано трицифрове число. Визначити добуток його цифр.
Технічні умови
Вхідні дані
У єдиному рядку задане трицифрове число.

Вихідні дані
У єдиному рядку добуток цифр заданого числа.
Приклад

Приклад вхідних даних	Приклад вихідних даних
235	30

Посилання на задачу: http://www.e-olimp.com/ua/problems/906

9. Кільце

Задано площу кільця й радіус зовнішнього кола. Визначити радіус внутрішнього кола.
Технічні умови
Вхідні дані
У єдиному рядку задано 2 дісних числа, спочатку площу кільця і через пропуск - радіус зовнішнього кола. Радіус кола не перевищує 100.
Вихідні дані
У єдиному рядку вивести радіус внутрішнього кола з точністю 2 знаки після десяткової крапки.
Приклад

Приклад вхідних даних	Приклад вихідних даних
50.2655 5	3.00

Посилання на задачу: http://www.e-olimp.com/ua/problems/924

Розгалуження

1. Сірникова модель

Професор Самодєлкін вирішив змайструвати об'ємну модель кубиків з сірників використовуючи сірники для ребер кубиків. Довжина ребра кожного кубика дорівнює одному сірнику. Для побудови моделі трьох кубів в нього пішло 28 сірників.

Яку найменшу кількість сірників потрібно Самодєлкіну для побудови моделі з N кубиків.

Всі числа задачі не перевищують 2·109.
Технічні умови
Вхідні дані
Одне число N - кількість кубиків.
Вихідні дані
Одне число - кількість сірників.
Приклад

Приклад вхідних даних	Приклад вихідних даних
3	28

Посилання на задачу: http://www.e-olimp.com/ua/problems/3

2. Два кола

Визначити в скількох точках перетинаються два кола.
Технічні умови
Вхідні дані:

6 чисел x1, y1, r1, x2, y2, r2, де x1, y1, x2, y2, - координати центрів кіл, r1, r2 – їх радіуси.
Всі числа - дійсні, не перевищують 1000000000 за модулем, та задані не більш ніж з 3 знаками після коми.
Вихідні дані:
одне число, яке показує кількість точок перетину.
0, 1, 2 – відповідна кількість точок перетину
-1 – безліч точок перетину.
Приклад

Приклад вхідних даних	Приклад вихідних даних
0 0 5 5 0 5	2

Посилання на задачу: http://www.e-olimp.com/ua/problems/4

3. Сірники

Яка мінімальна кількість сірників потрібна для того, щоб викласти на площині N квадратів зі стороною в один сірник? Сірники не можна ламати та класти один на одний. Вершинами квадратів повинні бути точки, де сходяться кінці сірників, а сторонами – самі сірники.
Завдання

Напишіть програму, що за кількістю квадратів N, які необхідно скласти, знаходить мінімальну необхідну для цього кількість сірників.
Технічні умови
Вхідні дані
Єдиний рядок вхідного файлу містить одне ціле число N ($1 \le N \le 109$).
Вихідні дані
Єдиний рядок вихідного файлу має містити одне ціле число – мінімальну кількість сірників потрібних для складання заданої кількості квадратів.
Приклад

Приклад вхідних даних	Приклад вихідних даних
4	12

Посилання на задачу: http://www.e-olimp.com/ua/problems/8

4. Кількість учасників олімпіади

Як відомо, на питання про те, скільки в нього учнів, давньогрецький вчений Піфагор відповідав так: "Половина моїх учнів вивчає математику, четверта частина вивчає природу, сьома частина проводить час у мовчазних роздумах, частину, що залишилась, складають 3 діви".

Секретар олімпіади на питання: "Скільки зареєстровано учасників олімпіади з інформатики?", відповідав подібно Піфагору: "K-та частина почала розв'язувати першу задачу, M-та частина – другу, а N-та – третю. В той же час D учасників вирішують проблему: "З чого почати?". Ваша задача вивести кількість учасників олімпіади S, або -1, якщо секретар помилився у своєму повідомленні.
Технічні умови
Вхідні дані

У єдиному рядку через пропуск числа K, N, M, D. 1 <= K, N, M, D <= 1000.
Вихідні дані
Вивести кількість учасників олімпіади S, або -1, якщо секретар помилився у своєму повідомленні.
Приклад

Приклад вхідних даних	Приклад вихідних даних
2 4 7 3	28

Посилання на задачу: http://www.e-olimp.com/ua/problems/43

5. Більярд

Більярд представляє собою прямокутник розмірами M x N, де M і N – натуральні числа. З верхньої лівої лузи вилітає куля під кутом 450 до сусідніх сторін. Лузи розміщено тільки в кутах більярда. Визначити кількість зіткнень кулі з бортами більярда, після яких вона знову попаде в одну з луз, та номер лузи в яку попаде куля. Вважати, що тертя відсутнє, зіткнення абсолютно пружні, а кулю - матеріальною точкою.
Технічні умови
Вхідні дані
У вхідному рядку міститься два числа M та N, 1 < = M, N <= 2000000000. Нумерація луз за годинниковою стрілкою, починаючи з лівої верхньої лузи, з якої вилетіла куля, згідно малюнка. M - горизонтальна сторона більярда, N - вертикальна сторона більярда.
Вихідні дані
Два числа: кількість відбивань кулі, та номер лузи, в яку впаде куля.
Приклад

Приклад вхідних даних	Приклад вихідних даних
2 1	1 2

Посилання на задачу: http://www.e-olimp.com/ua/problems/58

6. Нова шафа

Задано розміри прямокутних дверей A,B та розміри шафи, що має форму прямокутного паралелепіпеда X,Y,Z. Чи можна пронести шафу у двері, якщо проносити її дозволяється так, щоб кожне ребро шафи було паралельне або перпендикулярне кожній стороні дверей.
Технічні умови
Вхідні дані
У вхідному файлі записано дійсні числа A, B, X, Y, Z (0< A, B, X, Y, Z <10).
Вихідні дані
У вихідний файл потрібно записати 1, якщо шафу можна вільно пронести у двері і 0 у протилежному випадку.
Приклад

Приклад вхідних даних	Приклад вихідних даних
5 7 4 6 8	1

Посилання на задачу: http://www.e-olimp.com/ua/problems/76

7. Компакт-диски

Чисті компакт-диски продаються в трьох різних видах упаковок. Упаковка зі 100 дисків коштує 100 грн., з 20 дисків - 30 грн., а один окремий диск коштує 2 грн. Якої мінімальної суми має вистачити для покупки N таких дисків?

Технічні умови

Вхідні дані

Кількість N дисків, які потрібно купити. N <= 1000.

Вихідні дані.

Мінімальна сума, потрібна для покупки.

Приклад

Приклад вхідних даних	Приклад вихідних даних
123	136

Посилання на задачу: http://www.e-olimp.com/ua/problems/107

8. Середнє з чисел

Дано три різних числа a, b, c. Вивести середнє з них.

Технічні умови

Вхідні дані

Числа a, b, c цілі і по модулю не перевищують 1000

Вихідні дані.

Одне число, яке являється середнім з даних трьох чисел.

Приклад

Приклад вхідних даних	Приклад вихідних даних
11 3 7	7

Посилання на задачу: http://www.e-olimp.com/ua/problems/108

9. Торт

На день народження наступника Тутти королівський повар приготував великий святковий торт, який було подано на стіл Трьом Товстунам. Перший товстун сам міг би з'їсти його повністю за T1 годин, другий - за T2 годин, а третій - за T3 годин.

Напишіть програму, яка визначить скільки часу потрібно товстунам, щоб з'їсти святковий торт разом.

Технічні умови

Вхідні дані

В єдиному рядку вхідного файлу задано три невід'ємні цілі числа T1, T2 и T3, кожне з яких не перевищує 10000.

Вихідні дані.

В єдиному рядку вихідного файлу необхідно вивести час в годинах, за який товстуни можуть з'їсти торт. Результат округлити до 2 знаків після коми.

Приклад

Приклад вхідних даних	Приклад вихідних даних
3 3 3	1.00

Посилання на задачу: http://www.e-olimp.com/ua/problems/112

10. Прямокутник

Дано координати трьох точок, вершин прямокутника. Знайдіть координати четвертої точки.

Технічні умови

Вхідні дані:

В єдиному рядку записано шість чисел координати трьох точок.

Вихідні дані:

Два числа, координати шуканої вершини прямокутника.

Всі вхідні та вихідні дані - цілі числа по модулю не перевищують 100.

Приклад

Приклад вхідних даних	Приклад вихідних даних
0 0 0 1 2 1	2 0

Посилання на задачу: http://www.e-olimp.com/ua/problems/130

11. Банкомат

Банкомат має в достатній кількості купюри 10, 20, 50, 100, 200 і 500 грн. Знайти найменшу кількість купюр, яку необхідно використати, щоб видати суму N грн. або вивести -1, якщо вказану суму видати не можна.

Технічні умови

Вхідні дані:

У вхідному файлі одне число N. 1<=N<=1000000.

Вихідні дані:

До вихідного файлу потрібно записати відповідь – одне число.

Приклад

Приклад вхідних даних	Приклад вихідних даних
770	4

Посилання на задачу: http://www.e-olimp.com/ua/problems/138

12. Точка і трикутник

Чи належить точка O трикутнику ABC ?

Технічні умови

Вхідні дані:

До вхідного файлу записані координати точок O, A, B, C.

Числові значення по модулю не перевищують 100.

Вихідні дані:

У вихідний файл потрібно записати 1, якщо точка O належить трикутнику ABC і 0 у протилежному випадку.

Приклад

Приклад вхідних даних	Приклад вихідних даних
1 1 -1 0 2 4 4 1	1

Посилання на задачу: http://www.e-olimp.com/ua/problems/143

13. Відрізок і кола

На площині задано систему концентричних кіл, центри яких розміщені у початку координат, а радіуси дорівнюють 1, 2, 3, ... Також на площині

задано відрізок, кінці якого знаходяться у точках (X1, Y1) та (X2, Y2). Потрібно знайти кількість спільних точок цього відрізку і вказаної системи кіл.

Технічні умови

Вхідні дані:

У першому рядку вхідного файлу міститься 4 цілих числа X1, Y1, X2, Y2. Числа не перевищують за модулем 103. Відрізок має ненульову довжину.

Вихідні дані:

У вихідний файл виведіть відповідь на задачу.

Приклад

Приклад вхідних даних	Приклад вихідних даних
1 1 2 1	1

Посилання на задачу: http://www.e-olimp.com/ua/problems/197

14. Ремонт

Ваш любимий дядя – директор фірми, яка робить євроремонти в офісах. У зв'язку з фінансово-економічною кризою, дядечко вирішив оптимізувати своє підприємство. Давно ходять слухи, що бригадир у дядечкові фірмі купує зайву кількість будматеріалов, а залишки використовує для потреб своєї нової дачі. Ваш дядя зацікавився, скільки в дійсності банок фарби необхідно для покраски стін в офісі довжиною L метрів, шириною – W і висотою – H, якщо однієї банки хватає на 16 м2, а розмірами дверей та вікон можна знехтувати? Замовлень багато, тому дядя попросив написати програму, яка буде все це рахувати.

Технічні умови

Вхідні дані

У першому рядку міститься кількість замовлень. Опис кожного замовлення складається з трьох натуральних чисел L, W, H – довжини, ширини і висоти офісу в метрах відповідно, кожне з яких не перевищує 1000.

Вихідні дані

Для кожного замовлення виводиться в окремомоу рядку одне число – кількість банок фарби, необхідних для покраски офісу.

Приклад

Приклад вхідних даних	Приклад вихідних даних
2 8 8 2 1 1 3	4 1

Посилання на задачу: http://www.e-olimp.com/ua/problems/500

15. Олімпіада

Олімпіада почалася в h1 год m1 хв s1 сек, а закінчилася цієї ж календарної доби в h2 год m2 хв s2 сек. Скільки часу (год хв сек) тривала олімпіада?

Технічні умови

Вхідні дані

У першому рядку записано час початку, а у другому - час закінчення олімпіади у форматі год хв сек.
$0 \le h1 \le h2 \le 23$, $0 \le m1, m2 \le 59$, $0 \le s1, s2 \le 59$.
Вихідні дані
У єдиний рядок вихідного файлу потрібно записати час, який тривала олімпіада у форматі год хв сек.
Приклад

Приклад вхідних даних	Приклад вихідних даних
9 30 0 12 45 30	3 15 30

Посилання на задачу: http://www.e-olimp.com/ua/problems/125

16. Номер квартири

Багатоквартирний будинок з N квартир має P під'їздів і Q поверхів, причому на кожному поверсі кожного під'їзду розміщено однакову кількість квартир. Визначити в якому під'їзді та на якому поверсі знаходиться квартира з заданим номером K.
Технічні умови
Вхідні дані
В єдиному рядку файлу записано значення N, P, Q, K. $1 \le K \le N \le 1000$, $P * Q \le N$.
Вихідні дані
В єдиний рядок вихідного файлу треба вивести номер під'їзду і поверх, на якому знаходиться квартира з номером K.
Приклад

Приклад вхідних даних	Приклад вихідних даних
30 2 5 27	2 4

Посилання на задачу: http://www.e-olimp.com/ua/problems/126

17. Рівень навчальних досягнень

Встановити рівень навчальних досягнень учня (початковий, середній, достатній, високий) відповідно до заданої оцінки (від 1 до 12).
Технічні умови
Вхідні дані
Одне число - бал учня
Вихідні дані
Вивести Initial для початкового рівня, Average - для середнього, Sufficient - для достатнього і High - для високого.
Приклад

Приклад вхідних даних	Приклад вихідних даних
12	High

Посилання на задачу: http://www.e-olimp.com/ua/problems/902

18. Перша чи остання?

Задано трицифрове число. Визначити, яка цифра в ньому є більшою – перша чи остання.

Технічні умови

Вхідні дані

У єдиному рядку задано трицифрове число.

Вихідні дані

Вивести більшу з вказаних цифр. У випадку їх рівності вивести знак "=" (без лапок).

Приклад

Приклад вхідних даних	Приклад вихідних даних
328	8

Посилання на задачу: http://www.e-olimp.com/ua/problems/903

19. Який трикутник?

Визначити тип трикутника (рівносторонній, рівнобедрений, різносторонній) за заданими довжинами його сторін.

Технічні умови

Вхідні дані

В єдиному рядку задано 3 цілих числа - довжини сторін трикутника. Довжини сторін не перевищують 100.

Вихідні дані

В єдиному рядку вивести 1, якщо трикутник рівносторонній, 2 - якщо рівнобедрений і 3 - якщо різносторонній.

Приклад

Приклад вхідних даних	Приклад вихідних даних
3 4 3	2

Посилання на задачу: http://www.e-olimp.com/ua/problems/905

20. Квадратне рівняння

Скласти програму для розв’язання квадратного рівняння $ax2 + bx + c = 0$ ($a \neq 0$).

Технічні умови

Вхідні дані

У єдиному рядку задано через пропуск 3 цілі числа - коефіцієнти квадратного рівняння відповідно a, b та c. Значення коефіцієнтів не перевищують по модулю 100.

Вихідні дані

У єдиному рядку вивести у випадку відсутності коренів повідомлення "No roots" (без лапок), у випадку, якщо розв'язок містить один корінь вивести спочатку повідомлення "One root:" (без лапок), а далі через пропуск сам корінь, у випадку наявності двох коренів вивести спочатку повідомлення "Two roots:" (без лапок), а далі через пропуск спочатку менший, а потім більший корінь. Гарантується, що у випадку наявності розв'язків всі корені цілочисельні.

Приклад

Приклад вхідних даних	Приклад вихідних даних
1 -5 6	Two roots: 2 3

Посилання на задачу: http://www.e-olimp.com/ua/problems/911

21. Прямокутний чи ні?

Задано довжини сторін трикутника. Визначити, чи є цей трикутник прямокутним.

Технічні умови

Вхідні дані

У єдиному рядку задано 3 натуральні числа - довжини сторін трикутника. Довжини сторін не перевищують 1000.

Вихідні дані

Вивести "YES" (без лапок), якщо трикутник є прямокутним, або "NO" (без лапок) у протилежному випадку.

Приклад

Приклад вхідних даних	Приклад вихідних даних
3 5 4	YES

Посилання на задачу: http://www.e-olimp.com/ua/problems/915

22. Яка чверть?

Задано точку з координатами x та y. Визначити, в якій координатній чверті вона розміщена.

Технічні умови

Вхідні дані

У єдиному рядку через пропуск задано 2 дійсні числа - координати точки, значення координат по модулю не перевищують 100.

Вихідні дані

Єдине число - номер відповідної чверті, або 0, якщо однозначно визначити чверть неможливо.

Приклад

Приклад вхідних даних	Приклад вихідних даних
12 31	1

Посилання на задачу: http://www.e-olimp.com/ua/problems/918

23. Пора року

Визначити назву пори року за заданим номером місяця, використовуючи складені умови.

Технічні умови

Вхідні дані

Єдине число - номер місяця.

Вихідні дані

Для весняних місяців вивести Spring, для літніх - Summer, для осінніх - Autumn і для зимових - Winter.

Приклад

Приклад вхідних даних	Приклад вихідних даних
5	Spring

Посилання на задачу: http://www.e-olimp.com/ua/problems/923

24. Паралелограм

Задано 4 числа a, b, c, d, що визначають довжини відрізків. Визначити, чи можна з цих відрізків утворити паралелограм.

Технічні умови

Вхідні дані

У єдиному рядку задано 4 числа через пропуск.

Вихідні дані

Вивести у єдиному рядку слово "YES", якщо паралелограм утворти можна або "NO" (без лапок) у протилежному випадку.

Приклад

Приклад вхідних даних	Приклад вихідних даних
Sample 1	Sample 1
2 4 2 4	YES
Sample 2	Sample 2
2 4 2.5 4	NO

Посилання на задачу: http://www.e-olimp.com/ua/problems/929

Цикли: з параметром, передумовою та післяумовою

1. Цифри

Підрахувати кількість цифр цілого невід'ємного числа N (0 <= N <= 2000000000).

Технічні умови

Вхідні дані

Число N.

Вихідні дані

Кількість цифр у ньому.

Приклад

Приклад вхідних даних	Приклад вихідних даних
13243	5

Посилання на задачу: http://www.e-olimp.com/ua/problems/2

2. Садівник

Садівник посадив за день N дерев і повинен був вилити під кожен саджанець по відру води. Так як в день посадки йшов дощ, садівник почав поливку дерев не в день посадки, а починаючи з якось K-го дня. Скільки днів садівник не поливав дерева, якщо в останній день він під кожне з дерев вилив 1/N частину води з відра, у передостанній - 1/(N-1) частину, і т.д., а загалом під кожне з дерев вилив не більше, ніж по половині відра води?

Технічні умови

Вхідні дані

0 < N <= 1000000
Вихідні дані
Одне число – кількість днів.
Приклад

Приклад вхідних даних	Приклад вихідних даних
3	2

Посилання на задачу: http://www.e-olimp.com/ua/problems/10

3. Дракон

У кожної S-ніжки 1 голова. Знайти кількість ніг N у K-голового дракона, якщо разом у всіх A голів і B ніг.
Технічні умови
Вхідні дані
4 числа: S, K, A, B. Всі числа не перевищують 1000.
Вихідні дані
Кількість ніг у дракона. Якщо вхідні дані суперечні, вивести у вихідний файл -1, у випадку наявності декількох розв'язків – вивести довільний з них.
Приклад

Приклад вхідних даних	Приклад вихідних даних
4 7 35 36	2

Посилання на задачу: http://www.e-olimp.com/ua/problems/16

4. Рівень паліндромності

Задано натуральне число M. Якщо це не паліндром, то записуємо його у зворотному порядку та додаємо до заданого. Кроки повторюються, доки не буде отримано число-паліндром. Кількість виконаних операцій назвемо рівнем паліндромності заданого числа.

Знайти рівень паліндромності числа M.
Технічні умови
Вхідні дані
Єдине число M (0 < M < 10000).
Вихідні дані
Єдине число - рівень паліндромності.
Приклад

Приклад вхідних даних	Приклад вихідних даних
865	2

Посилання на задачу: http://www.e-olimp.com/ua/problems/29

5. Змій Горинич

В деякому царстві жив Змій Горинич. У нього було N голів та M хвостів. Іван-царевич вирішив знищити губителя людських душ, для чого йому його кума Баба Яга подарувала чарівний меч, оскільки тільки ним можна вбити Змія Горинича. Якщо відрубати одну голову, то на її місці виростає нова, якщо відрубати хвіст, то замість нього виросте 2 хвости. Якщо відрубати два хвости, то виросте 1 голова, і тільки коли зрубати 2 голови, то

не виростає нічого. Змій Горинич гине тільки в тому випадку, коли йому відрубати всі голови і всі хвости. Визначити мінімальну кількість ударів мечем, потрібну для знищення Змія Горинича.

Технічні умови

Вхідні дані

Два числа N, M ($0 \le N, M \le 1000$).

Вихідні дані

Єдине число – мінімальна кількість ударів мечем, або -1, якщо знищити Змія Горинича неможливо.

Приклад

Приклад вхідних даних	Приклад вихідних даних
3 3	9

Посилання на задачу: http://www.e-olimp.com/ua/problems/36

6. Нумерація

Для нумерації M сторінок в книжці використано N цифр. По заданому N вивести M або 0, якщо розв'язку не існує. Нумерація починається з першої сторінки.

Технічні умови

Вхідні дані

Число N.

Вихідні дані

Одне число M, якщо відомо, що в книжці не більше 1000 сторінок.

Приклад

Приклад вхідних даних	Приклад вихідних даних
27	18

Посилання на задачу: http://www.e-olimp.com/ua/problems/109

7. Годинник

Годинник з боєм відбиває кожної години таку кількість ударів, скільки їх є на циферблаті з цифрами від 1 до 12, та по одному разу тоді, коли хвилинна стрілка вказує на цифру 6. Знаючи початковий та кінцевий час однієї календарної доби (виражений в годинах і хвилинах), обчислити загальну кількість ударів за цей проміжок часу.

Технічні умови

Вхідні дані

0 <= H1 <= 23, 0 <= M1 <= 59, 0 <= H2 <= 23, 0 <= M2 <= 59

Вихідні дані

Одне число – кількість ударів.

Приклад

Приклад вхідних даних	Приклад вихідних даних
13 30 15 10	7

Посилання на задачу: http://www.e-olimp.com/ua/problems/111

8. Роботи

На деякому заводі вирішили модернізувати виробництво і закупили для цього роботів. Так як для обробки деталі потрібно виконання двох операцій, роботи також були двох типів: першу операцію виконували роботи типу A, а другу – роботи типу B. Щоб зекономити на покупці роботів, було вирішено купити не нових роботів останньої моделі, а вже таких, що були у використанні. В результаті, час, який різні роботи витрачали на виконання однієї і тієї ж операції, істотно відрізнявся, що призвело до труднощів у плануванні робіт.

Складіть програму, яка за заданим набором роботів обох типів визначає, за який мінімальний час вони зможуть опрацювати певну кількість деталей.

Технічні умови

Вхідні дані

У першому рядку задано натуральне число N, $1 \le N \le 100000$ – кількість деталей, які потрібно опрацювати.

У другому рядку міститься натуральне число Na, $1 \le Na \le 1000$ – кількість роботів, що виконують першу операцію.

У третьому рядку через пропуск задано Na натуральних чисел Ai, $1 \le Ai \le 100$ – час, який витрачає і-ий робот типу A на виконання операції.

У четвертому рядку задано натуральне число Nb, $1 \le Nb \le 1000$ – кількість роботів, які виконують другу операцію.

У п`ятому рядку задано через пропуск Nb натуральних чисел Bi, $1 \le Bi \le 100$ – час, який затрачає і-ий робот типу B на виконання операції.

Вихідні дані

У єдиному рядку вивести одне ціле число – мінімальний час, за який всі N деталей будуть оброблені спочатку роботом типу A, а потім роботом типу B. Часом передачі деталі від робота типа A роботу типу B знехтувати.

Приклад

Приклад вхідних даних	Приклад вихідних даних
6 3 1 3 2 2 2 3	9

Посилання на задачу: http://www.e-olimp.com/ua/problems/161

9. Юний садівник

Мама попросила Васю полити всі молоді деревця у саду. Вася знає, що поки дерева маленькі, їх потрібно дуже добре поливати. А ось скільки поливати - невідомо. Але Вася - дуже розумний хлопчик. Він уважно прочитав весь підручник ботаніки для середньої школи і вияснив, що полив прямо пропорційний кількості листочків на дереві. Для гарного росту дерев достатньо виливати під дерево щоденно по одному літру води для кожного листка.

На щастя Васі виявилось, що листки на деревах ростуть ярусами, причому на верхньому ярусі два листка, на другому - чотири, на наступному - шість, і так далі, на кожному наступному ярусі на два листки більше у порівнянні з попереднім. А на самій верхушці росте ще один листочок. Хитрий Вася послав молодшу сестричку Машеньку підрахувати кількість ярусів на кожному дереві, а Вас просить написати програму, яка для кожного дерева обрахує кількість літрів води для його поливу.

Технічні умови

Вхідні дані

У вхідному файлі задано число N ($0 \le N \le 1000$) — кількість ярусів на дереві.

Вихідні дані

Вивести кількість літрів води для поливу цього дерева.

Приклад

Приклад вхідних даних	Приклад вихідних даних
3	13

Посилання на задачу: http://www.e-olimp.com/ua/problems/248

10. Максимум

На днях пешокласник Василько навчився додавати числа. Йому цей процес дуже подобається, і він додає все підряд. Коли всі числа навколо виявляються доданими, Вася звертається до свого старшого брата Петра за новими числами. Після декількох звертань втомившись працювати генератором випадкових чисел, Петро придумав для Василька заняття, яке може надовго того відволікти.

Він запропонував Васильку знаходити суми цифр послідовних чисел — 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21 — і так далі, доки Васильку не надоїсть. Василько виявився у захваті від ідеї і прийнявся за роботу. За вчорашній день Василько знайшов суми цифр кожного з чисел від 1 до 115. Подивившись на результати молодшого брата, Петро помітив, що суми цифр послідовних чисел не є випадковими, часто вони йдуть підряд, але повністю закономірність він так і не зрозумів.

Щоб знайти закономірність, Петро вирішив дослідити крайні випадки, наприклад, яке з чисел дає максимальну суму цифр. Даних для чисел до 115 виявилось недостатньо для кінцевих висновків, і Петру прийшла в голову ідея для прискорення обчислень використати замість братика комп'ютер. Оскільки сам він в програмуванні не дуже сильний, він звернувся за розв'язком цієї задачі до Вас.

Технічні умови

Вхідні дані

У першому рядку вхідних даних знаходиться число N (1 <= N <= 2 147 483 647).

Вихідні дані

Виведіть число від 1 до N включно з максимальною сумою цифр. Якщо чисел з максимальною сумою цифр декілька, виведіть найбільше з них.

Приклад

Приклад вхідних даних	Приклад вихідних даних
115	99

Посилання на задачу: *http://www.e-olimp.com/ua/problems/474*

11. Бакси в банці

Папа Карло подарував Буратіно 1 долар на його перший день народження, а заощадливий Буратіно поклав подарунок у банку. Кожного наступного року папа Карло подвоював свій попередній подарунок і додавав до нього стільки доларів, скільки років виповнилось Буратіно, а той в свою чергу продовжував складати бакси у банку. На який N-й день народження у банці буде не менш ніж S доларів?

Технічні умови

Вхідні дані

Єдине число - значення S. $1 \le S \le 240$.

Вихідні дані

Шукане значення N.

Приклад

Приклад вхідних даних	Приклад вихідних даних
15	3

Посилання на задачу: http://www.e-olimp.com/ua/problems/127

12. Щасливі квитки

Підрахувати кількість щасливих квитків, у яких сума перших трьох цифр дорівнює N.

Щасливим квитком називається квиток з шестизначним номером в якого сума перших трьох цифр дорівнює сумі останніх трьох.

Технічні умови

Вхідні дані

У єдиному рядку записано одне натуральне число N ($N \le 27$).

Вихідні дані

Одне число - кількість таких щасливих квитків.

Приклад

Приклад вхідних даних	Приклад вихідних даних
1	9

Посилання на задачу: http://www.e-olimp.com/ua/problems/128

13. Мінімальна сума цифр

Скільки натуральних чисел з проміжку [M,N] мають найменшу суму цифр ?

Технічні умови

Вхідні дані

Два числа M і N. (1<=M<=N<=1000000).

Вихідні дані

Відповідь – одне число.

Приклад

Приклад вхідних даних	Приклад вихідних даних

1 100	3

Посилання на задачу: http://www.e-olimp.com/ua/problems/141

14. Просто Фібоначчі

Знайти N-е по порядку просте число Фібоначчі.

Технічні умови

Вхідні дані

Число N (1≤N≤10).

Вихідні дані

N-е по порядку просте число Фібоначчі.

Приклад

Приклад вхідних даних	Приклад вихідних даних
3	5

Посилання на задачу: http://www.e-olimp.com/ua/problems/192

15. Перетвореннння

Візьмемо деяке натуральне число N. Будемо змінювати його наступним чином: якщо число парне, то розділимо його на 2, якщо непарне, додамо 1. Після декількох таких змін ми завжди отримаємо число 1. Наприклад, з числа 11 отримується число 12, потім 6, 3, 4, 2 і, нарешті, 1. Таким чином, для отримання 1 з 11 потрібно виконати 6 перетворень.

Напишіть програму, яка зчитує натуральне число і виводить кількість перетворень даного числа до отримання 1.

Технічні умови

Вхідні дані

Число N (1 ≤ N ≤ 109).

Вихідні дані

Кількість перетворень.

Приклад

Приклад вхідних даних	Приклад вихідних даних
11	6

Посилання на задачу: http://www.e-olimp.com/ua/problems/388

16. Цікавий добуток

Визначити всі можливі значення добутку i*j, якщо цілочислові значення змінних i та j змінюються відповідно і від a до b та j від c до d (1 ≤ a, b, c, d ≤ 10).

Технічні умови

Вхідні дані

У єдиному рядку 4 числа через пропуск: a, b, c та d.

Вихідні дані

Єдине число - кількість можливих варіантів добутку.

Приклад

Приклад вхідних даних	Приклад вихідних даних
1 10 1 10	42

Посилання на задачу: http://www.e-olimp.com/ua/problems/916

17. Кількість іграшок

Задано кількість видів іграшок в магазині, кількість іграшок кожного виду та вартість іграшки кожного виду. Визначити загальну кількість іграшок, вартість яких менше 50 грн.

Технічні умови

Вхідні дані

У першому рядку задано кількість наявних у прейскуранті видів іграшок N ($0 \le N \le 1000$). У наступних N рядках задано по 2 числа через пропуск: спочатку кількість іграшок A ($0 \le A \le 1000$) чергового виду та їх ціна B ($0 < B \le 10000$) в грн.

Вихідні дані

У єдиному рядку вивести єдине число - відповідь до задачі.

Приклад

Приклад вхідних даних	Приклад вихідних даних
3 2 100.00 5 23.00 10 22.50	15

Посилання на задачу: http://www.e-olimp.com/ua/problems/927

Одновимірні та двовимірні масиви. Упорядкування масивів

1. Вибори вождя

Орки – одна з рас, що населяють світ Драенор. Не відрізняючись високим інтелектом, орки все ж таки славляться своєю силою та відвагою у бою. Щорічно орки з різних кланів збираються в Долині Сили для того, щоб вибрати вождя всієї Орди. На відміну від нерозумних людей, орки зневажають вибори шляхом голосування (да і, скажемо відверто, всі ці бюлетені, урни та виборчі дільниці - чужі і незрозумілі орку, який не тримав в руках нічого, крім палиці та сокири). Кандидати у вожді змагаються один з одним в чесних поєдинках. У кожному поєдинку приймають участь два претенденти, один з яких виходить з нього переможцем, а інший стає переможеним. Орк, що програв у одному з поєдинків, вибуває з числа претендентів і не может приймати участь в наступних поєдинках. Той, хто залишився останнім після всіх боїв кандидат і стає вождем Орди.

Старійшини орків завжди спостерігають за виборами і люблять передбачати, хто в них переможе. Проте далеко не завжди можна передбачити не те, что загальну перемогу на виборах, а навіть переможця в одному конкретному бою. Звичайно ж все залежить від сили суперників – хто сильніший, той і переможе, проте у випадку рівності сил може перемогти будь-хто з них – тут вже як зірки ляжуть.

Старійшини звернулись до вас з проханням написати програму для визначення кількості претендентів, які можуть стати вождями.

Технічні умови

Вхідні дані

У першому рядку вхідного файлу задано кількість N претендентів на звання вождя в цьому році ($1 \le N \le 1000000$), у другому – N цілих чисел в межах від 1 до 10000, кожне з яких визначає силу відповідного кандидата.

Вихідні дані

Вихідний файл повинен містити одне число – кількість претендентів, які можуть стати вождями.

Приклад

Приклад вхідних даних	Приклад вихідних даних
Приклад 1 5 1 2 3 4 5	Приклад 1 1
Приклад 2 6 2 2 2 2 2 2	Приклад2 6
Приклад 3 6 3 2 1 3 1 1	Приклад 3 2

Посилання на задачу: http://www.e-olimp.com/ua/problems/176

2. Сходинки

На кожній з N+2 сходинок сходів написане ціле число, причому на першій і останній сходинках написане число 0. На першій сходинці стоїть людина, якій потрібно піднятися на останню сходинку. За один крок вона може підніматись на довільну кількість сходинок, не більше K.

Підрахуємо суму всіх чисел, написаних на сходинках, на які наступила людина. Знайдіть найбільше можливе значення цієї суми.

Технічні умови

Вхідні дані

У першому рядку вхідного файлу записане число N, 0 <= N <= 1000. У другому рядку записано N цілих чисел, які не перевищують за модулем 1000, відкремлених пропусками - числа, написані на сходинках (за виключенням першої і останньої сходинки, на яких написані нулі). У третьому рядку записано максимальну величину кроку людини K, 1 <= K <= N.

Вихідні дані

Виведіть максимально можливу суму чисел, написаних на сходинках, на які наступила людина.

Приклад

Приклад вхідних даних	Приклад вихідних даних
3 1 -1 1 2	2

Посилання на задачу: http://www.e-olimp.com/ua/problems/262

3. Мінімальна зарплата

На малому підприємстві працює певна кількість працівників, але не менше двох – директора і головного бухгалтера. Знаючи зарплати всіх працівників, знайти найменшу зарплату на підприємстві.

Технічні умови

Вхідні дані

У єдиному рядку через пропуск задано заробітні плати працівників в гривнах. Всі вхідні дані – невід'ємні дійсні числа, відокремлені пропусками, їх кількість не перевищує 1000.

Вихідні дані

Єдине число – відповідь на задачу, виведена з двома цифрами після коми.

Приклад

Приклад вхідних даних	Приклад вихідних даних
1000.00 760.00	760.00

Посилання на задачу: http://www.e-olimp.com/ua/problems/352

4. Перестановка

Дано послідовність, що складається з N натуральних чисел. Написати програму, що визначає, чи є ця послідовність перестановкою перших N натуральних чисел.

Технічні умови

Вхідні дані

У єдиному рядку задано спочатку число N, а потім - N натуральних чисел через пропуск. N - не більше 10000, а кожне з чисел менше 2000000.

Вихідні дані

Вивести 0, якщо послідовність виявиться перестановкою, а якщо ні - мінімальне число, що не входить в цю послідовність.

Приклад

Приклад вхідних даних	Приклад вихідних даних
3 1 4 2	3

Посилання на задачу: http://www.e-olimp.com/ua/problems/354

5. Максимальна зарплата

На малому підприємстві працює певна кількість працівників, але не менше двох – директора і головного бухгалтера. Знаючи зарплати всіх працівників, знайти найбільшу зарплату на підприємстві.

Технічні умови

Вхідні дані

У єдиному рядку через пропуск задано заробітні плати працівників в гривнах. Всі вхідні дані – невід'ємні дійсні числа, відокремлені пропусками, їх кількість не перевищує 1000.

Вихідні дані

Єдине число – відповідь на задачу, виведена з двома цифрами після коми.

Приклад

Приклад вхідних даних	Приклад вихідних даних

1000.00 760.00 1000.00

Посилання на задачу: http://www.e-olimp.com/ua/problems/357

6. Збільшити на 2

Задано одновимірний масив A цілих чисел. Збільшити на 2 кожний невід'ємний елемент масиву.

Технічні умови

Вхідні дані

У першому рядку задано натуральне число h - кількість елементів масиву (h ≤ 100). У другому рядку через проміжок задано самі елементи масиву, значення кожного з яких за модулем не перевищує 100.

Вихідні дані

В єдиному рядку вивести через проміжок h чисел: нові значення елементів масиву, у тому ж порядку, в якому їх було задано.

Приклад

Приклад вхідних даних	Приклад вихідних даних
4 1 2 3 4	3 4 5 6

Пояснення: При виведенні результатів роботи числа повинні бути відокремлені одним пропуском, після останнього елементу пропуск повинен бути відсутнім.

Посилання на задачу: http://www.e-olimp.com/ua/problems/904

7. Перший не більший за 2,5

Задано одновимірний масив A дійсних чисел, пронумерованих від 1 до h. Визначити перший елемент масиву, який не перевищує 2.5.

Технічні умови

Вхідні дані

У першому рядку задано кількість елементів масиву h (0 < h ≤ 100), у наступному рядку задано h дісних чисел, відокремлених пропуском.

Вихідні дані

Вивести у одному рядку спочатку індекс знайденого першого вказаного елемента масива і через пропуск його значення з точністю 2 знаки після десяткової крапки. У випадку відсутності вказаного елементу в масиві вивести "Not Found" (без лапок).

Приклад

Приклад вхідних даних	Приклад вихідних даних
5 6 7.5 2.1 2.0 0	3 2.10

Посилання на задачу: http://www.e-olimp.com/ua/problems/907

8. Ті, що діляться на 6

Для N цілих чисел визначити суму й кількість додатніх чисел, які діляться на 6 без остачі.

Технічні умови

Вхідні дані

У першому рядку задано кількість чисел N (0 < N ≤ 100), у наступному рядку через пропуск задано самі числа, значення яких по модулю не перевищують 10000.

Вихідні дані

У єдиному рядку виведіть спочатку кількість вказаних чисел і через пропуск їх суму.

Приклад

Приклад вхідних даних	Приклад вихідних даних
3 12 15 18	2 30

Посилання на задачу: http://www.e-olimp.com/ua/problems/908

9. Середнє арифметичне додатних

Задано одновимірний масив A дійсних чисел, пронумерованих від 1 до h. Визначити середнє арифметичне додатних елементів масиву.

Технічні умови

Вхідні дані

У першому рядку задано число h - кількість елементів масиву (0 < h ≤ 100). У наступному рядку задано h елементів масиву, відокремлених пропусками. Значення елементів не перевищують по модулю 100.

Вихідні дані

У єдиному рядку вивести відповідь до задачі з точністю 2 знаки після десяткової крапки. У випадку відсутності у масиві вказаних елементів вивести повідомлення "Not Found".

Приклад

Приклад вхідних даних	Приклад вихідних даних
3 5.2 -2 4	4.60

Посилання на задачу: http://www.e-olimp.com/ua/problems/910

10. Модуль максимального

Задано одновимірний масив A дійсних чисел, пронумерованих від 1 до h. Визначити значення модуля максимального елемента масиву.

Технічні умови

Вхідні дані

У першому рядку задано натуральне число h - кількість елементів у масиві (h ≤ 100). У наступному рядку через пропуск задано h дійсних чисел - самі елементи масиву, значення яких не первищують по модулю 100.

Вихідні дані

Єдине число - відповідь до задачі, виведене з точністю 2 знаки після десяткової крапки.

Приклад

Приклад вхідних даних	Приклад вихідних даних
5 6 7.5 2.1 2.0 0	7.50

Посилання на задачу: http://www.e-olimp.com/ua/problems/914

11. Подвоєний мінімальний

Задано одновимірний масив A дійсних чисел пронумерованих від 1 до h. Обчислити подвоєне значення мінімального елемента масиву.

Технічні умови

Вхідні дані

У першому рядку задано натуральне число h - кількість елементів у масиві (h ≤ 100). У наступному рядку через пропуск h дійсних чисел - самі елементи масиву, значення кожного з яких по модулю не перевищує 100.

Вихідні дані

У єдиному рядку вивести відповідь до задачі з точністю до 2-х знаків після десяткової крапки.

Приклад

Приклад вхідних даних	Приклад вихідних даних
6 6 7.5 2.1 2.0 0 -3	-6.00

Посилання на задачу: http://www.e-olimp.com/ua/problems/917

12. Номер на 3

Задано одновимірний масив A дійсних чисел, пронумерованих від 1 до h. Визначити суму і кількість додатних елементів, номери яких діляться на 3 без остачі.

Технічні умови

Вхідні дані

У першому рядку задано натуральне число h - кількість елементів у масиві (h ≤ 100). У наступному рядку через пропуск h дійсних чисел - самі елементи масиву, значення кожного з яких по модулю не перевищує 100.

Вихідні дані

У єдиному рядку вивести спочатку кількість шуканих елементів масиву і через пропуск їх суму, обчислену з точністю до 2-х знаків після десяткової крапки.

Приклад

Приклад вхідних даних	Приклад вихідних даних
6 6 7.5 2.1 2.0 0 -3	1 2.10

Посилання на задачу: http://www.e-olimp.com/ua/problems/919

13. Від`ємні елементи

Задано одновимірний масив A дійсних чисел, пронумерованих від 1 до h. Визначити суму й кількість від’ємних елементів масиву.

Технічні умови

Вхідні дані

У першому рядку задано натуральне число h - кількість елементів масиву (h ≤ 100). У наступному рядку через пропуск задано h дійсних чисел - самі елементи масиву, значення яких не перевищують по модулю 100.

Вихідні дані

У єдиному рядку вивести спочатку кількість шуканих чисел і через пропуск їх суму, обчислену з точністю до 2-х знаків після десяткової крапки.

Приклад

Приклад вхідних даних	Приклад вихідних даних
5 6 -7.5 2.1 -2.0 0	2 -9.50

Посилання на задачу: http://www.e-olimp.com/ua/problems/921

14. Зсунь елементи

Задано одновимірний масив A цілих чисел довжини h. Зсунути елементи масиву циклічно праворуч на 1 крок.

Технічні умови

Вхідні дані

У першому рядку задано натуральне число h - кількість елементів масиву (h ≤ 100). У другому рядку задано самі елементи масиву, значення кожного з яких за модулем не перевищує 100.

Вихідні дані

В єдиному рядку вивести через проміжок h чисел: нові значення елементів масиву.

Приклад

Приклад вхідних даних	Приклад вихідних даних
4 1 2 3 4	4 1 2 3

Посилання на задачу: http://www.e-olimp.com/ua/problems/922

15. Сума найбільшого та найменшого

Задано одновимірний масив A цілих чисел. Визначити суму найменшого та найбільшого елементів масиву.

Технічні умови

Вхідні дані

У першому рядку задано натуральне число h - кількість елементів масиву (h ≤ 100). У другому рядку через проміжок задано самі елементи масиву, значення кожного з яких за модулем не перевищує 100.

Вихідні дані

В єдиному рядку вивести одне число - відповідь до задачі.

Приклад

Приклад вхідних даних	Приклад вихідних даних
4 1 2 3 4	5

Посилання на задачу: http://www.e-olimp.com/ua/problems/928

16. Розклад від "Дієз-Продукт"

Після завершення комп'ютеризації начального закладу, коли комп'ютери було встановлено у кожному кабінеті, директор та його заступник з навчальної частини зрозуміли, що без програми "Розклад" фірми "Дієз-продукт" їм аж ніяк не обійтись.

Судіть самі. У навчальному закладі N кабінетів, у яких потрібно провести K занять. Але біда в тому, що техніка у всіх кабінетах різна – тому у різних кабінетах можна працювати різний час. Згідно вимог техніки безпеки та санітарних вимог у кожному кабінеті встановлено графік обов'язкових прибирань протягом певного часу (свого для кожного кабінету, так як площа кабінетів різна, та й прибирають техпрацівники різного віку) після проведення вказаної кількості занять (знову ж таки, можливо і різної для різних кабінетів).

Допоможіть адміністрації навчального закладу визначити мінімальний час, за який вони зможуть провести всі заплановані заняття.

Технічні умови

Вхідні дані

У першому рядку через пропуск задано два числа: кількість кабінетів N та кількість занять K. У наступних N рядках через пропуск задано тривалість проведення заняття у i-му кабінеті Ui, кількість уроків у кабінеті Ci, після яких робиться технічна перерва, та її тривалість Ti. 1 <= N <= 50, 1 <= K <= 2000, 30 <= Ui <= 120, 1 <= Ci <= 100, 10 <= Ti <= 50.

Вихідні дані

Єдине число - мінімальний час, за який буде проведено всі заняття.

Приклад

Приклад вхідних даних	Приклад вихідних даних
3 100 10 30 40 30 100 30 20 50 25	570

Посилання на задачу: http://www.e-olimp.com/ua/problems/64

17. Спіраль

Числа від 1 до N2 записали до квадратної матриці NxN по спіралі починаючи з верхньої лівої клітинки за годинниковою стрілкою, як показано на малюнку. Знайти число, що знаходиться в I-му рядку і J-му стовпчику.

Технічні умови

Вхідні дані

У вхідному файлі натуральні числа N, I, J (1<=I,J<=N<=100).

Вихідні дані

До вихідного файлу потрібно записати число, що має координати I, J.

Приклад

Приклад вхідних даних	Приклад вихідних даних
5 4 2	23

Посилання на задачу: http://www.e-olimp.com/ua/problems/85

18. Підпроекти

Рамзі Сарнаєх заснував нову компанію приміських послуг, яку назвав Невирішені Ідеї (НІ). Поки що Рамзі в НІ ще не найняв працівників, тому він перші декілька місяців повинен працювати усім безпосередньо сам, доки він не може розширити свою компанію. Недавно він придбав деякі проекти від

урядових міністерств і розбив усі проекти на маленькі незалежні підпроекти з різними значеннями. Ми припускаємо, що всі підпроекти можуть бути виконаними за одиницю часу. На жаль, Рамзі, мав обмежений час і тому, будучи оптимістом хоче знати, скільки, у кращому випадку, він може заробити приймаючи цінніші підпроекти і відхиляючи інші.

Технічні умови

Вхідні дані

У першому рядку міститься ціле число – кількість тестових випадків. Далі йдуть дані, що розміщені в одному рядку для кожного тестового випадку. Кожен тестовий випадок починається з двох цілих чисел, якими є: доступний час Рамзі (T) та кількість підпроектів (P), відповідно (0 <= T, P <= 1000). Після цих двох чисел іде P невід'ємних цілих чисел (від 0 і до 32767, включно), які є значеннями вартості підпроектів. Всі числа в тестових випадках відокремлені пропуском.

Вихідні дані

Для кожного тестового випадку вивести у окремому рядку одне ціле число, яке дорівнює максимальній зробленій сумі грошей (сумі значень), яку можна досягнути в межах доступного часу Рамзі.

Приклад

Приклад вхідних даних	Приклад вихідних даних
3 3 5 1 1 1 1 1 4 2 161 5 4 7 8 2 9 17 4 4 10	3 166 44

Посилання на задачу: http://www.e-olimp.com/ua/problems/92

Літерні величини

1. Кількість слів

Є деяке речення на невідомій мові. Порахувати кількість слів у ньому. Літерами алфавіту у невідомій мові є літери латинського алфавіту та арабські цифри. Гарантується, що інших символів, крім пропусків та розділових знаків у реченні нема.

Технічні умови

Вхідні дані

У єдиному рядку дано речення на невідомій мові.

Вихідні дані

Єдине число - кількість слів у ньому.

Приклад

Приклад вхідних даних	Приклад вихідних даних
Hello, world!	2

Посилання на задачу: http://www.e-olimp.com/ua/problems/329

2. Слово чемпіон

Дано деяке речення на невідомій мові. Назвемо слово у ньому чемпіоном, якщо воно є паліндромом і кількість літер у ньому максимальна.

Літерами алфавіту у невідомій мові є літери латинського алфавіту та арабські цифри. Гарантується, що інші символи, крім пропусків та розділових знаків, у реченні відсутні.

Технічні умови

Вхідні дані

Речення на невідомій мові.

Вихідні дані

Номер слова чемпіона.

Приклад

Приклад вхідних даних	Приклад вихідних даних
Oo, it aaa is not bb.	3

Посилання на задачу: http://www.e-olimp.com/ua/problems/330

3. Речення чемпіон

Задано деякий абзац тексту на невідомій мові. Назвемо речення чемпіоном, якщо кількість паліндромів у ньому максимальна. Якщо таких речень декілька, то чемпіоном є те речення, яке зустрілось першим. Літерами алфавіту у невідомій мові є літери латинського алфавіту та арабські цифри. Гарантується, що інші символи, крім пропусків та розділових знаків, у тексті відсутні.

Технічні умови

Вхідні дані

Абзац тексту на невідомій мові.

Вихідні дані

Номер речення чемпіона.

Приклад

Приклад вхідних даних	Приклад вихідних даних
Oo, it aaa is not bb. More ana more non abc.	1

Посилання на задачу: http://www.e-olimp.com/ua/problems/331

4. Дитяча залізниця

Вітек не припиняє своїх забавок з кубиками. А тут ще татко подарував йому дитячу залізницю. Вітек розташовує якусь кількість кубиків по одному на вагончик і перед тим, як завершити гру, вирішує переформувати потяг таким чином, щоб отримане слово було лексикографічно найменшим із усіх можливих, але так як він до цього вже награвся, то може лише один раз від`єднати якусь кількість кубиків і паровозиком перевезти їх у хвіст свого потягу.

Яке слово отримає Вітек?

Технічні умови

Вхідні дані

Перший і єдиний рядок містить задане слово. Кубики у Вітека лише з великих літер латинського алфавіту і їх кількість не перевищує 500000.

Вихідні дані

Слово, утворене Вітеком.

Приклад

Приклад вхідних даних	Приклад вихідних даних
CBABC	ABCCB

Посилання на задачу: http://www.e-olimp.com/ua/problems/332

5. Дитяча залізниця - 2

Після того, як мама заборонила Вітеку займатись невідомою мовою і забрала у нього всі кубики, що не містили літер латинського алфавіту, він знайшов для себе нову забаву з дитячою залізницею. Спочатку він побудував декілька депо, куди міг відправляти зайві вагончики, але забирати їх звідти він не навчився. І ось, маючи деякий запас кубиків з великими літерами латинського алфавіту, він вирішив потай від мами зайнятись вивченням англійської мови. Але так як навіть словником з не рідною англійською мовою мама заборонила йому користуватись, Вітек вирішив скласти свій словник.

В словник він спочатку заносив слово, утворене з початкового розміщення кубиків з літерами на вагончиках, а далі утворював нові шляхом відправки в депо деякої кількості літер або з початку потягу, або з кінця, або з обох сторін, кожен раз використовуючи за початкове задане розміщення кубиків.

Скільки всього різних слів може бути у такому незвичному "англійському" словнику Вітека?

Технічні умови

Вхідні дані

У першому і єдиному рядку задано початкове розміщення кубиків. Кількість кубиків не перевищує 5000.

Вихідні дані

Кількість слів у словнику Вітека.

Приклад

Приклад вхідних даних	Приклад вихідних даних
ALPHABET	35

Посилання на задачу: http://www.e-olimp.com/ua/problems/333

6. Голосні

До голосних літер в латинському алфавіті відносяться літери A, E, I, O, U і Y. Інші літери вважаються приголосними. Напишіть програму, яка підраховує кількість голосних літер в тексті.

Технічні умови

Вхідні дані

У вхідному файлі міститься один рядок тексту, який складається лише з прописних латинських літер і пропусків. Двжина рядка не перевищує 100 символів.

Вихідні дані

У вихідний файл вивести одне ціле число – кількість голосних у вхідному тексті.

Приклад

Приклад вхідних даних	Приклад вихідних даних
COBRA	2

Посилання на задачу: http://www.e-olimp.com/ua/problems/494

7. Новий компілятор

Вам потрібно перетворити велику кількість старих програм для нової версії компілятора. Для цього потрібено замінити "->" на "." скрізь, крім коментарів. Коментарі в даній мові програмування починаються з символів "//" і продовжуються до кінця рядка. Напишіть програму, яка виконує таке перетворення.

Технічні умови

Вхідні дані

Вхідний файл містить від 1 до 500 рядків довжиною не більше 50 символів з ASCII-кодами від 32 до 127 – текст програми, яку потрібно перетворити.

Вихідні дані

У вихідний файл вивести перетворений текст програми.

Приклад

Приклад вхідних даних	Приклад вихідних даних

Test* t = new Test();
t->a = 1;
t->b = 2;
t->go(); // a=1, b=2 -> a=2, b=3
	Test* t = new Test();
	t.a = 1;
	t.b = 2;
	t.go(); // a=1, b=2 -> a=2, b=3

Посилання на задачу: http://www.e-olimp.com/ua/problems/506

8. Слова

Із слова «молоко» можна скласти слово «коло». Скільки слів з даного словника можна скласти використовуючи букви вихідного слова, кожну букву використовуючи не більше одного разу.

Технічні умови

Вхідні дані:

В першому рядку записано вихідне слово.
в другому - число N кількість слів словника.
Далі іде N рядків - слова словника.

Вихідні дані:

Одне число - кілкість слів, які можна скласти з вихідного слова.

Приклад

Приклад вхідних даних	Приклад вихідних даних
молоко 4 мило коло коліно око	2

Посилання на задачу: http://www.e-olimp.com/ua/problems/131

9. Таємнича записка

Недавно Маша та Катя взнали, що у світі існують злі хакери, які можуть легко відкрити чуже листування. Тому вирішили вони пересилати лише зашифровані повідомлення. З цією метою подруги стали використовувати перестановочний код, де кожна літера замінюється іншою. Наприклад:

Закодоване повідомлення: HPC PJVYMIY

Декодоване повідомлення: ACM CONTEST

У цьому прикладі виконані наступні заміни: H=A, P=C, C=M, J=O, V=N, Y=T, M=E и I=S.

Щоб не займатись кодуванням та декодуванням вручну, подруги просять Вас написати програму. Допоможіть дівчаткам!

Технічні умови

Вхідні дані

У першому рядку вхідних даних записане закодоване повідомлення. Другий рядок - 26 латинських літер верхнього регістру, що задають код для відповідного символа алфавіту: перший символ дає код для A, другий для B і так далі. Використовуються лише літери верхнього регістру. У закодованому повідомленні можуь з`явитись пропуски, які повинні бути збережені у вихідному рядку.

Вихідні дані

У вихідний файл вивести єдиний рядок, у якому міститься розшифроване повідомлення.

Приклад

Приклад вхідних даних	Приклад вихідних даних
Приклад 1	Приклад 1
HPC PJVYMIY BLMRGJIASOPZEFDCKWYHUNXQTV	ACM CONTEST
Приклад 2	Приклад 2
FDY GAI BG UKMY KIMHOTSQYRLCUZPAGWJNBVDXEF	THE SKY IS BLUE

Посилання на задачу: http://www.e-olimp.com/ua/problems/378

10. Римські числа

Підрахувати суму двох натуральних чисел A і B, записаних в римській системі числення. Відповідь записати також, в римській системі числення.

M = 1000, D = 500, C = 100, L = 50, X = 10, V = 5, I = 1 (Всі числа – менші 2000).

Технічні умови

Вхідні дані

В рядку записано два числа римською системою числення, між якими стоїть знак "+" .

Вихідні дані

Одне число, сума чисел також римською системою числення. Числа в римській системі числення записано великими латинськими літерами.

Приклад

Приклад вхідних даних	Приклад вихідних даних
VII+II	IX

Посилання на задачу: http://www.e-olimp.com/ua/problems/7

11. Кількість операцій

Визначити загальну кількість операцій додавання (+), віднімання (-) та множення (*) у заданому арифметичному виразі.

Технічні умови

Вхідні дані

У єдиному рядку задано арифметичний вираз без дужок та пропусків. Кількість символів у виразі не перевищує 250.

Вихідні дані

Єдине число - кількість вказаних операцій.

Приклад

Приклад вхідних даних	Приклад вихідних даних
-1+2*3+a	3

Посилання на задачу: http://www.e-olimp.com/ua/problems/901

12. Кількість слів

Визначити кількість слів у заданому фрагменті тексту.

Технічні умови

Вхідні дані

У єдиному рядку задано фрагмент тексту на англійській мові, кількість символів у якому не перевищує 250. Гарантується, що у тексті відсутні тире, дефіси, цифри і числа.

Вихідні дані

Єдине число - кількість слів у фрагменті.

Приклад

Приклад вхідних даних	Приклад вихідних даних
Hello world!	2

Посилання на задачу: http://www.e-olimp.com/ua/problems/909

13. Кількість речень

Визначити кількість речень у заданому фрагменті тексту.

Технічні умови

Вхідні дані

У єдиному рядку задано фрагмент тексту на англійській мові, кількість символів у якому не перевищує 250. Гарантується, що у тексті відсутні тире, дефіси, цифри і числа.

Вихідні дані

Єдине число - кількість речень у фрагменті.

Приклад

Приклад вхідних даних	Приклад вихідних даних
Hello world!	1

Посилання на задачу: http://www.e-olimp.com/ua/problems/912

14. Номер мобільного телефону

Задано номер мобільного телефону. Визначити, які цифри відсутні в цьому номері.

Технічні умови
Вхідні дані
У єдиному рядку задано номер мобільного телефону.
Вихідні дані
У першому рядку вивести кількість відсутніх у номері цифр. У другому рядку у порядку зростання вивести відсутні цифри, відокремлені пропуском.
Приклад

Приклад вхідних даних	Приклад вихідних даних
0631562976	2 4 8

Посилання на задачу: http://www.e-olimp.com/ua/problems/930

Процедури та функції

1. "Дзеркально прості" числа

Число назвемо "дзеркально простим", якщо воно є простим, і простим є число, записане тими ж цифрами у зворотному порядку.

На заданому проміжку [A, B] знайти кількість "дзеркально простих" чисел.

Технічні умови
Вхідні дані
На вхід подається 2 числа A та B. (1 <= A, B <= 1000)
Вихідні дані
Єдине число - шукана кількість "дзеркально простих" чисел
Приклад

Приклад вхідних даних	Приклад вихідних даних
10 25	3

Посилання на задачу: http://www.e-olimp.com/ua/problems/22

2. Рівень паліндромності

Задано натуральне число M. Якщо це не паліндром, то записуємо його у зворотному порядку та додаємо до заданого. Кроки повторюються, доки не буде отримано число-паліндром. Кількість виконаних операцій назвемо рівнем паліндромності заданого числа.

Знайти рівень паліндромності числа M.

Технічні умови
Вхідні дані
Єдине число M (0 < M < 10000).
Вихідні дані
Єдине число - рівень паліндромності.
Приклад

Приклад вхідних даних	Приклад вихідних даних
865	2

Посилання на задачу: http://www.e-olimp.com/ua/problems/29

3. Улюблені числа Діда Мороза

Дід Мороз полюбляв бавитись числами і цифрами. Найбільше він любив цифру 1, адже саме 1.01 починається Новий Рік. Йшли роки, але він

так і залишався забобонним - він не любив чисел, у яких після 1 стоїть 3, тобто утворюється число 13. На Новий рік він вирішив дати нове завдання: порахувати, скільки любимих Дідом Морозом простих чисел міститься на проміжку [A,B]?

Технічні умови

Вхідні дані:

Єдиний рядок, у якому міститься 2 числа: початок і кінець заданого проміжку. 1 < = A,B < = 500000

Вихідні дані:

Єдине число - кількість любимих Дідом Морозом простих чисел.

Приклад

Приклад вхідних даних	Приклад вихідних даних
9 23	4

Посилання на задачу: http://www.e-olimp.com/ua/problems/33

4. НСК

Знайти найменше спільне кратне (НСК) N натуральних чисел.

Технічні умови

Вхідні дані:

У першому рядку кількість чисел N (1 < N < 21).

У другому - через пропуск N натуральних чисел, що не перевищують 100.

Вихідні дані:

НСК цих чисел.

Приклад

Приклад вхідних даних	Приклад вихідних даних
2 2 3	6

Посилання на задачу: http://www.e-olimp.com/ua/problems/135

5. НСД

Знайти НСД (найбільший спільний дільник) N чисел.

Технічні умови

Вхідні дані:

У першому рядку кількість чисел N. (1 < N < 101)

У другому - через пропуск N натуральних чисел, що не перевищують 30000.

Вихідні дані:

НСД цих чисел.

Приклад

Приклад вхідних даних	Приклад вихідних даних
2 15 25	5

Посилання на задачу: http://www.e-olimp.com/ua/problems/137

6. Використовуй підпрограму

Обчислити суму і добуток N пар заданих дійсних чисел, скориставшись підпрограмою SumDob для обчислення суми і добутку двох дійсних чисел.

Технічні умови

Вхідні дані

У першому рядку задано натуральне число N - кількість пар чисел. У наступних N рядках через пропуск задано по 2 дісних числа. Всі вхідні дані по модулю не перевищують 100.

Вихідні дані

У N рядках вивести через пропуск по два числа: спочатку суму, а потім добуток чергової пари чисел. Результат виводити з точністю 4 знаки після десяткової крапки.

Приклад

Приклад вхідних даних	Приклад вихідних даних
2 6 7.5 2.1 2.0	13.5000 45.0000 4.1000 4.2000

Пояснення: Дозволяється використовувати 2 підпрограми, для знаходження суми і добутку відповідно кожну окремо.

Посилання на задачу: http://www.e-olimp.com/ua/problems/913

7. Використовуй функцію

Задано 3 дійсні числа x, y і z. Визначити min(max(x,y), max(y,z), x+y+z), скориставшись допоміжними функціями для обчислення мінімального та максимального елементів з двох заданих.

Технічні умови

Вхідні дані

У єдиному рядку задано 3 дійсні числа x, y і z, відокремлені пропуском. Значення чисел не перевищують по модулю 100.

Вихідні дані

Єдине число - відповідь до задачі, виведене з точністю 2 знаки після десяткової крапки.

Приклад

Приклад вхідних даних	Приклад вихідних даних
1.05 2.25 2.15	2.25

Посилання на задачу: http://www.e-olimp.com/ua/problems/920

8. Периметр і площа

Задано дійсні числа x1, y1, x2, y2, x3, y3, значення яких відповідають координатам вершин трикутника. Визначити периметр та площу трикутника, використовуючи підпрограму Vidrizok для обчислення довжин відрізка.

Технічні умови

Вхідні дані

У єдиному рядку через пропуск задано координати вершин трикутника: 6 чисел x1, y1, x2, y2, x3, y3, значення яких не перевищують по модулю 100.

Вихідні дані

У єдиному рядку через пропуск вивести спочатку периметр, а потім площу трикутника, обчислену з точністю до 4-х знаків після десяткової крапки.

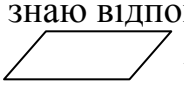
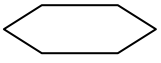
Приклад

Приклад вхідних даних	Приклад вихідних даних
3 2 7 6.5 10 1	19.3568 17.7500

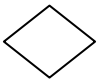
Посилання на задачу: http://www.e-olimp.com/ua/problems/925

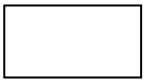
Тестові завдання

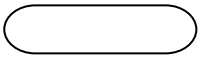
1. Алгоритм – це …:
   1) правила виконання певних дій;
   2) це скінченна та впорядкована послідовність вказівок (команд), формальне виконання яких дозволяє за обмежений час отримати розв'язок задачі
   3) це нескінченна та невпорядкована послідовність вказівок (команд);
   4) це нескінченна та впорядкована послідовність вказівок (команд), формальне виконання яких дозволяє за обмежений час отримати розв'язок задачі
2. Вкажіть найбільш повний перелік способів опису алгоритмів:
   1) графічним способом;
   2) природною мовою та графічним способом;
   3) природною мовою, графічним способом, засобами мови програмування;
   4) природною мовою та малюнковим способом;
3. Константа – це…:
   1) не можу сказати;
   2) комірка пам'яті, значення якої на протязі виконання програми може змінюватись;
   3) комірка пам'яті, значення якої на протязі виконання програми залишається незмінним
4. Змінна – це…:
   1) не можу сказати;
   2) комірка пам'яті, значення якої на протязі виконання програми може змінюватись;
   3) комірка пам'яті, значення якої на протязі виконання програми залишається незмінним
5. В яких випадках коментар записний вірно?
   1) {це є коментар};
   2) (*це є коментар*);
   3) //це є коментар;
   4) /*це є коментар*/;
6. Зарезервоване слово **program** означає:
   1) кінець програми;
   2) початок програми;
   3) заголовок програми;
   4) не знаю відповіді;
7. Вкажіть розділ підключення бібліотек:
   1) uses;
   2) label;
   3) const;
   4) type;
   5) var;

8. Вкажіть розділ оголошення змінних:
   1) uses;
   2) label;
   3) const;
   4) type;
   5) var;
9. Вкажіть розділ оголошення констант:
   1) uses;
   2) label;
   3) const;
   4) type;
   5) var;
10. Вкажіть розділ оголошення міток:
   1) uses;
   2) label;
   3) const;
   4) type;
   5) var;
11. Вкажіть розділ оголошення типів:
   1) uses;
   2) label;
   3) const;
   4) type;
   5) var;
12. Зарезервоване слово **begin** означає:
   1) кінець програми;
   2) початок програми;
   3) заголовок програми;
   4) не знаю відповіді;
13. Зарезервоване слово **end** означає:
   1) кінець програми;
   2) початок програми;
   3) заголовок програми;
   4) не знаю відповіді;
14. Блок  у блок-схемі означає:
   1) початок і кінець алгоритму;
   2) перевірку умови;
   3) введення, виведення даних;
   4) початок циклу з лічильником;
   5) обчислювальні дії;
15. Блок  у блок-схемі означає:
   1) початок і кінець алгоритму;
   2) перевірку умови;
   3) введення, виведення даних;

4) початок циклу з лічильником;
5) обчислювальні дії;

16. Блок  у блок-схемі означає:
1) початок і кінець алгоритму;
2) перевірку умови;
3) введення, виведення даних;
4) початок циклу з лічильником;
5) обчислювальні дії;

17. Блок  у блок-схемі означає:
1) початок і кінець алгоритму;
2) перевірку умови;
3) введення, виведення даних;
4) початок циклу з лічильником;
5) обчислювальні дії;

18. Блок  у блок-схемі означає:
1) початок і кінець алгоритму;
2) перевірку умови;
3) введення, виведення даних;
4) початок циклу з лічильником;
5) обчислювальні дії;

19. Позначте лише **цілочисельні** типи даних:
1) Word;
2) Boolean;
3) Integer;
4) Char;
5) Double;

20. Позначте лише **дійсні** типи даних:
1) Word;
2) Single;
3) Integer;
4) Char;
5) Double;

21. Позначте лише **символьні** типи даних:
1) Word;
2) Single;
3) String;
4) Char;
5) Double;

22. Вкажіть лише оператори **введення** даних:
1) readln(a,b);
2) read a;
3) read (n);
4) write(s);

5) writeln(a,’ ‘,b);

23. Вкажіть лише оператори **виведення** даних:
    1) readln(a,b);
    2) read a;
    3) read (n);
    4) write(s);
    5) writeln(a,’ ‘,b);
24. Вкажіть лише правильний вивід тексту на екран:
    1) write (* привет *);
    2) write (’ привет ’);
    3) write (« привет »);
    4) write { ‘ привет ‘ };
25. *readln(n)* – це команда :
    1) присвоювання;
    2) введення даних;
    3) виведення даних;
    4) команда перевірки умови;
26. *writeln(n)* – це команда :
    1) присвоювання;
    2) введення даних;
    3) виведення даних;
    4) команда перевірки умови;
27. *n:=10; – це* команда :
    1) присвоювання;
    2) введення даних;
    3) виведення даних;
    4) команда перевірки умови;
28. До операцій *відношення* належать:
    1) +, -, *, /;
    2) div, mod;
    3) <, >, <=, >=, <>;
    4) не знаю відповіді;
29. Вкажіть вірний запис виразу $y = \log_5 x$ на мові програмування Pascal:
    1) y:=log5(x);
    2) y:=ln(x)/ln(5);
    3) y:=log(5)(x);
    4) не знаю вірної відповіді;
30. Вкажіть вірний запис виразу $y = a^n$ на мові програмування Pascal:
    1) y:=exp(n*ln(a));
    2) y:=a^n;
    3) y:=a*n;
    4) вірної відповіді немає;

31. Вкажіть вірний запис виразу $y=\sqrt{\frac{a+b}{a^n}}\cdot\log_5(2x-6)+\sin^2(2x+5)$ на мові програмування Pascal:
    1) y:=sqrt((a+b)/exp(n*ln(a)))*ln(2*x-6)/ln(5)+sqr(sin(2*x+5));
    2) y:=sqrt((a+b)/exp(n*ln(a))*ln(2*x-6)/ln(5))+sqr(sin(2*x+5));
    3) y:=sqrt((a+b)/exp(n*lna))*ln(2*x-6)/ln(5)+sqr(sin(2*x+5));
    4) y:=sqrt((a+b)/exp(n^a)))*ln(2*x-6)/ln(5)+sqr(sin(2*x+5));
32. Вкажіть вірний запис виразу $y=\left|\frac{\sqrt{a+b}}{(a-b)^2}\cdot\left|3x-5\sin(3a+b)\right|\right|$ на мові програмування Pascal:
    1) y:=abs(sqrt(a+b)/sqr(a-b)*abs(3*x-5*sin(3*a+b)));
    2) y:=abs(sqrt(a+b)/sqr(a-b)*abs(3x-5sin(3a+b)));
    3) y:=abs(sqrt(a+b)/(a-b)^2*abs(3*x-5*sin(3*a+b)));
    4) y:=abs(sqr(a+b)/sqr(a-b)*abs(3*x-5*sin(3*a+b)));
33. Математична функція **div** в результаті дає:
    1) Остачу від ділення націло;
    2) Цілу частину від ділення націло;
    3) Зменшення на одиницю;
    4) Збільшення на одиницю;
34. Математична функція **mod** в результаті дає:
    1) Остачу від ділення націло;
    2) Цілу частину від ділення націло;
    3) Зменшення на одиницю;
    4) Збільшення на одиницю;
35. Математична функція **inc** в результаті дає:
    1) Остачу від ділення націло;
    2) Цілу частину від ділення націло;
    3) Зменшення на одиницю;
    4) Збільшення на одиницю;
36. Математична функція **dec** в результаті дає:
    1) Остачу від ділення націло;
    2) Цілу частину від ділення націло;
    3) Зменшення на одиницю;
    4) Збільшення на одиницю;
37. Математична функція **sqrt (x)** в результаті дає:
    1) квадрат деякого числа x;
    2) модуль числа x;
    3) корінь квадратний з деякого числа x;
    4) немає вірної відповіді;
38. Математична функція **sqr (x)** в результаті дає:
    1) квадрат деякого числа x;
    2) модуль числа x;
    3) корінь квадратний з деякого числа x;
    4) немає вірної відповіді;

39. Математична функція **abs (x)** в результаті дає:
    1) квадрат деякого числа x;
    2) модуль числа x;
    3) корінь квадратний з деякого числа x;
    4) немає вірної відповіді;
40. Що виконує математична функція **round (x)**?
    1) визначає, до парних чи непарних чисел відноситься дане число;
    2) визначає цілу частину числа: від числа відкидається дробова частина (результат дії даної функції набуває типу Integer або Longint);
    3) визначає дробову частину числа: відкидається ціла частина, а дробова частина записується як ціле число;
    4) округлення числа до цілого, згідно математичних правил округлення;
    5) визначає цілу частина числа (результат дії функції Int залишається типу real)
41. Що виконує математична функція **int (x)**?
    1) визначає, до парних чи непарних чисел відноситься дане число;
    2) визначає цілу частину числа: від числа відкидається дробова частина (результат дії даної функції набуває типу Integer або Longint);
    3) визначає дробову частину числа: відкидається ціла частина, а дробова частина записується як ціле число;
    4) округлення числа до цілого, згідно математичних правил округлення;
    5) визначає цілу частину числа (результат дії функції Int залишається типу real)
42. Що виконує математична функція **odd (x)**?
    1) визначає, до парних чи непарних чисел відноситься дане число;
    2) визначає цілу частину числа: від числа відкидається дробова частина (результат дії даної функції набуває типу Integer або Longint);
    3) визначає дробову частину числа: відкидається ціла частина, а дробова частина записується як ціле число;
    4) округлення числа до цілого, згідно математичних правил округлення;
    5) визначає цілу частину числа (результат дії функції Int залишається типу real)
43. Що виконує математична функція **frac (x)**?
    1) визначає, до парних чи непарних чисел відноситься дане число;
    2) визначає цілу частину числа: від числа відкидається дробова частина (результат дії даної функції набуває типу Integer або Longint);
    3) визначає дробову частину числа: відкидається ціла частина, а дробова частина записується як ціле число;

4) округлення числа до цілого, згідно математичних правил округлення;
5) визначає цілу частину числа (результат дії функції Int залишається типу real)

44. Що виконує математична функція **trunc (x)**?
1) визначає, до парних чи непарних чисел відноситься дане число;
2) визначає цілу частину числа: від числа відкидається дробова частина (результат дії даної функції набуває типу Integer або Longint);
3) визначає дробову частину числа: відкидається ціла частина, а дробова частина записується як ціле число;
4) округлення числа до цілого, згідно математичних правил округлення;
5) визначає цілу частину числа (результат дії функції Int залишається типу real)

45. Назвіть усі базові алгоритмічні конструкції:
1) слідування;
2) використання підпрограм;
3) розгалуження;
4) цикли;
5) усе вірно

46. Лінійна структура – передбачає:
1) багаторазове виконання певних дії;
2) виконання дій в залежності від певних умов;
3) послідовність команд, що виконуються підряд один за одним;
4) немає вірної відповіді.

47. Чому буде рівний вираз exp(3*ln(2)):
1) 4;
2) 8;
3) 6;
4) немає вірної відповіді.

48. Коментарі на мові програмування Pascal записуються у:
1) круглих дужках;
2) квадратних дужках
3) фігурних дужках;
4) між службовими словами begin … end.

49. Оператори у програмі відокремлюються один від одного:
1. двома крапками;
2. крапкою з комою;
3. комою;
4. пропуском.

50. У програмі обчислення суми елементів арифметичної прогресії, коли відомий перший її член, різниця та кількість членів, **var a,d,s : real; n:integer; begin readln(a,d,n); s:=…; writeln(s); end.** в операторі

присвоювання не вказаний арифметичний вираз. Його потрібно записати:

1) a*n+d*(n–1)*n/2;
2) a*(n+d*(n–1)*)n/2;
3) a +d*(n–1)*n/2;
4) a*n/2+d*(n–1)*n/2.

51. Відмінність у використанні константи від змінної визначається:
    1) записом ідентифікаторів;
    2) типом даних;
    3) можливістю зміни значення;
    4) неможливістю зміни значення.
52. Скільки змінних описано у фрагменті програми:

```
uses crt;
const a=5; b=7;
var x1, y1, x2 :real; t1, t2, r1, r2 : integer;
```

3. 2;
4. 3;
5. 7;
6. 9.

53. Вкажіть результат обчислення виразу 5+26 div 10*10:
3. 25;
4. 5;
5. 65;
6. немає вірної відповіді.

54. Вкажіть тип результату обчислення виразу 32 mod 10 – 5*7:
3. real;
4. word;
5. integer;
6. longint;
7. byte;
8. shortint.

55. Вкажіть відповідність між функцією та описом її призначення:

2. Inc (k);	а) збільшує значення змінної К на Р;
3. Dec (k);	б) зменшує значення змінної К на Р;
4. Inc(k,p);	в) збільшує значення змінної К на 1;
5. Dec(k,p);	г) зменшує значення змінної К на 1.

56. Величини, значення яких встановлені в описовій частині програми і в процесі виконання програми не змінюються називаються (у мові програмування Pascal):
    1) Змінними;
    2) Константами;
    3) Функціями;
    4) Мітками;
    5) Типами.

57. Операції div та mod у мові програмування Pascal можна виконувати лише з:
    1) дійсними величинами
    2) цілими величинами
    3) булівськими величинами
    4) символьними величинами
    5) рядковими величинами;
58. Чому дорівнюватиме x у мові програмування Pascal при виконанні наступних дій: x:=(396 mod 100) div 10?
    1) 3
    2) 9
    3) 6
    4) 36
    5) 39
59. Значення виразу sqr(x)-4*x+(x mod 10), у мові програмування Pascal при x:=15 дорівнює:
    1) 170
    2) 166
    3) 0
    4) 165
    5) 180
60. Ідентифікатор – це
    1) сукупність символів, які дозволяється використовувати при побудові опису програм;
    2) це назва (ім'я), яку користувач надає об'єктам, наприклад, змінним, сталим, функціям;
    3) сукупність правил (опису) побудови вказівок алгоритмів деякою мовою програмування;
    4) зарезервовані слова;
    5) неподільний елемент мови: слово, число, символ, операція;
61. Дійсне десяткове число 0.8500E+02 записане у форматі з плавоючою крапкою, у форматі з фіксованою десятковою крапкою буде записане як:
    1) 85
    2) 0.0085
    3) 0.85
    4) 850
    5) 8500
62. Виберіть правильний запис виразу $e^{x+5} + \cos^2(5x)$:
    1) exp(x+5)+sqr(cos(5*x))
    2) exp(x+5)+sqr(cos(5x))
    3) exp(x+5)+sqrt(cos(5*x))
    4) exp(x+5)+sqrt(cos5*x)
    5) exp(x+5)+(cos(5*x))^2

63. Компіляція програми – це…

- перевірка програми на відповідність до правил її написання;
- створення тексту програми;
- пошук помилок в програмі;
- переклад програми на «машинну мову»;
- виконання програми;

64. Оператор організації вводу даних з клавіатури в системі програмування Turbo Pascal записується з використанням службового слова…

2. Write
3. Input
4. Read
5. Deffn
6. Reset

65. Результатом обчислення виразу abs(x-23)/sqrt(sqr(y)+12), у мові програмування Pascal при x:= 7 та y:=2 буде:

2. -4
3. 16
4. 1
5. 24
6. 4

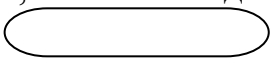
66. Результатом обчислення логічного виразу є:

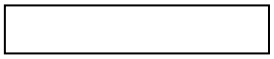
2. новий логічний вираз
3. одне з двох значень TRUE або FALSE
4. арифметичний вираз
5. дійсне значення
6. значення, що належить лише до цілочисельного типу

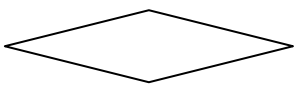
67. Установіть відповідність між функціями та типом змінної, результатом виконання якої він є у мові програмування Pascal:

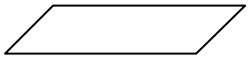
А) odd	1) цілочисельний
Б) div	2) дійсний
В) copy	3) логічний
Г) sqrt	4) рядковий

68. Установіть відповідність між зображенням елемента блок-схеми та типу дії, якій воно відповідає:

А) 

Б) 

В) 

Г) 

1. введення – виведення
2. розгалуження, початок циклу з передумовою або кінець циклу з післяумовою
3. початок – кінець програми
4. обчислення

69. Установіть відповідність між функціями та типом змінної, результатом виконання якої він є у мові програмування Pascal:

А) and	1) цілочисельний

Б) mod 2) дійсний
В) delete 3) логічний
Г) ln 4) рядковий

70. Установіть відповідність між функціями та типом змінної, результатом виконання якої він є у мові програмування Pascal:
А) or 1) цілочисельний
Б) trunc 2) дійсний
В) insert 3) логічний
Г) sin 4) рядковий

71. Вкажіть правильну послідовність етапів розв'язку задачі за допомогою ЕОМ:
А) розробка алгоритму;
Б) тестування та налагодження програми;
В) постановка задачі;
Г) опис алгоритму мовою програмування;
Д) експлуатація програми;
Е) побудова математичної моделі.

72. Вкажіть правильну послідовність розділу опису у мові програмування Pascal:
А) TYPE;
Б) VAR;
В) CONST;
Г) USES;
Д) LABEL;

73. Вкажіть правильну послідовність розділів структури програми у мові програмування Pascal:
А) Частина опису процедур та функцій;
Б) Основний блок програми (розділ операторів);
В) Описова частина;
Г) Заголовок програми;

74. Розгалуження – це …:
1) це вид управляючої структури, що передбачає можливість вибору з декількох варіантів, для кожного з яких в залежності від певної умови виконується своя послідовність операторів;
2) це вид управляючої структури, що передбачає можливість виконання групи команд декілька разів;
3) це вид управляючої структури, що передбачає можливість виконання команд одна за одною;
4) немає вірної відповіді;

75. Загальний вигляд умовного оператора (повна форма) :
1) **if** <умова> **then** <оператор 1> **else** <оператор2>;
2) **if** <умова> **then** <оператор 1>; **else** <оператор2>;
3) **if** <умова> **then** <оператор 1>;
4) **if** <умова> **then** <оператор 1> - **else** <оператор2>;

76. Загальний вигляд умовного оператора (скорочена форма) :

1) **if** <умова> **then** <оператор 1> **else** <оператор2>;
2) **if** <умова> **then** <оператор 1>; **else** <оператор2>;
3) **if** <умова> **then** <оператор 1>;
4) **if** <умова> **then** <оператор 1> - **else** <оператор2>;

77. Службове слово **if** переводиться як:
1) інакше;
2) то;
3) умова;
4) якщо;

78. Службове слово **then** переводиться як:
1) інакше;
2) то;
3) умова;
4) якщо;

79. Службове слово **else** переводиться як:
1) інакше;
2) то;
3) умова;
4) якщо;

80. В результаті виконання наступних команд що буде виведено на екран:

```
begin
   n:=5;
   if n>5 then n:=n+5 else n:=n-3;
   writeln(n);
end.
```

1) 10;
2) 5;
3) 2;
4) 4;
5) нічого;

81. В результаті виконання наступних команд що буде виведено на екран:

```
begin
   n:=5;
   if n<=5 then n:=n+5 else n:=n-3;
   writeln(n);
end.
```

1) 10;
2) 5;
3) 2;
4) 4;
5) нічого;

82. В результаті виконання наступних команд що буде виведено на екран:

```
begin
   x:=0;
   if x<>5 then y:=10 else y:=0;
```

writeln(y);
end.

1) 10;
2) 5;
3) 0;
4) y;
5) нічого;

83. Виберіть правильний варіант запису обчислення функції:

$$y = \begin{cases} x^2 + 5x - e^x, & x < 2 \\ \sin(x+5) + \ln(x), & 2 \le x < 4 \\ 0, & x \ge 4 \end{cases}$$

1) if x<2 then y:=sqr(x)+5*x-exp(x) else if x<4 then y:=sin(x+5)+ln(x) else y:=0;
2) if x<=2 then y:=sqr(x)+5x-exp(x) else if x<=4 then y:=sin(x+5)+ln(x) else y:=0;
3) if x<=2 then y:=sqr(x)+5*x-exp(x) else if x<4 then y:=sin(x+5)+ln(x) else y:=0;
4) if x<2 then y:=sqr(x)+5*x-exp(x) else if x<=4 then y:=sin(x+5)+ln(x) else y:=0;
5) if x>2 then y:=sqr(x)+5*x-exp(x) else if x<4 then y:=sin(x+5)+ln(x) else y:=0;

84. Яку умову можна використати при перевірці чи є число парним?

1) x mod 2=1;
2) x mod 2 <>1;
3) x mod 2=0;
4) x mod 2 <>0;
5) x div 2=0;

85. Яку умову можна використати при перевірці чи є число непарним?

1) x mod 2=1;
2) x mod 2 <>1;
3) x mod 2=0;
4) x mod 2 <>0;
5) x div 2=0;

86. Які умови називаються простими?

1) немає вірної відповіді;
2) це ті умови, в яких використовуються лише операції відношення;
3) це ті умови, в яких використовуються лише логічні операції;
4) це ті умови, в яких використовуються операції відношення та логічні операції;

87. Які умови називаються складними?

1) немає вірної відповіді;
2) це ті умови, в яких використовуються лише операції відношення;
3) це ті умови, в яких використовуються лише логічні операції;

4) це ті умови, в яких використовуються операції відношення та логічні операції;

88. Яку умову можна використати при перевірці чи є число додатнім?
1) x>0;
2) x>=0;
3) x<0;
4) x<=0;
5) немає вірної відповіді;

89. Яку умову можна використати при перевірці чи є число недодатнім?
1) x>0;
2) x>=0;
3) x<0;
4) x<=0;
5) немає вірної відповіді;

90. Яку умову можна використати при перевірці чи є число від'ємним?
1) x>0;
2) x>=0;
3) x<0;
4) x<=0;
5) немає вірної відповіді;

91. Яку умову можна використати при перевірці чи є число невід'ємним?
1) x>0;
2) x>=0;
3) x<0;
4) x<=0;
5) немає вірної відповіді;

92. Яку умову можна використати при перевірці чи входить число x в заданий діапазон [a;b]?
1) (x>a) and (x<b);
2) (x>=a) or (x<=b);
3) (x>=a) and (x<b);
4) (x>=a) and (x<=b);
5) (x>a) and (x<=b);
6) (x>=a) or (x<b);

93. Яку умову можна використати при перевірці чи входить число x в заданий діапазон (a;b]?
6. (x>a) and (x<b);
7. (x>=a) or (x<=b);
8. (x>=a) and (x<b);
9. (x>=a) and (x<=b);
10. (x>a) and (x<=b);
11. (x>=a) or (x<b);

94. Яку умову можна використати при перевірці чи входить число x в заданий діапазон (a;b)?
1. (x>a) and (x<b);
2. (x>a) or (x<b);
3. (x>=a) and (x<b);
4. (x>=a) and (x<=b);
5. (x>a) and (x<=b);
6. (x>=a) or (x<b);

95. Вкажіть відповідність між символами та описами операцій відношення:

1) >; а) рівність;
2) <; б) не рівність;
3) <>; в) більше;
4) >=; г) менше;
5) <=; д) більше або рівне;
6) =; е) менше або рівне;

96. Яке значення матиме змінна логічного типу C, якщо A:=TRUE; B:=TRUE; C:=A AND B;
1) TRUE;
2) FALSE;
3) Немає вірної відповіді;

97. Знайдіть у програмі рядок з помилкою:
1) var a, b : integer;
2) begin
3) readln(a,b);
4) if a>b j
5) then writeln(a);
6) else writeln(b);
7) end.

98. Вкажіть логічний вираз із результатом true при виконанні умови – "ціле число k кратне 7":
1) k mod 7=0;
2) k mod 7 <>0;
3) k div 7 <> 0;
4) k div 7=0;
5) k mod 7 ><0;
6) k div 7 ><0;

99. Вкажіть оператор, який виконається в операторі case, якщо:

```
a:=5;
case a of
0,1: x:=x+1;
2..7: x:=x+2;
8..10: x:=x+3;
end;
```

1) x:=x+1;
2) x:=x+2;
3) x:=x+3;
4) жоден не виконається;

100. Вкажіть значення, яке матиме змінна d, якщо:

```
x:=15; d:=10;
case x mod 4 of
0 : d:=x;
1: d:=x+1;
2: d:=x+2;
3: x:=x+3;
end;
```

2. d=15;
3. d=16;

4. d=17;
5. d=18;
6. d=10;

101. Вкажіть оператор, який виконається в операторі case, якщо:

```
a:=17;
case a mod 4 of
1 : x:=2*a;
2: x:=4*a;
3: x:=9*a;
end;
```

2. *x:=2*a*;
3. *x:=4*a*;
4. *x:=9*a*;
5. жоден не виконається;

102. Вкажіть значення, яке матиме змінна k, якщо:

```
m:=17; k:=10;
case m div 6 of
1 : k:=k+3;
2: k:=k-1;
3: k:=k*2;
4: k:=k+2;
end;
```

2. 9;
3. 10;
4. 12;
5. 13;
6. 20;

103. Вкажіть правильну послідовність виконання операторів у мові програмування Pascal, щоб в результаті змінна c набула значення 5, якщо a=12; b=7:

А) *c:=a+b*	1) Б
Б) *if b>=a then*	2) А
В) *c:=a-b*	3) Г
Г) *else*	4) В

104. Цикл – це …:

1) це вид управляючої структури, що передбачає можливість вибору з декількох варіантів, для кожного з яких в залежності від певної умови виконується своя послідовність операторів;
2) це вид управляючої структури, що передбачає можливість виконання групи команд деяку скінченну кількість разів;
3) це вид управляючої структури, що передбачає можливість виконання команд одна за одною;
4) немає вірної відповіді;

105. Цикл з параметром у загальному має вигляд:

1) **for** <параметр>:=<вираз1> **to** <вираз2> **do** <тіло циклу>;
2) **for** <параметр>=<вираз1> **to** <вираз2> **do** <тіло циклу>;
3) **for** <параметр>:=<вираз1> down**to** <вираз2> **do** <тіло циклу>;
4) **for** <параметр>:<вираз1> **to** <вираз2> **do** <тіло циклу>;

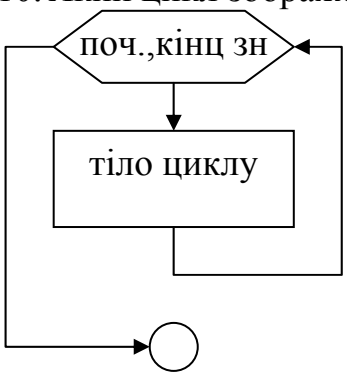
106. Чому рівний крок у циклі з параметром?
1) 1;
2) 2;
3) 3;
4) -1;

107. Який цикл називається циклом з параметром?
1) while;
2) repeat… until;
3) for;
4) немає вірної відповіді;

108. Який цикл називається циклом з передумовою?
1) while;
2) repeat… until;
3) for;
4) немає вірної відповіді;

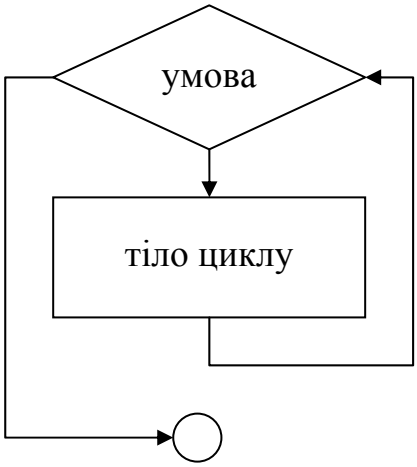
109. Який цикл називається циклом з післяумовою?
1) while;
2) repeat… until;
3) for;
4) немає вірної відповіді;

110. Який цикл зображений на малюнку?



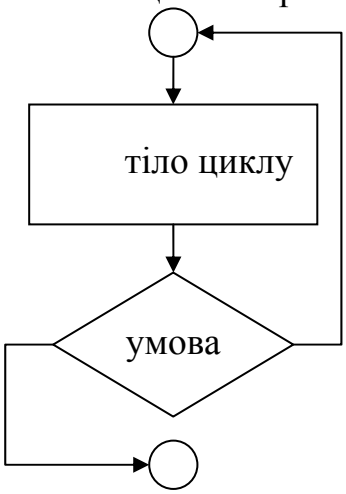
1) while;
2) repeat… until;
3) for;
4) немає вірної відповіді;

111. Який цикл зображений на малюнку?



1) while;
2) repeat… until;
3) for;
4) немає вірної відповіді;

112. Який цикл зображений на малюнку?



1) while;
2) repeat… until;
3) for;
4) немає вірної відповіді;

113. Якого типу може бути лічильник у циклі з параметром?
1) лише дійсного;
2) лише цілочисельного;
3) числового;
4) будь-якого;

114. Яка команда дає можливість перейти до наступного повторення?
1) next i;
2) break;
3) continue;
4) goto;

115. Яка команда дає можливість у будь-який момент часу вийти з циклу?
1) next i;
2) break;
3) continue;

4) goto;

116. Цикл з передумовою у загальному має вигляд:
   1) **for** <параметр>:=<вираз1> **to** <вираз2> **do** <тіло циклу>;
   2) **repeat** <оператор> **until** <умова>;
   3) **while** <умова> **do** <оператор>;
   4) немає вірної відповіді;
117. Цикл з післяумовою у загальному має вигляд:
   1) **for** <параметр>:=<вираз1> **to** <вираз2> **do** <тіло циклу>;
   2) **repeat** <оператор> **until** <умова>;
   3) **while** <умова> **do** <оператор>;
   4) немає вірної відповіді;
118. Параметр у циклі for може приймати значення:
   1) 1, 3, 5, 7, 9…;
   2) 1, 2, 3, 4, 5…;
   3) …10, 9, 8, 7, 6, …;
   4) 2, 4, 6, 8,…;
119. Скільки разів виконається цикл *for i:=12 to 19 do…*:
   1) 8 разів;
   2) 9 разів;
   3) 7 разів;
   4) жодного разу;
120. Скільки разів виконається цикл *for i:=12 to 7 do…*:
   1) 8 разів;
   2) 6 разів;
   3) 7 разів;
   4) жодного разу;
121. Скільки разів виконається цикл *for i:=12 downto 7 do…*:
   1) 8 разів;
   2) 6 разів;
   3) 7 разів;
   4) жодного разу;
122. Скільки разів виконається цикл *i:=10; while i>10 do i:=i+5;:*
   1) 8 разів;
   2) 6 разів;
   3) 7 разів;
   4) жодного разу;
123. Скільки разів виконається цикл *i:=3; while i<20 do i:=i+5;:*
   1) 4 рази;
   2) 3 рази;
   3) 5 раз;
   4) жодного разу;
124. Скільки разів виконається цикл *i:=3; while i<20 do s:=i+5;:*
   1) 4 рази;
   2) 3 рази;
   3) нескінченну кількість разів (цикл буде вічним);

4) жодного разу;

125. Чому буде рівна змінна і в результаті виконання такого фрагменту програми *i:=3; while i<20 do i:=i+5;:*
1) 20;
2) 23;
3) 17;
4) немає вірної відповіді;

126. Скільки разів виконається цикл *i:=3; repeat s:=i+5;until i>3;*
1) 1 раз;
2) 2 рази;
3) нескінченну кількість разів (цикл буде вічним);
4) жодного разу;

127. Чому буде рівна змінна і в результаті виконання такого фрагменту програми *i:=3; repeat i:=i+5;until i>20;* :
1) 20;
2) 23;
3) 17;
4) немає вірної відповіді;

128. Скільки разів виконається цикл *i:=3; repeat i:=i+5;until i>20;*
1) 4 рази;
2) 3 рази;
3) нескінченну кількість разів (цикл буде вічним);
4) жодного разу;

129. Чому буде рівна змінна і в результаті виконання такого фрагменту програми *i:=3; repeat i:=i+6;until i>20;* :
1) 21;
2) 15;
3) 18;
4) немає вірної відповіді;

130. Скільки разів виконається цикл: *i:=23; repeat i:=i-5; until i>23*;
1) 23;
2) 1;
3) жодного разу;
4) нескінченну кількість разів;

131. Чому буде рівна змінна sum після виконання фрагменту програми: *sum:=0; i:=5; repeat sum:=sum+4; i:=i+3; until i>11;*
1) 12;
2) 4;
3) 8;
4) цикл нескінченний;

132. Скільки разів виконається цикл: *i:=15; repeat i:=i-5; until i>5*;
1. 3;
2. 1;
3. жодного разу;
4. нескінченну кількість разів;

133. Чому буде рівна змінна sum після виконання фрагменту програми: *sum:=0; i:=1; repeat sum:=sum+3; i:=i-1; until i>11;*
   1) 3;
   2) 6;
   3) 8;
   4) цикл нескінченний;
134. Оператор while може бути використаний для опису
   1) будь-яких циклічних процесів;
   2) циклічних процесів, у яких відоме число повторень;
   3) циклічних процесів, які відбуваються при виконанні певної умови;
   4) циклічних процесів, які відбуваються при невиконанні певної умови;
135. При виконанні оператора while оператори тіла циклу
   1) завжди виконуються принаймні один раз;
   2) виконаються заздалегідь відому кількість разів;
   3) можуть не виконатись жодного разу;
136. Вкажіть помилку, якщо така має місце, у фрагменті програми, за яким на екран виводяться в рядок парні числа з проміжку [0;20]

```
i:=0;
while i<=20 do
write(i,' ');
i:=i+2;
end;
```

   1) логічний вираз має бути таким: *i<20*;
   2) зайве службове слово *end*;
   3) оператор *i:=i+2;* повинен передувати оператору *write(i,' ')* ;
   4) після слова *do* має бути слово *begin;*
   5) немає помилки;
137. Вкажіть помилку, якщо така має місце, у фрагменті програми, за яким на екран виводяться в стовпчик таблиця квадратів для чисел з проміжку [1;20]

```
i:=1;
while i<=20 do
writeln(i,'- ',i*i);
i:=i+1;
```

   1) оператор *i:=i+1;* повинен передувати оператору *writeln(i,'- ',i*i)* ;
   2) початкове значення змінної *i* має бути 0;
   3) після слова *do* має бути слово *begin;* а після оператора *i:=i+1;* має бути службове слово *end*;
   4) немає помилки;
138. Вкажіть помилку, якщо така має місце, у фрагменті програми, за яким на екран виводяться в рядок непарні числа з проміжку [1;19]

```
i:=1;
while i<=19 do
begin write(i,' ');
```

i:=i+2;
end;

2. логічний вираз має бути таким: *i<19*;
3. початкове значення змінної *i* має бути 0;
4. після слова *do* пропущено знак «крапка з комою»;
5. немає помилки;

139. Вкажіть результат виконання операторів

S:=0; i:=1;
while i<11 do
begin S:=S+I; i:=i+1; end;

2. обчислено суму перших 10 натуральних чисел;
3. обчислено суму перших 11 натуральних чисел;
4. обчислено суму перших 9 натуральних чисел;
5. не можна визначити;

140. Оператор *Repeat* використовується для опису

1) будь-яких циклічних процесів;
2) циклічних процесів, у яких відоме число повторень;
3) циклічних процесів, які відбуваються при виконанні певної умови;
4) циклічних процесів, які відбуваються при невиконанні певної умови;

141. При виконанні оператора repeat оператори тіла циклу

1) завжди виконуються принаймні один раз;
2) виконаються заздалегідь відому кількість разів;
3) можуть не виконатись жодного разу;

142. Вкажіть помилку, якщо така має місце, у фрагменті програми, за яким на екрана виводяться в рядок парні числа з проміку [0; 20]

i:=0;
repeat begin
write(i,' '); i:=i+2;
until i>20;

2. логічний вираз має бути таким *i<20*;
3. зайве ключове слове *begin*;
4. оператор *i:=i+2;* повинен передувати оператору *write(i,' ');*
5. піля слова *until* має бути слово *end*;
6. немає помилки.

143. Вкажіть результат виконання операторів

i:=1; s:=0;
repeat
s:=s+i;
i:=i+1;
until i>10;

2. обчислено суму перших 10 натуральних чисел;
3. обчислено суму перших 11 натуральних чисел;
4. обчислено суму перших 12 натуральних чисел;
5. не можна визначити;

144. Вкажіть результат виконання операторів

```
i:=1;
repeat
  write(i,' ');
   i:=i+2;
until i>21;
```

2. на екрані виведено непарні числа з проміжку від 1 до 19;
3. на екрані виведено перші 20 непарних чисел;
4. на екрані виведено парні числа з проміжку від 2 до 20;
5. на екрані виведено непарні числа з проміжку від 1 до 21;
6. на екрані виведено непарні числа з проміжку від 1 до 23;
7. не можна визначити;

145. Оператор *for* використовується для опису
   1) будь-яких циклічних процесів;
   2) циклічних процесів, у яких відоме число повторень;
   3) циклічних процесів, які відбуваються при виконанні певної умови;
   4) циклічних процесів, які відбуваються при невиконанні певної умови;

146. Значення лічильника у операторі *for* може змінюватись
2. тільки на 1;
3. тільки на -1;
4. як завгодно;
5. якщо форма to, то на 1; якщо форма downto, то на -1;

147. Вкажіть кількість повторень у операторі *for i:=5 to 16 do …*
2. дорівнює ___;
3. не визанчене;

148. Вкажіть кількість повторень у операторі *for i:=15 to 25 do …*
2. дорівнює ___;
3. не визанчене;

149. Вкажіть правильну послідовність виконання операторів у мові програмування Pascal, щоб в результаті змінна s набула значення 23:

А) begin;	1) В
Б) s:=s+i ;	2) Е
В) s:=3; i:=1;	3) А
Г) i:=i+1;	4) Г
Д) end;	5) Б
Е) while i<6 do.	6) Д

150. Вкажіть правильну послідовність виконання операторів у мові програмування Pascal, щоб в результаті змінна r набула значення 3:

А) r:=r+i;	1) Г
Б) until i>=3;	2) Д
В) i:=i+1;	3) В
Г) r:=1; i:=1;	4) А
Д) repeat	5) Б

151. Масив – це …:

1) це сукупність однотипних даних, які зберігаються в послідовних комірках пам’яті;
2) табличні дані;
3) набір даних, які мають порядковий тип;
4) немає вірної відповіді;

152. Індексом елемента називається:
1) номер елемента в масиві;
2) значення елемента масива;
3) останній елемент в масиві;
4) перший елемент в масиві;

153. Масив даних має:
1) Лише спільне ім’я;
2) Лише один тип;
3) Спільне ім’я та один тип;
4) Немає вірної відповіді;

154. В записі D[4]:=3.4 D означає:
1) ім’я комірки;
2) ім’я масиву;
3) позначення типу;
4) Немає вірної відповіді;

155. Який ряд даних являється масивом?
1) a, 4, б, 5, c, 6;
2) 2.3, 5.7, 10.89;
3) 3, 6, 3, 2, massiv;
4) a, b, c, d, e, f;

156. Виберіть правильний опис масиву
1) D: array[1..5] of real;
2) Array D: [1..5] of real;
3) D[1..5]:array of integer;
4) a : array [1..3] of char;

157. Для заповнення масиву випадковими числами, ми повинні підключити датчик випадкових чисел, використовуючи команду
1) clrscr;
2) D [i]:=?;
3) randomize;
4) readkey;

158. Що виконує наступний фрагмент програми: **For i:=1 to N do write (a[i], ‘ ’);**
1) виводить на екран усі значення елементів масиву;
2) виконує ввід даних з клавіатури;
3) виконує перевірку значень елементів масиву;
4) усі відповіді вірні;

159. Для підрахунку суми елементів масиву в циклі ми виконуємо команду?
1) Sum:=sum+a(i);

2) Sum:=a[i]+a[i+1];
3) Sum:=a[i]+sum;
4) Sum:=a(i)+sum;

160. Що виконує наступний фрагмент програми? **Min:=a[1]; For i:=1 to N do if a[i]< min then min:=a[i]; Writeln(min);**
1) знаходить мінімальний індекс масиву;
2) знаходить мінімальний елемент масиву;
3) робить масив мінімальним;
4) сортує масив методом мінімальних елементів;

161. Для підрахунку кількості додатніх елементів масиву потрібно використовувати умову:
1) If a[i]>0 then k:k+1;
2) If a[i]>=0 then k:=k+a[i];
3) If a[i]>=0 then k:=k+1;
4) If a[i]>0 then k:=k+1;

162. Для підрахунку кількості невід'ємних елементів масиву потрібно використовувати умову:
1) If a[i]>0 then k:k+1;
2) If a[i]>=0 then k:=k+a[i];
3) If a[i]>=0 then k:=k+1;
4) If a[i]>0 then k:=k+1;

163. Для підрахунку кількості недодатніх елементів масиву потрібно використовувати умову:
1) If a[i]<0 then k:k+1;
2) If a[i]<=0 then k:=k+a[i];
3) If a[i]<=0 then k:=k+1;
4) If a[i]<0 then k:=k+1;

164. Для підрахунку кількості від'ємних елементів масиву потрібно використовувати умову:
1) If a[i]<0 then k:k+1;
2) If a[i]<=0 then k:=k+a[i];
3) If a[i]<=0 then k:=k+1;
4) If a[i]<0 then k:=k+1;

165. Для підрахунку суми недодатніх елементів масиву потрібно використовувати умову:
1) If a[i]<0 then k:k+1;
2) If a[i]<=0 then k:=k+a[i];
3) If a[i]<=0 then k:=k+1;
4) If a[i]<0 then k:=k+a[i];

166. Для підрахунку суми від'ємних елементів масиву потрібно використовувати умову:
1) If a[i]<0 then k:k+1;
2) If a[i]<=0 then k:=k+a[i];
3) If a[i]<=0 then k:=k+1;
4) If a[i]<0 then k:=k+a[i];

167. Дано масив **a[1]=12, a[2]=3, a[3]=-5, a[4]=-6, a[5]=4, a[6]=9, a[7]=0, a[8]=8**. Визначте значення суми після виконання наступного фрагменту програми **sum:=0; for i:=1 to 8 do sum:=sum+a[i]; writeln (sum);**:
1) 25; 2) 36; 3) -11; 4) 0;

168. Дано масив **a[1]=12, a[2]=3, a[3]=-5, a[4]=-6, a[5]=4, a[6]=9, a[7]=0, a[8]=8**. Визначте значення суми після виконання наступного фрагменту програми **sum:=0; for i:=1 to 7 do if a[i]>5 sum:=sum+a[i]; writeln (sum);**:
1) 25; 2) 29; 3) 21; 4) 28;

169. Дано масив **a[1]=12, a[2]=3, a[3]=-5, a[4]=-6, a[5]=4, a[6]=9, a[7]=0, a[8]=8**. Визначте значення змінної **kol** після виконання наступного фрагменту програми **kol:=0; for i:=1 to 8 do if a[i]>-2 then inc(kol); writeln (kol);**:
1) 2;
2) 6;
3) 5;
4) 4;

170. Дано масив **a[1]=12, a[2]=3, a[3]=-5, a[4]=-6, a[5]=4, a[6]=9, a[7]=0, a[8]=8**. Визначте значення змінної **max** після виконання наступного фрагменту програми **max:=a[1]; for i:=1 to 8 do if a[i]>max then max:=a[i]; writeln (max);**:
1) 12;
2) -6;
3) 9;
4) 8;

171. Дано масив **a[1]=12, a[2]=3, a[3]=-5, a[4]=-6, a[5]=4, a[6]=9, a[7]=0, a[8]=8**. Визначте значення змінної **min** після виконання наступного фрагменту програми **min:=a[1]; for i:=1 to 8 do if a[i]<min then min:=a[i]; writeln (min);**:
1. 12;
2. -6;
3. 9;
4. 8;

172. Дано масив **a[1]=12, a[2]=3, a[3]=-5, a[4]=-6, a[5]=4, a[6]=9, a[7]=0, a[8]=8**. Визначте значення змінної **max** після виконання наступного фрагменту програми **max:=a[1]; for i:=1 to 8 do if a[i]>max then max:=a[1]; writeln (max);**:
1) 12;
2) -6;
3) 9;
4) 8;

173. Дано масив **a[1]=12, a[2]=3, a[3]=-5, a[4]=-6, a[5]=4, a[6]=9, a[7]=0, a[8]=8**. Визначте значення змінної **min** після виконання наступного фрагменту програми **min:=a[1]; for i:=1 to 8 do if a[i]<min then min:=a[1]; writeln (min);**:

2. 12;
3. -6;
4. 9;
5. 8;

174. Вкажіть результат виконання фрагменту програми
for i:=1 to n do readln(a[i]);:
   1) на екрані виведено значення елементів масиву а;
   2) на екрані виведено номери елементів масиву а;
   3) введено значення елементів масиву а;
   4) введено номери елементів маиву а;
   5) не можна визначити;

175. Вкажіть результат виконання фрагменту програми
for i:=1 to 5 do write(a[i],' ');:
   1) на екрані виведено значення елементів масиву а;
   2) на екрані виведено номери елементів масиву а;
   3) введено значення елементів масиву а;
   4) введено номери елементів маиву а;
   5) не можна визначити;

176. Вкажіть результат виконання фрагменту програми
for i:=1 to 20 do if a[i]>0 then write(a[i],' ');:
   1) на екрані виведено значення елементів масиву а;
   2) на екрані виведено від'ємні елементи масиву а;
   3) на екрані виведено додатні елементи масиву а;
   4) на екрані виведено номери елементи масиву а;
   5) не можна визначити;

177. Вкажіть результат виконання фрагменту програми
K:=1; for i:=2 to 20 do if a[i]>a[k] then k:=i;:
2. визначено найбільший елемент масиву;
3. визначено номер найбільшого елементу масиву;
4. визначено найменший елемент масиву;
5. визначено номер найменшого елементу масиву;
6. не можна визначити;

178. Вкажіть результат виконання фрагменту програми
K:=a[1]; for i:=2 to 20 do if a[i]>k then k:=a[i];:
2. визначено найбільший елемент масиву;
3. визначено номер найбільшого елементу масиву;
4. визначено найменший елемент масиву;
5. визначено номер найменшого елементу масиву;
6. не можна визначити;

179. Вкажіть результат виконання фрагменту програми
K:=a[i]; a[i]:=a[i+1]; a[i+1]:=k;:
2. елементам *a[i] і a[i+1]* надано значення *k*;
3. елементу *a[i]* надано значення *k*;
4. елементу *a[i+1]* надано значення *k*;
5. значення *a[i] і a[i+1]* переставлено місцями;

6. не можна визначити;

180. Вкажіть відповідність між даними, та описаними масивами:

1) array [1..10] of integer;	а) 256 цілих чисел;
2) array [-5..5] of real;	б) 256 дійсних чисел;
3) array [byte] of real;	в) 26 цілих чисел;
4) array [‘a’..’z’] of byte;	г) 21 символ;
5) array [1..255] of integer;	д) 10 цілих чисел;
6) array [10..30] of char;	е) 11 дійсних чисел;

181. Вкажіть відповідність між даними, та описаними масивами:

1) array [0..10] of integer;	а) 16 цілих чисел;
2) array [-5..0] of real;	б) 41 символ;
3) array [10..50] of char;	в) 21 ціле число;
4) array [100..120] of byte;	г) 6 символів;
5) array [0..15] of integer;	д) 11 цілих чисел;
6) array [10..15] of char;	е) 6 дійсних чисел;

182. Вкажіть відповідність між фрагментами програм та результатами їх виконання:

1) for i:=1 to n do write(a[i],’ ‘);	а) введення значень елементів масиву;
2) for i:=1 to n do writeln(a[i]);	б) виведення значень елементів масиву у рядок;
3) for i:=1 to n do begin write(‘a[‘,i,’]=’); readln(a[i]); end;	в) виведення значень елементів масиву у стовпчик;
4) for i:=1 to n do readln(a[i]);	г) введення значень елементів масиву з виведенням коментарів;

183. **String** – це..:
   1) рядковий тип даних;
   2) символьний тип даних;
   3) цілочисельний тип даних;
   4) логічний тип даних;

184. **Char** – це..:
   1) рядковий тип даних;
   2) символьний тип даних;
   3) цілочисельний тип даних;
   4) логічний тип даних;

185. ***Length(…)*** – ..:
   1) виділяє з літерної величини st n символів, починаючи з символу під номером m. Дана функція використовується для копіювання;
   2) визначає довжину літерної величини st. Іншими слова, просто підраховує кількість символів в літерній величині.;
   3) об’єднує дві літерні величини. Дана функція виконує ту ж саму дію, що й операція ‘+’, введена вона для сумісності з більш ранніми версіями мови Паскаль.;
   4) повертає позицію, з якої рядок S1 перший раз зустрічається у рядку S;

186. ***Copy (…)*** – ..:
1) виділяє з літерної величини st n символів, починаючи з символу під номером m. Дана функція використовується для копіювання;
2) визначає довжину літерної величини st. Іншими слова, просто підраховує кількість символів в літерній величині.;
3) об'єднує дві літерні величини. Дана функція виконує ту ж саму дію, що й операція '+', введена вона для сумісності з більш ранніми версіями мови Паскаль.;
4) повертає позицію, з якої рядок S1 перший раз зустрічається у рядку S;

187. ***Concat(…)*** – ..:
1) виділяє з літерної величини st n символів, починаючи з символу під номером m. Дана функція використовується для копіювання;
2) визначає довжину літерної величини st. Іншими слова, просто підраховує кількість символів в літерній величині.;
3) об'єднує дві літерні величини. Дана функція виконує ту ж саму дію, що й операція '+', введена вона для сумісності з більш ранніми версіями мови Паскаль.;
4) повертає позицію, з якої рядок S1 перший раз зустрічається у рядку S;

188. ***Pos (…)*** – ..:
1) виділяє з літерної величини st n символів, починаючи з символу під номером m. Дана функція використовується для копіювання;
2) визначає довжину літерної величини st. Іншими слова, просто підраховує кількість символів в літерній величині.;
3) об'єднує дві літерні величини. Дана функція виконує ту ж саму дію, що й операція '+', введена вона для сумісності з більш ранніми версіями мови Паскаль.;
4) повертає позицію, з якої рядок S1 перший раз зустрічається у рядку S;

189. ***Delete (…)*** – ..:
1) перетворює літерну величину st в число m;
2) перетворює числове значення величини x у рядок st;
3) з рядка st вилучає n символів, починаючи з позиції poz;
4) вставляє рядок s1 у рядок s, починаючи з позиції poz;

190. ***Insert (…)*** – ..:
1) перетворює літерну величину st в число m;
2) перетворює числове значення величини x у рядок st;
3) з рядка st вилучає n символів, починаючи з позиції poz;
4) вставляє рядок s1 у рядок s, починаючи з позиції poz;

191. ***Str (…)*** – ..:
1) перетворює літерну величину st в число m;
2) перетворює числове значення величини x у рядок st;
3) з рядка st вилучає n символів, починаючи з позиції poz;
4) вставляє рядок s1 у рядок s, починаючи з позиції poz;

192. ***Val (…)*** – ..:
   1) перетворює літерну величину st в число m;
   2) перетворює числове значення величини x у рядок st;
   3) з рядка st вилучає n символів, починаючи з позиції poz;
   4) вставляє рядок s1 у рядок s, починаючи з позиції poz;

193. Результатом виконання програми **Var x: string[6]; Begin x:=’мим’+’озадаченный’; Writeln(x) End**. буде слово:
   1) мим озадаченный;
   2) мимозадаченный;
   3) мимоза;
   4) озадаченный мим;

194. Результатом виконання команди **copy(‘програмування’,4,3)** буде слово:
   1) прог;
   2) про;
   3) гра;
   4) огра;

195. В результаті виконання послідовності команд **st:=’абракадабра’; n:=length(st);** величина **N** набуде значення:
   1) 11;
   2) 12;
   3) 10;
   4) ’абракадабра’;

196. В результаті виконання послідовності команд **st1:=’абра’; st1:=’кадабра’; st:=concat(st1,st2);** величина **st** набуде значення:
   1) Абракадабра;
   2) абракадабр;
   3) абра кадабра;
   4) абракадабра;

197. В результаті виконання команди **n:=pos(‘абракадабра’,’ра’);** величина **n** буде рівна
   1) 3;
   2) 10;
   3) 4;
   4) 1;

198. В результаті виконання команди **delete(‘абракадабра’,3,4);** величина **st** набуде значення:
   1. абракадабра;
   2. абадабра;
   3. абдабра;
   4. абрдабра;

199. В результаті виконання команди **insert(‘абракадабра’,’фыоврыфлоа’,3);** величина **st** набуде значення:
   1) фыоврыфлоа;
   2) абракадабра;
   3) фыабракадабраоврыфлоа;
   4) абфыоврыфлоаракадабра;

200. Вкажіть відповідність між підпрограмами опрацювання текстових рядків та їх призначенням:

1) перетворення числа в рядок;	а) concat;
2) поточна довжина рядка;	б) copy;
3) вилучення із рядка групи символів;	в) delete;
4) з'єднання рядків;	г) insert;
5) перетворення рядка в число;	д) length;
6) вставлення в рядок групи символів;	е) pos;
7) визначення входження одного рядка в інший;	ж) str;
8) копіювання певної групи символів рядка;	з) val;

201. Вкажіть результат виконання дій: *s1:='abc'; s2:='klmn'; r1:=s1+s2; r2:=s2+s1*;
   1. r1='abcklmn'; r2='klmnabc';
   2. r2='abcklmn'; r1='klmnabc';

202. Підпрограма – це …:
   1) логічно нескінченна група операторів мови, яку можна викликати для виконання по імені будь-яку завгодно кількість разів з різних місць програми;
   2) логічно закінчена група операторів мови, яку можна викликати для виконання по імені будь-яку завгодно кількість разів з різних місць програми;
   3) логічно закінчена група операторів мови, яку можна викликати для виконання по імені лише один раз з конкретного місця програми (а саме на початку виконання програми);
   4) логічно незакінчена група операторів мови, яку можна викликати для виконання по імені 2 рази з різних місць програми;

Частина ІІ. Візуальне програмування

ЛЕКЦІЙНИЙ КУРС

Технологія візуального програмування.

Мета: ознайомити студентів з поняттям технології візуального програмування.

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до викладання в школі.

Основні поняття: технологія, візуальне програмування, процедурне програмування.

План лекції:

1. Передумови виникнення та розвиток технології візуального об’єктно-орієнтованого програмування.

1.1. Основні поняття. Процедурне програмування.

1.2. Методології, що орієнтовані на дані (об’єкт). Об’єктно-орієнтоване програмування.

2. Основні положення та принципи технології візуального об’єктно-орієнтованого програмування.

2.1. Клас як новий тип даних.

Обрані методи: теоретичні, словесні, наочні, пояснювально-ілюстративний.

Наочність: схематичне зображення організації об’єкту, проблемно-символічні сигнали, що використовуються в ході викладання матеріалу.

Питання по темі для самостійного вивчення:

1. Історія створення та виникнення мови програмування Delphi.
2. Середовище програмування Delphi.

Запитання для самоаналізу та самоперевірки:

1. Що таке процедурне програмування?
2. Що таке візуальне програмування?

Текст лекції.

1. Передумови виникнення та розвиток технології візуального об’єктно-орієнтованого програмування.

1.1. Основні поняття. Процедурне програмування.

Delphi – одна з найбільш потужних візуально об'єктно-орієнтованих систем програмування, що дозволяє на найсучаснішому рівні створювати як окремі прикладні програми, орієнтовані на роботу у Windows, так і розгалужені комплекси, що призначені для роботи в кооперативних мережах та в Інтернет.

Середовище візуального об'єктно-орієнтованого програмування Delphi – це графічна автоматизована оболонка над об'єктно-орінтованою версією мови Pascal – Object Pascal.

Проект – це розроблюваний додаток, що включає сукупність файлів-модулів, файлів-форм та інші файли, які зв'язані власне файлом-проекту.

Об'єкт проекту – це діалогові вікна та елементи управління, що є в діалогових вікнах.

Форма – це діалогову вікно програми, яке відкривається під час роботи додатку на етапі розробки програми, на яке розміщуються об'єкти проекту.

Компонента – клас, ініціалізацію якого можна провести під час розробки форми, який має зручний інтерфейс для зміни своїх властивостей.

Процедурне програмування подає програму у вигляді набору алгоритмів, для оформлення яких можуть застосовуватися іменовані програмні блоки – процедури і функції. В останньому випадку передбачається наявність механізмів передачі параметрів і поверненні результату.

Мета процедурного програмування:

- ефективність програми;
- надійність програми;
- економія часу та витрат;
- читабельність.

Основні принципи процедурного програмування.

1. Принципи абстракції (можливість уявлення загального алгоритму без його деталізації).
2. Принцип формальності (поєднання творчості з інженерним аспектом програмування).
3. Принцип “поділяй та володій” (розподіл завдання на декілька незалежних фрагментів).
4. Принцип ієрархічного впорядкування (ієрархічне структурування взаємозв’язків між модулями програмного комплексу).

1.2. Методології, що орієнтовані на дані (об’єкт). Об’єктно-орієнтоване програмування.

Основні концепції об’єктно-орієнтованого програмування.

- Інкапсуляція
- Наслідування
- Поліморфізм

ІНКАПСУЛЯЦІЯ – об’єднання даних та методів, що їх опрацьовують, в середині одного класу (об’єкту)

Області видимості елементів класу.

Секція Public – обмеження на область видимості складових класу (об’єкту) даної секції відсутнє.

Звертання можливе як безпосередньо за ім’ям (для інших складових того ж класу), так і з посиленням на клас (для складових інших класів).

Секція Published – секція не обмежує області видимості - елементи доступні не тільки на етапі виконання, але й при проектуванні програми (у вікні Інспектора об’єктів).

Секція Private – елементи даної секції доступні тільки всередині даного класу та модулю, де міститься клас. Елементи секції Private не доступні навіть найближчим нащадкам даного класу.

Секція Protected - елементи, що були оголошені в даній секції, доступні для будь-яких об’єктів у середині даного модуля, а також для нащадків за його межами.

Секція Automated – область видимості даної секції не обмежена.

НАСЛІДУВАННЯ - процес спадкування об'єктами даних батьківського класу. Наслідування при створенні нових класів, а також додавання нових елементів (даних та методів) до вже існуючих та оголошених в батьківському класу сприяє створенню так званого ієрархічного дерева класів.

ПОЛІМОРФІЗМ - властивість класів вирішувати схожі за змістом завдання різними методами. При цьому методи класів можуть мати однакові імена, але різну функціональність.

2. Основні положення та принципи технології візуального об'єктно-орієнтованого програмування.

2.1. Клас як новий тип даних.

КЛАС – визначений користувачем тип даних, що використовується для опису об'єктів.

Синтаксис оголошення класу.

Type
<ім'я_класу> = class <ім'я_батьківського класу>
Public
<поля, методи, властивості, події>
Published
<поле, властивості>
Protected
<поле, методи, властивості, події>
Private
<поле, методи, властивості, події>
End.

Поле

ПОЛЕ – дані класу певного типу, що використовуються для зберігання інформації про клас.

Синтаксис оголошення:

<ім'я_поля>:<тип>
Наприклад,
Type TnewClass=class(TObject).
Private
Fcode : Integer;
End;

Методи

МЕТОД – процедура та функція, що використовується для обробки полей.

Синтаксис оголошення:

Procedure <ім'я_процедури>(<перелік параметрів>);
<оголошення змінних, типів, констант>
Begin <оператори процедури>
End.

Наприклад,

Type TNewClass = Class(TObject):
Private

FCode: Integer:
Protected
Procedure SetCode(Value: Integer);
End.
Procedure TnewClass.SetCode (Value: Integer);
Begin
If … Then Fcode:= Value;
End;

Властивість

ВЛАСТИВІСТЬ – спеціальний механізм, що регулює доступ до полей. Кожній властивості відповідає поле, що містить значення властивості та два методи, що забезпечують зчитування/запис диних.

Синтаксис оголошення:

Property <ім'я властивості>: <тип> Read <ім'я поля/ метода зчитування> write <ім'я поля/ метода запису>.

Наприклад,

Type TnewClass= Class (TObject)
Private
Fcode: Integer;
Protected
Procedure SetCode (Value: Integer);
Published
Property Code: Integer Read Fcode Write SetCode;
End;

Подія

ПОДІЯ – властивість процедурного типу, що призначена для забезпечення реакції на певну дію.

Значення властивості вказує на метод, що викликається у момент виникнення подій та називається обробкою подій.

ОБ'ЄКТ – сукупність полей, властивостей та подій, що характеризують певну категорію реального світу.

Наприклад,

Type TnewClass=Class (TObject)
Private
Fcode: Integer;
Protected
Procedure Set Code (Value: Integer);
Published
Property Code: Integer Read Fcode Write
SetCode;
End;

Клас Об'єкт

Var NewClass1: TnewClass;

Система візуального об'єктно-орієнтованого програмування Delphi.

Мета: ознайомитися із системою ВООП Delphi.

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до викладання в школі.

Основні поняття: система програмування.

План лекції:

1. Система ВООП Delphi. Загальна характеристика системи.
2. Огляд функціональних можливостей.
3. Загальна організація проекту в системі Delphi.

3.1. Структура файла проекту, його основні елементи та їх призначення.
3.2. Форма модулю: структура та призначення основних складових
3.3. Склад та структура файлу форми: файли опису та модуля форми
3.4.Файли ресурсів та параметрів проекту
3.5. Резервні файли

Обрані методи: теоретичні, словесні, наочні, пояснювально-ілюстративний.

Наочність: схематичне зображення загальної організації проекту системи Delphi, порівняльна таблиця версій системи Delphi.

Питання по темі для самостійного вивчення:

1. Характеристика проекту.

Запитання для самоаналізу та самоперевірки:

1. Структура файла проекту, його основні елементи та їх призначення.
2. Склад та структура файлу форми: файли опису та модуля форми.
3. Файли ресурсів та параметрів проекту.

Текст лекції.

1. Система візуального об'єктно-орієнтованого програмування Delphi. Загальна характеристика системи.

СИСТЕМА DELPHI – система візуального об'єктно-орієнтованого програмування, що базується на мові Object Pascal та підтримує основні принципи технологій швидкого творення додатків

Візуальне програмування
Інтегроване серед розробки
Компілятор мови Object Pascal
Об'єктно-орієнтоване програмування
Загальна характеристика
Класи та об'єкти
Концепції ООП (наслідування, інкапсуляція, поліморфізм)
Автоматичне генерування програмного коду
Компонентний підхід (бібліотека VCL)
Технологія Two Ways Tools
Інтегроване середовище розробки

2. Функціональні можливості системи Delphi.

1. Створення додатків (програм) для ОС Windows (а з версії 6.0 також і для ОС Linux) з розвинутим графічним інтерфейсом користувача.

2. Можливість створення власних динамічно приєднаних бібліотек (DLL), що місять різноманітні компоненти, форми та функції та можуть бути використані в інших системах програмування.

3. Створення потужних систем для роботи з локальними та віддаленими базами даних будь-яких типів. Ця можливість є характерною особливістю систем Delphi.

4. Формування та друк складні звіти, що містять таблиці, малюнки та графіки.

5. Розробка додатків для мережі Internet.

6. Публікація баз даних в мережі Internet.

7. Створення аплетів Панелі керування ОС Windows для керування системними програмами та апаратним (принтери, мережа) забезпеченням.

8. Керування роботою офісних додатків.

9. Створення довідкової системи (файли *.hlp) та інтегрування їх до власних додатків.

10. Створення професійних дистрибутивних пакетів для додатків, що працюють під керівництвом ОС Windows.

11. Підтримка групової роботи (TeamSourse), що дозволяє від слідкувати та контролювати зміни в мережі.

12. Автоматизація перенесення додатків на національні мовні платформи.

3. Загальна організація проекту в системі Delphi

Склад проекту в системі Delphi

- Файл проекту (*.dpr)
- Файл модулю (*.pas)
- Файл опису форми (*.dfm)
- Файл модулю форми (*.pas, *.dfm)
- Файл параметрів проектів (*.dof)
- Файл ресурсів проекту (*.res)
- Файли резервних копій (*.~dp, *.~pa)

3.1. Структура файла проекту, його основні елементи та їх призначення.

```
program Project1; //Заголовок проекту
uses Forms,
Unit1 in 'Unit1.pas' {Form1};
// Перелік модулів, що використовуються в програмі. Модуль Forms – системний, інші – модулі Ваших власних форм
{$R *.RES} // Директива компілятора для підключення файлу ресурсів, що має однойменну з проектом назву
begin // Оператори
Application.Initialize; // ініціалізації
Application.CreateForm(TForm1, Form1); // створення форми та
Application.Run; // завантаження застосування
end.
```

ВАЖЛИВО: внесення додаткового програмного коду є ігноруванням принципу модульності і вказує на поганий стиль програмування. Всі необхідні настроювання, що повинні виконуватися на початку роботи додатка, слід розміщувати в окремому модулі.

3.2. Файл модулю: структура та призначення основних складових

Модуль – автономна компільована програмна одиниця. Що містить процедури, функції, константи та змінні, загальні для декількох модулів проекту. Вся основна робота програми керується кодом модулів.

Структура модулю

Unit <i'мя_модулю> *<заголовок_модулю>*

Interface *<секція_інтерфейс них_оголошень >*

// В цій частині міститься перелік всіх глобальних об'єктів модулю

Implementation *<секція_реалізації>*

//Секція містить опис підпрограм, що оголошені в секції інтерфейсних оголошень

Initialization *<секція_ініціалізації>*

// Секція ініціалізації містить оператори, що виконуються до передачі керування основній формі

Finalization *<секція закінчення роботи>*

// Секція містить оператори, що виконуються після завершення роботи основної форми

End. *<термінатор>*

//вивільняються ресурси, закриваються файли

За звичай Initialization та Finalization опускаються

3.3. Склад та структура файлу форми: файли опису та модуля форми

Файл опису форми – файл ресурсу Delphi, що містить характеристики форми та її компонентів та визначає її зовнішній вигляд

Файл опису форми:

```
object Form1: TForm1
Left = 192
Top = 107
Width = 696
Height = 480
Caption = 'Form1'
Color = clBtnFace
Font.Charset= DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -11
Font.Name = 'MS Sans Serif'
Font.Style = []
OldCreateOrder = False
PixelsPerInch = 96
TextHeight = 13
object Label1: TLabel
Left = 176
Top = 72
Width = 32
Height = 13
```

Caption = 'Label1'
end
object Button1: TButton
Left = 168
Top = 32
Width = 75
Height = 25
Caption = 'Button1'
TabOrder = 0
end
end

Файл модуля форми:
unit Unit1;
interface
uses Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;
type TForm1 = class(TForm)
Button1: TButton;
Label1: TLabel;
private { Private declarations }
public { Public declarations }
end;
var Form1: TForm1;
implementation
*{$R *.DFM}*
end.

4.4 Файли ресурсів та параметрів проекту

Файли ресурсів (*.res) можуть містити наступні ресурси: піктограми компонентів та додатків; растрові зображення; курсори.

Починаючи з Delphi 5, файли ресурсів стали основою для інтернаціоналізації застосувань.

Для роботи з файлами ресурсів використовується графічний редактор Image Editor.

Файли параметрів проекту (*.dof)

Параметри проекту визначаються за допомогою сторінок Compiler та Linker діалогового вікна Project Options.

4.5 Резервні файли

Резервні файли (*.~dpr, * .~pas, *.~dfm) – файли резервних копій для файлів проекту, форми та модуля використовують у випадку втрати (псування оригіналів) шляхом заміни типу файлу. Наприклад, (*.~dpr Æ *.dpr).

Інтегроване середовище розробки (ІСР) системи Delphi.

Мета: ознайомити студентів з поняттям технології візуального програмування.

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до викладання в школі.

Основні поняття: технологія, візуальне програмування, процедурне програмування, інтегроване середовище розробки.

План лекції:

1. Інтегроване середовище розробки (ІСР) системи Delphi.

1.1. Головне вікно ІСР системи Delphi. Загальний вигляд, склад та призначення основних елементів.

1.2. Вікно Проектувальника форм.

1.3. Вікно Редактора та Дослідника коду.

1.4. Вікно Інспектора об’єктів. Вкладника Властивості (Properties) та Подій (Events).

2. Інструментальні засоби інтегрованого середовища системи Delphi.

2.1. Менеджер проектів (Manager Project).

2.2. Знавець коду (Code Insight).

2.3. Вбудований відлагоджувач додатків.

2.4. Сховище об’єктів (Repository).

2.5. Довідкова система.

Обрані методи: теоретичні, словесні, наочні, пояснювально-ілюстративний.

Наочність: графічне подання основних елементів інтегрованого середовища системи Delphi, таблиці зі складом Палітри компонентів та призначення її сторінок.

Питання по темі для самостійного вивчення:

Призначення сховища об'єктів.

Запитання для самоаналізу та самоперевірки:

1. Вікна Проектувальника форм, Редактора та Дослідника коду.
2. Вікно Інспектора об’єктів. Вкладинка Властивості та Події.
3. Менеджер проектів (Manager Project).
4. Знавець коду (Code Insight).
5. Вбудований відлагоджувач додатків.
6. Сховище об’єктів (Repository).

Текст лекції.

1. Інтегроване середовище розробки (ІСР) системи Delphi.

Інтегроване середовище розробки (ІСР) – середовище, що містить всі необхідні засоби для проектування, завантаження та тестування застосувань.

Склад інтегрованого середовища розробки (ІСР) :

- Головне вікно
- Вікно Проектувальника форм.
- Вікно Редактора та Дослідника коду.
- Вікно Інспектора об’єктів.

• Вікно Менеджера проектів.

1.1 Головне вікно ICP системи Delphi. Загальний вигляд, склад та призначення основних елементів.

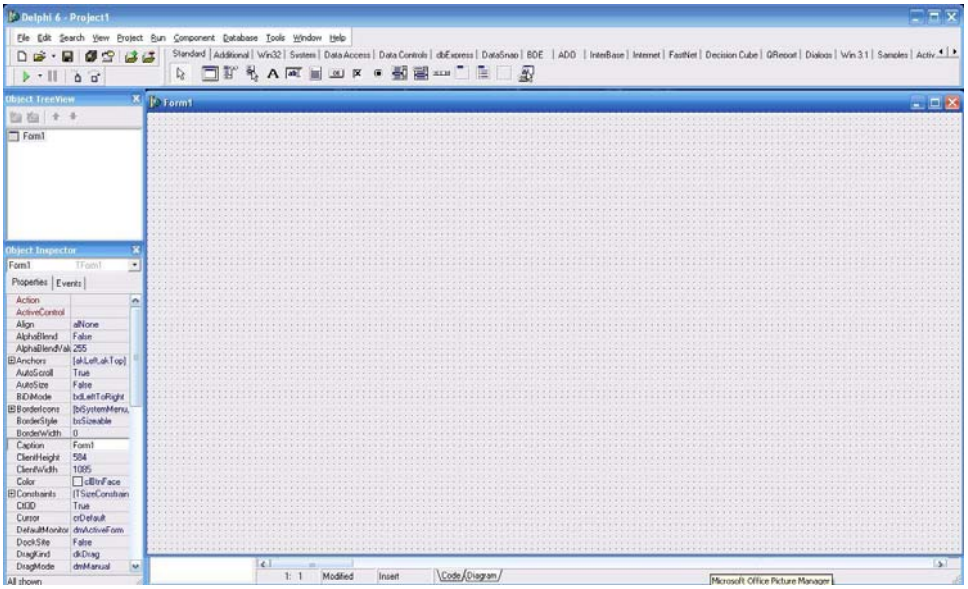


Рис. 1. Загальний вигляд інтегрованого середовища розробки системи Delphi.

Склад головного вікна ICP

• Головне меню системи Delphi.
• Інструментальні панелі.
• Палітра компонентів.

Склад та призначення пунктів Головного меню системи Delphi

Пункт меню "File"

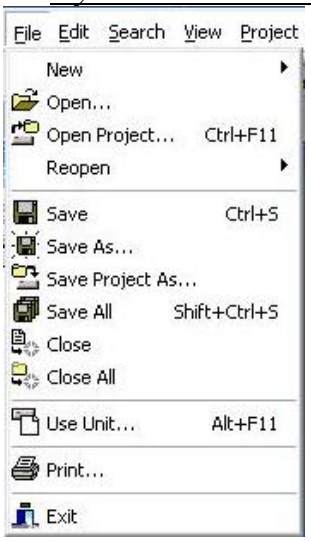


Рис. 2. Загальний вигляд пункту меню "File"

Використовується меню **File**, щоб відкрити, зберегти, закрити, і друкувати нові або існуючі проекти та файли.

New відкриває нові елементи діалогового вікна, яке містить об'єкти, які зберігаються в пам'яті і майстри для створення нових об'єктів.

Open відкриває для завантаження існуючого проекту, форми, одиниці, або текстові файли в редакторі коду.

Open Project відкриває проект діалогового вікна для завантаження існуючого проекту (.BPR або .BPK файли).

Reopen відображає каскадне меню, яке містить список найбільш недавно завершених проектів та модулів.

Save зберігає поточний файл, використовуючи свою нинішню назву.

Save As зберігає поточний файл, використовуючи нове ім'я, в тому числі зміни, внесені в проект файлу.

Save Project As зберігає поточний проект, використовуючи нове ім'я.

Save All зберігає всі відкриті файли, поточний проект і модулі.

Close закриває поточний проект і пов'язані з ним і форми.

Close All закриває всі відкриті файли.

Use Unit додає обрану одиницю для використання пункту активного модуля.

Print відправляє активний файл на принтер.

Exit закриває відкритий проект і виходить з програми.

Пункт меню "Edit"

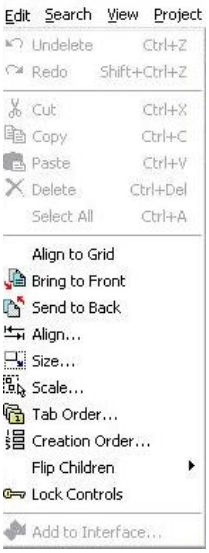


Рис. 3. Загальний вигляд пункту меню "Edit"

Використовується меню **Edit** для вибору команди для роботи з текстом і компонентами під час розробки.

Undo/Undelete скасовує останню дію або останнє видалення.

Redo змінює, відновлює або скасовує.

Cut видаляє вибраний елемент і поміщає його в буфер обміну.

Copy поміщає копію вибраного елемента до буфера обміну, залишаючи оригінал на місці.

Paste копіює вміст буфера обміну у вікні редактора коду або форми.

Delete видаляє вибраний елемент.

Select All вибрає всі компоненти на формі.

Align to Grid вирівнює обрані компоненти до найближчого вузла сітки.

Bring to Front переміщення вибраного компоненту на передній план.

Send to Back переміщення вибраного компоненту на задній план.

Align вирівнювання компонентів.

Size зміна розміру компонентів.

Scale змінює всі компоненти на формі.

Tab Order змінює порядок переходу компонентів на активну форму.

Creation Order змінює порядок, в якому створюються невізуальні компоненти.

Flip Children інвертує розташування елементів управління з права-наліво дзеркально.

Lock Controls забезпечує захист всіх компонентів на формі у своїй поточній позиції.

Add to interface визначає новий метод, подію або властивість для компонента ActiveX.

Пункт меню "Search"

Рис. 4. Загальний вигляд пункту меню "Search"

Використовується **Search** меню, щоб знайти текст, помилки, об'єкти, змінні і символи в редакторі коду.

Find пошук конкретного тексту, і підкреслення при першому згадуванні в редакторі коду.

Find in Files пошук конкретного тексту, відображення кожного входження у вікно в нижній частині Code editor.

Replace пошук конкретного тексту і поєднання його з новим текстом.

Search Again повторює останній пошук.

Incremental Search пошук тексту при введенні.

Go to Line Number перехід до конкретного номера рядка.

Find Error пошук самої останньої помилки виконання.

Browse Symbol пошук зазначеного символу.

Пункт меню "View"

За допомогою меню **View** відкриваються команди для відображення або приховуються різні елементи середовища і відкриваються вікна, які належать до інтегрованого відладчика.

Project Manager відображає керівника проекту.

Translation Manager відображає Translation Manager.

Object Inspector відображає Object Inspector.

Object TreeView відображає об'єкти на формі у вигляді дерева.

To-Do List дозволяє переглядати To-Do список, пов'язаний з поточним проектом.

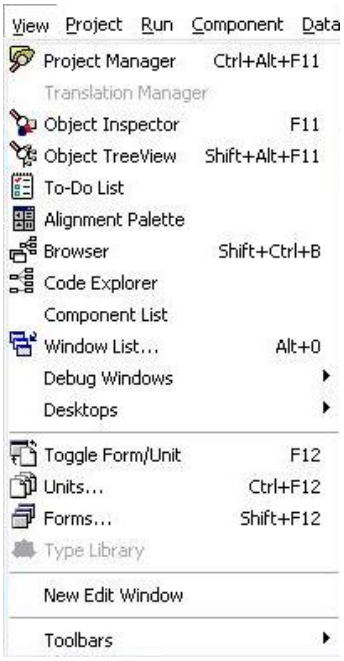


Рис. 5. Загальний вигляд пункту меню "View"

Alignment Palette відображає Alignment Palette.

Browser відображає Project Browser.

Code Explorer відображає Code Explorer.

Component List відображає діалогове вікно компоненти.

Window List відображає список відкритих вікон.

Debug Windows відображає підменю Debugger.

Desktops дозволяє відображати, зберігати або видаляти різні уявлення робочого столу.

Toggle Form/Unit перемикання між формою і її блоком вікна.

Units відображення діалогового вікна View Unit.

Forms відображає діалогове вікно View Form .

Type Library відображає тип бібліотеки редактора вікна .

New Edit Window відкривається новий редактор .

Toolbars приховує або показує панель інструментів або палітру компонентів.

Пункт меню "Project"

Використовується меню **Project** для компіляції або складання додатків.

Add to Project додає файл у проект.

Remove from Project видаляє файл з проекту.

Import Type Library імпорт бібліотека проекта.

Add to Repository додає проект Object Repository.

View Source показує файл проекту в редакторі коду.

Languages дозволяє додавати, видаляти і оновлювати ресурси DLL, або вибирати мову для тестування.

Add New Project Open the New Items dialog box за допомогою цього вікна ви можете або створити новий об'єкт або почати з будь-якого існуючого об'єкту, що зберігається у пам'яті.

Add Existing Project використання відкритого діалогового вікна проекту для додавання існуючого проекту.

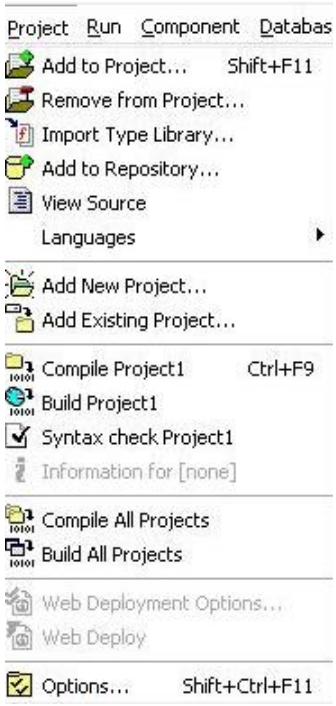


Рис. 6. Загальний вигляд пункту меню "Project"

Compile project компілює тільки ті файли, в поточному проекті, які були змінені з моменту останнього побудування.

Build project компілює все в проекті, незалежно від того, чи щось змінилося.

Syntax Check project компілює проект, але не пов'язує його.

Information for project відображення всієї побудованої інформації та створення статусу для Вашого проекту.

Compile All Projects компілює будь-який вихідний код, що змінився з моменту останньої компіляції у всіх проектах у проектної групи.

Build All Projects компілює все підряд, у проектної групи, незалежно від того, чи щось змінилося.

Web Deployment Options робить необхідні налаштування для розгортання готових ActiveX або ActiveForm.

Web Deploy після налаштування параметрів розгортає веб-додатки та складає проект, розгортає готові ActiveX або ActiveForm.

Options параметри проекту, де задаються параметри для компіляції, посилання, за замовчуванням форми, інформація про версію, і так далі.

Пункт меню "Run"

Run - команда меню яка допоможе вам налагоджувати результат програми.

Run компілює і виконує програми.

Attach to Process надає список запущених процесів, які ви можете налагоджувати.

Parameters вказує параметри запуску для вашої програми, які беруть виконуваний файл для DLL, або комп'ютери для віддаленого налагодження.

Register ActiveX Server сервер додає запис реєстру Windows для доступного управління **ActiveX**, коли поточний проект є проектом ActiveX.

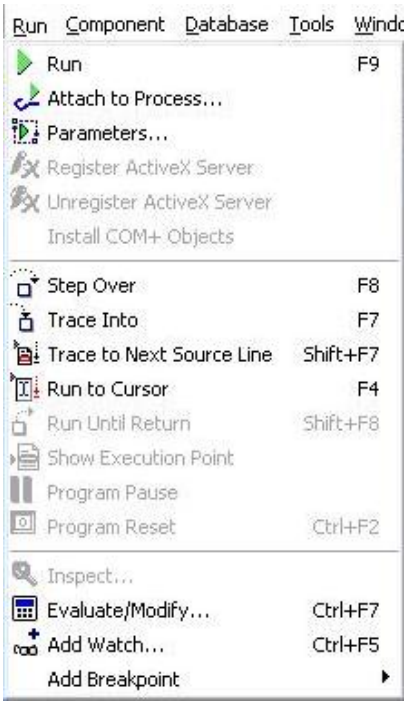


Рис. 7. Загальний вигляд пункту меню "Run"

Unregister ActiveX Server сервер видаляє проект з реєстру Windows.

Install COM+ Objects встановлює об'єкти в поточний проект в COM + додатки. З'являється, коли операційна система підтримує COM +.

Step Over виконує програму на один рядок за один раз, переступаючи через процедуру при виконанні їх як єдиного цілого.

Trace Into виконує програму на один рядок у той же час, відстежує процедуру після виконання кожного рядка.

Trace To Next Source Line виконує програму, зупиняючись на наступному виконуваному лінійному джерелі в коді.

Run To Cursor виконує завантажену програму до положення курсору в редакторі коду.

Run Until Return запускається процес, поки виконання повертається з поточної функції.

Show Execution Point позиція курсору на точку виконання в вікні редагування.

Program Pause тимчасово призупиняє виконання запущеної програми.

Program Reset завершує поточну програму і звільняє її з пам'яті.

Inspect відкриває вікно інспектора.

Evaluate/Modify Displays the Evaluate/Modify dialog box - діалогове вікно, де ви можете оцінити або змінити значення існуючого виразу.

Add Watch Opens the Watch Properties dialog box - діалогове вікно де ви можете створювати і змінювати годинник.

Add Breakpoint відкриває Edit Breakpoint діалогове вікно, де ви можете створювати і змінювати точки зупинки.

Пункт меню "Component"

Component меню відображаються наступні команди:

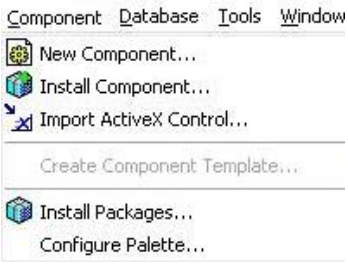


Рис. 8. Загальний вигляд пункту меню "Component"

New Component відкриває компонент експертів.

Install Component установка компонентів з існуючих або нових пакетів.

Import ActiveX Control додає бібліотеки елементів керування ActiveX у проект.

Create Component Template налаштовує компоненти і зберігає їх як шаблон з новим ім'ям, палітра сторінки і значок.

Install Packages вказує пакети, необхідні для вашого проекту.

Configure Palette відкриває вікно палітри діалог.

Пункт меню "Database"

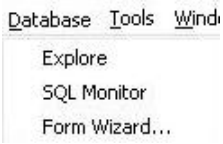


Рис. 9. Загальний вигляд пункту меню "DataBase"

Database команда меню, яка дозволяє створювати, модифікувати, відстежувати і переглядати бази даних.

Explore – дослідити.

SQL Monitor - SQL монітор.

Form Wizard – майстер форм.

Пункт меню "Tools"

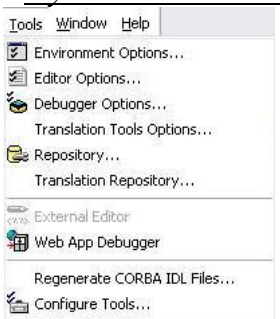


Рис. 10. Загальний вигляд пункту меню "Tools"

Environment Options відображає діалогове вікно, де ви можете вказати конфігурацію переваг, бібліотеку шляхів, Code Explorer варіанти, змінні оточення, і налаштовує появу палітри компонентів.

Editor Options відображає діалогове вікно Editor Options, де ви можете вказати код переваг редактора конфігурації.

Debugger Options діалогове вікно, де ви можете задати параметри відладчика.

Translation Tools Options відображає переклад діалогового вікна Tools Options.

Translation Repository відображає вікно Translation Repository.

External Editor запускає сторонні веб-редактори сторінок.

Web App Debugger відображає веб-відладчик додатка, в якому ви можете тестувати веб-сервера додатків.

Regenerate CORBA IDL Files відображає, регенерує CORBA IDL файли діалогового вікна.

Configure Tools відображає діалогове вікно параметри інструментів. Використовується це діалогове вікно для додавання команд, видалення команд, або команд редагування в меню Tools.

Пункт меню "Help"

Рис. 11. Загальний вигляд пункту меню "Help"

Delphi Help відкриття довідки Delphi. У діалоговому вікні перейдіть на вкладку (зміст, індекс або знайти), те що ви останній раз переглядали.

Delphi Tools відкриває діалогове вікно довідки для Delphi. Відкриває діалогове вікно на вкладку (зміст, індекс або знайти), те що ви останній раз переглядали.

Windows API/SDK Help відкриває діалогове вікно на вкладку (зміст, індекс або Знайти),), те що ви останній раз переглядали.

Borland Home Page відкриває веб-браузер і сторінки його сайта Всесвітній Borland's Wide Web.

Borland Community Page відкриває сторінку веб-браузера і вказує його на веб-сайті компанії Borland для розробників, включаючи новини, статті, особливості історії, і приклади коду.

Delphi Home Page відкриває веб-браузер на сторінку веб Delphi, де Ви можете знайти інформацію про Delphi, включаючи новини та оголошення, опис характеристик, і завантажувані програми.

Delphi Developer Support пряме посилання на сторінку розробника на сайт Всесвітньої Borland's Wide Web. Надає інформацію та технічну підтримку, останнього завантаження для Delphi, та інші послуги.

Delphi Direct пряме посилання на сторінку веб Delphi, де можна знайти більш детальну інформацію про завантаження програмного забезпечення, яка буде автоматично інформувати Вас про Delphi і Borland новини та оголошення.

Customize Launches OpenHelp утиліт, який дозволяє налаштувати, які довідки ви хочете зробити доступними у змісті.

Панелі інструментів

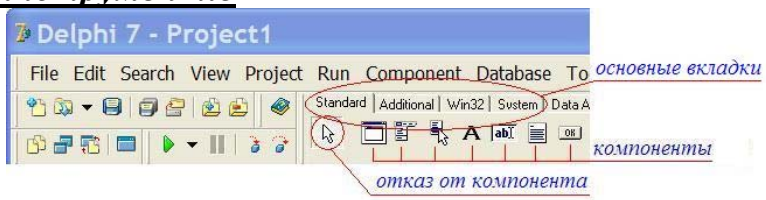


Рис. 12. Загальний вигляд панелі інструментів

Палітра компонентів являє собою каталог візуальних та прикладних об‘єктів, які можна включати до форм та програм. У Delphi компоненти об‘єднані по групах: стандартна (Standard), додаткова (Additional), група Windows 95 (Win95), група доступу до даних(Data Access), група керування даними (Data Controls), група Windows 3.1. (Win 3.1), діалогова група (Dialogs), системна група (System), група звітів (QReport), OCX група та група прикладів (Samples). Кожна з груп представлена на окремійй сторінці палітри компонентів. Щоб перенести необхідні компоненти в форму, ви просто переносите їх з палітри компонентів. Компонент, що був раз перенесений, стає справжнім об‘єктом, готовим до виконання будь-якої заданої інструкції. Якщо ви розташуєте курсор миші над будь-якої кнопкою палітри компонентів, під курсором миші з‘являється підказка, що пояснює, для чого потрібна вибрана кнопка. Нижче показано, для чого потрібні сторінки палітри компонентів:

• *Стандартна сторінка (Standart).* Стандартна сторінка містить найбільш часто вживані компоненти, що фігурують у всіх програмах Windows. Ці компоненнти мають однозначний зв‘язок зі стандартними об‘єктами Windows.

• *Додаткова сторінка (Additional).* Ця сторнка містить дещо більш специфічний набір компонентів, який ви, працюючи з базовими програмами Windows, могли й не зустріти. Компоненти цієї сторінки дуже корисні. Наприклад, такий компонент як MaskEdit дає вам кращій спосіб керування, ніж стандартний Edit. Крім того, тут містяться різноманітні, орієнтовані на графіку візуальні компоненти, такі як фігура (Shape) та образ (Image).

• *Win 95.* На цій сторінці розташовані компоненти, що існують тільки в Windows 95 і яких не було в Windows 3.1.

• *Доступ до даних (Data Access).* Ця сторінка містить компоненти, що дозволяють вам використовувати таблиці та запити.

• *Керування даними* (*Data Controls*). На цій сторінці розміщена та частина інтерфейсу користувача, що пов‘язана з даними. Тут є компоненти, що дозволяють вам представляти дані користувачу будь-яким способом, прийнятим у Windows.

• *Win 3.1.* Тут знаходяться застарілі компоненти Windows 3.1, які рідко використовуються, тому що в Windows 95 є більш потужні їхні відповідники.

• *Діалоги* (*Dialogs*). На цій панелі ви знайдите діалогові панелі для виконання таких задач загального характеру, як відкриття файлу, установка принтера, пошук тексту, тощо.

• *Системна* (*System*). Системна сторінка містить візуальні та

невізуальні компоненти. Тут містяться компоненти для таймера, дисковода, компоненти доступу до файла, а також компоненти динамічного обміну даними — DDE (Dynamic Data Exchange) — та зв‘язку-вбудови об‘єктів — OLE (Object Linking and Embedding).

• *Швидкі звіти* (*QReport*). Ця сторінка дозволяє швидко будувати різноманітні звіти по базам даних.

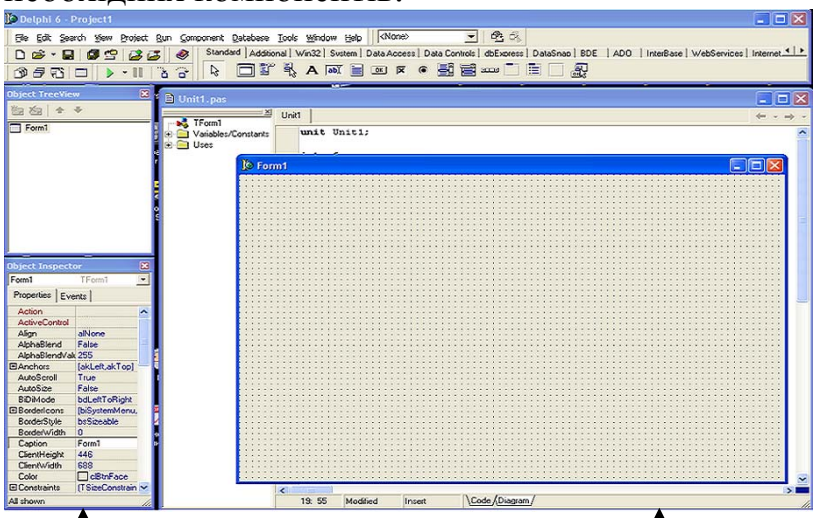
• *OCX*. Ця панель містить такі компоненти як графік, графічний сервер, перевірка орфографії, тощо.

• *Приклади* (*Samples*). Ця сторінка містить компоненти, що демонструють, як додавати власні компоненти до палітри.

1.2. Вікно Проектувальника форм.

Вікно Проектувальника форм подане у вигляді проекту Windows – вікно, що має стандартні інтерфейсні елементи – кнопку виклику системного меню, кнопки керування вікном, рядок заголовку.

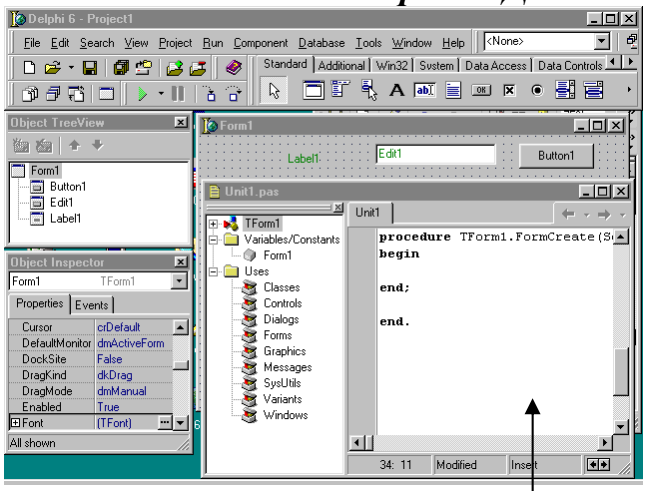
Візуальне проектування полягає у перенесенні на вікно форм необхідних компонентів.



Інспектор об'єктів Візуальний проектувальник робочих вікон (Форма)

Рис. 13. Вікно Проектувальника форм.

1.3 Вікно Редактора та Дослідника коду.



Вікно редактора програми

Рис. 14. Вікно Редактора та Дослідника коду.

Вікно Редактору коду призначено для створення та редагування програмного коду програми, що забезпечує її функціональність.

Дослідник коду призначений для виведення дерева всіх типів, класів, властивостей, методів, глобальних змінних та процедур, що знаходяться в модулі, який активний у вікні Редактору коду.

1.4 Вікно Інспектора об'єктів. Вкладника Властивості (Properties) та подій (Events).

Інспектор об'єктів – спеціальний механізм, що дозволяє встановлювати та змінювати параметри компонентів (сторінка Властивості) та визначати їх реакцію на певну подію (сторінка Події).

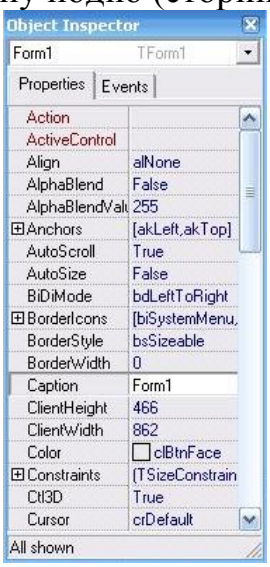


Рис. 15. Вікно Інспектора об'єктів.

Інспектор об'єктів це зв'язок між зовнішнім виглядом вашої форми і кодом, що дозволяє запускати вашу програму.

Інспектор об'єктів дозволяє:

Декорації-часових властивостей для компонентів, які ви розмістили на формі.

Створювати і допомагати зорієнтуватися в обробнику подій.

Фільтрувати видимі властивості та події.

Вибирати об'єкти, у верхній частині інспектора об'єктів є список, що розкривається, містить всі компоненти на активній формі, і показує тип кожного об'єкта, або класу, обраного компонента. Це дозволяє швидко відобразити властивості і події для різних компонентів на даній формі.

Ви можете змінити розмір стовпців Інспектор об'єктів, перетягуючи розділову лінію в нове положення.

Інспектор об'єктів складається з двох сторінок: властивості та події сторінки.

Вкладки Інспектора об'єктів забезпечують швидке перемикання між сторінками влативості та події Інспектор об'єктів. Для зміни сторінки, натисніть кнопку "Властивості" або вкладку Події.

Декілька слів про *основні властивості Object Inspector*:

Align – вирівнювання поля відносно об'єкта, що його містить (форми).

AutoScroll – наявність у формі смуг прокручування.

AutoSize – приведення розміру об'єкта до реальних розмірів зображення.

BorderStyle – можливість змінювати розміри вікна.

Caption – текст заголовка.

Color – колір фону форми.

Enable – доступність для дій об'єктів у формі під час виконання.

Font – шрифт.

Height – висота форми.

Icon – задає піктограму, яка буде в заголошу форми під час виконання програми.

Left – відстань від лівої межі форми до лівої межі екрану.

Name – ім'я форми.

Position – розміщення і розміри вікна у момент запуску програми.

Top – відстань від верхньої межі форми до верхньої межі екрану.

Visible – видимість об’єкта.

Widht – ширина форми.

WindowState – стан вікна у момент запуску програми.

Для встановлення значень властивостей компонентів в Інспекторі об’єктів використовують спеціальні редактори, що завантажуються автоматично.

Простий (Caption, Hint, Left) дані вносяться та редагуються як у звичайному рядку символів.

Перелічувальний (FormStyle, Visible) значення властивостей вибирається із переліку, що розкривається

Множина(BordersIcons, Anchors) значення властивостей є комбінацією значень із запропонованої множини. Активізація переліку допустимих значень властивості відбувається після натиснення “+”.

Об’єкт (Font, Lines, Items) властивість є об’єктом, що містить перелік інших властивостей (підвластивостей)

2. Інструментальні засоби інтегрованого середовища системи Delphi.

2.1 Менеджер проектів (Manager Project).

Менеджер проектів – спеціальне вікно, що дозволяє керувати як окремими модулями проекту, додавати їх чи вилучати, так групою проектів, додаючи до проекту нові, або проекти, що вже існують, та вилучати їх з групи.

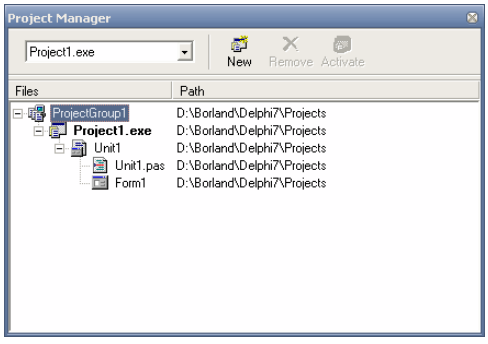


Рис. 16. Вікно Менеджера проектів.

2.2 Знавець коду (Code Insight)

Окрім стандартних можливостей, що притаманні редакторам програмного коду, Редактор Коду в Delphi має унікальний інструментарій, що зветься Знавець коду (Code Insight)

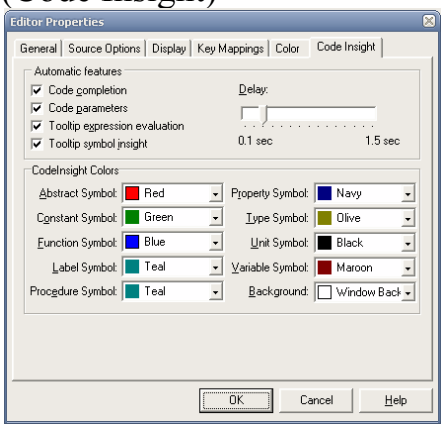


Рис. 17. Вікно Знавця коду.

2.3 Вбудований відлагоджувач застосувань.

Класифікація помилок

1)Синтаксичні 2)Логічні 3)Динамічні

Засоби Вбудованого відлагоджувача застосувань.

• Порядкове (покрокове) виконання програми
• Крапки переривання
• Обчислення виразу та зміна значень
• Вікно спостереження

Покрокове виконання програми

Основні команди та перелік функціональних клавіш покрокового виконання програми

Step Over (покрокове виконання без входження в ...) F8 – Порядкове виконання рядків програми з врахуванням виклику функцій, процедур за один рядок, тобто входження в процедуру/функцію не відбувалося

Trase Into (покрокове виконання із входженням в ...) F7 – Порядкове виконання із входженням у процедури та функції

Trase to Next Sourse (виконання до наступного рядку) Shift + F7 Перехід до наступного рядку програмного коду

Run to Cursor (виконати до курсора) F4 – Виконання програми до рядку, на якому встановлений курсор в Редактору коду

Show Execution Point (показати точку виконання) - Переміщення курсору на наступний оператор

Крапка переривання (**breakpoint**) View\DebugWindow\ breakpoint Крапки переривання – спеціальний механізм, що визначає оператор в програмі, перед виконанням такого програма припинить свою роботу та передасть керування середовищу системи Delphi.

Вікно спостереження **Watches** (View\DebugWindow\Watches) Вікно, що призначене для відслідковування значень змінних та виразів.

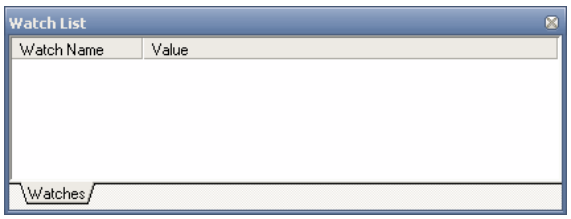


Рис. 18. Вікно спостереження.

2.4. Сховище об'єктів (Депозитарій).

File\ New\Other

Депозитарій – сховище об'єктів, що використовуються в якості шаблонів для розробки застосувань.

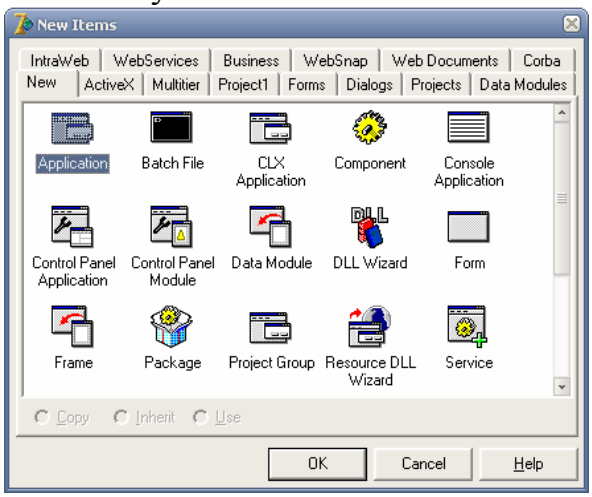


Рис. 19. Вікно Депозитарію ICP системи Delphi.

2.5 Довідкова система.

Довідкова система – спеціальний інструмент, що виконує конфігурування довідникової системи, додаючи/вилучаючи необхідні файли допомоги.

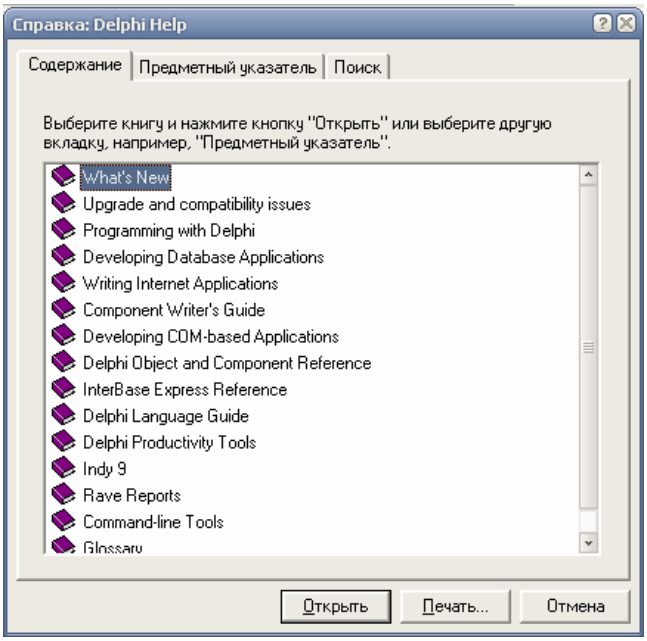


Рис. 20. Вікно довідкової системи ICP системи Delphi.

Компоненти. Загальний огляд основних властивостей, методів та подій

Мета: ознайомити студентів з поняттями компоненти, властивості, методи та події, а також з основними властивостями, методами та подіями.

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до викладання в школі.

Основні поняття: компоненти, властивості, методи, події.

План лекції:

1. Компоненти системи Delphi як особливий тип об'єктів. Класифікація компонентів, загальні принципи роботи з ними.
2. Властивості компонентів загального призначення.
3. Методи компонентів загального призначення.
4. Події. Класифікація подій. Обробники подій.
4.1. Події миші.
4.2. Операції Drag&Drop.
4.3. Події клавіатури.
4.4. Програмно-керовані події.
5. Форма як основний компонент системи Delphi.

Обрані методи: лекція-бесіда.

Наочність: схематичне зображення ієрархічної структури об'єктів в Delphi, таблиці з переліком основних властивостей, методів та подій компонентів.

Питання по темі для самостійного вивчення:

3. Характеристика проекту.

Запитання для самоаналізу та самоперевірки:

1. Компоненти системи Delphi як особливий тип об'єктів. Класифікація компонентів, загальні принципи роботи з ними.
2. Властивості та методи компонентів загального призначення.
3. Події. Класифікація подій. Обробники подій.
4. Форма як основний компонент системи Delphi

Текст лекції.

1. Компоненти системи Delphi як особливий тип об'єктів. Класифікація компонентів, загальні принципи роботи з ними.

ОБ'ЄКТИ: візуальні компоненти (Application, Screen); не візуальні (Stream, Printer, Clipboard); не елементи керування; елементи керування; віконні компоненти; не віконні компоненти.

2. Властивості компонентів загального призначення.

Таблиця 4.1.

Властивості	Клас	Призначення
Components	TComponent	Перелік компонентів, що належать певному компонентові.
Component Count	TComponent	Кількість компонентів, що належать певному компонентові.
Component Index	TComponent	Номер компонента в переліку компонентів,

Властивості	Клас	Призначення
		що належать певному компонентові.
Component State	TComponent	Стан поточного компоненту.
Name	TComponent	Ім’я компонента.
Owner	TComponent	Компонент, що володіє певним компонентом.
Tag	TComponent	Цілочислене значення, що використовує програміст.

3. Методи компонентів загального призначення.

Таблиця 4.2.

Метод	Клас-пращур	Призначення
Create	TObject	Створення об’єкту
Destroy	TObject	Знищення об’єкту
Assign	TPersistent	Передавання полів та властивостей іншим об’єктам
Destroy Components	TComponent	Знищення компонентів
Find Component	TComponent	Пошук компонентів в переліку Components

4. Події. Класифікація подій. Обробники подій.

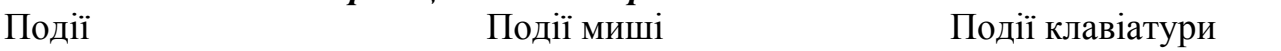


Схема 4.2. Класифікація подій

4.1. Події миші

• *OnClick* (Натиснення)
• *OnMouseDown* (натиснення кнопки миші)
• *OnMouseUp* (відпускання кнопки миші)
• *OnMouseMove* (переміщення миші)

Передумови виклику обробника події *OnClick*:

• Натиснення та відпускання кнопки миші в момент, коли вказівник знаходиться на компонентові.
• Вибір елемента переліку чи поля з переліком за допомогою клавіш керування курсором.
• Натиснення клавіш прогалини в момент активності будь-якої кнопки чи події.
• Натискання клавіші <Enter> в момент, коли активне форма містить попередньо вибрану кнопку за умовчуванням (властивість Default).
• Натиснення клавіші <Esc> в момент, коли активна форма містить кнопку Cancel (властивості Cancel).
• Натиснення клавіш мнемонічного доступу (кнопки швидкого доступу).
• Заміна значення властивості Checked.
• Виклик метода Click кнопки чи пункту меню.

4.2. Операції Drag&Drop

Основні етапи та обробники подій операції Drag&Drop:

• Початок переміщення (OnStartDrop)

• Визначення можливості прийняття елементів, що переміщуються (OnDragOver)

• “Відпускання” елемента (OnDragDrop)

• Кінець операції переміщення

4.3. Події клавіатури.

Таблиця 4.3.

Подія	Пояснення
OnKeyDown	Обробник викликається у момент натискання користувачем будь якої клавіші
OnKeyUp	Обробник, що реагує на відпускання клавіші.
OnKeyPress	Обробник викликається у момент натисканням користувачем клавіші, що відповідає символу з набору ASCII, та повертає його значення

Послідовність подій при натисканні клавіш.

Загальна буква t	Власна буква T	Прогалина	Alt
KeyDown [T]	KeyDown	KeyDown	KeyDown
KeyPress [t]	KeyDown	[Space]	KeyUp
KeyUp [T]	[Shift+T]	KeyPress	
	KeyPress [T]	KeyUp [Space]	
	KeyUp [Shift+T]		
	KeyUp		

Перехвачування подій клавіатури

Властивість компонентів класу TForm – Key Preview – дозволяє формі виконувати “перехвачування” подій клавіатури, а вже потім передати їх активному компонентові.

Приклад використання події клавіатури.

• KeyPress.

Procedure TForm1. FormKeyPress (Sender: TObject; Var Key: Char);

Begin

MessageDlg (‘Була натиснена клавіша’+Key, mtInformation, [mbOk],0);

End;

• KeyDown

Procedure TForm1. FormKeyDown (Sender: TObject; Var Key: Word; Shift: TShiftState);

Begin

If Shift = [ssAlt] and (Key=Vk_F10) then MessageDlg (‘була натиснена комбінація клавіш Alt+F10’, mtInformation, [mbOk],0);

End;

4.5. Програмно-керовані події.

OnEnter *OnExit* *OnChange*

5. Форма як основний компонент.

5.1. Властивості компонентів класу TForm.

Таблиця 4.4.

Властивість	**Пояснення**	**Можливі значення:**
Active	Активність вікна	
Border Icons	Визначення кнопок керування вікна	BiSystemMenu – наявна кнопка виклику системного меню BiMinimize – наявна кнопка мінімізації BiMaximize – наявність кнопки максимізації BiHelp –наявна кнопка виклику довідкової системи.
Border Style	Стиль рамки вікна	BsNone – відсутність рамки та заголовку вікна BsSingle – наявність рамки в 1 піксел BsSizeable – звичайна рамка BsDialog – рамка діалогового вікна
Client Height, Client Width	Висота та ширина клієнтської частини вікна	
Help File	Файл довідкової системи	
Icon	Піктограма форми	
Key Preview	Перехвачування формою подій клавіатури	
Menu	Головне меню вікна	
Position	Визначення положення та розмірів вікна в момент його з’явлення на екрані	poDesigned – розміри та положення такі ж як і на етапі проектування poDefault – положення та розміри визначаються Windows poScreenCenter – в центрі вікна poNone – відсутність масштабування poProportional – масштабування форми
Windows Style	Визначення стану вікна у момент появи на екрані	wsNormal – звичайне вікно wsMinimized – мінімізоване вікно wsMaximized – максимізоване вікно

5.2. Методи компонентів класу Tform

Таблиця 4.5.

Метод	**Пояснення**
Close	Закривання вікна
SetFocus	Передача фокусу форми

Метод	Пояснення
Show	Показ форми на екрані
ShowModal	Показ форми в модальному режимі

5.3. Події компонентів класу TForm.

Таблиця 4.6.

Подія	Пояснення
OnActivate	Виникає у момент активізація вікна
OnClose	Виникає у момент закриття вікна
OnCloseQuery	Запит на можливість закриття форми
OnCreate	Виникає у момент створення форми
OnDestroy	Виникає у момент знищення форми
OnHide	Виникає у момент приховування форми
OnShow	Виникає у момент появи форми на екрані

Огляд бібліотеки візуальних компонентів (VCL) системи Delphi. Компоненти введення та відображення текстової та чисельної інформації, дати та часу

Мета: ознайомити студентів з компонентами введення та відображення текстової та чисельної інформації, дати та часу.

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до викладання в школі.

Основні поняття: надпис, вікно редагування, компонента-перелік.

План лекції:

1. Компоненти введення та відображення текстової інформації. Загальний вигляд.

1.1. Відображення тексту в надписах компонентів Label та StaticText.

1.2. Вікна редагування Edit та MaskEdit.

1.3. Багаторядкові вікна редагування Memo та RichEdit.

1.4. Компонети-переліки: ListBox, ComboBox.

2. Компоненти введення числової інформації: UpDown та SpinEdit. Загальний огляд та особливості використання.

3. Компоненти введення та відображення дати та часу: DataTimePicker, MonthCalendar, Calendar. Загальний огляд та особливості використання.

Обрані методи: теоретичні, словесні, пояснювально-ілюстративний.

Наочність: таблиці з переліком основних компонентів введення та відображення текстової інформації, їх властивостей та методів.

Питання по темі для самостійного вивчення:

1. Основні положення та принципи технології ВООП.

Запитання для самоаналізу та самоперевірки:

1. Відображення тексту в надписах компонентів Label та StaticText.

2. Вікна редагування Edit та MaskEdit.
3. Багаторядкові вікна редагування Memo та RichEdit.
4. Компонети-переліки: ListBox, ComboBox.
5. Компоненти введення числової інформації: UpDown та SpinEdit. Загальний огляд та особливості використання.
6. Компоненти введення та відображення дати та часу: DataTimePicker, MonthCalendar, Calendar. Загальний огляд та особливості використання.

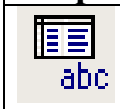
Текст лекції.

1. Компоненти введення та відображення текстової інформації.

Таблиця 5.1.

Загальний вигляд, коротка характеристика та перелік основних властивостей та методів.

Зображення	Компонент	Сторінка	Призначення
	Label (надпис)	Standard	Відображення тексту. Відсутність оформлення, окрім кольору надпису та тексту
	StaticText (надпис с бордюром)	Additional	Аналог компоненту Label з можливістю визначення стилю бордюру
	Edit (вікно редагування)	Standard	Відображення, введення та редагування однорядкового тексту з можливістю оформлення об’єктного бордюру
	MaskEdit (вікно маскового редагування)	Additional	Введення форматованих даних або символів у відповідності до визначеного шаблону
	Memo (багаторядкове вікно редагування)	Standard	Відображення, введення та редагування багаторядкового тексту
	RichEdit (багаторядкове вікно редагування в форматі RTF)	Win32	Вікно редагування у форматі RTF з можливостями вибору атрибутів тексту, пошуку фрагментів тексту
	ListBox (перелік)	Standard	Відображення стандартного вікна переліку, що дозволяє користувачеві вибирати з нього окремі пункти
	ComboBox (перелік із можливістю редагування)	Standard	Комбінація компонентів ListBox та Edit з можливістю як вибору, так і введення даних

Зображення	Компонент	Сторінка	Призначення
	StringGrid (таблиця символів)	Additional	Відображення текстової інформації у вигляді таблиці

1.1 Відображення тексту в надписах компонентів Label та StaticText

Таблиця 5.2.

Властивості компонентів Label та StaticText

Властивість	Призначення
Caption	Відображення текстової інформації. У разі необхідності відображення чисельної інформації використовують функцію floattostr та inttostr, що переводить в рядковий тип дійсні та цілі числа.
Top, Left	Координати компонента
Height, Width	Розміри компонента
Align	Вирівнювання компоненту. Можливі значення – alTop, alBottom, alRigth, визначають вирівнювання відносно верхньої, нижньої та правої частини клієнтської області. Значення alClient дозволяє заповнити компонентом всю клієнтську частину форми чи іншого контейнера.
Anchors	«Прив'язка» компонента. Можливі значення akTop, akLeft, akRight, akBottom – визначають прив'язку до верхнього, лівого, правого і нижнього берега батьківського компоненту.
Constraints	Визначення обмежень на допустимі зміни розмірів. Підвластивості MaxHeight, MaxWidth, MinHeight, MinWidth визначають максимальні та мінімальні розміри висоти та ширини компонента
Color	Колір фону компоненту
Font	Колір надпису компоненту
Visible	Видимість компоненти
WordWrap	Визначення можливості перенесення слів довгого надпису, що перевищує розміри компоненту, на новий рядок.

1.2 Вікна редагування Edit та MaskEdit.

Таблиця 5.3.

Основні властивості компонентів Edit та MaskEdit

Властивість	Призначення
Text	Відображення текстової інформації
MaxLength	Визначення максимальної довжини. Якщо MaxLength=0, то довжина необмежена
Modified	Визначення наявності змін в тексті вікна
PasswordChar	Перетворення символів, що вводить користувач, у символи, які встановлено в даній властивості (наприклад, символ *)
EditMask (компонента	Дозволяє встановити маску, що складається з розділів, які відокремлюються крапкою з комою. Перший розділ

Властивість	Призначення
MaskEdit)	визначає символи, які можна вносити в кожну позицію
Hint	Текст підказки, який буде висвічуватись при наведенні курсора на компоненту
ShowHint	Відображати чи не відображати підказку
ReadOnly	Забезпечує можливість вводу чи лише читання
Visible	Видимість компоненти

Таблиця 5.4.

Символи шаблону маски

Символ	Призначення
L	В даній позиції повинна бути літера
l	В даній позиції повинна бути літера або нічого
A	В даній позиції повинна бути буква або цифра
a	В даній позиції повинна бути буква або нічого
C	В даній позиції повинна бути будь-який символ
c	В даній позиції повинна бути будь-який символ або нічого
0	В даній позиції повинна бути цифра
g	В даній позиції повинна бути цифра або нічого
#	В даній позиції повинна бути цифра, знак “+”, “-” або нічого
:	Використовується для розділу годин, хвилин та секунд
/	Використовується для розділу місяців, днів та років

Визначення маски відбувається за допомогою вікна “Input MaskEditor” (у властивостях вибрати EditMask ... та натиснути …)

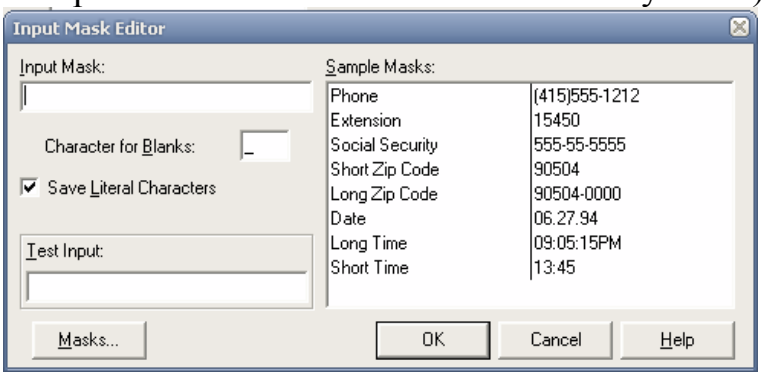


Рис. 21. Вікно визначення маски.

1.3 Багаторядкові вікна редагування Memo та RichEdit.

Таблиця 5.5.

Основні властивості компонентів Memo та RichEdit

Властивість	Призначення
Lines	Зміст тексту вікна у вигляді рядків. Доступ до певного рядку тексту відбувається за допомогою властивості Strings[index: integer]. Наприклад, Memo1.lines.Strings[0];
Count	Визначення загальної кількості рядків в тексті
ReadOnly	Встановлення режиму “Тільки для зчитування”
ScrollBars	Наявність скролінгу у вікні тексту

Таблиця 5.6.

Основні методи компонентів Memo та RichEdit

Методи	Призначення
Add /Append	Додавання нового рядку в кінець тексту
Clear	Очищення тексту у вікні
LoadFromFile	Завантаження тексту із файла
Savetofile	Збереження тексту у файл
Font (для Memo)	Встановлює атрибути шрифту для всього вікна
SelAttributes (для RichEdit)	дозволяє змінювати атрибути виділеного (нового) тексту

Таблиця 5.7.

Додаткові властивості компоненту RichEdit

Властивість	Призначення
AlignMent	Вирівнювання тексту. Можливі значення taleftjustify (ліворуч), tacenter (по центру), taRightJustify (праворуч)
FirstIndex	Кількість пікселів відступу в першому рядку абзацу
LeftIndend	Кількість пікселів відступу від лівого берега
RightIndent	Кількість пікселів відступу від правого берега

1.4 Компонети-переліки: ListBox, ComboBox.

Таблиця 5.8.

Основні властивості компонентів ListBox, ComboBox

Властивість	Компонент	Призначення
Text	ComboBox	Вибір користувача чи введення ним тексту
ItemIndex	ListBox, ComboBox	Індекс рядку, що був вибраний користувачем
Sorted	ListBox, ComboBox	Впорядкування переліку за алфавітом
Style	ListBox, ComboBox	Стиль компоненту
MultiSelect	ListBox	Можливість множинного вибору

2. Компоненти введення числової інформації: UpDown та SpinEdit. Загальний огляд та особливості використання.

Таблиця5.9.

Загальний огляд компонентів введення числової інформації

Зображення	Компонент	Сторінка	Призначення
	UpDown (лічильник)	Win32	Лічильник, що у комбінації з компонентом Edit, дозволяє вносити числову інформацію
	SpinEdit (лічильник)	Samples	Вікно редагування у комбінації з лічильником

Таблиця 5.10.

Перелік основних властивостей компонентів

Властивість	Компонент	Призначення
Associate	UpDown	Зв'язок лічильника із деяким віконним компонентом (наприклад, Edit)
Orientation	UpDown	Визначення кнопок лічильника по горизонталі чи вертикалі
Max/MaxValue Min/MinValue	UpDown, SpinEdit	Мінімальне та максимальне значення лічильника
Position/Value	UpDown, SpinEdit	Поточне значення лічильника
Wrap	UpDown	Реакція на досягнення мінімального/ максимального значень лічильника

3. Компоненти введення та відображення дати та часу: DataTimePicker, MonthCalendar, Calendar. Загальний огляд та особливості використання.

Таблиця 5.11.

Загальний огляд компонентів введення та відображення дати та часу

Зображення	Компонент	Сторінка	Призначення
	DataTimePicker (календар)	Win32	Введення дати та часу
	MonthCalendar (календар)	Win32	Введення дати з вибором із календаря
	Calendar (календар на місяць)	Samples	Відображення календаря на певний місяць

Таблиця 5.12.

Загальні властивості компонентів

Властивість	Компонент	Призначення
Kind	DataTimePicker	Режим роботи: DtkDate – внесення дати DtkTime – внесення часу
Date	DataTimePicker	Значення дати (Kind DtkDate)
Time	DataTimePicker	Значення часу (Kind DtkTime)
MinDate/ MaxDate	DataTimePicker	Мінімальне / максимальне значення дати
MultiSelect	MonthCalendar	Множинний вибір дати
First Day of Week	MonthCalendar	Визначення першого дня тижня
Year, Month, Day	Calendar	Рік, місяць, день

Компоненти відображення графічної та мультимедійної інформації.

Мета: ознайомити студентів з компонентами відображення графічної та мультимедійної інформації.

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до викладання в школі.

Основні поняття: компоненти віображення графічної та мультимедійної інформації.

План лекції:

1. Компоненти відображення графічної інформації. Загальний огляд та особливості використання.
   1.1. Компоненти відображення графічної інформації Image.
   1.2. Компоненти побудови геометричних примітивів – Shape.
   1.3. Компоненти створення області для малювання: PaintBox.
2. Компоненти відображення мультимедійної інформації. Загальний огляд та особливості використання.
   2.1. Компоненти відтворення стандартних мультиплікаційних файлів Windows та відеокліпів – Animate.
   2.2. Компоненти відтворення мультимедійної інформації – MediaPlayer.

Обрані методи: теоретичні, словесні, наочні, пояснювально-ілюстративний.

Наочність: таблиці з переліком основних компонентів відображення графічної та мультимедійної інформації, їх властивостей та методів.

Питання по темі для самостійного вивчення:

Компоненти відтворення стандартних мультиплікаційних файлів Windows та відеокліпів – Animate.

Запитання для самоаналізу та самоперевірки:

2. Компоненти відображення графічної інформації Image.
3. Компоненти побудови геометричних примітивів – Shape.
4. Компоненти створення області для малювання: PaintBox.
   1. Компоненти відтворення мультимедійної інформації – MediaPlayer.

Текст лекції.

1. Компоненти відображення графічної інформації. Загальний огляд та особливості використання.

Таблиця 6.1.

Загальний огляд та особливості використання компонентів відображення графічної інформації

Зображення	Компонент	Сторінка	Призначення
	Image (графічне зображення)	Additional	Відображення графічної інформації: піктограм, бітових матриць та метафайлів
	Shape (геом. фігура)	Additional	Формування геометричних фігур

	PaintBox (область для малювання)	System	Створення області малювання на формі

1.1. Компонент відображення графічної інформації - Image.

Таблиця 6.2.

Основні властивості компонентів відображення графічної інформації Image

Властивість	Призначення
Picture	Графічне зображення
AutoSize	Автоматична адаптація розмірів компонента до розмірів графічного зображення
Strech	Автоматична адаптація графічного зображення до розмірів компоненту
Center	Вирівнювання графічного зображення
Transparent	Прозорість компонента
Visible	Видимість компонента

Таблиця 6.3.

Основні методи компонентів відображення графічної інформації Image

Методи	Призначення
LoadFromFile	Завантаження графічного зображення з файлу (Наприклад: Image1.Picture.LoadFromFile(‘c:\1.bmp’))
SaveTofile	Збереження графічного зображення у файл (Наприклад: Image1. Picture.SaveToFile(‘c:\1.bmp’))

1.2 Компонент побудови геометричних примітивів – Shape.

Таблиця 6.4.

Основні властивості компонента Shape

Властивість	Призначення	Можливі значення
Shape	Форма геометричної фігури.	stRectangle – прямокутник stRoundRect – прямокутник з краями заокругленої форми stEllipse – еліпс; stCircle – коло stSquare – квадрат stRoundSquare – квадрат з краями заокругленої форми
Brush	Заповнення	
Pen	Олівець	Підвластивість Style має наступні значення: psSolid – суцільна лінія psDash – заштрихована лінія psdot – пунктирна лінія psDashDot – штрих-пунктирна лінія psDashDotDot – штрих і два пунктири psClear – лінія відсутня

1.3. Компоненти створення області для малювання PaintBox.

Окрім розглянутих вище компонентів, що дозволяють відображувати

графічну інформацію та стандартні примітиви, в системі Delphi існує можливість створення спеціальних областей для малювання. Така область (канва) реалізується за допомогою компоненту PaintBox. Основна властивість компоненту PaintBox – Canvas. Властивість Canvas є і у деяких інших компонентів, таких як Form, Image. Але використання компоненту PaintBox дозволяє реалізувати області малювання, зокрема і на компонентах, що не мають властивості Canvas.

Таблиця 6.5.

Методи класу Canvas

Методи	Призначення
Arc	Дуга кола та еліпса
Chord	Замкнута фігура, що обмежена дугою кола чи еліпса та хордою
Ellipse	Коло чи еліпс
Pie	Сектор кола чи еліпса
Rectangle	Прямокутник
RoundRect	Прямокутник із закругленими краями
MoveTo	Переміщення курсора на нову позицію
Draw	Виведення зображення
LineTo	Малювання ліній від поточної позиції курсора до точки, що вказана у якості параметра
TextOut	Виведення тексту в конкретну позицію канви

2. Компоненти відображення мультимедійної інформації. Загальний огляд та особливості використання.

Таблиця 6.6.

Зображення	Компонент	Сторінка	Призначення
	Animate (програвач відео-фрагментів)	Win32	Відображення стандартних мультиплікаційних файлів Windows та файлів .avi
	MediaPlayer (програвач мультимедійної інформації)	System	Універсальний програвач аудіо-, відеоінформації.

2.1 Компоненти відтворення стандартних мультиплікаційних файлів Windows та відеокліпів – Animate

Таблиця 6.7.

Властивості компоненту класу TAnimate

Властивість	Призначення
CommonAvi	Відтворення стану мультиплікації файлу. Можливі слідуючі значення властивості: AviCopyFile – копіювання файлу AviCopyFiles – копіювання файлів AviDeleteFile – знищення файлів AviEmptyRecycle – очищення корзини

	AviFindComputer – пошук комп'ютера AviFindFile – пошук файлів AviFindFolder – пошук папок AviNone – анімація відсутня AviRecycleFile – вилучення файлів в корзину
FileName	Ім’я відео кліпу (*.avi) Обмеження: • Відеокліп не повинен містити звук • Інформація не повинна бути стисненою • Розмір файлу не повинен перевищувати 64 КБ
Active	Активізація відтворення відео фрагменту
FrameCount	Кількість кадрів відео фрагменту
StartFrame, StopFrame	Початковий та кінцевий кадр
Repetitions	Кількість повторень відтворення файлу

Таблиця 6.8.

Методи компоненту класу TAnimate

Методи	Призначення
Play	Відтворення відео фрагменту, починаючи з кадру N до кадру K, певну кількість
Stop	Припинення відтворення

2.2 Компоненти відтворення мультимедійної інформації – MediaPlayer

Таблиця 6.9.

Основні властивості компоненту відтворення мультимедійної інформації – MediaPlayer

Властивість	Призначення
FileName	Ім’я файлу для відтворення
DeviceType	Тип пристрою мультиплікації
Display	Віконний елемент для відтворення відео фрагменту
Visible Buttons/ Enabled Buttons	Видимість/активність кнопок інтерфейсу програвача

Таблиця 6.10.

Основні властивості компоненту відтворення мультимедійної інформації – MediaPlayer

Методи	Призначення
Play	Відтворення інформації
Pause	Призупинення відтворення/запису інформації
Stop	Зупинка відтворення/запису інформації
Next	Перехід до наступного запису
Prew	Перехід до попереднього запису
Step	Переміщення вперед на певну кількість кадрів
Back	Переміщення назад на певну кількість кадрів

Record	Початок запису
Eject	Вивільнення об’єкту з програвача

Компоненти - елементи керування.

Мета: ознайомити студентів з елементами керування.

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до викладання в школі.

Основні поняття: елементи керування.

План лекції:

1. Компоненти - елементи керування. Загальний огляд та особливості використання.
2. Компоненти-кнопки керування.
2.1. Стандартна кнопка Button та кнопка з піктограмою BitBtn.
2.2. Кнопка швидкого доступу SpeedButton.
3. Компоненти-перемикачі.
3.1. Перемикачі із незалежною фіксацією ComboBox та CheckListBox.
3.2. Перемикачі із залежною фіксацією (кнопки вибору) RadioButton та група кнопок вибору RadioGroup.
4. Групування елементів керування. Компоненти GroupBox, Panel, ScrollBox.
5. Системний таймер – компонент Timer.

Обрані методи: теоретичні, словесні, пояснювально-ілюстративний.

Наочність: таблиці з переліком основних елементів керування, їх властивостей та методів.

Питання по темі для самостійного вивчення:

4. Системний таймер – компонент Timer.

Запитання для самоаналізу та самоперевірки:

1. Стандартна кнопка Button та кнопка з піктограмою BitBtn.
2. Кнопка швидкого доступу SpeedButton.
3. Перемикачі із незалежною фіксацією ComboBox та CheckListBox.
4. Перемикачі із залежною фіксацією та група кнопок вибору.

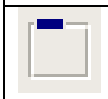
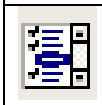
Текст лекції.

1. Компоненти - елементи керування. Загальний огляд та особливості використання.

Таблиця 7.1.

Загальний огляд та особливості використання елементів керування.

Зобр-ня	Компонент	Сторінка	Призначення
	Button (кнопка)	Standard	Створення кнопки
	BitBtn (кнопка з піктограмою)	Additional	Створення кнопки з піктограмою

Зобр-ня	Компонент	Сторінка	Призначення
	SpeedButton (“швидка” кнопка)	Additional	Створення кнопки, що використовується інструментальний панелях в якості “швидкої” кнопки для дублювання команди меню
	RadioButton (радіокнопка)	Standard	Створення залежного перемикача (альтернативи)
	RadioGroup (група радіо кнопок)	Standard	Перелік залежних перемикачів – поєднання вікна групування (GroupBox) та перемикачів (RadioButton)
	GroupBox (вікно групування)	Standard	Створення контейнера, що поєднує групу елементів керування
	CheckBox (контрольний індикатор з прапорцем)	Standard	Створення індикатора, що дозволяє встановлювати деякі параметри
	CheckListBox (перелік з індикатором)	Additional	Створення переліку з індикатором – комбінація компоненту-переліку ListBox з індикатором CheckBox
	Panel (панель)	Standard	Створення контейнера, що поєднує групу елементів керування
	TrackBar (“повзунок”)	Win32	Створення елементів керування у вигляді повзунка (ліфту)
	ScrollBar (смуга прокручуванн я)	Standard	Створення стандартної смуги прокручування для керування видимою частиною форми чи компонентів
	Timer (таймер)	System	Створення таймера для керування процедурами, функціями та подіями в певний інтервал часу

2. Компоненти-кнопки керування.

2.1 Стандартна кнопка Button та кнопка з піктограмою BitBtn.

Таблиця 7.2.

Основні властивості компонентів Button та BitBtn

Властивість	Компонент	Призначення
Caption	Button, BitBtn	Створення надпису на кнопці. Використання символу амперсанда “&” дозволяє організувати мнемонічний доступ до кнопки. Наприклад, значення властивості Caption “В&иконати” призводить до відображення надпису “Виконати” та активізації кнопки за допомогою комбінації клавіш Alt + и

Властивість	Компонент	Призначення
Default	Button, BitBtn	Властивість, що дозволяє активізувати кнопки при натисненні клавіші Enter
Cancel	Button, BitBtn	Властивість, що дозволяє активізувати кнопки при натисненні клавіші Esc
Glyph	BitBtn	Встановлення графічного зображення (*.bmp) на кнопці (піктограми)
Margin	BitBtn	Розміщення надпису на піктограмі
Layout	BitBtn	Положення піктограми відносно надпису
Spasing	BitBtn	Кількість пікселів, що розділяють надпис та піктограму
Kind	BitBtn	Визначення типу кнопки, що має власні параметри (надпис, піктограму)

Таблиця 7.3.

Основний метод компонентів Button та BitBtn

Методи	Компонент	Призначення
Click	Button, BitBtn	Активізація кнопки (натиснення лівої кнопки миші на компонентові)

2.2 Кнопка швидкого доступу SpeedButton.

Таблиця 7.4.

Основні властивості компонента SpeedButton

Властивість	Призначення
GroupIndex	Групування кнопок
Down	Переведення іконки в стан “ненатистена”
AllowAllUp	Виведення із стану не натиснення (активізацією іншої іконки групи) Повторити натиснення на кнопці
MouseInComponent	Визначення розташування курсора під та над іконкою

3. Компоненти-перемикачі

3.1 Перемикачі із незалежною фіксацією CheckBox та CheckListBox.

Таблиця 7.5.

Основні властивості компонентів ComboBox та CheckListBox

Властивість	Компонент	Призначення
Caption	CheckBox	Текст перемикача
Alignment	CheckBox	Розміщення тексту по відношенню до індикатора
Checked	CheckBox, CheckListBox	Активність перемикача (значення TRUE \ FALSE)
AllowGrayed	CheckBox, CheckListBox	Можливість встановлення трьох станів перемикача.
State	CheckBox, CheckListBox	Аналіз стану перемикача. Можливі значення: CbUnChecked – не активний CbChecked – активний, CbGrayed – частково активний
Visible	CheckBox, CheckListBox	Видимість перемикача

Enabled	CheckBox, CheckListBox	Доступність перемикача

3.2 Перемикачі із залежною фіксацією (кнопки вибору) RadioButton та група кнопок вибору RadioGroup

Таблиця 7.6.

Основні властивості компонентів RadioButton та RadioGroup

Властивість	Компонент	Призначення
Caption	RadioButton	Текст перемикача / групи перемикачів
Alignment	RadioButton	Розміщення тексту відносно перемикача
Checked	RadioGroup	Активність перемикача
Items	RadioGroup	Текст перемикачів та їх кількість
ItemIndex	RadioGroup	Доступ до окремого перемикача (ItemIndex=-1 – ні один із перемикачів групи неактивний)
Colums	RadioGroup	Кількість стовпчиків в групі перемикачів

4. Групування елементів керування. Компоненти GroupBox, Panel, ScrollBox.

Таблиця 7.7.

Основні властивості компонент GroupBox, Panel, ScrollBox

Властивість	Компонент	Призначення
Caption	Panel GroupBox	Заголовок вікна групування
BorderStyle	Panel	Стиль рамки
BevelInner BevelOuter	Panel	Вигляд внутрішньої та зовнішньої кромки
FullRepaint	Panel	Перемальовування панелі та всіх її компонентів при зміні її розмірів
AutoScroll	ScrollBox	Автоматичне появлення смуг прокручування
HorsScrollBar VertScrollBar	ScrollBox	Відображення горизонтальної та вертикальної смуги прокручування
ScrollInView	ScrollBox	Прогалина керування вікном прокручування

5. Системний таймер – компонент Timer.

Таблиця 7.8.

Основні властивості компонента Timer

Властивість	Призначення
Interval	Інтервал часу в мілісекундах (1 с. = 1000 одиниць), що встановлює період активізації таймера
Enabled	Доступність таймеру

Приклад використання компоненту Timer

```
Procedure Tform1.Timer1Timer (Sender:TObject);
Begin
    DateTime:=Time;
    Label1.caption:=TimeToStr(DateTime);
```

End;

Компоненти – меню та компоненти-панелі.

Мета: ознайомити студентів з компонентами-меню та компонентами-панелями.

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до викладання в школі.

Основні поняття: меню, панель.

План лекції:

1. Компоненти-меню. Загальний огляд та особливості використання.

1.1. Головне та контекстно-залежне меню - компонент MainMenu та PopupMenu.

1.2. Конструктор меню MenuDesigner.

1.3. Динамічна організація меню.

2. Компоненти - панелі. Загальний огляд та особливості використання.

2.1. Інструментальна панель – компонент ToolBar.

2.2. Розширена панель інструментів TcoolBar.

2.3. Рядок стану: компонент TstatusBar.

Обрані методи: теоретичні, наочні, пояснювально-ілюстративний.

Наочність: таблиці з переліком компонентів меню та панелей інструментів, їх властивостей та методів.

Питання по темі для самостійного вивчення:

1. Особливості об'єктно-орієнтованого програмування.

Запитання для самоаналізу та самоперевірки:

1. Конструктор меню MenuDesigner.
2. Динамічна організація меню
3. Інструментальна панель – компонент ToolBar
4. Розширена панель інструментів TcoolBar

Текст лекції.

1. Компоненти-меню. Загальний огляд та особливості використання.

Таблиця 8.1.

Основні властивості компонентів - меню та панелі інструментів

Зображення	Компонент	Сторінка	Призначення
	MainMenu (головне меню)	Standard	Створення головного меню додатка
	PopUpMenu (контекстно-залежне меню)	Standard	Створення контекстно-залежного меню додатка

1.1 Головне та контекстно-залежне меню - компонент MainMenu та PopupMenu

Таблиця 8.2.

Основні властивості компонентів MainMenu та PopupMenu

Властивість	Призначення
Caption	Текст розділу та пункту меню. Використання символу амперсанд “&” дозволяє організувати клавіші мнемонічного доступу. Символ “-” в значенні властивості дозволяє створити розподільну смугу
ShortCut	Визначення клавіші швидкого доступу
Default	Визначення пункту за умовчуванням для окремого розділу
Break	Розподіл переліку розділів на декілька стовпчиків
Checked	Вибір пункту меню
Items	Масив пунктів певного розділу
BitMap	Графічне зображення, що відображається зліва від тексту пункту меню
ImageIndex	Індекс зображення із компоненту ImageList, що визначається властивістю Images компонента MainMenu
Create Submenu	Створення розгалуженого меню

Таблиця 8.3.

Динамічна організація меню

Методи	Призначення
Add	Додавання пункту меню
Insert	Додавання пункту меню у вказану позицію
Delete	Знищення вказаного пункту меню
SelectMenu	Вибір меню
SaveAsTemplate	Збереження меню у якості шаблону
InsertfromTemplate	Додавання меню із шаблону
DeleteTemplate	Знищення шаблону меню
InsertfromResourse	Додавання меню із ресурсу (файли *.rc та *.mnu)

Приклад:

```
Procedure Tform1.mnuItemAddClick(Sender:TSender)
Var NewItem:Tmenuitem;
Begin
NewItem:=TmenuItem.Create(self);
NewItem.caption:=’Новий елемент’;
MnuFile.add(NewItem); // АбоMnuFile.Insert(2, NewItem);
End;
```

2. Компоненти - панелі. Загальний огляд та особливості використання.

Таблиця 8.4.

Зобр-ня	Компонент	Сторінка	Призначення
	ToolBar (інструментальна панель)	Win32	Створення інструментальної панелі для швидкого доступу до функції застосування, що використовуються найчастіше
	CoolBar (інструментальна панель довільного розміру)	Win32	Створення інструментальної панелі, розміри якої може змінювати користувач
	StatusBar (рядок стану)	Win32	Створення рядку стану застосування

2.1 Інструментальна панель – компонент ToolBar

Таблиця 8.5.

Властивості компонентів класу TToolBar

Властивість	Призначення
ButtonCount	Кількість дочірніх компонентів в панелі інструментів
Flat	“Прозорість” кнопки
HotImages	Зазначення контейнера зображень для кнопок в момент, коли над ними розташований вказівник миші
Images	Визначення переліку зображень для кнопок у звичайному стані
RowCount	Кількість рядків кнопок панелі інструментів
ShowCaption	Виведення тексту на кнопках
Wrapable	Дозвіл на розташування кнопок в декілька рядків

Таблиця 8.6.

Властивості класу TToolButton

Властивість	Призначення
Caption	Текст, що пов’язаний з конкретною кнопкою
Down	Стан кнопки
DropDownMenu	Зв’язок кнопки з контекстно-залежним меню
ImageIndex	Індекс зображення, що пов’язаний з кнопкою
MenuItem	Визначення пункту головного чи залежного меню, що буде дублюватися кнопкою
Style	Стиль кнопки. Можливі значення: TbsButton – звичайна кнопка TbsCheck – кнопка, що залишається в “натиснутому стані” TbsDropDown – кнопка-меню TbsSeparator – кнопка, що розділяє кнопки панелі

Властивість	Призначення
	інструментів
Wrap	Можливість розміщення кнопок у декілька рядків

2.2 Розширена панель інструментів CoolBar

Таблиця 8.7.

Властивості розширеної панелі інструментів CoolBar

Властивість	Призначення
BitMap	Фоновий малюнок
Images	Перелік зображень для панелі інструментів
ShowText	Заголовок панелі інструментів

Таблиця 8.8.

Властивості класу TCoolBand

Властивість	Призначення
Break	Розташування панелі з нового рядку
Control	Елемент керування, що розташований на панелі
ImageIndex	Індекс зображення для панелі інструментів
Text	Заголовок панелі інструментів
BitMap	Фоновий малюнок

2.3 Рядок стану: компонент класу TStatusBar

Таблиця 8.9.

Властивості рядку стану TStatusBar

Властивість	Призначення
Panels	Перелік об’єктів-панелей
SimplePanel	Можливість створення декількох панелей
SimpleText	Заголовок панелі (у випадку однієї панелі)

Таблиця 8.10.

Властивості класу TStatusPanel

Властивість	Призначення
Style	Стиль панелі. Можливі значення: PsText – панель містить тільки текст, що відображається автоматично PsOwnerDraw – панель перемальовується програмно
Text	Текст надпису в секції
Width	Ширина секції в панелі

Компоненти – системні діалоги.

Мета: ознайомити студентів з системними діалогами.

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до викладання в школі.

Основні поняття: системний діалог.

План лекції:

1. Компоненти – системні діалоги. Загальний огляд та особливості використання.

2. Компоненти-діалоги відкриття та збереження файлів: OpenDialog, OpenPictureDialog, SaveDialog, SavePictureDialog.

3. Компоненти-діалоги вибору шрифту: FontDialog.

4. Компоненти-діалоги вибору кольору: ColorDialog.

5. Компоненти-діалоги визначення параметрів друку та принтера: PrintDialog та PrintSetupDialog.

6. Компоненти-діалоги пошуку та заміни тексту: FindDialog, ReplaceDialog.

Обрані методи: теоретичні, словесні, наочні, пояснювально-ілюстративний.

Наочність: таблиці з переліком компонентів – системних діалогів, їх властивостей та методів.

Питання по темі для самостійного вивчення:

1. Компоненти-діалоги визначення параметрів друку та принтера.

Запитання для самоаналізу та самоперевірки:

7. Компоненти-діалоги відкриття та збереження файлів.
8. Компоненти-діалоги вибору шрифту та кольору.
9. Компоненти-діалоги пошуку та заміни тексту.

Текст лекції.

1. Компоненти – системні діалоги. Загальний огляд та особливості використання.

Таблиця 9.1.

Загальний огляд та особливості використання компонентів – системних діалогів

Зобр-ня	Компонент	Сторінка	Призначення
	OpenDialog, OpenPictureDialog (відкривання файлу/ графічного зображення)	Dialogs	Виклик стандартного діалогового вікна для відкривання файлу/ графічного зображення
	SaveDialog, SavePictureDialog (збереження файлу/ графічного зображення)	Dialogs	Виклик стандартного діалогового вікна для збереження файлу/ графічного зображення
	FontDialog (вибір шрифту)	Dialogs	Виклик стандартного діалогового вікна для вибору шрифту
	ColorDialog (вибір кольору)	Dialogs	Виклик стандартного діалогового вікна для вибору кольору
	PrintDialog (друк)	Dialogs	Виклик стандартного діалогового вікна для визначення параметрів друку

Зобр-ня	Компонент	Сторінка	Призначення
	PrintSetupDialog (параметри принтера)	Dialogs	Виклик стандартного діалогового вікна для визначення параметрів принтера
	FindDialog (пошук)	Dialogs	Виклик стандартного діалогового вікна для пошуку фрагменту тексту
	ReplaceDialog (заміна)	Dialogs	Виклик стандартного діалогового вікна для заміни фрагменту тексту

Основні етапи роботи з компонентами – системними діалогами

1. Вибір необхідного компонента - діалогу та визначення його параметрів

2. Виклик діалогового вікна за допомогою методу execute (if <ім'я компонента-діалога> execute then <оператори, що виконують вибір користувача>)

3. Аналіз результатів роботи з діалоговим вікном

2. Компоненти-діалоги відкриття та збереження файлів: OpenDialog, OpenPictureDialog, SaveDialog, SavePictureDialog.

Таблиця 9.2.

Основні властивості компонентів OpenDialog, OpenPictureDialog, SaveDialog, SavePictureDialog

Властивість	Призначення
DefaultExt	Тип файлів, що визначено за умовчуванням. Властивість дозволяє записувати тільки ім'я файлу – тип додається автоматично
FileName	Ім'я файлу. Повне ім'я файлу, який визначив користувач.
Files	Перелік файлів, які вибрав користувач
Filter	Перелік шаблонів, у відповідності до яких відображаються файли в діалоговому вікні. Наприклад: текстові |*.txt| програмні |*.pas|
FilterIndex	Номер шаблону, який буде показано за умовчуванням
InitialDir	Початковий каталог, який буде відображено у вікні діалогу.
Options	Параметри, що визначають роботу діалогового вікна.
Title	Заголовок діалогового вікна

Приклад використання компонентів:

```
Procedure TForm1.Button1Click (Sender: TObject);
Begin
    OpenDialog1.Filter:= "Всі файли |*.*| текстові |*.txt|";
    OpenDialog1.Title:="Вибір необхідного файлу";
If OpenDialog1.Execute then Memo1.Lines.LoadFromFile (OpenDialog1.FileName)
End;
```

3. Компоненти-діалоги вибору шрифту: FontDialog

Таблиця 9.3.

Основні властивості компоненту FontDialog

Властивість	**Призначення**
Device	Визначення пристрою, перелік шрифтів якого буде відображено. Можливі значення: FdScreen – екран FdPrinter – принтер FdDoth – обидва пристрої Font Шрифт, який вибрав користувач
MaxFontSize MinFontSize	Максимальні/мінімальні розміри шрифту
Options	Додаткові характеристики зовнішнього вигляду діалогового вікна

Приклад роботи із діалоговим вікном вибору шрифту:

```
Procedure TForm1.Button1Click (Sender: Tobject);
Begin
    If FontDialog1.Execute then Memo1.Font.Assign(FontDialog1.Font);
End;
```

4. Компоненти-діалоги вибору кольору: ColorDialog

Таблиця 9.4.

Основні властивості компоненту ColorDialog

Властивість	**Призначення**
Color	Колір, що був вибраний користувачем
CustomColor	Кольори додаткової палітри, що були визначені користувачем. Діапазон ColorA - ColorB.
Options	Додаткові характеристики зовнішнього вигляду діалогового вікна

Приклад роботи із діалоговим вікном вибору кольору:

```
Procedure TForm1.Button1Click (sender: Tobject);
Begin
    If ColorDialog1.Execute then Form1.Color:=ColorDialog1.Color;
End;
```

5. Компоненти-діалоги визначення параметрів друку та принтера.

Таблиця 9.5.

Основні властивості компонентів PrintDialog та PrintSetupDialog

Властивість	**Призначення**
Collate	Визначення другу по копіям (перемикач “Разобрать”)
Copies	Кількість копій
FromPage	Визначення початкової сторінки друку
MaxPage MinPage	Визначення кордонів діапазону друку “Страницы с ... по ...”
Options	Додаткові характеристики зовн. вигляду діалогового вікна

Властивість	Призначення
PrintRage	Визначення діапазону друку. Можливі значення: PrAllPages – всі сторінки PrSelection – виділений фрагмент тексту PrPageNum – сторінки за номерами
PrintToFile	Вибір друку за параметром “Печать в файл”
ToPage	Визначення кінцевої сторінки друку

Діалог TprintSetupDialog працює безпосередньо з драйвером принтера і не має специфічних властивостей, значення яких необхідно встановлювати.

6. Компоненти-діалоги пошуку та зміни тексту.

Таблиця 9.6.

Основні властивості компонентів-діалогів пошуку та зміни тексту

Властивість	Призначення
FindText	Фрагмент тексту для пошуку
ReplaceText	Фрагмент тексту для заміни (для компоненту ReplaceDialog)
Left, Top	Координати лівого та верхнього краю діалогового вікна
Options	Додаткові характеристики зовнішнього вигляду діалогового вікна
Position	Координати лівого та верхнього позицій діалогового вікна

Приклад роботи із діалоговим вікном пошуку:

```
Procedure Tform1.FotDialog1Find (Sender: TObject);
Var SelPos:Integer;
Begin
With TfindDialog1 (Sender) do
Begin
SelPos:=Pos(FintText, Memo1.lines.text);
If SelPos>0 then
Begin
Memo1.SelStart:=SelPos-1;
Memo1.SelLength:=Length(FindText)
End
Else showmessage(‘Текст’+FindText+ ‘ не знайдено’);
End;
End;
```

Повторне використання програмного коду. Загальний огляд механізмів збереження та повторного використання програмного коду. Шаблони та Депозитарії.

Мета: ознайомити студентів з механізмами збереження та повторного використання програмного коду.

Професійна спрямованість: дана лекція є складовою частиною професійно***ї*** підготовки вчителя інформатики до викладання в школі.

Основні поняття: програмний код, шаблон, депозитарій.

План лекції:

1. Механізм створення, збереження та використання шаблонів компонентів.

1.1. Створення компоненту (групи компонентів) для використання у якості шаблону.

1.2. Збереження шаблону компонента.

1.3. Використання шаблонів компонентів.

2. Депозитарій – сховище проектів та форм.

2.1. Активізація вікна Депозитарію.

2.2. Додавання об’єкту до Депозитарію.

2.3. Режим використання форм та проектів з Депозитарію.

3. Бібліотеки, що під’єднуються динамічно (Dinamic Link Library - DLL).

3.1. Бібліотеки, що під’єднуються динамічно (DLL) як механізм повторного використання програмного коду. Переваги використання.

3.2. Створення бібліотек.

3.3. Виклик бібліотек.

4. Пакети (Paskeges). Загальний огляд пакетів.

Обрані методи: теоретичні, словесні, пояснювально-ілюстративний.

Наочність: схема механізмів збереження та повторного використання програмного коду, графічні подання діалогових вікон, що відображають основні етапи роботи над матеріалом лекції.

Питання по темі для самостійного вивчення:

1. Бібліотеки, що під’єднуються динамічно.

Запитання для самоаналізу та самоперевірки:

1. Збереження шаблону компонента.
2. Використання шаблонів компонентів.
3. Активізація вікна Депозитарію.
4. Режим використання форм та проектів з Депозитарію.
5. Виклик бібліотек.

Текст лекції.

1. Механізм створення, збереження та використання шаблонів компонентів.

1.1 .Створення компоненту (групи компонентів) для використання у якості шаблону.

Для створення компоненту, що надалі буде використовуватися у якості шаблону, скористаємося компонентом класу TEdit.

Внесемо в обробник подій OnKeyPress оператор

if not (key in ['0'..'9']) then key:=#0, який дозволяє ввести тільки цифри.

1.2 Збереження шаблону компонента

Component \ Create Component Template

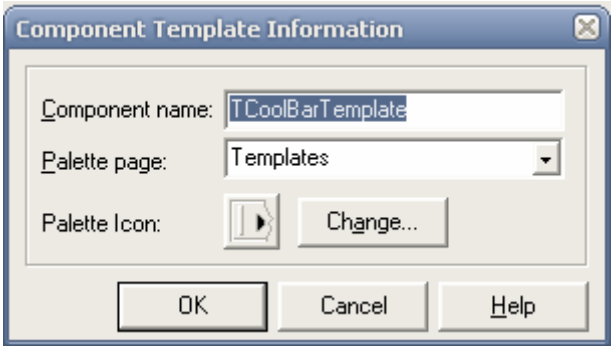


Рис.22. Вікно визначення шаблону компонента

1.3 Використання шаблонів компонентів.

Component \ Configure Palette

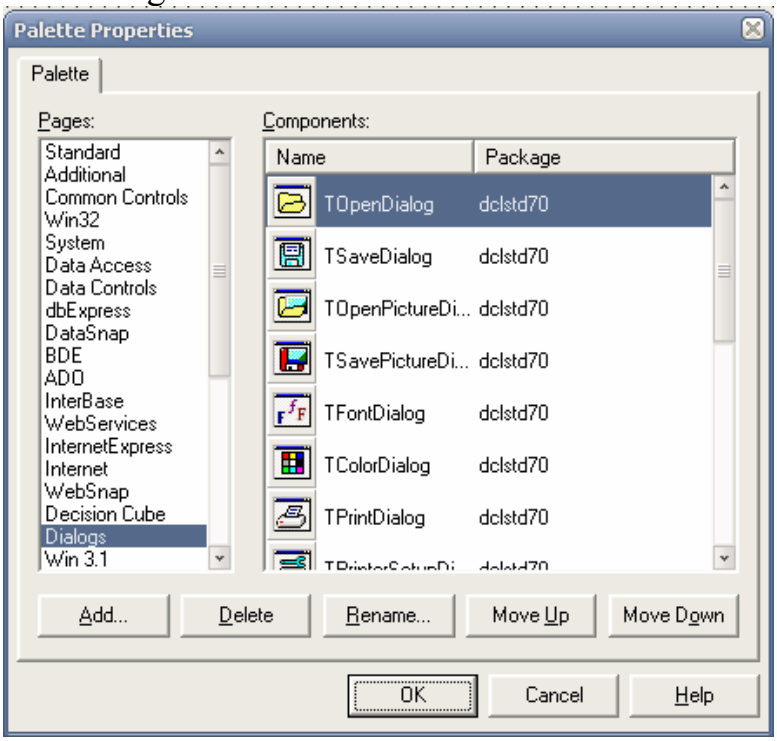


Рис. 23. Вікно конфігурації Палітри компонентів

2. Депозитарій – сховище проектів та форм.

2.1 Активізація вікна Депозитарію.

File \ New

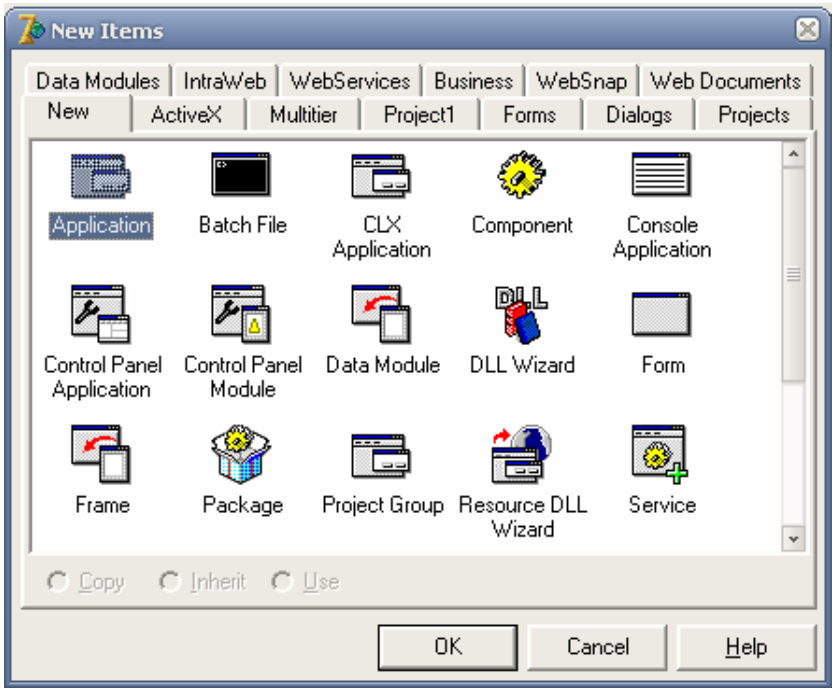


Рис. 24. Вікно Депозитарію.

2.2 Додавання об'єкту до Депозитарію.

Project \ Add to Repository …

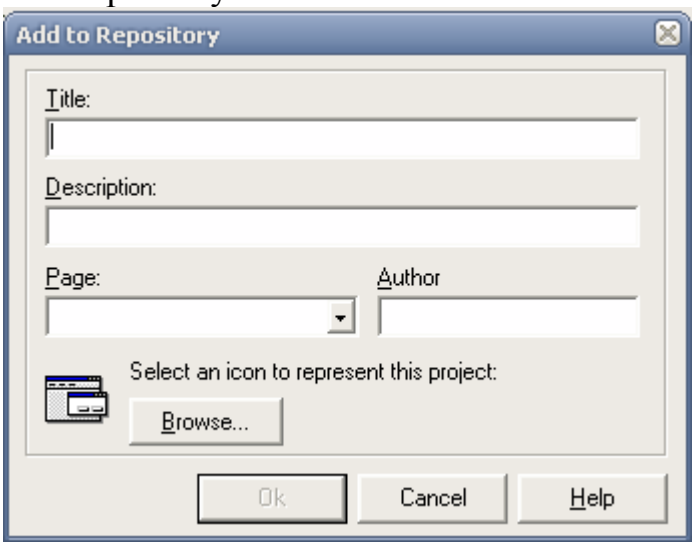


Рис. 25. Вікно додавання форми до Депозитарію

Title – назва форми

Description – розширене пояснення (активізація команди View Details контекстно-залежного вікна Депозитарію)

Page – сторінка Депозитарію для збереження форми

Author – відомості про автора

2.3 Режим використання форм та проектів з Депозитарію.

• **Copy** (копіювання) – копіювання форми з Депозитарію до проекту (зв'язок вихідної форми з Депозитарію і копією переривається).

• **Inherit** (наслідування) – включення до проекту форми, що наслідує форму з Депозитарію (зміни у формі Депозитарію після перекомпіляції відображається на формах проекту)

• **Use** (використання) - включення до проекту форми, що використовується для внесення змін до форми з Депозитарію

3. Бібліотеки, що під'єднуються динамічно (Dinamic Link Library - DLL).

3.1 Бібліотеки, що під'єднуються динамічно (DLL) як механізм повторного використання програмного коду. Переваги використання та основні етапи роботи з бібліотеками.

Переваги використання

• Економія оперативної пам'яті

• Можливість обміну коду між різноманітними системами програмування

• Поліпшення процесу модифікації додатку

Етапи роботи з бібліотеками

• Створення бібліотеки

• Виклик бібліотеки

3.2 Створення бібліотек.

File \ New \ DLL

library Project1;

{ коментарі щодо необхідності включення модулю Share Mem до переліку uses, якщо функції бібліотеки передають рядки в якості параметрів чи результатів }

uses SysUtils, Classes;
begin
end.

Приклад роботи з бібліотекою:

Library MyDLL;
uses SysUtils, Classes;
function Average(A, B, C:real); export; stdCall;
begin
average:=(a+b+c)/3;
end;
Export
Average; //Перелік функцій, що повинні бути експортовані
End.

Ключові слова:

Export (призначення функцій для зовнішнього використання)

StdCall (використання стандартних домовленостей щодо передачі параметрів при виклику)

3.3 Виклик бібліотек.

Приклад виклику бібліотеки із додатка:

Implamentation
*{$R *.dfm}*
function Average(A, B, C:real): real; far; external ('MyDLL');
procedure Tform1.Button1Click (Sender:TObject);

begin

Label1.caption:=FloatToStr((Average(StrToFloat(Edit1.text)+StrToFloat(Edit2.text)+ StrToFloat(Edit3.text))/3);

end;

4. Пакети (Packeges). Загальний огляд пакетів.

Project \ Option

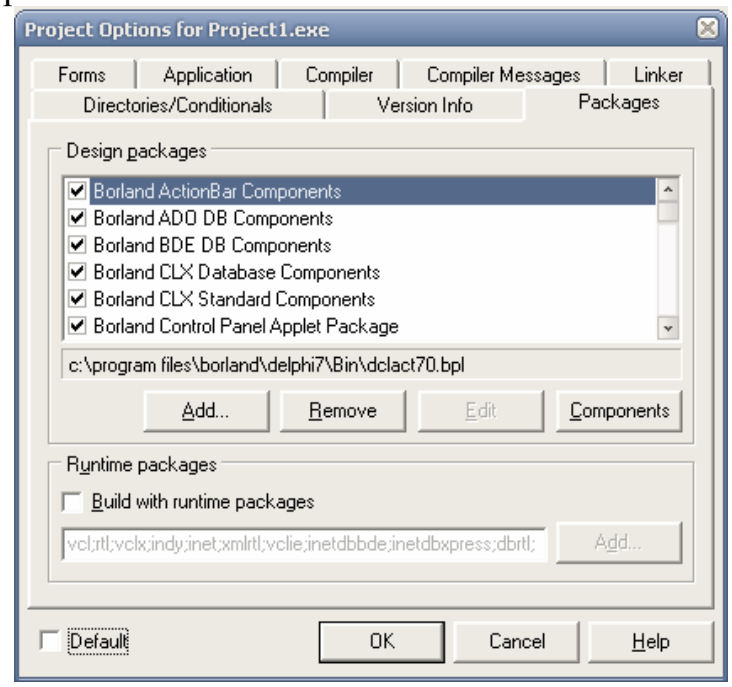


Рис. 26. Вікно пакетів проекту.

ПАКЕТИ

Пакети часу проектування (містять коди виклику об’єктів VCL та редактори їх властивостей)

Пакети часу виконання (містять коди компонентів VCL та модулів Delphi)

Project \ Information For Project

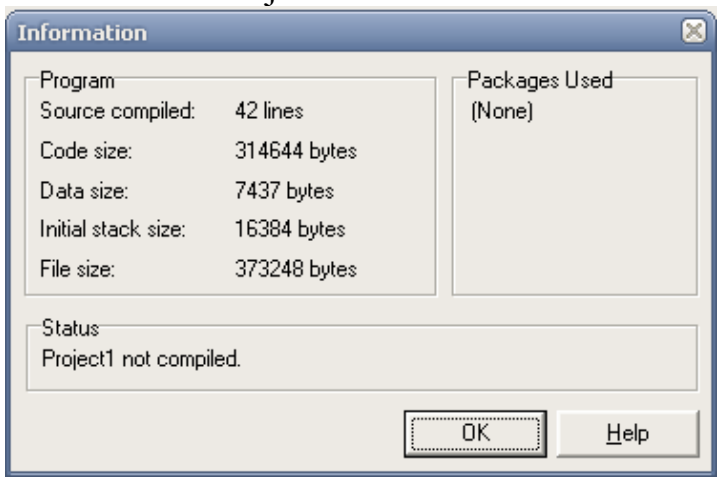


Рис. 27. Інформація про модуль без підтримки пакетів.

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Загальні вимоги до виконання лабораторних робіт

Порядок виконання лабораторних робіт

1. Ознайомитись з темою, метою роботи та засвоїти матеріал теоретичної частини.
2. Користуючись прикладами, які наведені в кожній лабораторній роботі, вивчити методи роботи з компонентами.
3. Вибрати відповідний варіант (за вказівкою вчителя) та ознайомитись із завданням.
4. Розробити проект, що означає виконати наступні етапи:
5. Проектування інтерфейсу проекту.
6. Програмування: здійснити програмування процедур опрацювання подій.
7. Зберегти результати виконаної роботи в індивідуальній робочій папці.
8. Продемонструвати результати виконаної роботи викладачу.
9. Оформити звіт про виконання лабораторної роботи, який має включати: назву теми, завдання, хід роботи, тексти програм, коментарі до них.
10. Захистити звіт про виконання лабораторної роботи та відповісти на контрольні питання, які зазначені на початку лабораторної роботи.

Лабораторна робота №1.

Тема: Знайомство з візуальними середовищами програмування
Мета: Ознайомитись з візуальним середовищем Delphi 6.0.
Програмне забезпечення: Delphi 6.0.
Контрольні питання.

5. Що таке проект?
6. Перерахуйте склад вікна Delphi.
7. Назвіть основні елементи вікна форми.
8. Основні елементи Інспектора об’єктів.
9. Які файли включає Delphi-додаток?
10. Як здійснюється компіляція програми?

Практичне завдання.
Варіант 0-24

1. Запустість візуальне середовище.
2. Ознайомтесь з головним меню та основними командами.

Лабораторна робота №2

Тема: **Ознайомлення з середовищем візуального програмування Delphi. Створення найпростіших програм**

Мета: ознайомитись з середовищем візуального програмування Delphi, ознайомитись з складовими проекту, навчитись створювати найпростішу програму.

Програмне забезпечення: Delphi 7.0

Контрольні питання.

1. Що таке проект?
2. Перерахуйте склад вікна Delphi.
3. Назвіть основні елементи вікна форми.
4. Основні елементи Інспектора об'єктів.
5. Яким чином здійснюється розміщення компонент на форму?
6. Дайте означення події в Delphi.
7. Які файли включає Delphi-додаток?
8. Як здійснюється компіляція програми?

Приклад виконання практичного завдання.

Варіант 0

6. Розробити програму підрахунку суми двох чисел.

Етап проектування інтерфейсу проекту:

При формуванні інтерфейсу проекту використаємо компоненти класу TEdit, TLabel:

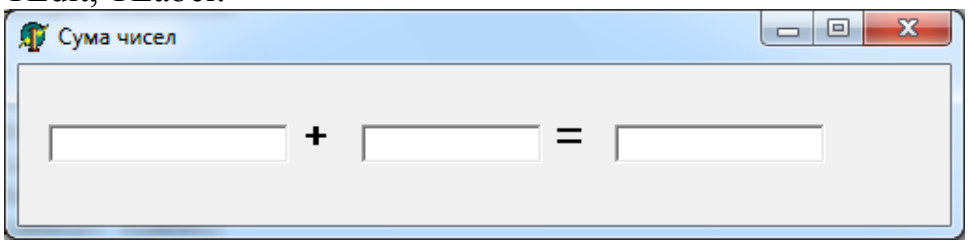


Рис. 28 Орієнтовне зображення проекту

При цьому встановимо такі властивості для об'єктів Form, TEdit, TLabel:

Для об'єкта Form у властивість Caption запишемо «Сума чисел». Для Об'єктів Edit1, Edit2, Edit3 у властивості Text зітремо весь текст.

Для об'єкта Label1 у властивості Caption запишемо «+», Label2 у властивості Caption запишемо «=».

Етап програмування проекту:

При натисканні мишкою на знак «=» маємо отримати результат сумування двох чисел. Обробку події натискання на текстовий надпис «=» здійснює наступна процедура (яку можна викликати двічі клачнувши на потрібному значку «=»):

```
procedure TForm1.Label2Click(Sender: TObject);
begin
 edit3.Text:=FloatToStr(StrToFloat(edit1.Text)+StrToFloat(edit2.Text));
{FloatToStr – команда, яка переводить дійсне число у текстовий рядок}
```

{StrToFloat – команда, яка переводить тектосвий рядок у дійсне число}
end;

Результат виконання проекту:

Реузльтат виконання проекту подамо у вигляді малюнку:

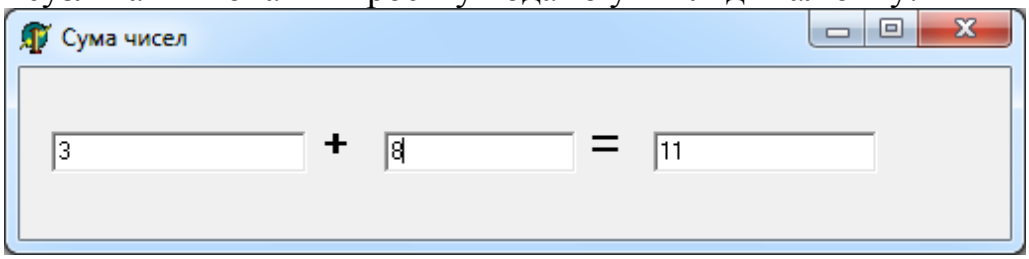


Рис. 29. Орієнтовне зображення результату виконання проекту

Практичне завдання по варіантам.

В середовищі програмування Delphi розробити проект для виконання нижче наведених варіантів завдань. Розробити зручний інтерфейс, забезпечити правильність введення даних. При подвійному натисканні миші реалізувати очищення полів для введення даних.

Варіанти індивідуальних завдань:

Варіант 1.

1. Написати програму для знаходження площі рівностороннього трикутника, якщо відомий його периметр.

Варіант 2.

1. Дано три числа a, b, c, що задають сторони трикутника. Написати програму для визначення площі трикутника використовуючи формулу Герона.

Варіант 3.

1. Написати програму для підрахунку швидкості, з якою спортсмен пробігає дистанцію, якщо задано дистанцію та час витрачений на біг.

Варіант 4.

1. Капітал в сумі 1000 грн. внесений на депозит до банку під 4,5 відсотків річних. Написати програму, яка б розраховувала накопичення капіталу через рік.

Варіант 5.

1. Визначити площу рівнобедреного трикутника, якщо відоо бічну сторону та основу.

Варіант 6.

1. Написати програму для обчислення площі паралелограма, якщо задано його сторони та гострий кут між ними.

Варіант 7.

2. Написати програму для знаходження площі ромба за даними діагоналями.

Варіант 8.

1. Написати програму для знаходження площі рівнобедреного прямокутного трикутника, якщо відома його гіпотенуза.

Варіант 9.

1. Відома довжина кола. Знайти площу круга, обмеженого цим колом.

Варіант 10.

1. Написати програму знаходження об’єму піраміди, якщо задані площа основи та висота піраміди.

Варіант 11.

1. Написати програму для знаходження площі сфери за заданим радіусом.

Варіант 12.

1. Дано три сторони трикутника. Визначити висоти трикутника.

Варіант 13.

1. Знаючи площу прямокутного трикутника і один катет, знайти периметр даного трикутника.

Варіант 14.

1. Написати програму, яка б розраховувала вартість купівлі із врахуванням знижки. Вхідними даними мають бути: кількість продукції, вартість за одиницю продукції, знижка.

Варіант 15.

1. Написати програму для обчислення вартості поїздки, якщо задано відстань, вартість палива та споживання палива (літрів/100 км).

Варіант 16.

1. Скласти програму знаходження площі прямокутного трикутника, якщо відомі два його катети.

Варіант 17.

1. Написати програму для обчислення площі прямокутника, якщо задано його сторони..

Варіант 18.

1. Написати програму для знаходження площі трапеції за заданими основами та висотою.

Варіант 19.

1. Дано сторони рівнобедреного трикутника. Визначити висоту трикутника, опущену на основу.

Варіант 20.

1. Написати програму для знаходження площі круга з даним діаметром.

Варіант 21.

1. Написати програму для визначення оптимальної ваги людини, якщо задано її зріст.

Варіант 22.

1. Написати програму для знаходження катету b прямокутного трикутника, якщо відомий інший катет a та гіпотенуза c.

Варіант 23.

1. Написати програму для знаходження об’єму кулі, якщо відомо її радіус.

Варіант 24.

1. Написати програму, яка б дозволяла проводити операції над двома числами (міні калькулятор): додавання, віднімання, множення і ділення.

Лабораторна робота №3

Тема: **Розробка лінійних програм та програм з розгалуженням**

Мета: навчитись працювати з основними візуальними компонентами палітри ***Standart***: ***TEdit, TButton, TcheckBox, Tlabel, TRadioGroup, TRadioButton*** та палітри ***Additional: TShape, TStaticText,TBitBtn, TLabeledEdit***. Розгялнути та вивчити властивості перелічених об'ктів та методи роботи з ними.

Програмне забезпечення*:* Delphi 7.0

Контрольні питання.

1. Назвіть основні властивості об'єкта *TEdit.*
2. Назвіть основні властивості об'єкта *TButton.*
3. Назвіть основні властивості об'єкта *TcheckBox.*
4. Назвіть основні властивості об'єкта *Tlabel.*
5. Назвіть основні властивості об'єкта *TRadioGroup.*
6. Назвіть основні властивості об'єкта *TRadioButton.*
7. Назвіть основні властивості об'єкта *TShape.*
8. Назвіть основні властивості об'єкта *TBitBtn.*
9. Назвіть основні властивості об'єкта *TStaticText.*
10. Назвіть основні властивості об'єкта *TLabeledEdit.*

Приклад виконання практичного завдання.

Варіант 0

1. Розробити програму обчислення значення функції: $y=\cos^2(2x)-\sin x$ при заданому значенні аргументу x. Залежно від результату забезпечити зміну кольору вікна для виведення результату за наступним правилом: якщо y>0, то колір жовтий, якщо y=0 то червоний, і якщо жодна з умов не виконується – блакитний.

Етап проектування інтерфейсу проекту:

При формуванні інтерфейсу проекту використаємо компоненти TLabeledEdit, Tlabel, TImage, TButton.

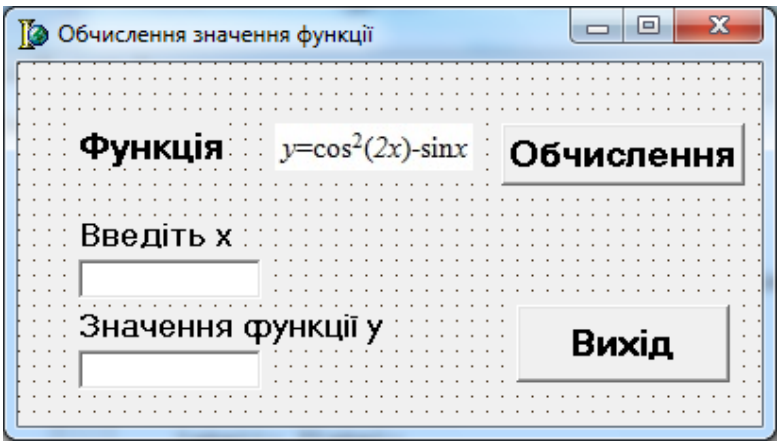


Рис. 30. Орієнтовне зображення проекту

При цьому встановимо такі властивості для об'єктів Button, TLabeledEdit, TImage, TLabel:

Для об'єкта Form у властивість Caption запишемо «Обчислення значення функції».

Для об'єкта Label1 у властивості Caption запишемо «Функція».

Для об'єкта LabeledEdit1 у властивості EditLabel вибираємо підвластивість Caption і записуємо «Введіть x».

Для об'єкта LabeledEdit2 у властивості EditLabel вибираємо підвластивість Caption і записуємо «Значення функції y».

Для об'єкта Button1 у властивості Caption запишемо «Обчислення».

Для об'єкта Button2 у властивості Caption запишемо «Вихід».

Для об'єкта Image1 у властивості Picturevибираємо файл, який наперед піготували з написаною формулою.

Етап програмування проекту:

При натисканні мишкою на кнопку «Обчислення» маємо отримати результат. Обробку події натискання на задану кнопку здійснює наступна процедура (яку можна викликати двічі клачнувши на потрібній кнопці):

```
procedure TForm1.Button1Click(Sender: TObject);
var x, y : real;
    st : string;
begin
    x:=StrToFloat(LabeledEdit1.Text);
    y:=sqr(cos(2*x))-sin(x);
    if y>0 then labelededit2.Color:=clyellow
    else if y=0 then labelededit2.Color:=clred
    else labelededit2.Color:=clblue;
    str(y:5:3,st);
    labelededit2.Text:=st;
end;
```

При натисканні мишкою на кнопку «Вихід» програма має завершувати свою роботу. Обробку події натискання на задану кнопку

здійснює наступна процедура (яку можна викликати двічі клачнувши на потрібній кнопці):

```
procedure TForm1.Button2Click(Sender: TObject);
begin
 close;
end;
```

Результат виконання проекту:

Результат виконання проекту подамо у вигляді малюнку:

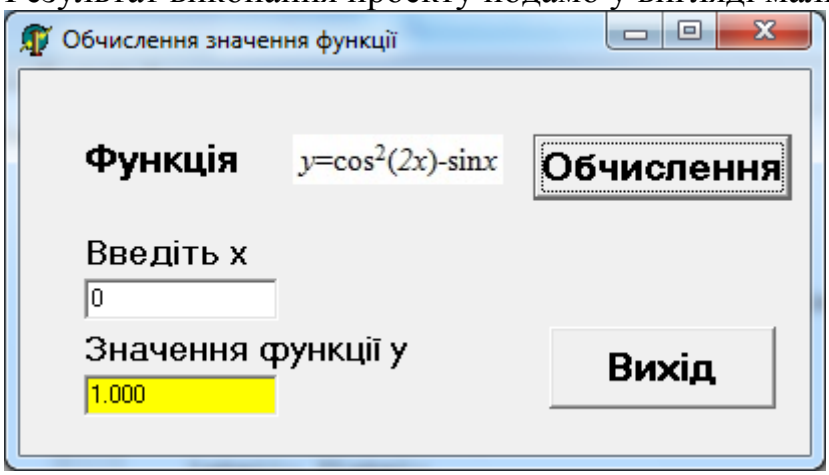


Рис. 31. Орієнтовне зображення результату виконання проекту

Практичне завдання по варіантам.

В середовищі програмування Delphi розробити програму обчислення значення функції при заданому значенні аргументу.

Вимоги до проекту:

- Значення змінної вводиться на формі (використати при введенні компоненту ***TEdit*** або ***TLabeledEdit***)
- Значення функції виводиться на формі (використати при виведенні компоненту ***TEdit*** або ***TLabeledEdit***)
- Для текстових підписів використатити компоненту ***TLabel***
- Обчислення функції має відбуватись при натисканні кнопки миші на командній кнопці
- Залежно від результату забезпечити зміну кольору вікна для виведення результату за наступним правилом: якщо y>0, то колір жовтий, якщо y=0 то червоний, і якщо жодна з умов не виконується – блакитний.
- Забезпечити правильність введення даних
- При подвійному натисканні мишею на формі забезпечити очищення ***полів для введення та виведення даних.***

Варіанти індивідуальних завдань:

Варіант 1.

1. $y=cos^3(2x)-log_2(x^2+2)$

Варіант 2.

1. $y=cos^2(2x)-sinx$

Варіант 3.

1. $y= sin^3(3x)-cosx$

Варіант 4.

1. $y = \sqrt{x^2 + 3} - 2\log_2(x^2 + 1)$

Варіант 5.

1. $y = \sqrt{x^2 + 3x} + 2\sin x$

Варіант 6.

1. $y = \sqrt[5]{x^4 + 3x^2} - 2\sin x$

Варіант 7.

1. $y = (x^2 - 2x)^2 - 4\log_2(x^4 + 1)$

Варіант 8.

1. $y = 2^{2x} - 4\cos x$

Варіант 9.

1. $y = \sin(3x) + \cos x$

Варіант 10.

1. $y = \sqrt[3]{x^2 + 3x^2} + 2\sin x$

Варіант 11.

1. $y = \sqrt[5]{x^5 + 3x^4} - 4x^{\frac{2}{3}}$

Варіант 12.

1. $y = \sqrt{\ln^2(\sin x + 2) + x^2}$

Варіант 13.

1. $y = 2^{\cos x} + \sin x\sqrt{1 + 3ctgx}$

Варіант 14.

1. $y=cos^3(2x)-log_2(x^2+2)$

Варіант 15.

1. $y=cos^2(2x)-sinx$

Варіант 16.

1. $y= sin^3(3x)-cosx$

Варіант 17.

1. $y = \sqrt{x^2 + 3} - 2\log_2(x^2 + 1)$

Варіант 18.

1. $y = \sqrt{x^2 + 3x} + 2\sin x$

Варіант 19.

1. $y = \sqrt[5]{x^4 + 3x^2} - 2\sin x$

Варіант 20.

1. $y = (x^2 - 2x)^2 - 4\log_2(x^4 + 1)$

Варіант 21.

1. $y = 2^{2x} - 4\cos x$

Варіант 22.

1. $y = \sin(3x) + \cos x$

Варіант 23.

1. $y = \sqrt[3]{x^2 + 3x^2} + 2\sin x$

Варіант 24.

1. $y = \sqrt[5]{x^5 + 5x^4} - 4x^{\frac{2}{3}}$

Лабораторна робота №4

Тема: **Розробка програм з розгалуженням**

Мета: навчитись працювати з основними візуальними компонентами палітри ***Standart***: ***TEdit, TButton, TcheckBox, Tlabel, TRadioGroup, TRadioButton*** та палітри ***Additional: TShape, TStaticText,TBitBtn, TLabeledEdit***. Розгялнути та вивчити властивості перелічених об'ктів та методи роботи з ними.

Програмне забезпечення*:* Delphi 7.0

Контрольні питання.

1. Назвіть основні властивості об'єкта *TEdit* та методи роботи з ними.
2. Назвіть основні властивості об'єкта *TButton* та методи роботи з ними.
3. Назвіть основні властивості об'єкта *TcheckBox* та методи роботи з ними.
4. Назвіть основні властивості об'єкта *Tlabel* та методи роботи з ними.
5. Назвіть основні властивості об'єкта *TShape* та методи роботи з ними.
6. Назвіть основні властивості об'єкта *TBitBtn* та методи роботи з ними.
1. Назвіть основні властивості об'єкта *TLabeledEdit* та методи роботи з ними.

Приклад виконання практичного завдання.

Варіант 0

1. В средовищі програмування Delphi розробити програму для визначення, чи є число парним, чи непарним. Для введення числа використаи компоненту ***TEdit.*** Якщо число є парним, то намалювати червоний квадрат, якщо непарне, то намалювати жовте коло.

Етап проектування інтерфейсу проекту:

Під час проектування використаємо компоненти класів ***TEdit***, ***Tlabel***, ***TShape***, ***TButton***.

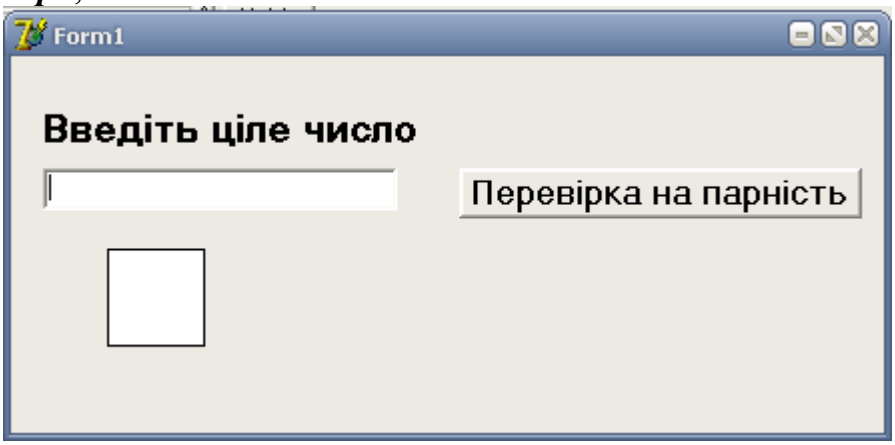


Рис. 32. Орієнтовне зображення проекту

При цьому встановимо такі властивості для об'єктів ***TButton***, ***Tlabel***, ***TShape***:

Для об'єкта ***Label1*** у властивості ***Caption*** запишемо «Введіть ціле число».

Для об'єкта ***Label2*** у властивості ***Caption*** запишемо « ».

Для об'єкта ***Button1*** у властивості ***Caption*** запишемо «Перевірка на парність».

Для об'єкта ***TShape*** у властивості ***Shape*** вибираємо значення «stSquare».

Етап програмування проекту:

Процедура обробки події, яка виникає при натисканні на кнопку «***Перевірка на парність***» матиме вигляд:

```
procedure TForm1.Button1Click(Sender: TObject);
var x : integer;
begin
      if Edit1.Text='' then Showmessage('Введіть число') else
      begin
       Shape1.Visible:=true;
          x:=StrToInt(Edit1.Text);
          if x mod 2=0 then
          begin
                Shape1.Shape:=stsquare;
                Shape1.Brush.Color:=clred;
                Label2.Caption:='Число парне'
           end
          else
          begin
           Shape1.Shape:=stcircle;
           Shape1.Brush.Color:=clyellow;
                Label2.Caption:='Число непарне'
          end
       end
end;
```

При введенні нового значення наступна процедура обробки події для компоненти ***Edit1*** забезпечує знищення попередніх результатів:

```
procedure TForm1.Edit1Change(Sender: TObject);
begin
 shape1.Visible:=false;
 label2.caption:='';
end;
```

Результат виконання проекту:

Результат виконання проекту подамо у вигляді малюнку:

1) Вікно форми виконання проекту, якщо введене число парне:

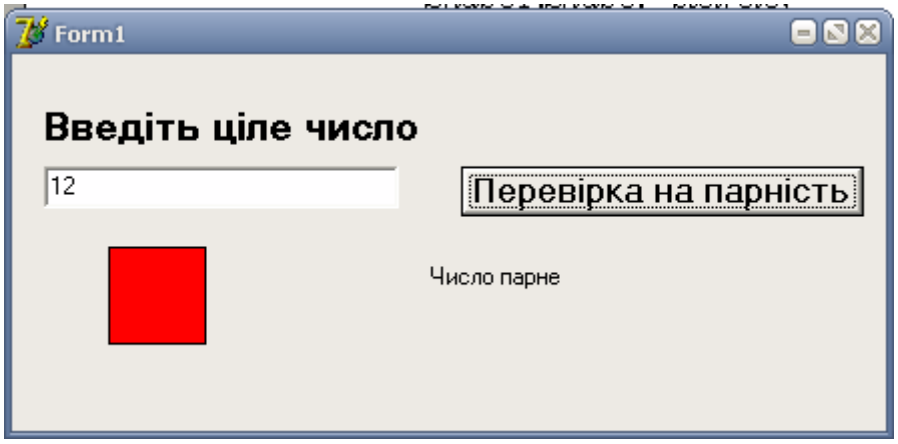


Рис. 33. Орієнтовне зображення результату виконання проекту

2) Вікно форми виконання проекту, якщо введене число непарне:

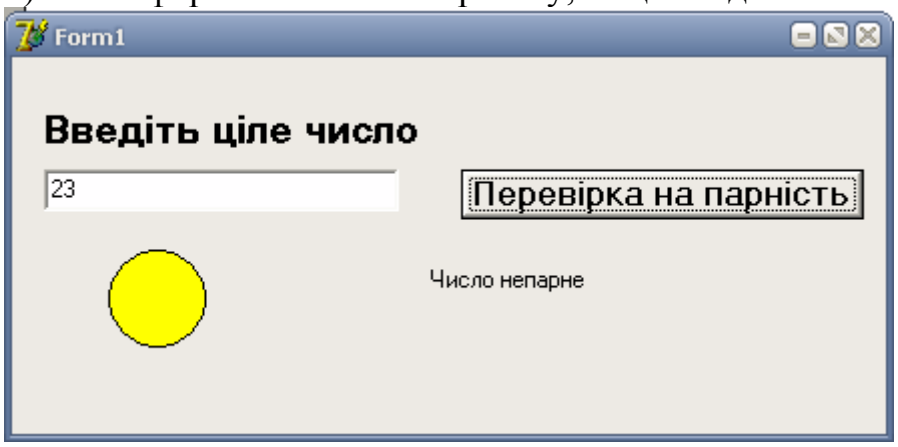


Рис. 34. Орієнтовне зображення результату виконання проекту

Практичне завдання по варіантам.

В середовищі програмування Delphi розробити програму обчислення значення арифметичного виразу.

Вимоги до проекту:

- Значення змінної вводиться на формі (використати при введенні компоненту ***TEdit*** або ***TLabeledEdit***)
- Значення арифметичного виразу виводиться на формі (використати при виведенні компоненту ***TEdit*** або ***TLabeledEdit***)
- Для текстових підписів використатити компоненту ***TLabel***
- Обчислення арифметичного виразу має відбуватись при натисканні мишею на командній кнопці (використати компоненту ***Tbutton*** або ***TBitBtn***)
- Забезпечити правильність введення даних;

2. На формі має бути група залежних перемикачів (використати компоненту ***TRadioGroup***), яка б забезпечувала зміну кольору компоненти ***TEdit***, в якій виводиться значення арифметичного виразу;

- На формі має бути незалежний перемикач (використати компоненту ***TCheckBox***), якщо він увімкнений, то всі аргументи беруться по модулю;
- Передбачити вивід відповідного повідомлення, якщо введені значення аргументів виходять за межі області визначення арифметичного виразу, або якщо зроблена спроба обчислити значення арифметичного виразу при неповних даних.

Варіанти індивідуальних завдань:

(значення змінних a та b вводяться при запуску програми)

Варіант 1.

1. $z = \dfrac{|x-1| + e^{-y}}{12.34 - \lg\sqrt{|x|}}$, *якщо* $y = 2a\sqrt[3]{a+b}$, $x = arcctg\dfrac{e^a + e^{\frac{1}{b}}}{\sqrt{a+e}}$

Варіант 2.

1. $m = \lg^2|y-5.5| + \sin^2\dfrac{y}{4}$, *якщо* $y = \ln|\pi - x| + \lg\left|\dfrac{\pi}{x}\right|$, $x = \sqrt{a^2+b}$

Варіант 3.

1. $z = arctg\left(\sin^2 x + tg^3 y\right)$, *якщо* $y = \sin^2(a-b)^3$, $x = \ln|a+2.3| - \lg|b-3.2|$

Варіант 4.

1. $z = \sqrt{\left|\sin^3(x-1) + \cos y\right|}$, *якщо* $x = \log_b|a+1| + tgb$, $y = a+b$;

Варіант 5.

1. $z = \sqrt[3]{\left|\ln|x|^{-1} - \sqrt{|x+1|}\right|}$, *якщо* $x = \dfrac{\arcsin a^{-1} + \ln|b|}{(-2)e^{-a}}$

Варіант 6.

2. $z = \sqrt[3]{|\pi - y|} + \sin^2 \pi x + 1.67$, *якщо* $y = tg^4(b-1)^2 - 0.035$, $x = ctg\dfrac{a-1}{e} + 2^{\frac{a+1}{2}}$

Варіант 7.

1. $z = \ln\left|\dfrac{x\sqrt{x} + \cos^3 y^2}{1.604 - arcctgy}\right|^{2,1}$, *якщо* $x = \log_{|a|} b + \dfrac{(a-b)^2}{b}\cdot e^{-a}$, $y = \sqrt[3]{\left|\cos a^2 + ab + 0.06\right|}$

Варіант 8.

2. $z = b\cdot e^{-2,5x+y} - \sqrt[3]{bx}$, *якщо* $x = \dfrac{\lg|a+b|}{arcctg\dfrac{\pi}{a}} + 0.17$, $y = \dfrac{\sin^2\dfrac{a^3}{2} - ctg\dfrac{b}{4}}{\ln|a| + \ln b^2}$.

Варіант 9.

2. $z = \dfrac{e^{-xy} + 17.4}{\sqrt[3]{\sin^2 xy}}$, *якщо* $x = \left(a^2+b^2\right)^{-4,1}$, $y = arctg^3\dfrac{1}{b}$.

Варіант 10.

2. $z = \arccos\left(\frac{x^2}{0.13}\right)^{-1} + \ln\left|y^{-1}\right|$, *якщо* $x = \sqrt{(a+6.1)^3}$, $y = \ln a^4 + \lg b^{-6}$

Варіант 11.

1. $z = \ln|x-y| + \cos^3 xy$, *якщо* $x = \sqrt{|x+a|} + 17.14$, $y = a \cdot \sqrt[3]{\sin^4 x^3} + 12.47$

Варіант 12.

1. $z = \frac{x^2 - y^2}{\lg|x-7|}$, *якщо* $x = \frac{\sin^2 a^3 - \arcsin\frac{1}{b}}{\ln|a+b|-1}$, $y = \sqrt{\left|\frac{a+b}{ab}\right| + \pi}$

Варіант 13.

1. $z = -\sqrt{|\lg x - \ln y| + 1.31}$, *якщо* $x = \frac{e^{-2,5a} + \sin^2 a^3}{2\lg|ba|}$,

$y = \frac{arctg^3(b-a) + \sqrt[3]{ab}}{a + \log_b a}$

Варіант 14.

1. $z = \ln\left|\frac{x\sqrt{x} + \cos^3 y^2}{1.604 - arcctgy}\right|^{2,1}$, *якщо* $x = \log_{|a|} b + \frac{(a-b)^2}{b} \cdot e^{-a}$,

$y = \sqrt[3]{\left|\cos a^2 + ab + 0.06\right|}$

Варіант 15.

1. $m = \lg^2|y-5.5| + \sin^2\frac{y}{4}$, *якщо* $y = \ln|\pi - x| + \lg\left|\frac{\pi}{x}\right|$, $x = \sqrt{a^2+b}$

Варіант 16.

2. $z = arctg\left(\sin^2 x + tg^3 y\right)$, *якщо* $y = \sin^2(a-b)^3$, $x = \ln|a+2.3| - \lg|b-3.2|$

Варіант 17.

2. $z = b \cdot e^{-2,5x+y} - \sqrt[3]{bx}$, *якщо* $x = \frac{\lg|a+b|}{arcctg\frac{\pi}{a}} + 0.17$, $y = \frac{\sin^2\frac{a^3}{2} - ctg\frac{b}{4}}{\ln|a| + \ln b^2}$.

Варіант 18.

2. $z = \frac{|x-1| + e^{-y}}{12.34 - \lg\sqrt{|x|}}$, *якщо* $y = 2a\sqrt[3]{a+b}$, $x = arcctg\frac{e^a + e^{\frac{1}{b}}}{\sqrt{a+e}}$

Варіант 19.

1. $z = \frac{e^{-xy} + 17.4}{\sqrt[3]{\sin^2 xy}}$, *якщо* $x = \left(a^2 + b^2\right)^{-4,1}$, $y = arctg^3\frac{1}{b}$.

Варіант 20.

2. $z = \sqrt[3]{|\pi - y|} + \sin^2 \pi x + 1.67$, *якщо* $y = tg^4(b-1)^2 - 0.035$, $x = ctg\dfrac{a-1}{e} + 2^{\frac{a+1}{2}}$

Варіант 21.

1. $z = \arccos\left(\dfrac{x^2}{0.13}\right)^{-1} + \ln|y^{-1}|$, *якщо* $x = \sqrt{(a+6.1)^3}$, $y = \ln a^4 + \lg b^{-6}$

Варіант 22.

1. $z = \sqrt[3]{\left|\ln|x|^{-1} - \sqrt{|x+1|}\right|}$, *якщо* $x = \dfrac{\arcsin a^{-1} + \ln|b|}{(-2)e^{-a}}$

Варіант 23.

1. $z = \dfrac{x^2 - y^2}{\lg|x-7|}$, *якщо* $x = \dfrac{\sin^2 a^3 - \arcsin\dfrac{1}{b}}{\ln|a+b|-1}$, $y = \sqrt{\left|\dfrac{a+b}{ab}\right| + \pi}$

Варіант 24.

2. $z = \sqrt{|\sin^3(x-1) + \cos y|}$, *якщо* $x = \log_b|a+1| + tgb$, $y = a + b$;

Лабораторна робота №5

Тема: **Розробка власного міні-проекта.**

Мета: навчитись працювати з основними візуальними компонентами палітри ***Standart***: ***TEdit, TButton, TcheckBox, Tlabel, TRadioGroup, TRadioButton*** та палітри ***Additional: TShape, TStaticText,TBitBtn, TLabeledEdit***. Розгялнути та вивчити властивості перелічених об'ктів та методи роботи з ними.

Програмне забезпечення*:* Delphi 7.0

Контрольні питання.

1. Що таке проект?
2. Перерахуйте склад вікна Delphi.
3. Назвіть основні елементи вікна форми.
4. Основні елементи Інспектора об'єктів.
5. Яким чином здійснюється розміщення компонент на форму?
6. Які файли включає Delphi-додаток?
7. Як здійснюється компіляція програми?
8. Що являє собою вікно редактора коду?
9. Перерахувати способи створення нового вікна форми.
10. Перерахувати способи переходу з вікна форми у вікно Редактора коду.
11. Які можливості підтримує вікно Редактора коду?

1. Назвіть основні елементи Інспектора об'єктів.

Практичне завдання по варіантам.

В середовищі програмування Delphi розробити віконні додатки із використанням основних компонент палітр Standard та Additional. Розробити зручний інтерфейс та забезпечити правильність введення даних.

Варіанти індивідуальних завдань:

Варіант 1.

1. На формі розміщені три фігури: круг, квадрат та трикутник. Якщо натискаємо мишкою на круг, то трикутник змінює свій колір, а квадрат – ні. Якщо на квадрат – то всі фігури змінюють свій колір. При натисканні на трикутник – круг і квадрат змінюють свій колір.

Варіант 2.

1. На формі розміщено три круги: червоний, жовтий і зелений (вертикально). При натисканні мишкою на круги має з'являтися відповідний надпис: "стій", "зачекай", "іди".

Варіант 3.

1. Організувати малювання на формі фігур (коло, квадрат і ін.), зафарбованих певним кольором за вибором користувача. Для вибору фігур та кольору використати компоненту TRadioGroup, для відображення фігур - компоненту TShape.

Варіант 4.

1. Вводиться послідовність чисел через кому, яка закінчується крапкою. На формі встановлено незалежний перемикач. Якщо він ввімкнений, то шукаються кількість тільки парних чисел, якщо ні – непарних.

Варіант 5.

1. Вводиться послідовність чисел через кому. Визначити довжину найдовшої підпослідовності однакових чисел. Якщо довжина менша або рівна 2 на формі з'являється червоний круг.

Варіант 6.

1. На формі вводяться три числа (компонента TEdit). Якщо всі три числа рівні, то змінити колір компоненти TEdit на зелений, в протилежному випадку – на червоний. Перевірку на рівність здійснювати натискання командної кнопки (компонент TButton).

Варіант 7.

1. На формі вводяться п'ять чисел. Якщо більшість з них парні, то вивести на форму зелений квадрат, в протилежному випадку – червоний.

Варіант 8.

1. На формі розміщені 3 фігури: круг, квадрат і трикутник. Написати програму, яка б змінювала колір будь-яких двох при натисканні на будь-яку третю фігуру.

Варіант 9.

1. На форму вводиться ціле число. Підрахувати кількість цифр в ньому і вивести її на форму. Якщо дана кількість парна – то змінити колір фону форми на жовтий, в протилежному випадку – на зелений.

Варіант 10.

1. На форму вводиться п'ять чисел. Якщо дана послідовність є строго спадною, то змінити колір фону форми на салатовий, в протилежному випадку ніяких дій не відбувається.

Варіант 11.

1. На формі вводяться два числа. Якщо остача від ділення першого на друге дорівнює 1, то на формі відобразити червоне коло, якщо дорівнює 2 – синій квадрат. У всіх інших випадках ніяких дій не відбувається.

Варіант 12.

1. На формі вводяться чотири додатних цілих числа. Користувач випадково клацає мишкою на будь-якому з них. Якщо це число є середнім арифметичним інших трьох, то з'являється зелений трикутник, якщо більше за середнє арифметичне – то червоне коло, якщо менше – то жовтий квадрат, сторона якого дорівнює сумі всіх чотирьох чисел.

Варіант 13.

1. Вводиться три числа. Визначити, чи утворюють дані числа арифметичну прогресію. За позитивність чи негативність відповіді відповідає круг розміщений на формі (компонента TShape). Якщо так – круг набуває зеленого кольору, ні - червоного.

Варіант 14.

1. Написати програму для визначення, чи існує такий трикутник ABC із заданими сторонами. Значення сторін AB, AC, BC задаються із використання компонент TEdit або TLabeledEdit. Якщо трикутник існує, то повинно з'явитися діалогове вікно із відповідним текстом, в протилежному випадку відбувається закриття форми.

Варіант 15.

1. Розробити програму, яка дозволяє змінювати колір форми та колір шрифтів. Для цього розмістити на формі дві групи незалежних перемикачів, які відповідатимуть за зміну кольору та зміну шрифту.

Варіант 16.

1. На формі вводиться три числа. Визначити, чи утворюють дані числа геометричну прогресію. За позитивність чи негативність відповіді відповідає круг розміщений на формі (компонент TShape). Якщо так – круг набуває жовтого кольору, ні - синього.

Варіант 17.

1. На формі вводиться п'ять чисел. Визначити, чи утворюють дані числа послідовність Фібоначчі. За позитивність чи негативність відповіді відповідає квадрат на формі (компонента TShape). Якщо так – квадрат набуває зеленого кольору, ні – червоного.

Варіант 18.

1. На формі вводиться число. Якщо введене число більше нуля, то відбувається закриття додатку, якщо нуль – виводиться повідомлення про повторне введення числа. У випадку, коли число більше одиниці, форма зафарбовується у блакитний колір.

Варіант 19.

1. На формі вводяться три числа (компонента - TEdit). Змінити колір компоненти, в якій знаходиться найбільше число, на зелений, середнє число – на жовтий, найменше число – на червоний колір. У випаду рівності будь-яких чисел – змінити на синій колір. Перевірка здійснюється після натискання командної кнопки (компонента TButton).

Варіант 20.

1. На формі вводяться п'ять чисел. Якщо більшість з них непарні, то вивести на форму зелений круг, в протилежному випадку – червоний круг.

Варіант 21.

1. На формі розміщений круг. Він змінює своє забарвлення наступним чином: введене число є простим – зелений колір, інший варіант – червоний. Якщо ввімкнений незалежний перемикач, то відбувається перевірка на парність та непарність введеного числа. Про це повідомляє відповідний напис на формі.

Варіант 22.

- На форму вводиться ціле число. Перевірити чи є воно симетричним (яке читається зліва направо і справа наліво однаково). Якщо так-то вивести зелений круг, в протилежному випадку – синій. Також підрахувати кількість цифр у ньому.

Варіант 23.

1. На форму вводиться п'ять чисел. Визначити чи дана послідовність є строго зростаючою. Якщо так, то на формі з'являється круг, зафарбований в зелений колір, якщо ні – то квадрат червоного кольору.

Варіант 24.

1. На формі задається натуральне число. Знайти середнє арифметичне цифр у записі цього числа. Якщо отриманий результат більше 5, то зобразити круг червоного кольору, в протилежному випадку – зеленого кольору.

Лабораторна робота №6

Тема: **Розробка циклічних програм (цикли з передумовою та післяумовою).**

Мета: навчитись використовувати візуальні компоненти сторінки палітри ***Samples***: ***TSpinEdit***, сторінки ***Standard***: ***TEdit***, ***TButton***, ***TLabel***, Tmemo, сторінки ***Additional***: ***TBitBtn***, ***TLabeledEdit***, ***TStringGrid*** та сторінки ***Win 32***: ***TRichEdit***, ***TUpDown;*** вивчити основні властивості та методи роботи з компонентами панелі ***Additional***, ***Standard*** та ***Samples***.

Програмне забезпечення*:* Delphi 7.0

Контрольні питання.

1. Перерахуйте типи змінних, які використані в даній роботі.
2. Назвіть стандартні арифметичні функції, які використані проекті.
3. Оператори циклів, їх призначення.

4. Дайте означення проекту в Delphi.
5. Подія-це....
6. Які існують процедури обробки подій?

Приклад виконання практичного завдання.

Варіант 0

1. В средовищі програмування Delphi розробити віконний додаток для табулювання значення функції $y=\sqrt{x^3+x+1}$ на проміжку $[a,b]$ із кроком h, що задається користувачем.

Етап проектування інтерфейсу проекту:

Під час проектування використаємо компоненти класів ***TLabelEdit***, ***Tlabel***, ***TButton, TMemo***.

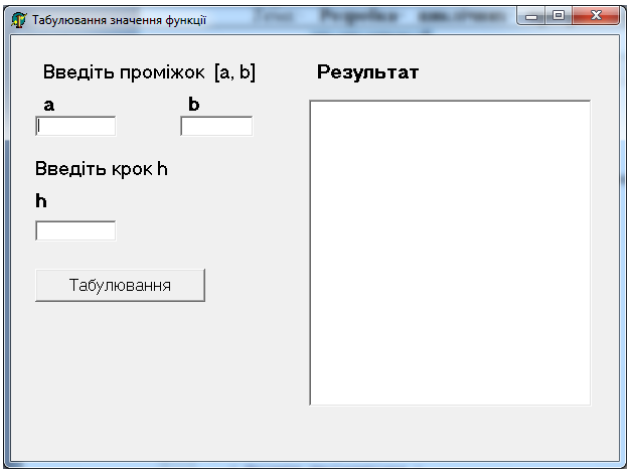


Рис. 35. Орієнтовне зображення проекту

Етап програмування проекту:

Процедура обробки події ***OnClick*** матиме вигляд:

```
procedure TForm1.Button1Click(Sender: TObject);
var x,y,a,b,h:real;
begin
 a:=StrToFloat(labelededit1.Text);
 b:=StrToFloat(labelededit2.Text);
 h:=StrToFloat(labelededit3.Text);
 memo1.Lines.Add('x y');
 x:=a;
 while x<=b do
 begin
 y:=sqrt(x*x*x+x+1);
 memo1.lines.Add(floattostrf(x,fffixed,7,2)+' '+floattostrf(y,fffixed,7,2));
 x:=x+h;
 end;
end;
```

Результат виконання проекту:

На проміжку [1; 2] з кроком 0,09 матимемо наступні результати:

Результат виконання проекту подамо у вигляді малюнку:

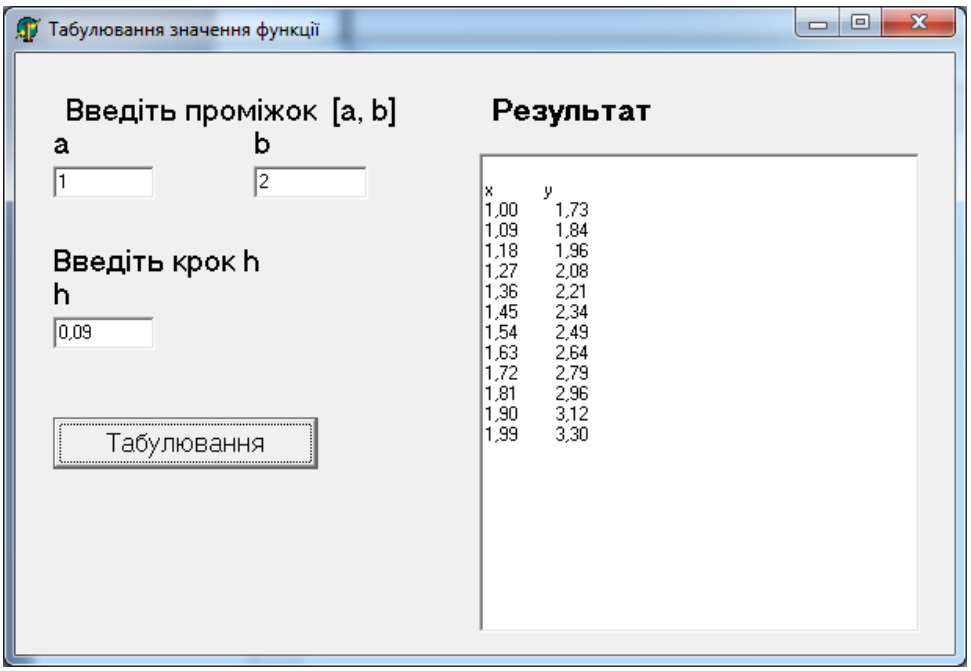


Рис. 36 Орієнтовне зображення результату виконання проекту

Практичне завдання по варіантам.

В середовищі програмування Delphi розробити програму обчислення значення функції на вказаному проміжку із заданим кроком для відповідного аргументу функції (провести табулювання функції)..

Вимоги до проекту:

1. Значення *a, b, h*, вводяться на формі (використати при введенні компоненти ***Tlabel, Tedit, TLabeledEdit***)
2. Результати табулювання функції (значення функції) виводяться на форму (використати компоненти ***Tmemo*** або ***TRichEdit***)
3. Реалізувати контроль наявності факту введення даних;
4. Забезпечити конроль правильності введення числових даних для меж табулювання та кроку
5. Реалізувати контроль за межами табуляції (a<b).

Варіанти індивідуальних завдань:

(значення змінних a та b, *h*, вводяться при запуску програми)

Варіант 1.

1. $y = \dfrac{\cos^3 x^2}{1.5x + 2}.$

Варіант 2.

1. $y = \dfrac{2x + 8}{|\cos 3x| + 1}.$

Варіант 3.

1. $y = \dfrac{6x + 4}{\sin 3x - x}.$

Варіант 4.

2. $y = \dfrac{x + \cos 2x}{3x}.$

Варіант 5.

6. $y = \dfrac{x^3 + 2x}{3\cos\sqrt{x} + 1}.$

Варіант 6.

2. $y = \dfrac{\ln|x + 1| + 5}{2x + 3}.$

Варіант 7.

1. $y = \dfrac{tg0.5x}{x^3 + 7.5}$

Варіант 8.

2. $y = \dfrac{x + \cos 2x}{x + 2}.$

Варіант 9.

1. $y = \dfrac{e^{2x} - 8}{x + 3}.$

Варіант 10.

1. $y = \dfrac{x - \ln 2x}{3x + 1}.$

Варіант 11.

1. $y = \dfrac{x + \sin 2x}{x^2 - 3}.$

Варіант 12.

1. $y = \dfrac{x^2 + 2x}{\cos 5x + 2}.$

Варіант 13.

1. $y = \frac{(x+2)^2}{\sqrt{x^2+1}}$.

Варіант 14.

1. $y = \frac{\cos^2 x}{x^2+1}$.

Варіант 15.

2. $y = \frac{x^3-2}{3\ln x}$

Варіант 16.

2. $y = \frac{(3x+2)^2}{\sin x+3}$.

2. $y = \frac{tg2x-3x}{x+3}$.

Варіант 18.

2. $y = \frac{3x+1}{arctgx}$.

Варіант 19.

1. $y = \frac{\arccos x}{2x+1}$.

Варіант 20.

1. $y = \frac{2.5x^3}{e^{2x}+2}$.

1. $y = \frac{5\lg x}{x^2-1}$.

Варіант 22.

1. $y = \frac{2\sin^2(x+2)}{x^2+1}$

.

Варіант 23.

1. $y = \frac{5tg(x+7)}{(x+3)^2}$.

Варіант 24.

2. $y = \frac{1.5x-\ln 2x}{3x+1}$.

Варіант 21.

Варіант 17.

Лабораторна робота №7

Тема: **Розробка циклічних програм (цикли з параметром).**
Мета: навчитись використовувати візуальні компоненти сторінки палітри ***Samples***: ***TSpinEdit***, сторінки ***Standard***: ***TEdit***, ***TButton***, ***TLabel***, Tmemo, сторінки ***Additional***: ***TBitBtn***, ***TLabeledEdit***, ***TStringGrid*** та сторінки ***Win 32***: ***TRichEdit***, ***TUpDown;*** вивчити основні властивості та методи роботи з компонентами панелі ***Additional***, ***Standard*** та ***Samples***.

Програмне забезпечення*:* Delphi 7.0

Контрольні питання.

1. Перерахуйте типи змінних, які використані в даній роботі.
2. Назвіть стандартні арифметичні функції, які використані проекті.
3. Оператори циклів, їх призначення.
2. Які існують процедури обробки подій?

Приклад виконання практичного завдання.

Варіант 0

1. В средовищі програмування Delphi розробити віконний додаток для табулювання значення функції $y = \sqrt{x^3+x+1}$ починаючи із заданої точки x_0 , коли заданий крок Δx та кількість кроків n, що задається користувачем (з використанням компоненти ***TStringGrid***).

Етап проектування інтерфейсу проекту:

Під час проектування використаємо компоненти класів ***TEdit***, ***Tlabel***, ***TButton, TStringGrid***.

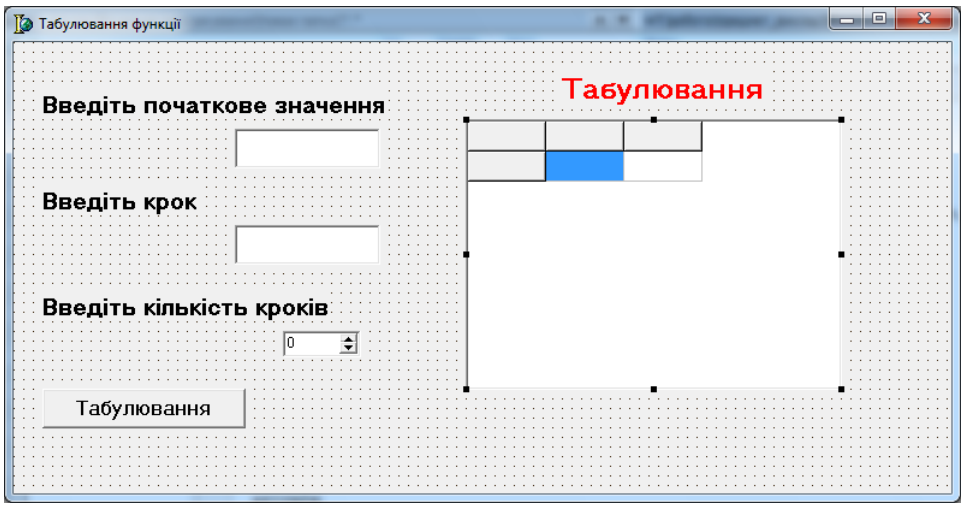


Рис. 37. Орієнтовне зображення проекту

При цьому встановимо такі властивості для об'єктів ***SpinEdit***, ***StringGrid***:

Для об'єкта ***SpinEdit1*** у властивості ***MaxValue*** запишемо «7».

Для об'єкта ***SpinEdit1*** у властивості ***MinValue*** запишемо «1».

Для об'єкта ***StringGrid1*** у властивості ***ColCount*** запишемо «3».

Для об'єкта ***StringGrid1*** у властивості ***RowCount*** запишемо «2».

Для об'єкта ***StringGrid1*** у властивостях ***goFixedVertLine, goFixedHorzLine, goVertLine, goHorzLine, goRangeSelect, goEditing*** встановлюємо значення *true*.

Етап програмування проекту:

Процедура обробки події ***OnClick*** матиме вигляд:

```
procedure TForm1.Button1Click(Sender: TObject);
var x, dx,y :real;
n, i : integer;
begin
 x:=strtofloat(edit1.Text); dx:=strtofloat(edit2.Text);
 n:=Spinedit1.Value;
 stringgrid1.Cells[0,0]:='n'; stringgrid1.Cells[1,0]:='x';
 stringgrid1.Cells[2,0]:='y';
 for i:=1 to n do
 begin
 stringgrid1.RowCount:=i+1;
 y:=sqrt(x*x*x+x+1);
 stringgrid1.Cells[0,i]:=inttostr(i);
 stringgrid1.Cells[1,i]:=floattostr(x);
 stringgrid1.Cells[2,i]:=floattostrf(y,fffixed,7,3);
 x:=x+dx;
 end;
end;
```

Результат виконання проекту:

З початковим значенням 1 та кроком 0,09 матимемо наступні результати:

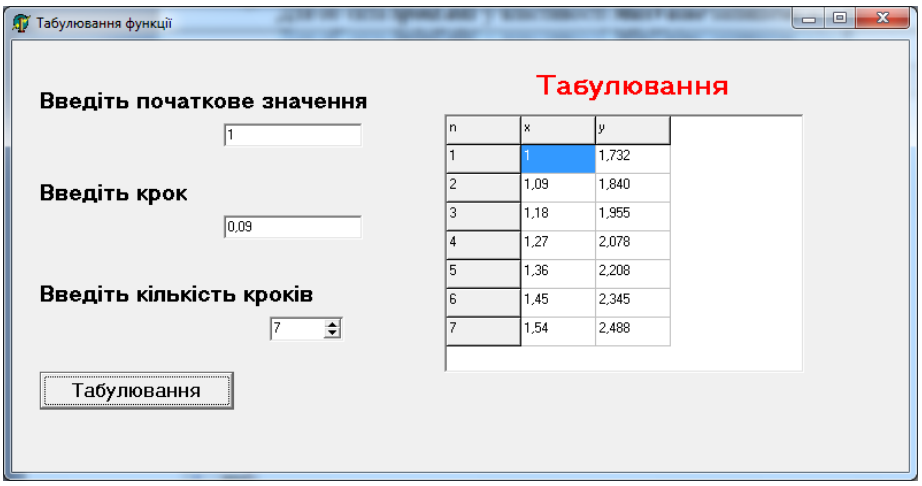


Рис. 38. Орієнтовне зображення результату виконання проекту

Практичне завдання по варіантам.

В середовищі програмування Delphi розробити програму обчислення значення функції починаючи із заданої точки x_0 , коли заданий крок Δx та кількість кроків n, для відповідного аргументу функції (провести табулювання функції). Серед значень функції знайти мінімальне, максимальне та середнє арифметичне значення.

Вимоги до проекту:

1. Значення x_0 *та* Δx вводяться на формі (використати при введенні компоненти ***Tlabel, Tedit, TLabeledEdit***)
2. Значення n (кількість кроків) вводиться на формі з допомогою компоненти ***SpinEdit***
3. Значення функції виводяться на форму з використанням компоненти ***TstringGrid***
4. Реалізувати контроль наявності факту введення даних;
5. Забезпечити конроль правильності введення числових даних для меж табулювання та кроку
6. Мінімальне, максимальне та середнє арифметичне значення вивести на форму за допомогою компонент ***Tlabel, Tedit, TlabeledEdit.***

Варіанти індивідуальних завдань:

(значення змінних a та b, Δx вводяться при запуску програми)

Варіант 1.

1. $y = \dfrac{\cos^3 x^2}{1.5x + 2}.$

Варіант 2.

1. $y = \dfrac{2x + 8}{|\cos 3x| + 1}.$

Варіант 3.

1. $y = \dfrac{6x + 4}{\sin 3x - x}.$

Варіант 4.

1. $y = \dfrac{x + \cos 2x}{3x}.$

Варіант 5.

1. $y = \dfrac{x^3 + 2x}{3\cos\sqrt{x} + 1}.$

Варіант 6.

1. $y = \dfrac{\ln|x + 1| + 5}{2x + 3}.$

Варіант 7.

1. $y = \dfrac{tg0.5x}{x^3 + 7.5}$

Варіант 8.

1. $y = \dfrac{x + \cos 2x}{x + 2}.$

Варіант 9.

1. $y = \dfrac{e^{2x} - 8}{x + 3}.$

Варіант 10.

1. $y = \dfrac{x - \ln 2x}{3x + 1}$.

Варіант 11.

1. $y = \dfrac{x + \sin 2x}{x^2 - 3}$.

Варіант 12.

1. $y = \dfrac{x^2 + 2x}{\cos 5x + 2}$.

Варіант 13.

1. $y = \dfrac{(x+2)^2}{\sqrt{x^2 + 1}}$.

Варіант 14.

1. $y = \dfrac{\cos^2 x}{x^2 + 1}$.

Варіант 15.

1. $y = \dfrac{x^3 - 2}{3\ln x}$

Варіант 16.

1. $y = \dfrac{(3x+2)^2}{\sin x + 3}$.

Варіант 17.

1. $y = \dfrac{tg2x - 3x}{x + 3}$.

Варіант 18.

1. $y = \dfrac{3x + 1}{arctgx}$.

Варіант 19.

1. $y = \dfrac{\arccos x}{2x + 1}$.

Варіант 20.

1. $y = \dfrac{2.5x^3}{e^{2x} + 2}$.

Варіант 21.

1. $y = \dfrac{5\lg x}{x^2 - 1}$.

Варіант 22.

1) $y = \dfrac{2\sin^2(x+2)}{x^2 + 1}$.

Варіант 23.

1. $y = \dfrac{5tg(x+7)}{(x+3)^2}$

Варіант 24.

1. $y = \dfrac{1.5x - \ln 2x}{3x + 1}$.

Лабораторна робота №8

Тема: **Розробка програм з використання одновимірних масивів.**

Мета: Навчитись працювати з одновимірними масивами.

Програмне забезпечення*:* Delphi 7.0

Контрольні питання.

1. Як в загальному описуються масиви?
1. Який масив називається одновимірним?
2. Властивості компоненти StringGrid.
2. Методи роботи із StringGrid.

Приклад виконання практичного завдання.

Варіант 0

В середовищі програмування ***Delphi*** *розробити віконний додаток находження максимального елементу заданого масиву із десяти цілих чисел.*

Етап проектування інтерфейсу проекту:

Під час проектування використаємо компоненти класів ***TEdit***, ***Tlabel***, ***TButton, TStringGrid***.

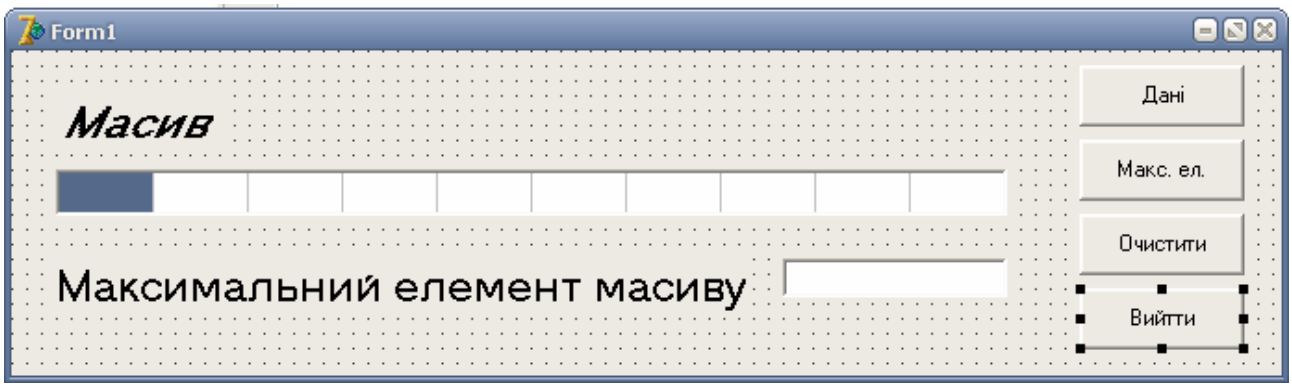


Рис. 39. Орієнтовне зображення проекту

Для відображення масиву використаємо компоненту ***TStringGrid,*** *для якої* під час проектування визначимо необхідну кількість комірок (властивість ***ColCount),*** заберемо фіксовані комірки, залишимо тільки один *рядок* ***(RowCount:=1).***

Етап програмування проекту:

При натисканні на кнопку ***'Дані',*** випадковим чином генерується масив цілих чисел від **-100** до **100.** Процедура обробки даної події має ***вигляд:***

```
procedure TForml.ButtonlClick(Sender: TObject);
var i:integer;
begin
        Randomize;
         For i:= 0 to 9 do
        StringGrid1. Cells[i,0]:=IntToStr(Random(200)-100);
end;
```

Максимальний елемент обчислюється при натисканні на кнопку 'Макс. Ел'. Обробка події реалізується наступною функцією:

```
procedure TForm1.Button2Click(Sender: TObject);
var a: array [0..9] of integer;
i,max:integer;
begin
 for i:=0 to 9 do
 a[i]:= StrToInt(StringGrid1.cells[i,0]);
 max:=a[0];
 for i:=1 to 9 do
 if a[i]>max then max:=a[i];
 Edit1.Text:=IntToStr(max);
end;
```

Також на формі наявна кнопка очищення даних 'Очистити'. Обробка події реалізується наступною функцією:

```
procedure TForm1.Button3Click(Sender: TObject);
var i : integer;
begin
 For i:= 0 to 9 do
 StringGrid1. Cells[i,0]:='';
```

```
 edit1.Text:='';
end;
```

Також на формі наявна кнопка виходу з програми **'Вийти'**. Обробка події реалізується наступною функцією:

```
procedure TForm1.Button4Click(Sender: TObject);
begin
 close;
end;
```

Результат виконання проекту:

Для заданого масиву, що генерується випадковим чином визначено максимальний елемент 99

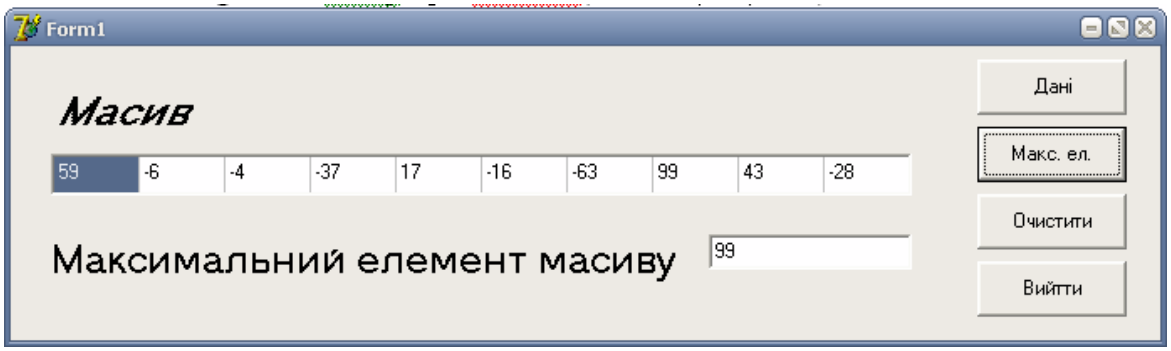


Рис.40. Орієнтовне зображення результату виконання проекту

Практичне завдання по варіантам.

В середовищі програмування Delphi розробити програму, яка б для заданого масиву D(n) із n елементів виконувала наведеш нижче варіанти завдань:

Вимоги до проекту:

Для введення масивів використати компонент TStringGrid або TMemo.

Забезпечити правильність введення даних.

Варіанти індивідуальних завдань:

Варіант 1.

1. Кожен елемент дорівнює 1,2 або 3. Впорядкувати масив таким чином, щоб всі одиничкі йшли першими, а трійки останніми.

Варіант 2.

1. Визначити кількість від'ємних елементів масиву.

Варіант 3.

1. Знайти кількість додатних елементів масиву.

Варіант 4.

1. Визначити порядкові номери від'ємних елементів масиву.

Варіант 5.

1. Обчислити добуток додатних елементів масиву.

Варіант 6.

1. Забезпечити заміну елементів, значення яких менші за 1 на середнє арифметичне. Підрахувати кількість проведених замін.

Варіант 7.

1. Знайти і надрукувати суму додатних елементів масиву.

Варіант 8.

1. Визначити середнє арифметичне додатних елементів та суму від'ємних.

Варіант 9.

1. У масиві D(n) із n елементів є хоча б один нуль. Визначити кількість елементів після першого нуля.

Варіант 10.

1. Сформувати новий масив, замінивши елементи, які стоять на парних місцях на елементи, які стоять на непарних місцях та навпаки

Варіант 11.

1. Знайти середнє арифметичне елементів масиву і на місці від'ємних елементів поставити це значення.

Варіант 12.

1. Знайти різницю між максимальним та мінімальним серед від'ємних елементів.

Варіант 13.

1. Знайти максимальний серед від'ємних елементів та вказати його номер.

Варіант 14.

1. Знайти мінімальний серед від'ємних елементів та вказати його номер.

Варіант 15.

1. Визначити середнє арифметичне додатних елементів масиву.

Варіант 16.

1. Знайти мінімальний серед додатних елементів масиву.

Варіант 17.

1. У масиві D(n) із n елементів є хоча б один нуль. Знайти мінімальний по модулю елемент у частині масиву після останнього нуля.

Варіант 18.

1. Визначити номери додатних елементів масиву.

Варіант 19.

1. Визначити порядкові номери невід'ємних елементів масиву.

Варіант 20.

1. Знайти максимальний по модулю елемент масиву.

Варіант 21.

1. Обчислити добуток елементів масиву.

Варіант 22.

1. Знайти середнє арифметичне невід'ємних елементів масиву, які стоять на непарних місцях.

Варіант 23.

1. Підрахувати кількість елементів масиву, значення яких більше 2,3.

Варіант 24.

1. Знайти суму від'ємних елементів, які стоять на парних місцях в масиві.

Лабораторна робота №9

Тема: **Розробка програм з використання одновимірних масивів та головного і контекстного меню.**

Мета: Навчитись працювати з одновимірними масивами, головним та контекстним меню.

***Програмне забезпечення**:* Delphi 7.0

Контрольні питання.

1. Як в загальному описуються масиви?
2. Який масив називається одновимірним?
3. Властивості компоненти StringGrid.
4. Методи роботи із StringGrid.
5. Властивості компоненти MainMenu.
6. Методи роботи із MainMenu.
7. Властивості компоненти PopupMenu.
8. Методи роботи із PopupMenu.

Приклад виконання практичного завдання.

Варіант 0

В середовищі програмування Delphi розробити програму, яка б для *заданих масивів цілих чисел А та В розмірності n визначала:*

- *кількість відповідних елементів масиву А та В, які рівні між собою;*
- *кількість відповідних елементів масиву А, які є квадратами відповідних елементів масиву В;*
- *кількість відповідних елементів масиву А та В, які не рівні між собою.*

Етап проектування інтерфейсу проекту:

Під час проектування використаємо компоненти класу ***TEdit, TLabel, TButton, TStringGrid, TSpinEdit, TMemo.***

Основні властивості використаних компонент:

SpinEditl.MaxValue:=20;

SpinEditl.MinValue:=1;

При формуванні проекту використані три компоненти ***Checkbox,*** які визначають потрібне завдання:

CheckBoxl. Caption:= 'Кількість відповідних елементів **масивів** ***в А*** **і** **В** ***рівних між собою';***

CheckBox2.Caption:='Кількість відповідних елементів масивів в А, що є квадратами відповідних елементів у В';

CheckBox3.Caption:= 'Кількість відповідних елементів **масивів в** ***А*** **і** **В** ***нерівних між собою'.***

Тобто за бажанням можна вибрати кожне із завдань окремо, або всі разом.

Для відображення масиву використаємо компоненту ***TStringGrid***, для якої під час проектування заберемо фіксовані комірки, залишимо лише один

рядок, а під час виконання формування необхідної кількості комірок забезпечуватиметься за допомогою компоненти ***SpinEdit1***.

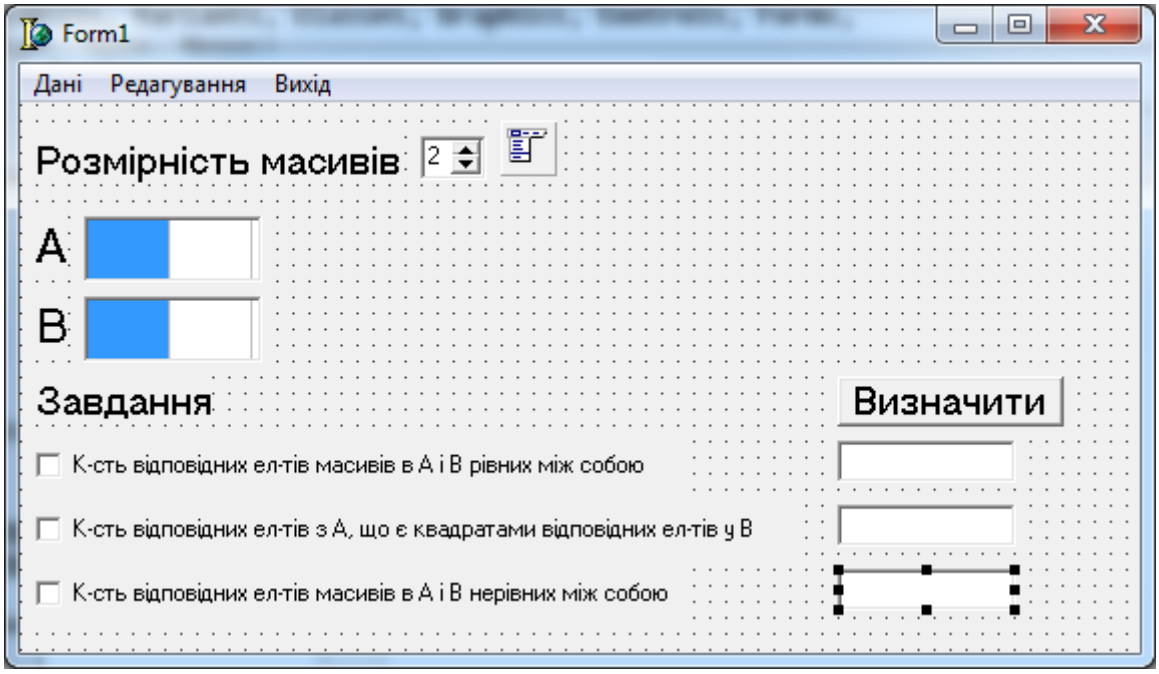


Рис. 41. Орієнтовне зображення проекту

На етапі проектування в проекті сформовано наступне меню (використовуючи компоненту ***TMainMenu***):

1. Дані

Генеруються випадковим чинам

Задаються користувачем

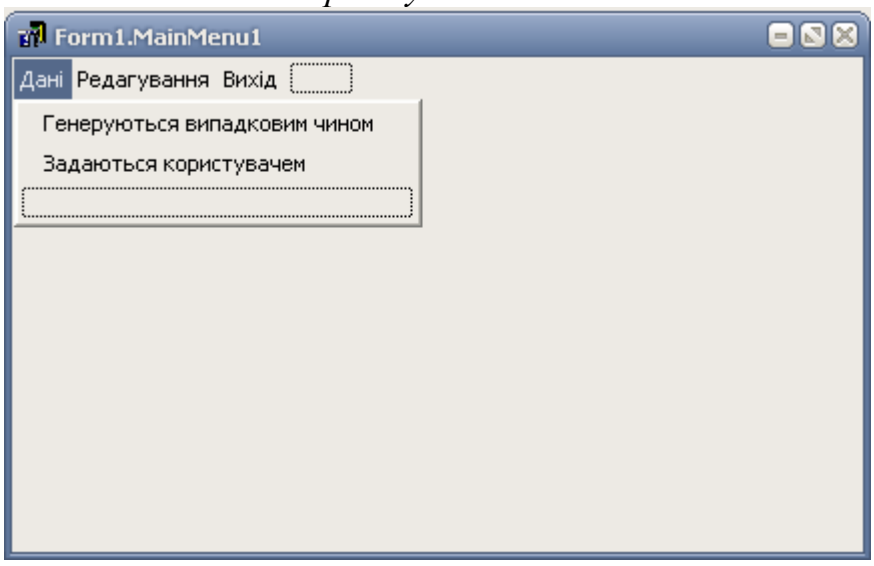


Рис. 42 Орієнтовне зображення проекту

2. Редагування

Очистити

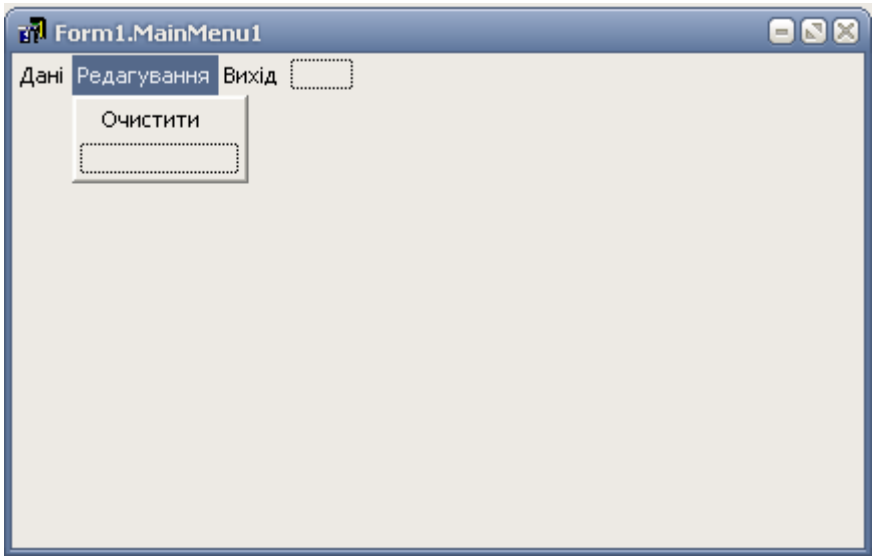


Рис. 43. Орієнтовне зображення проекту

3. Вихід

Етап програмування проекту:

Як глобальні описані наступні змінні:

```
var size:integer;
a:array[1..20] of integer;
b:array[1..20] of integer;
```

Відповідно до зміни значення наступна процедура змінює кількість комірок для введення значень масиву:

```
procedure TForm1.SpinEdit1Change(Sender: TObject);
begin
 Stringgrid1.ColCount:=spinedit1.Value;
 stringgrid1.Width:=spinedit1.Value*31+10;
 stringgrid2.ColCount:= spinedit1.Value;
 stringgrid2.Width:=spinedit1.Value*31+10;
end;
```

Процедура, яка дозволяє закривати додаток:

```
PROCEDURE TFORM1.N6CLICK(SENDER: TOBJECT);
BEGIN
 CLOSE;
END;
```

Процедура, яка визначає можливість вибору першого завдання.

```
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
 if checkbox1.Checked then edit1.Visible:=true else edit1.Visible:=false;
end;
```

Процедура, яка визначає можливість вибору другого завдання:

```
PROCEDURE TFORM1.CHECKBOX2CLICK(SENDER: TOBJECT);
BEGIN
 IF CHECKBOX2.CHECKED THEN EDIT2.VISIBLE:=TRUE ELSE EDIT2.VISIBLE:=FALSE;
END;
```

Процедура, яка визначає можливість вибору третього завдання:

```
procedure TForm1.CheckBox3Click(Sender: TObject);
begin
 if checkbox3.Checked then edit3.Visible:=true else edit3.Visible:=false;
end;
```

Обробка події ***OnClick*** для кнопки ***'Визначити'*** забезпечу ***наступною*** процедурою:

```
procedure TForm1.Button1Click(Sender: TObject);
var i, a1, b1, c : integer;
begin
 for i:=1 to spinedit1.Value do
 begin
 a[i]:=strtoint(stringgrid1.Cells[i-1,0]);
 b[i]:=strtoint(stringgrid2.Cells[i-1,0]);
 end;
 a1:=0; b1:=0; c:=0;
 for i:=1 to spinedit1.Value do
 begin
 if a[i]=b[i] then a1:=a1+1;
 if a[i]=b[i]*b[i] then b1:=b1+1;
 if a[i]<>b[i] then c:=c+1;
 end;
 if checkbox1.Checked=true then edit1.Text:=inttostr(a1);
 if checkbox2.Checked=true then edit2.Text:=inttostr(b1);
 if checkbox3.Checked=true then edit3.Text:=inttostr(c);
end;
```

Процедура, що забезпечує роботу підрозділу **меню** *Генерація випадковим чином* для заповнення масиву випадковими значеннями із діапазону від **0** до **100.**

```
procedure TForm1.N2Click(Sender: TObject);
var i:integer;
begin
 randomize;
 for i:=1 to spinedit1.Value do
 begin
 stringgrid1.Cells[i-1,0]:=inttostr(random(100));
```

```
 stringgrid2.Cells[i-1,0]:=inttostr(random(100));
 end;
end;
```

Процедура, що забезпечує роботу підрозділу меню ***Задається користувачем*** для заповнення масиву випадковими значеннями із діапазону від 0 до 100.

```
procedure TForm1.N3Click(Sender: TObject);
var i:integer;
begin
 stringgrid1.Options:=[gofixedvertline,gofixedhorzline,govertline,
 gohorzline,gorangeselect,goediting];
 stringgrid2.Options:=[gofixedvertline,gofixedhorzline,govertline,
 gohorzline,gorangeselect,goediting];
 for i:=1 to spinedit1.value do
 begin
 stringgrid1.cells[i-1,0]:='';
 stringgrid2.cells[i-1,0]:='';
 end;
end;
```

Процедура, що забезпечує роботу підрозділу меню *'Очистити',* **для** *очищення* масиву від даних:

```
procedure TForm1.N5Click(Sender: TObject);
var i:integer;
begin
 for i:=1 to spinedit1.value do
 begin
 stringgrid1.cells[i-1,0]:='';
 stringgrid2.cells[i-1,0]:='';
 end;
 checkbox1.Checked:=false;
 checkbox2.Checked:=false;
 checkbox3.Checked:=false;
end;
```

Результат виконання проекту:

Приклад виконання програми:

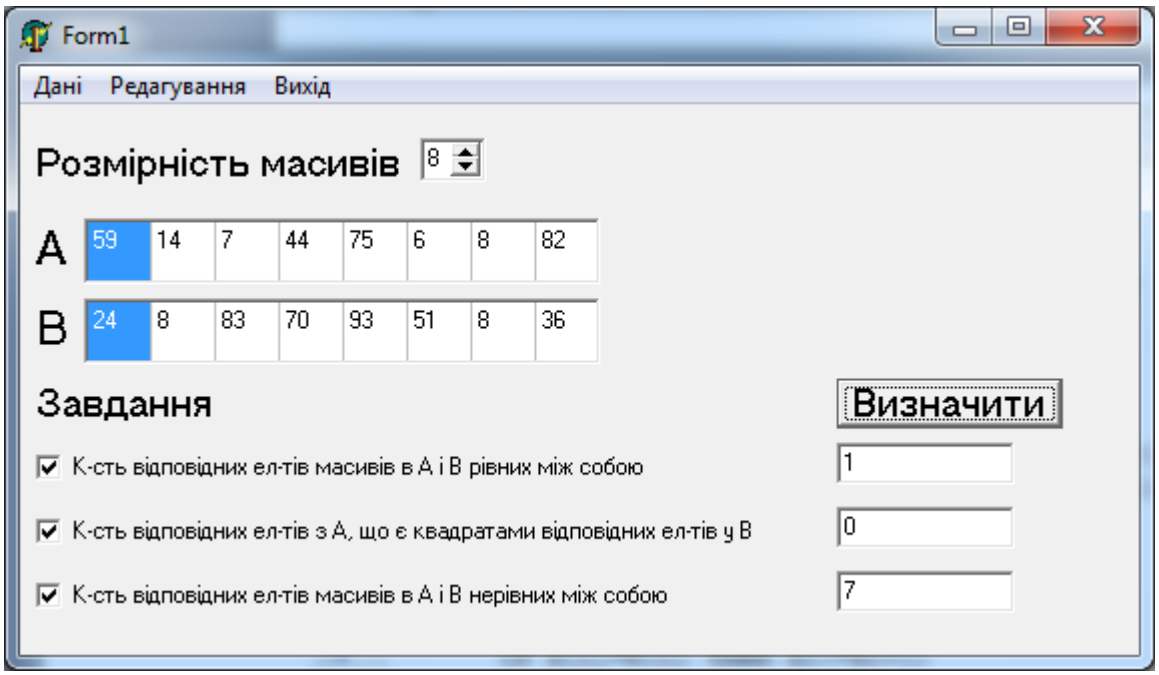


Рис. 44. Орієнтовне зображення результату виконання проекту

Практичне завдання по варіантам.

В середовищі програмування Delphi розробити проект для реалізації завдань, які наведені нижче

1. Розробити головне меню (використати компоненту ***TMainMenu),*** яке б мало, наприклад, наступні розділи:

Дані
Розмірність масиву
Задаються користувачем
Редагування
Допомога
Інформація про користування програмою

Вихід

2. Розробити контекстні мені для кожної компоненти, що знаходиться на ***формі,*** та для форми (використати компоненту ***TPopupMenu);*** в контекстному меню мають бути команди, які стосуються окремої компоненти, або додатку в цілому (якщо меню для форми).

3. Для введення масивів використати компонент ***TStringGrid*** або ***TMemo;***

4. Забезпечити правильність введення даних.

Вимоги до проекту:

• забезпечити створення головного меню;

• розмірність масивів має вводитись на окремій формі (використати компоненту ***TSpinEdit).***

• для вибору пункту, за яким буде здійснюватись пошук, скористатись компонентою ***TListBox*** або ***TComboBox*****,**

• для введення масивів використати компонент ***TStringGrid*** або ***TMemo:***

• вивід результатів для кожного пунктів пошуку має здійснюватись на різних формах.

Варіанти індивідуальних завдань:

Варіант 1.

В середовищі програмування Delphi розробити програму, яка б для заданих масивів A та B, елементами яких є стрічки (розмірності n) визначала: 1) кількість взаємних пар A[i] та B[i], де довжина стрічки A[i] дорівнює сумі цифр, що зустрічаються в елементі B[i]; 2) кількість взаємних пар A[i] та B[i], де довжини стрічок рівні; 3) кількість взаємних пар A[i] та B[i], де кількість символів A[i] дорівнює кількості цифр в B[i].

Варіант 2.

В середовищі програмування Delphi розробити програму, яка б для заданих масивів цілих чисел A та B розмірності n визначала: 1) кількість відповідних елементів масиву A, які діляться на відповідні елементи масиву B; 2) номери відповідних елементів масиву A, в яких добуток цифр у числі співпадає із кубами відповідних елементів масиву B; 3) номери відповідних елементів масиву A, які є коренями квадратними відповідних елементів масиву B.

Варіант 3.

В середовищі програмування Delphi розробити програму, яка б для масиву дійсних чисел розмірності n визначала: 1) максимальний елемент; 2) мінімальний елемент, серед елементів які мають парні номери; 3) кількість перестановок елементів в масиві, якщо впорядкувати його за зростанням.

Варіант 4.

В середовищі програмування Delphi розробити програму, яка б ддя заданих масивів цілих чисел A та B розмірності n виконувала: 1) злиття двох масивів в один масив, відсортований у порядку зростання; 2) підрахунок середнього арифметичного значення елементів масивів A та B; 3) підрахунок суми елементів двох масивів, які кратні 2.

Варіант 5.

В середовищі програмування ***Delphi*** розробити програму, яка б для двох фіксованих, впорядкованих по зростанню масиви A розмірності n та B розмірності m, де n>0, m>0 визначала 1) кількості значень, які зустрічаються як масиві A, так і в B; 2) кількість значень, які зустрічаються тільки в масиві A; 3) кількість значень, які зустрічаються тільки в масиві B. Відомо, що в межах масивів A та B немає однакових елементів.

Варіант 6.

В середовищі програмування Delphi розробити програму, яка б для заданих масивів цілих чисел А та В розмірності n визначала: 1) кількість відповідних елементів масиву А та В, які кратні 2; 2) номери відповідних елементів масиву А, які більші відповідних елементів масиву В; 3) кількість відповідних елементів масиву А та В, які не рівні між собою.

Варіант 7.

В середовищі програмування Delphi розробити програму, яка б для заданої таблиці розмірності (nxm), елементи якої можуть бути набори довільних символів, визначала: 1) кількість комірок a_{ij}, в яких довжина відповідного слова більша за суму цифр, які зустрічаються у слові; 2) кількість комірок a_{ij}, в яких кількість символів переважає над кількість цифр; 3) кількість комірок a_{ij}, в яких довжина кратна 3.

Варіант 8.

В середовищі програмування Delphi розробити програму, яка б для двох фіксованих, впорядкованих по зростанню масиви А розмірності n та В розмірності m, де n >0, m >0 визначала: 1) суму значень, які зустрічаються як масиві А, так і в В; 2) кількість значень, які зустрічаються тільки в масиві А та стоять на парних місцях; 3)кількість значень, які зустрічаються тільки в масиві В та стоять на непарних місцях. Відомо, що в межах масивів А та В немає однакових елементів.

Варіант 9.

В середовищі програмування Delphi розробити програму яка б заданих масивів цілих чисел А та В розмірності n виконувала: 1) злиття двох масивів в один масив із видаленням елементів менші 10; 2) сортування новоствореного масиву у порядку спадання його елементів; 3) підрахунок суми елементів масиву із парними номерами.

Варіант 10.

В середовищі програмування Delphi розробити програму, яка б для заданих масивів цілих чисел А та В розмірності n визначала: 1) кількість відповідних елементів масиву А, які більші відповідних парних елементів масиву В; 2) кількість відповідних елементів масиву А, які менші відповідних непарних елементів масиву В; 3) кількість відповідних елементів масиву А, які кратні відповідним елементам масиву В.

Варіант 11.

В середовищі програмування *Delphi* розробити програму, яка **б** для *заданих* масивів цілих чисел А та В розмірності n визначала: 1) номери відповідних елементів масивів А та В, які рівні між собою; *2) номери* відповідних елементів масиву А, які є кубами відповідних *елементів* масиву В; *3) номери* відповідних елементів масиву А, які є коренями квадратними відповідних елементів масиву В.

Варіант 12.

В середовищі програмування Delphi розробити програму, яка б для масиву цілих чисел розмірності n визначала: 1) максимальний елемент, серед парних елементів масиву із непарними номерами; 2) мінімальний елемент, серед елементів, які мають непарні номери; 3) кількість елементів в масиві, які кратні 3.

Варіант 13.

В середовищі програмування Delphi розробити програму, яка б для заданої таблиці розмірності (nxm). елементи якої можуть бути набори довільних символів, визначала: 1) кількість комірок a_{ij}, в яких довжина відповідного слова більша за i*j; 2) кількість комірок a_{ij}, в яких довжина відповідного слова менша за i+j; 3) кількість комірок a_{ij}, в яких довжина відповідного слова більша суми цифр, що зустрічаються в даному слові.

Варіант 14.

В середовищі програмування **Delphi** розробити програму, яка б для заданих масивів цілих чисел A та B розмірності n виконувала: 1) злиття двох масивів в один масив; 2) сортування новоствореного масиву у порядку зростання його; 3) підрахунок суми елементів масиву.

Варіант 15.

В середовищі програмування Delphi розробити програму, яка б для заданих масивів цілих чисел A та B розмірності n визначала: 1) номери відповідних елементів масиву A, які є добутками цифр відповідного значення масиву B; кількість елементів масиву A, які є більші відповідних елементів масиву B; 3) номери *i* такі, що A[i] = sin(2* B[i])

Варіант 16.

В середовищі програмування ***Delphi*** розробити програму, яка б для заданих масивів цілих чисел A та B розмірності n виконувала: 1) злиття двох масивів в один масив, відсортований у порядку зростання; 2) підрахунок суми мінімального елемента масиву A та максимального цента масиву B; 3) підрахунок суми елементів двох масивів.

Варіант 17.

В середовищі програмування Delphi розробити програму, яка б для заданих масивів A та B, елементами яких є стрічки (розмірності n) визначала: 1) кількість взаємних пар A[i] та B[i], де довжина стрічки A[i] дорівнює сумі цифр, що зустрічаються в елементі B[i]; 2) кількість взаємних пар A[i] та B[i], де символи, що стоять на i-му місці в A[i] та B[i] рівні між собою; 3) кількість взаємних пар A[i] та B[i], де кількість символів A[i] дорівнює кількості символів в B[i].

Варіант 18.

В середовищі програмування ***Delphi*** розробити програму, яка б для заданих масивів цілих чисел А та В розмірності n визначала: ***1)*** номери відповідних елементів масиву А, які є квадратами ***відповідних*** елементів масиву В; ***2)*** номери відповідних елементів масиву А, які є більші відповідних ***елементів*** масиву В; ***3)*** номери *i* такі, що A[i] = sin(B[i]).

Варіант 19.

В середовищі програмування delphi розробити програму, яка б для заданих масивів цілих чисел А та В розмірності n визначала: 1) кількість відповідних елементів масиву А та В, сума цифр яких більша 5; 2) кількість відповідних елементів масиву А, які є менші відповідних елементів масиву В; 3) кількість відповідних елементів масиву А та В, які рівні між собою;

Варіант 20.

В середовищі програмування Delphi розробити програму, яка б для заданих масивів цілих чисел А та В розмірності n визначала: 1) кількість відповідних елементів масиву А, які більші відповідних елементів масиву В; 2) кількість відповідних елементів масиву А, які менші відповідних елементів масиву В; 3) кількість відповідних елементів масиву А, які кратні відповідним елементам масиву В.

Варіант 21.

В середовищі програмування Delphi розробити програму, яка б для масиву дійсних чисел розмірності n визначала: 1) максимальний елемент; 2) мінімальний елемент; 3) кількість перестановок елементів в масиві, якщо впорядкувати його за зростанням.

Варіант 22.

В середовищі програмування Delphi розробити програму, яка 6 для заданих масивів цілих чисел А та В розмірності n визначала: 1) номери відповідних елементів масивів А та В, які не рівні між собою; 2) номери відповідних елементів масиву А, які є сумою цифр елементів масиву В; 3) номери і такі, шо A[i] = cos(B[i]).

Варіант 23.

В середовищі програмування Delphi розробити програму, яка б для заданої таблиці розмірності (nxm), елементи якої можуть бути набори довільних символів, визначала: 1) кількість комірок a_{ij}, в яких довжина відповідного слова більша за i+j; 2) кількість комірок a_{ij}, в яких довжина відповідного слова менша за i+j; 3) кількість комірок a_{ij},, в яких довжина відповідного слова дорівнює сумі цифр, що зустрічаються в даному слові.

Варіант 24.

В середовищі програмування Delphi розробити програму, яка б длямасивів цілих чисел А та В розмірності n визначала: 1) номери відповідних елементів масивів А та В, які рівні між собою; 2) номери відповідних елементів масиву А, які є кубами відповідних елементів масиву В; 3) номери відповідних елементів масиву А, які с коренями квадратними відповідних елементів масиву В.

Лабораторна робота №10

Тема: **Розробка програм з обробки багатовимірних масивів. Створення бататовіконних додатків.**

Мета:

1. Навчитись працювати з візуальними компонентами сторінки Standard: TPanel, сторінки Win32: TTabControl та TStatusBar.
2. Закріпити навики роботи з компонентами сторінки Samples: TSpinEdit, сторінки Standard: TMemo, TEdit, TButton, TLabei та сторінки Additional: TBitBtn, TLabeledEdit, TStringGrid.
3. Вивчити призначення, основні властивості та методи роботи з компонентами сторінки Standard: TPanel, сторінки Win32: TTabControl та TStatusBar.
4. Вивчити елементи створення баготовіконного Windows-додатку

Програмне забезпечення*:* Delphi 7.0

Контрольні питання.

1. Принципи роботи з масивами?
2. Який масив називається багатовимірним?
3. Який масив називається статичним?
4. Що таке динамічний масив?
5. Властивості компоненти TStringGrid та TMeto.
6. Назвіть основні властивості компоненти TTabControl.

Практичне завдання по варіантам.

В середовищі програмування Delphi розробити програми, що реалізують нижченаведені варіанти завдань. Забезпечити виконання наступних вимог:

1) створити головне меню (використати головне меню TMainMenu);

2) розмірність матриць вводити на окремій формі (компонента TSpinEdit);

3) для введення матриць використати компоненти TMemo та *TStringgrid (для вибору способу введення використати компоненту TTabControl);*

4) виведення результатів має здійснюватися на окрему форму (відповідно до способу введення даних).

Варіанти індивідуальних завдань:

Варіант 1.

Дана квадратна матриця А розмірності nxn. Визначити: скільки її стовпців впорядковані по зростанню та скільки її стовпців впорядковані за спаданням.

Варіант 2.

В квадратній матриці С (nxn) підрахувати та вивести на екран окремо суму додатних та від'ємних елементів, підрахувавши при цьому їх кількість.

Варіант 3.

Дана квадратна матриця А розмірності nxn (n - парне число), яка складається із цілих чисел. Якщо в непарних рядочках кількість парних чисел непарна, то дані рядки поміняти з непарними стовпчиками, інакше -з парними.

Варіант 4.

Дана квадратна матриця А розмірності nxn (n - парне число), яка складається з цілих чисел. Якщо в парних стовпчиках кількість непарних чисел парна, то дані в стовпчиках замінити на дані непарних рядків, а якщо ні - то парних рядків.

Варіант 5.

Дана матриця А (nxn). Замінити всі елементи даної матриці, сума індексів яких парна, добутками відповідних індексів.

Варіант 6.

Дана матриця А (nxn). Отримати нову матрицю шляхом ділення всіх її елементів на найбільший по модулю елемент,

Варіант 7.

Дана матриця А (nxn). Знайти кількість від'ємних елементів, що лежать нижче головної діагоналі.

Варіант 8.

Дана матриця а(nxn). Знайти мінімальний елемент для даної матриці та вивести його. Замінити елементи матриці, що лежать нижче головної діагоналі, на мінімальний елемент

Варіант 9.

Дано матриця В (nxn). Перетворити матрицю, замінивши додатні елементи - номером рядка, в якому вони знаходяться, від'ємні елементи - номером стовпчика, в якому вони знаходяться, нульові елементи - сумою відповідного рядка та стовпчика.

Варіант 10.

Провести перетворення матриці А таким чином: додатні елементи замінити на число 2, від'ємні - на 1, нульові - на 3.

Варіант 11.

В квадратній матриці С (nxn) підрахувати та вивести на екран окремо суму додатних та від'ємних елементів, підрахувавши при цьому їх кількість.

Варіант 12.

Дана квадратна матриця А розмірності nxn. Визначити: скільки її стовпців впорядковані по зростанню та скільки її стовпців впорядковані за спаданням.

Варіант 13.

Дана квадратна матриця А розмірності nxn (n - парне число), яка складається із цілих чисел. Якщо в непарних рядочках кількість парних чисел непарна, то дані рядки поміняти з непарними стовпчиками, інакше -з парними.

Варіант 14.

Дана квадратна матриця А розмірності nxn (n - парне число), яка складається з цілих чисел. Якщо в парних стовпчиках кількість непарних чисел парна, то дані в стовпчиках замінити на дані непарних рядків, а якщо ні - то парних рядків.

Варіант 15.

Дана матриця А (nxn). Замінити всі елементи даної матриці, сума індексів яких парна, добутками відповідних індексів.

Варіант 16.

Дана матриця А (nxn). Отримати нову матрицю шляхом ділення всіх її елементів на найбільший по модулю елемент.

Варіант 17.

Дана матриця А (nxn). Знайти кількість від'ємних елементів, що лежать нижче головної діагоналі.

Варіант 18.

Дана матриця а(nxn). Знайти мінімальний елемент для даної матриці та вивести його. Замінити елементи матриці, що лежать нижче головної діагоналі, на мінімальний елемент

Варіант 19.

Дано матриця В (nxn). Перетворити матрицю, замінивши додатні елементи - номером рядка, в якому вони знаходяться, від'ємні елементи -

номером стовпчика, в якому вони знаходяться, нульові елементи - сумою відповідного рядка та стовпчика.

Варіант 20.

Провести перетворення матриці А таким чином: додатні елементи замінити на число 2, від'ємні - на 1, нульові - на 3.

Варіант 21.

Дана матриця А (nxn). Замінити всі елементи даної матриці, сума індексів яких парна, добутками відповідних індексів.

Варіант 22.

Дана квадратна матриця А розмірності nxn. Визначити: скільки її стовпців впорядковані по зростанню та скільки її стовпців впорядковані за спаданням.

Варіант 23.

Дано матриця В (nxn). Перетворити матрицю, замінивши додатні елементи - номером рядка, в якому вони знаходяться, від'ємні елементи -номером стовпчика, в якому вони знаходяться, нульові елементи - сумою відповідного рядка та стовпчика.

Варіант 24.

Дана квадратна матриця А розмірності nxn (n - парне число), яка складається із цілих чисел. Якщо в непарних рядочках кількість парних чисел непарна, то дані рядки поміняти з непарними стовпчиками, інакше -з парними.

Лабораторна робота №11

Тема: **Розробка додатків з використанням графічних можливостей.**
Мета: навчитись будувати графіки та діаграми

Програмне забезпечення*:* Delphi 7.0

Контрольні питання.

1. Які існують методи для малювання?
2. Як зобразити зафарбований прямокутник?
3. Як зробити текстовий надпис?
4. Охарактеризуйте змінні в побудові прямокутника.
5. Як намалювати чорну крапку на екрані?

Практичне завдання по варіантам.

Завдання 1.

В середовищі програмування Delphi розробити програму побудови графіка функції y = f(x) на відрізку [a,b].

Вимоги до проекту:

1) всі вхідні дані вводяться на окремих формах (наприклад, задання відрізка [a, b]);
2) розробити масштабування по осях;
3) програма повинна містити головне меню;
4) передбачити вибір кольору графіка, фону, осей координат (використати компоненту TColorDialog)
5) графік намалювати не на формі, а окремій компоненті (використати компоненту TImage, або TPaintBox)

Варіанти індивідуальних завдань:

Варіант 1.

$$y = \big||x| - 1\big|.$$

Варіант 2.

$$y = 3\sin\left(\frac{x}{2}\right)e^{x/2}$$

Варіант 3.

$$y = \sqrt{2x - x^2}$$

Варіант 4.

$$y = \frac{1}{2}\big(|\sin x| - \sin x\big).$$

Варіант 5.

$$y = \frac{1}{5}\big(|\sin x| + \cos x\big)$$

Варіант 6.

$$y = \cos x + |\cos x|$$

Варіант 7.

$$y = \sqrt{x} + \sin x$$

Варіант 8.

$$y = \frac{\sin x}{e^x}$$

Варіант 9.

$$y = \sin^3 x$$

Варіант 10.

$$y = \sin^{4/3} x.$$

Варіант 11.

$$y = 2^{\cos x}$$

Варіант 12.

$$y = \frac{\ln x}{x}$$

Варіант 13.

$$y = x\sin x$$

Варіант 14.

$$y = \sqrt{x} - \cos x$$

Варіант 15.

$$y = \sin x + \cos x$$

Варіант 16.

$$y = x + \sin|x|$$

Варіант 17.

$$y = 2 + 3|\cos x|$$

Варіант 18.

$$y = e^{-x}\sin x$$

Варіант 19.

$$y = 2|\sin x| - 1$$

Варіант 20.

$$y = \frac{1}{3}\sin|x|$$

Варіант 21.

$$y = x - \sin x$$

Варіант 22.

$$y = |\cos(2x)|$$

Варіант 23.

$$y = \sin(\sin x)$$

Варіант 24.

$$y = \arcsin(\sin x)$$

Завдання 2.

В середовищі програмування Delphi розробити програму побудови діаграми (використати компоненту TChart), яка б відображала інформацію про розподіл доходу вашої сім'ї.

Лабораторна робота №12

Тема: **Робота з файлами.**

Мета: навчитись працювати з файлами

Програмне забезпечення*:* Delphi 7.0

Контрольні питання.

1) Дайте визначення файлу.
2) Технологія роботи з файлами.
3) Які існують типи файлів?
4) Що являють собою типізовані файли?
5) Що являють собою файли без типу?
6) Що являють собою текстові файли?
7) Які ви знаєте процедури та функції для роботи з файлами?

Практичне завдання по варіантам.

Завдання 1.

В середовищі програмування Delphi розробити програму відповідно до варіанта і потрібно врахувати, що вхідні та вихідні дані зчитуються і записуються у файл.

Вимоги до проекту:

1) програма повинна містити головне меню;
2) програма повинна зчитувати дані і виводити результат у файл.

Варіанти індивідуальних завдань:

Варіант 1.

Розробити проект, який би дозволяв визначити всі слова тексту, що складаються із тих самих букв, що і друге слово цього тексту.

Варіант 2.

Розробити проект, який би забезпечував знаходження суми елементів файлу.

Варіант 3.

Розробити проект, який би забезпечував читання з файлу та виведення на екран спочатку однобуквені слова, а потім решту слів.

Варіант 4.

Розробити проект, який забезпечував читання з файлу та виведення на екран спочатку запитання, а потім речень зі знаком оклику.

Варіант 5.

Розробити проект, який забезпечував читання з файлу та виведення на екран тільки тих речень, які мають максимальну кількість знаків пунктуації.

Варіант 6.

Розробити проект, який би забезпечував читання з файлу і виведення на екран тільки тих речень, які мають визначену кількість слів.

Варіант 7.

Розробити проект, який би забезпечував читання з файлу та виведення на екран тільки тих речень, які не містять ком.

Варіант 8.

Розробити проект, який забезпечував читання з файлу і виведення на екран тільки тих речень, які містять двозначні числа.

Варіант 9.

Розробити проект, який забезпечував читання з файлу і виведення на екран тільки тих речень, які містять задане слово.

Варіант 10.

Розробити проект, який би дозволяв читати з файлу три послідовні речення, порядковий номер першого речення при цьому задається із клавіатури, і виводила їх на екран у зворотному напрямку

Варіант 11.

Розробити проект, який би дозволяв визначати всі слова тексту, що складаються із тих самих букв, що і перше слово цього тексту.

Варіант 12.

Розробити проект, який забезпечував знаходження добутку елементів файлу.

Варіант 13.

Розробити проект, який би визначав всі слова тексту, що складаються з тих самих букв, що і останнє слово цього тексту.

Варіант 14.

Розробити проект для визначення в тексті максимальну довжину послідовності символів, що не є буквами.

Варіант 15.

Розробити проект для підрахунку в тексті кількості слів, які є паліндромами.

Варіант 16.

Розробити проект для підрахунку кількості знаків пунктуації.

Варіант 17.

Розробити проект для підрахунку кількості слів у тексті.

Варіант 18.

Задано два типізованих файли, які містять цілі числа. Відсортувати файли за спаданням, забезпечити злиття файлів та впорядкувати отриманий файл за зростанням.

Варіант 19.

Задано текст, який складається зі слів. Слова розділені пропусками. Знайти порядковий номер першого найдовшого слова.

Варіант 20.

Розробити проект сортування типізованого файлу (впорядкування його компонент за зростанням), що містить цілі числа.

Варіант 21.

Розробити проект, який би дозволив обробляти інформацію результатів модуля із курсу „Програмування" для груп 21,22,23 (Прізвище, кількість балів). Забезпечити перегляд, редагування, збереження інформації.

Варіант 22.

Розробити проект, який би забезпечував знаходження суми елементів файлу.

Варіант 23.

Розробити проект, який би забезпечував читання з файлу і виведення на екран тільки тих речень, які мають визначену кількість слів.

Варіант 24.

Розробити проект, який забезпечував читання з файлу і виведення на екран тільки тих речень, які містять задане слово.

Завдання 2.

В середовищі програмування Delphi розробити віконний додаток для реалізації варіантів завдань лабораторної роботи №10. Забезпечити завантаження даних з файлу та збереження результатів.

Лабораторна робота №13

Тема: **Створення багато віконного додатку.**
Мета: навчитись працювати з багатовіконними додатками
Програмне забезпечення*:* Delphi 7.0

Практичне завдання по варіантам.

Створити власний проект згідно варіанту.

Варіанти індивідуальних завдань:

Варіант 1.
Файловий менеджер.
Варіант 2.
Текстовий редактор.
Варіант 3.
Гра «Пятнашки»
Варіант 4.
Гра «Шашки»
Варіант 5.
Гра «Теніс»
Варіант 6.
Гра «Морський бій»
Варіант 7.
Електронний журнал групи.
Варіант 8.
Електронний календар.
Варіант 9.
Телефонний довідник.
Варіант 10.
Калькулятор
Варіант 11.
Гра «Сапер»
Варіант 12.
Тестова оболонка
Варіант 13.
Гра «Змійка»
Варіант 14.
Гра «Тетріс»
Варіант 15.
Кулінарні рецепти.
Варіант 16.
Програма перегляду та редагування графічних файлів.
Варіант 17.
Розв'язання систем n лінійних рівнянь з n невідомими.
Варіант 18.
Українсько-англійський словник
Варіант 19.
Тестова перевірка знань студентів з математики
Варіант 20.
Тестова перевірка знань студентів з мови програмування Pascal.
Варіант 21.
Квадрат Піфагора.
Варіант 22.
Гороскоп (за знаками зодіака).
Варіант 23.
Гороскоп (за роком народження).
Варіант 24.
Тестова перевірка знань студентів з мови програмування Delphi.

Тестові завдання

1) Об’єкт **Form** використовується для:
    1. Створення програмою нового вікна;
    2. Створення нової програми;
    3. Створення текстового файлу;
    4. Створення програмою нового файлу.
2) Властивість **Caption** об’єкта **Form** слугує для:

1. Зміни заголовку форми;
2. Зміни імені форми;
3. Зміни підказки форми;
4. Зміни імені графічного файлу;

3) Властивість **Icon** об’єкта **Form** слугує для:
   1. Зміни заголовку форми;
   2. Зміни імені форми;
   3. Зміни підказки форми;
   4. Задання піктограми, яка буде в заголовку форми під час виконання програми.
4) Об’єкт **Image** використовується для:
   1. Створення програмою нового вікна;
   2. Вставляння графічних файлів;
   3. Створення текстового файлу;
   4. Створення тектових полів у вікні програми.
5) Властивість **ReadOnly** об’єкта **Edit** - це:
   1. Можливість змінити текст (доступність поля);
   2. Текст у полі редагування;
   3. Текст підказки, яка висвітлюється, якщо навести курсор миші;
   4. Висвітлювати/не висвітлювати підказку.
6) Властивість **ShowHint** об’єкта **Edit** - це:
   1. Можливість змінити текст (доступність поля);
   2. Текст у полі редагування;
   3. Текст підказки, яка висвітлюється, якщо навести курсор миші;
   4. Висвітлювати/не висвітлювати підказку.
7) Об’єкт **PopurMenu** використовується для:
   1. Створення незалежного дво- чи трипозиційного прапорця;
   2. Створення головного меню;
   3. Створення контексного меню;
   4. Створення багаторядкового редактора тексту.
8) Властивість **Lines** об’єкта **Memo** - це:
   1. Максимальна можлива кількість введених символів;
   2. Стан прапорця;
   3. Задання початкового тексту у полі редагування;
   4. Наявність смуг прокручування.
9) Властивість **State** об’єкта **CheckBox** - це:
   1. Максимальна можлива кількість введених символів;
   2. Стан прапорця;
   3. Задання початкового тексту у полі редагування;
   4. Наявність смуг прокручування.
10) Властивість **Break** об’єктів **PopurMenu** та **MainMenu** - це:
   1. Автоматичний виклик контексного меню;
   2. Розбиття меню у горизонтальному напрямку;
   3. Команди меню;
   4. Комбінація «гарячих» клавіш для виклику команди меню.
11) Об’єкт **SpeedButton** призначений для:
   1. Створення кнопки панелі інструментів;
   2. Створення головного меню;
   3. Створення контексного меню;

4. Створення у формі двовимірної таблиці символьних рядків.

12) Властивість **RowCount** об'єкта **StringGrid**- це:
1. Кількість рядків таблиці;
2. Кількість стовпців таблиці;
3. Кількість фіксованих стовпців таблиці, які не прокручуються зліва;
4. Кількість фіксованих рядків таблиці, які не прокручуються вгору.

13) Властивість **DropDownCount** об'єкта **ComboBox**- це:
1. Кількість рядків у списку;
2. Впорядкування за списком;
3. Стиль оформлення і використання списку;
4. Кількість рядків у випадаючому списку, які видимі без використання смуг прокручування

14) Об'єкт **RadioButton** використовується для:
1. Введення користувачем рядка символів з клавіатури;
2. Створення нової програми;
3. Створення у формі засобу для використання однієї альтернативної можливості серед декількох;
4. Створення тектових полів у вікні програми.

15) Які з перелічених властивостей є властивостями об'єкта **Form** (можливо декілька варіантів відповідей):
1. Align;
2. Visible;
3. Icon;
4. Picture;
5. Name;
6. Enabled

16) Властивість **Name** об'єкта **Form** слугує для:

1) Зміни заголовку форми;
2) Зміни імені форми;
3) Зміни підказки форми;
4) Зміни імені графічного файлу

17) Об'єкт **Label** використовується для:

3) Створення програмою нового вікна;
4) Створення нової програми;
5) Створення текстового файлу;
6) Створення тектових полів у вікні програми.

5. Об'єкт **Edit** використовується для:
1) Введення користувачем рядка символів з клавіатури;
2) Створення нової програми;
3) Створення у формі засобу для використання однієї альтернативної можливості серед декількох;
4) Створення тектових полів у вікні програми

6. Властивість **Hint** об'єкта **Edit** - це:

18) Можливість змінити текст (доступність поля);

19) Текст у полі редагування;
20) Текст підказки, яка висвітлюється, якщо навести курсор миші;
21) Висвітлювати/не висвітлювати підказку.

7. Властивість **Checked** об'єкта **RadioButton** - це:
   2) Можливість змінити текст (доступність поля);
   3) Стан перемикача;
   4) Текст підказки, яка висвітлюється, якщо навести курсор миші;
   5) Висвітлювати/не висвітлювати підказку.
8. Об'єкт **MainMenu** використовується для:
   5) Створення незалежного дво- чи трипозиційного прапорця;
   6) Створення головного меню;
   7) Створення контексного меню;
   8) Створення багаторядкового редактора тексту.
9. Властивість **ScrollBars** об'єкта **Memo** - це:
   1) Максимальна можлива кількість введених символів;
   2) Стан прапорця;
   3) Задання початкового тексту у полі редагування;
   4) Наявність смуг прокручування.
10. Властивість **Items** об'єкта **MainMenu** - це:
    5) Автоматичний виклик контексного меню;
    6) Розбиття меню у горизонтальному напрямку;
    7) Команди меню;
    8) Комбінація «гарячих» клавіш для виклику команди меню.
11. Властивість **ShortCut** об'єктів **PopurMenu** та **MainMenu** - це:
    1) Автоматичний виклик контексного меню;
    2) Розбиття меню у горизонтальному напрямку;
    3) Команди меню;
    4) Комбінація «гарячих» клавіш для виклику команди меню.
12. Властивість **ColCount** об'єкта **StringGrid**- це:
    1) Кількість рядків таблиці;
    2) Кількість стовпців таблиці;
    3) Кількість фіксованих стовпців таблиці, які не прокручуються зліва;
    4) Кількість фіксованих рядків таблиці, які не прокручуються вгору.
13. Властивість **FixedCols** об'єкта **StringGrid**- це:
    1) Кількість рядків таблиці;
    2) Кількість стовпців таблиці;
    3) Кількість фіксованих стовпців таблиці, які не прокручуються зліва;
    4) Кількість фіксованих рядків таблиці, які не прокручуються вгору.
14. Властивість **Style** об'єкта **ComboBox**- це:
    1) Кількість рядків у списку;
    2) Впорядкування за списком;
    3) Стиль оформлення і використання списку;
    4) Кількість рядків у випадаючому списку, які видимі без використання смуг прокручування

15. Властивість **Text** об’єкта **Edit** - це:
    1. Можливість змінити текст (доступність поля);
    2. Текст у полі редагування;
    3. Текст підказки, яка висвітлюється, якщо навести курсор миші;
    4. Висвітлювати/не висвітлювати підказку.
16. Об’єкт **Memo** використовується для:
    1. Створення незалежного дво- чи трипозиційного прапорця;
    2. Створення головного меню;
    3. Створення контексного меню;
    4. Створення багаторядкового редактора тексту.
17. Об’єкт **CheckBox** використовується для:
    1) Створення незалежного дво- чи трипозиційного прапорця;
    2) Створення головного меню;
    3) Створення контексного меню;
    4) Створення багаторядкового редактора тексту.
18. Властивість **MaxLength** об’єкта **Memo** - це:
    1) Максимальна можлива кількість введених символів;
    2) Стан прапорця;
    3) Задання початкового тексту у полі редагування;
    4) Наявність смуг прокручування.
19. Властивість **AutoPopup** об’єкта **PopurMenu** - це:
    1) Автоматичний виклик контексного меню;
    2) Розбиття меню у горизонтальному напрямку;
    3) Команди меню;
    4) Комбінація «гарячих» клавіш для виклику команди меню.
20. Об’єкт **StringGrid** призначений для:
    1) Створення кнопки панелі інструментів;
    2) Створення головного меню;
    3) Створення контексного меню;
    4) Створення у формі двовимірної таблиці символьних рядків.
21. Властивість **FixedRows** об’єкта **StringGrid**- це:
    1) Кількість рядків таблиці;
    2) Кількість стовпців таблиці;
    3) Кількість фіксованих стовпців таблиці, які не прокручуються зліва;
    4) Кількість фіксованих рядків таблиці, які не прокручуються вгору.
22. Об’єкт **ComboBox** призначений для:
    1) Створення кнопки панелі інструментів;
    2) Створення головного меню;
    3) Створення випадаючого списку;
    4) Створення у формі двовимірної таблиці символьних рядків.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Абрамов В. Г., Трифонов Н.П., Трифонова Г.Н. Введение в язык Паскаль. — М.: Наука, 1988.
2. Абрамов С.А., Гнездилова Г.Г., Капустина Е.Н., Селюн М.И. Задачи по программированию. —М.: Наука, 1988.
3. Антоненко В. М. Турбо Паскаль у прикладах і задачах. Навчальний посібник .-Ірпінь : Академія ДПС України, 2001.- 244 с
4. Архангельский А.Я. Программирование в Delphi 5.- М.:БИНОМ, 2000.- 1072 с.
5. Бобровский С Delphi 6: Учебный курс — СПб: Издательство "Питер", 2000. — 640 с: ил
6. Бондарев В.М., Рублинецкий В.И., Качко Е.Г. Основы программирования. - Харьков: Фолио, 1997. - 368 с.
7. Боон К. Паскаль для всех. — М.: Энергоатомиздат, 1988.
8. Бородин Ю.С., Вальвачев А.Н., Кузьмич А.И. Паскаль для персональних компьютеров. — Минск: Выш. шк., 1991.
9. Буч Г. Объектно-ориентированное проектирование с примерами применения. — Киев: Диалектика, М.: И.В.К , 1992
10. Вакалюк Т. А. Використання Інтернет-порталу e-olimp при проведенні занять з програмування у вищих навчальних закладах [Електронний ресурс] / Т. А. Вакалюк // Інформаційні технології і засоби навчання. – 2013. – №4 (36). – С. 84-97. – Режим доступу до журн. : http://journal.iitta.gov.ua/index.php/itlt/article/view/877/650
11. Вакалюк Т. А. Візуальне програмування : навчально-методичний посібник для студентів фізико-математичного факультету / Тетяна Анатоліївна Вакалюк – Житомир : Вид-во ЖДУ, 2013. – 116 с.
12. Вакалюк Т. А. Засвоєння загальної схеми розв'язування задач з програмування / Т. А. Вакалюк // Комп'ютер у школі та сім'ї. – № 7 (111). – 2013. – С. 7–10.
13. Вакалюк Т. А. Математичні основи розв’язування олімпіадних задач з інформатики на сайті e-olimp / Т. А. Вакалюк // Інформаційні технології в освіті : Збірник наукових праць. Випуск 7. – Херсон : Вид-во ХДУ, 2010. – С. 139–144.
14. Вакалюк Т. А. Підготовка майбутніх учителів інформатики до розвитку логічного мислення старшокласників : теоретико-методологічний аспект : Монографія. / Тетяна Анатоліївна Вакалюк. – Житомир: Вид-во ЖДУ імені І. Франка, 2013. – 236 с.
15. Вакалюк Т. А. Підготовка майбутніх учителів інформатики до тестування програмного забезпечення / Т. А. Вакалюк // Автоматизація та комп’ютерно-інтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку: матеріали Всеукраїнської науково-практичної Internet-конференції. – Черкаси, 2013. – С. 275-277.

16. Вакалюк Т. А. Програмування мовою Pascal: навчально-методичний посібник для студентів фізико-математичного факультету / Тетяна Анатоліївна Вакалюк. – Житомир: ФО-П Левковець Н.М., 2016. – 232 с.
17. Вакалюк Т. А. Програмування мовою С++. Структурне програмування (лабораторний практикум). Навчальний посібник для студентів фізико-математичного факультету. – Вид. 2-ге, виправ. та допов. / Тетяна Анатоліївна Вакалюк, Сергій Станіславович Жуковський. – Житомир : Вид-во ЖДУ, 2010. – 92 с.
18. Вакалюк Т. А. Програмування: курс лекцій. Навчальний посібник для студентів фізико-математичного факультету / Тетяна Анатоліївна Вакалюк. – Житомир : Вид-во ЖДУ, 2012. – 120 с.
19. Вакалюк Т. А. Розв'язування творчих задач з програмування майбутніми учителями інформатики / Т. А. Вакалюк // Вісник Чернігівського національного педагогічного університету імені Т.Г. Шевченка. – Вип. 113. – Чернігівський національний педагогічний університет імені Т.Г. Шевченка; гол. ред. Носко М.О. – Чернігів : ЧНПУ, 2013. – 210 с. (Серія: педагогічні науки) – С. 109-114
20. Вакалюк Т. А. Структурне програмування мовою Pascal (лабораторний практикум). Навчальний посібник для студентів фізико-математичного факультету. – Вид. 2-ге, виправ. та допов. / Тетяна Анатоліївна Вакалюк, Сергій Станіславович Жуковський. – Житомир : Вид-во ЖДУ, 2010. – 124 с.
21. Вакалюк Т. А. Технології тестування програм : навчально-методичний посібник для студентів фізико-математичного факультету / Тетяна Анатоліївна Вакалюк. – Житомир : Вид-во ЖДУ, 2013. – 96 с.
22. Вирт Н. Алгоритмы и структуры данных. — М.: Мир, 1989
23. Глинський Я.М., Анохін В.Є., Ряжська В.А. Паскаль. Turbo Pascal і Delphi. Навч. посібн. 8-ме вид. – Львів: «СПД Глинський», 2007. – 192 с.
24. Гофман В.Э., Хомоненко А.Д. Delphi 6. –СПб.:БХВ-Петербург, 2001.-1152с.
25. Грогоно П. Программирование на языке Паскаль,— М.; Мир, 1982.
26. Дарахвелидзе П.П., Марков И. Delphi - среда визуального программирования. СПБ «BNV - Санкт Петербург»,2000, 352 с.
27. Джефф Дантеманн, Джим Мишель, Дон Тайлер Программирование в среде Delphi. К.,МПФ «Диа Софт» 1995, 606 с.
28. Динамическое программирование (сборник задач с рекомендациями по их решению) / М.Г. Медведев, С.С. Жуковский, Т.А. Вакалюк. – Житомир: Издательство ФОП "О.О.Євенок", 2017. – 152 с.
29. Епанешников А., Епанешников В. Программирование в среде Turbo Pascal 7.0. -М.: Диалог-МИФИ, 1993.
30. Зуев, Программирование на языке Turbo Pascal 6.0, 7.0 М.:Радио и связь. Веста, 1993.

31. Інформатика: Комп'ютерна техніка. Комп'ютерні технології. Посіб. /За ред. О.І.Пушкаря — К.: Видавничий центр "Академія", 2001. — 696 с. (Альма-матер)
32. Климов Ю.С., Касаткин А.И., Мороз СМ. Программирование в среде Turbo Pascal 6.0. — Минск: Выш. шк., 1992.
33. Конопка П., Создание оригинальных компонент в среде Delphi. К.,1996, 571 с.
34. Культин Н.Б. Delphi 6. Программирование на Object Pascal. — СПб.: БХВ-Петербург, 2001. —528 с: ил
35. Макелви M.Visual Basic (серия "Без проблем"): пер. с антл.- М.:Бином,1996.- 576 с.
36. Мануйлов В.Г. Разработка программного обеспечения на Паскале. М., 1998, 240 с
37. Марченко А.И. Программирование в среде Борланд Паскаль 7.0. К., 1997. 476 с.
38. Мизрохи СВ. Turbo Pascal и объектно-ориентированное программирование — М.: Финансы и статистика, 1992.
39. Миллер Т., Пауэл Д. Специальное издание Использование Delphi. К., 2001,768 с.
40. Перминов О.Н. Программирование на языке Паскаль. — М.; Радио и связь, 1988.
41. Поляков Д.Б., Круглов И.Ю. Программирование в среде Турбо Паскаль (версия 5.5). —М.: Издательство МАИ, 1992.
42. Прайс Д. Программирование на языке Паскаль: практическое руководство. — М.: Мир, 1987.
43. Прокудин Г.С., Оленина Л.М. Компьютерная техника и программирование. Часть П. Алгоритмизация и программирование. - К: УФИМБ , 1998.
44. Спірін О. М. Початки алгоритмізації та процедурного програмування : метод. посіб. для студ. вищих пед. навч. закл-ів фізико-математичних спец-тей / О. М. Спірін, О. М. Кривонос. – Житомир: ЖДПУ, 2002. – 93 с..
45. Фаронов В.В. Delphi 6. Учебный курс. - М.:Издатель Молгачева С.В., 2001.-672 с.
46. Фаронов В.В. Турбо-Паскаль 7.0. Начальный курс: Учебное пособие. - М.: Нолидж, 1997.-616 с.
47. Федоров А.Г. Создание Windows - приложений в среде Delphi. М., ТОО «Компьютер пресс», 1995, 297с
48. Фокс Дж. Программное обеспечение и его разработка. Пер. с англ. — М.: Мир, 1985. —368 с.
49. Эрбс Х.-Э., Штольц О. Введение в программирование на языке Паскаль. — М.: Мир, 1989.

Навчальне видання

ВАКАЛЮК Тетяна Анатоліївна
ШЕВЧУК Лариса Дмитрівна
ПОСТОВА Світлана Анатоліївна

Структурне та візуальне програмування

Навчальний посібник для студентів
фізико-математичного факультету

Надруковано з оригінал-макета авторів

Підписано до друку 24.06.19. Формат 60х90/16. Папір офсетний.
Гарнітура Times New Roman. Друк різографічний.
Ум. друк. арк. 18,48. Обл. вид. арк. 12,35. Наклад 300. Зам. 125.

Виробник ФОП Домбровська Я. М., свідоцтво про
державну реєстрацію №2 340 000 0000 003646 від 15.07.2015 р.
08055, Київська обл., Макарівський р-он., с. Вільне,
e-mail: kalian_print@ukr.net