МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ФРАНКА

Укладачі:
Вербівський Дмитрій, Карплюк Світлана,
Вербовський Ігор

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ТА АРХІТЕКТУРА КОМП'ЮТЕРА

Навчально-методичний посібник

Житомир 2021

УДК 368.60/69 378:377.091.12011.3-368.60/69
ББК 74.200

Рекомендовано до друку Вченою радою Житомирського державного університету імені Івана Франка 28.12.2021 р., протокол № 24___

Рецензенти:

Ольга ГОРАЙ – кандидат педагогічних наук, доцент кафедри природничих і соціально-гуманітарних дисциплін Проректор з соціально-гуманітарного розвитку та міжнародного співробітництва
Олександр МАЄВСЬКИЙ – кандидат технічних наук, доцент кафедри комп'ютерних технологій і моделювання Поліського національного університету
Світлана ПОСТОВА – кандидат педагогічних наук, доцент кафедри комп'ютерних наук та інформаційних технологій

Вербівський Дмитрій, Карплюк Світлана, Вербовський Ігор.
Програмне забезпечення та архітектура комп'ютера: навч.-метод. посібн. / укладачі: Вербівський Дмитрій, Карплюк Світлана, Вербовський Ігор. – Житомир : Вид-во ЖДУ, 2017. – 157 с.

У запропонованому посібнику представлено теоретичний і практичний матеріал щодо викладання курсів із програмного забезпечення та архітектури комп'ютера для здобувачів вищої освіти. Його структура передбачає розгляд теоретичної, практичної, лабораторної та контролюючої частини. Завдяки цьому даний посібник може бути використаний різними категоріями педагогічних працівників, а також здобувачами вищої освіти. Представлено методичний інструментарій щодо специфіки побудови змістового наповнення занять на засадах використання програмного забезпечення та архітектури комп'ютера та особливостей його застосування у цілісному освітньому процесі закладів вищої освіти.

Навчально-методичний посібник адресовано науково-педагогічним працівникам, вчителям загальноосвітніх шкіл та здобувачам вищої освіти ЗВО.

УДК 368.60/69 378:377.091.12011.3-368.60/69

ЗМІСТ

ВСТУП

У суспільстві розвиток комп’ютерних та інформаційних технологій відбувається бурхливими темпами. В даний час людині складно зорієнтуватися у величезній багатоманітності різних програмних рішень, методів та технологій. Цей навчально-методичний посібник містить короткі відомості про сам комп’ютер, апаратний засіб для обслуговування програмного забезпечення, роботу операційних систем, прикладне програмне забезпечення різного призначення, вирішальні завдання управління ресурсами, впорядкувати знання в галузі інформаційних технологій та розглянути «зсередини» роботу ЕОМ та системного програмного забезпечення.

Посібник представлений п’ятьма розділами.

- Історія розвитку комп'ютерної техніки.
- Ресурси ЕОМ та архітектура комп’ютера .
- Операційні системи.
- Системи програмування.
- Прикладне програмне забезпечення.

У першому розділі розглядаються історичний аспект виникнення комп’ютерної техніки.

Другий розділ присвячено основам архітектури ЕОМ та види їх ресурсів. У ньому наводяться принципи Джеймса-фон-Неймана і докладно описується пристрій сучасних ЕОМ.

У третьому – поняття «Операційна система» та її функції. Також наводяться класифікація та призначення операційних систем.

Четвертий розділ містить основні підходи управління ресурсами. Розглядаються питання та проблеми управління процесорним часом (управління процесами), описуються методи управління оперативною пам’яттю ЕОМ, підходи управління вводом/виводом.

П’ятий розділ містить огляд видів прикладного програмного забезпечення та їх призначення.

1. ІСТОРІЯ РОЗВИТКУ КОМП'ЮТЕРНОЇ ТЕХНІКИ

Історія рахункових пристроїв налічує багато століть. Найдавнішим рахунковим інструментом, який сама природа надала в розпорядження людини, була його рука. Для полегшення рахунку люди почали використовувати пальці спочатку однієї руки, потім обох, а деяких племенах і пальці ніг.

1.1. Перші обчислювальні машини.

У 1623 р. Вільгельм Шиккард – професор Тюбинського університету описав пристрій «годин для рахунку». Це була перша механічна машина, яка могла тільки складати та віднімати. В наш час за його описом побудовано її модель.

У 1642 р. французький математик Блез Паскаль (1623-1662) сконструював лічильний устрій, щоб полегшити працю свого батька – податкового інспектора. Цей пристрій дозволяв підсумовувати десяткові числа. Зовні воно було ящиком з численними шестернями. Основою підсумовуючої машини став лічильник-реєстратор, або лічильна шестерня. Вона мала десять виступів, на кожному з яких було нанесено цифри.

Для передачі десятків на шестірні розташовувався один подовжений зуб, що зачіпляв і повертає проміжну шестірню, яка передавала обертання шестірні десятків. Додаткова шестерня була необхідна для того, щоб обидві рахункові шестірні – одиниць та десятків – оберталися в одному напрямку. Рахункова шестерня за допомогою храпового механізму (що передає прямий рух і не передає зворотного) з'єднувалися з важелем. Відхилення важеля на той чи інший кут дозволяло вводити в лічильник однозначні числа

та підсумовувати їх. У машині Паскаля храповий привід був приєднаний до всіх лічильників, що дозволяло підсумовувати і багатозначні числа.

У 1673 р. німецький філософ, математик, фізик Готфрід Вільгельм Лейбніц (1646-1716) створив "ступінчастий обчислювач" - лічильну машину, що дозволяє складати, віднімати, множити, ділити, витягувати квадратне коріння, при цьому використовувалася двійкова система числення. Це був більш досконалий прилад, в якому використовувалася частина (прообраз каретки) і ручка, за допомогою якої оператор обертав колесо. Машина була прототипом арифмометра, що використовується з 1820 до 60-х років XX століття.

У 1804 р. французький винахідник Жозеф Марі Жаккар (1752-1834) придумав спосіб автоматичного контролю над ниткою під час роботи на ткацькому верстаті. Робота верстата програмувалася за допомогою цілої колоди перфокарт, кожна з яких управляла одним ходом човна. Переходячи до нового малюнку, оператор просто заміняв одну колоду перфокарт на іншу. Створення ткацького верстата, керованого картами з пробитими на них отворами і з'єднаними один з одним у вигляді стрічки, відноситься до одного з ключових відкриттів, що зумовили розвиток обчислювальної техніки.

Чарльз Ксав'єр Томас (1785-1870) у 1820р. створив перший механічний калькулятор, який міг не тільки складати та множити, а й віднімати та ділити. Бурхливий розвиток механічних калькуляторів призвело до того, що до 1890 додався ряд корисних

функцій: запам'ятовування проміжних результатів з використанням їх у наступних операціях, друк результату і т.п. Створення недорогих, надійних машин дозволило використовувати їх для комерційних цілей та наукових розрахунків.

У 1822р. англійський математик Чарлз Беббідж (1792-1871) висунув ідею створення програмно-керованої лічильної машини, що має арифметичний пристрій, пристрій управління, введення та друку. Перша спроектована Беббіджем машина, різнизна машина, працювала на паровому двигуні. Вона вираховувала таблиці логарифмів методом постійної диференціації та заносила результати на металеву пластину. Працююча модель, яку він створив у 1822 році, була шестицифровим калькулятором, здатним проводити обчислення та друкувати цифрові таблиці.

Вже нашого часу аналітичну машину Беббиджа побудували ентузіасти з Лондонського музею науки. Вона складається з чотирьох тисяч залізних, бронзових та сталевих деталей та важить три тонни. Щоправда, користуватися нею дуже важко – при кожному обчисленні доводиться кілька сотень (а то й тисяч) разів крутити ручку автомата. Числа записуються (набираються) на дисках, розташованих по вертикалі і встановлених положення від 0 до 9. Двигун приводиться в дію послідовністю перфокарт, що містять інструкції (програму).

Поруч із британським ученим працювала леді Ада Лавлейс (1815-1852). Вона розробила перші програми для машини, заклала багато ідей і ввела ряд понять і термінів, що збереглися до нашого часу. Леді Лавлейс була єдиною дочкою Джорджа Гордона

Байрона. Вона передбачила появу сучасних комп'ютерів як багатофункціональних машин як для обчислень, але й роботи з графікою, звуком. У середині 70-х років. нашого сторіччя міністерство оборони США офіційно затвердило назву єдиної мови програмування американських збройних сил. Мова називається Ada. З недавнього часу у програмістів усього світу з'явилося своє професійне свято. Він так і називається - "День програміста" - і святкується 10 грудня. Якраз на день народження Ади Лавлейс.

У 1855 р. брати Джорж і Едвард Шутц зі Стокгольма збудували перший механічний комп'ютер, використовуючи роботи Ч. Беббіджа.

У 1878 р. російський математик і механік Пафнутий Львович Чебишев створює підсумовуючий апарат з безперервною передачею десятків, а 1881 року – приставку до нього для множення та ділення.

У 1880р. Вільгодт Теофілович Однер, швед за національністю, який жив у Санкт-Петербурзі, сконструював арифмометр. Його арифмометри відрізнялися надійністю, середніми габаритами та зручністю в роботі. Над арифмометром Однер почав працювати 1874 року, а 1890 року він налагоджує масовий випуск арифмометрів. Їхня модифікація «Фелікс» випускалася до 50-х років XX століття.

1.2. Початок XX ст. та друга половина XX ст.

1918 рік. Російський вчений М.А. Бонч-Бруєвич та англійські вчені В. Ікклз та Ф. Джордан (1919) незалежно один від одного

створили електронне реле, назване англійцями тригером, яке зіграло велику роль у розвитку комп'ютерної техніки.

У 1930р. Віннівер Буш (1890–1974) конструює диференціальний аналізатор. По суті це перша успішна спроба створити комп'ютер, здатний виконувати громіздкі наукові обчислення. Роль Буша в історії комп'ютерних технологій дуже велика, але найчастіше його ім'я спливає у зв'язку з пророчою статтею As We May Think (1945), в якій він описує концепцію гіпертексту.

У 1937 році гарвардський математик Говард Ейкен запропонував проект створення великої лічильної машини. Спонсував роботу президент компанії IBM Томас Вотсон, який вклав у неї 500 тис. $. Проектування Mark-1 почалося 1939 року, будувало цей комп'ютер нью-йоркське підприємство IBM. Комп'ютер містив близько 750 тис. деталей, 3304 реле та понад 800 км дротів.

У 1946 року Джон фон Нейман запропонував низку нових ідей організації ЕОМ, зокрема концепцію збереженої програми, тобто. зберігання програми в запам'ятовуючому пристрої. Через війну реалізації ідей фон Неймана було створено архітектура ЕОМ, багато в чому збереглася до нашого часу.

У 1947 році з'явилася лічильна машина Mark-2, яка була першою багатозадачною машиною – наявність декількох шин дозволяло одночасно передавати з однієї частини комп'ютера в іншу кілька чисел. 23 грудня 1947р. співробітники Bell Telephone Laboratories Джон Бардін та Уолтер Бремен вперше

продемонстрували свій винахід, який отримав назву транзистор. Цей пристрій через десять років відкрило нові можливості.

1948 року академіком С.А. Лебедєвим (1890-1974) та Б.І. Рамєєвим був запропонований перший проект вітчизняної цифрової електронно-обчислювальної машини: спочатку МЕСМ – мала електронна лічильна машина (1951 рік, Київ), потім БЕСМ – швидкодіюча електронна лічильна машина (1952 рік, Москва). Паралельно з ними створювалися Стріла, Урал, Мінськ, Роздан, Наїрі.

У 1951 році в Англії з'явилися перші серійні комп'ютери Ferranti Mark-1 та LEO-1. А через 5 років фірма Ferranti випустила ЕОМ Pegasus, у якій вперше знайшла втілення концепція регістрів загального призначення. Джей Форрестер запатентував пам'ять на магнітних сердечниках.

Вперше така пам'ять використана на машині Whirlwind-1. Вона була два куби з 32х32х17 сердечниками, які забезпечували зберігання 2048 слів для 16-розрядних двійкових чисел з одним розрядом контролю парності. У цій машині була вперше використана універсальна неспеціалізована шина (взаємозв'язки між різними пристроями комп'ютера стають гнучкими) і як системи вводу-виводу використовувалися два пристрої: електронно-променева трубка Вільямса і машинка з перфострічкою (флексорайтер).

У 1952р. розпочалася дослідна експлуатація вітчизняного комп'ютера БЭСМ-1.

У СРСР 1952-1953 роках А.А.Ляпунов розробив операторний

метод програмування (операторне програмування), а 1953-1954 роках Л.В.Канторович – концепцію великоблочного програмування.

У 1955 році побачив світ першу алгоритмічну мову FORTRAN (FORmule TRANslator – перекладач формул). Він використовувався для вирішення науково-технічних та інженерних завдань та розроблений співробітниками фірми IBM під керівництвом Джона Бекуса.

У 1958р. Джек Кілбі з Texas Instruments і Роберт Нойс із Fairchild Semiconductor незалежно один від одного винаходять інтегральну схему.

1959 р. під керівництвом С.А. Лебедєва створено машину БЭСМ-2 продуктивністю 10 тис. опер./с. З її застосуванням пов'язані розрахунки запусків космічних ракет і перших у світі штучних супутників Землі, а потім машина М-20 – для свого часу одна з найшвидших у світі (20 тис. опер./с.).

У 1960 року з'явився ALGOL (Algoritmic Language – алгоритмічний мову), орієнтований наукове застосування. У нього введено багато нових понять, наприклад, блокова структура. Ця мова стала концептуальною основою багатьох мов програмування. Тринадцять європейських та американських фахівців із програмування у Парижі затвердили стандарт мови програмування ALGOL-60.

1963 – початок випуску ЕОМ «Мінськ-32» із зовнішньою пам'яттю на змінних магнітних дисках. З'явилися машини другого покоління, побудовані на напівпровідниковій елементній базі – на

магнітних елементах. Так, у МДУ ім. М.В. Ломоносова колективом під керівництвом Н.П. Брусенцова була створена машина Сетунь (що вироблялася серійно у 1962-1964 роках).

У 1964р. Співробітник Стенфордського дослідницького центру Дуглас Енгельбарт продемонстрував роботу першої миші-маніпулятора. Фірма IBM оголосила про створення шести моделей сімейства IBM 360 (System 360), що стали першими комп'ютерами третього покоління. Моделі мали єдину систему команд і відрізнялися один від одного обсягом оперативної пам'яті та продуктивністю.

У 1967р. під керівництвом С.А.Лебедєва та В.М.Мельникова в ІТМ та ВТ створено швидкодіючу обчислювальну машину БЭСМ-6. За ним відбувся «Ельбрус» – ЕОМ нового типу, продуктивністю 10 млн. опер./с.

1968р. США фірма «Барроуз» випустила першу швидкодіючу ЕОМ на БИСах (великих інтегральних схемах).

9 жовтня 1969 прийнято вважати днем народження Мережі. Цього дня була зроблена перша, щоправда, не зовсім вдала, спроба дистанційного підключення до комп'ютера, що знаходився в дослідному центрі Стенфордського університету (SRI), з іншого комп'ютера, що стояв у Каліфорнійському університеті в Лос-Анджелесі (UCLA). Видалені один від одного на відстань 500 кілометрів, SRI та UCLA стали першими вузлами майбутньої мережі ARPANet.

У 1971р. фірмою Intel створено перший мікропроцесор (МП) - програмований логічний пристрій, виготовлений за технологією

НВІС. З'явився комп'ютер IBM/370 модель 145 – перший комп'ютер, основний пам'яті якого використовувалися виключно інтегральні схеми. У світ виходить перший кишеньковий калькулятор Poketronic.

Денніс Рітчі з Bell Lab's розробив мову програмування "С" (Сі). Так його назвали тому, що попередня версія мала назву «В».

1974 Фірма Intel розробила перший універсальний восьмирозрядний мікропроцесор 8080 з 4500 транзисторами.

У 1975р. Джин Амдал розробив комп'ютер четвертого покоління на ВІС – AMDAL-470 V/6. Гаррі Кілдалл із фірми Digital Reseach розробив операційну систему CP/M. Молодий програміст Пол Аллен та студент Гарвардського університету Білл Гейтс реалізували для Альтаїра мову Бейсік. Згодом вони заснували фірму Microsoft, яка сьогодні є найбільшим виробником програмного забезпечення.

У 1976р. молоді американці Стів Джобс та Стів Возняк організували підприємство з виготовлення персональних комп'ютерів Apple, призначених для великого кола непрофесійних користувачів.

1981 р. Фірма IBM випустила перший персональний комп'ютер IBM PC з урахуванням мікропроцесора 8088. 1982 р. Фірма Intel випустила мікропроцесор 80286.

У 1982 році було покладено початок знаменитої серії x86. 16-розрядний мікропроцесор Intel 80286 на базі 134 тис. транзисторів за продуктивністю втричі випереджав моделі конкурентів. Відмінною особливістю цієї розробки було те, що вперше

реалізований принцип програмної сумісності з процесорами наступних поколінь за рахунок вбудованих засобів управління пам'яттю.

1984 р. Корпорація Apple Computer випустила комп'ютер Macintosh - першу модель знаменитого згодом сімейства Macintosh с зручною для користувача операційною системою, розвиненими графічними можливостями, набагато перевершують у той час ті, якими мали стандартні IBM-сумісні ПК з MS-DOS. Ці комп'ютери швидко придбали мільйони шанувальників і стали обчислювальною платформою для цілих галузей, таких як видавнича справа та освіта.

Sony та Philips розробляють стандарт CD-ROM-стандарт запису компакт-дисків. Також розроблено стандарти MIDI та DNS. Компанія IBM випустила персональний комп'ютер IBM PC/AT.

1985р. фірма Intel випустила 32-бітний процесор 80386, що складається з 250 тис. транзисторів. Фірма Microsoft випустила першу версію графічного операційного середовища Windows. Того ж року сталася поява нової мови програмування C++.

У 1989 р. Intel випускає черговий чіп - 80486. Це перший процесор із кількістю транзисторів, що перевищує 1 млн. Microsoft випустила текстовий процесор WORD. Розроблено формат графічних файлів GIF.

У 1992р. з'явилася перша безкоштовна операційна система з більшими можливостями – Linux. Фінський студент Лінус Торвальдс вирішив поекспериментувати із командами процесора Intel 386 і те, що вийшло, виклав у Internet. Сотні програмістів із

різних країн світу стали дописувати та переробляти програму. Вона перетворилася на повнофункціональну діючу операційну систему.

У 1993р. фірма Intel випустила 64-розрядний мікропроцесор Pentium, який складався з 3,1 млн транзисторів і міг виконувати 112 млн операцій в секунду. З'явився формат стиснення MPEG.

1.3. Покоління ЕОМ.

Розвиток ЕОМ ділиться кілька періодів. Покоління ЕОМ кожного періоду відрізняються один від одного елементною базою та математичним забезпеченням.

Перше покоління ЕОМ.

Перше покоління (1945 - 1958) ЕОМ було побудовано на електронних лампах - діодах та тріодах. Більшість машин першого покоління були експериментальними пристроями та будувалися з метою перевірки тих чи інших теоретичних положень. Застосування вакуумно-лампової технології, використання систем пам'яті на ртутних лініях затримки, магнітних барабанах, електронно-променевих трубках (трубках Вільямса), робило їхню роботу дуже ненадійною. Крім цього, такі ЕОМ мали велику вагу і займали площею значні території, іноді цілі будівлі. Для введення-виведення даних використовувалися перфострічки та перфокарти, магнітні стрічки та друкуючі пристрої.

Була реалізована концепція програми, що зберігається. Програмне забезпечення комп'ютерів 1-го покоління складалося переважно зі стандартних підпрограм, швидкодію вони мали від 10 до 20 тис. оп./сек.

Машини цього покоління: ENIAC (США), МЕСМ (СРСР), БЕСМ-1, М-1, М-2, М-3, "Стріла", "Мінськ-1", "Урал-1", "Урал-2" », «Урал-3», М-20, «Сетунь», БЭСМ-2, «Роздан», IBM -701, використовували багато електроенергії та складалися з дуже великої кількості електронних ламп. Наприклад, машина «Стріла» складалася з 6400 електронних ламп та 60 тис. штук напівпровідникових діодів. Їхня швидкодія не перевищувала 2-3 тис. операцій на секунду, оперативна пам'ять не перевищувала 2 Кб. Тільки машина «М-2» (1958) оперативна пам'ять була 4 Кб, а швидкодія 20 тис. операцій на секунду.

Друге покоління ЕОМ.

ЕОМ 2-го покоління було розроблено 1959 — 1967 гг. Як основний елемент були використані вже не електронні лампи, а напівпровідникові діоди і транзистори, а як пристрої пам'яті стали застосовуватися магнітні сердечники і магнітні барабани - далекі предки сучасних жорстких дисків. Комп'ютери стали надійнішими, швидкодія їх підвищилася, споживання енергії зменшилося, зменшилися габаритні розміри машин.

З появою пам'яті на магнітних сердечниках цикл її роботи зменшився до десятків мікросекунд. Головний принцип структури – централізація. З'явилися високопродуктивні пристрої роботи з магнітними стрічками, пристрої пам'яті на магнітних дисках.

Крім цього, з'явилася можливість програмування алгоритмічними мовами. Було розроблено перші мови високого рівня – Фортран, Алгол, Кобол. Швидкодія машин 2-го покоління вже сягала 100-5000 тис. оп./сек.

Приклади машин другого покоління: БЭСМ-6, БЭСМ-4, Мінськ-22 – призначені на вирішення науково-технічних і планово-экономических завдань; Мінськ-32(СРСР), ЕОМ М-40, -50 – для систем протиракетної оборони; Урал -11, -14, -16 – ЕОМ загального призначення, зорієнтовані рішення інженерно-технічних завдань.

Третє покоління ЕОМ.

У ЕОМ третього покоління (1968 - 1973 рр..) Використовувалися інтегральні схеми. Розробка в 60-х роках інтегральних схем - цілих пристроїв і вузлів з десятків і сотень транзисторів, виконаних на одному кристалі напівпровідника (те, що називають мікросхемами) призвело до створення ЕОМ 3-го покоління. У цей же час з'являється напівпровідникова пам'ять, яка й досі використовується в персональних комп'ютерах як оперативна. Застосування інтегральних схем набагато збільшило можливості ЕОМ.

Тепер центральний процесор отримав можливість паралельно працювати та керувати численними периферійними пристроями. ЕОМ могли одночасно обробляти кілька програм (принцип мультипрограмування). Внаслідок реалізації принципу мультипрограмування з'явилася можливість роботи в режимі поділу часу в діалоговому режимі. Віддалені від ЕОМ користувачі отримали можливість, незалежно друг від друга, оперативно взаємодіяти з машиною.

Комп'ютери проектувалися на основі інтегральних схем малого ступеня інтеграції (MIC – 10-100 компонентів на кристал) та середнього ступеня інтеграції (CIC – 10-1000 компонентів на

кристал). З'явилася ідея, яка і була реалізована, проектування сімейства комп'ютерів з тією ж архітектурою, в основу якої покладено головним чином програмне забезпечення. Наприкінці 60-х з'явилися міні-комп'ютери. 1971 року з'явився перший мікропроцесор. Швидкодія комп'ютерів 3-го покоління досягла близько 1 млн. оп/сек.

У ці роки виробництво комп'ютерів набуває промислового розмаху. Починаючи з ЕОМ 3-го покоління, традиційним стала розробка серійних ЕОМ. Хоча машини однієї серії сильно відрізнялися один від одного за можливостями та продуктивністю, вони були інформаційно, програмно та апаратно сумісні. Найпоширенішим у роки було сімейство System/360 фірми IBM. Країнами РЕВ були випущені ЕОМ єдиної серії «ЄС ЕОМ»: ЄС-1022, ЄС-1030, ЄС-1033, ЄС-1046, ЄС-1061, ЄС-1066 та ін. До ЕОМ цього покоління також відноситься «IBM-370» "Електроніка-100/25", "Електроніка-79", "СМ-3", "СМ-4" та ін.

Для серій ЕОМ було розширено програмне забезпечення (операційні системи, мови програмування високого рівня, прикладні програми тощо.). У 1969 році одночасно з'явилися операційна система Unix і мова програмування С («Сі»), що вплинули на програмний світ і досі зберігають своє передове положення.

Четверте покоління ЕОМ.

У комп'ютерах четвертого покоління (1974 - ... рр..), Використання великих інтегральних схем (ВІС - 1000-100000 компонентів на кристал) і надвеликих інтегральних схем (СВІС -

100000-10000000 компонентів на кристал), збільшило їх швидкодію до 1000000. оп./сек.

Початком цього покоління вважають 1975 рік – фірма Amdahl Corp. випустила шість комп'ютерів AMDAHL 470 V/6, у яких були застосовані ВІС як елементну базу. Стали використовуватися швидкодіючі системи пам'яті на інтегральних схемах - МОП ЗУПВ ємністю кілька мегабайт. У разі вимкнення машини дані, що містяться в МОП ЗУПВ, зберігаються шляхом автоматичного перенесення на диск. При включенні машини запуск системи здійснюється за допомогою програми самозавантаження, що зберігається в ПЗУ (постійний запам'ятовуючий пристрій), що забезпечує вивантаження операційної системи та резидентного програмного забезпечення в МОП ЗУПВ.

Розвиток ЕОМ 4-го покоління пішло за двома напрямками:

Створення суперЕОМ – комплексів багатопроцесорних машин. Швидкодія таких машин сягає кількох мільярдів операцій на секунду. Вони здатні обробляти великі масиви інформації. Сюди входять комплекси ILLIAS-4, CRAY, CYBER, «Ельбрус-1», «Ельбрус-2» та ін. всього, у оборонній галузі.

Подальший розвиток на основі БІС та НВІС мікро-ЕОМ та персональних ЕОМ (ПЕОМ) . Першими представниками цих машин є комп'ютери фірми Apple, IBM - PC (XT, AT, PS/2), вітчизняні "Іскра", "Електроніка", "Мазовія", "Агат", "ЄС-1840", "ЄС-1841" та ін. Починаючи з цього покоління ЕОМ стали називати комп'ютерами. Програмне забезпечення доповнюється базами та банками даних.

2. ОСНОВИ АРХІТЕКТУРИ ЕОМ. РЕСУРСИ ЕОМ

2.1. Апаратні засоби ЕОМ

Обчислювальна техніка – це сукупність пристроїв, призначених для автоматичного або автоматизованого оброблення даних. Під обчислювальною системою розуміють конкретний набір пристроїв і програм, що взаємодіють між собою, призначений для обслуговування однієї робочої ділянки. Центральним засобом більшості обчислювальних систем є комп'ютер.

Комп'ютер – це електронний пристрій, призначений для автоматизації роботи з інформацією. Комп'ютер також часто називають електронно-обчислювальною машиною (ЕОМ).

Склад обчислювальної системи називається конфігурацією. Розрізняють апаратну конфігурацію, куди входять апаратні засоби обчислювальної техніки, і програмну конфігурацію, куди входять програмні засоби обчислювальної техніки. Відповідно в обчислювальній системі прийнято розглядати:

Апаратне забезпечення – це сукупність пристроїв, які можуть входити до складу комп'ютера або підключатися до нього. Набір таких пристроїв утворює апаратну конфігурацію.

Програмне забезпечення – це сукупність програм, які можуть застосовуватися на комп'ютері для реалізації інформаційних процесів. Набір таких програм на комп'ютері утворює програмну конфігурацію. Під комп'ютерною програмою розуміється впорядкована послідовність команд комп'ютера.

Іноді також розглядають окремо:

Інформаційне забезпечення як об'єднання комп'ютерних програм і попередньо підготовлених даних для обробки.

Математичне забезпечення, до якого можна віднести сукупність алгоритмів, методів, моделей та інших ідей, що використовуються для вирішення завдань на комп'ютері.

Комп'ютери можуть бути класифіковані за різними ознаками, але найчастіше згадується класифікація за такими ознаками.

За призначенням комп'ютери поділяються на:

1) надвеликі ЕОМ (супер-ЕОМ, суперкомп'ютери);

2) великі ЕОМ (мейнфрейми);

3) малі ЕОМ (міні-ЕОМ);

4) надмалі ЕОМ (мікро-ЕОМ);

5) персональні комп'ютери;

По елементній базі комп'ютери бувають:

1) на електронних радіолампах;

2) на транзисторах;

3) на мікросхемах;

4) на надвеликих інтегральних схемах.

За принципом дії комп'ютери поділяються на:

1) аналогові обчислювальні машини (АВМ);

2) цифрові обчислювальні машини (ЦОМ);

3) гібридні обчислювальні машини (ГВМ).

У мережі комп'ютери можуть виступати як:

1) сервера;

2) робочої станції;

3) терміналу.

За ступенем доступності комп'ютери поділяються на:

1) колективні ЕОМ (з безліччю терміналів);

2) персональні ЕОМ чи персональні комп'ютери.

2.2. Основи архітектури ЕОМ

Архітектура ЕОМ – найбільш загальні принципи побудови ЕОМ, що реалізують програмне управління роботою та взаємодією основних її функціональних вузлів.

Загальні принципи побудови ЕОМ, що належать до архітектури.

1. Структура пам’яті ЕОМ.
2. Способи доступу до пам’яті та зовнішніх пристроїв.
3. Можливість зміни конфігурації.
4. Система команд.
5. Формати даних.
6. Організація інтерфейсу.

Класичні принципи побудови архітектури ЕОМ були запропоновані у роботі Дж. фон Неймана, Г. Голдстейга та А. Беркса у 1946 р. і відомі як «принципи фон Неймана».

Вони зводяться до чотирьох положень [1]:

1. Використання двійкової системи представлення даних.
2. Принцип програми, що зберігається.
3. Принцип послідовного виконання операцій.

4. Принцип довільного доступу до осередків оперативної пам’яті.

Розглянемо їх окремо.

Перше положення – використання двійкової системи представлення даних. Будь-яка інформація на ЕОМ зберігається як послідовності нулів і одиниць. Такий спосіб подання інформації для машин найбільш зручний, оскільки вона легко обробляється («нуль» – немає сигналу, «одиниця» – сигнал є). Для двійкової системи існує відповідний апарат обчислень, описаний практично в будь-якому підручнику з інформації.

Друге положення - принцип програми, що зберігається. Як дані програма зберігається у вигляді нулів та одиниць, тобто принципова різниця між програмою (виконуваним кодом та даними) відсутня. Фон Нейманом було запропоновано наступну структуру логічного устрою ЕОМ (рис. 1).

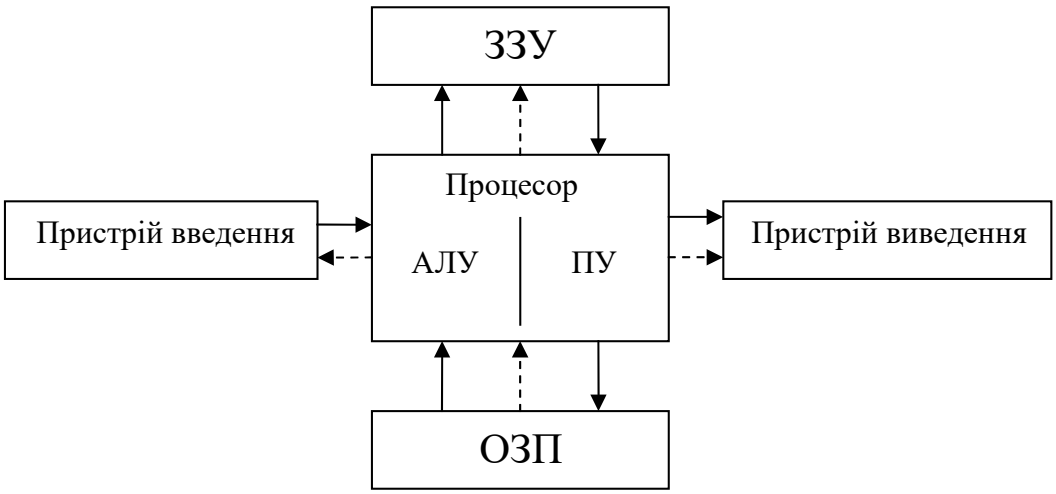


Рис. 1. Схема логічного устрою ЕОМ

У сучасних ЕОМ процесор складається з арифметико-логічного пристрою (АЛУ) та пристрою управління (ПУ). Він здійснює керування пристроями вводу/виводу, зовнішніми запам’ятовуючими пристроями (ЗЗП) і оперативним запам’ятовуючим пристроєм (ОЗУ). Також процесор здійснює

обробку інформації, що надходить із пристроїв. Усі види запам'ятовуючих пристроїв (ЗП) зберігають інформацію (дані) та коди програм. У сучасних ЕОМ ЗП «багатоярусні». ОЗП зберігає інформацію, з якою ЕОМ працює безпосередньо зараз. ЗЗУ забезпечує зберігання інформації більших обсягів, ніж ОЗП, але при цьому працює значно повільніше.

Третє положення – принцип послідовного виконання операцій. Одним і тим же пристроєм, наприклад, процесором операції виконуються послідовно один за одним. Наприклад, якщо процесор здійснює обчислення, а користувач у цей час вводить дані з клавіатури, то обчислення перериваються, здійснюється обробка введених символів і лише потім поновлюються обчислення.

Четверте становище – принцип довільного доступу до осередків оперативної пам'яті. Структурно основна пам'ять складається з пронумерованих осередків. Процесору в довільний момент часу доступна будь-яка комірка. Слід надавати імена областям пам'яті так, щоб до запам'ятованих у них значень можна було б згодом звертатися або змінювати їх у процесі виконання програми з використанням наданих імен.

2.3. Влаштування сучасного персонального комп'ютера

Розглянувши загальну концепцію пристрою комп'ютера, перейдемо до вивчення складу сучасних персональних комп'ютерів (ПК).

Основними характеристиками ПК є:

1) тактова частота процесора;

2) об'єм оперативної пам'яті;

3) об'єм пам'яті жорсткого диска.

Практично кожен комп’ютер складається із системного блоку, монітора, клавіатури, маніпуляторів та периферії.

Системний блок основний, а то й головний блок ЕОМ [2]. У ньому містяться центральний процесор, материнська плата, контролери, накопичувачі тощо.

Усі апаратні засоби стосовно системного блоку поділяються на:

- внутрішні пристрої (всередині системного блоку).
- зовнішні пристрої (поза системним блоком): монітор,
- клавіатура, миша та всі периферійні пристрої.

Для забезпечення інформаційної взаємодії між процесором та накопичувачами, зовнішніми пристроями тощо існують відповідні контролери. Керування монітором процесор здійснює через відеоконтролер або відеокарту. Жорсткі диски та інші носії інформації спілкуються з процесором через контролер накопичувачів. І так далі. Фактично контролер можна розглядати як спеціалізований процесор, який керує роботою «ввіреного йому» зовнішнього пристрою за спеціальними вбудованими програмами обміну. Такий процесор має власну систему команд. Наприклад, контролер накопичувача на гнучких магнітних дисках (дисковода) вміє позиціонувати головку на потрібну доріжку диска, читати або записувати сектор, форматувати доріжку та інше.

Таким чином, наявність інтелектуальних зовнішніх пристроїв може суттєво змінювати ідеологію обміну. Центральний процесор

у разі необхідності здійснити обмін видає завдання на його здійснення контролеру. Подальший обмін інформацією може протікати під керівництвом контролера без центрального процесора. Останній отримує можливість «займатися своєю справою», тобто продовжувати виконання командних кодів програми.

У материнській платі, а також у системному блоці (у торці) знаходяться роз'єми, які забезпечують фізичне приєднання пристроїв. Логічно роз'єми характеризуються портами. Останні виконують дві функції.

1. Служать посередником при передачі даних між комп'ютером та пристроями вводу/виводу.

2. Видають процесору сигнал переривання, за яким розпочинається процес переривання.

Для зв'язку між окремими функціональними вузлами ЕОМ використовується загальна шина або магістраль. Шина складається із трьох частин.

1. Шина даних, через яку передається інформація.
2. Шина адреси, яка визначає, куди передаються дані.
3. Шина керування, що регулює обмін інформацією.

Вищенаведені тези можна віднести до наступної схеми пристрою сучасного ПК та потоків даних (рис. 2).

Рис. 2. Схема влаштування сучасного ПК

Магістрально-модульний принцип функціонування ЕОМ означає, що до системної шини (магістралі) можуть підключатися різні пристрої, які називаються модулями.

Обов'язково до шини повинні підключатися:

1) процесор (у якому поєднуються функції АЛУ, УУ, кеш-пам'яті та регістрової пам'яті);

2) пристрої введення (зазвичай клавіатура та миша);

3) пристрої виведення (зазвичай монітор);

4) ПЗП (постійний пристрій);

6) ОЗП (оперативний пристрій);

7) ЗЗП (зовнішні пристрої).

ПЗП і пристрої введення передають інформацію в інші

пристрої через шину, а пристрої виведення отримують інформацію через неї від інших пристроїв. Інші пристрої можуть бути як приймачами, так і передавачами інформації.

Пояснимо, що таке CMOS та BIOS.

CMOS (Complementary Metal-Oxide Semiconductor) – спеціальна пам’ять, що має низьке енергоспоживання, і призначена для зберігання параметрів конфігурації комп’ютера.

BIOS (Basic Input-Output System) – постійна пам’ять (ПЗП), призначена для зберігання спеціальної програми, що забезпечує перевірку обладнання ЕОМ, ініціалізацію завантаження операційної системи та виконання базових функцій з обслуговування пристроїв комп’ютера. У BIOS міститься також програма конфігурації комп’ютера (SETUP). Вона дозволяє встановити деякі характеристики пристроїв комп’ютера.

Ресурси ЕОМ: тактова частота процесора, кеш-пам’ять першого та другого рівнів, розмір оперативної пам’яті, ємність жорсткого диска, апаратні переривання, частота системної шини тощо. Ідентифікуються вони двома способами: перед завантаженням операційної системи (натисканням клавіші Pause), в операційній системі за допомогою спеціальних програм, що діагностують. Також ресурси комп’ютера можна визначити за рядком специфікацій. Подібні рядки найчастіше наводяться у прейскурантах цін на обчислювальну техніку. Під час формування таких рядків певними стандартами не користуються. Тим не менш, з вмісту рядка специфікації можна легко визначити всі ресурси ЕОМ, що цікавлять.

3. ОПЕРАЦІЙНІ СИСТЕМИ

3.1. Види програмного забезпечення

Програма – це послідовність команд комп'ютера, що призводить до вирішення поставленого завдання.

Програмний продукт – це комплекс комп'ютерних програм, що надають широкий спектр можливостей з певної тематики і використовуються як промисловий товар.

Програмне забезпечення (ПЗ) – це сукупність програм, які можуть виконуватися на комп'ютері.

Програма-сервер – це програма, що надає будь-які послуги з комп'ютерної мережі.

Програма-клієнт – це програма, яка отримує будь-які послуги з комп'ютерної мережі.

Пакет програм – це сукупність програм із подібним інтерфейсом, об'єднаних виробником, які виконують різні функції з певної тематики.

Наприклад, до складу офісного пакета Microsoft Office входять:

1) текстовий процесор Microsoft Word;

2) табличний процесор Microsoft Excel;

3) система керування базами даних Microsoft Access;

4) програма презентаційної графіки Microsoft PowerPoint.

За сферою використання на ПК комп'ютерні програми бувають:

1) системні, які забезпечують роботу всіх систем ПК;

2) прикладні, які дозволяють прикладати можливості ПК до будь-якої предметної області;

3) інструментальні, що містять інструменти для створення нових програм.

Програмне забезпечення поділяється на:

3) системне програмне забезпечення (СПО);

4) прикладне програмне забезпечення (ППО);

5) інструментарій технології програмування.

Системна програма – програмний продукт, що надає користувачеві допоміжні послуги з взаємодії з файловою системою та апаратним забезпеченням.

Прикладна програма – програмний продукт, що полегшує користувачеві вирішення завдань у певній сфері діяльності.

Інструментальна програма – програмний продукт, що дозволяє створювати нові програми за допомогою будь-яких мов програмування.

3.2. Поняття «Операційна система»

Робота на ЕОМ користувачами різних рівнів забезпечується операційною системою (ОС). Вона є сполучною ланкою між ЕОМ як набором електронних плат і користувачем.

Усі машинні операції, які забезпечують виконання роботи з файлом, перебирає операційна система. Таким чином, можна сказати, що ОС бере на себе практично всі «низькорівневі» проблеми: обробка переривань, керування таймерами й оперативною пам’яттю та інші. Завдяки їй користувач постає не

перед реальною апаратурою з усіма наслідками, а перед її абстракцією (спрощеним відображенням).

Крім інтерфейсної функції, ОС виконує функцію розподілу ресурсів. Відповідно до даного підходу функцією ОС є розподіл процесорів, пам'яті, пристроїв та даних між процесами, які претендують на ці ресурси. ОС повинна керувати всіма ресурсами обчислювальної машини таким чином, щоб забезпечити максимальну ефективність її функціонування за заданими критеріями: пропускну здатність, реактивність системи тощо. Управління ресурсами включає рішення двох загальних завдань, що не залежать від типу ресурсу [3]:

1) планування ресурсу – тобто визначення, кому, коли, а для ділених ресурсів й у якій кількості, необхідно виділити даний ресурс;

2) відстеження стану ресурсу – тобто підтримка оперативної інформації про те, зайнятий чи не зайнятий ресурс, а для ділених ресурсів – яка кількість ресурсу вже розподілена, а яка вільно.

Використовуючи вищесказане, можна дати таке визначення операційної системи.

Операційною системою називають комплекс програм, що забезпечують автоматизацію управління апаратними ресурсами ЕОМ та виконання команд користувача [4].

Для вирішення цих загальних завдань управління ресурсами різні ОС використовують різні алгоритми, які визначають вигляд системи, включаючи характеристики продуктивності, сферу застосування і навіть інтерфейс користувача. Наприклад, алгоритм

управління процесором значною мірою визначає, чи є ОС системою поділу часу, системою пакетної обробки чи системою реального часу.

3.3. Класифікація операційних систем

Операційні системи можна класифікувати за багатьма ознаками: способи реалізації внутрішніх алгоритмів управління ресурсами, методи проектування, типи апаратних платформ тощо. На рис. 3 представлено схему класифікації ОС.

За алгоритмами управління ресурсами ОС можна розподілити на такі підкласи:

- з підтримкою багатозадачності;
- за типами багатозадачності;
- по підтримці багатокористувацького режиму;
- з підтримки багатонитковості;
- за алгоритмами багатопроцесорної обробки.

За підтримкою багатозадачності ОС є однозадачними та багатозадачними.

- *Однозадачні* ОС забезпечують зручну взаємодію користувача з ЕОМ, керування периферійними пристроями. До її складу входить файлова система та командний інтерпретатор.

- *Багатозадачні* ОС, крім перерахованих вище функцій для однозадачних ОС, керують поділом ресурсів, що спільно використовуються, наприклад, процесором, оперативною пам’яттю, дисковим накопичувачем тощо.

Рис. 3. Класифікація операційних систем

За типами багатозадачності ОС діляться на системи з невитісняючою багатозадачністю та витісняючою багатозадачністю.

Види багатозадачності визначаються способом розподілу процесорного часу. Коли йдеться про невитісняючу багатозадачність, то механізм планування процесів зосереджено в операційній системі. У цьому випадку активний процес виконується доти, доки він не передає керування операційною системою. При витісняючій багатозадачності механізм планування розподілений між системою та прикладними програмами. У цьому випадку операційна система приймає рішення про переключення процесора з одного процесу на інший, а не самі процеси, включаючи активний.

За підтримкою багатокористувацького режиму ОС підрозділяються на однокористувацькі та розраховані на багато

користувачів.

Основна відмінність розрахованих на багато користувачів систем від однокористувальних – наявність засобів захисту інформації кожного користувача від несанкціонованого доступу інших користувачів.

За підтримкою багатонитковості ОС розрізняють із розпаралелюванням обчислень у рамках одного завдання і без нього. Операційні системи з розпаралелюванням обчислень поділяють процесорний час на окремі гілки (нитки) процесу.

За алгоритмами багатопроцесорної обробки системи характеризуються наявністю чи відсутністю засобів мультипроцесорної обробки – мультипроцесування. Системи з наявністю засобів мультипроцесування поділяються на асиметричні та симетричні.

Асиметричні ОС виконуються лише на одному процесорі. Ресурси інших процесорів розподіляються за прикладними програмами.

Симетричні ОС повністю децентралізовані та використовують ресурси всіх процесорів для виконання системних завдань та прикладних програм.

По апаратним платформам розрізняють операційні системи для персональних комп’ютерів, мінікомп’ютерів, мейнфреймів, кластерів та мереж ЕОМ. Тут специфіка апаратних засобів (конфігурації) визначає типи операційних систем.

Існують спеціальні операційні системи, які можуть переноситися між ЕОМ різного класу. Такі системи називають мобільними. В них

апаратно-залежні блоки чітко локалізовані. Тому при перенесенні системи переписуються лише вони.

По областях використання системи класифікують на такі:

- системи пакетної обробки;
- системи розподілу часу;
- системи реального часу.

Критерієм ефективності для *систем пакетної обробки* є пропускна можливість – обсяг розв'язуваних завдань за одиницю часу.

Такі системи працюють наступним чином. На початку відведеного періоду часу (наприклад, робочого дня) у системі формується пакет завдань. Для кожного завдання відомі вимоги до системних ресурсів. З пакета формується набір процесів, що одночасно виконуються. Як правило, ОС відбирає завдання таким чином, щоб збалансувати роботу всіх пристроїв ЕОМ. Наприклад, ефективним вважається наявність задач з більшим об'ємом обчислень та задач з інтенсивним введенням/виведенням даних. Таким чином, система з невиконаних завдань обирає ту, яка найбільш «вигідна» в ситуації, що склалася. Отже, користувач не може інтерактивно взаємодіяти з машиною, на якій встановлена система пакетної обробки. Процес роботи зводиться до того, що на початку періоду користувач ставить машині задачу і отримує результат в кінці періоду.

У *системах поділу часу* кожному користувачеві надається термінал та деяка частина процесорного ресурсу (квант). При цьому пропускна здатність машини знижується, але користувач у цей час

не ізольований від виконання його завдань. Критеріями ефективності систем поділу часу є зручність та ефективність роботи користувача.

Критерієм ефективності *систем реального часу* є реактивність, іншими словами, здатність у заданий інтервал часу відреагувати на певну подію. Приклад такої системи – система керування експериментальною установкою. У певному інтервалі часу установка передає дані, які система попередньо обробляє та записує на диск. Якщо якийсь фрагмент даних не буде оброблений у заданий проміжок часу, а нові дані почнуть надходити до ЕОМ, то виникне аварійна ситуація: втрата даних, контроль над установкою тощо.

Для цих систем мультипрограмна суміш являє собою фіксований набір заздалегідь розроблених програм. Вибір програми виконання здійснюється виходячи з поточного стану керованого об’єкта чи відповідно до розкладу планових робіт.

Існують операційні системи, які здатні поєднувати у собі системи різних типів. Наприклад, система може виконувати пакетну обробку та здійснювати діалог з користувачем. У цьому випадку пакетна обробка здійснюється у так званому *фоновому* режимі.

За методами проектування ОС класифікуються за наступними базовими концепціями:

- способи побудови ядра системи;
- об’єктно-орієнтований підхід побудови ОС;
- множинність прикладних середовищ;

- розподілена організація операційної системи.

За способами побудови ядра операційні системи поділяються на системи з *монолітним ядром* та системи з *мікроядром*.

Монолітне ядро є єдиною програмою, яка працює в привілейованому режимі. В даному випадку не потрібно перемикання з привілейованого режиму на користь користувацького і навпаки при переході з однієї процедури на іншу.

Мікроядро також працює у привілейованому режимі, але при цьому виконує мінімум функцій з управління апаратурою. Функції ОС вищого рівня виконують спеціалізовані компоненти системи – служби (сервіси), що працюють у режимі користувача. ОС з мікроядром повільніші порівняно з ОС, побудованими на базі мікроядра, за рахунок того, що в них здійснюється велика кількість переходів з привілейованого режиму в користувацький і навпаки. Але такі системи гнучкіші – їх можна нарощувати, модифікувати, тобто адаптувати для вирішення завдань різного класу та рівня складності.

Побудова ОС на базі *об'єктно-орієнтованого підходу* (ООП) дає можливість використовувати всі його переваги всередині операційної системи: накопичення працюючих блоків у вигляді стандартних об'єктів, можливість створення нових об'єктів на базі наявних за допомогою механізму наслідування. Інкапсуляція забезпечує гарний захист даних. ООП дозволяє структурувати системи, що складаються з набору добре визначених об'єктів.

Наявність *множинності прикладних середовищ* у межах однієї ОС дозволяє виконувати програми, розроблені для кількох ОС.

Багато сучасних операційних систем підтримують одночасно прикладні середовища MS-DOS, Windows, UNIX (POSIX), OS/2 або хоча б деякої підмножини з цього популярного набору. Такий підхід забезпечений на базі мікроядра та відповідних сервісів (серверів), що реалізують прикладне середовище для тієї чи іншої операційної системи.

Розподілена організація операційної системи дозволяє спростити роботу користувачів та програмістів у мережевих середовищах. У розподіленій ОС реалізовані механізми, які дозволяють користувачеві представляти та сприймати мережу як традиційний однопроцесорний комп'ютер. Характерні ознаки розподіленої організації ОС: наявність єдиної довідкової служби ресурсів, що розділяються, єдиної служби часу. Для ефективного розподілу програмних процедур по машинах у таких ОС реалізовано механізм виклику віддалених процедур (RPC – Remote Procedure Call). Багатониткова обробка дозволяє розпаралелювати обчислення в рамках одного завдання і виконувати це завдання відразу на кількох комп'ютерах мережі. Існують інші розподілені служби.

4. УПРАВЛІННЯ ЛОКАЛЬНИМИ РЕСУРСАМИ

У рамках цього розділу ми познайомимося з принципами та алгоритмами управління локальними ресурсами ЕОМ, що беруть на себе операційні системи. До основних завдань, що вирішуються операційними системами, входять:

- управління процесами (розподіл процесорного ресурсу);
- керування пам’яттю;
- управління введенням/виведенням;
- керування перериваннями тощо.

Розглянемо перші три завдання.

4.1. Управління процесами

Підсистема управління процесами безпосередньо впливає на функціонування операційної системи. Тут процесом називають абстракцію, що визначає і описує програму, що виконується. У рамках ОС процес є одиницею роботи та заявкою на використання системних ресурсів, наприклад, процесорного часу. Підсистема управління процесом здійснює планування виконання процесів, наприклад, створення чи знищення процесів.

Поняття та визначення у плануванні процесів. Щоб говорити про управління процесами, необхідно ввести поняття станів у багатозадачній ОС:

1) *виконання* – активний стан процесу, під час якого процес має всі необхідні ресурси і безпосередньо виконується процесором;

2) *очікування* – пасивний стан процесу, коли він не може

виконуватися за своїми внутрішніми причинами, він чекає на здійснення певної події, наприклад, завершення операції введення/виведення, отримання повідомлення від іншого процесу, звільнення якого-небудь необхідного йому ресурсу (процес заблокований);

3) *готовність* – пасивний стан процесу, коли він заблокований у зв'язку із зовнішніми відносно нього обставинами: процес має всі необхідні йому ресурси, він готовий виконуватися, проте процесор зайнятий виконанням іншого процесу.

Життєвий цикл процесу характеризується його перебуванням у одному зі станів і переходами між ними.

Тепер розглянемо ще два поняття, які характеризують стан операційного середовища та хід роботи процесу – контекст та дескриптор [5].

Контекст містить інформацію про стан регістрів та програмного лічильника, стан введення/ виведення, про помилки тощо.

Дескриптор містить ідентифікатор процесу, адресу розташування сегмента коду, ступінь привілейованості процесу тощо.

Створення та планування процесів. Щоб активувати процес, операційній системі необхідно насамперед знати його ідентифікатор, адресу сегмента, тобто прочитати дескриптор. Тільки потім ОС має відновити контекст процесу. Таким чином, дескриптор зберігає більш актуальну (оперативну) інформацію про процес щодо контексту.

Створення процесу відбувається у три етапи:

1) створити контекст та дескриптор;

2) включити дескриптор нового процесу у чергу готових процесів;

3) завантажити кодовий сегмент процесу на оперативну пам’ять.

Планування процесів ділиться на вирішення наступних задач [8]:

1) визначення моменту часу для зміни виконуваного процесу;

2) вибір процесу на виконання з черги готових процесів;

3) перемикання контекстів «старого» та «нового» процесів.

Алгоритми планування процесів поділяються на дві великі групи: алгоритми, засновані на *квантуванні*, та алгоритми, що ґрунтуються на *пріоритетах*.

В алгоритмах, заснованих на квантуванні процесорного часу, зміна активного процесу здійснюється у таких випадках:

- процес завершив свою роботу і вийшов із системи;
- під час виконання процесу сталася помилка;
- процес перейшов у стан «Очікування»;
- вичерпано квант процесорного часу, відведеного процесу.

На рис. 4 наведено хронологічну схему виконання процесу, який керується алгоритмом, заснованим на квантувані.

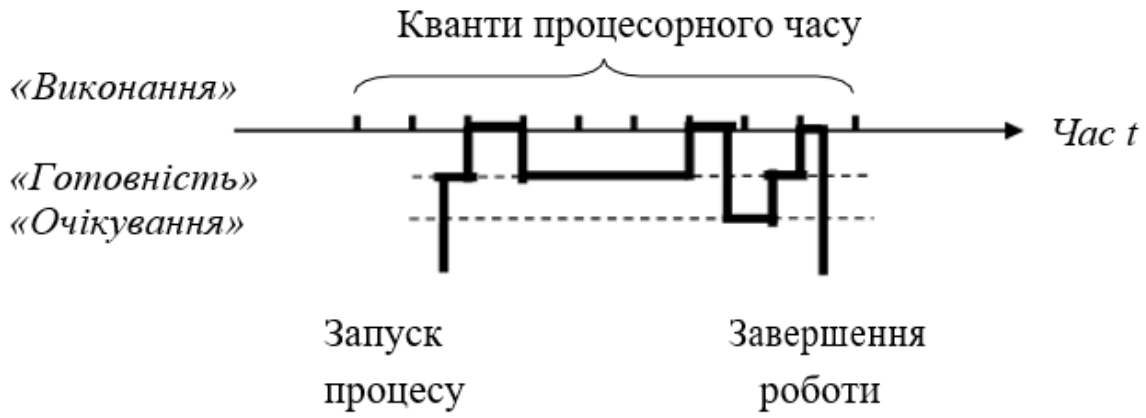


Рис. 4. Виконання процесу, керованого алгоритмом, що базується на квантуванні

Вичерпаний квант процес переходить у стан «Готовність» і чекає на новий квант. Ресурс ОС, що звільнився, надає іншому процесу, який за певними правилами вибирається з відповідної черги готових процесів. Таким чином, процес не займає процесор протягом тривалого часу. Звідси алгоритми квантування набули широкого поширення в ОС із розподілом часу.

Кванти процесорного часу, що виділяються для процесів, можуть бути як однаковими, так і різними в залежності від алгоритму розподілу процесорного часу. Понад те, протягом життєвого циклу процесу виділені йому кванти можуть різнитися за величиною.

По-різному може бути організована черга процесів у стані «Готовність»: за принципом «перший увійшов – останній вийшов» (принцип зливу), за принципом «перший увійшов – перший вийшов» (принцип черги).

Стан процесів, що плануються за допомогою алгоритмів, заснованих на квантуванні, можна подати у вигляді орієнтованого графа, зображеного на рис. 5.

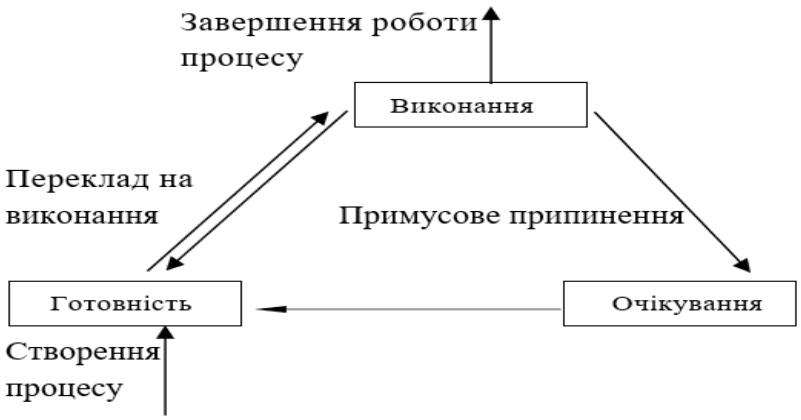


Рис. 5. Граф станів процесу, що планується за допомогою алгоритму, заснованого на квантуванні

Обидва різновиди ґрунтуються на пріоритетах і вибирають із черги готових процесів той, у якого найвищий пріоритет. Відмінності алгоритмів полягають у визначенні моменту зміни активного процесу. В алгоритмі з відносними пріоритетами активний процес виконується доти, доки він сам не покине процесор (перейде в стан «Очікування», завершить роботу або помилка). В алгоритмі з абсолютними пріоритетами виконання активного процесу перерветься також у тому випадку, якщо в черзі з’явиться процес із великим пріоритетом. У цьому випадку перерваний процес перейде у стан «Готовність». Граф станів процесу в ОС з алгоритмами планування, заснованих на абсолютному пріоритеті, точно повторює граф, зображений на рис. 5. Граф станів процесу в ОС з алгоритмами планування, що ґрунтуються на відносному пріоритеті, відрізняється одним ребром,

що переходить зі стану «Виконання» у стан «Готовність» (рис. 6).

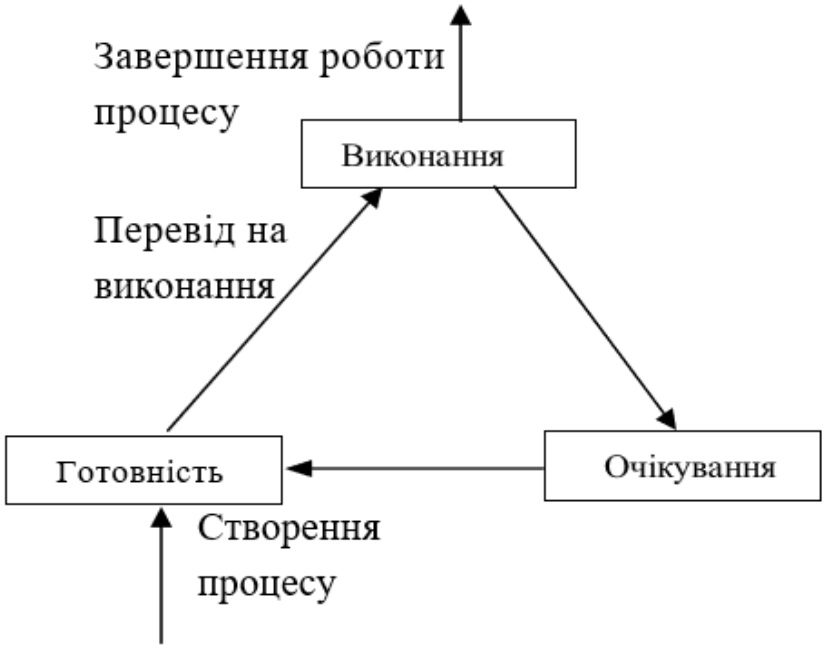


Рис. 6. Граф станів процесу, що планується
за допомогою алгоритму, заснованого на пріоритетах

Вищеописані групи алгоритмів також можна підрозділити на витісняючі та невитісняючі.

Non-preemptive multitasking – невитісняюча багатозадачність – це спосіб планування процесів, при якому активний процес виконується доти, доки він сам, з власної ініціативи, не віддасть управління планувальнику операційної системи для того, щоб той вибрав з черги інший, готовий до виконання процес.

Preemptive multitasking – витісняюча багатозадачність – це такий спосіб, при якому рішення про перемикання процесора з виконання одного процесу на виконання іншого процесу приймається планувальником операційної системи, а не найактивнішим завданням.

Засоби синхронізації та взаємодії процесів. Процесам часто

потрібно взаємодіяти один з одним, наприклад, один процес може передавати дані до іншого процесу, або кілька процесів можуть обробляти дані з загального файлу. У всіх цих випадках виникає проблема синхронізації процесів, яка може вирішуватися призупиненням та активізацією процесів, організацією черг, блокуванням та звільненням ресурсів.

Нехтування питаннями синхронізації процесів, що виконуються в режимі мультипрограмування, може призвести до їх неправильної роботи або навіть краху системи. Розглянемо, наприклад, програму друку файлів (принт-сервер; рис. 7). Ця програма друкує по черзі всі файли, імена яких послідовно в порядку надходження записують до спеціального загальнодоступного файлу «замовлень» інші програми. Особлива змінна NEXT, також доступна всім процесам-клієнтам, містить номер першої вільної для запису імені файлу позиції файлу «замовлень». Процеси-клієнти читають цю змінну, записують у відповідну позицію файлу «замовлень» ім’я свого файлу та нарощують значення NEXT на одиницю. Особлива змінна NEXT, також доступна всім процесам-клієнтам, містить номер першої вільної для запису імені файлу позиції файлу "замовлень". Процеси-клієнти читають цю змінну, записують у відповідну позицію файлу «замовлень» ім’я свого файлу та нарощують значення NEXT на одиницю.

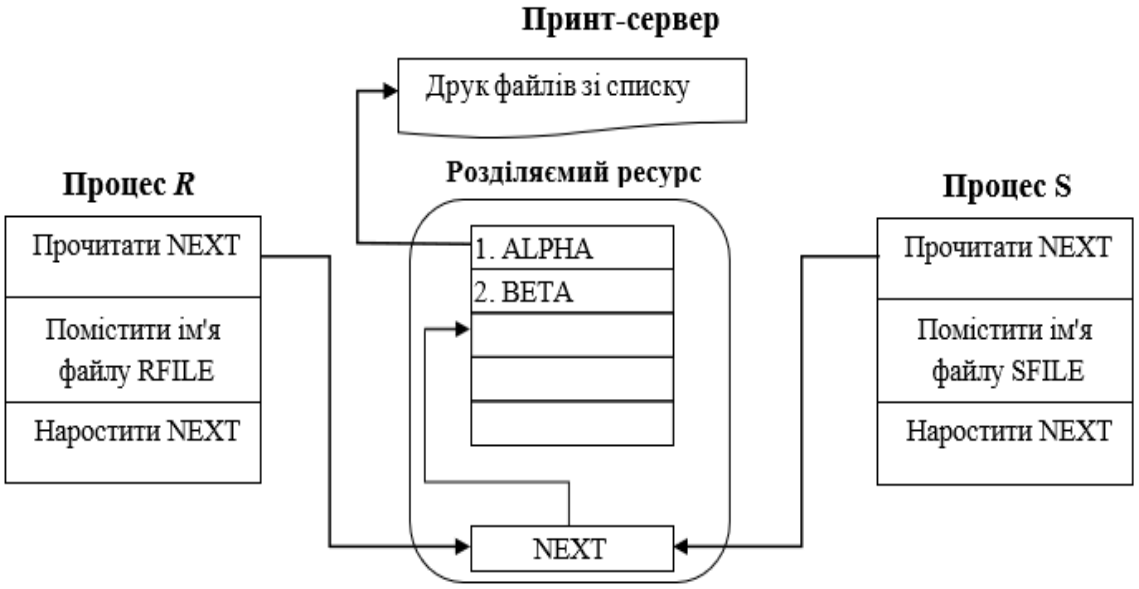


Рис. 7. Схема взаємодії процесів R і S з розділяємим ресурсом

Припустимо, що в певний момент процес R вирішив роздрукувати свій файл, для цього він прочитав значення змінної NEXT, значення якої для визначеності припустимо рівним 4. Процес запам’ятав це значення, але помістити ім’я файлу не встиг, оскільки його виконання було перервано (наприклад, внаслідок вичерпання кванта). Черговий процес S, який бажає роздрукувати файл, прочитав те саме значення змінної NEXT, помістив у четверту позицію ім’я свого файлу і наростив значення змінної на одиницю. Коли черговий раз управління буде передано процесу R, то він, продовжуючи своє виконання, у повній відповідності до значення поточної вільної позиції, отриманого під час попередньої ітерації, запише ім’я файлу також у позицію 4, поверх імені файлу процесу S.

На рис. 8 представлена діаграма перемикання між процесами.

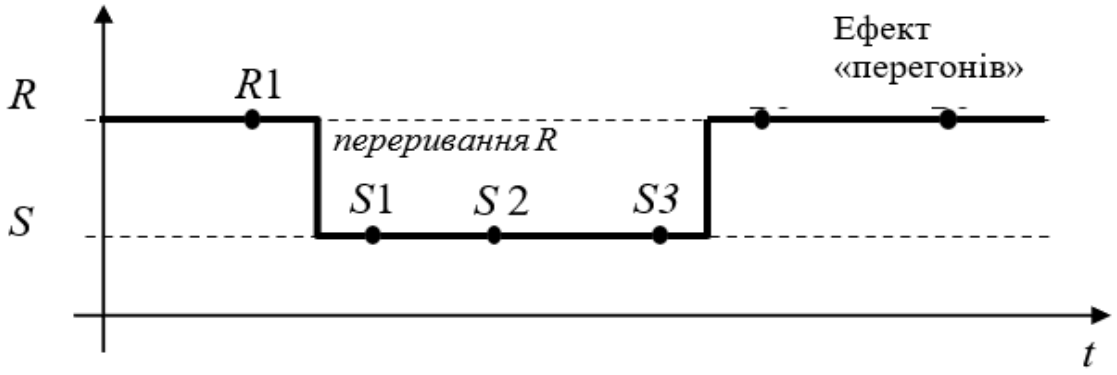


Рис. 8. Діаграма перемикання між процесами R та S

Таким чином, процес S ніколи не побачить свій файл роздрукованим. Складність проблеми синхронізації полягає в нерегулярності ситуацій, що виникають. У попередньому прикладі можна уявити й інший розвиток подій: було втрачено файли кількох процесів або, навпаки, не було втрачено жодного файлу. У разі все визначається взаємними швидкостями процесів і моментами їх переривання. Тому налагодження взаємодіючих процесів – складне завдання. Ситуації подібні до тієї, коли два або більше процесів обробляють дані і кінцевий результат залежить від співвідношення швидкостей процесів, називаються перегонами.

Важливим поняттям синхронізації процесів є поняття «*критична секція*» програми. Критична секція – це частина програми, в якій здійснюється доступ до даних, що розділяються. Для виключення ефекту перегонів по відношенню до деякого ресурсу необхідно забезпечити, щоб у кожний момент у критичній секції, пов’язаній із цим ресурсом, знаходився максимум один процес. Цей прийом називають взаємним винятком.

Взаємний виняток реалізується такими способами:

- заборона всіх переривань;
- метод блокуючих змінних;
- використання апарату подій;
- використання семафорів;
- застосування моніторів.

Глухі кути. Наведений на рис. 7 приклад допоможе нам проілюструвати ще одну проблему синхронізації – взаємні блокування, які також називають *діділоками* (deadlocks), *клінчами* (clinch) або *глухого кута*. Розглянемо приклад глухого кута. Нехай двом процесам, які виконуються в режимі мультипрограмування, для виконання їх роботи потрібно два ресурси, наприклад, принтер і диск. На рис. 9, а показані фрагменти відповідних програм. І нехай після того, як процес А зайняв принтер (встановив блокуючу змінну), він був перерваний [8]. Управління отримав процес В, який спочатку зайняв диск, але при виконанні наступної команди був заблокований, оскільки принтер виявився вже зайнятим процесом А. Управління знову отримав процес А, який у відповідності до своєї програми зробив спробу зайняти диск і був заблокований: диск вже розподілений процесу В. У такому становищі процеси А і В можуть перебувати як завгодно довго.

Залежно від співвідношення швидкостей процесів, вони можуть або взаємно блокувати один одного (9, *б*), або утворювати черги до ресурсів (9, *в*), що розділяються, або незалежно використовувати ресурси, що розділяються (9, *г*).

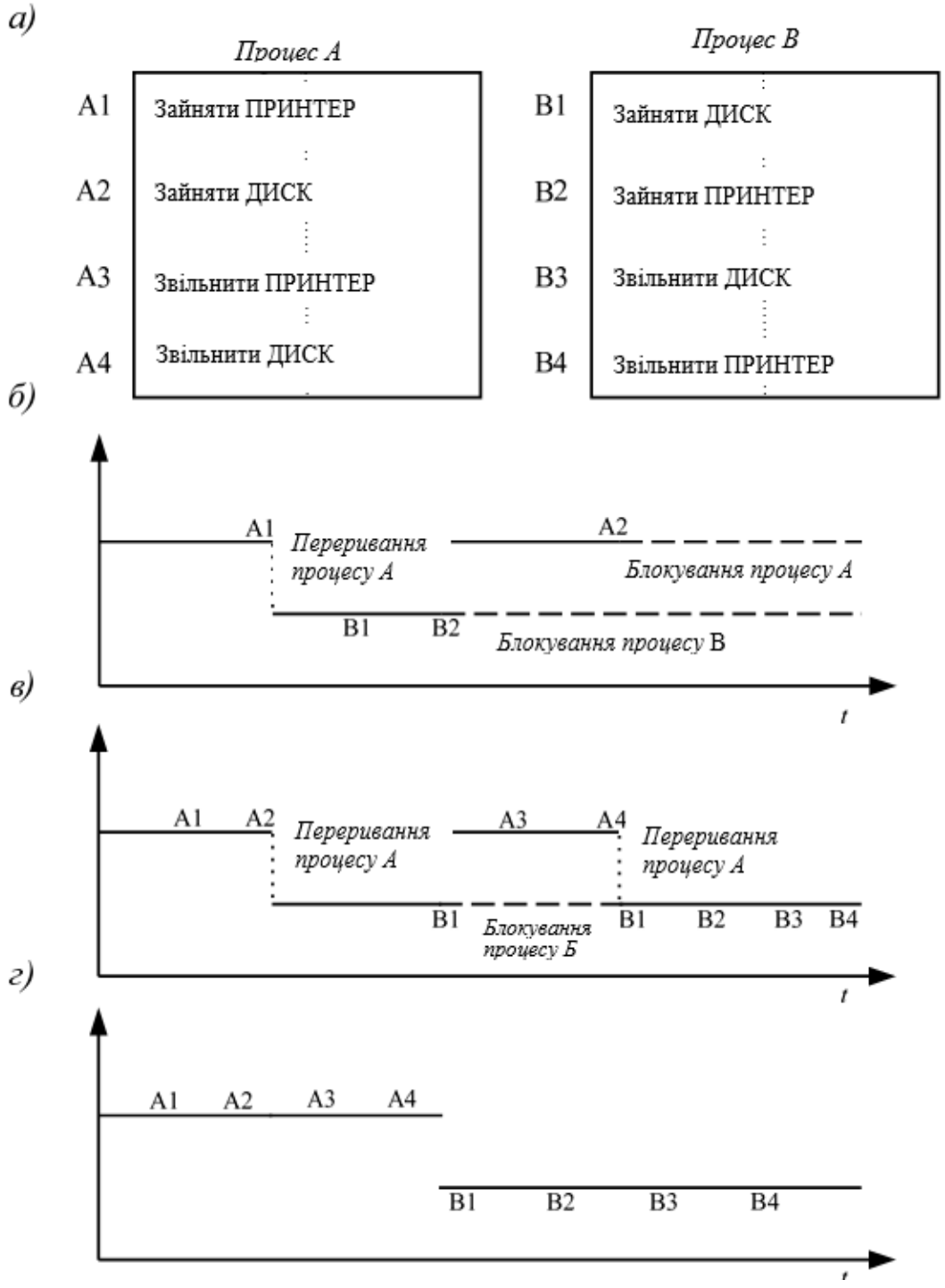


Рис. 9. Приклад глухого кута в роботі двох процесів

Ситуації глухих кутів слід відрізняти від простих черг, хоча й ті й інші виникають при спільному використанні ресурсів і зовні виглядають схоже: процес припиняється і чекає на звільнення ресурсу. Однак черга – це нормальне явище, невід'ємна ознака високого коефіцієнта використання ресурсів при випадковому

надходженні запитів. Вона виникає тоді, коли ресурс недоступний на даний момент, але через деякий час він звільняється, і процес продовжує виконання. Тупик же, що видно з його назви, є певною мірою нерозв'язною ситуацією.

У розглянутих прикладах глухий кут був утворений двома процесами, але взаємно блокувати один одного можуть і більше процесів.

Проблема глухих кутів включає наступні завдання:

- запобігання глухих кутів,
- розпізнавання глухих кутів,
- відновлення системи після глухих кутів.

Запобігання глухим кутам можливе на стадії написання програм, тобто програми повинні бути написані таким чином, щоб глухий кут не міг виникнути ні при якому співвідношенні взаємних швидкостей процесів. Так, якби в попередньому прикладі процес А та процес В запитували ресурси в однаковій послідовності, то глухий кут був би в принципі неможливий. Інший підхід до запобігання глухим кутом називається динамічним і полягає у використанні певних правил при призначенні ресурсів процесам, наприклад, ресурси можуть виділятися в певній послідовності, загальної для всіх процесів.

У деяких випадках, коли тупикова ситуація утворена багатьма процесами, які використовують багато ресурсів, розпізнавання глухого кута – нетривіальне завдання. Існують формальні, програмно-реалізовані методи розпізнавання глухих кутів, засновані на веденні таблиць розподілу ресурсів та таблиць запитів

до зайнятих ресурсів. Аналіз цих таблиць дозволяє виявити взаємні блокування.

Якщо ж ситуація глухого кута все-таки виникла, то не обов'язково знімати з виконання всі заблоковані процеси. Можна зняти тільки частину з них, при цьому звільняються ресурси, очікувані іншими процесами, можна повернути деякі процеси в область свопінгу, можна здійснити «*відкат*» деяких процесів до так званої контрольної точки, в якій запам'ятовується вся інформація, необхідна для відновлення виконання програми з цього місця. Контрольні точки розставляються у програмі в місцях, після яких можливе виникнення глухого кута.

4.2. Управління пам'яттю

Пам'ять – найважливіший ресурс, який потребує ретельного управління з боку мультипрограмної операційної системи. Розподілу підлягає вся оперативна пам'ять, не зайнята операційною системою. Зазвичай ОС розташовується в наймолодших адресах, проте може займати і найстарші адреси. Функції ОС з управління пам'яттю [8]:

- відстеження вільної та зайнятої пам'яті;
- виділення пам'яті процесам та звільнення пам'яті при завершенні процесів;
- витіснення процесів з оперативної пам'яті на диск;
- розподіл даних процесів між оперативною та зовнішньою пам'яттю у разі недостатності оперативної пам'яті для всіх процесів;

- налаштування адрес програми на конкретну область фізичної пам’яті.

Типи адрес. Для ідентифікації змінних та команд використовуються символічні імена (мітки), віртуальні адреси та фізичні адреси (рис. 10).

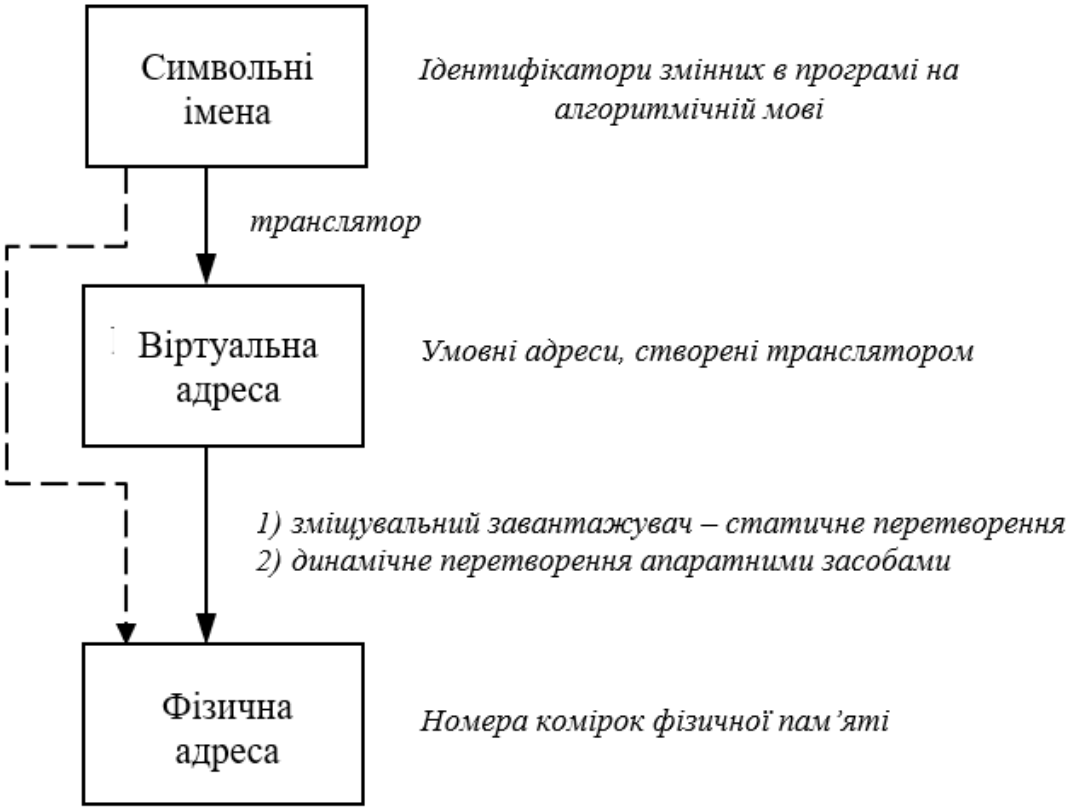


Рис. 10. Типи адрес

Символьні імена присвоює користувач при написанні програми алгоритмічною мовою або асемблером.

Віртуальні адреси створює транслятор, який перекладає програму машинною мовою. Так як під час трансляції в загальному випадку не відомо, в яке місце оперативної пам’яті буде завантажена програма, то транслятор надає змінним і командам віртуальні (умовні) адреси, зазвичай вважаючи за замовчуванням,

що програма буде розміщена, починаючи з нульової адреси. Сукупність віртуальних адрес процесу називається *віртуальним адресним простором*. Кожен процес має власний віртуальний адресний простір.

Фізичні адреси відповідають номерам осередків оперативної пам’яті, де насправді розташовані або будуть розташовані змінні та команди. Перехід від віртуальних адрес до фізичних може здійснюватися двома способами. У першому випадку заміну віртуальних адрес на фізичні робить спеціальна системна програма – зміщувальний завантажувач. Зміщувальний завантажувач на підставі наявних у нього вихідних даних про початкову адресу фізичної пам’яті, в яку слід завантажувати програму, та інформацію, надану транслятором про адресно-залежні константи програми, виконує завантаження програми, поєднуючи її із заміною віртуальних адрес фізичними.

Другий спосіб полягає в тому, що програма завантажується в пам’ять у незміненому вигляді у віртуальних адресах, при цьому операційна система фіксує зміщення дійсного розташування програмного коду щодо віртуального адресного простору. Під час виконання програми при кожному зверненні до оперативної пам’яті виконується перетворення віртуальної адреси на фізичну. Другий спосіб більш гнучкий, він допускає переміщення програми під час її виконання, у той час як зміщувальний завантажувач, жорстко прив’язує програму до спочатку виділеної їй ділянці пам’яті. Водночас використання зміщувального завантажувача зменшує накладні витрати, оскільки перетворення кожної віртуальної адреси

відбувається лише один раз під час завантаження, а у другому випадку – щоразу при зверненні на цю адресу.

У деяких випадках (зазвичай у спеціалізованих системах), коли заздалегідь точно відомо, в якій галузі оперативної пам’яті виконуватиметься програма, транслятор видає код, що виконується, відразу у фізичних адресах.

Методи розподілу пам’яті без використання дискового простору. Дана група методів характеризується тим, що при розподілі пам’яті дисковий простір (зазвичай, ресурс жорсткого диска) не залучається.

Розрізняють такі методи розподілу пам’яті без використання дискового простору [8]:

- *розподіл пам’яті фіксованими розділами* – пам’ять заздалегідь розбивається на стаціонарні розділи фіксованої довжини, для програми, що запускається, відводиться відповідний розділ;
- *розподіл пам’яті розділами змінної довжини* – під кожну програму виділяється стаціонарний розділ необхідного розміру;
- *розподіл пам’яті переміщуваними розділами* – під кожну програму виділяється розділ необхідного розміру.

Методи розподілу пам’яті з використанням дискового простору. Будь-який додаток на етапі виконання вимагає ресурсів оперативної пам’яті. Часто потреби додатків перевищують максимально доступний обсяг вільної фізичної пам’яті. Спочатку дану проблему «обходили» шляхом залучення оверлеїв, які дозволяли частину коду програми та її даних зберігати на диску, а

не в ОЗП. Істотним недоліком даного підходу є те, що вся відповідальність за розподіл пам'яті лежить на розробнику (програмісті) з усіма наслідками, що випливають.

Розвиток методів організації обчислювального процесу у цьому напрямі призвело до появи методу, відомого під назвою *віртуальна пам'ять*. Віртуальним називається ресурс, який користувачеві або програмі користувача має характерні властивості, якими він насправді не володіє. Так, наприклад, користувачеві може бути надана віртуальна оперативна пам'ять, розмір якої перевищує всю реальну оперативну пам'ять, що є в системі. Користувач пише програми так, ніби в його розпорядженні є однорідна оперативна пам'ять великого об'єму, але насправді всі дані, що використовуються програмою, зберігаються на одному або кількох різнорідних запам'ятовуючих пристроях, зазвичай на дисках, і при необхідності частинами відображаються у реальну пам'ять.

Таким чином, віртуальна пам'ять – це сукупність програмно-апаратних засобів, що дозволяють користувачам писати програми, розмір яких перевищує наявну оперативну пам'ять; для цього віртуальна пам'ять вирішує такі завдання:

- розміщує дані в пристроях різного типу, наприклад, частина програми в оперативній пам'яті, а частина на диску;
- переміщує при необхідності дані між запам'ятовуючими пристроями різного типу, наприклад, підвантажує потрібну частину програми з диска в оперативну пам'ять;
- перетворює віртуальні адреси на фізичні.

Всі ці дії виконуються автоматично, без участі програміста, тобто механізм віртуальної пам’яті прозорий стосовно користувача.

Найбільш поширеними реалізаціями віртуальної пам’яті є сторінковий, сегментний та сегментно-сторінковий розподіл пам’яті, а також свопінг.

На рис. 11 показано схему *сторінкового розподілу* пам’яті. Віртуальний адресний простір кожного процесу ділиться на частини однакового, фіксованого для даної системи розміру, що називаються віртуальними сторінками. Загалом розмір віртуального адресного простору не є кратним розміру сторінки, тому остання сторінка кожного процесу доповнюється фіктивною областю.

Вся оперативна пам’ять машини також ділиться на частини такого самого розміру, які називаються фізичними сторінками (або блоками). Розмір сторінки зазвичай вибирається, рівним степеню двійки: 512, 1024 і так далі, це дозволяє спростити механізм перетворення адрес.

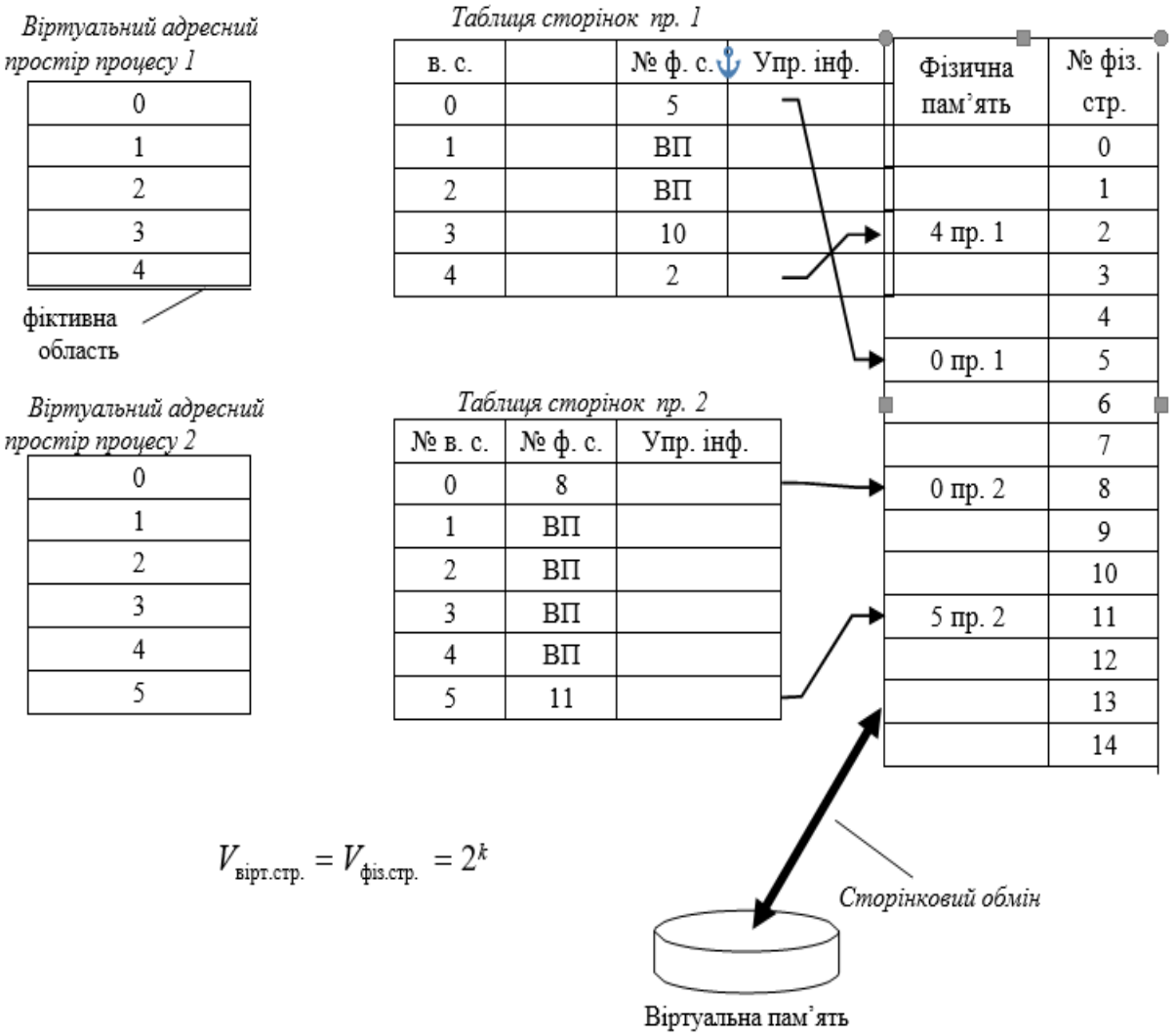


Рис. 11. Сторінковий розподіл пам'яті

При завантаженні процесу частина його віртуальних сторінок поміщається в оперативну пам'ять, а решта – на диск. Суміжні віртуальні сторінки не обов'язково розміщуються у суміжних фізичних сторінках. Під час завантаження операційна система створює для кожного процесу інформаційну структуру – таблицю сторінок. У ній встановлюється відповідність між номерами віртуальних та фізичних сторінок для сторінок, завантажених у оперативну пам'ять, або робиться відмітка про те, що віртуальна сторінка вивантажена на диск. Крім того, у таблиці сторінок

міститься керуюча інформація, така як ознака модифікації сторінки, ознака невивантажуваності (вивантаження деяких сторінок може бути заборонене), ознака звернення до сторінки (використовується для підрахунку числа звернень за певний період часу) та інші дані, що формуються та використовуються механізмом віртуальної пам’яті.

При активізації чергового процесу спеціальний регістр процесора завантажує адресу таблиці сторінок даного процесу. При кожному зверненні до пам’яті відбувається зчитування з таблиці сторінок інформації про віртуальну сторінку, до якої відбулося звернення. Якщо ця віртуальна сторінка знаходиться в оперативній пам’яті, то перетворюється віртуальна адреса на фізичну. Якщо потрібна віртуальна сторінка в даний момент вивантажена на диск, то відбувається так зване сторінкове переривання. Процес, що виконується, переводиться в стан очікування, і активізується інший процес з черги готових. Паралельно програма обробки сторінкового переривання знаходить на диску потрібну віртуальну сторінку і намагається завантажити її в оперативну пам’ять. Якщо у пам’яті є вільна фізична сторінка, завантаження виконується негайно, якщо ж вільних сторінок немає, то вирішується питання, яку сторінку слід вивантажити з оперативної пам’яті.

Після того, як вибрано сторінку, яка має залишити оперативну пам’ять, аналізується її ознака модифікації (з таблиці сторінок). Якщо виштовхувана сторінка з моменту завантаження була модифікована, то її нова версія має бути переписана на диск. Якщо ні, вона може бути просто знищена, тобто відповідна фізична

сторінка оголошується вільної. При сегментному розподілі віртуальний простір програми умовно розбивається на сегменти, у тому числі на сегменти даних та коду. На відміну від сторінок, сегменти можуть бути різної довжини в залежності від особливостей і принципів побудови програми. Як правило, кожен сегмент відповідає логічній структурі даних або коду, що формується розробником додатка. Ще одна корисна властивість сегментного розподілу - можливість створювати загальні сегменти для двох і більше програм, що виконуються одночасно.

При запуску програми операційна система формує таблиці сегментів, аналогічні таблицям сторінок. Фізична пам’ять також розбивається на сегменти. Вони можуть вивантажуватись у віртуальну пам’ять або знову завантажуватися з неї в оперативну пам’ять. Доступ до осередків оперативної пам’яті виконується за допомогою пари адрес: адреси початку сегмента та адреси зміщення всередині нього. При 16-розрядній адресації розмір сегмента не перевищував 64 кілобайти. При 32- та 64-розрядній адресації максимальний розмір сегмента міг бути значно вищим.

Сегментно-сторінковий розподіл поєднує властивості та підходи двох попередніх методів. Віртуальний адресний простір розбивається на сегменти, а самі сегменти на сторінки фіксованої довжини. Фізична пам’ять поділяється лише на сторінки тієї ж довжини, що й у віртуальному адресному просторі програми. Для доступу до необхідних даних використовуються три числа: номер сегмента, номер сторінки, зміщення всередині сторінки.

4.3. Керування введенням/виведенням

Одна з головних функцій ОС – керування всіма пристроями введення/виведення комп'ютера. ОС повинна передавати пристрої команди, перехоплювати переривання і обробляти помилки; вона також повинна забезпечувати інтерфейс між пристроями та іншою частиною системи. З метою уніфікації інтерфейс має бути однаковим для всіх типів пристроїв (незалежність від пристроїв).

Фізична організація пристроїв введення/виведення. Пристрої введення/виведення поділяються на два типи: блок-орієнтовані пристрої та байт-орієнтовані пристрої. Блок-орієнтовані пристрої зберігають інформацію в блоках фіксованого розміру, кожен з яких має свою власну адресу. Найпоширеніший блок-орієнтований пристрій – диск. Байт-орієнтовані пристрої не адресовані і не дозволяють робити операцію пошуку, вони генерують або споживають послідовність байтів. Приклад: термінали, сканери, адаптери мережі. Однак деякі зовнішні пристрої не належать до жодного класу, наприклад, годинник, який, з одного боку, не адресований, а з іншого боку, не породжує потоку байтів. Цей пристрій лише видає сигнал переривання у певні моменти часу.

Зовнішній пристрій зазвичай складається з механічного та електронного компонента. Електронний компонент називається контролером пристрою або адаптером. Механічний компонент представляє власне пристрій. Деякі контролери можуть керувати кількома пристроями. Якщо інтерфейс між контролером та пристроєм стандартизований, незалежні виробники можуть випускати сумісні як контролери, так і пристрої.

Операційна система зазвичай має справу не з пристроєм, а з відповідним контролером. Контролер, як правило, виконує прості функції, наприклад, перетворює потік біт у блоки, що складаються з байт, та здійснює контроль та виправлення помилок. Кожен контролер має кілька регістрів, які використовуються для взаємодії із центральним процесором. У деяких комп'ютерах ці регістри є частиною фізичного адресного простору. У таких комп'ютерах немає спеціальних операцій введення/виведення. В інших комп'ютерах адреси регістрів введення/виведення, які часто називають портами, утворюють власний адресний простір за рахунок введення спеціальних операцій введення/виведення. ОС виконує введення/виведення, записуючи команди до реєстру контролера.

Організація програмного забезпечення введення/виводу.

Основна ідея організації програмного забезпечення введення/виведення полягає у розбитті його на кілька рівнів. Нижні рівні забезпечують екранування особливостей апаратури від верхніх, а ті, в свою чергу, забезпечують зручний інтерфейс для користувачів.

Ключовим принципом є незалежність пристроїв. Вигляд програми не повинен залежати від того, чи вона читає дані з гнучкого диска або з жорсткого диска. Дуже близька до ідеї незалежності від механізмів ідея одноманітного іменування. Іншими словами, для іменування пристроїв повинні бути прийняті єдині правила.

Інше важливе питання для програмного забезпечення

введення/виведення – обробка помилок. Взагалі, помилки слід обробляти якомога ближче до апаратури. Якщо контролер виявляє помилку читання, він повинен спробувати її скорегувати. Якщо це йому не вдається, то виправленням помилок повинен зайнятися драйвер пристрою. Багато помилок можуть зникати при повторних спробах виконання введення/виведення, наприклад, помилки, викликані наявністю порошинок на головках читання або на диску. І тільки якщо нижній рівень не може дати раду помилці, він повідомляє про помилку верхньому рівню.

Ще одне ключове питання – це використання блокуючих (синхронних) та неблокуючих (асинхронних) передач. Більшість операцій фізичного введення/виведення виконується асинхронно – процесор починає передачу і переходить на іншу роботу, доки не настає переривання. Користувацькі програми набагато легше писати, якщо операції введення/виведення блокуючі. Наприклад, після команди читання програма автоматично припиняється доти, доки дані не потраплять у буфер програми. ОС виконує операції введення/виведення асинхронно, але представляє їх для програм користувача в синхронній формі.

Остання проблема полягає в тому, що одні пристрої є роздільними, інші – виділеними. Диски – це пристрої, що розділяються, оскільки одночасний доступ кількох користувачів до диска не є проблемою. Принтери – це виділені пристрої, тому що не можна «змішувати» дані, які друкуються різними користувачами. Наявність виділених пристроїв створює для операційної системи деякі проблеми.

Для вирішення поставлених проблем доцільно розділити програмне забезпечення введення/виведення на чотири шари (рис. 12):

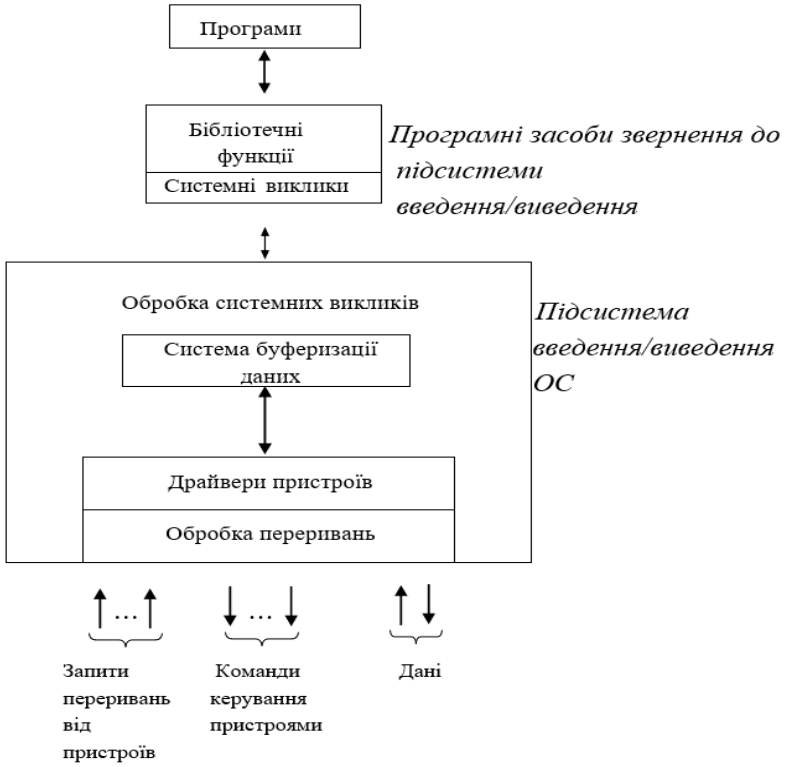


Рис. 12. Чотирирівнева організація введення/виведення

- обробка переривань;
- драйвери пристроїв;
- незалежний від пристроїв шар операційної системи;
- користувацький прошарок програмного забезпечення.

Обробка переривань ховається якнайглибше в надрах операційної системи. Тим самим деяке програмне забезпечення стикається з перериваннями. Весь залежний від пристрою код міститься в драйвері пристрою. Кожен драйвер керує пристроями одного типу або одного класу. Більшість програмного забезпечення введення/виведення є незалежним від пристроїв. Воно формує *незалежний від пристроїв шар операційної системи*. Точна межа

між драйверами та незалежними від пристроїв програмами визначається системою, оскільки деякі функції, які могли б бути реалізовані незалежним способом, насправді виконані у вигляді драйверів для підвищення ефективності або з інших причин. Користувацький шар програмного забезпечення є набором системних бібліотек, які використовуються при створенні програм для користувача.

5. ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

До прикладних програм належать:

1. Програми обробки тексту (текстові редактори та текстові процесори, видавничі системи, програми перекладу, програми розпізнавання, програми-коректори, програми-кодувальники, текстові утиліти).

2. Програми обробки таблиць і масивів даних (табличні процесори, системи управління базами даних, утиліти баз даних, програми-енциклопедії).

3. Програми обробки графічних зображень (графічні редактори, графічні утиліти, програми ділової графіки, програми презентаційної графіки, програми інженерної графіки, картографічні програми, програми анімації).

4. Програми обробки аудіо-відеосигналів (звукові програми, відеопрограми, системи мультимедіа, програми віртуальної реальності, комп'ютерні ігри).

5. Програми обробки чисел (програми-калькулятори, математичні системи, статистичні системи, системи моделювання).

6. Програми обробки знань (системи штучного інтелекту, експертні системи, системи управління базами знань, навчальні програми, програми тестування).

7. Комунікаційні програми (браузери, поштові програми, веб-редактори, клієнти мережевих служб).

8. Програми автоматизації робіт (інтегровані пакети, програми-органайзери, системи діловодства, бухгалтерські, економічні, юридичні, управлінські та ін).

До прикладного програмного забезпечення (ППЗ) належать програми, розроблені для користувачів або самими користувачами, для вирішення за допомогою ЕОМ конкретної задачі або групи завдань. Програми обробки замовлень, ведення бухгалтерського обліку, роботи з електронною поштою – лише мала частина великого переліку видів ППЗ.

Класифікувати прикладне програмне забезпечення можна наступним чином.

Корпоративні інформаційні системи [6]. Ці системи забезпечують частковий або повний інформаційний супровід діяльності підприємства або корпорації. До таких систем входять системи електронного документообігу та контролю за виконанням рішень, системи корпоративного бухгалтерського обліку, спеціалізовані корпоративні бази даних, інфраструктурні системи управління локальною мережею, надання віддаленого доступу та

електронної пошти.

Програмне забезпечення офісного призначення [7]. Такі програми (пакети програм) обслуговують потреби індивідуальних користувачів у створенні та управлінні інформацією. До найпоширеніших «офісних» програм входять: текстовий процесор (редактор), табличний процесор (електронна таблиця), засоби роботи з презентаціями, клієнти електронної пошти.

Програмне забезпечення для роботи з віддаленими ресурсами. Дані програми дозволяють отримати через мережу доступ до ресурсів різного виду: html-сторінки, віддалені сховища мультимедіа (наприклад фотоальбом), віддалені файлові сховища тощо. До таких програм належать інтернет-оглядачі, клієнти доступу до файлових сховищ (FTP-клієнт), спеціалізовані клієнти.

Програмні засоби передачі для віддаленого спілкування. Вони дозволяють здійснювати надсилання повідомлень між користувачами, розташованими на певній дистанції. До складу таких програмних засобів входять інтернет пейджери (протоколи ICQ, Google Talk, MailRu Agent і т.д.), системи інтерактивних текстових телеконференцій – «чати», засоби IP-телефонії (забезпечення голосового та відеоканалу за допомогою інтернет-протоколів) , засоби інтерактивних аудіо- та відеоконференцій.

Освітнє програмне забезпечення. Воно за змістом близьке до ПЗ для роботи з медіа та для розваг. Однак на відміну від нього має чіткі вимоги щодо тестування знань користувача та відстеження засвоєння того чи іншого матеріалу. Багато освітніх програм включають функції спільного віддаленого використання тих чи

інших ресурсів.

Імітаційне програмне забезпечення – використовується для симуляції фізичних чи абстрактних систем з метою наукових досліджень, навчання чи розваги.

Інструментальні програмні засоби для роботи з мультимедійним вмістом. Дані програми дозволяють створювати та змінювати мультимедіа ресурси різних типів: музичні ресурси, зображення (включаючи оброблені фото), дизайнерські рішення, відеоролики, комплексні мультимедіа ресурси (наприклад, складні флеш-ролики).

Прикладні програми для проектування та конструювання. Використовуються для розробки (моделювання або прототипування) апаратного та програмного забезпечення. Охоплюють автоматизований дизайн (computer aided design – CAD), автоматизоване проектування (computer aided engineering – CAE), редагування та компілювання мов програмування, програми інтегрованого середовища розробки (Integrated Development Environments -IDE), інтерфейси для прикладного програмування (Application Interfaces - API).

Пакети прикладних програм (ППП) також прийнято розділяти на такі класи:

1) ППП загального призначення;
2) офісні ППП;
3) проблемно-орієнтовані ППП;
4) методо-орієнтовані ППП;
5) ППП автоматизованого проектування;

6) програмні засоби мультимедіа;

7) настільні видавничі системи;

8) інтелектуальні системи.

Окремими видами прикладних програм є:

1. *Текстовий редактор* – програма для створення та редагування текстових документів.

2. *Табличний процесор* - Програма для створення та обробки електронних таблиць з даними.

3. *Система управління базами даних* (СУБД) – програмний засіб для створення та обробки баз даних.

4. *Графічний редактор* – програма для створення та обробки малюнків, картинок, графічних зображень.

5. *Програма презентаційної графіки* – програма для створення та обробки електронних презентацій зі слайдами.

6. *Програма інженерної графіки* (або система автоматизованого проектування) - програма для створення креслень, а також для проектування тривимірних деталей і споруд.

7. *Картографічна програма* (Геоінформаційна система) – програма для оцифрування знімків місцевості та для створення на їх основі географічних карт.

8. *Система мультимедіа* - Програма, що дозволяє обробляти текст, малюнки, відео, звук та ін. види інформації.

9. *Математична система* – програма для математики ческої обробки числових даних та для проведення аналітичних перетворень.

10. *Експертна система* – програма, що містить знання експертів у певній предметній галузі та видає поради щодо дій у конкретних ситуаціях.

11. *Браузер* – програма для перегляду Web-сайтів у мережі Інтернет та інших гіпертекстових документів.

12. *Бухгалтерська програма* – програма, що дозволяє автоматизувати ведення бухгалтерської документації.

Основні відомості про редактор CORELDRAW.

Одним із лідерів серед програм для роботи з комп'ютерною векторною графікою є редактор CorelDraw, оскільки він має зручний інтерфейс**,** містить значний набір засобів для створення та редагування графічних **об**'єктів, що мають високу якість зображення при друці. УCorelDraw за потреби можна також використовувати і растрові зображення, вставляючи їх **у** документ. Кожен такий малюнок можна окремо редагувати.

Для запуску програми потрібно виконати команду головного меню Пуск/Программы/CorelDraw Graphics Cuite12/CorelDraw 12.

Після відкриття програми в початковому діалоговому вікні буде запропоновано: створити новий документ за **п**оточним шаблоном або шаблоном користувача, відкрити останній робочий файл або інший**,** раніше створений, файл. Вибравши потрібний варіант, можна розпочати роботу.

Елементи робочого вікна програми

Структура робочого вікна редактора є типовою для додатків ОС Widows (має заголовок, кнопки управління вікном, панелі, рядок меню та рядок стану, робочу область, лінійку, смуги прокрутки). У

центрі вікна програми розташовано аркуш паперу, який є робочою областю. Можна створювати зображення як на аркуші, так і за його межами, проте на друк буде виведено лише те, що міститься всередині робочої області. Смуги прокрутки надають змогу переміщуватися по зображенню, а горизонтальна та вертикальна лінійки - точно позиціонувати елементи малюнка та вимірювати їх розміри. Для роботи з кольором у правій частині вікна розташована палітра кольорів.

Меню містить команди:

Файл – відкривання, закривання, збереження файлів[1] та їх імпорт **і** експорт;

Редагувати – робота з буфером обміну та вставленими об'єктами, режим пошуку об'єктів, відміна останньої дії тощо;

Показати (Вид) – різні режими перегляду документа, зазначення зображення допоміжних ліній та розмітки сторінки;

Розміщення – робота із сторінками документа;

Компоновути (упорядкувати) – керування взаємним розташуванням об'єктів;

Ефекти – використання додаткових ефектів (коригування, трансформація тощо);

Бітові (растрові) зображення – робота з імпортованими растровими зображеннями;

Текст – команди розміщення та форматування тексту;

Інструменти – налагодження інтерфейсу та режиму роботи CorelDraw;

Вікно – стандартні команди управління декількома відкритими вікнами;

Допомога – виклик довідкової служби.

Робота із сторінками. За замовчуванням параметри нового документа будуть стандартними: розмір сторінки А4, орієнтація портретна (у редакторі Word така орієнтація називається книжною). Для зміни цих параметрів потрібно виконати команду меню Размещение/Настройки страницы. Також пункт меню Размещение використовується для: додавання нової або видалення непотрібної сторінки; перейменування поточної сторінки; переходу між сторінками.

Панелі інструментів. Панель *Стандартная* майже повністю ідентична такій же панелі редактора Word (або електронних таблиць Excel). Основні засоби роботи в CorelDraw розташовані на панелі *Инструменты*.

Якщо кнопка інструмента містить зображення маленького чорного трикутничка, то це означає, що вона має допоміжну панель (призначення цієї кнопки можна змінити). Такими є переважна більшість кнопок (виключення становлять лише кнопки "робота з текстом", "швидке малювання" та "вказівник"). Для зміни призначення кнопки потрібно клацнути на трикутничку та вибрати потрібний інструмент. На мал. 7 також зображено піктограми деяких часто вживаних інструментів та їх призначення.

Панель *Свойства є* властивою саме редактору CorelDraw, її кнопки можуть бути активними або ні, залежно від дій користувача (активними є ті кнопки, які можна застосувати в поточній ситуації).

Крім того, кнопки цієї панелі можуть змінюватися залежно від інструмента, з яким працює користувач.

Створення простих об'єктів. Як уже зазначалося, основним поняттям векторної графіки с об'єкт: пряма**,** круг, крива, замкнена крива тощо. Такі об'єкти можна поєднувати в групу для подальшого редагування цієї групи як одного цілого. З метою зображення об'єктів використовують вже згадувану панель *Инструменты.*

Довільний об'єкт CorelDraw має низку загальних характеристик: деяку кількість точок - вузлів, що поєднані прямими або кривими лініями - сегментами. Координати вузлів і параметри сегментів визначають загальний вигляд об'єкта. Вузли бувають трьох типів: симетричні, гладкі та загострені. Незамкнені об'єкти мають лише контур. Замкнені об'єкти, окрім контуру, можуть мати ще й заливку.

Основи роботи з простими об'єктами. Перед тим, як виконати дію з об'єктом, його потрібно виділити. Виділений об'єкт містить вісім квадратних маркерів.

1. Для виділення вже існуючого об'єкта потрібно вибрати інструмент *Вказівник* та один раз клацнути лівою кнопкою миші по об'єкту.

2. Для повороту об’єкта потрібно вибрати інструмент *Вказівник* та двічі клацнути лівою кнопкою миші по об’єкту. Далі, утримуючи ліву кнопку миші на маркері повороту та переміщуючи мишу, здійснити повтор об’єкта.

3. Для масштабування об’єкта потрібно вибрати інструмент *Вказівник* та один раз клацнути лівою кнопкою миші по об’єкту.

Утримуючи ліву кнопку миші на маркері зміни розміру об’єкта та переміщуючи мишу, змінити розмір.

4. Область всередині виділеного об'єкта можна зафарбувати. Для цього вибирають інструмент *Заливка* та зазначають тип заливки (однорідну, градієнтну або візерунчасту). У запропонованому діалоговому вікні можна вибрати колір заливки та натиснути кнопку вікна <*Ок*>.

5. Контур об'єкта також можна змінювати: збільшувати або зменшувати ширину лінії, її вигляд (пунктир, подвійна тощо) та колір. Для цього використовують інструмент *Контур.*

6. CorelDraw має набір графічних примітивів: прямокутник, многокутник, еліпс. Для зображення будь-якого з цих об'єктів потрібно:

- обрати відповідний інструмент на панелі та перемістити курсор миші в робочу область;
- натиснути ліву кнопку миші та протягнути її, поки об'єкт не набуде потрібного розміру (для багатокутника попередньо на панелі *Свойства* зазначити кількість вершин).

7. Для того, щоб отримати квадрат, круг чи правильний многокутник, потрібно під час побудови відповідного примітиву утримувати клавішу <CTRL> на клавіатурі.

8. Для побудови графічного примітива з точно зазначеним центром потрібно встановити вказівник миші в місце розташування центру та побудувати| відповідну фігуру, утримуючи клавішу <*Shift*> на клавіатурі.

9. Модифікуючи графічний примітив еліпс, можна отримати сектори та дуги. Для цього потрібно скористатися панеллю *Свойства* та визначити на ній, залежно від потреби, початковий і кінцевий кути, еліпс, дугу або сектор.

10. Доречно використовувати контекстне меню для роботи з об'єктами, що значно полегшує роботу. Зокрема, вибравши в контекстному меню пункт *Свойства,* можна отримати окреме діалогове вікно *Свойства объекта,* у якому зазначити всі необхідні

параметри виділеного об'єкта (контур, заливка) та натиснути кнопку вікна *Применить.* Крім того, виділивши об'єкт та вибравши інструмент *Форма,* за допомогою контекстного меню можна збільшити або зменшити кількість вузлових точок об'єкта та модифікувати тип вузлів.

11. Для зміни форми графічних примітивів за вузловими точками, застосовуючи інструмент *Форма,* потрібно:

- виділити об'єкт;
- виконати команду меню Компоновать/Преобразовать в кривую;
- вибрати інструмент *Форма;*
- за допомогою контекстного меню змінити вузлові точки, а відтак і сам об'єкт.

12. Для виконання операцій копіювання, вставки, видалення об'єктів можна скористатися командами в пункті меню Редактировать (або комбінаціями клавіш, що зазначені для кожної **з** команд у цьому пункті).

13. Невдалий об'єкт одразу після створення (не виконуючи ніяких інших дій) можна видалити, натиснувши клавішу <*Delete*>. Для видалення довільного об'єкта його потрібно попередньо виділити.

14. Створені об'єкти можуть накладатися один на інший**.** Для зазнане порядку накладання виділеного об'єкта потрібно скористатися командою меню Компоновать/Порядок. У додатковому меню буде запропоновано варіанти розташування: над усіма іншими об'єктами, на нижньому рівні тощо, допомогою

команди меню Компоновать можна також включити декілька об'єктів в одну групу. Зазначені команди аналогічні командам**,** що містяться на панелі *Рисование* в текстовому редакторі Word.

15. Створене зображення можна в подальшому використовувати й у інших додатках. Попередньо таке зображення потрібно зберегти в певному форматі, використавши команду меню Файл/Экспорт... . У вікні *Экспорт* потрібно визначити місце та формат збереження файла та натиснути кнопку вікна *Экспорт>*.

Криві Без'є. Важливими об'єктами CorelDraw є криві Без'є - плавно вигнуті криві лінії, за допомогою яких можна побудувати довільний контур. Криві названі на честь математика П'єра Без'є, який у середині 70-х років минулого століття визначив, що довільну криву можна задати за допомогою двох векторів, які розташовано на початку та в кінці кривої. Саме це й лягло в основу описання кривих Без'є.

Крім початкової та кінцевої точок кривої (вузлів), її зовнішній вигляд визначається кривизною між цими вузлами. Графічно кривизна визначається за допомогою двох відрізків, що виходять з вузла і називаються маніпуляторами кривизни. Першим параметром кривизни є *нахил кривої* при її входженні у вузол. Цей нахил визначається нахилом маніпулятора кривизни. Другим параметром, що визначає кривизну, є *ступінь кривизни.* Цей параметр визначається довжиною маніпулятора. Отже, координати вузлів, нахил та довжина маніпуляторів кривизни повністю визначають вигляд кривої Без'є. Якщо довжини маніпуляторів

кривизни матимуть нульове значення, то сегмент між вузлами буде прямим.

Основні роботи з тестом. У графічному редакторі CorelDraw можна здійснювати також набір та форматування тексту. Для створення текстових об'єктів використовують інструмент *Текст* з панелі *Инструменты.* Кнопка на панелі має вигляд Ж.

Після вибору цього інструмента панель *Свойства* буде схожа на панель *Форматирование* текстового редактора Word: наявними будуть поля із списком стилів, параметрами форматування шрифтів та абзаців тощо. У редакторі розрізняють два різновиди тексту: фігурний та звичайний.

Фігурний текст. Після вибору інструмента *Текст* - вказівник миші набуде вигляду хрестика. Для *введення фігурного тексту* цей вказівник необхідно перемістити в робочу область, де розташовуватиметься текст, клацнути лівою кнопкою миші і ввести текст.

Фігурний різновид призначено для введення невеликого тексту. У подальшому з ним можна працювати як із звичайним графічним об'єктом. Зокрема, рядок фігурного тексту можна розташувати вздовж контура довільної форми (круга, еліпса тощо). Для цього потрібно виконати такі дії:

- створити об'єкт, навколо якого **у** подальшому розташовуватиметься текст;
- виконати введення фігурного тексту;
- виконати команду меню Текст/Подогнать Текст к пути, після чого на екрані з'явиться велика стрілка =>;

- підвести стрілку до об'єкта, вздовж якого потрібно розташувати текст та клацнути по ньому один раз лівою кнопкою миші.

Для редагування існуючого фігурного тексту потрібно виділити його та виконати команду меню Текст/Редактировать текст.... Для форматування існуючого тексту виконують команду Текст/Форматировать текст....

Звичайний текст. Звичайний текст призначено для роботи з великим обсягом інформації, яка вводиться в блоках. Всередині такого блоку текст розміщується як на сторінці текстового процесора: із розбивкою по рядках та абзацах і відповідним форматуванням.

Як уже зазначалося, після вибору інструмента *Текст* вказівник миші набуде вигляду хрестика. Для *введення звичайного тексту* спочатку потрібно виділити область, у якій буде набрано текст. Для цього вказівник миші потрібно перемісити у робочу область, клацнути лівою кнопкою миші і, не відпускаючи її, протягнути: на екрані з'явиться пунктирна рамка, яка й буде визначати область введення тексту. Редагування та форматування звичайного тексту використовують ті самі команди меню, що й для фігурного тексту. Проте, звичайний текст не можна розташувати вздовж кривої.

Робота з текстом та його імпортування. Зрозуміло, що окрім передачі звичайної інформації, шрифти мають ще одне призначення: різні типи шрифту можуть бути використані в якості елемента дизайну, і є складовими ілюстрації, логотипа тощо.

Кожен шрифт є окремим файлом, тому на різних комп'ютерах можуть бути встановлені різні шрифти. Якщо ілюстрація **з** використанням шрифтів була створена на одному комп'ютері, а відкрити її потрібно на іншому, то немає гарантії, що в наявності будуть усі потрібні шрифти. У таких випадках текст доцільно переводити в криві (попередньо виділивши його): команда меню Компоновать/Преобразовать в кривую. Переводити в криві можна як фігурний, так і звичайний текст. Проте так змінений текст у подальшому не можна буде редагувати.

Якщо великий обсяг тексту є набраним у іншому додатку ОС Windows (наприклад, у текстовому редакторі Word), то цей текст можна скопіювати та вставити у CorelDraw, використовуючи буфер обміну (імпортувати).

Текст, набраний за допомогою Microsoft Equation Editor (редактора формул), потрібно вставляти у CorelDraw за допомогою команди меню Редактировать/Специальная вставка. У вікні *Специальная вставка* потрібно вибрати пункт Microsoft Equation. У подальшому таку формулу можна переміщати, збільшувати або зменшувати та редагувати безпосередньо в CorelDraw. Для редагування потрібно двічі клацнути лівою кнопкою миші по формулі, що призведе до появи панелі *Формула* (ця панель матиме стандартний вигляд - як у текстовому редакторі Word).

Використання EXCEL для математичних розрахунків та побудови графіків. Електронні таблиці (ЕТ) – комп'ютерна програма, за допомогою якої можна зберігати, обробляти, модифіковувати дані, представлені у вигляді таблиць. Числові дані

обробляти за допомогою ЕТ найзручніше, адже в ЕТ можна застосовувати формули для встановлення зв'язку між значеннями, що зберігаються в різних комірках. Автоматизація розрахунків підвищує ефективність та якість роботи. Саме тому ЕТ використовують у бухгалтерському обліку, торгівлі, при калькуляції робіт та послуг, у роботі банків та інших фінансових установ. Також за допомогою ЕТ можна легко будувати графіки, діаграми, гістограми. Використання ЕТ є надзвичайно корисним при розв'язуванні багатьох математичних задач.

Створення математичної моделі – один із важливих етапів розв'язування задач за допомогою ЕОМ. Побудова математичної моделі передбачає описання реальних об'єктів у термінах математики. Значну частину математичних моделей різноманітних об'єктів та процесів можна подати в досить зручному та компактному вигляді - у матричній формі.

Робота з матрицями. Багато прикладних задач економіки, техніки та інших галузей зводяться до розв'язування систем лінійних рівнянь. Засоби MS Excel є досить корисними для розрахунків лінійної алгебри, зокрема робота з матрицями та розв'язування систем лінійних рівнянь.

Транспонування матриці. Нехай існує матриця розмірністю *M·N* у певному діапазоні. Для транспонування потрібно в порожньому місці робочого аркуша виділити діапазон *N·M,* використати функцію TRANSPOSE (ТРАНСП) з категорії *Посилання й масиви.*

Обчислення детермінанта матриці. Нехай у певному діапазоні робочого аркуша існує матриця розмірністю $N \cdot N$. Для відшукання її детермінанта потрібно зробити активною комірку за межами діапазону цієї матриці та скористатися функцією MDETERM (МОП-РЕД) з категорії *Математичні.*

Знаходження матриці, оберненої до даної. Нехай існує матриця розмірністю $N \cdot N$ у повному діапазоні. Для відшукання матриці, оберненої до даної, потрібно в порожньому місці робочого аркуша виділити діапазон під обернену матрицю $N \cdot N$ та використати функцію MINVERSE (МОБР) з категорії *Математичні.*

Додавання, віднімання матриць та множений матриці на число. Додавання та віднімання матриць здійснюють над матрицями однакової розмірності. У MS Excel для додавання, віднімання матриць та множення матриці на число немає відповідних вбудованих стандартних функцій. Для здійснення цих операцій використовують формули, що вволять у відповідні комірки.

Множення матриць. Нехай у певному діапазоні робочого аркуша існує матриця розмірністю $M \cdot N$, а в іншому діапазоні - матриця розмірністю $N \cdot K$. Для множення цих двох матриць потрібно в порожньому місці робочого аркуша виділити діапазон розмірністю $M \cdot K$ під матрицю-результат та скористатися функцією МУМНОЖ (MMULT) з категорії *Математичні.*

Відшукання розв'язку системи N лінійних рівнянь з N невідомими.

Нехай задано систему *n* лінійних рівнянь з *n* невідомими:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_{21} \\ ..... \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n \end{cases} \quad (1)$$

Запишемо систему у вигляді матричного рівняння A • X = B, де

A – матриця системи $\begin{pmatrix} a_{11};a_{12};...;a_{1n} \\ a_{21};a_{22};...;a_{2n} \\ ..... \\ a_{n1};a_{n2};...;a_{nn} \end{pmatrix}$ (2)

X – матриця-стовпчик невідомих:

$$X=\begin{pmatrix} x_1 \\ x_2 \\ .... \\ x_n \end{pmatrix} \quad (3)$$

B – матриця-стовпчик вільних членів:

$$B=\begin{pmatrix} b_1 \\ b_2 \\ .... \\ b_n \end{pmatrix} \quad (4)$$

У розгорнутому вигляді систему можна представити таким чином:

$$\begin{pmatrix} a_{11};a_{12};...;a_{1n} \\ a_{21};a_{22};...;a_{2n} \\ ..... \\ a_{n1};a_{n2};...;a_{nn} \end{pmatrix} * \begin{pmatrix} x_1 \\ x_2 \\ .... \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ .... \\ b_n \end{pmatrix}$$ або

$$A \cdot X = B \quad (5)$$

Метод оберненої матриці

Розв'язуючи системи лінійних рівнянь за допомогою комп'ютера, доцільно розглядати метод оберненої матриці. У цьому випадку вважатимемо

матрицю такою, що має обернену (детермінант матриці не дорівнює нулю). Очевидно, що розв'язком системи (5) буде матриця-стовпчик $X=A^{-1}\cdot B$.

Метод Крамера

Для розв'язання систем лінійних рівнянь можна також скористатися методом Крамера, який базується на застосуванні теореми Крамера.

Теорема Крамера: Система и лінійних рівнянь з *n* невідомими, детермінант якої відмінний від нуля, має єдиний розв'язок, що визначається за формулами:

$x_1 = \frac{\Delta_1}{\Delta}$, $x_2 = \frac{\Delta_2}{\Delta}$, ……, $x_n = \frac{\Delta_n}{\Delta}$ де Δ - детермінант матриці А (2),

Δ_n - детермінант матриці А, у якої n-тий стовпчик замінено матрицею - стовпчиком вільних членів (4).

Відшукавши детермінанти потрібних матриць, легко знайдемо розв'язок системи (1).

6. МОВИ ПРОГРАМУВАННЯ

6.1. Мови програмування високого рівня

Традиції викладання інформатики і програмування, що склалися за останні десятиліття такі, що знайомство з мовами програмування починається, як правило, з імперативних алгоритмічних мов, таких як, наприклад, BASIC, Pascal, C++, C# та ін. Імперативні мови зазвичай протиставляються функціональним (Lisp, Haskell, F#) і логічним (Prolog). Кожен із видів мов програмування має свої особливості, які потрібно враховувати при виборі найефективнішого шляху вирішення задачі програмування.

Відмінною рисою програми, написаної імперативною мовою програмування, є те, що вона описує процес обчислення у вигляді послідовних інструкцій виконання, що змінюють внутрішній стан виконавця програми (наприклад, комп'ютера). Історично саме такий імперативний підхід був застосований першим для програмування ЕОМ (комп'ютерів): спочатку у вигляді машинних кодів, а згодом – за допомогою мов програмування високого рівня. У зв'язку з цим поняття «алгоритм», як покроковий опис процесу виконання програми, і «алгоритмізація», як процес розробки алгоритму, стали тієї фундаментальної основою, на якій будується все подальше вивчення технологій і методологій програмування.

Мова програмування Паскаль (Pascal) – це одна з найвідоміших алгоритмічних мов програмування загального призначення. Вона була створений Ніклаусом Віртом, як мова, що сприяє хорошому стилю програмування, використовує структурне програмування і структуровані дані. Свою назву нова мова

отримала на честь французького математика, фізика, літератора та філософа Блеза Паскаля.

Найбільш відомою реалізацією Паскаля, що забезпечила його широке поширення та розвиток, є мова та середовище програмування Turbo Pascal фірми Borland (з 1983 по 1994 рр.). Згодом мова Паскаль стала основою для мови Object Pascal та професійного середовища програмування Delphi (середовище успішно розвивається з 1995 року по теперішній час; первісна і найбільш відома її назва – Borland Delphi). Крім того, існує велика кількість інших компіляторів і середовищ розробки, що підтримують мову Паскаль або його діалекти (наприклад, Free Pascal, GNU Pascal). Окремо слід виділити версію мови та середовище програмування PascalABC.NET, розроблену в Південному федеральному університеті і засновану на платформі Microsoft .NET Framework.

Мова програмування C# (вимовляється «сі-шарп») – це універсальна об'єктно-орієнтована мова програмування. Перша загальнодоступна версія цієї мови з'явилася в лютому 2002 одночасно з виходом середовища розробки Microsoft Visual Studio.NET. Починаючи з того часу, мова C#, як і програмна платформа Microsoft .NET, постійно розвиваються і набувають все більш і більш широкі додаткові функціональні можливості. Для створення програм для платформи .NET з використанням мови C# існують різні засоби розробки. Найбільш популярним є інтегроване середовище розробки (ICP) Microsoft Visual Studio, і навіть його безкоштовна версія Microsoft Visual C# Express Edition.

В даний час мова програмування C# набуває широкого поширення не тільки як універсальна мова професійної розробки програмного забезпечення (порівняно з можливостями таких мов, як C + + і Java), але і як засіб початкового навчання програмування. У цьому сенсі мова C# становить серйозну конкуренцію таким мовам програмування, як BASIC і Pascal.

Відмінною особливістю мови програмування C# є те, що вона побудована на засадах об'єктно-орієнтованого програмування (ООП). При цьому будь-яка, навіть найпростіша програма є об'єктно-орієнтованою. Таким чином, ефективне навчання програмування мовою C# вимагає правильного та впевненого розуміння таких фундаментальних понять, як «об'єкт», «клас», «абстракція», «ієрархія», «інкапсуляція», «успадкування», «поліморфізм». Тим самим, одночасно з вивченням мови C# відбувається природне освоєння принципів об'єктно-орієнтованого програмування.

6.2. Мова програмування PascalABC

PascalABC.NET – це система програмування та мова Pascal нового покоління для платформи Microsoft .NET. Мова PascalABC.NET містить усі основні елементи сучасних мов програмування: модулі, класи, перевантаження операцій, інтерфейси, винятки, узагальнені класи, лямбда-вирази, а також деякі засоби паралельності. Система PascalABC.NET включає в себе також просте інтегроване середовище, орієнтоване на ефективне навчання сучасному програмуванню.

Створення PascalABC.NET диктувалося двома основними причинами: старіння стандартної мови Pascal і систем, побудованих на її основі, а також необхідність у сучасному простому, безкоштовному і потужному інтегрованому середовищі програмування.

Мова PascalABC.NET включає практично всю стандартну мову Паскаль, а також більшість мовних розширень мови Delphi. Проте цих засобів недостатньо для сучасного програмування. Тому PascalABC.NET доповнений рядом конструкцій, підпрограм, типів і класів, що дозволяє створювати додатки середньої складності, що легко читаються.

PascalABC.NET базується на передовій платформі програмування Microsoft.NET, яка забезпечує мову PascalABC.NET величезною кількістю стандартних бібліотек і дозволяє легко поєднувати її з іншими .NET-мовами: C#, Visual Basic.NET, керовану C++ тощо. Платформа .NET надає також такі мовні засоби як єдиний механізм обробки винятків, єдиний механізм управління пам'яттю у вигляді складання сміття, а також можливість вільного використання класів, успадкування, поліморфізму та інтерфейсів між модулями, написаними різними мовами. Мова PascalABC.NET використовує більшість засобів, що надаються платформою *NET*: єдина система типів, класи, інтерфейси, виключення, перевантаження операцій, узагальнені типи, методи розширення, лямбда-вирази. Мова дозволяє програмувати в класичному процедурному стилі, в об'єктно-орієнтованому стилі і містить безліч елементів для програмування у функціональному стилі.

Ключовими особливостями мови програмування та інтегрованого середовища розробки PascalABC.NET є:

1. Розширення мови Pascal, серед яких оператор foreach, внутрішньоблочні описи змінних, автовизначення типу при описі, вбудовані множини довільних типів, *case* по рядках, спрощений синтаксис модулів, методи в записах, операція *new* для створення об'єктів, визначення тіл методів всередині класів, цілі довільної довжини, багатовимірні динамічні масиви.

2. Найсучасніші засоби мов програмування: узагальнені класи та підпрограми, інтерфейси, перевантаження операцій, λ-вирази, виключення, збирання сміття, методи розширення, безіменні класи, автокласи.

3. Генерація ефективного коду для платформи.

4. Висока сумісність із Delphi.

5. Висока швидкість виконання програм.

6. Можливість доступу до великої кількості .NET-бібліотек від контейнерних класів до засобів роботи з мережею.

7. Середовище розробки з вбудованим налагоджувачем, що забезпечує підказки за кодом, перехід до визначення та реалізації підпрограми, шаблони коду, автоформатування коду.

8. Вбудований у середовище розробки дизайнер форм для швидкого створення віконних додатків.

9. Проста та ефективна растрова графічна бібліотека.

10. Засоби паралельного програмування у вигляді директив OpenMP.

11. Механізм перевірених завдань, що забезпечує автоматичну постановку та перевірку завдань.

12. Можливість запуску консольного компілятора під Mono у сучасних версіях Linux, можливість вбудовування PascalABC.NET у редактор Geany.

6.3. Розробка додатків мовою PascalABC

Інтегроване середовище розробки (ICP) PascalABC.NET орієнтоване на створення проектів малої та середньої складності. Вона неперевантажена і в той же час забезпечує розробника всіма необхідними засобами, такими як вбудований відладчик, засоби Intellisense (підказка за точкою, підказка за параметрами, підказка по імені), перехід до визначення і реалізації підпрограми, шаблони коду, автоформатування коду. У середу PascalABC.NET вбудований також дизайнер форм, що дозволяє створювати повноцінні віконні додатки в стилі RAD (Rapid Application Development - швидке створення додатків).

На відміну від багатьох професійних середовищ, середовище розробки PascalABC.NET не має громіздкого інтерфейсу і не створює безліч додаткових допоміжних файлів на диску при компіляції програми. Для невеликих програм це дозволяє дотримати принципу "Одна програма - один файл на диску".

Середі PascalABC.NET велику увагу приділено зв'язку запущеної програми з оболонкою: консольна програма, запущена з-під оболонки, здійснює введення-виведення в спеціальне вікно,

вбудоване в оболонку. Можна також запустити кілька програм одночасно – всі вони будуть контролюватись оболонкою.

Крім цього, внутрішні уявлення PascalABC.NET дозволяють створювати компілятори інших мов програмування і вбудовувати їх в середу розробки за допомогою спеціальних плагінів.

7. ЛАБОРАТОРНИЙ ПРАКТИКУМ
МЕТОДИЧНІ ВКАЗІВКИ

Опис лабораторного практикуму включає навчально-методичні матеріали до виконання дев'яти лабораторних робіт з курсу «Програмне забезпечення ПЕОМ».

Робота виконується як під час аудиторних занять, так і у вигляді самостійної позааудиторної роботи. Виконання кожної лабораторної роботи складається з трьох етапів:

1. Підготовка та отримання допуску до роботи.
2. Отримання індивідуального завдання та виконання основної частини роботи.
3. Оформлення та захист звіту про виконану роботу.

На початку кожної лабораторної роботи здійснюється повторення теоретичного матеріалу та перевірка готовності до виконання роботи за допомогою контрольних питань. Після отримання допуску до виконання роботи видається індивідуальний варіант завдання для самостійної роботи. На заключному етапі оформляється звіт про виконану роботу з описом отриманих результатів і виконується процедура захисту звіту.

Процедура захисту звіту полягає у перевірці:

1) правильності структури та оформлення звіту;

2) коректності одержаних результатів;

3) здатності дати пояснення та необхідне обґрунтування отриманим результатам.

Звіт повинен включати:

1. Титульний лист.

2. Завдання до лабораторної роботи.

3. Зміст звіту.

4. Опис результатів з кожної частини завдання.

5. Додаток (діаграми, тексти програм, зміст документів тощо).

Лабораторна робота №1

АПАРАТНІ ЗАСОБИ ЕОМ

Цілі та завдання лабораторної роботи

Цілями виконання лабораторної роботи є:

1. Закріплення знань про види та призначення апаратних засобів ЕОМ. Вивчення галузі застосування та функціональних можливостей сучасних комп'ютерів.

2. Придбання практичних навичок пошуку, обробки та аналізу інформації на задану тему в мережі інтернет.

В У процесі виконання лабораторної роботи вирішуються такі завдання:

1. Виконується пошук та аналіз інформації про види апаратних засобів ЕОМ, про призначення та характеристики окремих видів апаратних засобів.

2. Розробляється приклад можливого застосування апаратних засобів ЕОМ, персонального комп'ютера, сучасних інформаційно-комунікаційних технологій для вирішення деякої задачі.

Контрольні питання для допуску до роботи

1.Обчислювальна техніка. Комп'ютери.

2.Ознаки класифікації обчислювальної техніки.

3.Апаратні засоби ЕОМ.

4.Принципи впливу ЕОМ.

5.Покоління ЕОМ. Елементна база ЕОМ.

6.Архітектура ЕОМ.

7.Види забезпечення ЕОМ.

8.Програмне та апаратне забезпечення ЕОМ.

9.Інформаційне забезпечення ЕОМ.

10.Математичне забезпечення ЕОМ.

Порядок виконання роботи

Варіант індивідуального завдання визначає одне із завдань, для вирішення якого використовуються обчислювальні системи (електронно-обчислювальні машини, персональні комп'ютери, інші інформаційно-комунікаційні технології).

В процесі виконання лабораторної роботи необхідно:

1. Описати завдання з точки зору використовуваної інформації та вихідних даних: що є вихідними даними, що є результатом вирішення задачі? У якій формі вводяться вихідні дані та виводяться результати розв'язання задачі?

2. Охарактеризувати алгоритм розв'язання задачі: як реалізований процес введення та обробки даних, обчислень, подання проміжних і остаточних результатів?

3. Вказати, які апаратні засоби можуть використовуватися для реалізації обчислювальної частини розв'язання задачі. Сформулювати вимоги до обчислювальної системи з погляду продуктивності та ефективності.

4. Перерахувати апаратні засоби, які можуть використовуватися для введення вихідних даних і виведення результатів вирішення задачі. Вказати джерела та форму подання вихідних даних.

5. Розглянути варіанти використання сучасних інформаційно-комунікаційних технологій для роботи локальної мережі, у мережі Інтернет.

6. Розглянути варіанти використання сучасних засобів доповненої та віртуальної реальності.

7. Навести приклад програмного забезпечення, використання якого можливе для вирішення задачі. Описати вимоги програмного забезпечення до апаратних засобів: склад засобів, продуктивність, зручність використання.

8. Описати організаційні та технічні вимоги до роботи з обчислювальною системою. Описати вимоги щодо кваліфікації користувачів.

Варіанти індивідуальних завдань

1. Виконання математичних розрахунків.

2. Паралельні та розподілені обчислення.

3. Математичне та комп'ютерне моделювання фізичних процесів та технічних систем.

4. Імітаційне моделювання в галузі техніки та економіки.

5. Вирішення оптимізаційних завдань.

6. Розробка програмного забезпечення.

7. Проектування технічних приладів. Системи автоматизованого проектування.

8. Розробка тривимірних моделей технічних пристроїв.

9. Розробка проектної документації та креслень технічних пристроїв.

10. Системи навчання, тренажери, симулятори.

11. Системи віртуальної дійсності.

12. Системи управління базами даних.

13. Робота з базою даних як клієнт.

14. Робота комп'ютера як сервер у локальній мережі, в мережі Інтернет.

15. Робота комп'ютера як клієнта в локальній мережі, в мережі Інтернет.

16. Обробка відео та аудіо.

17. Робота з растровими та векторними зображеннями.

18. Розпізнавання образів.

19. Системи штучного інтелекту.

20. Експертні системи. Системи підтримки та прийняття рішень.

21. Комп'ютерні ігри.

22. Автоматизовані системи управління технологічним процесом.

23. Автоматизовані інформаційні системи.

24. Системи автоматизації бізнес-процесів.

25. Системи електронного документообігу.

26. Геоінформаційні системи.

Лабораторна робота №2

АРХІТЕКТУРА КОМП'ЮТЕРА

Цілі та завдання лабораторної роботи

Цілями виконання лабораторної роботи є:

1. Закріплення знань про архітектуру комп'ютера. Вивчення елементів архітектури, їх призначення і характеристика.

2. Формування практичних навичок пошуку, обробки та аналізу інформації на задану тему в мережі Інтернет.

В У процесі виконання лабораторної роботи вирішуються такі завдання:

1. Виконується пошук та аналіз інформації про заданий вид архітектури комп'ютера, про її елементи і характеристики.

2. Вивчаються приклади практичного застосування архітектури комп'ютера заданого виду для вирішення завдань розробки обчислювальних систем різного призначення.

Контрольні питання для допуску до роботи

1. Архітектура комп'ютера.

2. Класифікаційні ознаки та характеристики архітектури комп'ютера.

3. Архітектура фон Неймана.

4. Гарвардська архітектура. Її переваги та недоліки.

5. Магістрально-модульний принцип роботи ЕОМ.

6. Види пристроїв, що підключаються до системної шини.

7. Процесор.

8. Арифметико-логічний пристрій.

9. Пристрій керування.

10. Системна шина.

11. Пристрої введення.

12. Пристрої виведення.

13. Пристрої запам'ятовування.

14. Види архітектур за набором команд.

15. Процесорна архітектура CISC.

16. Процесорна архітектура RISC.

Порядок виконання роботи

Варіант індивідуального завдання визначає вид архітектури комп'ютера для вивчення та опису.

В процесі виконання лабораторної роботи необхідно:

1. Вивчити класифікаційні ознаки та характеристики архітектури комп'ютера (розрядність інтерфейсу, набір команд, кількість процесорів тощо).

2. Описати архітектуру комп'ютера, задану варіантом індивідуального завдання. Розглянути: рівні та елементи архітектури, використовувані технічні рішення, елементну базу.

3. Описати призначення архітектури, ефективність її використання в обчислювальних системах для вирішення завдань різного виду.

4. Описати окремі рівні архітектури: набір команд, що використовується, мікроархітектуру, мікропрограму (мікрокод).

5. Навести приклади використання архітектури в обчислювальних системах. Описати поширеність архітектури та її

перспективність. Вказати процесори, розроблені відповідно до даної архітектури.

6. Підготувати доповідь і презентацію з інформацією про архітектуру комп'ютера заданого виду, її практичного застосування, приклади використання архітектури в сучасних обчислювальних системах.

Варіанти індивідуальних завдань

1. Види архітектур з набору команд:

1) ASIP;

2) CISC;

3) EDGE;

4) EPIC;

5) MISC;

6) URISC;

7) RISC;

8) VLIW;

9) ZISC.

2. Процесорні архітектури на базі CISC-технологій:

1) x86, IA-32;

2) x86-64.

3. Процесорні архітектури на базі RISC-технологій:

1) ARM;

2) Atmel (AVR);

3) DEC Alpha;

4) PA-RISC;

5) Intel i960;

6) MIPS;

7) POWER;

8) PowerPC;

9) SPARC.

Лабораторна робота № 3

ПРИСТРІЙ ПЕРСОНАЛЬНОГО КОМП'ЮТЕРА

Цілі та завдання лабораторної роботи

Цілями виконання лабораторної роботи є:

1. Закріплення знань про влаштування персонального комп'ютера. Вивчення окремих елементів обчислювальної системи, їх призначення, характеристик, взаємної сумісності.

2. Придбання навичок аналізу конфігурації персонального комп'ютера, складання нової конфігурації з урахуванням розв'язуваних за його допомогою завдань.

У процесі виконання лабораторної роботи вирішуються такі завдання:

1. Виконується пошук та аналіз інформації про елементи пристрою персонального комп'ютера.

2. Розробляється конфігурація персонального комп'ютера на вирішення конкретної задачі.

Контрольні питання для допуску до роботи

1. Персональний комп'ютер. Характеристики ПК.

2. Основні устрою ПК.

3. Периферійні пристрої комп'ютера.

4. Обладнання обміну інформацією.

5. Пристрої обробки інформації.

6. Носії інформації.

7. Монітор. Клавіатура. Миша.

8. Материнська плата.

9. Процесор.

10. Відеокарти.

11. Оперативна пам'ять.

12. Жорсткий диск.

Порядок виконання роботи

Варіант індивідуального завдання визначає завдання (область застосування обчислювальної системи), для вирішення якої необхідно розробити конфігурацію персонального комп'ютера.

В процесі виконання лабораторної роботи необхідно:

1. Для заданого завдання (сфери застосування) визначити загальні вимоги до продуктивності обчислювальної системи.

2. Виконати аналіз окремих факторів, що впливають на загальні вимоги до продуктивності, і сформулювати окремо:

1) вимоги до швидкодії центрального процесора;

2) вимоги до обсягу та швидкодії оперативної пам'яті;

3) вимоги до продуктивності відеоадаптера (відеокарти);

4) вимоги до пропускної спроможності системної шини;

5) вимоги до швидкості читання та запису інформації з використанням жорсткого диска.

3. Виконати аналіз вимог до персонального комп'ютера з погляду:

1) способу введення вихідних даних;

2) способу виведення результатів;

3) необхідності використання периферійних пристроїв;

4) необхідності забезпечення відмовостійкої роботи у разі виникнення різних аварійних ситуацій;

5) необхідності роботи в локальній мережі, в мережі Інтернет.

4. Виконати пошук сучасних моделей комплектуючих (елементів пристрою) персонального комп'ютера, що задовольняють сформульованим вимогам.

5. Відібрати комплектуючі (елементи пристрою) персонального комп'ютера, які є сумісними один з одним. Вказати загальну вартість обчислювальної системи.

6. Скласти документ, що описує склад підготовленої конфігурації персонального комп'ютера. Обгрунтувати рішення, прийняті в процесі відбору комплектуючих.

7. Дати оцінку вартості та продуктивності складеної конфігурації персонального комп'ютера.

Варіанти індивідуальних завдань

В як список варіантів індивідуальних завдань використовується перелік завдань (областей застосування обчислювальної системи) з лабораторної роботи № 1.

Лабораторна робота №4

ПРОГРАМНІ ЗАСОБИ ЕОМ

Цілі та завдання лабораторної роботи

Цілями виконання лабораторної роботи є:

1. Закріплення знань про види та призначення програмних засобів ЕОМ. Вивчення галузі застосування та функціональних можливостей сучасного програмного забезпечення.

2. Придбання практичних навичок аналізу програмного забезпечення, встановленого на персональному комп'ютері.

В У процесі виконання лабораторної роботи вирішуються такі завдання:

1. Виконується аналіз системного та прикладного програмного забезпечення, встановленого на персональному комп'ютері.

2. Складається звіт про встановлене системне та прикладне програмне забезпечення.

3. Перевіряється наявність оновлень та нових версій програмного забезпечення.

Контрольні питання для допуску до роботи

1. Програмні засоби ЕОМ.

2. Програма. Програмний продукт.

3. Програмне забезпечення. Версії ПЗ. Оновлення програмного забезпечення.

4. Програма сервер. Програма-клієнт.

5. Пакет програм.

6. Види програмного забезпечення.

7. Системне програмне забезпечення.

8. Прикладне програмне забезпечення.

9. Інструментарій технології програмування.

Порядок виконання роботи

Зміст індивідуального завдання визначається конфігурацією персонального комп'ютера та складом встановленого на ньому системного та прикладного програмного забезпечення.

В процесі виконання лабораторної роботи необхідно:

1. Виконати аналіз системного програмного забезпечення наявного в розпорядженні персонального комп'ютера, у тому числі:

1.1. операційної системи;

1.2. програми BIOS;

1.3. драйверів встановлених пристроїв.

2. Виконати аналіз прикладного програмного забезпечення, встановленого на персональному комп'ютері.

3. Скласти спільний звіт про програмне забезпечення, яке встановлено на комп'ютері. Привести у звіті такі дані про програмні продукти:

3.1. назву;

3.2. версія;

3.3. загальне призначення;

3.4. характеристики;

3.5. функціональні можливості.

4. Перевірити встановлене програмне забезпечення на

наявність:

4.1. функціональних оновлень;

4.2. пакетів для виправлення помилок;

4.3. нових версій.

5. Перелічити вимоги системного та прикладного програмного забезпечення до апаратних засобів комп'ютера:

5.1. вимоги до оперативної пам'яті;

5.2. вимоги до місця на жорсткому диску;

5.3. вимоги до загальної швидкодії комп'ютера;

5.4. вимоги до відеокарти;

5.5. інші вимоги.

6. Для встановлених на комп'ютері програм знайти та описати аналоги, що мають таке ж призначення та схожу функціональність.

Варіанти індивідуальних завдань

В якості персонального комп'ютера для виконання лабораторної роботи може виступати:

1. Комп'ютер загального користування, встановлений у обчислювальній лабораторії.

2. Власний комп'ютер (ноутбук).

Лабораторна робота №5

ОПЕРАЦІЙНІ СИСТЕМИ

Цілі та завдання лабораторної роботи

Цілями виконання лабораторної роботи є:

1. Закріплення знань про види та призначення операційних систем. Вивчення характеристик та функціональних можливостей сучасних операційних систем.

2. Формування практичних навичок установки, настройки та обслуговування операційної системи, встановленої на персональному комп'ютері.

У процесі виконання лабораторної роботи вирішуються такі завдання:

1. Виконується аналіз та вивчення операційної системи, встановленої на персональному комп'ютері. Складається звіт про поточні налаштування операційної системи.

2. Здійснюється переустановка операційної системи з використанням наявного дистрибутива. Після встановлення операційної системи виконується відновлення налаштувань і перевірка працездатності.

Контрольні питання для допуску до роботи

1. Операційна система.
2. Класифікація операційних систем.
3. Мережеві ОС.
4. Розраховані на багато користувачів ОС.
5. Багатозадачні ОС.

6. Системи пакетної обробки.
7. Системи поділу часу.
8. Системи реального часу.
9. Розділи жорсткого диска.
10. Дистрибутив операційної системи.
11. Перевірка та дефрагментація жорсткого диска.

Порядок виконання роботи

Зміст індивідуального завдання визначається конфігурацією персонального комп'ютера та версією встановленої на ньому операційної системи.

В процесі виконання лабораторної роботи необхідно:

1. Виконати аналіз того, яка операційна система встановлена на виділеному для роботи персональному комп'ютері.

2. Скласти загальний звіт про поточні налаштування операційної системи:

2.1. користувачі (з правами адміністратора та звичайні користувачі);

2.2. мережеві підключення (до локальної мережі, до мережі інтернет);

2.3. встановлені драйвери пристроїв (внутрішні пристрої, зовнішні пристрої);

2.4. встановлене прикладне програмне забезпечення;

2.5. версія операційної системи;

2.6. налаштування відеоадаптера (дозвіл і частота оновлення екрана монітора);

2.7. існуючі розділи жорсткого диска.

3. Перевірити операційну систему на наявність:

3.1. функціональних оновлень;

3.2. пакетів для виправлення помилок;

3.3. нових версій.

4. У разі оновлення операційної системи – встановити їх.

5. Виконати діагностику жорсткого диска:

5.1. перевірити жорсткий диск на помилки;

5.2. виконати дефрагментацію жорсткого диска.

6. Виконати переустановку операційної системи, використовуючи наявний дистрибутив.

7. Після переустановки операційної системи застосувати налаштування, зафіксовані у звіті, і перевірити загальну працездатність системи.

Варіанти індивідуальних завдань

В якості персонального комп'ютера для виконання лабораторної роботи може виступати:

1. Комп'ютер загального користування, встановлений у обчислювальній лабораторії.

2. Власний комп'ютер учня (ноутбук).

Лабораторна робота № 6

ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Цілі та завдання лабораторної роботи

Цілями виконання лабораторної роботи є:

1. Закріплення знань про види та призначення різних видів прикладного програмного забезпечення. Вивчення характеристик і функціональних можливостей сучасних програмних продуктів.

2. Формування практичних навичок встановлення, налаштування та обслуговування прикладного програмного забезпечення, встановленого на персональному комп'ютері.

У процесі виконання лабораторної роботи вирішуються такі завдання:

1. Виконується аналіз та вивчення прикладного програмного забезпечення, встановленого на персональному комп'ютері.

2. Проводиться переустановка програмного забезпечення з використанням наявних дистрибутивів.

Контрольні питання для допуску до роботи

1. Прикладне програмне забезпечення.
2. Види прикладних програм.
3. Програма обробки тексту.
4. Програми обробки таблиць та масивів даних.
5. Програма обробки графічних зображень.
6. Програми обробки аудіо- та відео сигналів.
7. Програми обробки чисел.
8. Програми опрацювання знань.

9. Комунікаційні програми.

10. Програма автоматизації робіт.

Порядок виконання роботи

Зміст індивідуального завдання визначається конфігурацією персонального комп'ютера та конкретним переліком встановленого на ньому прикладного програмного забезпечення (див. завдання до попередньої лабораторної роботи).

В процесі виконання лабораторної роботи необхідно:

1. Виконати аналіз того, яке прикладне програмне забезпечення встановлено на виділеному для роботи персональному комп'ютері.

2. Скласти загальний звіт про встановлене програмне забезпечення. Привести у звіті такі дані про програмні продукти:

2.1. назву;

2.2. версія;

2.3. загальне призначення;

2.4. характеристики;

2.5. функціональні можливості.

3. Перевірити встановлене прикладне на наявність:

3.1. функціональних оновлень;

3.2. пакетів для виправлення помилок;

3.3. нових версій.

4. У разі наявності оновлень програмного забезпечення - встановити їх.

5. Перерахувати вимоги прикладного програмного забезпечення до апаратних засобів комп'ютера:

5.1. вимоги до оперативної пам'яті;

5.2. вимоги до місця на жорсткому диску;

5.3. вимоги до загальної швидкодії комп'ютера;

5.4. вимоги до відеокарти;

5.5. інші вимоги.

6. Для встановлених програмних продуктів визначити і описати аналоги, що мають таке ж призначення і схожу функціональність.

7. Виконати перевстановлення програмного забезпечення, використовуючи наявні дистрибутиви.

8. Після переустановки прикладного програмного забезпечення перевірити його працездатність.

Лабораторна робота № 7

МОВИ ПРОГРАМУВАННЯ ВИСОКОГО РІВНЯ

Цілі та завдання лабораторної роботи

Цілями виконання лабораторної роботи є:

1. Ознайомлення з інтегрованим середовищем розробки (ICP) Microsoft Visual C#.

2. Формування навичок розробки та тестування консольних програм в ICP Microsoft Visual C#.

В У процесі виконання лабораторної роботи вирішуються такі завдання:

1. Відпрацьовуються основні операції з роботі з проектом консольної програми в ICP Microsoft Visual C #.

2. Вивчаються можливості середовища розробки для налагодження програм (пошук та виправлення синтаксичних та логічних помилок, обробка помилок часу виконання).

Контрольні питання для допуску до роботи

1. Програмування.
2. Мови програмування.
3. Мови програмування високого рівня.
4. Алгоритмічні мови програмування.
5. Мови функціонального програмування.
6. Мови логічного програмування.
7. Мова Паскаль.
8. Мова C#.

Порядок виконання роботи

1. Запустіть ICP Microsoft Visual C# 2008 Express Edition.

2. У меню *Файл* виберіть пункт *Створити проект*. На екрані з'явиться вікно з переліком усіх видів проектів, які можна створювати серед розробки Microsoft Visual C#.

3. В основному вікна *Створити проект* виберіть шаблон *Консольний додаток*, а в полі *Ім'я* у нижній частині вікна введіть назву першого проекту в даній лабораторній роботі: *Lab01_01*.

4. Для створення проекту консольної програми із заданим ім'ям натисніть у вікні *Створити проект* кнопку *ОК*.

5. Перейдіть на вкладку *Початкова сторінка* шляхом виконання клацання лівої кнопки миші по заголовку вкладки. Після перемикання на цю вкладку закрийте її, вибравши меню *Файл* пункт *Закрити*.

6. Ознайомтеся з текстом програми, що відображається на вкладці *Program.cs*. Звертайте увагу на можливості редактора коду: підсвічування синтаксису, можливість згортання блоків програми, поява підказок при наведенні покажчика миші тощо.

```
using System;
using System.Collections.Generic;
using System.Linq; using System.Text;
namespace Lab01_01
{
    class Program
    {
        static void Main(string[] args)
```

```
        {
        }
    }
}
```

7. Для перегляду структури рішення, відкритого в середовищі розробки та проекту, що входить до його складу, відкрийте панель *Оглядач рішень*. Для цього в меню *Вид* виберіть пункт *Оглядач рішень*.

8. Панель *Оглядач рішень* за замовчуванням є спливаючою: вона автоматично ховається, якщо не є активною. Щоб панель відображалася в середовищі розробки постійно, натисніть кнопку *Автоматично приховувати* у заголовку цієї панелі.

9. Збережіть усі файли, що входять до складу створеного проекту нової консольної програми. Для цього в меню *Файл* виберіть пункт *Зберегти все*. Після вибору цього пункту на екрані з'явиться вікно *Зберегти проект*.

10. Задайте як розташування файлів поточного рішення шлях до каталогу: *H:\ПІАС\Lab01*.

11. Натисніть кнопку *Зберегти*. Вікно *Зберегти проект* закриється, і всі файли, що входять до складу рішення, будуть збережені в зазначених каталогах із заданими іменами.

12. Закрийте файл *Program.cs*, що відкритий у редакторі коду. Для цього в меню *Файл* виберіть пункт *Закрити*.

13. Виконайте подвійне клацання по елементу *Program.cs* ієрархічного дерева рішення, що відображається в панелі *Оглядач рішень*. В результаті виконання цієї дії буде знову відкрито вкладку

редактора коду з завантаженим в неї текстом файлу програми *Program.cs.*

14. Переведіть панель *Оглядач рішень* в стан, при якому вона в неактивному стані буде автоматично ховатися.

15. Запустіть програму для виконання. Для цього необхідно виконати одну з наступних дій: вибрати в меню *Налагодження* пункт *Почати налагодження*, скористатися кнопкою з піктограмою зеленого трикутника на панелі інструментів або натиснути клавішу <*F5*> клавіатурі. Зверніть увагу на те, що після запуску програми та створення вікна консольної програми воно буде відразу закрито.

16. Змініть текст програми таким чином, щоб після запуску користувачеві пропонувалося б ввести своє ім'я, після чого програма виводила б вітальне повідомлення. Для вирішення цього завдання до коду методу Main додайте наступні оператори:

```
Console.WriteLine("Додаток Lab01_01.");
Console.WriteLine("Введіть своє ім'я:");
string name = Console.ReadLine();
Console.WriteLine("Доброго дня, " + name + "!");
Console.WriteLine("Для завершення роботи" +
        "програми натисніть будь-яку клавішу.");
Console.ReadKey();
```

17. Запустіть програму для виконання. Якщо в процесі набору тексту програми не були допущені синтаксичні помилки, то після запуску програми у створеному вікні консольної програми буде відображатися пропозиція ввести Ваше ім'я.

18. Спробуйте змінити розмір та розташування вікна консольної програми на екрані. Зверніть увагу на пов'язану з виконуваним додатком кнопку, яка була створена на панелі завдань Windows після запуску програми.

19. Не виконуючи жодних дій у консольному додатку, перейдіть в середу розробки за допомогою кнопки *Lab01_01 (Виконання)* на панелі завдань Windows або клавіші <*Alt*>+<*Tab*>. Зверніть увагу на зміни у зовнішньому вигляді середовища розробки, що відбулися після запуску програми, що розробляється *Lab01_01*.

20. Перейдіть назад до консольної програми *Lab01_01*. Для продовження роботи з програмою введіть Ваше ім'я та натисніть клавішу *Enter*. На екран має бути виведене вітальне повідомлення, що включає раніше введене ім'я.

21. Перевірте правильність роботи програми, ввівши в одному рядку свої прізвище, ім'я та по батькові. Для завершення роботи програми натисніть будь-яку клавішу.

22. Відкрийте та закріпіть на екрані панель *Оглядач рішень*. Закрийте вкладку вікна редактора коду з текстом програми.

23. Завершіть роботу над додатком, закривши в середовищі розробки рішення *Lab01_01*. Для цього в меню Файл виберіть пункт *Закрити рішення*. Зверніть увагу на зміни у зовнішньому вигляді середовища розробки, що відбулися після закриття рішення.

24. Відкрийте рішення *Lab01_01* з каталогу, що був вказаний Вами у вікні *Зберегти проект* під час виконання одного з попередніх пунктів. Для цього в меню *Файл* виберіть пункт

Відкрити проект. В результаті виконання цієї дії на екрані з'явиться вікно діалогу, в якому в структурі каталогів необхідно знайти і потім вибрати файл рішення *Lab01_01.sln*.

25. Запустіть програму. Перевірте правильність її роботи. Завершіть роботу програми. Закрийте рішення. Закрийте середовище розробки.

Лабораторна робота № 8

МОВА ПРОГРАМУВАННЯ PASCALABC

Мета та завдання лабораторної роботи

Метою виконання лабораторної роботи вивчення об'єктно-орієнтованого програмування на мові PascalABC, закріплення навичок розробки об'єктно-орієнтованих консольних додатків в ICP PascalABC.NET, а також практичне застосування засобів реалізації поліморфізму: динамічного перекриття, віртуальних та абстрактних методів.

У процесі виконання лабораторної роботи вирішуються такі завдання:

1. Розробляється консольний додаток «Lab02_01», що ілюструє на прикладі ієрархії класів транспортних засобів реалізацію поліморфізму за допомогою віртуальних методів та динамічного перекриття.

2. Розробляється консольний додаток «Lab02_02», що ілюструє на прикладі ієрархії класів геометричних фігур опис та використання абстрактних методів.

3. Розробляється консольна програма «Lab02_03» для вирішення завдань, що входять до складу індивідуального завдання.

Контрольні питання для допуску до роботи

1. Мова програмування PascalABC.
2. Інтегроване середовище розробки PascalABC.NET.
3. Платформа Microsoft.
4. Використання бібліотек платформи .NET у програмах мовою PascalABC.NET.
5. Основні особливості мови PascalABC.NET.
6. Розширення мови PascalABC.NET
7. Можливості мови PascalABC.NET для створення програм з графічним інтерфейсом користувача.

Порядок виконання роботи

1. Запустіть ICP PascalABC.NET. Створіть нову консольну програму. Збережіть файли проекту в каталог "H:\PAS\Lab02\Lab02_01\". Створіть новий модуль "Transport.pas" і збережіть його в цей же каталог.

2. Наберіть текст модуля, що ілюструє реалізацію поліморфізму з прикладу ієрархії класів транспортних засобів:

```
unit Transport;
interface
type
    CTransport = class
```

```
        Speed, Heading: Real;
        procedure Move;       virtual;
        procedure Stop; virtual;
    end;
    CCar = class (CTransport)
        procedure Move; override;
        procedure Stop; override;
    end;
    CPlane = class (CTransport)
        procedure Move; override;
        procedure Stop; override;
    end;
    CShip = class (CTransport)
        procedure Move; override;
        procedure Stop; override;
    end;
implementation
procedure CTransport.Move;
begin
    WriteLn(' Транспортний засіб почав рух');
    WriteLn(' зі швидкістю ', Speed:6:2 , ' км/ч');
    WriteLn('курсом', Heading:6:2, 'градусів.');
end;
procedure CTransport.Stop;
begin
    WriteLn(' Транспортний засіб припинив рух.');
```

```
    end;
    procedure CCar.Move;
    begin
        WriteLn (' Автомобіль від'їхав від узбіччя,');
        WriteLn (' розігнався до  до швидкості', Speed:6:2,
    км/год');
        WriteLn(' і рухається  курсом',  Heading:6:2,
    'градусів.');
end;
procedure CCar.Stop;
begin
    WriteLn(' Автомобіль загальмував і зупинився біля
узбіччя.')
end;
procedure CPlane.Move;
begin
    WriteLn ('Літак виконав зліт і летить зараз');
    WriteLn(' зі швидкістю ', Speed:6:2 , ' км/ч');
    WriteLn('курсом', Heading:6:2, 'градусів.');
end;
procedure CPlane.Stop;
begin
    WriteLn('Літак виконав посадку.');
end;
procedure CShip.Move;
begin
```

```
        WriteLn ( 'Корабель відійшов від пристані, ');
        WriteLn(' ліг на курс ', Heading:6:2 , ' градусів');
        WriteLn(' і рухається зі швидкістю ', Speed:6:2 , ' км/год.')
    end;
    procedure CShip.Stop;
    begin
        WriteLn(' Корабель зупинився.');
    end;
    end.
```

3. Вставте в текст основної програми оголошення змінної класу *CTransport* та оператори:

```
var
    Transp : CTransport;
begin
    Randomize;
    Writeln ('Transp - об'єкт класу CTransport');
    Transp := CTransport.Create();
    Transp.Speed := Random(500)+1;
    Transp.Heading := Random(360);
    Transp.Move();
    Transp.Stop();
    Transp.Free();

    Writeln;
    Writeln('Transp - об'єкт класу CCar');
    Transp := CCar.Create();
```

```
Transp.Speed := Random(200)+1;
Transp.Heading := Random(360);
WriteLn('Транспортний засіб не рухається');
WriteLn('Натисніть <Enter>');
ReadLn;
Transp.Move();
WriteLn('Транспортний засіб почав рух.');
WriteLn('Натисніть <Enter>');
ReadLn;
Transp.Stop();
WriteLn('Транспортний засіб зупинено.');
WriteLn('Натисніть <Enter>');
ReadLn;
Transp.Free();

ReadLn
end.
```

4. Запустіть програму. Зверніть увагу на текст, який виводиться у вікно консольної програми. Завершіть роботу програми.

5. Додайте до тексту програми оголошення наступних змінних:

```
Car   : CCar;
Ship  : CShip;
Plane : CPlane;
T     : array of CTransport;
```

I : Integer;

6. Перед останнім викликом процедури *ReadLn* додайте в текст програми оператори:

```
Car := CCar.Create();
Car.Speed := Random(200)+1;
Car.Heading := Random(360);
Car.Move();
Car.Stop();

Plane := CPlane.Create();
Plane.Speed := Random(1000)+301;
Plane.Heading := Random(360);
Plane.Move();
Plane.Stop();

Ship:= CShip.Create();
Ship.Speed := Random(100)+1;
Ship.Heading := Random(360);
Ship.Move();
Ship.Stop();

WriteLn('Автомобіль, Літак та Корабель', 'не рухаються.');
WriteLn('Натисніть <Enter>');
ReadLn;

SetLength(T, 3);
```

```
T[0]: = Car;
T[1]: = Plane;
T[2]: = Ship;

for I:=0 to 2 do
        T[I].Move();
WriteLn('Автомобіль, Літак та Корабель', 'почали рух.');
WriteLn('Натисніть <Enter>');
ReadLn;

for I:=0 to 2 do
        T[I].Stop();
WriteLn('Всі транспортні засоби зупинені.');

for I:=0 to 2 do
T [I]. Free ();
T := nil;
```

7. Запустіть програму. Зверніть увагу на текст, який виводиться у вікно консольної програми. Завершіть роботу програми.

8. Самостійно допрацюйте програму, додавши до ієрархії класів транспортних засобів класи *CTrain* (поїзд) та *CHelicopter* (вертоліт). Збільшіть розмір масиву *T* до п'яти елементів і, за аналогією з об'єктами, що вже є, опишіть, створіть і використовуйте як елементи масиву *T* об'єкти класів *CTrain* і *CHelicopter*.

9. Завершіть роботу над проектом «Lab02_01».

10.Створіть нову консольну програму. Збережіть файли проекту в каталог "H: PAS Lab02 Lab02_02". Створіть новий модуль «Figure.pas» і збережіть його в цей же каталог.

11.Наберіть текст модуля, що ілюструє реалізацію поліморфізму на прикладі ієрархії класів геометричних фігур:

```
unit Figure;
interface
type
    CFigure = class
        procedure    InputDataAndCalcS;
        procedure    InputData;    virtual; abstract;
        функція S: Real;           virtual; abstract;
    end;
    CRect = class (CFigure)
        a, b: Real;
        procedure  InputData;    override;
        функція S: Real;         override;
    end;
    CCircle = class (CFigure)
        r: Real;          InputDat
        procedure  a;             override;
    ункція S: Real;                   override;
  end;
implementation

procedure CFigure.InputDataAndCalcS;
```

```
begin
    InputData();
    WriteLn( 'Площа фігури S = ', S():8:3)
end;

procedure CRect.InputData;
begin
    WriteLn('Введіть розміри сторін прямокутника:');
    ReadLn(a,b)
end;

function CRect.S: Real;
begin
    S := a*b
end;

procedure CCircle.InputData;
begin
    WriteLn('Введіть радіус кола:'); ReadLn(r)
end;

function CCircle.S: Real;
begin
    S: = PI * r * r;
end;
```

```
end.
```

12. Вставте в текст основної програми оголошення змінних та оператори:

```
var
    F: array of CFigure;
    I, Kind: Integer;
    STotal: Real;

begin
    SetLength(F, 5);
    STotal: = 0;

for I := 0 to 4 do
    begin
        WriteLn('Задайте вигляд', I, '-ї фігури');
        WriteLn('1 - коло, 2 - прямокутник');
        ReadLn (Kind);
        case Kind of
                1: F[I]: = CCircle.Create();
                2: F[I]: = CRect.Create();
    else
        WriteLn('Помилка!');
        ReadLn
    end;
    F[I].InputDataAndCalcS();
```

```
        STotal := STotal + F[I].S();
        end;

WriteLn('Загальна площа всіх фігур STotal = '+ FloatToStr(STotal) );

for I:=0 to 4 do
        F [I]. Free ();
F := nil;
        ReadLn
end.
```

13.Запустіть програму. Задайте вигляд фігур і вводьте дані про їх розмір. Зверніть увагу на текст, який виводиться у вікно консольної програми. Завершіть роботу програми.

14.Самостійно допрацюйте програму таким чином, щоб за її допомогою окрім кола та прямокутника можна бути обчислювати площі та таких геометричних фігур, як квадрат, трикутник, паралелограм, ромб, кільце. Для цього в модулі «Figure.pas» виконайте опис та реалізацію класів-нащадків *CSquare, CTriangle, CRing* та ін., а в основній програмі – передбачте можливість створення екземплярів цих класів та виклику їх методів. Знайдіть значення сумарної площі всіх фігур.

Варіанти індивідуальних завдань

Створити консольну програму, для якої реалізувати одне із запропонованих нижче завдань. У всіх варіантах, крім зазначених у завданні операцій, обов'язково повинні бути реалізовані такі

методи:

- метод ініціалізації Init;
- введення з клавіатури Read;
- виведення на екран Display.

Усі операції мають бути реалізовані у класі. Крім того, звернення до полів класу має здійснюватися через їх властивості.

1. Створити клас *Vector2D*, який задається парою чисел. Реалізувати: додавання, віднімання векторів та множення вектора на скаляр.

2. Створити клас *Vector2D*, який задається парою чисел. Реалізувати: обчислення довжини вектора, порівняння векторів та скалярний добуток векторів.

3. Створити клас *Money* для роботи із грошовими сумами. Число має бути представлене двома полями: гривні та копійки. Реалізувати: додавання, віднімання та операцію порівняння.

4. Створити клас *Triangle* для представлення трикутника. Поля даних повинні включати кути та сторони. Потрібно реалізувати операції: обчислення площі, обчислення периметра, і навіть визначення виду трикутника (рівносторонній, рівнобедрений, прямокутний).

5. Створити клас *Angle* для роботи з кутами на площині, що задаються величиною в градусах та хвилинах. Обов'язково мають бути реалізовані: переведення в радіани, отримання синуса, збільшення та зменшення кута на задану величину.

6. Створити клас *Point* для роботи з точками на площині. Координати точки – декартові. Обов'язково мають бути реалізовані:

визначення відстані на початок координат, відстань між двома точками, порівняння збіг і розбіжність.

7. Створити клас *Fraction* для роботи з дрібними числами. Число має бути представлене двома полями: ціла частина зі знаком і дробова частина – те, що після коми. Реалізувати арифметичні операції додавання, віднімання та операцію порівняння.

8. Створити клас *Drobi* для роботи з дрібними числами. Число має бути представлене двома полями: чисельником та знаменником. Реалізувати операції складання, віднімання та виділення цілої частини.

9. Створити клас *Cylinder*. Полями класу є висота циліндра та радіус основи. Потрібно реалізувати операції: обчислення об'єму циліндра, обчислення площі основи та площі бічної поверхні.

10. Створити клас *Time*. Полями класу є два числа: години та хвилини. Реалізувати метод приведення часу на хвилини, операцію збільшення часу на задану величину.

11. Створити клас *LineFunc* для роботи з лінійними функціями виду $y(x)=Ax+B$. Полями класу є пара чисел: коефіцієнти A і B. Реалізувати обчислення значення функції y (x) для заданого значення x.

12. Створити клас *SqrEqn* для роботи з рівняннями виду $Ax^2+Bx+C=0$. Полями класу є коефіцієнти A, B і C. Реалізувати обчислення коренів рівняння. Обов'язково реалізувати перевірку додатності дискримінанта.

Лабораторна робота №9

РОЗРОБКА НАЙПРОСТІШИХ ДОДАТКІВ НА МОВІ ПРОГРАМУВАННЯ PASCALABC

Цілі та завдання лабораторної роботи

Цілями виконання лабораторної роботи є:

1. Придбання практичних навичок реалізації на мові програмування PascalABC алгоритмів лінійної структури, алгоритмів з розгалуженнями та циклами.

2. Закріплення навичок розробки, тестування та налагодження консольних програм в ІСР PascalABC.

У процесі виконання лабораторної роботи вирішуються такі завдання:

1. На прикладі вирішення конкретних завдань вивчаються можливості мови програмування PascalABC з реалізації алгоритмів лінійної, розгалуженої та циклічної структури.

2. Освоюються можливості ІСР PascalABC для налагодження програм. Набуваються навички налагодження консольних додатків.

3. Виконується самостійна робота: розробляються консольні програми для вирішення завдань, що входять до складу індивідуального завдання.

Контрольні питання для допуску до роботи

1. Інтегроване середовище розробки PascalABC.NET.
2. Підтримка методології Rapid Application Development.
3. Текстовий редактор. Редагування коду.
4. Дизайнер форм. Розробка інтерфейсу.

5. Компіляція та запуск програми.

6. Інтегровані засоби трасування та налагодження програм.

7. Використання вбудованої довідки щодо розробки та мови програмування.

Порядок виконання роботи

Завдання 1. Обчислення значень виразів. Розробіть консольну програму для обчислення значень z_1 і z_2 за формулами, які визначаються у відповідності з наявним номером варіанта з табл. 9.1 (номер варіанта видається викладачем перед виконанням лабораторної роботи).

№ вар.	z_1	z_2
1.	$z_1 = 2\sin^2(3\pi - 2\alpha)\cos^2(5\pi + 2\alpha)$	$z_2 = \frac{1}{4} - \frac{1}{4}\sin\left(\frac{5}{2}\pi - 8\alpha\right)$
2.	$z_1 = \cos\alpha + \sin\alpha + \cos 3\alpha + \sin 3\alpha$	$z_2 = 2\sqrt{2}\cos\alpha \times \sin\left(\frac{\pi}{4} + 2\alpha\right)$
3.	$z_1 = \frac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha + 1 - 2\sin^2 2\alpha}$	$z_2 = 2\sin\alpha$
4.	$z_1 = \frac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha - \cos 3\alpha + \cos 5\alpha}$	$z_2 = \operatorname{tg} 3\alpha$
5.	$z_1 = 1 - \frac{1}{4}\sin^2 2\alpha + \cos 2\alpha$	$z_2 = \cos^2\alpha + \cos^4\alpha$
6.	$z_1 = \cos\alpha + \cos 2\alpha + \cos 6\alpha + \cos 7\alpha$	$z_2 = 4\cos\frac{\alpha}{2} \cdot \cos\frac{5}{2}\alpha \times \cos 4\alpha$
7.	$z_1 = \cos^2\left(\frac{3}{8}\pi - \frac{\alpha}{4}\right) - \cos^2\left(\frac{11}{8}\pi + \frac{\alpha}{4}\right)$	$z_2 = \frac{\sqrt{2}}{2}\sin\frac{\alpha}{2}$
19.	$z_1 = \frac{5 - 2a^2}{\frac{1 + a + a^2}{2a + a^2} + 2 - \frac{1 - a + a^2}{2a - a^2}}$	$z_2 = \frac{4 - a^2}{2}$
20.	$z_1 = \frac{(m-1)\sqrt{m} - (n-1)\sqrt{n}}{\sqrt{m^3 n} + nm + m^2 - m}$	$z_2 = \frac{\sqrt{m} - \sqrt{n}}{m}$

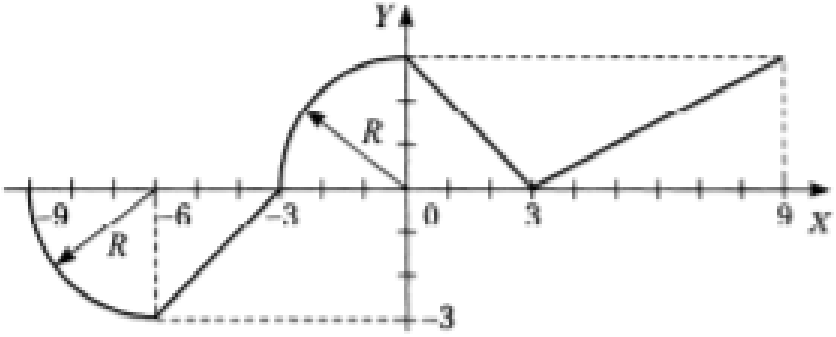
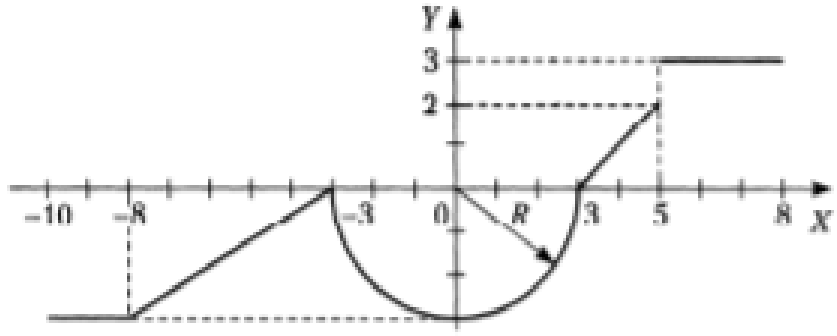
Таблиця 9.1

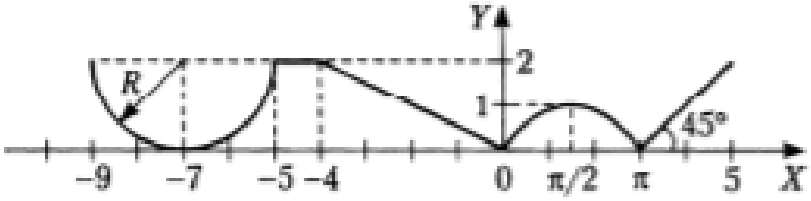
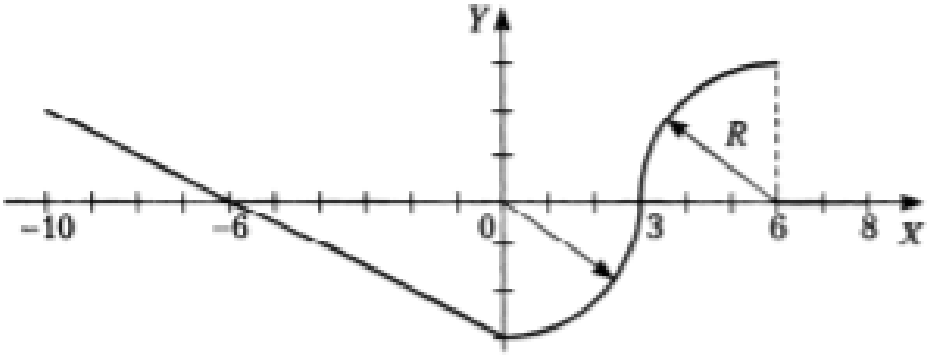
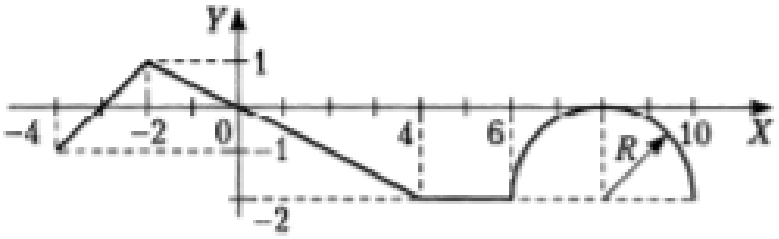
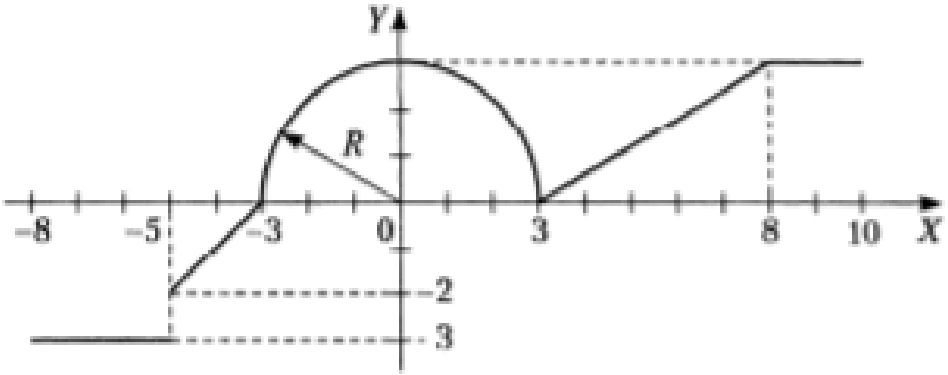
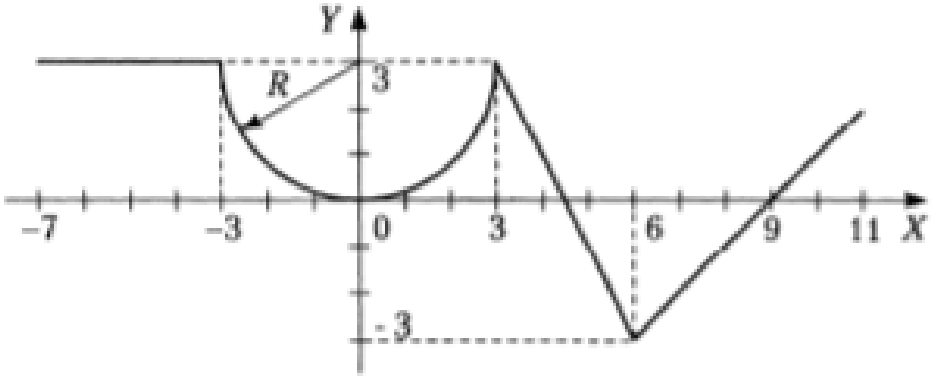
№ вар.	z_1	z_2
8.	$z_1 = \cos^4 x + \sin^2 y + \frac{1}{4}\sin^2 2x - 1$	$z_2 = \sin(y+x) \times \sin(y-x)$
9.	$z_1 = (\cos\alpha - \cos\beta)^2 - (\sin\alpha - \sin\beta)^2$	$z_2 = -4\sin^2\frac{\alpha-\beta}{2} \times \cos(\alpha+\beta)$
10.	$z_1 = \frac{\sin\left(\frac{\pi}{2}+3\alpha\right)}{1-\sin(3\alpha-\pi)}$	$z_2 = \operatorname{ctg}\left(\frac{5}{4}\pi + \frac{3}{2}\alpha\right)$
11.	$z_1 = \frac{1-2\sin^2\alpha}{1+\sin 2\alpha}$	$z_2 = \frac{1-\operatorname{tg}\alpha}{1+\operatorname{tg}\alpha}$
12.	$z_1 = \frac{\sin 4\alpha}{1+\cos 4\alpha} \cdot \frac{\cos 2\alpha}{1+\cos 2\alpha}$	$z_2 = \operatorname{ctg}\left(\frac{3}{2}\pi - \alpha\right)$
13.	$z_1 = \frac{\sin\alpha + \cos(2\beta-\alpha)}{\cos\alpha - \sin(2\beta-\alpha)}$	$z_2 = \frac{1+\sin 2\beta}{\cos 2\beta}$
14.	$z_1 = \frac{\cos\alpha + \sin\alpha}{\cos\alpha - \sin\alpha}$	$z_2 = \operatorname{tg} 2\alpha + \sec 2\alpha$
15.	$z_1 = \frac{\sqrt{2b + 2\sqrt{b^2-4}}}{\sqrt{b^2-4}+b+2}$	$z_2 = \frac{1}{\sqrt{b+2}}$
16.	$z_1 = \frac{x^2+2x-3+(x+1)\sqrt{x^2-9}}{x^2-2x-3+(x-1)\sqrt{x^2-9}}$	$z_2 = \sqrt{\frac{x+3}{x-3}}$
17.	$z_1 = \frac{\sqrt{(3m+2)^2-24m}}{3\sqrt{m}-\frac{2}{\sqrt{m}}}$	$z_2 = -\sqrt{m}$
18.	$z_1 = \left(\frac{a+2}{\sqrt{2a}} - \frac{a}{\sqrt{2a}+2} + \frac{2}{a-\sqrt{2a}}\right) \times \frac{\sqrt{a}-\sqrt{2}}{a+2}$	$z_2 = \frac{1}{\sqrt{a}+\sqrt{2}}$

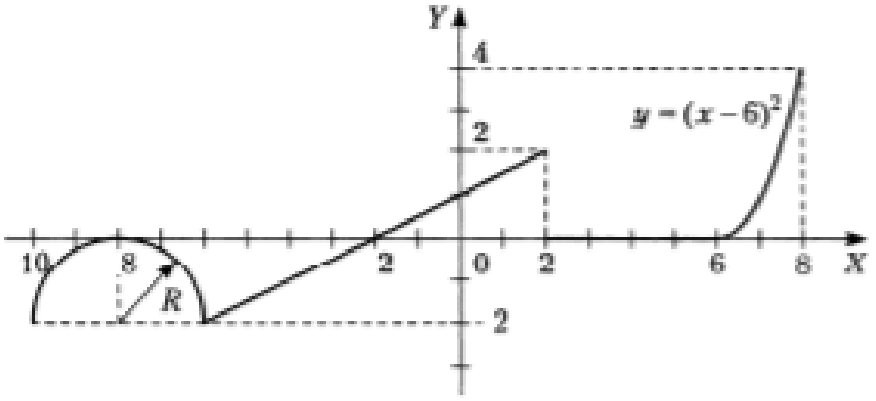
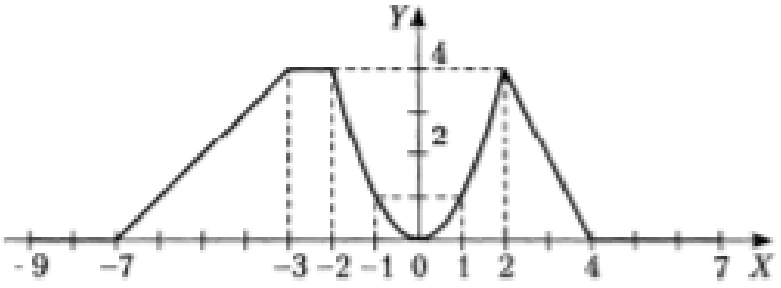
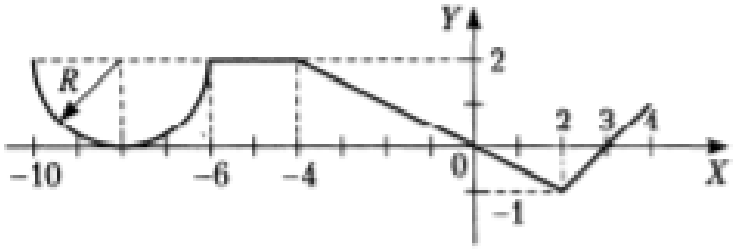
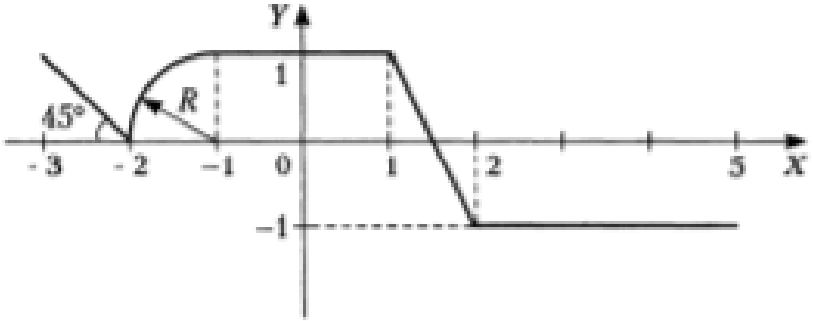
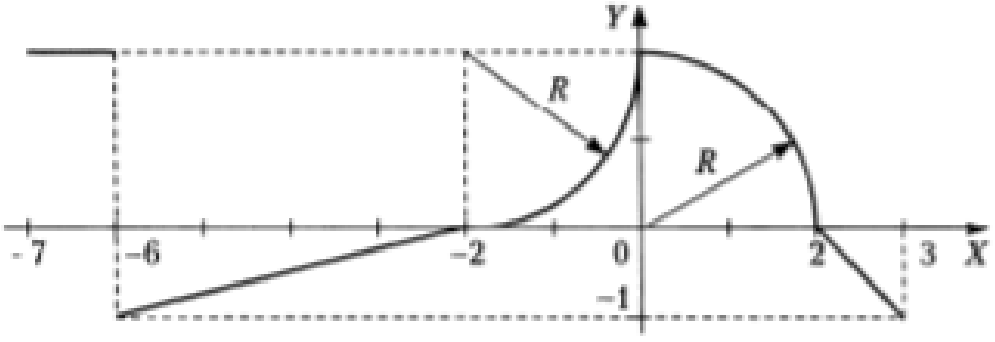
Завдання 2. Обрахунок значень функції. Розробіть консольний додаток для обрахунку значень функції y=f(x) в залежності від одного заданого користувачем значення аргументу x. Інші додаткові дані, необхідні для обрахунку значення функції, також вводяться користувачем.

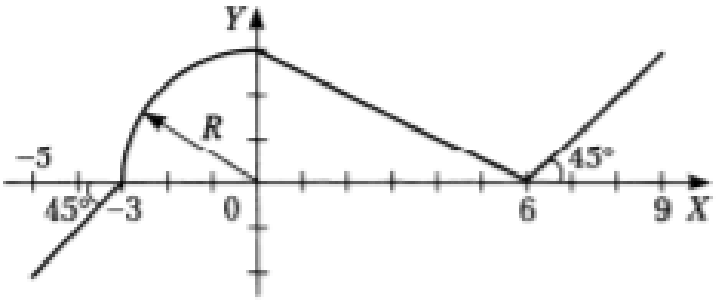
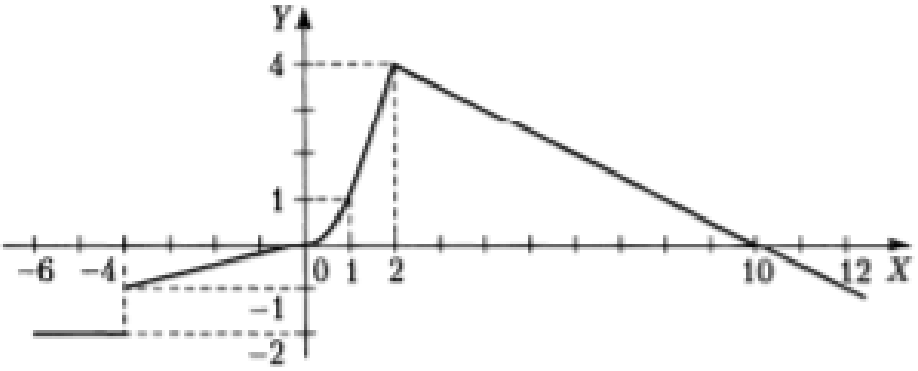
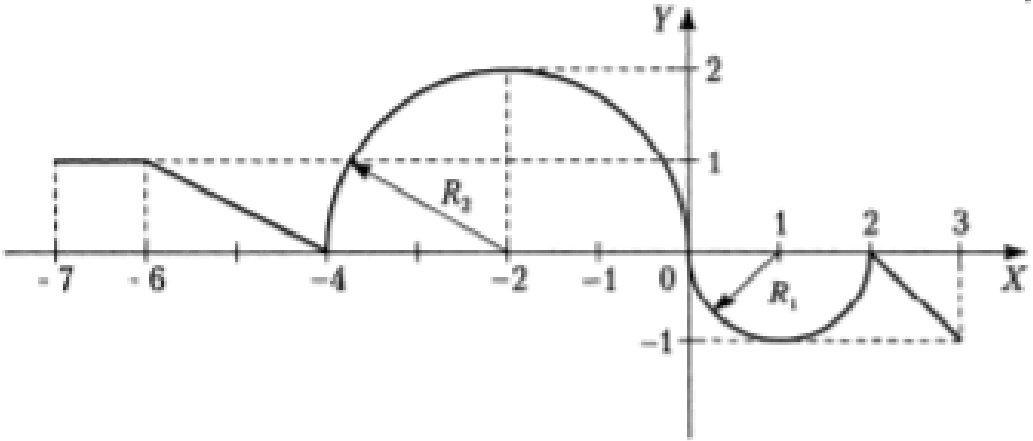
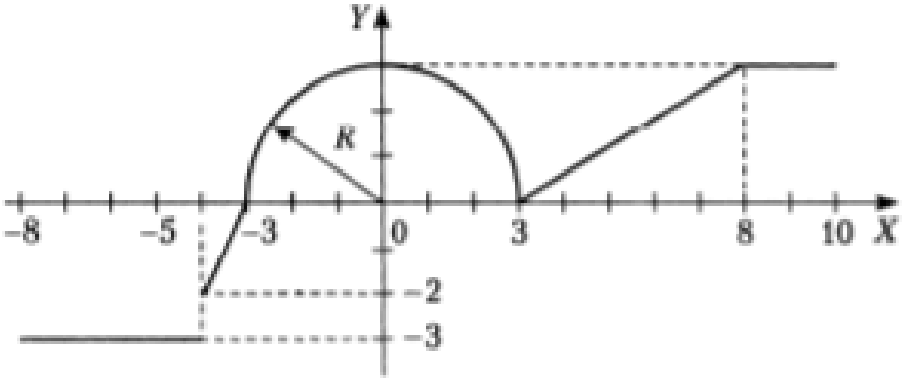
Функція y=f(x) задана графічним способом. Графік функції визначається відповідно до номеру варіанта з табл. 9.2 (номер варіанта видається викладачем перед виконанням лабораторної роботи).

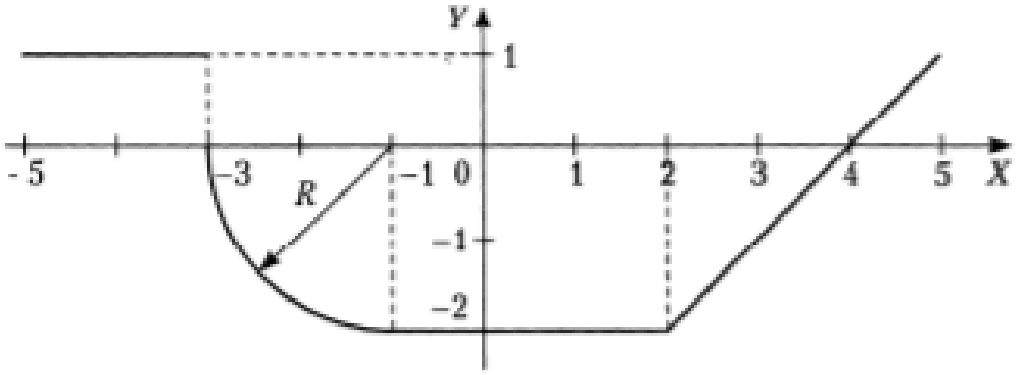
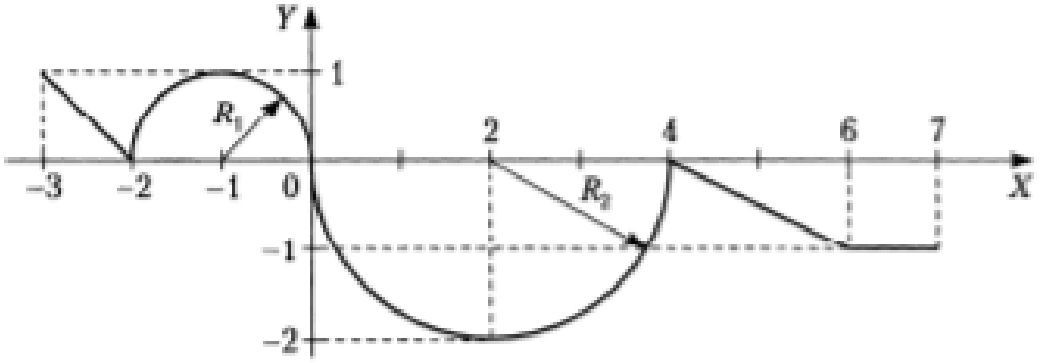
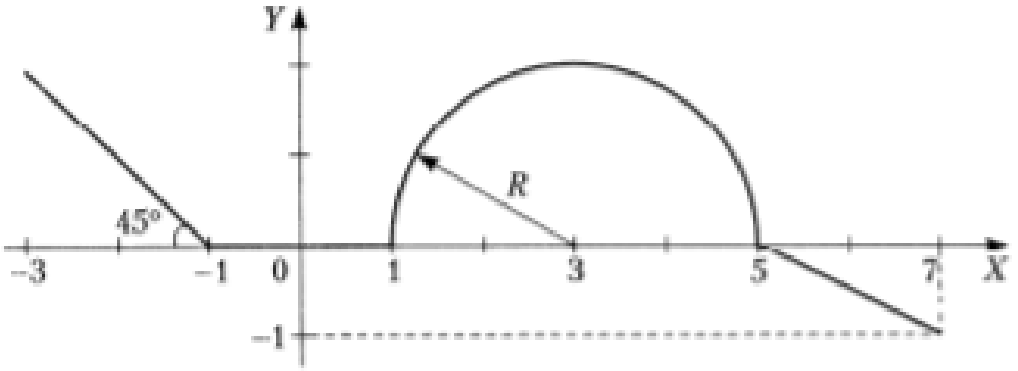
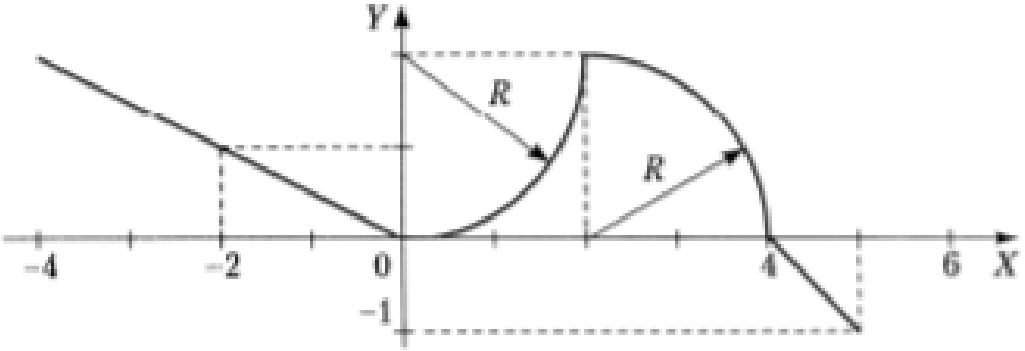
Таблиця 9.2

№ вар.	*Графік функції f=y(x)*
1.	
2.	

№ вар.	*Графік функції f=y(x)*
3.	
4.	
5.	
6.	
7.	

№ вар.	*Графік функції f=y(x)*
8.	
9.	
10.	
11.	
12.	

№ вар.	*Графік функції f=y(x)*
13.	
14.	
15.	
16.	

№ вар.	*Графік функції f=y(x)*
17.	
18.	
19.	
20.	

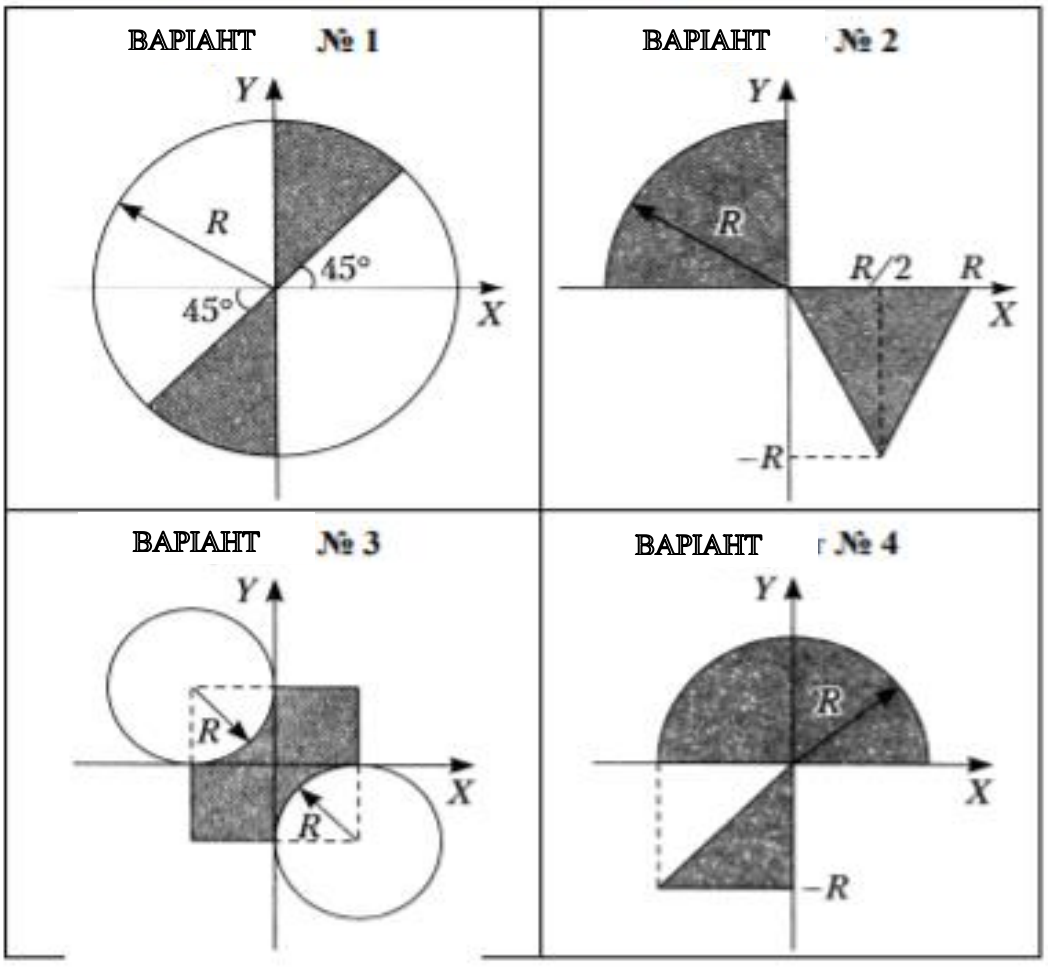
Завдання 3. Перевірка належності точки заданої області на площині.

Розробіть консольну програму, яка дозволяє перевірити, чи належить точка із заданими координатами x та y деякій обмеженій області на площині.

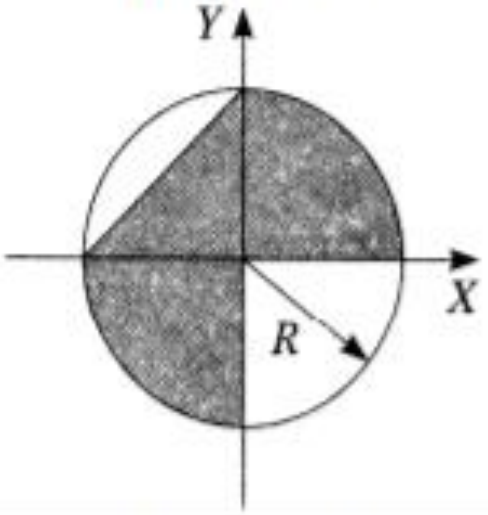
Область задана графічним способом. Координати точки та додаткові дані, що визначають геометрію області та необхідні для вирішення задачі, вводяться користувачем.

Схема, на якій задана область виділена штрихуванням, визначається відповідно до наявного номеру варіанта з табл. 9.3.

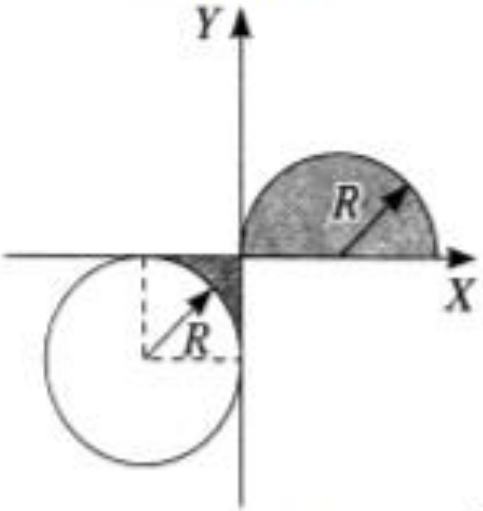
Таблиця 9.3



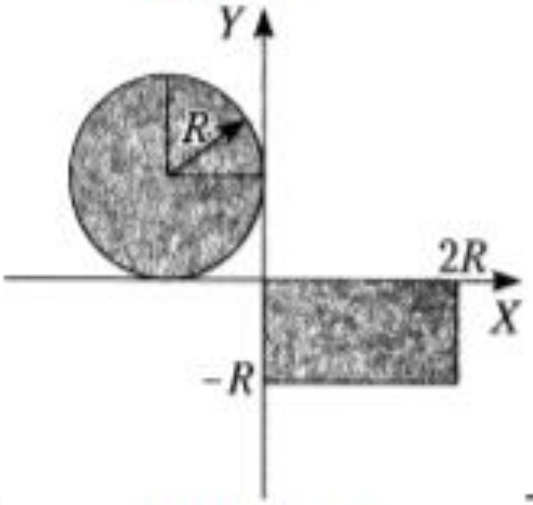
ВАРІАНТ № 5



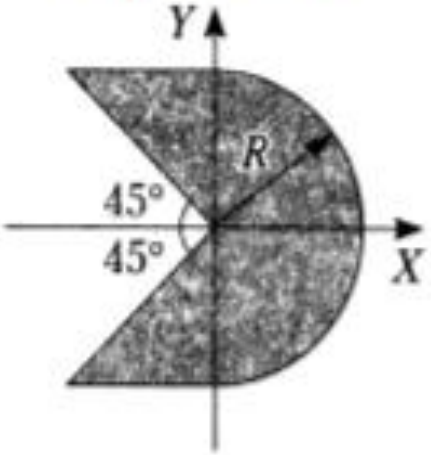
ВАРІАНТ № 6



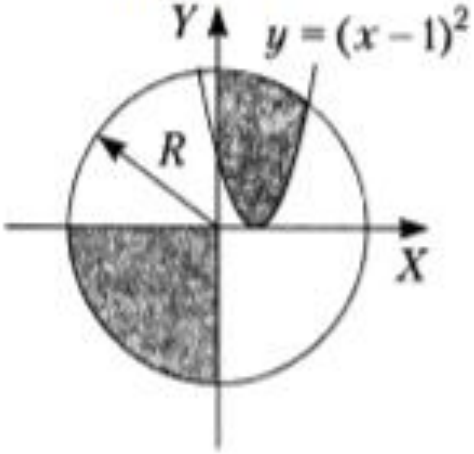
ВАРІАНТ № 7



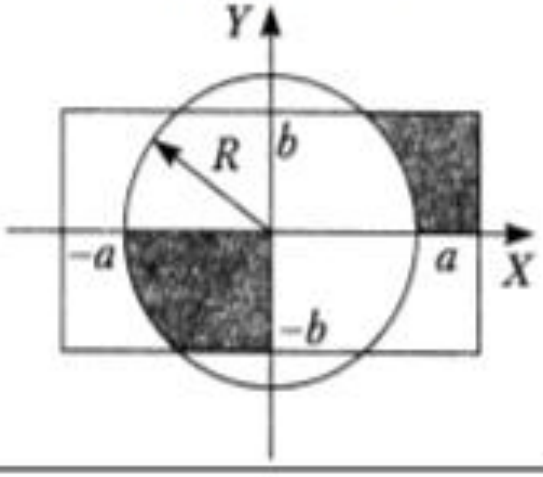
ВАРІАНТ № 8



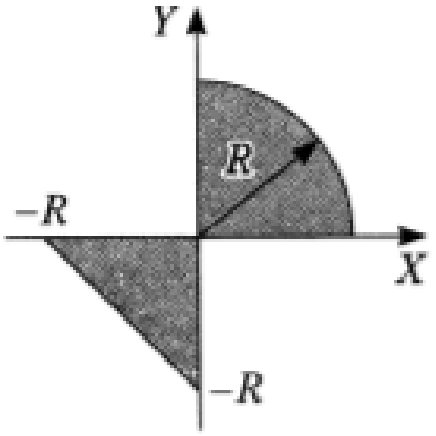
ВАРІАНТ № 9



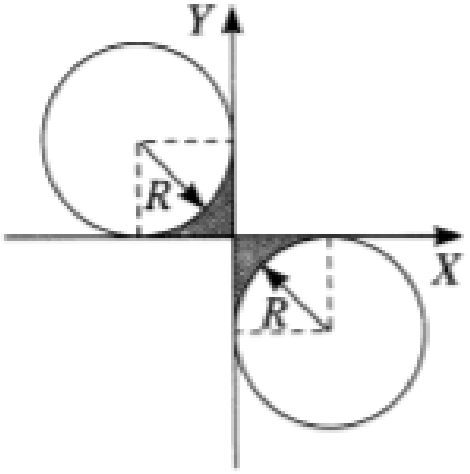
ВАРІАНТ № 10



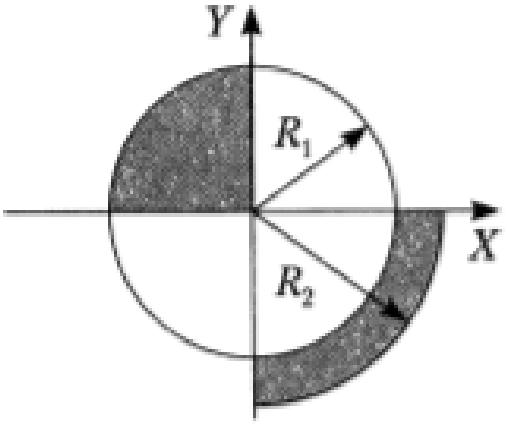
ВАРІАНТ № 11



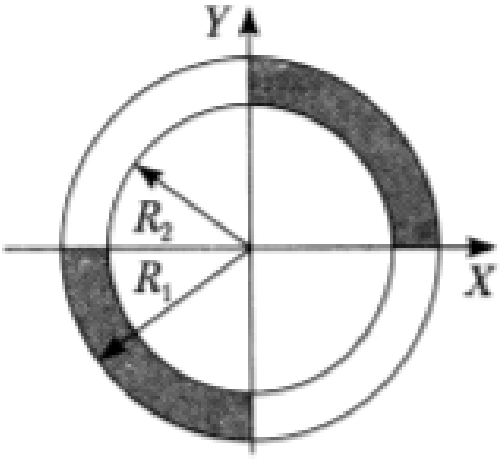
ВАРІАНТ № 12



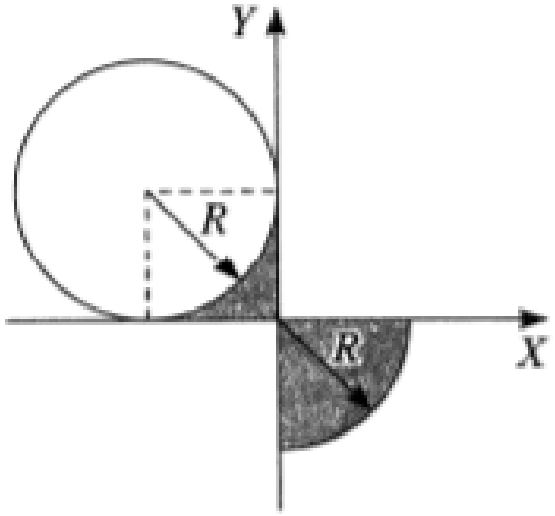
ВАРІАНТ № 13



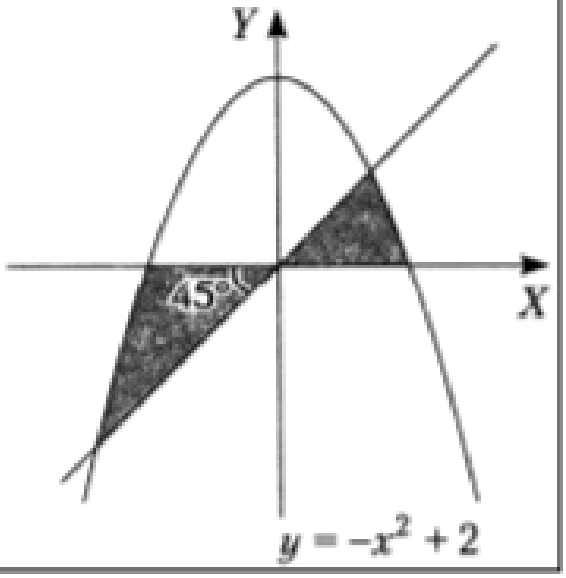
ВАРІАНТ № 14

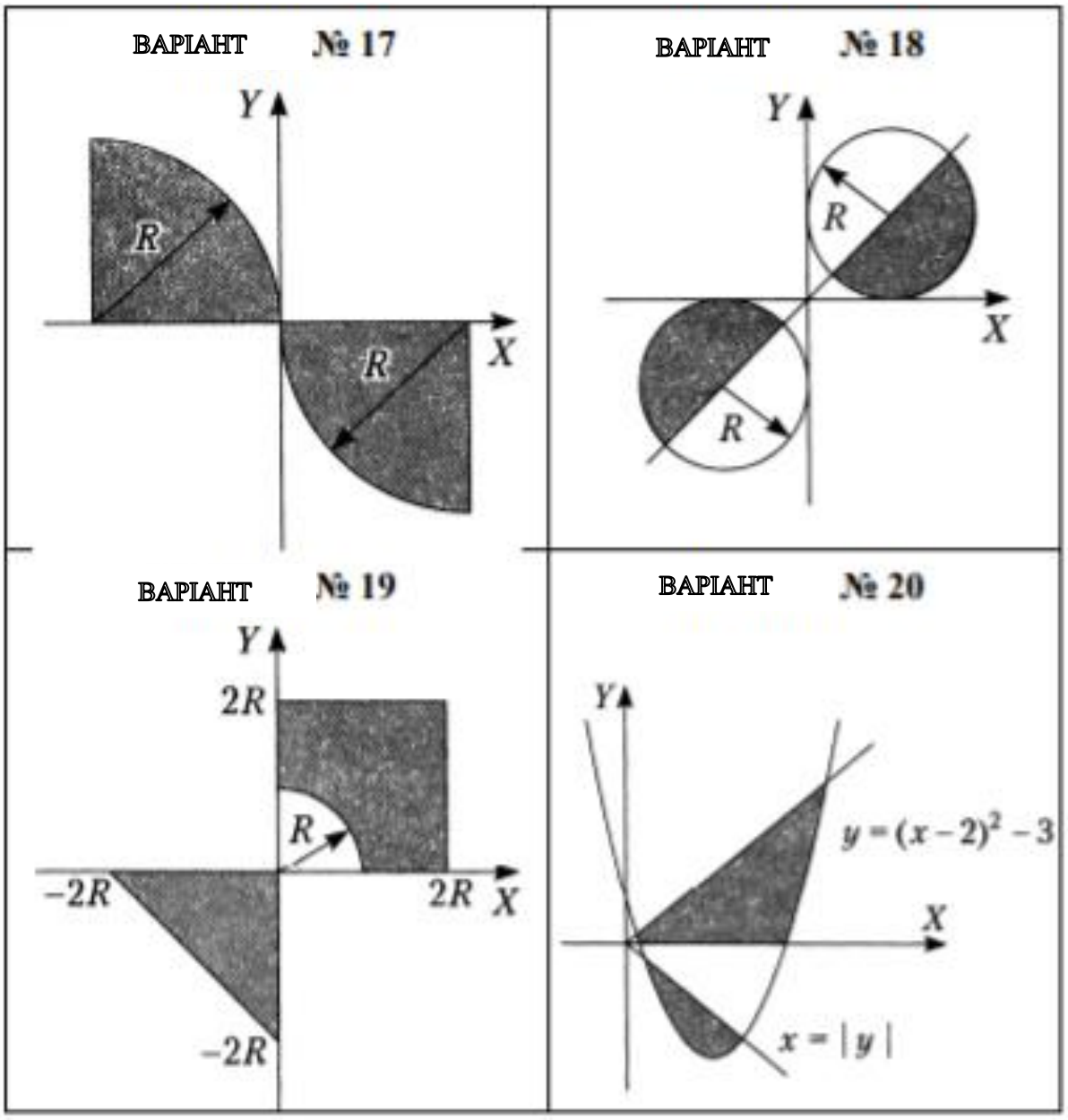


ВАРІАНТ № 15



ВАРІАНТ № 16





Завдання 4. Виведення таблиці значень функції.

Розробіть консольну програму для виведення таблиці значень функції y=f(x) в точках, отриманих поділом відрізка [a,b] на N рівних частин. Усі дані, необхідні для обчислення значень функції та виведення таблиці, вводяться користувачем.

Функція y=f(x) задана графічним способом. Графік функції визначається відповідно до наявного номера варіанта з табл. 9.2.

Завдання виведення таблиці має бути вирішене трьома

способами: за допомогою циклу з лічильником, циклу з передумовою та циклу з післяумовою. Розв'язання задачі всіма трьома способами має бути виконане в одному додатку і для тих самих вихідних даних.

Завдання 5. Серія «пострілів по мішені».

Розробіть консольну програму, яка імітуватиме виконання серії «пострілів по мішені». Кількість пострілів та інші необхідні дані задаються користувачем. Для кожного «пострілу» необхідно виводити повідомлення про влучення або промах.

«Мішень» є область на площині, що має певні геометричні розміри. «Постріл» імітується вибором деякої точки на площині. Для кожного пострілу відповідна точка на площині визначається випадковим чином (у межах прямокутника або кола, що обмежують область навколо мішені). «Влучення в ціль» зараховується, якщо точка потрапляння належить заданій області мішені.

Область задана графічним способом. Схема, на якій задана область виділена штрихуванням, визначається відповідно до наявного номеру варіанта з табл. 9.3.

Завдання 6. Обчислення суми ряду із заданою точністю.

Розробіть програму для обчислення суми ряду Тейлора із заданою точністю ε. Усі дані, необхідні для обчислення суми ряду, вводяться користувачем.

Формула для обчислення суми ряду Тейлора визначається відповідно до наявного номеру варіанта з табл. 9.4 (у лівій частині

формули вказано функцію, якій відповідає ряд Тейлора, після формули наводиться область допустимих значень аргументу x).

Таблиця 9.4

№ вар.	Формула та область допустимих значень x
1.	$\ln\frac{x+1}{x-1}=2\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=2\left(\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\ldots\right),\ \lvert x\rvert>1$
2.	$e^{-x}=\sum_{n=0}^{\infty}\frac{(-1)^n x^n}{n!}=1-x+\frac{x^2}{2!}-\frac{x^3}{3!}+\frac{x^4}{4!}-\ldots,\ \lvert x\rvert<\infty$
3.	$e^{x}=\sum_{n=0}^{\infty}\frac{x^n}{n!}=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}+\ldots,\ \lvert x\rvert<\infty$
4.	$\ln(x+1)=\sum_{n=0}^{\infty}\frac{(-1)^n x^{n+1}}{n+1}=x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}-\ldots,\ \lvert x\rvert<1$
5.	$\ln\frac{1+x}{1-x}=2\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1}=2\left(x+\frac{x^3}{3}+\frac{x^5}{5}+\ldots\right),\ \lvert x\rvert<1$
6.	$\ln(1-x)=-\sum_{n=1}^{\infty}\frac{x^n}{n}=-\left(x+\frac{x^2}{2}+\frac{x^3}{3}+\ldots\right),\ \lvert x\rvert<1$
7.	$\operatorname{arcctg}x=\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}x^{2n+1}}{2n+1}=\frac{\pi}{2}-x+\frac{x^3}{3}-\frac{x^5}{5}+\ldots,\ \lvert x\rvert<1$
8.	$\operatorname{acrtg}x=\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}+\ldots,\ x>1$
9.	$\operatorname{arctg}x=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{2n+1}=x-\frac{x^3}{3}+\frac{x^5}{5}-\frac{x^7}{7}+\ldots,\ \lvert x\rvert<1$
10.	$\operatorname{arth}x=\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1}=x+\frac{x^3}{3}+\frac{x^5}{5}+\frac{x^7}{7}+\ldots,\ \lvert x\rvert<1$
11.	$\operatorname{arth}x=\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\ldots,\ \lvert x\rvert>1$
12.	$\operatorname{arctg}x=-\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=$ $=-\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}+\ldots,\ x<-1$

№ вар.	Формула та область допустимих значень x
13.	$e^{-x^2}=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{n!}=1-x^2+\frac{x^4}{2!}-\frac{x^6}{3!}+\frac{x^8}{4!}-\ldots,\ \lvert x\rvert<\infty$
14.	$\cos x=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n)!}=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\ldots,\ \lvert x\rvert<\infty$
15.	$\frac{\sin x}{x}=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n+1)!}=1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}+\ldots,\ \lvert x\rvert<\infty$
16.	$\ln x=2\sum_{n=0}^{\infty}\frac{(x-1)^{2n+1}}{(2n+1)(n+1)^{2n+1}}=$ $=2\left(\frac{x-1}{x+1}+\frac{(x-1)^3}{3(x+1)^3}+\frac{(x-1)^5}{5(x+1)^5}+\ldots\right),\ x>0$
17.	$\ln x=\sum_{n=0}^{\infty}\frac{(x-1)^{n+1}}{(n+1)x^{n+1}}=\frac{x-1}{x}+\frac{(x-1)^2}{2x^2}+\frac{(x-1)^3}{3x^3}+\ldots,\ x>\frac{1}{2}$
18.	$\ln x=\sum_{n=0}^{\infty}\frac{(-1)^2(x-1)^{n+1}}{(n+1)}=$ $=(x-1)-\frac{(x-1)^2}{2}+\frac{(x-1)^3}{3}-\ldots,\ 0<x<2$
19.	$\arcsin x=x+\sum_{n=1}^{\infty}\frac{1\cdot3\cdot\ldots\cdot(2n-1)\cdot x^{2n+1}}{2\cdot4\cdot\ldots\cdot2n\cdot(2n+1)}=$ $=x+\frac{x^3}{2\cdot3}+\frac{1\cdot3\cdot x^5}{2\cdot4\cdot5}+\frac{1\cdot3\cdot5\cdot x^7}{2\cdot4\cdot6\cdot7}+\frac{1\cdot3\cdot5\cdot7\cdot x^9}{2\cdot4\cdot6\cdot8\cdot9}+\ldots,\ \lvert x\rvert<1$
20.	$\arccos x=\frac{\pi}{2}-x+\sum_{n=1}^{\infty}\frac{1\cdot3\cdot\ldots\cdot(2n-1)\cdot x^{2n+1}}{2\cdot4\cdot\ldots\cdot2n\cdot(2n+1)}=$ $=\frac{\pi}{2}-x+\frac{x^3}{2\cdot3}+\frac{1\cdot3\cdot x^5}{2\cdot4\cdot5}+\frac{1\cdot3\cdot5\cdot x^7}{2\cdot4\cdot6\cdot7}+\frac{1\cdot3\cdot5\cdot7\cdot x^9}{2\cdot4\cdot6\cdot8\cdot9}+\ldots,\ \lvert x\rvert<1$

Завдання 7. Виведення таблиці значень функції, що обчислюється за допомогою ряду Тейлора.

Розробіть консольну програму для виведення таблиці значень функції $y=f(x)$ у точках, отриманих розподілом відрізка $[a,b]$ на N рівних частин. Функція $y=f(x)$ обчислюється як кінцева сума відповідного ряду Тейлора із заданою користувачем точністю ε. Усі дані**,** необхідні обчислення значень функції та виведення таблиці, вводяться користувачем.

Формула для знаходження значення функції визначається відповідно до наявного номеру варіанта з табл. 9.4.

Лабораторна робота №10

СТВОРЕННЯ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ CORELDRAW

Цілі та завдання лабораторної роботи

Сормувати вміння та навички побудови малюнків і просторових зображень фігур засобами CorelDraw, а також роботи з текстом у редакторі.

У процесі виконання лабораторної роботи вирішуються такі завдання:

1. Виконати в CorelDraw зображення фрагменту графіку функції.

2. Відповідно до варіанта виконати зображення просторової фігури із зазначенням усіх основних її елементів (радіус, похила, твірна, висота тощо).

№	Фігура
1	Циліндр
2	Похилий циліндр
3	Конус
4	Похилий конус
5	Зрізаний конус
6	Зрізаний похилий конус
7	Похилена п’ятикутна піраміда
8	Зрізана шестикутна піраміда
9	Правильна шестикутна піраміда
10	Куля
11	Зрізана похила чотирикутна піраміда
12	Похила шестикутна призма
13	Правильна п’ятикутна призма
14	Похила семикутна призма

3. У редакторі Word набрати формули площі загальної поверхні та об'єму побудованої фігури та вставити їх у документ CorelDraw.

4. Створити та розташувати внизу аркуша звичайний текст, який міститиме означення побудованої фігури.

5. Зберегти створений документ у файл Lab10, помістивши його в папку Corel.

6. Здійснити експорт створеного зображення (без формул та звичайного тексту) у формат EPS (завдання самостійної роботи).

Контрольні питання

1. Растрові та векторні зображення.
2. Моделі представлення кольорів.
3. Формати графічних файлів.
4. Елементи робочого вікна та панелі інструментів редактора CorelDraw.
5. Графічні примітиви та робота з ними.
6. Побудова кривих Без'є.
7. Особливості роботи з фігурним текстом.
8. Особливості роботи з звичайним текстом.

КОНТРОЛЬНІ ПИТАННЯ

1. Обчислювальна техніка. Комп'ютери.
2. Ознаки класифікації обчислювальної техніки.
3. Принципи впливу ЕОМ.
4. Покоління ЕОМ. Елементна база ЕОМ.
5. Види забезпечення ЕОМ.
6. Програмне та апаратне забезпечення ЕОМ.
7. Інформаційне забезпечення ЕОМ.
8. Математичне забезпечення ЕОМ.
9. Архітектура комп'ютера. Класифікаційні ознаки та характеристики архітектури комп'ютера.
10. Архітектура фон Неймана.
11. Гарвардська архітектура. Її переваги та недоліки.
12. Магістрально-модульний принцип роботи ЕОМ.
13. Види пристроїв, що підключаються до системної шини.
14. Процесор. Арифметико-логічний устрій. Пристрій керування.
15. Системна шина. Пристрої введення. Пристрої виведення.
16. Персональний комп'ютер. Характеристики ПК.
17. Основні та периферійні пристрої ПК.
18. Пристрої обміну. Пристрої обробки інформації.
19. Носії інформації. Жорсткий диск. Розділи жорсткого диска
20. Монітор. Клавіатура. Миша.
21. Материнська плата. Процесор. Оперативна пам'ять.
22. Відеокарта.
23. Програмні засоби ЕОМ.

24. Програма. Програмний продукт. Пакет програм. Програмне забезпечення.

25. Версії ПЗ. Оновлення ПЗ.

26. Програма-сервер. Програма-клієнт.

27. Види ПЗ. Системне ПЗ. Прикладне програмне забезпечення.

28. Інструментарій технології програмування. Інтегровані середовища розробки ПЗ.

29. Операційна система. Класифікація ОС.

30. Мережеві ОС. Розраховані на багато користувачів ОС. Багатозадачні ОС.

31. Операційні системи пакетної обробки. Системи розподілу часу. Системи реального часу.

32. Перевірка та дефрагментація жорсткого диска.

33. Прикладне ПЗ. Види прикладних програм.

34. Програми обробки тексту.

35. Програми обробки таблиць та масивів даних.

36. Програми обробки графічних зображень.

37. Програми обробки аудіо-відео сигналів.

38. Програми обробки чисел.

39. Програми обробки знань.

40. Комунікаційні програми.

41. Програми автоматизації робіт.

42. Програмування. Мови програмування.

43. Алгоритмічні мови програмування.

44. Мови функціонального програмування.

45. Мови логічного програмування.

46. Мова Паскаль.
47. Мова C#.
48. Мова програмування Pascal ABC
49. Інтегроване середовище розробки Pascal ABC.NET.
50. Платформа Microsoft.

РЕКОМЕНДОВАНИЙ СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Проект Закону України про використання відкритих форматів та вільного програмного забезпечення в державних установах та державному секторі господарства // http://www.rada.gov.ua.

2. Гомонай-Стрижко М.В. Інформаційні системи та технології на підприємстві:. Конспект лекцій. – Львів: НЛТУ, 2014. – 200 с. [Електрон. ресурс]. / Гомонай-Стрижко М.В., Якімцов В.В. – http://ep.nltu.edu.ua/images/Kafedra_EP/Kafedra_EP_PDFs/kl_isitp.pdf.

3. Баженов В.А., Венгерський П.С., Горлач В.М., Левченко О.М., Лізунов П.П., Гарвона В.С., Ананьєв О.М., Інформатика. Комп'ютерна техніка. Комп'ютерні технології., підручник для студентів вищих закладів освіти. – Київ, "Каравела", 2003.

4. Грицунов О. В. Інформаційні системи та технології: навч. посіб. для студентів за напрямом підготовки «Транспортні технології»/ О. В. Грицунов; Харк. нац. акад. міськ. госп-ва. – Х.: ХНАМГ, 2010. – 222 с.

5. Інформаційні системи в менеджменті: Навчальний посібник//Батюк А.Є., Двуліт З.П., Обельовська К.М., Огородник І.М. та ін. – К.: ІнтелектЗахід, 2004. – 520с.

6. 4. Інформатика: комп'ютерна техніка. Комп'ютерні технології : підручник для студентів вищих навчальних закладів / за ред. О. І. Пушкаря. – К. : Видавничий центр «Академія», 2002. – 704 с.

7. Інформаційне забезпечення менеджменту//Новак В.О., Макаренко Л. Г., Луцький И. Г. – К.: Кондор, 2006. – 462 с

8. Жуванов Д., Стогній Є. Яку форму правової охорони обрати для комп'ютерної програми? // Інтелектуальна власність. - 2003. - № 9; http://www.inventa-ua.com.

9. Закон України про авторське право і суміжні права // Відом. Верховної Ради України. - 2004. - № 13. - С. 181.

10. Кобиляцький, Л.С. Управління проектами: Навч. посібник. / Л.С. Кобиляцький. – К.: МАУП, 2002. – 200с.

11. Лозинський А.О., Мороз В.І., Паранчук Я.С. Розв'язання задач електромеханіки в середовищах пакетів MathCAD і MATLAB: Навчальний посібник. — Львів: вид-во держ. ун-ту «Львівська політехніка», 2000. — 166 с.

12. Молодцова О. П. Прикладне програмне забезпечення: Навч.-метод. посібник для самост. вивч. дисц. — К.: КНЕУ, 2000. — 264 с.

13. Нелюбов В. О., Куруца О. С. Основи інформатики. Microsoft Excel 2016: навчальний посібник. Ужгород: ДВНЗ «УжНУ», 2018. - 58 с.: іл.: [Електронний ресурс]. – Режим доступу: https://www.uzhnu.edu.ua/uk/infocentre/get/15617.

14. Нелюбов В.О., Куруца О.С. Основи інформатики. Microsoft PowerPoint 2016: навчальний посібник. Ужгород: ДВНЗ «УжНУ», 2018. – 122 с.: іл.: [Електронний ресурс]. – Режим доступу: https://www. uzhnu.edu.ua/uk/infocentre/get/15627.

15. Нелюбов В.О., Куруца О.С. Основи інформатики. Microsoft Word 2016: електронний навчальний посібник. Ужгород: ДВНЗ

УжНУ, 2018. – 96 с.: іл.: [Електронний ресурс]. – Режим доступу: https://www.uzhnu.edu.ua/uk/infocentre/get/16001.

16. Огляд технологій та сервісів Веб 2.0. Веб-спільноти. Вікітехнології.: [Електронний ресурс]. – Режим доступу: http://www.ndu.edu.ua/liceum/html/web20.pdf.

17. Організація комп'ютерних мереж [Електронний ресурс]: підручник: для студ. спеціальності 121 «Інженерія програмного забезпечення» та 122 «Комп'ютерні науки» / КПІ ім. Ігоря Сікорського; Ю. А. Тарнавський, І. М. Кузьменко. – Електронні текстові дані (1 файл: 45,7 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2018. – 259 с.: [Електронний ресурс]. – Режим доступу: http://ela.kpi.ua/bitstream/123456789/25156/1/Tarnavsky_Kuzmenko_Org_Komp_merej.pdf.

18. Пасічник В. В. Організація баз даних та знань: підруч. для студ. вищ. навч. закл. за напрямами «Комп'ютерні науки» / В. В. Пасічник, В. А. Резніченко; за заг. ред. М. З. Згуровського – К.: Видавнича група BHV, 2006. − 384 с.

19. Пора переходити на програмне забезпечення з відкритим кодом // http://www.observer.sd.org.ua.

20. Руденко В. Д. Інформатика (профільний рівень) / В. Д. Руденко, Н.В. Речич, В. О. Потієнко. – Харків : Вид-во «Ранок», 2019. – 256 с. : іл.

21. Руденко В. Д. Інформатика (профільний рівень): / В. Д. Руденко, Н. В. Речич, В. О. Потієнко. – Харків: Вид-во «Ранок», 2019. – 256 с.: [Електронний ресурс]. – Режим доступу: http://interactive.ranok.com.ua/upload/file/2019/Informprof_11kl.pdf.

22. Трофименко О.Г., Буката Л.М. СУБД ACCESS СТВОРЕННЯ ТА ОПРАЦЮВАННЯ БАЗ ДАНИХ. Методичні вказівки до лабораторних, практичних занять та самостійної роботи студентів напряму. – Одеса, 2016: [Електронний ресурс]. – Режим доступу: http://docplayer.net/ 83488378-Ministerstvo-osviti-i-nauki-ukrayini-odeska-nacionalna- akademiya-zv-yazku-im-o-s-popova-kafedra-informaciynih-tehnologiysubd-access.html.

23. Шпетний І. О., Проценко С. І., Тищенко К. В. Інформатика. Навчальний посібник. – Суми, 2018.: [Електронний ресурс]. – Режим доступу: https://essuir.sumdu. edu.ua/bitstream/123456789/67760/3/ Shpetnyi_informatyka.pdf.

24. Юринець В.Є., Юринець Р.В. Інформаційні системи управління персоналом, діловодства і документообігу: Навч.посіб. – Львів: «Тріада плюс», 2008.

25. Lisa R. Wolfisch, Rachael LaPorte Taylor. Open Source at the Census Bureau and FedStats // Proc. Of Conf. “Open Source: A Case for e-government”, Washington, D.C., 2002. - Oct. 16-18.