В.П. Лисенко, І.М. Болбот

КОМП'ЮТЕРИ ТА КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

Частина І

«Програмування в математичному пакеті MathCAD»

Навчальний посібник

Рекомендовано Міністерством освіти і науки України як навчальний посібник для студентів вищих навчальних закладів, які навчаються за напрямом підготовки "Енергетика та електротехнічні системи в агропромисловому комплексі"

Київ
Аграрна освіта
2010

УДК 004.4(075.8)
ББК 32973я73
Л 88

Гриф надано Міністерством освіти і науки України (лист №1/11-4167 від 18.05.2010 р.)

Р е ц е н з е н т и:

А.П. Ладанюк – доктор технічних наук, професор, заслужений діяч науки і техніки України, завідувач кафедри автоматизації та комп'ютерно-інтегрованих технологій Національного університету харчових технологій;

В.С. Федорейко – доктор технічних наук, професор, завідувач кафедри машинобудування та комп'ютерної інженерії Тернопільського національного педагогічного університету ім. Володимира Гнатюка.

Лисенко В.П.
Л 88 Комп'ютери та комп'ютерні технології : навч. посіб. Ч. 1. Програмування в математичному пакеті MathCAD / В.П. Лисенко. І.М. Болбот. – К. : Аграрна освіта, 2010. – 229 с.
ISBN 978-966-7906-90-0

Навчальний посібник призначений для студентів за напрямом підготовки: «Енергетика та електротехнічні системи в агропромисловому комплексі» факультету енергетики і автоматики, що вивчають курс «Комп'ютери та комп'ютерні технології», а також корисний аспірантам та інженерам, що використовують у своїх розрахунках математичний пакет MathCAD.

УДК 004.4(075.8)
ББК 32973я73

ISBN 978-966-7906-90-0

ЗМІСТ

ВСТУП

Мільйони людей займаються математичними розрахунками за професійної або іншої необхідності, не говорячи вже про навчання. Жодна серйозна розробка в будь-якій галузі науки та виробництва не уникає громіздких математичних розрахунків. Для їх проведення застосовують складні програми з використанням конструкцій мов високого рівня. Проте розробка таких програм, що мають сучасний графічний інтерфейс, вимагає відповідної підготовки в практиці програмування та досить тривалого часу (і те, й інше може бути відсутнім в інженера або дослідника).

У навчальному посібнику розглянуто питання алгоритмізації; використання математичного пакета MathCAD в інженерних розрахунках, які пов’язані з енергетикою та електротехнічними системами в агропромисловому комплексі; розглянуто основні конструкції вбудованої мови програмування в математичному пакеті MathCAD та їх застосування для програмування основних типів обчислювальних алгоритмів (лінійних, що розгалужуються та циклів).

Виклад теоретичного матеріалу супроводжується розглядом прикладів та завдань, пов’язаних з енергетикою та електротехнічними системами, що сприяє кращому засвоєнню теоретичного матеріалу дисципліни, який безпосередньо пов’язаний з професійною діяльністю майбутнього інженера.

Широкої популярності ще в середині 80-х років XX ст. набули інтегровані системи для автоматизації математичних розрахунків класу MathCAD, розроблені фірмою MathSoft. MathCAD один з небагатьох математичних пакетів, у яких опис рішення математичних задач дається

за допомогою звичних математичних формул і знаків. Такий самий вид мають і результати обчислень.

В останніх версіях MathCAD користувачу надана можливість створювати «власні» програми-функції та використовувати принципи модульного програмування для реалізації власних оригінальних обчислювальних алгоритмів. Однак у літературі ці нові можливості освітлено досить слабко. Отже у цьому навчальному посібнику викладають способи програмування різних алгоритмів з використанням конструкцій пакета MathCAD.

РОЗДІЛ 1
АРХІТЕКТУРА КОМП'ЮТЕРА

- ❖ Принципи архітектури комп’ютера
- ❖ Комп'ютери та комп’ютерне обладнання
- ❖ Периферійні пристрої
- ❖ Зовнішні носії інформації
- ❖ Операційні системи
- ❖ Програмне забезпечення комп’ютерів

1.1. Принципи архітектури комп’ютера

Поняття архітектури обчислювальних систем є одним з основних в інформатиці. Уперше термін «архітектура комп'ютера» було введено фірмою IBM під час розробки обчислювальних систем серії IBM 360 і застосовано до тих засобів, які може використовувати програміст під час написання програм на рівні машинних команд [10].

Під цим терміном розуміли структуру комп'ютера з погляду програміста, так звану логічну архітектуру, що є поняттям архітектури у вузькому розумінні. Воно охоплює формати команд, форми зображення даних, методи адресації й управління операціями введення-виведення. При цьому з розгляду випадають такі аспекти, як фізична організація пристроїв комп'ютера, ієрархія пам'яті, методи паралельної обробки інформації, а також багато-багато інших, які часто узагальнюються терміном «організація комп'ютера» чи «структурна організація комп'ютера».

Далі ми будемо використовувати термін «архітектура», розуміючи під цим поняттям логічну структуру у сукупності з фізичною структурною організацією.

На сьогодні найбільшого поширення в ЕОМ набули 2 типи архітектури: принстонська (фон Неймана) і гарвардська. Обидва вони виділяють два основних вузла ЕОМ: центральний процесор і пам’ять комп’ютера. Відмінність полягає в структурі пам'яті: у принстонській архітектурі програми і дані зберігаються в одному масиві пам'яті і передаються до процесора одним каналом, тоді як гарвардська архітектура передбачає окремі сховища і потоки передачі для команд і даних.

В основу архітектури більшості сучасних комп'ютерів покладено принципи, які ще 1946 року було сформульовано у звіті «Попереднє обговорення логічного конструювання електронного обчислювального пристрою» Джоном фон Нейманом і його колегами Г. Голдстайном та А. Берксом. Усі комп'ютери, побудовані згідно з цими принципами, тепер відомі як комп'ютери з фоннеймановською архітектурою.

Основні принципи архітектури фон Неймана такі:

- використання двійкової системи числення для кодування інформації у комп'ютері;
- програмне керування роботою комп'ютера;
- збереження програм у пам'яті комп'ютера;
- адресація пам'яті.

1.1.1. Принцип використання двійкової системи числення

Інформація (команди і дані) у комп'ютері кодується у *двійковій системі числення. Система числення* – це система позначення чисел. У повсякденній практиці використовується десяткова система, в якій числа

записуються за допомогою десяти арабських цифр 0,1,2,...,9. У двійковій системі числення є тільки дві цифри: 0 та 1. Зручність використання двійкової системи в обчислювальній техніці зумовлена тим, що електронні перемикачі можуть перебувати тільки в одному із двох станів: увімкненому чи вимкненому (рис. 1.1).

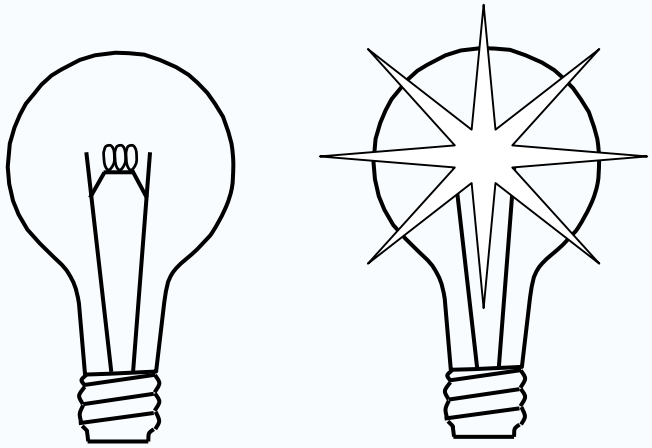


Рис. 1.1. Стани електричної лампи: увімкнена та вимкнена

Ці стани можна кодувати двома цифрами: одиницею або нулем. Так само в одному з двох станів у кожен окремий момент часу може перебувати канал передачі даних: «рівень напруги високий» або «рівень напруги низький» (рис.1.2). Крім того, логіка висловлень є двійковою логікою: будь-яке висловлення в кожен момент є істинним або хибним. Якщо висловлення є істинним, йому відповідає значення 1, якщо хибне – значення 0.

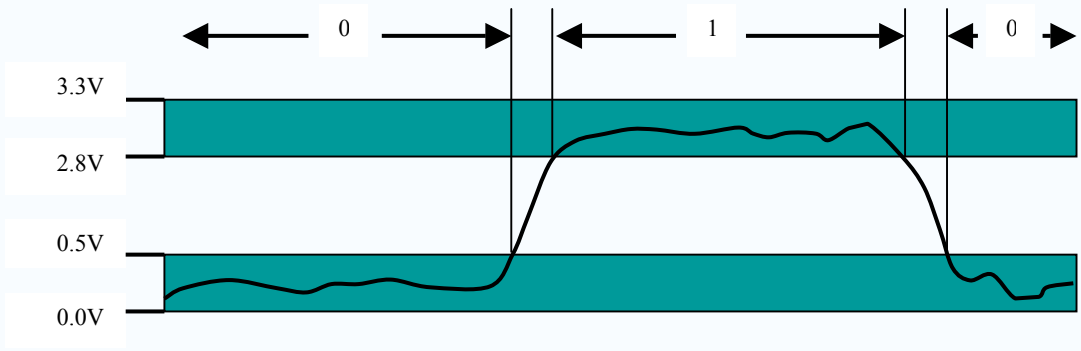


Рис.1.2. Рівень напруги реалізації двійкового коду

Одна двійкова цифра називається *бітом* (від англ. *binary* digit – двійкова цифра). За допомогою одного біта можна закодувати два інформаційних повідомлення, що умовно позначаються символами «0» та «1». Із двох бітів можна утворити коди чотирьох інформаційних повідомлень: «00», «01», «10» та «11»; із трьох бітів – восьми. Загалом за допомогою *n* бітів можна закодувати 2^n інформаційних повідомлень.

Послідовність, що складається з восьми бітів, у сучасній обчислювальній техніці прийнято називати *байтом*. За допомогою одного байта можна закодувати $2^8 = 256$ різних повідомлень і зобразити, наприклад, значення цілих чисел від 0 до 255. Ці самі повідомлення можна розглядати і як числа від -128 до 127 (їх кількість теж дорівнює 256), символи, логічні значення тощо.

Байт є одиницею обсягу пам'яті. Для позначення тисяч та мільйонів байтів використовуються такі одиниці, як кілобайт (1 Кбайт = 2^{10} = 1024 байт), мегабайт (1 Мбайт = 2^{20} = 1 048 576 байт) і гігабайт (1 Гбайт = 2^{30} = 1 073 741 824 байт).

Послідовностями двійкових цифр можна кодувати не лише числа, а й довільні інформаційні повідомлення. Зокрема, загальноприйнята система кодування ASCII ставить коди у відповідність 256 символам. Наприклад, латинській літері «a» відповідає код $97_{10}= 1100001_2$, а символу «@» – код $64_{10}= 1000000_2$. Складніші системи кодування дають можливість закодувати зображення, звук тощо [10].

1.1.2. Принцип програмного керування роботою комп'ютера

Сучасний комп'ютер – це сукупність технічних і програмних засобів, які призначені для автоматизованої обробки дискретних даних відповідно до заданого алгоритму. Алгоритм – це основне поняття математики та обчислювальної техніки, і згідно зі стандартом

ISO 2382/1-84 його визначають як «скінченний набір інструкцій, які описують процес розв'язування задачі за допомогою скінченної кількості операцій».

Усі обчислення, що виконує комп'ютер, мають бути зображені в пам'яті у вигляді програми, складеної з інструкцій, які прийнято називати командами. Кожна команда вказує на певну операцію, яку має виконати комп'ютер. Сукупність команд, що використовує конкретний комп'ютер, називають системою команд. Програма складається з послідовності команд, які процесор виконує автоматично в певному порядку. Змінювати порядок виконання команд залежно від проміжних результатів обчислень дозволяють команди умовного та безумовного переходів. Завдяки цьому можна багаторазово виконувати одну й ту саму послідовність команд програми, а отже, набагато скоротити її розміри.

1.1.3. Принцип збереження програм у пам'яті комп'ютера

Один із найважливіших принципів, який називають також принципом однорідності пам'яті, полягає в тому, що команди та дані зберігаються в одних і тих самих областях пам'яті і нерідко кодуються тими самими станами комірок пам'яті. Комп'ютер обробляє і команди, і дані однаково, тобто над командою можна виконати ті самі операції, що й над будь-якими даними. Це розкриває нові можливості для перетворення програми безпосередньо під час її виконання. Наприклад, команди однієї частини програми можна отримати як результат виконання команд іншої. Ця можливість покладена в основу трансляторів, які перетворюють текст програми мовою високого рівня у послідовність команд конкретного комп'ютера [10].

Принцип однорідності пам'яті, запропонований у статті фон Неймана, використовувався в обчислювальних системах, які

створювались у Принстонському університеті, тому архітектура таких обчислювальних систем дістала назву принстонської. Практично водночас у Гарвардському університеті запропонували іншу архітектуру, згідно з якою команди і дані повинні використовувати окрему пам'ять. Принстонська архітектура була і є домінуючою, проте останнім часом завдяки широкому використанню нових технологій пам'яті все більше поширюються комп'ютери з гарвардською архітектурою.

1.1.4. Принцип адресності пам'яті

Для виконання програми необхідно, щоб команди та дані перебували в основній пам'яті [10]. Основна пам'ять – це послідовність комірок, кожна з яких має свій номер – *адресу*. Розмір комірки, зазвичай, дорівнює восьми двійковим розрядам – байту. Числове значення у пам'яті, як правило, займає декілька сусідніх байтів. Для збереження цілих чисел найчастіше використовують один, два або чотири байти, для нецілих (дійсних) – 4, 6, 8 або 10 байт.

Адресою числа вважається адреса першого байта області пам'яті, відведеної для збереження числа. Так, якщо 32-розрядне двійкове число зберігається в комірках 200, 201, 202 та 203, то його адресою буде 200. Такий метод адресації називається адресацією молодшого байта і використовується в комп'ютерах фірми Intel і міні-комп'ютерах компанії DEC. Існує й інший метод, коли адресою числа є його старший байт (метод адресації старшого байта). Такий метод використовується в комп'ютерах фірм IBM та Motorola.

Процесор у будь-який момент може отримати доступ до будь-якої комірки пам'яті. Така пам'ять називається пам'яттю з довільним доступом.

1.1.5. Архітектура комп'ютерів фон Неймана

У класичній роботі фон Неймана перелічено основні пристрої, з використанням яких може бути побудований комп'ютер. Вважають, що комп'ютери такого типу мають *фоннеймановську архітектуру*. Типова фоннеймановська обчислювальна машина має такі складові [20]: арифметико-логічний пристрій, пристрій керування, пам'ять і пристрої введення-виведення (рис. 1.3).

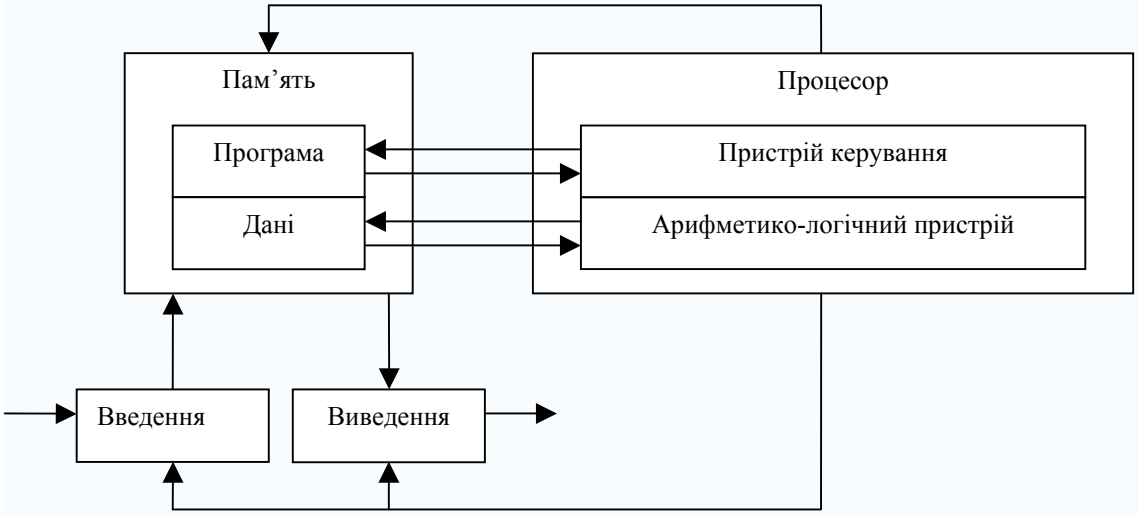


Рис. 1.3. Структура фоннеймановської обчислювальної машини

В обчислювальних машинах із класичною фоннеймановською архітектурою функції обробки інформації покладені на *арифметико-логічний пристрій* (АЛП). До функцій АЛП входить у першу чергу виконання арифметичних і логічних команд та команд зсуву. Отже, цей пристрій забезпечує обробку вхідних даних і формування результату. АЛП – це не один, а ціла група операційних пристроїв, кожен з яких реалізує певну підмножину операцій.

Пристрій керування (ПК) – найголовніша частина комп'ютера, що координує роботу всіх його пристроїв та забезпечує виконання всіх програм. Основною функцією цього пристрою є формування сигналів, необхідних для вибирання команд із пам'яті в порядку, що задається програмою, і подальше виконання цих команд. Крім того, ПК формує сигнали, необхідні для синхронізації та координації дій зовнішніх і внутрішніх пристроїв комп'ютера. Синхронізуючі сигнали – це сигнали, що визначають, коли має бути виконана певна операція. Реальні синхронізуючі сигнали, які керують пересиланням даних і виконанням команд, генеруються схемами керування.

Пристрій керування можна уявити собі як окремий блок, котрий взаємодіє з іншими блоками комп'ютера. Однак на практиці так буває рідко. Більша частина керуючих схем фізично розподілена між різними компонентами комп'ютера: процесором, чипсетом, контролерами введення-виведення та шин.

АЛП і ПК дуже тісно взаємодіють між собою, і їх часто реалізують єдиним пристроєм, відомим як центральний процесор чи просто процесор. Процесор – це основа будь-якого комп'ютера, і швидкість роботи комп'ютера цілковито залежить від моделі та параметрів процесора.

Крім сумарних даних, АЛП формує також ознаку, яка визначає коректність отриманого результату, а також інші його характеристики (рівність нулю, переповнення, парність і т. ін.). Значення ознаки результату може аналізуватися ПК для прийняття рішення стосовно подальшої послідовності виконання програми.

Пам'ять комп'ютера призначена для збереження інформації й оперативного обміну нею з іншими компонентами комп'ютера. Пам'ять поділяють на внутрішню та зовнішню. Внутрішня пам'ять складається з регістрів процесора, основної пам'яті та кеш-пам'яті.

Регістри процесора – це найбільш швидкодіючий, але найменший за обсягом різновид пам'яті комп'ютера. Зазвичай регістрів у процесорі небагато, і тільки в комп'ютерах з неповним набором команд (Reduced Instruction Set Computer, RISC) їх кількість може сягати кількох сотень. Регістри залежно від свого призначення можуть мати обсяг від 1 до 10 байт.

Основна пам'ять може включати пам'ять двох типів – *постійну* й *оперативну*.

Постійна пам'ять (Read Only Memory, ROM) реалізується у вигляді постійного запам'ятовувального пристрою, дані в якому не можна модифікувати, їх можна тільки читати. Основне призначення ROM – підтримання процедур початкового завантаження операційної системи та обслуговування переривань, які припиняють виконання програми з метою реалізації спеціальних системних дій.

Процес зчитування інформації з постійної пам'яті майже не відрізняється від процесу зчитування з оперативної пам'яті. З іншого боку, процес записування інформації в ROM набагато складніший і потребує великих затрат часу й енергії. Записування інформації в постійну пам'ять називають її програмуванням. Сучасні пристрої постійної пам'яті – це мікросхеми, інформація до яких може записуватись як під час виготовлення, так і після нього.

Оперативна пам'ять (Ramdom Access Memory, RAM) використовується і для читання, і для запису інформації. Під час роботи комп'ютера в ній зберігаються програми та дані. Оперативна пам'ять комп'ютера організована у вигляді множини байтів, або комірок, у яких зберігаються числові та символьні значення. Байти оперативної пам'яті послідовно пронумеровані, починаючи з 0. Ці номери байтів є їх *адресами*. Залежно від типів даних, над якими виконуються дії, байти групуються у слова (два байти) або подвійні слова (чотири байти).

Число, що зберігається в комірці, – це її *значення*, або *вміст*. Якщо в *k*-й комірці міститься, наприклад, число *m*, то прийнято говорити «вміст комірки з адресою *k* дорівнює *m*». Під час читання даних з оперативної пам'яті вміст комірки не змінюється. Для його зміни потрібно записати в цю комірку нове значення, або, інакше кажучи, присвоїти комірці нове значення. Поняттю адреси оперативної пам'яті повною мірою відповідає поняття змінної в алгебрі. Змінну позначають ідентифікатором. Цьому ідентифікатору ставиться у відповідність адреса комірки оперативної пам'яті. Якщо змінній надати певне значення, то воно записується за адресою відповідної комірки пам'яті.

Особливістю RAM є неможливість зберігання інформації в ній після вимикання живлення комп'ютера. Саме цим вона відрізняється від пам'яті ROM, в якій дані зберігаються незалежно від наявності живлення.

Між регістрами та процесором часто розміщується *кеш-пам'ять*, призначена для узгодження швидкостей роботи основної пам'яті та процесора. Сучасні комп'ютери мають декілька рівнів кеш-пам'яті, які позначаються літерою L з певним номером. У більшості персональних комп'ютерів кеш-пам'ять має два рівні – L1 та L2. В останніх розробках все частіше з'являється кеш-пам'ять третього рівня, а в проектах – навіть і четвертого [20]. Обсяг кеш-пам'яті вимірюється десятками і сотнями кілобайтів, а оперативної – десятками і сотнями мегабайтів. Оперативна пам'ять сучасних персональних комп'ютерів сягає кількох гігабайтів.

Для довгострокового збереження великого обсягу даних і програм потрібна *зовнішня пам'ять*. Зокрема, у зовнішній пам'яті зберігається все програмне забезпечення комп'ютера (системне та прикладне). До складу зовнішньої пам'яті входять різноманітні пристрої: накопичувачі на жорстких і гнучких магнітних дисках, пристрої на касетній магнітній стрічці (стримери), накопичувачі на оптичних дисках CD-ROM та CD-RW (від англ. Compact Disk Read Only Memory – компакт-диск тільки для

читання, Compact Disk Read Write Memory – компакт-диск для запису та читання відповідно) тощо. Накопичувачі на жорстких магнітних дисках відрізняються від накопичувачів на гнучких магнітних дисках за конструкцією, обсягом даних, що зберігаються, а також часом пошуку, запису і зчитування інформації. Необхідно пам'ятати, що інформація, яка зберігається у зовнішній пам'яті, стане доступною для процесора тільки після того, як буде переписана в основну пам'ять.

Накопичувач можна розглядати як пристрій, у якому носій інформації поєднаний з приводом, що є механізмом зчитування-запису інформації, та відповідним керуючим пристроєм. Дисковий привід називається дисководом, а стрічковий – стримером. Комп'ютери зазвичай мають декілька дисководів для роботи з дисками різних типів.

Пристрої введення-виведення є важливою складовою будь-якого комп'ютера. Вони забезпечують взаємодію комп'ютера з навколишнім середовищем, користувачами, об'єктами керування та іншими комп'ютерами.

Зовнішні пристрої за їх призначенням можна розділити на два основних різновиди – для тривалого зберігання інформації та для взаємодії комп'ютера із зовнішнім середовищем, а саме користувачами, іншими комп'ютерами й об'єктами керування.

Серед них можна виділити пристрої введення (клавіатура, сканер, графічний планшет, маніпулятори типу «миша», джойстик, світлове перо тощо), виведення (принтер, плотер), діалогові засоби користувача (дисплеї, пристрої мовного введення-виведення – мікрофонні акустичні системи, синтезатори звуку тощо), засоби зв'язку та телекомунікацій (мережні плати, модеми).

Більшість зовнішніх пристроїв мають свої процесори (контролери), які є простішими за центральний процесор і виконують інші набори команд. Контролери можуть переносити дані із зовнішніх носіїв до

оперативної пам'яті (читання, або введення із «зовнішнього світу») чи навпаки (запис, або виведення даних у «зовнішній світ»). Кожному пристрою введення-виведення виділено окрему ділянку оперативної пам'яті – порт. Із нього пристрій бере дані для зовнішнього носія, записуючи їх, наприклад, на диск або на екран комп'ютера. І саме в порт записуються дані, що надходять з клавіатури або дисковода.

Сигнали синхронізації дій усіх пристроїв передаються керуючими лініями – шинами. Шини комп'ютера мають забезпечити передачу інформації між:

- процесором та оперативною пам'яттю (шина процесор-пам'ять);
- процесором і портами введення-виведення зовнішніх пристроїв;
- оперативною пам'яттю і портами введення-виведення зовнішніх пристроїв у режимі прямого доступу до пам'яті.

Шина процесор-пам'ять забезпечує безпосередній зв'язок між процесором комп'ютера та основною пам'яттю. У сучасних комп'ютерах таку шину іноді називають шиною переднього плану і позначають як FSB (від англ. Front-Side Bus). Інтенсивний обмін даними між процесором та пам'яттю вимагає, щоб кількість інформації, яка передається за одиницю часу (секунду) цією шиною (пропускна спроможність шини), була якомога більшою. Від пропускної спроможності шини процесор-пам'ять значною мірою залежить продуктивність комп'ютерів із фон-нейманівською архітектурою. Функції різних шин конструктори іноді покладають на єдину системну шину, але з погляду швидкодії краще, коли дані між процесором і пам'яттю передаються окремою шиною.

Зв'язок процесора чи пам'яті із пристроями введення-виведення забезпечують *шини введення-виведення*. На відміну від шини процесор-пам'ять, такі шини містять менше ліній, але фізична довжина цих ліній може бути більшою. Велика кількість і різноманітність зовнішніх пристроїв у різних типах комп'ютерів обумовили необхідність

розроблення стандартів для таких шин. Стандартизація шин дозволяє розробникам зовнішніх пристроїв працювати незалежно, а користувачам – самостійно формувати потрібну конфігурацію комп'ютера.

Фізично шини складаються з великої кількості паралельних металевих провідників – ліній, розміщених на системній платі чи кристалі мікросхеми. Серед ліній будь-якої шини можна виділити три функціональні групи: *шина адреси*, *шина даних* і *шина керування*.

Шиною адреси передаються адреси комірок пам'яті, номери регістрів процесора, адреси портів введення-виведення тощо. Кількість ліній, виділених для передавання адреси, становить ширину шини адреси і визначає максимально можливий обсяг пам'яті комп'ютера, який може адресуватися.

Лінії, якими дані (команди чи операнди) передаються між блоками системи, називаються шиною даних. Найважливіші параметри шини даних – ширина та пропускна спроможність. Ширина шини даних вказує на кількість бітів інформації, які можуть бути передані шиною за один її цикл.

Використання окремих шин для адрес і даних дає можливість значно підвищити продуктивність комп'ютера, особливо під час записування інформації в пам'ять – адреса комірки пам'яті та дані, що записуються, передаються паралельно.

Поряд із лініями для передавання адрес та даних обов'язковим атрибутом будь-якої шини є лінії, якими передається керуюча інформація та інформація про стан пристроїв введення-виведення. Сукупність таких ліній прийнято називати шиною керування. Цією шиною передаються сигнали синхронізації, переривання, арбітра шини тощо.

Загалом, функціонування фоннеймановського комп'ютера можна описати так: комп'ютер за допомогою пристроїв введення приймає інформацію у вигляді програм та даних і записує її в пам'ять; інформація,

що зберігається в пам'яті, під керуванням програми шинами пересилається в арифметико-логічний пристрій для подальшої обробки; дані, отримані після обробки інформації, спрямовуються (також шинами) на пристрої виведення.

1.2. Типи комп'ютерів та комп'ютерне обладнання

Різнотиповість комп'ютерів обумовлюється передусім різнотиповістю обчислень і задач, для виконання яких вони призначені. З часом виникають все нові класи задач, що, у свою чергу, призводить до появи комп'ютерів нових типів. Тому наведена нижче класифікація містить лише найбільш поширені типи комп'ютерів [15]:

- суперкомп'ютери;
- мейнфрейми;
- сервери різних типів;
- персональні комп'ютери та робочі станції.

Суперкомп'ютери – спеціальний тип комп'ютерів, що створюється для розв'язування надзвичайно складних обчислювальних задач (підготовки прогнозів погоди, моделювання складних процесів та явищ природи, оброблення великих обсягів інформації). Суперкомп'ютери – це багатопроцесорні або багатомашинні комплекси продуктивністю понад сотні мільярдів операцій за секунду. Основний принцип роботи всіх суперкомп'ютерів – принцип паралелізму. Суть його полягає в тому, що комп'ютер здатний виконувати одночасно (паралельно) велику кількість операцій.

Однією з провідних компаній світу з виробництва суперкомп'ютерів багато років вважалася компанія Cray Research. Її засновник, людина-легенда Сеймур Крей, уже в середині 70-х років побудував комп'ютер

Cray-1, що вражав світ своєю швидкодією: десятки і навіть сотні мільйонів арифметичних операцій за секунду. Сьогодні на ринку суперкомп'ютерів домінують більш сучасні системи компаній NEC, HP, IBM.

Мейнфрейм – це синонім поняття «велика універсальна електронно-обчислювальна машина». Мейнфрейми і на сьогодні залишаються найбільш потужними (якщо не враховувати суперкомп'ютери) обчислювальними системами загального призначення, які можуть експлуатуватися у безперервному режимі. Зазвичай мейнфрейми – це багатопроцесорні системи, що містять один або кілька центральних і периферійних процесорів із спільною пам'яттю, зв'язаних між собою високошвидкісними магістралями передавання даних. При цьому основне навантаження щодо обчислень покладено на центральні процесори, а периферійні процесори забезпечують роботу з різноманітними периферійними пристроями.

Основними постачальниками мейнфреймів є відомі комп'ютерні компанії Amdahl, ICL, Siemens Nixdorf і деякі інші, але провідна роль у цій галузі, безумовно, належить компанії IBM. Саме архітектура випущеної в 1964 році системи IBM/360 і її наступних поколінь стала зразком обчислювальних систем цього типу. В СРСР протягом багатьох років випускалися машини серії ЄС ЕОМ (Єдина серія електронно-обчислювальних машин), що були вітчизняним аналогом цієї системи.

Комп'ютер, що працює в локальній або глобальній мережі, може спеціалізуватися на обслуговуванні інших комп'ютерів. Такий комп'ютер називається *сервером* (від англ. *serve* – обслуговувати, керувати). Є кілька типів серверів, орієнтованих на різні сфери застосування: файловий сервер, сервер бази даних, сервер друку, обчислювальний сервер, сервер програмних застосувань. Тип сервера визначається видом ресурсу, яким

він керує (файлова система, база даних, принтери, процесори або пакети прикладних програм).

Існує також класифікація серверів на основі масштабу мережі, в якій вони використовуються: сервер робочої групи, сервер відділу або сервер масштабу підприємства (його ще називають корпоративним сервером). Загалом, ці класифікації досить умовні. Наприклад, розмір групи може змінюватися в діапазоні від кількох людей до декількох сотень, а сервер відділу може обслуговувати від 20 до 150 користувачів. Очевидно, що вимоги до складу устаткування і програмного забезпечення сервера, його надійності і продуктивності дуже варіюються залежно від числа користувачів і характеру розв'язуваних ними задач.

Незважаючи на велике різноманіття типів і моделей обчислювальних систем, у більшості користувачів поняття «комп'ютер» асоціюється в першу чергу з *персональним комп'ютером*. Перший персональний комп'ютер з'явився у 1975 році. І відразу стало зрозуміло, що невисока ціна і досить потужні обчислювальні можливості комп'ютерів цього класу сприятимуть їх швидкому поширенню.

Персональні комп'ютери зробили революцію у професійній діяльності мільйонів людей, вплинули на всі сфери розвитку суспільства. Комп'ютери цього типу стали незамінним інструментом у роботі інженерів і вчених. Особливо значною є їх роль під час проведення наукових експериментів, що вимагають складних і тривалих обчислень.

Сучасний світ наповнений не лише звичними персональними комп'ютерами, а й комп'ютерами-невидимками – мікропроцесорами, що є комп'ютерами у мініатюрі. Крім блоку обробки даних такий пристрій містить блок керування і навіть пам'ять. Це означає, що мікропроцесор здатний автономно виконувати всі необхідні дії з інформацією. Масове поширення мікропроцесори одержали у сучасній техніці, якою можна керувати за допомогою обмеженої послідовності команд. Наприклад,

керування сучасним двигуном (забезпечення економних витрат палива, обмеження максимальної швидкості руху, контроль за справністю тощо) немислиме без використання мікропроцесорів. Ще однією сферою їхнього використання є побутова техніка – застосування мікропроцесорів додає їй нових споживчих якостей.

Методи класифікації комп'ютерів. Номенклатура видів комп'ютерів на сьогодні величезна: машини розрізняються за призначенням, потужністю, розмірами, елементною базою тощо. Тому класифікують ЕОМ за різними ознаками. Слід зауважити, що будь-яка класифікація є певною мірою умовною, оскільки розвиток комп'ютерної науки і техніки настільки стрімкий, що, наприклад, сьогоднішня мікро-ЕОМ не поступається за потужністю міні-ЕОМ п'ятирічної давності і навіть суперкомп'ютерам віддаленого минулого. Крім того, зарахування комп'ютерів до певного класу досить умовне як через нечіткість розмежування груп, так і внаслідок упровадження в практику замовного складання комп'ютерів, коли номенклатуру вузлів і конкретні моделі їх адаптують до вимог замовника. Розглянемо найбільш поширені критерії класифікації комп'ютерів.

За призначенням ПК класифікують: великі електронно-обчислювальні машини (ЕОМ); міні-ЕОМ; мікро-ЕОМ; персональні комп'ютери.

Великі ЕОМ (Main Frame) застосовують для обслуговування великих галузей народного господарства. Вони характеризуються 64-розрядними паралельно працюючими процесорами (кількість яких досягає до 100), інтегральною швидкодією до десятків мільярдів операцій за секунду, багатокористувальним режимом роботи. Домінуюче положення у випуску комп'ютерів такого класу займає фірма IBM (США). Найбільш відомими моделями супер-ЕОМ є: IBM 360, IBM 370, IBM ES/9000, Cray 3, Cray 4, VAX-100, Hitachi, Fujitsu VP2000.

Міні-ЕОМ подібна до великих ЕОМ, але менших розмірів. Використовують на великих підприємствах, у наукових закладах і установах. Часто використовують для керування виробничими процесами. Характеризуються мультипроцесорною архітектурою, підключенням до 200 терміналів, дисковими запам'ятовувальними пристроями, що нарощуються до сотень гігабайт, розгалуженою периферією. Для організації роботи з міні-ЕОМ потрібен обчислювальний центр, але менший, ніж для великих ЕОМ.

Мікро-ЕОМ доступні багатьом установам. Для обслуговування достатньо обчислювальної лабораторії у складі декількох осіб, з наявністю прикладних програмістів. Необхідні системні програми купуються разом з мікро-ЕОМ, розробку прикладних програм замовляють у великих обчислювальних центрах або спеціалізованих організаціях.

Програмісти обчислювальної лабораторії займаються втіленням придбаного або замовленого програмного забезпечення, налагоджують його і узгоджують його роботу з іншими програмами та пристроями комп'ютера. Можуть вносити зміни в окремі фрагменти програмного та системного забезпечення.

Персональні комп'ютери бурхливого розвитку набули в останні 20 років. Персональний комп'ютер (ПК) призначений для обслуговування одного робочого місця і спроможний задовольнити потреби малих підприємств та окремих осіб [3]. З появою Інтернету популярність зросла значно вище, оскільки за допомогою персонального комп'ютера можна користуватись науковою, довідковою, навчальною та розважальною інформацією.

Персональні комп'ютери умовно можна поділити на професійні та побутові, але в зв'язку із здешевленням апаратної частини, межі між ними розмиваються. З 1999 року задіяний міжнародний сертифікаційний стандарт – специфікація PC99: масовий персональний комп'ютер

(Consumer PC); офісний персональний комп'ютер (Office PC); портативний персональний комп'ютер (Mobile PC); робоча станція (WorkStation); розважальний персональний комп'ютер (Entertaiment PC).

Більшість персональних комп'ютерів на ринку підпадають до категорії масових ПК. Офісні ПК мають мінімум засобів відтворення графіки та звуку. Портативні ПК відрізняються наявністю засобів з'єднання віддаленого доступу (комп'ютерний зв'язок). Робочі станції мають збільшені вимоги до пристроїв збереження даних. У розважальних ПК основний акцент роблять на засобах відтворення графіки та звуку.

За рівнем спеціалізації ПК поділяють на універсальні та спеціалізовані. На базі універсальних ПК можна створити будь-яку конфігурацію для роботи з графікою, текстом, музикою, відео тощо. Спеціалізовані ПК створені для рішення конкретних задач, зокрема, бортові комп'ютери у літаках та автомобілях. Спеціалізовані міні-ЕОМ для роботи з графікою (кіно-, відеофільми, реклама) називають графічними станціями. Спеціалізовані комп'ютери, що об'єднують комп'ютери у єдину мережу, називають файловими серверами. Комп'ютери, що забезпечують передачу інформації через Інтернет, називають мережними серверами.

Існує безліч видів і типів комп'ютерів, що збираються з деталей, які виготовлені різними виробниками. Важливим є сумісність забезпечення комп'ютера: апаратна сумісність (платформа IBM PC та Apple Macintosh); сумісність на рівні операційної системи; програмна сумісність; сумісність на рівні даних.

Розрізняють три основних типи персональних комп’ютерів: стаціонарні персональні комп’ютери; портативні персональні комп’ютери; нестандартні конструкції персональних комп’ютерів.

Найбільш поширеними є стаціонарні персональні комп’ютери, які дають змогу легко змінювати конфігурацію. Мобільні персональні

комп’ютери зручні для користування, мають засоби комп'ютерного зв'язку. Нестандартні конструкції персональних комп’ютерів призначені для забезпечення вимог користувача, пов’язаних з необхідністю специфічного використання ПК.

1.2.1. Стаціонарні персональні комп’ютери

Апаратне забезпечення персональних комп’ютерів. Персональний комп’ютер на сьогодні – це складна технічна система, до складу якої може входити велика кількість функціонально завершених пристроїв. Кількість цих пристроїв та їх призначення, у більшості випадків, визначається колом задач, які повинні розв’язуватися з використанням комп’ютера. І якщо максимальну кількість пристроїв, які можуть входити до складу технічної системи, визначити досить складно, то так звана мінімальна конфігурація персонального комп’ютера обмежується наявністю чотирьох складових (рис.1.4): системного блока, монітора, клавіатури та миші.

Рис. 1.4. Персональний комп’ютер

Усі пристрої, з яких складається технічна система персонального комп’ютера, прийнято поділяти на *внутрішні* та *зовнішні* (периферійні). Внутрішні пристрої знаходяться всередині системного блоку. Зовнішні пристрої підключаються до нього ззовні за допомогою спеціальних провідників через спеціальні роз’єми (порти) або безпровідними засобами (інфрачервоний зв’язок, радіозв’язок).

Системний блок є основним вузлом, всередині якого розміщені

найбільш важливі компоненти. На зовнішній вигляд системні блоки розрізнюються формою корпусу, які випускають у *горизонтальному* (desktop) *і вертикальному* (tower) виконанні. Корпуси, що мають вертикальне виконання, розрізняють за габаритами: *повнорозмірний (big tower), середньорозмірний (midi tower) і малорозмірний (mini tower).* Габарит корпусу впливає на можливість установлення в ньому різної кількості таких пристроїв, як дисководи гнучких магнітних дисків, дисководи оптичних дисків, картрідери, жорсткі магнітні диски тощо, для монтажу яких передбачена відповідна кількість відсіків. Серед корпусів, що мають горизонтальне виконання, виділяють *плоскі* і *особливо плоскі (slim).*

Важливим узагальненим параметром системного блоку є *формфактор*. Це стандарт технічного виробу, що описує певну сукупність його параметрів, наприклад, форму, розмір, розміщення і типи роз'ємів, вимоги до вентиляції, рівнів напруги та інших параметрів [2]. Найчастіше він вживається стосовно складних технічних виробів (системних блоків комп'ютерів, стільникових телефонів, ноутбуків, жорстких дисків). Від формфактора наведених виробів залежать вимоги до пристроїв, що розміщуються в них. Існує значна кількість формфакторів системних блоків, основними з яких є *AT, ATX, mini-ITX, BTX.*

Внутрішні пристрої системного блоку. Незалежно від параметрів та зовнішнього вигляду системного блоку в ньому розміщується майже однотипний набір пристроїв.

Блок живлення (рис.1.5) – пристрій, призначений для формування потрібної комп'ютерній системі напруги із напруги мережі живлення. Найчастіше блоки живлення перетворюють змінну напругу мережі живлення величиною 220 В та частотою 50 Гц в постійну напругу величиною в 3,3, 5 та 12 В. Основними характеристиками блока живлення є тип формфактора (*AT, ATX, BTX*), потужність (180 – 450 Вт), кількість роз'ємів для підключення пристроїв, кількість вентиляторів.

Рис. 1.5. Блок живлення

Материнська плата (рис.1.6) – діелектрична пластина із розгалуженою системою провідників, на якій здійснюється монтаж більшості внутрішніх компонентів комп'ютерної системи. Як і для багатьох пристроїв персонального комп'ютера, важливим показником материнської плати є формфактор – стандарт, що визначає розміри материнської плати, місця її кріплення до корпусу, розташування на ній портів введення/виведення, сокета (роз'єму) центрального процесора, роз'ємів для модулів оперативної пам'яті, а також тип роз'єму для підключення блока живлення. Існує досить велика кількість формфакторів материнських плат: застарілими з них вважаються *Baby-AT, Mini-ATX, AT, LPX*; сучасними – *ATX, micro-ATX, Flex-ATX, NLX, WTX, Mini-ITX, Nano-ITX, BTX, Micro-BTX, PicoBTX*. Візуальною відмінністю сучасних материнських плат від застарілих є наявність на них роз'ємів *USB (Universal Serial Bus – універсальна послідовна шина)*, через які здійснюється підключення більшості сучасних периферійних пристроїв.

Найважливішими компонентами комп'ютерної системи, розміщеними на материнській платі, є:

Рис. 1.6. Материнська плата

- *процесор (central processing unit – CPU*) – основна мікросхема, що виконує більшість математичних і логічних операцій під час обробки інформації;
- *оперативна пам'ять (оперативно запам'ятовувальний пристрій – ОЗП)* – набір мікросхем, призначених для тимчасового зберігання даних і команд, які необхідні процесору для виконання ним операцій;
- *постійна пам'ять (постійний запам'ятовувальний пристрій – ПЗП)* – мікросхема, призначена для тривалого зберігання комплекту програм, який утворює базову систему введення/виведення (BIOS);
- *мікропроцесорний комплект (чіпсет)* – набір мікросхем, який здійснює керування внутрішніми пристроями комп'ютера та визначає основні функціональні можливості материнської плати;
- *роз'єми (слоти)* – для підключення плат розширення (відеокарти, звукової карти, мережевої карти тощо). Сучасна материнська плата персонального комп'ютера, як правило, включає слоти розширення форматів *PCI, AGP, PCI Express*, а також *IDE/ATA, SATA і USB* контролери. Під'єднання до материнської плати більшості пристроїв здійснюється за допомогою одного або декількох слотів розширення, а деякі сучасні материнські плати підтримують підключення безпровідних пристроїв, використовуючи протоколи *IrDA (інфрачервоний зв'язок), Bluetooth (радіозв'язок),* або *802.11 (Wi-Fi)*.

Сокет (Socket) – роз'єм, призначений для встановлення процесора. Материнські плати оснащуються стандартизованими типами сокетів, які визначають можливість встановлення певного переліку модифікацій процесорів різних фірм-виробників.

Жорсткий магнітний диск, «вінчестер» (рис. 1.7) – основний енергонезалежний накопичувач даних комп'ютерної системи. Дисковий накопичувач складається з набору металевих пластин, покритих магнітним матеріалом, і сполучених між собою за допомогою центрального шпинделя.

Рис. 1.7. Жорсткий магнітний диск

Шпиндель обертається з високою постійною швидкістю. Для запису даних використовуються обидві поверхні пластини. В сучасних дискових накопичувачах використовується від 4 до 9 пластин. Кожна пластина містить набір концентричних доріжок для запису. Зазвичай доріжки діляться на *сектори* об'ємом 512 байт. Кількість секторів, що записуються на одну доріжку, залежить від фізичних розмірів пластини і щільності запису.

Основними характеристиками жорстких магнітних дисків є:

- *Інтерфейс* – спосіб передавання даних.
- *Ємність* – кількість даних, які можуть зберігатися накопичувачем.
- *Фізичний розмір пластин* – майже всі сучасні накопичувачі для персональних комп'ютерів і серверів мають розмір 3,5 або 2,5 дюйма. Останні частіше застосовуються в ноутбуках. Інші поширені формати – 1,8 дюйма, 1,3 дюйма і 0,85 дюйма.
- *Швидкість обертання шпинделя* – кількість обертів шпинделя за хвилину.
- *Ємність кеш-пам'яті* – кількість даних, які можуть розміщуватися у спеціальній швидкодіючій пам'яті, що розміщена в корпусі жорсткого диска. Цей показник впливає на швидкість передачі даних.

- *Час довільного доступу* – це час, необхідний для позиціонування магнітних головок на відповідну доріжку, яка містить необхідні дані. Він знаходиться в межах від 3 до 15 мс.

Пристрої читання та запису інформації на зовнішні носії – низка пристроїв, які виконуються у розмірах 3,5″ (дюйма), або 5,25″ під відповідні відсіки системного блоку і забезпечують процес зберігання та обміну інформацією засобами зовнішніх носіїв (гнучких магнітних дисків, оптичних дисків, флеш-карток, магнітооптичних дисків, магнітних дисків підвищеної щільності тощо).

Рис. 1.8. Пристрої читання та запису інформації на зовнішні носії

Найдовше (з 1981 року) в комп'ютерних системах використовуються *дисководи (приводи) гнучких магнітних дисків (ГМД)* розміром 3,5″ (рис.1.8). Цей привід здійснює читання та запис інформації на відповідний носій (дискету), що обертається зі швидкістю 300 об/хв, під впливом магнітного поля, яке створюють дві магнітні головки (окремо для читання та запису). Дисковод *ГМД*, до недавнього часу, входив до стандартної комплектації системного блоку.

Невід'ємною складовою сучасних системних блоків є *дисководи оптичних дисків* типу: CD, DVD, HD DVD та Blu-ray. Найпершими з них почали застосовувати *CD-приводи* (рис 1.9), які здійснювали тільки операцію зчитування інформації з дисків *CD-ROM (Compact Disk Read Only Memory)*. Такі приводи почали розміщувати у відсіках замість застарілих

Рис. 1.9. CD-привід

дисководів гнучких магнітних дисків розміром 5,25″. Технологія зчитування інформації CD-приводом з диска базується на застосуванні променя лазера довжиною 780 нм, який, опромінюючи поверхню диска, або повертається на сприймальний фотодіод, або розсіюється. Таким чином подається інформація про логічний 0 чи 1. Для проведення операції запису інформації застосовуються *CD-R-приводи* (для одноразового запису дисків) та *CD-RW-приводи* (для багаторазового перезапису дисків), в яких використовуються відповідні диски. Основною характеристикою CD-приводів є швидкість читання та запису інформації, яка представляється у вигляді цілого числа. Це число кратне 150 Кбайт/с. Наприклад, 48-швидкісний привід забезпечує максимальну швидкість читання (або запису) CD дисків, яка дорівнює 48 · 150 = 7200 Кб/с (7,03 Мб/с). Для *CD-RW-приводів* вказують 3 таких числа, наприклад $52_{\times}32_{\times}52_{\times}$, де перше число вказує на швидкість запису, друге – швидкість перезапису, третє – швидкість читання інформації.

Наступним поколінням приводів для зчитування інформації з оптичних дисків типу *DVD (Digital Versatile Disc)*, якими зараз комплектують системні блоки, є *DVD-приводи* (рис. 1.10). Аналогічно CD-приводам вони є трьох типів: DVD-ROM – тільки для зчитування інформації з дисків, DVD-R та DVD-RW для запису та перезапису дисків відповідно.

Рис. 1.10. DVD-привід

DVD-приводи мають такі ж розміри, як і CD-приводи і розміщуються у відсіках 5,25″ системного блоку. Більший обсяг даних (до 17 Гб), який здатний записати DVD-привод на DVD диск, обумовлено застосуванням в ньому лазера з меншою довжиною хвилі, зокрема 650 нм. DVD-приводи здатні працювати також

із CD-дисками, оскільки фізичні розміри CD і DVD дисків однакові. Основними характеристиками DVD-приводів є швидкості читання, запису та перезапису інформації. Зазначені характеристики наводяться окремо для всіх типів дисків, з якими він може працювати. При цьому треба враховувати те, що одиниця швидкості ($1_{\times}$) читання/запису DVD становить 1,38 Мб/с, що приблизно відповідає 9-й швидкості ($9_{\times}$) читання/запису CD. Наприклад, 16-швидкісний DVD-привід забезпечує швидкість читання (або запису) DVD дисків рівну 16 · 1,32 = 21,12 Мб/с. Зокрема, характеристика DVD-приводу може виглядати так: $10_{\times}10_{\times}$ DVD+R9/-R9 DL, $12_{\times}$ DVD-RAM, $18_{\times}8_{\times}18_{\times}6_{\times}16_{\times}$ DVD+R/+RW/-R/-RW/-ROM, $40_{\times}32_{\times}40_{\times}$ CD-R/-RW/-ROM (де: R9 DL – це тип диска, зокрема односторонній двошаровий; DVD-RAM – Random Access Memory – диски багаторазового запису з довільним доступом. Поняття пам'яті з довільним доступом передбачає, що в процесі звернення до даних не враховується порядок їх розташування).

Рис. 1.11. HD DVD-привід

Найсучаснішими приводами оптичних носіїв є *HD DVD-привід* для роботи із *HD DVD* дисками (*High Definition DVD – DVD високої чіткості*) (рис. 1.11) та *BD-привід* для роботи із Blu-ray Disc *(blue ray* – голубий промінь) (рис.1.12).

Рис. 1.12. BD-привід

За зовнішньої схожості з попередніми типами приводів, головною відмінністю їх є те, що в них застосовується «синій» (технічно це синьо-фіолетовий) лазер з довжиною хвилі 405 нм. Це дозволяє записувати на відповідні HD DVD та BD диски до сотень гігабайт інформації. Проте, незважаючи на однотипність лазера, який застосовується в обох типах

приводів, технології запису та зчитування інформації у них різні. Кожний із приводів підтримує роботу також із оптичними дисками попередніх поколінь, тобто з CD та DVD, а деякі моделі і з дисками суміжної технології. Обидва типи приводів мають моделі, які здатні здійснювати зчитування, запис та перезапис інформації. Як і в попередніх випадках, характеристиками таких приводів є швидкості цих операцій, які зазначаються окремо для усіх типів дисків, з якими вони можуть працювати. За одиницю швидкості передавання інформації для BD-дисків прийнято значення 4,5 Мб/с. Тобто, якщо на BD-приводі вказано значення швидкості запису 2_X, то максимальна швидкість запису становить 2·4,5=9 Мб/с. Так для BD-приводу повна характеристика може мати вигляд: $2_\times 2_\times 2_\times$ BD-R/-RE/-ROM, $3_\times$ HD DVD-ROM, $4_\times 4_\times$ DVD+R9/-R9 DL, $8_\times 8_\times 8_\times 6_\times 8_\times$ DVD+R/+RW/-R/-RW/-ROM, $24_\times 16_\times 32_\times$ CD-R/-RW/-ROM (де: BD-RE – позначення диска з можливістю перезапису інформації). Для HD DVD-приводу запис характеристик майже аналогічний.

До категорії сучасних пристроїв читання та запису інформації на електронні носії *(флеш-карти)*, можна віднести *картрідери* (рис. 1.13), які розміщуються у відсіках 3,5″ системного блоку. Основними характеристиками картрідерів є кількість роз'ємів та перелік типів флеш-карт, з якими вони можуть проводити операції читання/запису інформації. Як правило, картрідери мають до 5 роз'ємів, хоча перелік типів флеш-карт завжди перевищує кількість роз'ємів. Це пояснюється здатністю одного роз'єму працювати з цілою групою носіїв, які об'єднуються за певними характеристиками.

Рис. 1.13. Картрідер

Компоненти комп'ютерної системи, що розміщені на материнській платі. Процесор (CPU- central processing unit) (рис. 1.14) – основна мікросхема комп'ютера, яка змонтована на кремнієвій пластині. Процесор містить мільйони транзисторів, зв'язаних між собою надтонкими алюмінієвими каналами, що забезпечують їх взаємодію під час запису і обробки даних.

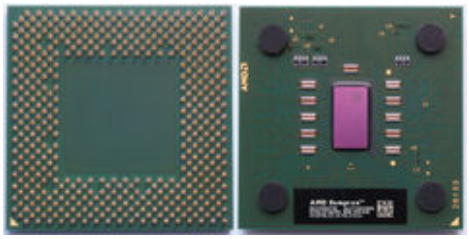


Рис. 1.14. Процесор

Робота процесора передбачає, як правило, виконання арифметичних дій, логічних операцій, управління і переміщення даних (між регістрами, оперативною пам'яттю і портами введення/виведення). З основними пристроями комп'ютерної системи мікропроцесор зв'язаний групою провідників, які називаються *шинами*. Таких шин є три: *адресна* шина, шина *даних* і *командна* шина. Адресна шина забезпечує формування адреси комірки оперативної пам'яті, до якої під'єднується процесор для копіювання необхідних даних. Шиною даних відбувається безпосереднє копіювання даних із оперативної пам'яті. Шиною команд також із оперативної пам'яті у мікропроцесор подаються команди, якими процесор керується під час подальшої обробки даних. Основним показником будь-якої шини є розрядність, тобто кількість паралельних ліній, які беруть участь в обміні інформацією. Якщо, наприклад, шина є 64-розрядною, то вона дозволяє одночасно передати інформацію у розмірі 64 біт. Це може бути 64-бітна адреса, дані, команда. Тобто, якщо команда описується одним байтом (1 байт = 8 біт), то за один раз (такт) по такій шині може бути передано 8 аналогічних команд.

Щодо самого процесора, то ефективність його роботи, а значною мірою і всієї комп'ютерної системи, визначається низкою параметрів, основними з яких є: *розрядність, тактова частота, розмір кеш-пам'яті,*

робоча напруга.

Розрядність показує, яку кількість біт даних процесор може обробити за один такт. Перші процесори так званого сімейства x86 були 16-розрядні, потім 32-розрядні, 64-розрядні і більше.

Тактова частота вказує, скільки елементарних операцій (тактів) виконує мікропроцесор за одну секунду. Її вимірюють в мегагерцах (МГц) або гігагерцах (ГГц). Цей показник динамічно змінювався разом із створенням нових моделей процесорів і вважався основним показником його швидкодії. Перші моделі мікропроцесорів фірми Intel Pentium працювали з частотою 75-100 МГц, у сучасних моделях цей показник наблизився до позначки 4 ГГц. Так, процесор з тактовою частотою 3,4 ГГц забезпечує виконання 3 мільярдів 400 тисяч елементарних операцій за секунду, що забезпечує досить велику продуктивність комп'ютерної системи, хоча загалом її визначають низкою показників, а не тільки тактовою частотою процесора.

Кеш-пам'ять – це пам'ять із більш швидким доступом, в якій дублюється частина даних з іншого носія, що працює повільніше. Найчастіше термін кеш-пам'ять використовується для позначення кеш-пам'яті, що знаходиться між регістрами центрального процесора та оперативною пам'яттю. Кеш-пам'ять може значно підвищити продуктивність комп'ютерної системи з обробки даних, оскільки тактова частота оперативної пам'яті значно нижча тактової частоти процесора, а тактова частота кеш-пам'яті знаходиться на рівні тактової частоти процесора, хоча і незначна за розміром.

Робоча напруга процесора впливає на низку показників. Зокрема, рівень напруги впливає на мінімально допустиму відстань між структурними елементами в кристалі процесора без загрози виникнення електричного пробою. Також пропорційно квадрату напруги відбувається виділення теплоти в процесорі, що накладає певні обмеження на його

продуктивність та систему охолодження. З розвитком процесорної техніки відбувається поступове зниження робочої напруги. У перших моделях x86 процесорів цей показник становив 5 В, потім 3 В, 2,2 В. На сьогодні він понижений до рівня 0,75 В.

Світовими лідерами у виробництві процесорів є компанії *Intel* та *AMD (Advanced Micro Devices)*.

а б

Рис. 1.15. Оперативна пам’ять

Оперативна пам’ять (рис.1.15) *(RAM – Random Access Memory)* – це масив кристалічних комірок, які здатні зберігати дані. Існує багато різних типів оперативної пам’яті, але за фізичним принципом дії розрізняють *динамічну пам’ять (DRAM)* та *статичну пам’ять (SRAM)*. Мікросхеми динамічної пам’яті використовують як основну оперативну пам’ять комп’ютера, а статичної – як допоміжну пам’ять (*кеш-пам’яті*). Конструктивно оперативну пам’ять виготовляють у вигляді модулів, які вставляють у відповідні роз’єми на материнській платі. Модулі пам’яті розрізняють за *типом модуля, ємністю, частотою роботи* (швидкістю передачі даних) та *часом доступу до комірки пам’яті*. За цими показниками розрізняють такі модулі пам’яті:

SIMM (Single In-line Memory Module) – модуль пам’яті з однорядним розміщенням контактів, які широко застосовувалися в комп’ютерних системах 90-х років. Найбільш поширені 30- і 72-контактні модулі. SIMM модулі мали ємність: 4 Мб, 8 Мб, 16 Мб, 32 Мб. Час доступу у цих модулях становив 50-70 нс (наносекунд – мільярдна доля секунди).

DIMM (Dual In-line Memory Module) – модуль пам'яті DRAM, який прийшов на зміну SIMM. Основною відмінністю DIMM є те, що контакти

в ньому розміщені на обох сторонах модуля, тобто в два ряди. Вони є найбільш поширеними у вигляді 168-контактних модулів. Ці модулі виготовлялися з показниками ємності: 32 Мб, 64 Мб, 128 Мб, 256 Мб, 512 Мб та 1024 Мб. Час доступу в них складав 7-10 нс.

DDR SDRAM (Double Data Rate Synchronous Dynamic Random Access Memory) – тип оперативної пам'яті з подвоєною швидкістю передавання даних, який виконується у вигляді 184-контактного DIMM-модуля (рис. 1.15, а). Ці модулі працюють на частотах 100, 133, 200, 266, 333, 400 МГц. Ємності таких модулів можуть становити 128 Мб, 256 Мб, 512 Мб, 1024 Мб, 2048 Мб. Час доступу, характерний для них – 5-8 нс.

DDR2 SDRAM – виконується у вигляді 240-контактного DIMM-модуля (рис. 1.15, б). Основна відмінність DDR2 від DDR – здатність працювати на набагато більшій тактовій частоті – 800 МГц і вище. Ємності модулів DDR2 можуть становити: 256 Мб, 512 Мб, 1024 Мб, 2048 Мб. Час доступу в таких модулях становить 3-5 нс.

DDR3 SDRAM забезпечує зменшення споживання енергії на 40% порівняно з модулями DDR2 завдяки застосуванню 90-нм технології виробництва, що дозволяє понизити експлуатаційні показники струму і напруги (1,5 В, порівняно з 1,8 В для DDR2 і 2,5 В для DDR). Крім того, тактова частота таких модулів досягне 1600 МГц.

1.2.2. Портативні персональні комп'ютери

Залежно від розмірів, ваги, можливостей та призначення розрізняють декілька типів мобільних персональних комп'ютерів: ноутбуки, нетбуки, планшетні ПК, кишенькові ПК.

Лептоп (англ. laptop – lap = коліна сидячої людини) – більш широкий термін, його застосовують як до ноутбуків, так і планшетних ПК. До ноутбуків, зазвичай, відносять лептопи, виконані в розкладному

формфакторі. Ноутбук переносять у складеному вигляді, це дозволяє захистити екран, клавіатуру і тачпад під час транспортування.

Ноутбук (англ. *notebook* – блокнот, блокнотний ПК рис. 1.16) – портативний персональний комп'ютер, у корпусі якого об'єднано типові компоненти ПК, включаючи дисплей, клавіатуру і сенсорну панель або тачпад, а також акумуляторні батареї. Ноутбуки відрізняються невеликими розмірами і вагою, час автономної роботи ноутбуків змінюється в межах від 1 до 15 годин.

Рис. 1.16. Ноутбук

Ноутбук за своєю суттю є повноцінним комп'ютером. Але для забезпечення мобільності, портативності і енергонезалежності всі комплектуючі мають своєрідні особливості.

Клавіатура ноутбука виконана за спеціальною технологією і являє собою декілька шарів тонкого пластику з контактами, що дозволяє зменшити товщину до декількох міліметрів.

Корпус ноутбука, зазвичай, виконаний із високоміцного пластику. Всередині покритий спеціальною тонкою металевою фольгою для ізоляції електронної начинки від впливу зовнішніх електромагнітних полів. За периметром, зазвичай, виконаний металевий корд, що надає додаткову міцність корпусу.

Як вказівний пристрій у ноутбуках широко поширений так званий тачпад – сенсорна панель, що реагує на дотик пальця.

Матриця ноутбука являє собою повноцінний рідкокристалічний монітор. Всередині верхньої кришки ноутбука вміщено все, що необхідно для її повноцінної роботи – безпосередньо матриця, шлейфи, що передають дані, інвертор для забезпечення роботи лампи підсвічування і

деякі додаткові пристрої (наприклад: веб-камера, колонки, мікрофон, антени бездротових модулів Wi-Fi і Bluetooth).

Привід ноутбука позбавлений механіки, що висуває лоток, тому його вдалося зробити настільки тонким за збереження всіх функцій повноцінного приводу. Більшість сучасних приводів мають стандарт DVD-RW, проте у дорогих мультимедійних ноутбуках часто можна зустріти привід стандарту Blu-ray.

Оперативна пам'ять ноутбука завдяки більш високій щільності розміщення чіпів за меншого розміру має характеристики, які можна порівняти з пам'яттю звичайного комп'ютера.

Система охолодження ноутбука складається з кулера, який забирає повітря з вентиляційних отворів на днищі ноутбука (саме тому ноутбук можна використовувати тільки на твердій рівній поверхні, інакше порушується охолодження) і продуває його через радіатор, який мідним тепловідводом з'єднаний з процесором, а іноді чіпсетом материнської плати.

Процесор ноутбука за зовнішнім виглядом і розмірами дуже схожий на процесор звичайного комп'ютера, однак усередині нього реалізовано велику кількість технологій, що знижують енергоспоживання і тепловиділення.

Жорсткий диск ноутбука, не дивлячись на маленький розмір (завдяки використанню магнітних носіїв діаметром 2,5 дюйма), має обсяг, який можна порівняти з обсягом жорсткого диска для стаціонарного комп'ютера. Найбільш розповсюджений інтерфейс підключення SATA, проте ще досить часто можна зустріти інтерфейс IDE, особливо в старих ноутбуках. Нещодавно з'явилися так звані твердотільні жорсткі диски (SSD), розроблені на основі flash-пам'яті.

Класифікація ноутбуків на основі призначення і технічних характеристик пристрою: бюджетні ноутбуки, ноутбуки середнього класу, бізнес-ноутбуки, мультимедійні ноутбуки, ігрові ноутбуки, мобільна робоча станція, іміджеві ноутбуки, захищені ноутбуки, ноутбуки із сенсорним дисплеєм.

Нетбук (англ. *netbook* рис.1.17) – невеликий мобільний комп'ютер (ноутбук), основне призначення якого полягає в доступі до мережі *Інтернет* та роботі з офісними програмами.

Рис. 1.17. Нетбук

Нетбуки як окрему категорію ноутбуків було виділено з категорії субноутбуків в першому кварталі 2008 року компанією Intel. Розмір діагоналі нетбуків від 7 до 12,1 дюйма. Нетбуки орієнтовані на перегляд веб-сторінок, роботу з електронною поштою та офісними програмами. Для цих ноутбуків розроблено спеціальні енергоефективні процесори. Малий розмір екрана, невелика клавіатура й низька продуктивність подібних пристроїв компенсується помірною ціною і відносно великим часом автономної роботи. Габарити, зазвичай, не дозволяють установити в нетбуку дисковод оптичних дисків, однак Wi-Fi-адаптер є обов'язковим компонентом.

Планшетний комп'ютер (англ. *tablet PC* рис. 1.18) – клас ноутбуків, обладнаних планшетним пристроєм рукописного введення, об'єднаним з екраном. Планшетний комп'ютер дозволяє працювати за допомогою спеціального пера, стилуса або пальців, без використання клавіатури і миші.

Користувач може вводити текст, використовуючи вбудовану програму розпізнавання рукописного введення, екранну (віртуальну) клавіатуру, розпізнавання мови, або звичайну клавіатуру (якщо вона входить до складу пристрою).

Рис. 1.18. Планшетний компютер

Конструктивно планшетні комп'ютери ділять на дві великі категорії: перша – планшети, тобто корпуси невеликої товщини, в які вмонтовано сенсорні дисплеї; друга – клавіатурні планшетники, що є повноцінними клавіатурними ноутбуками, в яких дисплей може повертатися і закривати собою клавіатуру.

Машини першої категорії набагато легші за комп'ютери другої категорії, і вони призначені саме під постійне носіння в одній руці і рукописне введення даних. Клавіатурні планшетники – важчі, проте до їх безперечних переваг належить можливість роботи як в «планшетному» режимі, так і режимі звичайного ноутбука. До планшетника теж можна підключити клавіатуру для стаціонарної роботи, проте, у клавіатурних планшетників вона завжди поруч, хоча це і призводить до збільшення ваги.

Рис. 1.19. Кишеньковий комп'ютер

Кишеньковий персональний комп'ютер (англ. *Pocket PC – Кишеньковий* ПК рис. 1.19), а також: ручний комп'ютер, надолонний комп'ютер, надолонник, палмтоп) – збірна назва класу портативних електронних обчислювальних пристроїв, спочатку запропонованих до використання як електронні органайзери.

Кишеньковий ПК складається з процесора, пам'яті, звукової і відеосистем, екрана, слотів розширення, за допомогою яких йому можна додати пам'яті або можливостей, та клавіатури.

Кишеньковий комп’ютер призначений для: читання книг, довідкових текстів, словників, енциклопедій тощо; перегляду електронної пошти, веб-сторінок, журналів та інших документів у різних текстових форматах; перегляду карт місцевості; запису щоденника і розкладу; прослуховування радіопередач; звукового запису та від руки; набору тексту; перегляду відеороликів та зображень; виходу в Інтернет; дистанційного керування та програмування; оснащений фотоапаратом та відеокамерою; інколи використовується як мобільний телефон.

До КПК, оснащеного хост-контролером *USB*, можна безпосередньо підключати різні пристрої *USB*, зокрема клавіатуру, мишу, тверді диски і флеш-накопичувачі.

Останнім часом набули великого поширення комунікатори і смартфони, які стали витісняти КПК, вони суміщають функції КПК з функціями мобільного телефону. Ці пристрої мають практично ідентичні звичайним КПК операційні системи з незначними відмінностями – додатковим програмним забезпеченням для роботи з мобільним зв'язком. Комунікатори і смартфони мають помітну перевагу за рахунок додаткової і потрібної функціональності спілкування. Смартфони поступаються лише в короткому часі автономної роботи.

1.2.3. Нестандартні конструкції персональних комп’ютерів

Залежно від конструктивних рішень, навколишніх умов використання та призначення розрізняють декілька типів нестандартних конструкцій персональних комп’ютерів: баребони, промислові ПК, мультипроцесорні ПК.

Баребон (англ. *Barebone PC – Базовий ПК* рис. 1.20.) – напівзібраний комп'ютер, що складається з корпусу з блоком живлення, материнської плати і системи охолодження. Зазвичай ці комп'ютери виготовляють для нестандартних рішень і можуть бути зібрані з різноманітних комплектуючих. Користувачу залишається лише доповнити систему процесором, оперативною пам'яттю, жорстким диском і пристроями вводу-виводу залежно від своїх потреб.

Рис. 1.20. Баребон

Промисловий персональний комп'ютер (англ. *Industrial PC ПК* рис. 1.21) – комп'ютер з архітектурою x86, сумісний з більшістю програмного і апаратного забезпечення персонального ПК, призначений для роботи з програмним забезпеченням для промисловості. *Промислові ПК* відрізняються від звичайних конструкцією, яка враховує вимоги до обладнання, що працює в несприятливих умовах, таких, як підвищена вібрація, забруднене повітря, підвищена вологість, підвищені або знижені температури.

Рис. 1.21. Промисловий персональний комп'ютер

Застосовується в промисловості у складі управляючих, контролюючих і вимірювальних комплексів. Як апаратна платформа для забезпечення систем із функціями взаємодії з оператором, що називають HMI/SCADA (Human Machine Interlace). Поняття SCADA або «SCADA –

система» (Supervisory Control And Data Acquisition System) – система збору даних і оперативного диспетчерського управління. SCADA системи – це сукупність апаратно-програмних засобів, що забезпечують можливість моніторингу, аналізу та управління параметрами технологічного процесу людиною. Вона є складовою частиною системи управління.

Основні відмінності промислових комп'ютерів від офісних: підвищений механічний захист від ударів і вібрацій та потрапляння пилу; розширений температурний діапазон роботи; промислові комп'ютери мають нормовані показники напрацювання на відмову; стійкі до електромагнітних хвиль, вони не перевантажуються і не дають збою в разі електростатичного розряду; їх легко обслуговувати порівняно з офісними; промислові монітори додатково мають ступінь захисту передньої панелі; корпус має спеціальні засоби, здатні вимірювати температуру усередині корпусу, струм і швидкість обертання охолоджувальних вентиляторів і сигналізувати різними способами про несправність; блоки живлення промислових комп'ютерів можуть мати вхідні напруги не тільки 220В, але і 24, і 48 В

Мультипроцесорний персональний комп'ютер має декілька процесорів, кожен з яких може відносно незалежно від інших виконувати свою програму. У мультипроцесорі існує загальна для всіх процесорів операційна система, яка оперативно розподіляє обчислювальне навантаження між процесорами. Взаємодія між окремими процесорами організовується найбільш простим способом – через загальну оперативну пам'ять.

Процесорний блок не є закінченим комп'ютером, отже, не може виконувати програми без решти блоків мультипроцесорного комп'ютера – пам'яті і периферійних пристроїв. Всі периферійні пристрої є для всіх процесорів мультипроцесорної системи загальними.

1.2.4. Периферійні пристрої та зовнішні носії інформації

Периферійні пристрої комп'ютерної системи забезпечують виконання низки задач, які можна розділити на три категорії: виведення, введення та обмін даними. Відповідно до такої класифікації поділяються і периферійні пристрої [11].

Пристрої виведення даних. Враховуючи ту обставину, що сучасні комп'ютерні системи здатні працювати з усіма видами інформації (вони є мультимедійні), доступні для сприйняття людиною, перелік пристроїв виводу інформації досить різноманітний. До нього можна віднести такі пристрої: *монітори, проектори, принтери, плотери, акустичні системи.*

Монітор (дисплей) – це пристрій візуального представлення даних, який входить до базової конфігурації персонального комп'ютера. За принципом формування зображення на сьогодні поширені такі види моніторів: *з електронно-променевою трубкою, рідинно-кристалічні* та *плазмові.*

Рис. 1.22. Монітор з електронно-променевою трубкою

Монітори з електронно-променевою трубкою (CRT-монітор (Cathode Ray Tube)). CRT-монітор (рис. 1.22) в основі своєї будови має скляну трубку, всередині якої утворено вакуум. З фронтальної сторони внутрішня частина трубки покрита люмінофором – речовиною, яка випромінює світло під час бомбарду-вання її зарядженими частинками. Як люмінофор для кольорових *ЕПТ* використовують досить складні речовини на основі рідкоземельних металів – ітрію, ербію тощо. Для створення зображення у CRT-моніторі використовується три електронні гармати, які випускають потік електронів на внутрішню поверхню скляного екрану монітора. Для

отримання кольорового зображення люмінофорне покриття має частинки трьох типів, які здатні світитися червоним, зеленим та синім кольором. Кожна з трьох гармат відповідає за формування цих кольорів і посилає потік електронів на різні частинки люмінофору, які засвічуються основними кольорами з різною інтенсивністю. В результаті їх комбінування формується зображення з необхідним кольором. Для того, щоб на екрані всі три промені сходилися в одну точку і зображення було чітким, перед люмінофором ставлять *маску* – панель з рівномірно розміщеними отворами. Чим менша відстань між отворами (*крок маски*), тим чіткіше зображення на екрані.

Основними характеристиками CRT-моніторів є: *розмір екрана, максимальна частота регенерації зображення, крок маски, клас захисту.*

Розмір екрана вимірюється в дюймах між протилежними кутами трубки кінескопа за діагоналлю. Стандартними розмірами екранів є: 14", 15", 17", 19", 20", 21". До того ж слід враховувати ту обставину, що видимий розмір таких типів моніторів дещо менший від розміру діагоналі. Наприклад, для монітора з розміром діагоналі 19" видимий розмір становить 17,8".

Частота регенерації зображення показує, скільки разів протягом секунди монітор може цілком змінити зображення. Ця величина вимірюється в герцах (*Гц*). Чим вище цей показник, тим чіткіше і стійкіше зображення на моніторі, що сприяє меншій втомлюваності очей під час роботи. Сучасні монітори здатні забезпечувати роботу в діапазоні від 50 до 160 Гц. Мінімальною частотою, з точки зору безпеки для очей, вважається частота 75 Гц. Проте, в характеристиках моніторів показник частоти регенерації зображення часто наводять у поєднанні із *роздільною здатністю екрана,* яка вказує з якої кількості точок (*пікселів*) буде сформоване зображення. Наприклад, запис вигляду 1600x1200 @ 85 Гц означає, що монітор з такою характеристикою здатний 85 разів за секунду

змінити зображення, утворене із 1200 рядків, у кожному з яких по 1600 точок. Проте, якщо в такому моніторі понизити показник роздільної здатності до одного із стандартних (800×600, 1024×768, 1280×1024), то можна суттєво підвищити частоту регенерації зображення.

Крок маски, як уже зазначалося, показник, який впливає на чіткість зображення. Вимірюється у сотих частках міліметра. Його значення можуть знаходитися в межах від 0,2 до 0,3 мм.

Клас захисту монітора визначається стандартом, якому відповідає монітор з точки зору безпечної роботи. Загальновизнаними вважаються такі міжнародні стандарти: *MPR II, MPR III, TCO'92, TCO'95, TCO'99, TCO'03* (цифра вказує рік прийняття стандарту).

Основними недоліками CRT-моніторів, що стали причинами скорочення їх використання, є: значні габарити, високі рівні електромагнітних випромінювань та енергоспоживання.

Рідинно-кристалічні монітори (LCD-монітори (Liquid Crystal Display) (рис. 1.23). Основою для виготовлення таких моніторів є речовина, яка знаходиться в рідкому стані, але при цьому володіє деякими властивостями, що характерні кристалічним тілам. Молекули рідких кристалів під впливом електричної напруги можуть змінювати свою орієнтацію в просторі і, внаслідок цього, змінювати властивості світлового променя, який проходить крізь них. Перше своє застосування рідкі кристали знайшли в дисплеях калькуляторів і годинників, а потім їх стали використовувати для створення моніторів ноутбуків, пізніше – настільних комп'ютерів.

Рис. 1.23. Рідинно-кристалічний монітор

Рідинно-кристалічні дисплеї поділяться на два класи за принципом

формування зображення: з *пасивною* і *активною матрицею*. *Матриця* – це масив дрібних сегментів (пікселів), які утворюються за рахунок переорієнтації окремих кристалів під дією електричної напруги. Для управління окремо кожним сегментом, на площині, заповненій шаром рідкокристалічного матеріалу, розміщують велику кількість електродів, які управляють окремо кожним сегментом. Виготовлення якісних моніторів з *пасивною* матрицею практично неможливо, оскільки сама технологія має низку суттєвих недоліків, основними з яких є низька швидкість формування та зміни зображення на екрані. У зв'язку з цим виготовлення сучасних LCD-моніторів передбачає застосування *активних* матриць. Активна матриця побудована на тонкоплівкових транзисторах – *TFT* (*Thin Film Transistor*). *TFT-технологія* виготовлення LCD-моніторів передбачає таку технологію формування зображення на екрані: світло від неонової лампи підсвічування, яка розташована в задній частині екрана, проходить через систему відбивачів і фільтрів. Після цього світло потрапляє на шар рідких кристалів, де кожним окремо взятим пікселем керує транзистор. Потім світло проходить через спеціальні фільтри, які формують на виході три кольори червоний, зелений, синій (система кольору RGB). Керуючий транзистор регулює електричне поле, що визначає просторову орієнтацію рідких кристалів (кут повороту кристалу). Зміна кута повороту кристала впливає на інтенсивність пропущеного ним світла. Завдяки цьому, співвідношення інтенсивності пропущених трьох основних кольорів у кожній точці екрана різне, що забезпечує формування різних кольорових відтінків. Необхідно відзначити також, що яскравість окремого елемента зображення (пікселя) на LCD-моніторі залишається сталою впродовж усього інтервалу часу зміни зображення, а не являє собою короткий спалах, як у CRT-моніторів. Саме через це для LCD-моніторів достатньою частотою регенерації зображення є 60 Гц.

Основними характеристиками матриць є: *контрастність, кут огляду, час відгуку.*

Контрастність – це співвідношення між максимальною і мінімальною яскравістю. У сучасних матрицях цей показник досяг значення 700:1.

Кут огляду – це кут, оглядаючи з якого, контрастність зображення падає в 10 разів. Розрізняють горизонтальний та вертикальний кут огляду. Цей показник досягає 170^o і більше.

Час відгуку – параметр, що визначає час, за який транзистор встигає змінити просторову орієнтацію рідких кристалів. Іноді цей параметр трактується як час включення пікселя, час виключення пікселя, час переходу від одного крайнього положення до іншого, середнє значення між часом включення і виключення. Час відгуку для різних типів матриць може мати 25 мс, 16мс, 12 мс, 8 мс, 4 мс, 2 мс.

Основними характеристиками LCD-моніторів є:

- *Розмір діагоналі* – від 15" до 46". Найбільш поширені розміри: 17", 19", 20", 21", 22". Розмір діагоналі відповідає видимому розміру.
- *Максимальна яскравість* – від 200 кд/м^2 до 450 кд/м^2.
- *Контрастність* – від 350:1 до 700:1.
- *Кут огляду* – горизонтальний: від 150^o до 178^o, вертикальний: від 130^0 до 178^0.
- *Час відгуку* – від 25 мс до 2 мс.
- *Роздільна здатність* – на відміну від CRT-моніторів, LCD-панелі розраховані на роботу з одним показником роздільної здатності, який визначається фізичною кількістю пікселів на екрані. Для 15-дюймових моніторів це 1024×768, для 17 та 19-дюймових – 1280×1024.
- *Розмір пікселя* – від 0,2 мм до 0,3 мм.

Перевагами LCD-моніторів, які визначили їх як стандарт, під час вибору пристрою візуального представлення даних, є: малий об’єм та

вага, відсутність електромагнітного випромінювання, порівняно висока економічність, відсутність мерехтіння, велика видима зона екрана.

Плазмові монітори (PDP-монітори (Plasma Display Panels) (рис. 1.24). Робота плазмових моніторів дуже схожа на роботу неонових ламп, які зроблені у вигляді трубки, заповненої інертним газом низького тиску. Плазмові екрани створюються шляхом заповнення простору між двома скляними поверхнями, вкритими шаром люмінофору, інертним газом, наприклад, аргоном або неоном. Потім на скляній поверхні розміщують маленькі прозорі електроди, на які подається високочастотна напруга. Під дією цієї напруги, в прилеглій до електроду газовій області, виникає електричний розряд. Плазма газового розряду випромінює світло в ультрафіолетовому діапазоні, яке спричиняє світіння частинок люмінофора у видимому людиною діапазоні. Фактично, кожний піксель на екрані працює як звичайна флуоресцентна лампа (лампа денного світла). Висока яскравість і контрастність разом з відсутністю мерехтіння зображення є основними перевагами таких моніторів. Головними недоліками такого типу моніторів є досить висока споживана потужність, що зростає в разі збільшення діагоналі монітора, і низька роздільна здатність, зумовлена великим розміром елемента зображення. Такі монітори використовують здебільшого для конференцій, презентацій, інформаційних щитів, тобто там, де потрібні великі розміри екранів для відображення інформації.

Рис. 1.24. Плазмовий монітор

Основними характеристиками плазмових моніторів є:

- *Розмір діагоналі* – від 37" до 102".
- *Максимальна яскравість* – від 500 кд/м^2 до 1500 кд/м^2.
- *Контрастність* – від 1000:1 до 10000:1 і вище.

- *Кут огляду* – до 170^0.
- *Розподільна здатність* – 852×480, 800×600, 1024×1080, 1280×768, 1366×768.
- *Розмір пікселя* – від 0,2 мм до 1,3 мм.

Принтер – пристрій виведення цифрової інформації (символьної та графічної) на твердий носій (папір, прозорі плівки, пластикові картки тощо). Принтери класифікують за низкою показників: *технологією формування зображення* (матричні, струминні, лазерні, термосублімаційні тощо), *видом формування зображення* (монохромні та повнокольорові), *механічною дією* (ударні та безударні).

Основними характеристиками принтерів є: *роздільна здатність, швидкість друку, ємність внутрішньої пам'яті, ресурс картриджа або собівартість друку 1 сторінки.*

Роздільна здатність вказує, з якої кількості точок на дюйм буде сформоване зображення на твердому носії. Вимірюється цей показник у *dpi* (*dots per inch – точок на дюйм*), зазначається двома цифрами – горизонтальної та вертикальної роздільної здатності. Наприклад, для лазерних принтерів він може мати значення: 600×600 dpi, 1200×1200 dpi, для струминних – 2400×1200 dpi, 4800×1200 dpi і вище.

Швидкість друку визначає, яку кількість сторінок здатний вивести принтер на друк за одну хвилину. Вимірюється цей показник у *сторінках за хвилину (ст/хв, ppm – pages per minute)*. Для різних типів принтерів цей показник різний, зокрема для струминних принтерів від знаходиться в межах від 3 до 20 ст/хв, для лазерних від 6 до 35 ст/хв і вище.

Ємність внутрішньої пам'яті визначає якість і швидкість друку. Оскільки інформація, яка призначена для виведення на друк, передається з комп'ютера у власну пам'ять принтера, то ємність пам'яті принтера визначає, який обсяг інформації зможе вміститися у ній. Ємність

внутрішньої пам’яті лазерних принтерів може бути 2 Мб, 8 Мб, 16 Мб, 32 Мб, 64 Мб і більше. Для струминних принтерів типовими є значення 12 Кб, 32 Кб.

Ресурс картриджа визначає, яку кількість сторінок, з певним відсотком заповнення сторінки (наприклад 5%), можна вивести на друк на ресурсі одного картриджа. Для лазерних принтерів цей показник становить 2-5 тисяч сторінок. Для струминних він значно нижчий. Інколи користуються показником «*собівартість друку 1 сторінки»*, який визначається відношенням вартості картриджа до ресурсу картриджа.

Матричні принтери (рис.1.25.) Це найстаріший з нині вживаних типів принтерів, механізм дії якого був винайдений в 1964 році корпорацією *Seiko Epson*. Технологія друку матричними принтерами полягає у формуванні зображення друкувальною головкою, що складається з набору голок, сформованих у матрицю, які приводяться в дію електромагнітами. Головка переміщується рядок за рядком уздовж аркуша, при цьому голки ударяють по паперу через фарбувальну стрічку, формуючи зображення з точок. Випускалися принтери з 9, 12, 14, 18 і 24 голками у головці. Якість і швидкість друку таких принтерів залежить від кількості голок, чим більше голок, тим вища швидкість і якість друку. Швидкість друку матричних принтерів вимірюється в *cps (characters per second – символів у секунду).*

Рис. 1.25. Матричний принтер

Матричні принтери на сьогодні використовуються для спеціалізованих напрямів друку, зокрема: для друку на безперервний рулон паперу; в банках, бухгалтеріях, торговельних закладах для друку фінансових документів (чеків, бланків, квитків тощо), а також, коли важливий сам факт друку ударним способом. Вважається, що друк

пристроями ударної дії ускладнює внесення несанкціонованих змін у фінансовий документ.

Основними недоліками матричних принтерів є: монохромність, низька швидкість друку та високий рівень шуму.

Струминні принтери (рис. 1.26). У струминних друкувальних пристроях зображення на папері формується з плям, що утворюються в разі потрапляння мікрокрапель барвника на папір. Барвники для таких пристроїв перебувають у рідкому стані та розміщуються у картриджах, окремо чорний та кольорові. Викид мікрокрапель барвника на папір відбувається під тиском через відповідні отвори (*сопла*) друкувальної головки, які утворюють матрицю із сотень таких елементів. Тиск у друкувальній головці, необхідний для викиду барвника, може створюватися термічним способом – за рахунок різкого нагрівання барвника до 500^{o}С або під дією деформації п'єзокристалів. Розмір краплі барвника при цьому становить близько 5 пл (*піколітри*).

Рис. 1.26. Струминний принтер

Основною конструктивною відмінністю різних моделей струминних принтерів є те, що в деяких із них (принтери фірм *Hewlett-Packard, Lexmark)* друкувальна головка вбудована у картридж з барвником, а в інших (принтери фірм *Epson, Canon)* – є деталлю принтера. Ця обставина важлива з огляду на те, що сопло друкувальної головки має діаметр всього близько 3 мікрон. А, враховуючи те, що барвник, який перебуває в картриджі у рідкому стані, має властивість засихати і утворювати при цьому тверді механічні частинки, може виникати засмічення сопел, відновлення працездатності яких у моделях з незмінними головками може бути трудомістким та дорогим.

До основних переваг, які забезпечили струминним принтерам

лідируючі позиції у виробничій та побутовій сферах, можна віднести низьку вартість пристроїв, можливість кольорового друку з високою роздільною здатністю та якістю передання кольорів. Однак таким пристроям притаманні й серйозні недоліки, зокрема підвищені вимоги до якості паперу, низький ресурс картриджів та, відповідно, висока собівартість друку, низька швидкість друку кольорових зображень, можливість засмічення сопел тощо, що в деяких випадках створюють серйозні обмеження на область їх застосування.

Лазерні принтери (рис.1.27). Головною частиною друкувального пристрою лазерного принтера є металевий барабан, що обертається, робоча поверхня якого виготовлена із напівпровідникового матеріалу. У процесі друку поверхня барабана електризується від високовольтного джерела і отримує електростатичний заряд.

Рис. 1.27. Лазерний принтер

Після цього в потрібних місцях барабан опромінюється лазерним променем, який знімає заряд. Таким чином, на поверхні барабана формується приховане зображення у вигляді наелектризованих ділянок. До тих місць барабана, що зберегли електростатичний заряд, прилипають найдрібніші частинки фарби (тонера), які переносяться на папір. Далі лист з готовим зображенням проходить через нагрівальний елемент, під дією якого тонер плавиться і вбирається папером. Після завершення циклу друку барабан до кінця розряджається та очищається від частинок тонера, що залишилися.

Під час формування зображення лазерним принтером важливо, щоб інформація про всю сторінку, що друкується, вміщувалася в пам'яті принтера, інакше отримати якісне зображення досить складно.

На сьогодні лазерні пристрої є одними з найпопулярніших для виведення на друк чорно-білих зображень завдяки високим показникам швидкості та якості друку, надійності, низькій собівартості друку. Лазерні принтери, з можливістю кольорового друку, на сьогодні мають високі показники вартості як пристроїв, так і матеріалів для друку.

Рис. 1.28. Термосублімаційний принтер

Термосублімаційні принтери (рис.1.28). Принцип дії таких принтерів ґрунтується на ефекті термосублімації, який полягає у перетворенні спеціального барвника із твердого стану одразу в газоподібний за рахунок стрімкого нагрівання, тобто, коли відсутня рідка фаза. Барвник кожного з основних кольорів, а їх може бути три або чотири, знаходиться на окремій (або на загальній багатошаровій) тонкій поліефірній стрічці. Кожна стрічка послідовно протягується під щільно притиснутою до неї термоголовкою, що складається з великої кількості термоелементів. Ці термоелементи, нагріваючись, перетворюють барвник у пару, яка завдяки малій відстані між головкою і листом паперу, стабільно позиціонується і утворює точки досить малого розміру. Утворення остаточного зображення відбувається в декілька проходів. У разі термосублімаційної технології друку сусідні пікселі частково перекриваються, що створює ефект безперервності зображення, як на аналоговому кольоровому фотопапері. На відміну від цього, струминна і лазерна технології друку формують зображення із точок з різкими межами, без перекриття.

Пристрої введення даних. Ця група периферійних пристроїв значною мірою визначає ефективність всієї комп’ютерної системи щодо обробки даних, оскільки, зазвичай, процес введення даних повільніший, ніж їх обробка. Тому вибір та ефективне використання пристроїв уведення є важливим фактором автоматизації обробки інформації. Всі

пристрої введення можна розділити на декілька категорій:

- *пристрої введення знакових даних (клавіатури);*
- *пристрої командного управління (миші, трекболи, пенмауси, ігрові маніпулятори);*
- *пристрої введення графічних даних (сканери, графічні планшети (диджитайзери), цифрові фото- та відеокамери).*

Сканер – пристрій введення інформації, який здатний перетворити зображення на твердому носії в цифрову копію. Принцип дії цих пристроїв полягає в тому, що промінь світла, утворений спеціальною лампою, спрямовується на документ, що розміщений на твердому носії. Цей промінь, відображений від поверхні матеріалу, через систему дзеркал потрапляє на світлочутливу матрицю *CCD (Couple-Charged Device),* яка перетворює світло у цифровий сигнал. Зображення сканується рядок за рядком. Чим більша кількість елементів розміщена у матриці, тим вищу якість сканування забезпечує пристрій.

Рис. 1.29. Сканер

Залежно від способу та об’єктів сканування розрізняють такі види сканерів:

Планшетні (рис. 1.29) – найпоширеніший вид сканерів, які забезпечують максимальну зручність для користувача – високу якість і достатню швидкість сканування. Виготовляють їх у вигляді планшета, всередині якого під прозорим склом розміщений механізм сканування.

У ручних сканерів відсутній механізм переміщення. Тобто, сканер над об’єктом сканування переміщується користувачем вручну.

Листопротяжні – характерною особливістю їх є те, що вони дозволяють сканувати лише матеріал, який розміщений на окремих аркушах паперу. При цьому аркуш вставляється у відповідну щілину і протягується повз систему сканування.

Планетарні – застосовуються для сканування книг або документів, що легко ушкоджуються. Під час сканування у них відсутній контакт з об'єктом сканування.

Барабанні – застосовуються здебільшого у поліграфії, оскільки мають найвищу якість сканування (розрізняють до 10 тисяч точок на дюйм). У сканерах цього типу початковий матеріал закріплюється на циліндричній поверхні барабана, що обертається з високою швидкістю.

Слайд-сканери застосовуються для сканування зображень, які розміщені на позитивних та негативних фотоплівках. Такі сканери випускаються як самостійні пристрої або у вигляді додаткових модулів до планшетних сканерів.

Сканери штрих-коду – вид сканерів, які виготовляють у вигляді ручних сканерів або розміщених у стаціонарних пристроях. Вони призначені для сканування штрих-кодів.

Основними характеристиками сканерів є: *роздільна здатність, швидкість роботи, глибина кольору, динамічний діапазон.*

Роздільна здатність – основна характеристика сканера, яка визначає кількість точок на одиницю довжини зображення, про які сканер здатний зберегти інформацію у цифровому вигляді. Цей показник вимірюється у *dpi (dots per inch - кількість точок на дюйм)* і залежить від щільності розміщення елементів у світлочутливій матриці, а також від точності механічного позиціонування лінійки під час сканування. Кількість точок, отримана оптичною системою сканера в процесі роботи, називається *оптичною* або *фізичною* роздільною здатністю. Для сканерів зазначають горизонтальну та вертикальну роздільну здатність, яка може мати значення: 600×1200 dpi, 1200×2400 dpi, 2400×4800 dpi. Однак часто в характеристиках зазначають *інтерполяційну роздільну здатність* – штучний показник роздільної здатності, отриманий за допомогою математичної обробки зображення, який може досягати 24000 dpi.

Швидкість роботи – під цим показником розуміють не швидкість обробки однієї сторінки (як у принтерах), а швидкість сканування однієї лінії зображення (у мілісекундах). Цей показник може знаходитися у межах від 30 мс до 5 мс.

Глибина кольору визначає, яку кількість відтінків здатний розпізнати сканер. Бувають 24, 30, 36 та 48 бітні сканери (24 біт відповідає 16 777 216 відтінкам).

Динамічний діапазон визначається логарифмом відношення яскравості найбільш світлих ділянок зображення до яскравості найбільш темних ділянок. Типовий показник для сканерів офісного застосування становить 1,8-2,0, а для сканерів професійного застосування – від 2,5 (для непрозорих матеріалів) до 3,5 (для прозорих матеріалів).

Пристрої обміну даними між віддаленими ПК каналами зв’язку. Головним представником цієї групи є *модем*.

Модем (*МОдулятор+ДЕМодулятор*) (рис. 1.30) – пристрій, призначений для обміну інформацією між віддаленими комп'ютерами каналами зв'язку. При цьому під каналом зв'язку розуміють фізичні лінії (дротяні, оптоволоконні, радіочастотні), спосіб їх використання (комутовані, виділені) і спосіб передавання даних (цифрові або аналогові сигнали) [11]. Залежно від типу каналу зв'язку, пристрою приймання-передавання модеми поділяють на:

Рис. 1.30. Модем

- *Аналогові* – найпоширеніший тип модемів для звичайних комутованих телефонних ліній.
- *ISDN (Integrated Services Digital Network – цифрова мережа з інтеграцією служб)* – модеми для цифрових комутованих телефонних ліній.

- *DSL (Digital Subscriber Loop - цифрова абонентська лінія)* – модеми, що використовуються для побудови виділених (некомутованих) ліній, застосовуючи звичайну телефонну мережу. Дозволяють одночасно з обміном даними використовувати телефонну лінію безпосередньо за призначенням.

- *Кабельні* використовують для обміну даними спеціалізованими кабелями, наприклад, кабелями систем колективного телебачення.

- *Радіо* – це модем, що використовує мережу операторів мобільного зв'язку для передавання і приймання інформації. Під час використання в модем вставляють SIM-карту.

- *PLC (Power line communication – енергетичні комунікаційні лінії)* – використовують технологію передавання даних дротами побутової електричної мережі 220 В.

До основних параметрів модемів відносять: *продуктивність, протоколи зв'язку та коригування помилок, що підтримуються.*

Продуктивність визначає, з якою швидкістю модем здійснить обмін даними мережею. Вимірюється у *біт/с (біт за секунду)* або *bps (bits per second)*. Продуктивність модемів може знаходитися у межах від 300 біт/с до 54 Кбіт/с.

Протоколи зв'язку та коригування помилок, що підтримуються, – показник, який визначає ефективність взаємодії модему з суміжними модемами (вірогідність того, що вони вступлять у взаємодію один з одним за оптимального настроєння).

Існують також модеми з додатковими можливостями.

- *Факс-модем* дозволяє комп'ютеру, до якого він приєднаний, приймати і передавати факсиміле зображення на інший факс-модем або звичайний факс.

- *Голосовий модем* має функцію оцифрування сигналу з телефонної лінії і можливість передачі довільного звуку через телефонну лінію на інший віддалений голосовий модем.

Зовнішні носії інформації. Необхідність у зовнішніх носіях інформації виникає у декількох випадках:

- коли необхідно перенести дані з одного комп'ютера на інший;
- коли дані мають підвищену цінність і необхідно виконувати резервне копіювання на зовнішній носій (копіювання даних у межах розділів на жорсткому диску не є резервним);
- коли обчислювальною системою обробляється даних більше, ніж можна розмістити на базовому жорсткому диску.

Для забезпечення перерахованих вище потреб використовують такі типи зовнішніх носіїв: *магнітні, оптичні, магнітооптичні та Flesh-носії*. Основними характеристиками будь-якого типу носія є: *ємність, швидкість читання/запису інформації, вартість одиниці об'єму*.

Рис. 1.31. Гнучкий магнітний диск

Магнітні носії. Носії такого типу набули значного поширення на початковому етапі розвитку персональних комп'ютерів. Головною їх особливістю є спосіб читання та запису інформації, який базується на використанні магнітного поля. Носій, який може бути виконаний у вигляді диска або стрічки, покривається феромагнітним матеріалом. Засобами магнітного поля відбувається відповідне намагнічування певних ділянок носія, що і забезпечує збереження інформації у вигляді логічної 1 в зоні намагніченої ділянки та логічного 0 – у не намагніченій ділянці. Типовими представниками магнітних носіїв є *гнучкі магнітні диски* (рис. 1.31) (floppy disk, дискети) розміром 3,5 дюйма, *магнітні диски*

підвищеної щільності ZIP та *JAZ*, *касети з магнітними стрічками для стримерів.*

Гнучкі магнітні диски 3.5", зазвичай, мають ємність 1,44 Мб. Швидкість передавання даних для цього носія становить 62 Кб/с. На сьогодні гнучкі диски поступово витісняються, оскільки вони не задовольняють вимогам щодо обсягів і надійності зберігання даних, а також швидкості доступу до них.

ZIP-накопичувачі (рис. 1.32а) випускаються компанією Iomega з 1995 року. Ці носії значно перевищують за обсягом зберігання даних стандартні гнучкі диски і можуть мати такі показники ємності: 100 Мб, 250 Мб, 750 Мб. Швидкість передавання даних для них знаходиться в межах 2 Мб/с. ZIP-носії набули популярності, коли оптичні диски з функцією запису інформації та відповідні пристрої коштували досить дорого. Проте розвиток технологій та здешевлення виготовлення оптичних дисків сповільнили темпи застосування таких носіїв, вартість яких залишилася досить суттєвою. *Накопичувачі JAZ* (рис. 1.32б) були випущені компанією Iomega зразу після ZIP-накопичувачів. За своїми характеристиками JAZ-носій, у свій час, наближався до жорстких дисків, але на відміну від них був змінним, що давало змогу оперативно передавати значні обсяги інформації. Залежно від моделі накопичувача на одному диску можна розмістити 1 або 2 Гб даних. Швидкість передачі даних для них становить близько 8 Мб/с. Сучасне застосування таких носіїв обмежено з аналогічних причин як і ZIP-накопичувачів.


а)

б)

Рис. 1.32. Накопичувачі:
а – ZIP; б – JAZ.

Рис. 1.33. Стример

Одним із найстаріших магнітних накопичувачів є *стример* (рис. 1.33) – це накопичувачі на магнітній стрічці. Їх відрізняє порівняно низька ціна за значного обсягу збереженої інформації. Найчастіше їх застосовують для резервного копіювання даних. До недоліків стримерів відносять малу продуктивність (від 2 до 9 Мб/хв, що пов'язано передусім з тим, що цей пристрій послідовного доступу) і має недостатню надійність (крім впливу електромагнітних полів, стрічки стримерів отримують підвищені механічні навантаження і можуть фізично виходити з ладу). Ємність магнітних касет (картриджів) для стримерів перших поколінь становила декілька сотень Мбайт. Сучасні носії забезпечують зберігання до 1,6 Тб (*Терабайт*) інформації.

Оптичні носії. Для вирішення задач, зумовлених застосуванням зовнішніх носіїв, за постійного збільшення обсягу інформації, що обробляється комп’ютерними системами, замість магнітних почали застосовувати оптичні носії.

Першими представниками таких носіїв стали диски *CD-ROM (Compact Disk Read Only Memory) – ком пакт-диск тільки для читання пам’яті* (рис. 1.34). Конструктивно такі диски виготовляють із полікарбоната, який покритий надтонким шаром алюмінію. Інформацію на такий диск записують у вигляді насічок (впадин), розміщених уздовж доріжок на алюмінієвому шарі. Для читання інформації з таких дисків використовують лазерний промінь з довжиною хвилі 780 нм, який просвічує полікарбонатний шар і відбивається від алюмінієвого шару або

Рис. 1.34. Компакт-диск тільки для читання пам’яті

розсіюється, що означає логічний 0 або 1. Існує два фізичних розміри таких дисків: 12 см (4,7") та 8 см (3,1") з товщиною 1,2 мм. Диски розміром 4,7" можуть мати ємність 650 Мб, 700 Мб і навіть 800 Мб. Ємність дисків розміром 3,1" може знаходитися в межах від 140 до 210 Мб.

Компакт-диски CD-ROM виготовляють шляхом штампування із заготовлених матриць і перезапису не підлягають. Для запису інформації на оптичні диски власноруч почали випускати диски *CD-R (Compact Disc-Recordable)* та однойменні пристрої – для одноразового запису, *CD-RW (Compact Disc-ReWritable)* – для багаторазового перезапису. Інформація на такі диски може записуватись до тисячі разів і використовуватись багато років.

Швидкість передавання даних для CD-дисків позначається числом-множником (1_X = 150 Кб/с). Зокрема, зазначення такого числа на диску може означати максимально-можливу швидкість запису інформації. Слід мати на увазі, що швидкість запису залежить не тільки від характеристик диска, але і від характеристик приводу диска, що використовується для запису. Швидкість читання інформації з *CD-диска* визначається числом-множником, яке зазначене на *CD-приводі.*

Наступним поколінням оптичних носіїв після CD стали *DVD-диски (Digital Versatile Disc) – цифровий багатофункціональний диск*. Основною вимогою під час розробки DVD-дисків було збільшення обсягу збережених даних за рахунок розміщення якомога більшої кількості насічок уздовж доріжок на диску за дешевої технології їх виготовлення. Це було досягнуто, в першу чергу, за рахунок застосування променя лазера з меншою довжиною хвилі – 650 нм, на відміну від 780 нм, який застосовувався під час роботи із CD дисками.

Існує два фізичні розміри DVD-дисків, які цілком аналогічні дискам CD. Це забезпечило можливість приводам DVD-дисків

працювати також із всіма типами CD-дисків. За конструктивним виконанням DVD-диски обох розмірів поділяють на 4 різних типи. Вони бувають одно- та двошаровими (тобто, можуть мати один або два інформаційні шари), а також одно- та двосторонніми (тобто, інформація може записуватися на одній або двох сторонах диска). Залежно від розміру, типу та об'єму DVD-диски мають відповідне маркування.

Як і у випадку із CD-дисками, швидкість передавання даних для *DVD* позначається числом-множником (1_X = 1,38 Мб/с).

Сучасними представниками оптичних носіїв є *HD DVD-диски (High Definition DVD – DVD високої чіткості)* та *Blu-ray Disc* або *BD (blue ray disc* – диск з голубим променем). Принципова відмінність HD DVD від Blu-Ray полягає в тому, що в HD DVD збережена фізична структура диска DVD, тоді як у Blu-Ray застосовують нову структуру і використовують іншу технологію запису.

Стандарт *HD DVD* був спільно розроблений компаніями *Toshiba* і *NEC*. Він є подальшим розвитком формату DVD і може зберігати втричі більше даних, ніж його попередник – 15 Гб на одному шарі замість 4,7 Гб. HD DVD-диски розміром 12 см можуть мати об'єм 15 Гб (одношарові односторонні), 30 Гб (одношарові двосторонні або двошарові односторонні), 60 Гб (двошарові двосторонні). У стандарті HD DVD існують також диски розміром 8 см, які здатні зберігати 4,7 Гб даних (одношарові односторонні), 9,4 Гб – (одношарові двосторонні або двошарові односторонні), 18,8 Гб – (двошарові двосторонні).

Магнітооптичні носії. Ці носії поєднують магнітні і оптичні технології запису та зчитування інформації. Запис на магнітооптичний диск здійснюється таким чином: промінь лазера розігріває поверхню інформаційного шару диска до певної температури, після чого електромагнітний імпульс змінює показник переломлення матеріалу інформаційного шару в точці запису. Читання диска здійснюється тим же

самим лазером, але на меншій потужності, недостатній для розігрівання диска. При цьому лазерний промінь проходить крізь матеріал диска, відображається від підкладки, проходить крізь оптичну систему і потрапляє на датчик, або розсіюється. Отриманий сигнал свідчить про логічну 1. Перші магнітооптичні диски були розміром 5,25", потім з'явилися диски розміром 3,5" (рис.1.35.). Для дисків розміром 5,25" характерні такі ємності: 1,2 Гб, 1,3 Гб, 2,6 Гб, 4,8 Гб, 5,2 Гб, 9,1 Гб. Диски розміром 3,5" на сьогодні мають ємність: 128 Мб, 230 Мб, 540 Мб, 640 Мб, 1,3 Гб, 2,3 Гб. Швидкість передавання даних у таких носіях становить близько 4 Мб/с.

Рис. 1.35. Магніто-оптичний диск

Однак досить висока вартість приводів і носіїв цього типу не дозволяє віднести їх до пристроїв масового попиту.

Накопичувачі на Flash-носіях. Цей тип носіїв набув широкого поширення останнім часом. Перевагою флеш-носіїв над магнітними, оптичними та магнітооптичними носіями є відсутність частин, що рухаються. За рахунок цього флеш-пам'ять більш компактна, дешева (з урахуванням вартості пристроїв читання/запису) і забезпечує більш швидкий доступ до даних. У зазначеному типі носіїв можна виділити *USB Flash-накопичувачі* та *Flash-карти*. Перші із них, зазвичай, виготовляють у вигляді брелоків різноманітних конструкцій і форм, відмінною особливістю яких є *USB-конектор,* за допомогою якого їх під'єднують до стандартних *USB-роз'ємів* ПК.

Рис. 1.36. USB Flash-накопичувачі

USB Flash-накопичувачі (рис. 1.36) можуть мати такі значення ємності: 64 Мб, 128 Мб,

256 Мб, 512 Мб, 1Гб, 2 Гб, 4 Гб, 8 Гб. На сьогодні також існують моделі ємністю 64 Гб. Швидкість передавання даних для таких накопичувачів досягає 24 Мб/с у режимі читання та 10 Мб/с у режимі запису інформації.

Flash-карти (рис.1.37) завдяки своїй компактності, відносній дешевизні та відсутності потреби в енергії широко використовуються в портативних пристроях, що працюють на батареях і акумуляторах, цифрових фото- і відеокамерах, кишенькових персональних комп'ютерах (КПК), цифрових диктофонах, MP3-програвачах, мобільних телефонах тощо. На сьогодні існує велике різноманіття форматів флеш-карт, які постійно вдосконалюються та видозмінюються.

Розвиток будь-яких видів пам'яті відбувається в трьох основних напрямах: нарощування ємності носія, збільшення швидкодії і зниження вартості. Проте для флеш-карт, на сучасному етапі, крім зазначених, не менш важливим є такий параметр, як фізичний розмір. Тенденція зменшення розміру флеш-карти, за збереження, а то і покращення інших показників, лежить в основі створення нових типів карт у певних форматах. Розглянемо найбільш поширені формати флеш-карт.

а б в г д е

Рис. 1.37. Flash-карти

CompactFlesh (CF) (рис. 1.37а) – один із найстаріших форматів носіїв такого типу, розроблений компанією *SanDisk* в 1994 році. Фізичний розмір карти становить 42,8×36,4×3,3 мм. У міру розвитку технологій у зазначеному форматі замість формату *CompactFlash (Type I)*

був реалізований формат *CompactFlash Type II*. Ємності флеш-карт зазначених форматів на сьогодні представлені від 128 Мб до 16 Гб. Усереднений показник швидкості запису/зчитування даних для них становить 8 Мб/с.

Memory Stick (MS) (рис. 1.37б) – формат карт пам'яті, створений корпорацією *Sony* в 1998 році, використовується в різних електронних пристроях переважно марки Sony. Повнорозмірний варіант карти має таке співвідношення сторін: 21,5×50×2,8 мм. Як й інші формати флеш-карт, *MS* зазнала в процесі свого розвитку низку змін, внаслідок чого з'явилися такі підвиди, як *MS Duo* (де відбулося зменшення габаритів), *MS Pro* і *MS Hight Speed* (збільшення швидкості роботи) і різні їх комбінації. Ємності флеш-карт форматів *MS Pro* (повнорозмірного) та *MS Pro Duo* на сьогодні досягли 8 Гб. Такі карти забезпечують проведення операції читання/запису даних, у середньому, зі швидкістю 15 Мб/с.

Multi Media Card (MMC) (рис. 1.37в) – формат карт пам'яті, розроблений в 1997 році компаніями *Siemens* та *Transcend*, розмір якої 24×32×1,5 мм. Спочатку *MMC* призначалася для зберігання даних у мультимедійних пристроях, проте завдяки невеликим габаритам і зниженому енергоспоживанню ММС почали оснащувати мобільні телефони, КПК і цифрові фотоапарати. З 2004 року випускається також у зменшеному корпусі – *RS-MMC (Reduced size MMC)*. Найбільш поширені на сьогодні ММС-карти ємністю від 128 Мб до 4 Гб. Усереднений показник швидкості читання/запису даних для них становить 10 Мб/с.

Secure Digital (SD) (рис. 1.37г) – формат карт пам'яті, розроблений в 2001 році фірмою *SanDisk* на основі ММС-карти, розмір якої становить 24×32×2,1 мм. Карта забезпечена власним контролером і спеціальною областю, здатною, на відміну від ММС, записувати інформацію так, щоб було заборонено незаконне читання інформації, що і було закріплено в назві – «Secure Digital». У більшості випадків

SD-карту можна замінити MMC-картою. Для мініатюрних приладів розроблені *miniSD* розміром 20×21,5×1,4 мм і найменша із всіх карт – *microSD* (раніше відома як TransFlash) розміром 11×15×1 мм. Карти *miniSD* і *microSD* мають адаптери, за допомогою яких їх можна вставляти в будь-який слот для звичайної SD-карти.

Ємність SD-карт може бути від 8 Мб до 8 Гб. Розробляються карти з обсягом пам'яті в 16 Гб. Найнижчі показники швидкості для них становлять 6_X (0,9 Мб/с), найвищі – 150_X (22,5 Мб/с).

xD-Picture Card (xD) (рис. 1.37д) – формат карт пам’яті, створений у 2002 році компаніями *Olympus* і *Fuji*, який набув популярності завдяки численності моделей цифрових камер, що використовують цей формат. Розміри карти *xD* 20×25×1,7 мм. На сьогодні максимальна ємність цих карт пам'яті становить 2 Гб. Швидкісні показники для карти *xD* знаходяться в межах лише 5 Мб/с.

SmartMedia (SM) (рис. 1.37е) – формат карт пам’яті, створений компанією *Toshiba* в 1995 році, як альтернатива форматам *MiniCard*, *CompactFlash* і *PC Card*. Спочатку *SM* проголошувалася спадкоємцем Floppy-дисків. Розмір карти становить 45×37×0,76 мм. Вона була найтоншою і найдешевшою з перших карт пам'яті. На початковому етапі карти *SM* широко застосувалися у цифрових фотокамерах, досягнувши піку в 2001 році. Ємності цих карт можуть бути від 2 Мб до 128 Мб. Швидкість передавання даних – 2 Мб/с.

1.3. Операційні системи та програмне забезпечення комп'ютерів

1.3.1. Операційні системи

Поняття операційної системи. Причиною появи операційних систем була необхідність створення зручних у використанні комп'ютерних систем (під *комп'ютерною системою* будемо розуміти сукупність апаратного і програмного забезпечення комп'ютера). Комп'ютерні системи від самого початку розроблялися для розв'язання практичних задач користувачів. Оскільки робити це за допомогою лише апаратного забезпечення виявилося складно, було створено прикладні програми [4]. Для таких програм знадобилися загальні операції керування апаратним забезпеченням, розподілу апаратних ресурсів тощо. Ці операції згрупували в рамках окремого рівня програмного забезпечення, який і стали називати операційною системою.

Далі можливості операційних систем вийшли далеко за межі базового набору операцій, необхідних прикладним програмам, але проміжне становище таких систем між прикладними програмами й апаратним забезпеченням залишилося незмінним.

Можна дати таке означення операційної системи.

Операційна система (ОС) – це програмне забезпечення, що реалізує зв'язок між прикладними програмами й апаратними засобами комп'ютера.

Призначення операційної системи. Операційні системи забезпечують, по-перше, зручність використання комп'ютерної системи, по-друге, ефективність і надійність її роботи.

Перша функція властива ОС як розширеній машині, друга – ОС як розподілювача апаратних ресурсів.

Операційна система як розширена машина. За допомогою операційної системи у прикладного програміста (а через його програми і в користувача) має створюватися враження, що він працює з розширеною машиною.

Апаратне забезпечення комп'ютера недостатньо пристосоване до безпосереднього використання у програмах. Наприклад, якщо розглянути роботу із пристроями введення-виведення на рівні команд відповідних контролерів, то можна побачити, що набір таких команд обмежений, а для багатьох пристроїв – примітивний (є навіть вислів: «апаратне забезпечення потворне») [23]. Операційна система приховує такий *інтерфейс апаратного забезпечення,* замість нього програмістові пропонують *інтерфейс прикладного програмування* (рис. 1.38), що використовує поняття вищого рівня (їх називають абстракціями).

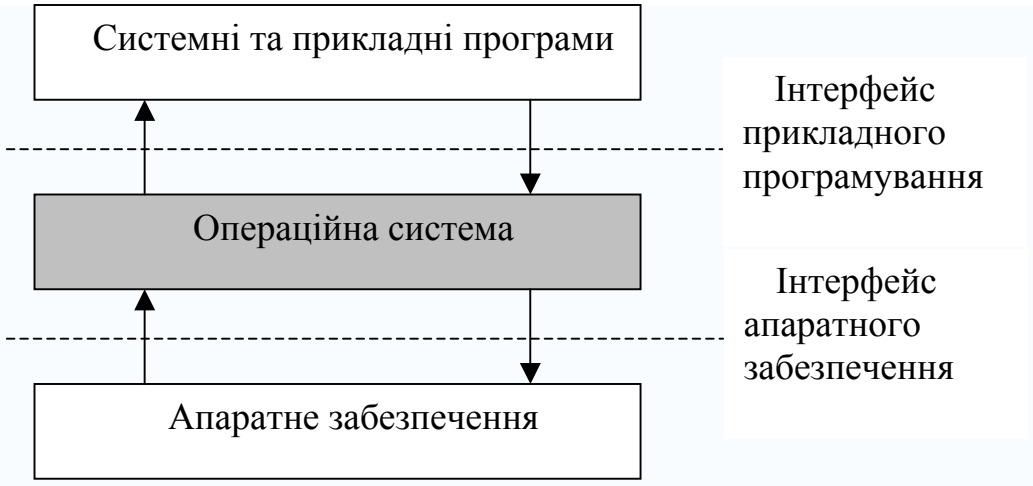


Рис. 1.38. Взаємодія ОС із апаратним забезпеченням і застосуванням

Наприклад, під час роботи з диском типовою абстракцією є файл. Працювати з файлами простіше, ніж безпосередньо з контролером диска (не потрібно враховувати переміщення головок дисковода, запускати й зупиняти мотор тощо), внаслідок цього програміст може зосередитися на суті свого прикладного завдання. За взаємодію з контролером диска відповідає операційна система.

Виділення абстракцій дає змогу досягти того, що код ОС і прикладних програм не потребуватиме зміни у разі переходу на нове апаратне забезпечення. Наприклад, якщо встановити на комп'ютері дисковий пристрій нового типу (за умови, що він підтримується ОС), всі його особливості будуть враховані на рівні ОС, а прикладні програми продовжуватимуть використовувати файли, як і раніше. Така характеристика системи називається *апаратною незалежністю.* Можна сказати, що ОС надають апаратно-незалежне середовище для виконання прикладних програм.

Операційна система як розподілювач ресурсів. Операційна система має ефективно розподіляти ресурси. Під ресурсами розуміють процесорний час, дисковий простір, пам'ять, засоби доступу до зовнішніх пристроїв. Операційна система виступає в ролі менеджера цих ресурсів і надає їх прикладним програмам на вимогу.

Розрізняють два основні види розподілу ресурсів. У разі *просторового розподілу* ресурс доступний декільком споживачам одночасно, до того ж кожен із них може користуватися частиною ресурсу (так розподіляється пам'ять). У разі *часового розподілу* система ставить споживачів у чергу і згідно з нею надає їм змогу користуватися всім ресурсом обмежений час (так розподіляється процесор в однопроцесорних системах).

Під час розподілу ресурсів ОС розв'язує можливі конфлікти, запобігає несанкціонованому доступу програм до тих ресурсів, на які вони не мають прав, забезпечує ефективну роботу комп'ютерної системи.

Класифікація сучасних операційних систем. Розглянемо *класифікацію сучасних операційних систем* залежно від сфери їх застосування [23].

Насамперед відзначимо *ОС великих ЕОМ* (мейнфреймів). Основною характеристикою апаратного забезпечення, для якого їх розробляють, є

продуктивність введення-виведення: великі ЕОМ оснащують значною кількістю периферійних пристроїв (дисків, терміналів, принтерів тощо). Такі комп'ютерні системи використовують для надійної обробки значного обсягу даних, при цьому ОС має ефективно підтримувати цю обробку (в пакетному режимі або в режимі розподілу часу). Прикладом ОС такого класу може бути OS/390 фірми IBM.

До наступної категорії можна віднести *серверні ОС.* Головна характеристика таких ОС – здатність обслуговувати велику кількість запитів користувачів до спільно використовуваних ресурсів. Важливу роль для них відіграє мережна підтримка. Є спеціалізовані серверні ОС, з яких виключені елементи, не пов'язані з виконанням їхніх основних функцій (наприклад, підтримка застосувань користувача). Нині для реалізації серверів частіше застосовують універсальні ОС (UNIX або системи лінії Windows XP).

Наймасовіша категорія – *персональні ОС.* Деякі ОС цієї категорії розробляли з розрахунком на непрофесійного користувача (лінія Windows 95/98/Me фірми Microsoft, яку далі називатимемо Consumer Windows), інші є спрощеними версіями універсальних ОС. Особлива увага в персональних ОС приділяється підтримці графічного інтерфейсу користувача і мультимедійних технологій.

Виділяють також *ОС реального часу.* У такій системі кожна операція має бути гарантовано виконана в заданому часовому діапазоні. ОС реального часу можуть керувати польотом космічного корабля, технологічним процесом або демонстрацією відеороликів. Існують спеціалізовані ОС реального часу, такі як QNX і VxWorks.

Ще однією категорією є *вбудовані ОС.* До них належать керуючі програми для різноманітних мікропроцесорних систем, які використовують у військовій техніці, системах побутової електроніки, смарт-картах та інших пристроях. До таких систем ставлять особливі

вимоги: розміщення в малому обсязі пам'яті, підтримка спеціалізованих засобів введення-виведення, можливість прошивання в постійному запам'ятовувальному пристрої. Часто вбудовані ОС розробляються під конкретний пристрій; до універсальних систем належать Embedded Linux і Windows CE.

Функціональні компоненти операційних систем. Операційну систему можна розглядати як сукупність *функціональних компонентів*, кожен з яких відповідає за реалізацію певної функції системи. У цьому розділі описано найважливіші функції сучасних ОС і компоненти, що їх реалізують [23].

Спосіб побудови системи зі складових частин та їх взаємозв'язок визначає архітектура операційної системи.

Керування процесами й потоками. Як ми вже згадували, однією з найважливіших функцій ОС є виконання прикладних програм. Код і дані прикладних програм зберігаються в комп'ютерній системі на диску в спеціальних виконуваних файлах. Після того як користувач або ОС вирішать запустити на виконання такий файл, у системі буде створено базову одиницю обчислювальної роботи, що називається *процесом* (process).

Можна дати таке означення: процес – це програма під час її виконання.

Операційна система розподіляє ресурси між процесами. До таких ресурсів належать процесорний час, пам'ять, пристрої введення-виведення, дисковий простір у вигляді файлів. Під час розподілу пам'яті з кожним процесом пов'язується його *адресний простір* – набір адрес пам'яті, до яких йому дозволено доступ. В адресному просторі зберігаються код і дані процесу. Під час розподілу дискового простору для кожного процесу формується список відкритих файлів, аналогічним чином розподіляють пристрої введення-виведення.

Процеси забезпечують захист ресурсів, якими вони володіють. Наприклад, до адресного простору процесу неможливо безпосередньо звернутися з інших процесів (він є захищеним), а під час роботи з файлами може бути задано режим, що забороняє доступ до файлу всім процесам, крім поточного.

Розподіл процесорного часу між процесами необхідний через те, що процесор виконує інструкції одну за одною (тобто в конкретний момент часу на ньому може фізично виконуватися тільки один процес), а для користувача процеси мають виглядати як послідовності інструкцій, виконувані паралельно. Щоб домогтися такого ефекту, ОС надає процесор кожному процесу на деякий короткий час, після чого перемикає процесор на інший процес; при цьому виконання процесів відновлюється з того місця, де їх було перервано. У *багатопроцесорній системі* процеси можуть виконуватися паралельно на різних процесорах.

Сучасні ОС крім багатозадачності можуть підтримувати *багатопотоковість* (multithreading), яка передбачає в рамках процесу наявність кількох послідовностей інструкцій *(потоків,* threads), які для користувача виконуються паралельно, подібно до самих процесів в ОС. На відміну від процесів потоки не забезпечують захисту ресурсів (наприклад, вони спільно використовують адресний простір свого процесу).

Керування пам'яттю. Під час виконання програмного коду процесор бере інструкції й дані з оперативної (основної) пам'яті комп'ютера. При цьому така пам'ять відображається у вигляді масиву байтів, кожен з яких має адресу.

Як уже згадувалося, основна пам'ять є одним із видів ресурсів, розподілюваних між процесами. ОС відповідає за виділення пам'яті під захищений адресний простір процесу і за вивільнення пам'яті після того, як виконання процесу буде завершено. Обсяг пам'яті, доступний процесу,

може змінюватися в ході виконання, у цьому разі говорять про динамічний розподіл пам'яті.

ОС має забезпечувати можливість виконання програм, які окремо або в сукупності перевищують за обсягом доступну основну пам'ять. Для цього в ній має бути реалізовано технологію *віртуальної пам'яті.* Така технологія дає можливість розміщувати в основній пам'яті тільки ті інструкції й дані процесу, які потрібні в поточний момент часу, при цьому вміст іншої частини адресного простору зберігається на диску.

Керування введенням-виведенням. Операційна система відповідає за керування пристроями введення-виведення, підключеними до комп'ютера. Підтримка таких пристроїв в ОС зазвичай здійснюється на двох рівнях. До першого, нижчого, рівня належать *драйвери пристроїв* – програмні модулі, які керують пристроями конкретного типу з урахуванням усіх їх особливостей. До другого рівня належить *універсальний інтерфейс введення-виведення,* зручний для використання у прикладних програмах.

ОС має реалізовувати загальний інтерфейс драйверів введення-виведення, через який вони взаємодіють з іншими компонентами системи. Такий інтерфейс дає змогу спростити додавання в систему драйверів для нових пристроїв.

Сучасні ОС надають великий вибір готових драйверів для конкретних периферійних пристроїв. Що більше пристроїв підтримує ОС, то більше в неї шансів на практичне використання.

Керування файлами та файлові системи. Для користувачів ОС і прикладних програмістів дисковий простір надається у вигляді сукупності *файлів,* організованих у *файлову систему.*

Файл – це набір даних у файловій системі, доступ до якого здійснюється за іменем. Термін «файлова система» може вживатися для двох понять: принципу організації даних у вигляді файлів і конкретного набору даних (зазвичай відповідної частини диска), організованих

відповідно до такого принципу. У межах ОС може бути реалізована одночасна підтримка декількох файлових систем.

Файлові системи розглядають на логічному і фізичному рівнях. Логічний рівень визначає зовнішнє подання системи як сукупності файлів (які звичайно перебувають у каталогах), а також виконання операцій над файлами і каталогами (створення, вилучення тощо). Фізичний рівень визначає принципи розміщення структур даних файлової системи на диску або іншому пристрої.

Мережеві системи. Сучасні операційні системи пристосовані до роботи в мережі, їх називають *мережевими операційними системами* [11]. Засоби мережевої підтримки дають ОС можливість:

- надавати локальні ресурси (дисковий простір, принтери тощо) у загальне користування через мережу, тобто функціонувати як сервер;
- звертатися до ресурсів інших комп'ютерів через мережу, тобто функціонувати як клієнт.

Реалізація функціональності сервера і клієнта базується на *транспортних засобах,* відповідальних за передавання даних між комп'ютерами відповідно до правил, зумовлених мережевими протоколами.

Розподілені системи. Мережеві ОС не приховують від користувача наявність мережі, мережева підтримка в них не визначає структуру системи, а збагачує її додатковими можливостями. Є також *розподілені ОС* [6], які дають змогу об'єднати ресурси декількох комп'ютерів у *розподілену систему.* Вона виглядає для користувача як один комп'ютер з декількома процесорами, що працюють паралельно. Розподілені та багатопроцесорні системи є двома основними категоріями ОС, які використовують декілька процесорів та мають багато спільного.

Під *безпекою даних* в ОС розуміють забезпечення надійності системи (захисту даних від втрати у разі збоїв) і захист даних від

несанкціонованого доступу (випадкового чи навмисного).

Для захисту від несанкціонованого доступу ОС має забезпечувати наявність засобів *аутентифікації* користувачів (такі засоби дають змогу з'ясувати, чи є користувач тим, за кого себе видає; зазвичай для цього використовують систему паролів) та їх *авторизації* (дозволяють перевірити права користувача, що пройшов аутентифікацію, на виконання певної операції).

Розрізняють два типи засобів взаємодії користувача з ОС: *командний інтерпретатор* (shell) і *графічний інтерфейс користувача* (GUI).

Командний інтерпретатор дає змогу користувачам взаємодіяти з ОС, використовуючи спеціальну командну мову (інтерактивно або через запуск на виконання командних файлів). Команди такої мови змушують ОС виконувати певні дії (наприклад, запускати програми, працювати із файлами).

Графічний інтерфейс користувача надає йому можливість взаємодіяти з ОС, відкриваючи вікна і виконуючи команди за допомогою меню або кнопок. Підходи до реалізації графічного інтерфейсу доволі різноманітні: наприклад, у Windows-системах засоби його підтримки вбудовані в систему, а в UNIX вони є зовнішніми для системи і спираються на стандартні засоби керування введенням-виведенням.

1.3.2. Програмне забезпечення комп'ютерів

Структура програмного забезпечення. Для початку роботи з комп'ютерною системою недостатньо лише наявності апаратних засобів. Повинно бути також відповідне програмне забезпечення (ПЗ). Під програмним забезпеченням розуміють сукупність програм системи обробки інформації і програмних документів, необхідних для експлуатації

цих програм [19]. Також, це сукупність програм, процедур і правил, а також документації, що належить до функціонування системи обробки даних. Програми – це впорядковані послідовності команд.

Між програмами, як і між фізичними пристроями і блоками комп'ютерної системи, існує взаємозв'язок, тобто *міжпрограмний інтерфейс*. Можливість функціонування такого зв'язку заснована на існуванні технічних умов і протоколів взаємодії, а на практиці – забезпечується розподілом програмного забезпечення на декілька рівнів, що взаємодіють між собою. Кожний наступний рівень спирається на програмне забезпечення попередніх рівнів. Таке розчленовування зручно для всіх етапів роботи з комп'ютерною системою, починаючи з установлення програм – до практичної експлуатації і технічного обслуговування.

За призначенням ПЗ поділяють на такі рівні: базовий, системний, службовий, прикладний, інструментальний. Поділ ПЗ на рівні є умовним, оскільки деякі програми можуть входити як до одного рівня, так і до іншого.

Базовий рівень – найнижчий рівень ПЗ. Базове ПЗ відповідає за взаємодію з базовими апаратними засобами. Як правило, базові програмні засоби безпосередньо входять до складу базового устаткування і зберігаються в спеціальних мікросхемах, що називаються постійними запам'ятовувальними пристроями (ПЗП). Програми і дані записуються в мікросхеми ПЗП на етапі їх виробництва і не можуть бути змінені в процесі їх експлуатації.

Коли зміна базових програмних засобів під час експлуатації є технічно доцільною, замість схем ПЗП застосовують постійні запам'ятовувальні пристрої з можливістю їх перепрограмування (ППЗП). У такому випадку вміст ППЗП можна змінювати як безпосередньо у складі обчислювальної системи (така технологія називається *флеш-*

технологією), так і за її межами на спеціальних пристроях, які називаються програматорами.

Системний рівень – перехідний рівень ПЗ. Програми, що працюють на цьому рівні, забезпечують взаємодію інших програм комп'ютерної системи з програмами базового рівня і безпосередньо з апаратним забезпеченням, тобто виконують посередницькі функції. Від ПЗ цього рівня залежать експлуатаційні показники всієї обчислювальної системи загалом. До складу системного ПЗ входять: операційна система, завантажувач операційної системи, драйвери пристроїв, програмні кодеки.

Службовий рівень. ПЗ цього рівня взаємодіє як з програмами базового рівня, так і з програмами системного рівня. Основне призначення службових програм (їх також називають утилітами) полягає в автоматизації робіт із перевірки і налагодження комп'ютерної системи. У багатьох випадках їх використовують для розширення або поліпшення функцій системних програм. Деякі службові програми (як правило, це програми обслуговування) входять до складу ОС, але більшість службових програм є зовнішніми і слугують для розширення функцій системних програм.

У розробці і експлуатації службових програм існує два альтернативні напрями: інтеграція з ОС і автономне функціонування. У першому випадку службові програми можуть змінювати споживацькі властивості системних програм, роблячи їх більш зручними для практичної роботи. У другому випадку – вони мало пов'язані зі системним програмним забезпеченням, але надають користувачу більше можливостей для персональної настройки їх взаємодії з апаратним і програмним забезпеченням.

Прикладний рівень. ПЗ прикладного рівня є комплексом прикладних програм, за допомогою яких на конкретному робочому місці виконуються

конкретні завдання. Спектр цих завдань надзвичайно широкий – від виробничих до творчих і розважально-навчальних. Оскільки між прикладним програмним забезпеченням і системним існує безпосередній взаємозв’язок (перше спирається на друге), то можна стверджувати, що універсальність, функціональні можливості комп’ютерної системи, доступність прикладного програмного забезпечення залежать від типу операційної системи, що використовується, від того, як вона забезпечує взаємодію триєдиного комплексу «користувач – програма – устаткування».

Інструментальний рівень (системи програмування) – інтегроване програмне середовище для створення нових програм для комп'ютера. Використовується програмістами для розробки ПЗ. Зазвичай середовище розробки включає текстовий редактор, компілятор і/або інтерпретатор, засоби автоматизації складання елементів програми і налагодження. Інколи до його складу входять система керування версіями та засоби для спрощення конструювання графічного інтерфейсу користувача. Багато сучасних середовищ програмування також включають Браузери класів, інспектор об’єктів і діаграму ієрархії класів, які використовуються під час розробки об’єктно-орієнтованого ПЗ.

Більшість користувачів комп'ютерів використовують програми, призначені для виконання конкретних прикладних завдань, таких як підготовка і оформлення документів, математичні обчислення, обробка зображень тощо. Відповідні програмні засоби називають прикладними програмами або прикладним програмним забезпеченням. Управління компонентами обчислювальної системи і формування середовища для функціонування прикладних програм бере на себе системне програмне забезпечення, найбільш важливою складовою якого є операційна система [2].

Зазвичай, програмні середовища передбачають використання однієї мови програмування, наприклад Visual Basic. Існують також пакети

програмних середовищ, призначені для використання декількох мов програмування, наприклад, Microsoft Visual Studio. Visual Studio може містити такі компоненти: Visual Basic .NET, Visual C++, Visual C#, Visual J#. До складу Microsoft Visual Studio можуть також входити програмні середовища: Microsoft SQL Server або MSDE, Visual Source Safe. До недавнього часу до пакета також входили: Visual InterDev, Visual J++, Visual FoxPro.

Приклади середовищ для розробки ПЗ: Turbo Pascal, Borland C++, GNU toolchain, DrPython, Borland Delphi, Dev-C++, Lazarus, Assembler, Cobol, Fortran, Basic, Lips, Java, Html та інші.

Класифікація прикладних програмних засобів. Завдяки програмам цього класу користувач безпосередньо може виконувати конкретні практичні завдання з використанням комп'ютерної системи. Вид прикладного програмного забезпечення визначає якість виконання завдань і коло задач, що розв'язують.

Текстові редактори і процесори. Програми для роботи з текстом умовно поділяють на два типи: текстові редактори і текстові процесори.

Текстові редактори – комп'ютерні програми, призначені для створення і редагування текстових файлів, їх перегляду на екрані, виведення на друк, пошуку фрагментів тексту тощо.

Основні функції прикладних програм цього класу полягають у введенні і редагуванні текстових даних. Додаткові функції – це автоматизація процесів введення і редагування. Для операцій введення, виведення і збереження даних текстові редактори використовують системне програмне забезпечення.

Програми цього типу ще називають редакторами кодів, оскільки основне їх призначення – це написання вихідних кодів комп'ютерних програм. До текстових редакторів відносять програми Блокнот, WordPad,

Word Pro, EditPlus, Emacs, EmEditor, jEdit, Kate, Notepad, Just Write та інші.

Текстові процесори. Основна відмінність текстових процесорів від текстових редакторів в тому, що вони дозволяють не тільки вводити і редагувати текст, але і форматувати його в широких межах, тобто оформляти. Відповідно, до основних засобів текстових процесорів відносять засоби забезпечення взаємодії тексту, графіки, формул, таблиць та інших об'єктів, що становлять підсумковий документ, а до додаткових – засоби автоматизації процесу форматування.

Найбільш поширено використовують текстові процесори Microsoft Word, Corel WordPerfect, TeX, OpenOffice.org Writer, AbiWord, GNU Aspell та інші.

Останнім часом додатково виокремлюють більш загальний клас програм – *текстові робочі середовища*. Такі програмні засоби представляють повноцінне робоче середовище, в якому можна розв'язувати різноманітні задачі: писати і читати листи, web-канали, працювати у Wiki і у Web, вести щоденник тощо. Вони також можуть слугувати для розробки програмного забезпечення, оскільки містять текстовий редактор як необхідний інструмент програмування. До програм цього класу належать Emacs, Archy, Vim, Acme та інші.

Редактори HTML (Web-редактори) – комп'ютерні програми, які дозволяють створювати і редагувати сторінки у форматі HTML. Це особливий клас програм, що об'єднують властивості текстових і графічних редакторів. Вони призначені для створення і редагування *Web-документів (Web-сторінок Інтернету*). Web-документи – це електронні документи, під час підготовки яких слід враховувати деякі особливості, пов'язані з прийманням/передаванням інформації в Інтернеті. Теоретично для створення Web-документа можна використовувати звичайні текстові редактори і процесори, а також деякі з графічних редакторів векторної

графіки, але Web-редактори володіють низкою корисних функцій, що підвищує продуктивність праці Web-дизайнерів. Програми цього класу можна також ефективно використовувати для створення електронних документів і мультимедійних видань. Поширеними версіями Web-редакторів є FCKeditor, Quanta Plus, HTMLArea, Microsoft FrontPage Express, Nvu, OpenOffice.org, WISIWYG Web Builder, Adobe GoLive, Namo Web Editor.

Графічні редактори. Це загальний клас програм, призначених для створення і (або) обробки графічних зображень. У цьому класі розрізняють такі категорії: растрові редактори, векторні редактори, програмні засоби для створення і обробки тривимірної графіки (3D – редактори).

Растрові редактори. У цих редакторах графічний об'єкт представляється у вигляді комбінації крапок, що утворюють растр, які мають певний колір і яскравість. Такий підхід ефективний, коли графічне зображення має багато тонів та напівтонів, а інформація про колір елементів, що становлять об'єкт, важливіша, ніж інформація про їх форму. Це характерно для фотографічних і поліграфічних зображень. Растрові редактори широко застосовують для обробки зображень, їх ретуші, створення фотоефектів і художніх композицій. До растрових редакторів відносяться такі досить популярні програми, як Adobe Photoshop, Mac OS X, GIMP.

Векторні редактори відрізняються від растрових способом представлення даних про зображення. Такий підхід характерний для креслярсько-графічних робіт, в яких форма лінії має більше значення, ніж інформація про колір окремих крапок, що її складають. У векторних редакторах кожна лінія розглядається як математична крива третього порядку і, відповідно, представляється не комбінацією крапок, а математичною формулою (в пам'яті комп'ютера зберігаються числові

коефіцієнти цієї формули). Таке представлення набагато компактніше, ніж растрове, відповідно дані займають набагато менше місця. До векторних редакторів відносять Corel Draw, Macromedia Free Hand, Inkscape.

Редактори тривимірної графіки використовують для створення тривимірних композицій. Вони мають дві характерні особливості. По-перше, дозволяють гнучко управляти взаємодією властивостей поверхні об'єктів, що зображаються, з властивостями джерел освітлення. По-друге, дозволяють створювати тривимірну анімацію. Широке розповсюдження також здобули такі версії 3D-редакторів: Autodesk Maya, Newtek Lightwave, 3DS Max, SoftImage XSI і порівняно нові: Rhinoceros 3D, Cinema 4D, Blender, PovRay, Wings3D.

Системи керування базами даних (СКБД). Базами даних називають величезні масиви даних, сформовані у табличні структури. Основними функціями СКБД є: створення порожньої (незаповненої) структури баз даних; надання засобів для її заповнення або імпорту даних з таблиць іншої бази; забезпечення можливості доступу до даних, а також пошуку і фільтрування.

Багато СКБД додатково надають можливість проведення найпростішого аналізу даних та їх обробки. В результаті можливе створення нових таблиць і баз даних на основі тих, що існують. У зв'язку з широким розповсюдженням мережевих технологій до сучасних систем управління базами даних пред'являються вимоги до можливості роботи з віддаленими і розподіленими ресурсами, що знаходяться на серверах всесвітньої комп'ютерної мережі. До сучасних баз даних відносять Microsoft Access, Oracle, FoxPro, Clarion, Informix, Dataflex, Hyper Card, SQL Server, DB2 та інші.

Табличні процесори – категорія ПЗ, призначеного для роботи з електронними таблицями. *Електронні таблиці* – комплексні засоби для

зберігання різних типів даних та їх обробки. Відрізняються від баз даних тим, що в них основний акцент зміщений не на зберігання масивів даних, а на перетворення даних відповідно до їх внутрішнього змісту. Основна властивість електронних таблиць полягає в тому, що в разі зміни вмісту будь-якої комірки таблиці може відбуватися автоматична зміна вмісту в інших комірках, пов'язаних із зміною співвідношення, заданим математичним або логічним виразом.

Сучасні табличні процесори підтримують двовимірні таблиці, дозволяють створювати власні вхідні і вихідні форми, включати в таблиці малюнки, працювати з базами даних тощо. Крім того, є безліч можливостей декоративного характеру, включення звукових ефектів, створення слайд-шоу та ін. Табличні процесори застосовуються у сферах комп'ютерного моделювання та аналізу даних, тобто під час виконання науково-дослідних робіт. Найбільшою популярністю користуються табличні процесори Microsoft Excel, SuperCals, Abacus, Lotus 1-2-3, OpenOffice.org Cale, Gnumeric, KSpread і Quattro Pro.

Перекладачі – програми цього класу слугують для автоматизації перекладу тексту з однієї мови на іншу. Розрізняють *машинний* переклад і *автоматизований* переклад.

Машинний переклад – це переклад текстів з однієї мови на іншу за допомогою комп'ютера. Сучасні програмні продукти для реалізації машинного перекладу: PROMT, InterTran, засоби компанії Google, Socrat, Language Master, Рута Плай. Автоматизований переклад – переклад текстів на комп'ютері з використанням комп'ютерних технологій. Від машинного перекладу він відрізняється тим, що весь процес перекладу здійснюється людиною, комп'ютер лише допомагає їй відтворити готовий текст або за менший час, або з вищою якістю. Програмним забезпеченням у цьому випадку є набір електронних словників, найбільш поширеним із яких є ABBYY Lingvo.

Системи автоматизованого проектування (CAD-системи). Програми такого класу призначені для проектно-конструкторських робіт. Застосовуються в машинобудуванні, приладобудуванні, архітектурі та інших галузях. Окрім креслярсько-графічних робіт ці системи дозволяють проводити найпростіші розрахунки і вибір готових конструктивних елементів з баз даних.

Найбільш популярними CAD-системами є Компас, T-FLEX CAD, AutoCAD, FreeCAD, Cadmech, QCAD (відкрита двовимірна система проектування), BRL-CAD (відкрита 3D система проектування), Electric (проектування електропроводки), MathCAD (математичні розрахунки), P-CAD та OrCAD – проектування електронних пристроїв.

Настільні видавничі системи. Призначення цих програм полягає в автоматизації процесу верстки поліграфічних видань. Цей клас програмного забезпечення займає проміжне положення між текстовими процесорами і системами автоматичного проектування: Corel Ventura, Finess, Micro Design.

Браузери (оглядачі або засоби перегляду Web-сторінок). До цієї категорії відносять програмні засоби, призначені для перегляду електронних документів, виконаних у форматі HTML. Сучасні браузери відтворюють не тільки текст і графіку, але і музику, людську мову. Вони можуть забезпечувати прослуховування радіопередач в Інтернеті, перегляд відеоконференцій, роботу зі службами електронної пошти, зі системою телеконференцій (групами новин) і багато іншого. Найбільш популярним браузером є Internet Explorer версій 6.0, 7.0, 8.0. Досить поширеними є Firefox, Mozilla, Netscape, SeaMonkey, Maxthon, Safari, Konqueror, Swift, Opera, Nintendo DS Browser та інші.

Окрему категорію прикладних програмних засобів, що мають розвинені внутрішні системами класифікації, представляють навчальні, розвиваючі, довідкові і розважальні системи і програми. Характерною

особливістю програмного забезпечення цих класів є підвищені вимоги до мультимедійної складової (використання музичних композицій, засобів графічної анімації, відеоматеріалів).

Класифікація службових програмних засобів. Диспетчери файлів (файлові менеджери). За допомогою програм цього класу виконується більшість операцій, пов'язаних з обслуговуванням файлової структури: копіювання, переміщення і перейменування файлів, створення каталогів (папок), вилучення файлів і каталогів, пошук файлів і навігація за файловою структурою. Базові програмні засоби, призначені для цього, зазвичай входять до складу програм системного рівня і встановлюються разом з операційною системою. Проте, для підвищення зручності роботи більшість користувачів встановлює додаткові програми цього класу.

Засоби стиснення даних (архіватори) – призначені для створення архівів. Архівація даних спрощує їх зберігання за рахунок того, що великі групи файлів і каталогів зводяться в один архівний файл. При цьому підвищується ефективність використання носія інформації, оскільки архівні файли зазвичай мають підвищену щільність запису інформації. Архіватори часто використовують для створення резервних копій цінних даних. Найбільш поширеними програмами-архіваторами є Rar, WinRAR, Zip, WinZip, Ark, 7-Zip, File Roller, PeaZip, DGCA, GCA, ZipGinius та інші.

Засоби діагностики – призначені для автоматизації процесів діагностики програмного і апаратного забезпечення. Вони виконують необхідні перевірки і видають зібрану інформацію в зручному і наочному вигляді. Їх використовують не тільки для усунення неполадок, але і для оптимізації роботи комп'ютерної системи. Деякі з них входять до складу операційної системи і встановлюються на комп'ютер разом із системою. Користувач має змогу додатково встановити програмне забезпечення цього класу з метою одержання додаткових сервісних можливостей.

Засоби контролю (моніторингу) – ці програмні засоби іноді

називають моніторами. Вони дозволяють стежити за процесами, що відбуваються у комп'ютері. При цьому можливі два підходи: спостереження в реальному режимі часу або контроль із записом результатів у спеціальному протокольному файлі. Перший підхід використовують під час пошуку шляхів для оптимізації роботи обчислювальної системи і підвищення її ефективності. Другий підхід використовують, коли моніторинг виконується автоматично і (або) дистанційно. В останньому випадку результати моніторингу можна передати віддаленій службі технічної підтримки для встановлення причин конфліктів у роботі програмного і апаратного забезпечення.

Монітори установки – призначені для контролю за установленням програмного забезпечення. Необхідність у цьому програмному забезпеченні пов'язана з тим, що між різними категоріями програмного забезпечення можуть встановлюватися зв'язки. Вертикальні зв'язки (між рівнями) є необхідною умовою функціонування всіх комп'ютерів. Горизонтальні зв'язки (усередині рівнів) характерні для комп'ютерів, які працюють з операційними системами, що підтримують принцип сумісного використовування одних і тих самих ресурсів різними програмними засобами. І в тих і в інших випадках під час установлення або вилучені програмного забезпечення може відбуватися порушення працездатності інших програм. Монітори установки стежать за станом і зміною навколишнього програмного середовища, відстежують і протоколюють утворення нових зв'язків і дозволяють відновлювати зв'язки, втрачені в результаті вилучення раніше встановлених програм.

Засоби комунікації дозволяють встановлювати з'єднання з віддаленим комп'ютером, обслуговують передавання повідомлень електронної пошти, роботу з телеконференціями (групами новин), забезпечують пересилання повідомлень факсиміле і виконують безліч інших операцій в комп'ютерних мережах.

Засоби забезпечення комп'ютерної безпеки – це засоби пасивного і активного захисту даних від пошкодження, а також засоби захисту від несанкціонованого доступу, перегляду і змінювання даних. Як засоби пасивного захисту використовують службові програми, призначені для резервного копіювання. Як засоби активного захисту застосовують антивірусне програмне забезпечення. Для захисту даних від несанкціонованого доступу, їх перегляду і зміни застосовують спеціальні системи, засновані на криптографії.

Найбільш поширеними антивірусними програмами є AVP, Norton Antivirus, Doctor Web, NukeNabber, ActiveVirusShield, AhnLab, Aladdin Knowledge Systems, AVG, AVZ, Avira, ClamAV, Eset NOD32, Український Національний Антивірус та інші.

Разом з апаратним і програмним забезпеченням засобів обчислювальної техніки в деяких випадках доцільно розглядати *інформаційне забезпечення*, під яким розуміють сукупність програм і заздалегідь підготовлених даних, необхідних для роботи програм. Наприклад, робота системи автоматичної перевірки орфографії в редагованому тексті полягає в тому, що лексичні одиниці початкового тексту порівнюють з наперед підготовленим еталонним масивом даних (словником). У цьому випадку для успішної роботи системи необхідно мати окрім апаратного і програмного забезпечення спеціальні набори словників, що підключаються ззовні. Це приклад інформаційного забезпечення обчислювальної техніки.

У спеціалізованих комп'ютерних системах (бортових комп'ютерах автомобілів, судів, ракет, літаків тощо) сукупність програмного і інформаційного забезпечення називають математичним забезпеченням. Як правило, воно жорстко записується в мікросхеми ПЗП і може бути змінене шляхом заміни ПЗП або його перепрограмуванням на спеціальному устаткуванні.

Висновки

Принципи програмного керування фон Неймана визначають ідеологію архітектури електронних обчислювальних машин.

У комп'ютері числа зображують у двійковій системі числення цифрами 0 та 1, які кодують два різних стійких стани елемента пам'яті, що називається бітом.

Оперативна пам'ять може розглядатись як послідовність байтів. Кожен байт має свій номер – адресу.

Персональний комп’ютер являє собою складну технічну систему, у складі якої велика кількість функціонально завершених пристроїв.

Периферійні пристрої комп’ютерної системи забезпечують виконання низки задач, пов’язаних з виведенням, введенням та обміном даними.

Зовнішні носії інформації призначені для резервного копіювання, збільшення обсягу пам’яті та перенесення даних з однієї ЕОМ на іншу.

Операційна система – це програмне забезпечення, що забезпечує функціонування апаратного забезпечення керування роботою комп'ютера.

Основні функціональні компоненти ОС: керування процесами, керування пам'яттю, керування введенням-виведенням, керування файлами і підтримка файлових систем, мережева підтримка, забезпечення захисту даних, реалізація інтерфейсу користувача.

З погляду процесора програма – це розміщена в оперативній пам'яті послідовність команд. Команди виконуються арифметико-логічним пристроєм процесора.

Контрольні запитання та завдання

1. Що таке архітектура комп'ютера? Наведіть загальну структуру комп'ютера.

2. Принципи сучасної архітектури комп'ютера.

3. Методи класифікації комп'ютерів. Класифікація за призначенням.

4. Структура обчислювального центру на базі великої ЕОМ.

5. Призначення міні-ЕОМ та мікро-ЕОМ.

6. Персональні комп'ютери, їх класифікація за міжнародним сертифікаційним стандартом.

7. Класифікація за рівнем спеціалізації, розміром, сумісністю.

8. Які основні функції операційної системи?

9. Сформулюйте основні принципи фон Неймана.

10. Які функціональні блоки входять до складу комп'ютера?

11. У чому полягає призначення процесора та його регістрів?

12. Чим відрізняється кеш-пам'ять від інших типів оперативної пам'яті?

13. Які компоненти входять до складу зовнішньої пам'яті?

14. Чому інформація в комп'ютері записується у двійковій системі числення?

15. Для чого використовують шістнадцяткову систему числення?

16. Вкажіть основні форми зображення чисел у комп'ютері.

17. Назвіть компоненти комп’ютерної системи, що розміщені на материнській платі.

18. Які ви знаєте периферійні пристрої комп’ютерної системи?

19. Яке призначення зовнішніх носіїв інформації?

20. Класифікуйте прикладні програмні засоби.

21. Класифікуйте службові програмні засоби.

Вправи

1. Заповніть таблицю, перетворюючи десяткові числа у двійкові та шістнадцяткові і навпаки.

Десяткові числа	Двійкові числа	Шістнадцяткові числа
55		
	01010010	
		A7
136		
	10101100	
		3E
243		
	111000111	
		BC
43		
	010101010	
		DD
189		
	110011001	
		EF

2. Яка кількість вільного місця має бути у поштовій скриньці, щоб можна було отримати 3 повідомлення, які містять по 3500 символів кожне (1 символ займає 8 біт) та по 2 прикріплених файли з графічною інформацією розміром по 75 Кбайт?

3. Яку мінімальну пропускну здатність (Мбат/с) повинна мати локальна комп'ютерна мережа для забезпечення передавання текстової документації об'ємом 190 сторінок (одна сторінка містить 29 рядків по 67 символів у рядку (1 символ займає 8 біт)) за 3 с?

4. Відомості про співробітника підприємства зберігаються в електронній базі відділу кадрів у вигляді тексту в середньому із 4048 символів (1 символ займає 8 біт). На якій мінімальній кількості дискет ємністю 1,4 Мбайт можна розмістити відомості про всіх 1456

співробітників підприємства?

5. З якою регулярністю (днів) необхідно видаляти повідомлення із електронної поштової скриньки, яка має об'єм 2 Мбайт, якщо кожний день на неї надходить 15 повідомлень? Кожне повідомлення містить 1500 символів і два прикріплених файли по 10 Кбайт кожний.

6. Середня швидкість введення текстової інформації із клавіатури оператором ПК становить 70 слів за хвилину (одне слово в середньому – 6 символів (1 символ займає 8 біт)). Який об'єм (Кбайт) тексту встигне ввести оператор за три години безперервної роботи?

7. З якою мінімальною продуктивністю (бит/с) має працювати модем, щоб забезпечити завантаження 1 сторінки гіпертекстової структури (www) з мережі Internet за 10 секунд? Сторінка містить 4000 символів (1 символ займає 8 біт) та 3 графічних об'єкта по 9 Кбайт кожний.

РОЗДІЛ 2
ПОНЯТТЯ АЛГОРИТМУ ТА СТВОРЕННЯ ПРОГРАМ

- ❖ Поняття алгоритму
- ❖ Властивості алгоритму
- ❖ Способи опису алгоритму
- ❖ Основні алгоритмічні структури
- ❖ Розробка програмного забезпечення

2.1. Алгоритм та основні алгоритмічні структури

Одним з базових понять інформатики є поняття алгоритму. Алгоритм вказує, які операції, пов'язані з обробкою даних, і в якій послідовності треба виконати, щоб отримати розв'язок задачі. Алгоритм розрахований на певного виконавця, з погляду котрого вказівки мають бути елементарними, тобто такими, що можуть бути виконані безпосередньо, без подальшого тлумачення. Слово «алгоритм» походить від назви латинського перекладу трактату арабського математика IX століття Абу Абдуллаха Мухаммеда ібн Муси Аль-Хорезмі (рис. 2.1) («Трактат Аль-Хорезмі про арифметичне мистецтво індусів»).

Рис.2.1. Математик Аль-Хорезмі

Сучасне формальне визначення алгоритму було дане в 30-50-х роках XX століття в роботах Черча (тезис Черча – Тюринга), Н. Винера, А.А. Маркова.

Алгоритм – система точно сформульованих правил, що визначає процес перетворення припустимих початкових даних (вхідної інформації) у бажаний результат (вихідну інформацію) за кінцеву кількість кроків.

2.1.1. Властивості та способи опису алгоритму

Будь-який алгоритм передбачає наявність виконавця. Оскільки в інформатиці йдеться про розв'язування задач за допомогою комп'ютера, то виконавцем є комп'ютер [6]. Тому в інформатиці алгоритм – це скінченна однозначна послідовність точно визначених кроків або дій, яка забезпечує розв'язування задачі і для виконання якої потрібний скінченний об'єм оперативної пам'яті і скінченний час.

Виконавець алгоритму – це деяка абстрактна або реальна технічна, біологічна або біотехнічна система, здатна виконувати дії, що передбачені алгоритмом.

Виконавцем алгоритму може бути людина, тварина і автоматичний пристрій. В інформатиці універсальним виконавцем алгоритмів є комп'ютер.

Виконавця характеризують:

♦ середовище (або обстановка);

♦ система команд – кожен виконавець може виконувати команди тільки з деякого певного списку. Для кожної команди повинні бути задані умови її застосування (у яких станах середовища може бути виконана команда) і описані результати її виконання;

♦ елементарні дії – прості операції, які виконавець здійснює після виклику відповідної команди;

♦ відмови – неможливість виконання тієї або іншої дії, якщо команда викликається в разі неприпустимого для неї стану середовища.

Зазвичай виконавець нічого не знає про мету алгоритму. Він

виконує всі отримувані команди, не ставлячи питань «чому» і «навіщо».

Алгоритмічний процес – це процес послідовного перетворення конструктивних об'єктів (слів, чисел, пар слів, пар чисел, речень тощо), дискретними «кроками». Кожен крок полягає в зміні одного конструктивного об'єкта іншим.

В алгоритмі відбиваються логіка та спосіб формування результатів рішення із вказівкою необхідних розрахункових формул, логічних умов, співвідношень для контролю вірогідності вихідних результатів. В алгоритмі обов'язково повинні бути передбачені всі ситуації, які можуть виникнути в процесі рішення комплексу завдань.

Алгоритм має задовольняти певним вимогам, серед яких потрібно виділити найважливіші.

♦ *Визначеність* – кожен крок алгоритму має інтерпретуватися виконавцем однозначно.

♦ *Результативність* – за скінченну кількість кроків алгоритм має приводити до розв'язання задачі або зупинятися через неможливість її розв'язати з тієї або іншої причини (наприклад, обчислення ірраціонального числа π).

♦ *Дискретність* – кроки обчислювального процесу мають бути відокремлені один від одного, тобто виконання кожного чергового кроку алгоритму повинне розпочинатися лише після повного завершення попереднього кроку.

Масовість – алгоритм розробляється у загальному вигляді, тобто його можна застосувати не лише до окремої задачі, але і до деякого класу задач, що розрізняються лише вхідними даними. При цьому вхідні дані мають належати деякій області, яка називається *областю застосовності алгоритму.*

Ефективність – під час розв'язання задачі може використовуватися лише обмежений обсяг комп'ютерних ресурсів. Найчастіше аналіз

алгоритму (або, як говорять, аналіз складності алгоритму) полягає в оцінюванні часових витрат на розв'язування задачі залежно від «обсягу» вихідних даних.

Є декілька способів опису алгоритму: словесний опис послідовності дій, алгоритмічна мова, аналітичний опис у вигляді набору формул, псевдокод, графічний – у вигляді блок-схеми тощо. Формалізована система правил текстуального опису алгоритмів є одним із різновидів мов програмування. А мови, призначені для запису алгоритмічних структур, називаються *мовами структурного програмування.*

На практиці використовують такі способи представлення алгоритмів:

♦ *вербальний запис* – неформалізований запис алгоритму на природній мові, наприклад, рецепт приготування манної каші;

♦ *псевдокод* – напівформалізований опис алгоритму на базі однієї з мов програмування, які застосовуються в тих випадках, коли немає можливості викладати основи алгоритмізації з використанням ЕОМ;

♦ *алгоритмічна мова* – жорстко формалізований запис алгоритму, доступний для розуміння комп'ютером. Для розробки комп'ютерних програм використовуються інструментальні засоби, так звані мови програмування;

♦ *блок-схема* – найбільш наочна графічна форма представлення алгоритмів, що використовується професіоналами особливо в тих випадках, коли алгоритм володіє витонченою логікою виконання.

Приклад. Нехай квадратне рівняння $ax^2 + bx + c = 0$ задане дійсними коефіцієнтами a, b, c за умови, що $a \neq 0$. Розглянемо задачу обчислення дійсних коренів цього рівняння. Послідовність операцій, які необхідно виконати, є такою.

1. Прочитати коефіцієнти a, b, c.

2. Обчислити дискримінант $D = b^2 - 4ac$.

3. Якщо D>0, то $x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$; якщо D=0, то $x = \frac{-b}{2a}$ і написати це число, інакше написати «дійсних коренів немає».

Отже, у нас є задача – обчислити дійсні корені квадратного рівняння, заданого коефіцієнтами, і є опис процесу розв'язування задачі, або алгоритм. За цим описом ми виконуємо певну послідовність дій, тобто розв'язуємо задачу. Наведений приклад ілюструє словесний спосіб опису алгоритму.

Блок-схема містить блоки, позначені геометричними фігурами. Усередині блоків записують елементарні дії. Блоки з'єднуються стрілками – так задається послідовність дій. Стрілки не є обов'язковими, якщо їх напрямок відповідає просуванню «униз» і «праворуч». Кожній геометричній фігурі відповідає певний клас алгоритмічних інструкцій.

Зокрема, прямокутниками позначають *операторні блоки.* Операторний блок може мати декілька входів і тільки один вихід. Це забезпечує однозначність у визначенні послідовності виконуваних дій. Дії, що позначаються такими блоками, змінюють значення, форму подання чи розташування даних.

Ромбами позначають перевірку умови, від результату якої залежить вибір напрямку подальших обчислень. Тому блоки, що позначають ромбами, називають *умовними.* Оскільки результатом перевірки умови, записаної в умовному блоці, може бути значення «так» або «ні», тобто «істина» чи «хибність», блок має два виходи.

Алгоритм виконується у зовнішньому середовищі, де перебувають користувачі алгоритму. Від користувачів надходить інформація, яка в той чи інший спосіб оброблятиметься алгоритмом. Користувачі також повинні мати можливість отримати результати роботи алгоритму. Тому виникає потреба у блоках введення і виведення даних. Такі блоки позначаються паралелограмами.

Заокругленими прямокутниками позначають початок та кінець алгоритму. У початковому блоці немає входів, а кінцевому – виходу. Обмежень на геометричні розміри блоків не існує.

Перелік, найменування, позначення і розміри обов'язкових символів і функції, що відображаються ними, в алгоритмі і програмі обробки даних повинні відповідати позначенням, вказаним у Додатку 1.

Під час виконання умовних графічних позначень автоматизованим способом розміри геометричних елементів символів округлюють до значень, які визначені технічними можливостями пристроїв, що використовуються.

За допомогою алгоритму кожний конкретний результат отримують за скінченну кількість кроків із скінченної множини даних. Якщо для певних початкових даних процес виконання алгоритму завершується із отриманням результату, кажуть, що до таких даних *алгоритм застосовують*. Проте, в деяких ситуаціях процес виконання алгоритму для певних початкових даних продовжується необмежено. Кажуть, що до таких початкових даних *алгоритм не застосовують*.

Схеми алгоритмів, програм, даних і систем (далі – схеми) складаються з тих, що мають задане значення символів, короткого тексту, пояснення і ліній для сполучення.

Схеми можуть використовуватися на різних рівнях деталізації, до того ж кількість рівнів залежить від розмірів і складності завдання обробки даних. Рівень деталізації має бути таким, щоб різні частини і взаємозв'язок між ними були зрозумілі загалом.

Символи, призначені для використання в документації з обробки даних, і керівництво щодо умовних позначень для застосування їх у:

♦ схемах даних, що відображають шлях даних під час рішення завдань і визначають етапи обробки, а також різні носії даних, що застосовують;

♦ схемах роботи системи, що відображають управління операціями і потік даних в системі;

♦ схемах взаємодії програм, що відображають шлях активації програм і взаємодій з відповідними даними. Кожна програма в схемі взаємодії програм показується тільки один раз (у схемі роботи системи програма може зображатися більш ніж в одному потоці управління);

♦ схемах ресурсів системи, що відображають конфігурацію блоків даних, які потрібні для вирішення завдання або набору завдань.

Є три елементарні алгоритмічні структури: послідовності, розгалуження та повторення. Всі інші алгоритмічні структури утворюються з елементарних шляхом заміни операторних блоків елементарними структурами. *Алгоритмічна структура послідовності* – це послідовність двох операторних блоків. Така структура дає вказівку виконувати одну інструкцію після іншої. Алгоритмічні структури розгалуження та повторення детально розглядатимуться в двох наступних розділах.

2.1.2. Алгоритмічна структура розгалуження

Алгоритмічна структура, що дозволяє виконавцеві алгоритму вибрати сценарій подальших дій залежно від істинності певного умовного твердження, називається *розгалуженням.* На блок-схемі (рис. 2.2) структури розгалуження позначають ромбами. Дві стрілки, які відгалужуються від ромба, позначені словами «Так» і «Ні». Якщо записане всередині ромба умовне твердження є істинним, виконують дії, на які вказує стрілка, позначена словом «Так». Якщо це твердження є хибним, виконують дії, на які вказує стрілка, позначена словом «Ні».

Є декілька різновидів структури розгалуження. Структура, використана в алгоритмі обчислення коренів квадратного рівняння, є

альтернативним розгалуженням. Альтернативне розгалуження припускає вибір виконавцем одного з двох можливих сценаріїв подальших дій залежно від істинності деякого умовного твердження. Крім альтернативного розгалуження є ще розгалуження у формі *множинного вибору альтернатив.* За множинного вибору може існувати більше двох сценаріїв дій виконавця. Вибір сценарію зумовлюється значенням деякого виразу.

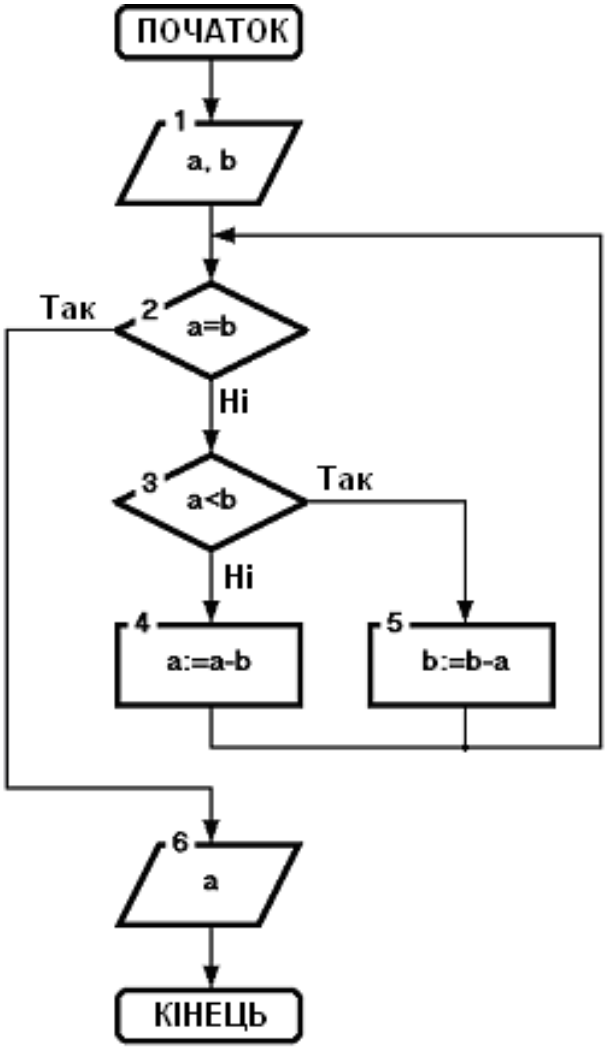


Рис. 2.2. Блок-схема запису алгоритму Евкліда знаходження НСД для двох натуральних чисел

2.1.3. Алгоритмічна структура повторення

Розглянемо задачу знаходження найбільшого спільного дільника двох натуральних чисел, застосувавши для її розв'язання алгоритм Евкліда.

Приклад. Використовуючи структуру повторення, опишемо алгоритм Евкліда для знаходження найбільшого спільного дільника (НСД) двох натуральних чисел, використавши вербальний запис алгоритму.

1. Задати два числа *a* та *b*.

2. Якщо числа рівні, то узяти перше з них як відповідь і припинити виконання алгоритму, інакше йти до пункту 3.

3. Визначити більше з двох чисел.

4. Замінити більше число на різницю між більшим і меншим числами.

5. Йти до пункту 2.

6. Вивести значення НСД.

Блок-схема програми являє собою структурний план, відповідно до якого буде здійснюватися розрахунок необхідних у завданні параметрів (рис. 2.2).

Алгоритм Евкліда вкрай незручно записувати за допомогою вказівок присвоєння та розгалуження, оскільки кількість таких вказівок залежить від значень *a* і *b,* тобто не є відомою наперед. Ефективний спосіб розв'язання цієї задачі полягає у застосуванні структури *повторення* (яка називається ще *циклічною структурою).* Алгоритмічна структура повторення дає виконавцеві алгоритму вказівку повторювати деякі дії, поки певне умовне твердження істинне.

Твердження, істинність якого перевіряється під час виконання циклічної структури, на блок-схемі записується всередині ромба (як і у випадку структури розгалуження). Особливістю зображення циклічної

структури на блок-схемі є те, що одна зі стрілок повинна «повертатися назад», тобто має утворюватися замкнений «цикл» із блоків та стрілок. Такий цикл має містити умовний блок, в якому записано умову продовження повторення. Одна зі стрілок, що відгалужуються від цього блоку, має брати участь у циклі, а інша – вказувати на блок поза циклом.

Реалізувати алгоритм обчислення в пакеті MathCAD можна двома способами:

- вставляючи відповідні оператори або функції в текст документа MathCAD. Такий спосіб називається програмуванням у тексті документа;

- використовуючи так звані програми-функції, які містять конструкції, багато в чому подібні до конструкцій таких мов, як PASCAL або FORTRAN: оператори присвоювання, оператори циклів, умовні оператори тощо. Написання програм-функцій в MathCAD дозволяє вирішити задачі, які неможливо вирішити, використовуючи тільки оператори та функції MathCAD. Такий спосіб має назву програмуванням у програмі-функції.

Алгоритм рішення комплексу завдань і його програмна реалізація тісно взаємозалежні. Специфіка застосованих методів проектування алгоритмів і використаних при цьому інструментальних засобів розробки програм може вплинути на форму подання та зміст алгоритму обробки даних [7].

Під час розробки алгоритму необхідно застосовувати такі технологічні рекомендації, за дотримання яких алгоритм виходив би найбільш зрозумілим, а помилки під час запису алгоритму у вигляді програми були б найменш вірогідні. Відповідно до сучасних переконань у сьогоднішній технології розробки програм, які є однією із сторін методу структурного програмування, алгоритм повинен бути структурований.

2.2. Засоби створення програм

До засобів створення програм належать насамперед мови і системи програмування. Основна функція всіх мов програмування, крім машинної, полягає у тому, щоб надати програмісту засоби абстрагування від характеристик та особливостей апаратного забезпечення, на якому виконуватимуться програми. Системи програмування містять автоматизовані засоби розробки програм.

2.2.1. Класифікація мов програмування

Машинна мова – це «природна мова» певного комп'ютера, яка визначається під час проектування його апаратних засобів. Машинні мови важкі для людського сприйняття. Тому природним виявилося прагнення автоматизувати процес написання програм, для того щоб полегшити працю програміста, частково поклавши його роботу на саму машину. Програмісти почали використовувати більш звичну для людини символьну форму опису обчислень, а перетворення програм у машинні коди здійснювали за допомогою програм трансляції (від англ. translation – переклад), які називаються *асемблерами* (від англ. to assemble – збирати) [10].

Транслятор (англ. translator) – програма або технічний засіб, який виконує трансляцію програми з однієї мови програмування на іншу. Транслятор зазвичай виконує також діагностику помилок, формує словники ідентифікаторів, видає для друку тексти програми тощо. Транслятори поділяють на компілятори та інтерпретатори.

Компілятор (англ. Compiler від англ. to compile збирати в ціле) – комп'ютерна програма, що перетворює (компілює) програмний код,

написаний певною мовою програмування, на семантично еквівалентний код іншою мовою програмування. Що, зазвичай, є необхідним для виконання програми на машині, наприклад, на комп'ютері. Компілятором можна визначити програму або технічний засіб, що виконує компіляцію.

Мови програмування, у яких числове кодування команд було замінено їх символьним зображенням, називалися мовами символічного кодування, а системи програмування – системами символічного кодування (ССК). Нині такі мови перетворилися в досить потужні засоби програмування, названі асемблерами.

Під час написання програм мовами асемблерного типу в ролі засобів програмування використовують такі абстракції, як змінна та символьне зображення операцій, що дає можливість програмісту позбутися проблем, пов'язаних із формою зображення чисел, кодуванням операцій і розподілом пам'яті. Тепер перераховані вище проблеми має вирішувати програма-транслятор на основі інформації, яку їй передає розроблювач програм.

Для прискорення процесу програмування були розроблені *мови програмування високого рівня,* які дозволяли писати програми, за формою близькою до людської мови, та використовували загальноприйняту математичну нотацію. Перша з них з'явилася наприкінці 1950-х років і називалася Fortran. Назва була скороченням від слів FORmula TRANslation, що у перекладі означає трансляція формул.

Одночасно з мовами високого рівня розроблялися *транслятори (компілятори)* – програмні засоби, призначені для перекладу високорівневих програм у машинні. Досвід створення мов високого рівня та їх трансляторів з роками накопичувався. Зокрема, було розроблено математичні основи та технологію реалізації цих програмних засобів. На сьогоднішній день кількість мов програмування й трансляторів вимірюється уже тисячами і продовжує зростати.

2.2.2. Методологія розробки програм

Методологія структурного програмування – це сукупність методів проектування та написання програм за жорсткими правилами, дотримання яких підвищує продуктивність праці програмістів, поліпшує читабельність і полегшує процес тестування програм [10]. Витоки цієї методології сягають 60-х років XX ст., коли швидкими темпами почала зростати складність програм, а отже, постала потреба у методах подолання такої складності. Методологія структурного програмування ґрунтується на трьох методах: низхідного проектування, модульного програмування і структурування програм.

В основу методу *низхідного проектування* покладено *алгоритмічну декомпозицію*, згідно з якою велика задача поділяється на дрібніші підзадачі, які можна розв'язувати окремо. Алгоритми розв'язання підзадач розглядаються як цілісні алгоритмічні блоки, або підпрограми, іменами яких можна оперувати під час розв'язання загальної задачі.

Отже, програма, що розв'язує загальну задачу, спочатку розглядається як незалежний модуль. Згодом її поділяють на підпрограми, які декомпонують на підмодулі наступного рівня. Процес декомпозиції триває доти, доки не будуть отримані блоки, що є достатньо малими для їх безпосереднього кодування. При цьому керуючу програму проектують раніше, ніж реалізують її складові частини.

Таким чином, програма ієрархічно структурується і розробляється шляхом послідовного уточнення на кожному рівні ієрархії. В основу цього процесу, крім принципу ієрархічності, покладено принципи абстрагування, специфікації інтерфейсів і модульності.

Абстрагування – це спрощений опис системи, в якому зосереджують увагу на певних властивостях і деталях, а на інші не зважають. Вдалою є та абстракція, що підкреслює суттєві деталі і відкидає несуттєві [8]. Під час

низхідного проектування програми на верхніх рівнях абстракції деталі реалізації приховуються, а на нижніх рівнях їх описують конкретною мовою програмування.

Специфікація інтерфейсів – це формалізований опис входів, виходів і функцій, що мають бути реалізовані програмним модулем. Коли інтерфейс модуля специфіковано, можна спробувати в явному вигляді записати його код. Якщо це зробити неможливо, модуль поділяють на певну кількість невеликих підмодулів. Коли модуль не можна ані закодувати, ані декомпонувати, його вважають *модулем-заглушкою,* інтерфейс якого відомий, а реалізація – ні.

Одним із прийомів формалізованого підходу до низхідного проектування є *метод ієрархічних діаграм,* що позначається абревіатурою HIPO (від англ. Hierarchical Input Processing Output – діаграма входу, обробки, виходу). Згідно з цим методом структуру всієї програми подають у вигляді дерева, в якому підпрограми зображують вузлами, а їх виклики – ребрами.

Мова програмування – формальна мова представлення програм для системи програмування.

Формальна мова – множина скінченних послідовностей символів, які описуються правилами певного виду, які називають граматикою, або синтаксисом мови.

У тому випадку, коли кожному слову формальної мови зіставляють його семантику (сенс, значення, інтерпретація), формальну мову називають *інтерпретованою*.

Мови програмування низького рівня орієнтовані на конкретний тип процесора, враховують його особливості і називаються асемблерами. Для кожного типу процесора існує своя мова асемблера, тому для перенесення програми на асемблері на іншу апаратну платформу її потрібно майже цілком переписати.

Мови низького рівня, зазвичай, використовують для написання невеликих системних застосувань, драйверів пристроїв, модулів стиків з нестандартним обладнанням, коли найважливішими вимогами є компактність, швидкодія і можливість прямого доступу до апаратних ресурсів. До мови низького рівня належить мова «Асемблер».

Мови програмування високого рівня близькі до природної англійської мови людини. Особливості конкретних комп'ютерних архітектур у них не враховуються, тому створені програми легко переносяться з комп'ютера на комп'ютер, де встановлено транслятор цієї мови. До мови високого рівня належать мови «Фортран», «Кобол», «Алгол», «Pascal», «Java», «C», «C++», «C#», «Objective C», «Smalltalk».

2.2.3. Технологія створення програм

Розробка програми починається з ставлення задачі, яку пропонує замовник. Іноді аналіз і уточнення задачі дають можливість формалізувати її ставлення, як результат з'являється математично точний і однозначний опис задачі. Після уточнення ставлення задачі починається *проектування програми.* Зазвичай, в задачі можна виділити декілька *підзадач* і описати процес їх розв'язування окремо. Відповідно й алгоритм складається зі зв'язаних та узгоджених між собою частин, які описують процес розв'язання підзадач. У початковому алгоритмі дії подано в абстрактному вигляді, далекому від того, що може виконувати комп'ютер. Алгоритм уточнюють декілька разів і надають йому вигляд, за яким легко написати програму або її частину [10].

Написання програми або окремих її частин прийнято називати *кодуванням,* або *розробкою.* Найчастіше програму записують однією з мов високого рівня, але іноді деякі її частини записують різними мовами. Далі програму перекладають (транслюють) на машинну мову (зазвичай

частинами). Під час кодування програмісти можуть припускатися помилок. Процес виявлення й виправлення помилок називається *налагодженням* програми. Він дозволяє виявити помилки перелічених нижче типів.

1. Помилки, пов'язані з порушенням правил граматики в тексті програми, написаної мовою високого рівня. Їх можна виявити у процесі трансляції, тому вони називаються *помилками часу трансляції* (compiler error).

2. Помилки, що виявляються під час виконання робочої програми. Вони можуть виникати, наприклад, в результаті переповнення розрядної сітки чи під час спроби видобути квадратний корінь із від'ємного числа. Такі помилки називаються *помилками часу виконання* (run time error).

3. Помилки, що не виявляються ні під час трансляції, ні під час виконання програми. Це змістові помилки, пов'язані з некоректністю логічних умов, неправильним використанням розрахункових формул і т. ін., їх називають *семантичними.*

4. Помилки у вихідних даних.

Налагодження – це процес багаторазового виконання програми з різними варіантами даних, які вона має обробляти. Дані спеціально добирають таким чином, щоб можна було виявити якнайбільше помилок, якщо такі існують. Ця цілеспрямована перевірка працездатності програми називається *тестуванням.* Тестування не гарантує відсутності помилок у програмі, а лише дозволяє виявити деякі з них. Чим ретельніше проводиться тестування, тим більше помилок виявляється та виправляється.

Після налагодження програма проходить *дослідно-виробничу експлуатацію,* для якої необхідно розробити супровідні документи під назвами «Керівництво розробника програми» і «Керівництво користувача», які описують побудову та використання програми. Перший

документ дає можливість виправляти помилки під час експлуатації програми та розвивати її надалі, а у другому пояснюється, як використовувати програму.

Проте може з'ясуватися, що під час проектування програми було обрано не найкращий алгоритм, через що програма виконується надто повільно або витрачає забагато ресурсів пам'яті, тобто є неефективною. До програми нерідко вносяться зміни, які вимагають повторного кодування і налагодження. Якщо у замовника з'являються нові ідеї, необхідно заново ставити задачу та проектувати програму.

З метою розробки ефективних з точки зору використаних ресурсів програм і прискорення процесу їх проектування були створені технології, тобто системи методів, які дозволяють «не робити зайвих кроків» на кожному з етапів, від аналізу задачі до налагодження програми [10]. У 70-х роках минулого століття домінувала *технологія структурного програмування* – система правил створення програм, що характеризуються ясністю, простотою тестування та налагодження, легкістю модифікації. Починаючи з 90-х років, ключовою є *технологія об'єктно-орієнтованого програмування* – програмування на основі абстрактних типів даних. Застосування різних технологій потребує постійного удосконалення інструментів, які дозволяють створювати програми швидко, якісно й економічно. Такі інструменти називаються *системами програмування* і реалізують дуже важливий *принцип повторного використання коду* завдяки створенню модулів і компонентів.

2.2.4. Перетворення програми і система програмування

Розглянемо, як із програми, написаної мовою високого рівня, утворюється інша – машинна. Програму (вихідний текст) за допомогою спеціальної програми (вона називається *текстовим редактором*)

найчастіше записують на диск у вигляді вихідного файлу (рис. 2.3). Програма може складатися з кількох вихідних файлів – у великих програмах їх може нараховуватися десятки [10].

Під час роботи транслятора прочитується вихідний файл і створюється його машинний еквівалент – *об'єктний код.* Процес виконання програми-транслятора називається *трансляцією,* або *компіляцією* вихідного тексту.

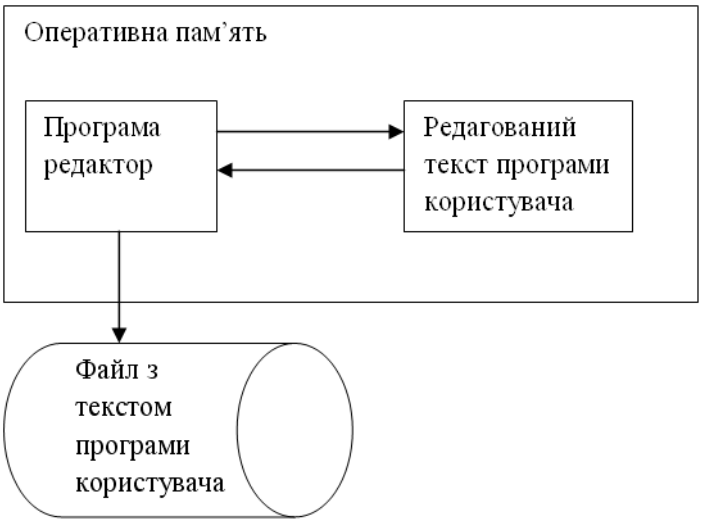


Рис. 2.3. Схема створення тексту програми

Зазвичай, об'єктний код програми містить далеко не всі необхідні команди – програма може складатися з частин або включати підпрограми з бібліотек. Об'єктний код обробляється ще однією програмою – *редактором зв'язків,* або *компонувальником,* яка «збирає» (компонує) повний код програми і записує (завантажує) його або в оперативну пам'ять, або на диск у вигляді готового до виконання файлу (рис. 2.4), який можна завантажити пізніше.

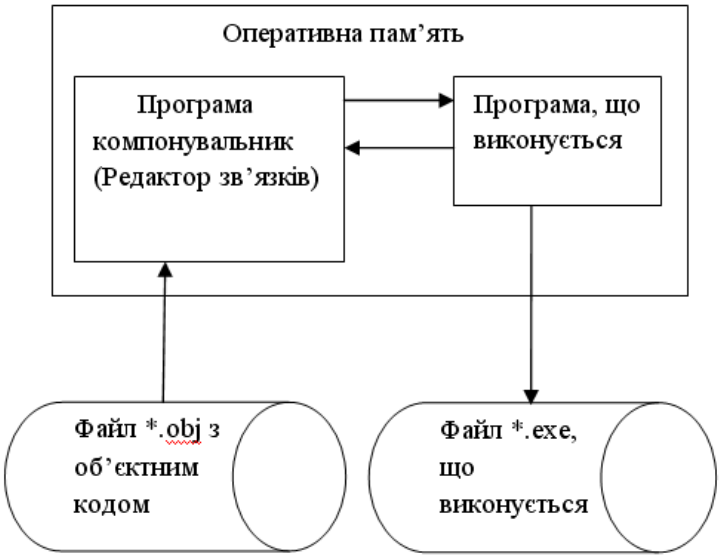


Рис. 2.4. Схема створення коду програми

Інтерпретатор на відміну від транслятора не створює машинну програму. Вхідні дані для інтерпретатора – це високорівнева програма й дані, що мають зчитуватися під час її виконання (рис. 2.5). *Інтерпретація* програми полягає в тому, що дії, задані програмою, відразу виконуються. Зазвичай інтерпретація вихідної програми відбувається повільніше, ніж виконання відповідної машинної програми.

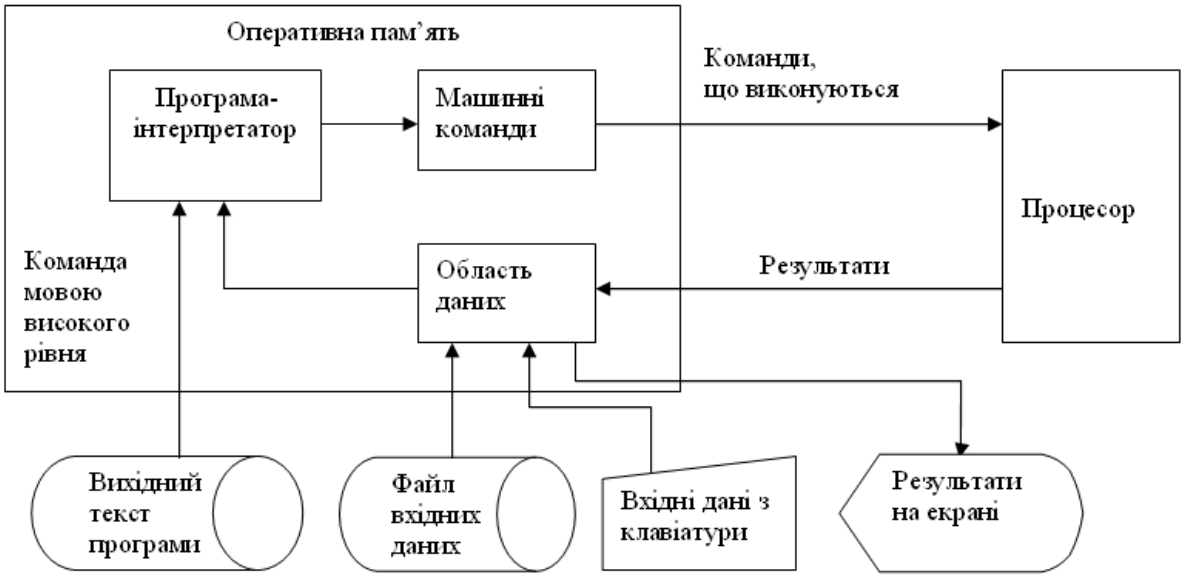


Рис. 2.5. Інтерпретація високорівневої програми

Інтерпретатор мови програмування (interpreter) – програма чи технічні засоби, необхідні для виконання інших програм, вид транслятора, який здійснює пооператорну (покомандну) обробку, перетворення у машинні коди та виконання програми або запиту (на відміну від компілятора, який транслює у машинні коди всю програму без її виконання). Інтерпретатори можуть працювати як з вихідним кодом програми, написаним мовою програмування, так і з байт-кодом.

Ще один спосіб обробки вихідної програми поєднує трансляцію й інтерпретацію. Програма перекладається (транслюється) не в машинні команди, а в деяке проміжне зображення, що потім інтерпретується. Такий підхід реалізовано, зокрема, в мові Java, яка швидко знайшла собі багатьох прихильників серед програмістів.

Інтерпретація програми здійснюється за допомогою такого інструменту як *налагоджувач.* Він забезпечує інтерпретацію вихідної програми невеликими порціями (кроками) і дає можливість побачити результати виконання кожного кроку. Це полегшує пошук помилки (її локалізацію) у вихідній програмі.

Описані засоби (текстовий редактор, транслятор та (або) інтерпретатор, компонувальник, завантажувач і налагоджувач) разом утворюють *систему програмування,* або *інтегроване середовище розробки* (Integrated Development Environment, IDE). Крім них до складу IDE входить бібліотека стандартних підпрограм, які можна використовувати під час створення програми.

Запис початкових текстів програм за допомогою мов програмування зручний для розуміння і редагування людиною. Цьому, зокрема, допомагають коментарі, допустимі в синтаксисі більшості мов. Для виконання на комп'ютері готовий текст програми перетворюється (компілюється) в машинний код.

Деякі мови програмування дозволяють обходитися без попередньої компіляції програми і переводять її в інструкції машинного коду безпосередньо під час виконання [10]. Цей процес називається динамічною компіляцією. Він дозволяє домогтися більшої переносимості програм між різними апаратними і програмними платформами за збереження багатьох плюсів компіляції.

Програми, які інтерпретуються операційною системою або спеціальними програмами-інтерпретаторами, і для яких, зазвичай, не застосовують процес компіляції, називають скриптами або «сценаріями».

Початкові тексти комп'ютерних програм у більшості мов програмування складаються із списку інструкцій, що точно описують закладений алгоритм. Подібний підхід у програмуванні називають імперативним. Проте застосовуються й інші методології програмування. Наприклад, опис початкових і необхідних характеристик оброблюваних даних та надання вибору відповідного алгоритму рішення спеціалізованій програмі-інтерпретатору. Такий підхід називається декларативним програмуванням.

Висновки

Алгоритм – це набір елементарних вказівок, розташованих у певній послідовності. Алгоритм розрахований на певного потенційного виконавця, з погляду якого вказівки повинні бути елементарними, тобто такими, що можуть бути виконані безпосередньо, без подальшого тлумачення.

Розгалуження – алгоритмічна структура, яка дозволяє виконавцеві алгоритму вибрати сценарій подальших дій залежно від виконання певних умов.

Різновидами розгалуження є альтернативне розгалуження, що

припускає вибір виконавцем одного з двох можливих сценаріїв подальших дій, та множинний вибір альтернатив, за якого таких сценаріїв може бути більше двох.

Алгоритмічна структура повторення дає виконавцеві алгоритму вказівку повторювати деякі дії доти, доки певне умовне твердження є істинним.

Етапи розробки програмного забезпечення – це ставлення задачі, аналіз задачі та побудова її моделі, вибір або розробка алгоритму розв'язання задачі, кодування, налагодження та тестування, дослідно-виробнича експлуатація і супровід програмного забезпечення.

Мови програмування високого рівня дозволяють записувати програми зрозумілими для людини термінами.

Переклад програми, що записана мовою високого рівня, на машинну мову називається трансляцією і здійснюється програмою-транслятором.

Програма-інтерпретатор виконує визначені програмою дії, не транслюючи програми у машинний код.

Редактор зв'язків, або компонувальник, створює повний код програми шляхом поєднання об'єктного коду її окремих модулів.

Розробка програмного забезпечення – це складний ітеративний процес, що, як правило, складається з таких етапів: ставлення задачі, аналіз задачі та побудова її моделі, вибір або розробка алгоритму розв'язання задачі, кодування, налагодження та тестування, дослідно-виробнича експлуатація і супровід програмного забезпечення.

Контрольні запитання та завдання

1. Дайте визначення терміна «блок-схема».
2. Дайте визначення терміна «псевдокод».
3. Назвіть форми представлення алгоритмів.
4. Які основні властивості алгоритму ви знаєте?
5. Дайте визначення терміна «виконавець алгоритму».
6. Для чого призначено алгоритм?
7. Що таке алгоритм? Наведіть приклади.
8. Назвіть основні вимоги до алгоритмів і поясніть суть кожного з них.
9. Які критерії порівняння алгоритмів вам відомі?
10. Що таке програма?
11. Чим мова програмування високого рівня відрізняється від машинної мови?
12. Опишіть процес створення програми.
13. Що таке транслятор?
14. Чим відрізняється компіляція від інтерпретації?
15. Що таке програма?

Вправи

1. Побудувати блок-схему для визначення типу трикутника (рівносторонній, рівнобедрений, різносторонній) за довжинами його сторін. Якщо трикутник не можна побудувати, слід вивести повідомлення.

2. Побудувати блок-схему алгоритму піднесення цілого числа до цілої степені.

3. Побудувати блок-схему алгоритму переведення цілого числа з десяткової системи числення до будь-якої іншої.

4. Побудувати блок-схему алгоритму переведення числа з будь-якої системи числення у десяткову.

5. Побудувати блок-схему алгоритму обчислення факторіала натурального числа.

6. Побудувати блок-схему алгоритму визначення необхідної кількості повітря для системи вентиляції пташника, за умов видалення з приміщення шкідливих газів, надлишкової вологи та надлишкового тепла. Необхідно розробити алгоритм визначення найбільшого значення з трьох зазначених умов, які впливають на необхідну кількість повітря для системи вентиляції пташника.

РОЗДІЛ 3
ПРОГРАМУВАННЯ В МАТЕМАТИЧНОМУ ПАКЕТІ MATHCAD

- Використання математичного пакета MathCAD в інженерних розрахунках
- Основи програмування в MathCAD
- Основні оператори програмування в MathCAD
- Додаткові оператори програмування в MathCAD
- Модульне програмування в MathCAD

3.1. MathCAD в інженерних розрахунках

Математичне середовище MathCAD – це програмний пакет для роботи з формулами, числами, текстами та графіками. Він дозволяє отримувати числові розв'язки простих та складних функцій, які містять всі математичні дії, проводити символьне обчислення виразів, створювати програмні алгоритми. Однією із найбільш суттєвих переваг цього середовища є можливість запису математичних виразів у звичній, максимально наближеній до прийнятої в математиці формі.

3.1.1. Прості обрахунки в MathCAD

MathCAD працює з *документами*. З погляду користувача, документ – це чистий аркуш паперу, на якому можна розмістити блоки трьох основних типів: математичні вирази, текстові фрагменти й графічні області.

У середовищі MathCAD є низка особливостей, які треба враховувати під час запису алгоритму будь-якого обчислення:

- символи, що використовують в розрахунках (крім текстових зон), записуються за латинської (англійської) розкладки клавіатури;
- десяткові значення чисел відокремлюють крапкою;
- робочий документ читають зверху вниз, зліва направо, що в свою чергу вимагає чітко визначеної послідовності розміщення на робочому полі символів та операцій.

Якщо розрахунок проводять за визначеною функцією, яка містить сталі та змінні, то має бути така послідовність запису:

- спочатку записують символи, які входять до виразу, та відповідні їм числові значення;
- потім записують діапазон або конкретні значення змінних;
- нижче записується вираз у загальному вигляді;
- під ним виводять результати обчислень;
- нижче, за необхідності, виводять шаблон для побудови графіка на основі записаного виразу.

У разі порушення зазначеного порядку розміщення даних або операцій червоним кольором виокремлюють ті місця та символи, де допущена помилка [9].

До основних елементів *математичних виразів* MathCAD відносяться *типи даних, оператори, функції* й *керуючі структури.*

Оператори – елементи MathCAD, за допомогою яких можна створювати математичні вирази. До них, наприклад, відносять символи арифметичних операцій, знаки обчислення сум, добутків, похідної та інтеграла тощо.

Оператор визначає:

- дію, що має виконуватися за наявності тих або інших значень операндів;
- скільки, де і які операнди мають бути введені в оператор.

Операнд – число або вираз, на яке діє оператор. Наприклад, у виразі 5! + 3 число 3 і вираз 5! – операнди оператора + (плюс), а число 5 – операнд оператора факторіал (!). Після вказівки *операндів* оператори стають блоками, що виконують у документі. У Додатку 2 цього посібника наведено список операторів, що найчастіше використовують.

Типи даних – це числові константи, звичайні й системні змінні, масиви (вектори й матриці) і дані файлового типу.

Константами називають іменовані об'єкти, що зберігають деякі значення, які не можуть бути змінені. Змінними є іменовані об'єкти, що мають деяке значення, яке може змінюватися за ходом виконання програми. Тип змінної визначається її значенням; змінні можуть бути числовими, строковими, символьними й т.д. Імена констант, змінних та інших об'єктів називають ідентифікаторами. Ідентифікатори в MathCAD являють собою набір латинських або грецьких букв і цифр [12].

У MathCAD є невелика група особливих об'єктів, які не можна віднести ні до класу констант, ні до класу змінних, значення яких визначені відразу після запуску програми. Їх вважають *системними змінними*, що мають визначені системою початкові значення. Для того, щоб відкрити блок «математика», необхідно відкрити меню **View** та кликнути мишею на вкладку **Math**.

Звичайні змінні величини відрізняються від системних тим, що вони мають бути попередньо *означені* користувачем, тобто їм необхідно хоча б один раз *присвоїти значення*. Як *оператор присвоювання* використовують знак «**:=**», тоді як знак «=» відведений для *висновку значення* константи або змінної.

Якщо змінній присвоюється початкове значення за допомогою

оператора «:=», таке присвоювання називають *локальним*. До цього присвоювана змінна неозначена і її не можна використати. Однак за допомогою знака «≡» можемо зробити *глобальне* присвоювання. У вікні документа MathCAD (рис. 3.1) наведено приклад означення змінних. MathCAD прочитує весь документ двічі ліворуч, праворуч і зверху вниз. Під час першого проходу виконуються всі дії, запропоновані глобальним оператором присвоювання (≡), а другого – виконуються дії, запропоновані локальним оператором присвоювання (:=), і відображаються всі необхідні результати обчислень (=).

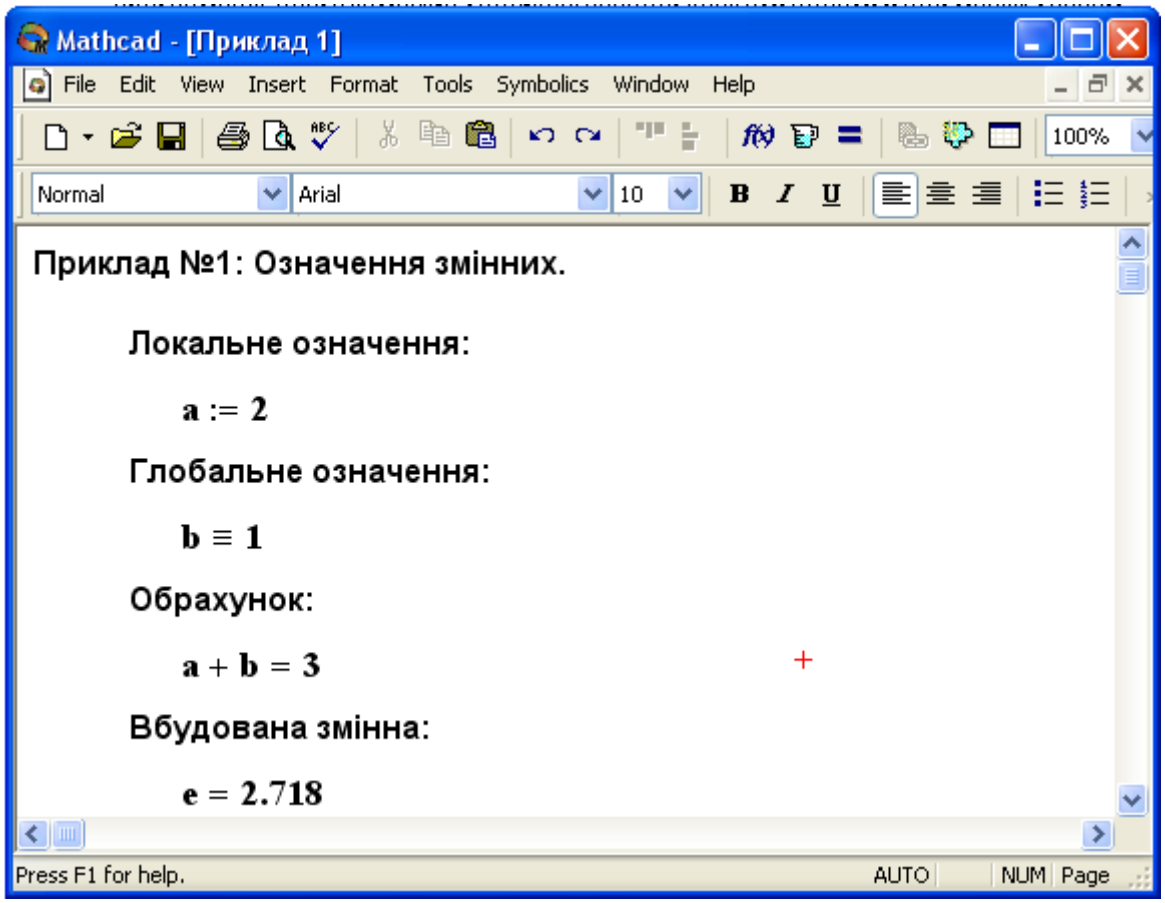


Рис. 3.1. Означення змінних у вікні документа MathCAD

Існують також жирний знак рівності «=» (комбінація клавіш Ctrl + =), що використовується, наприклад, як оператор наближеної

рівності під час рішення систем рівнянь, і символьний знак рівності «→» (комбінація клавіш Ctrl + **.**).

Дискретні аргументи – особливий клас змінних, який у пакеті MathCAD найчастіше замінює *керуючі структури*, що називаються циклами. Ці змінні мають низку фіксованих значень, або цілочисельних (1 спосіб), або у вигляді чисел із певним кроком, що змінюються від початкового значення до кінцевого (2 спосіб).

1. *Name* := *Nbegin* .. *Nend*,

де *Name* – ім'я змінної, *Nbegin* – її початкове значення, *Nend* – кінцеве значення, **..** – символ, що вказує на зміну змінної у заданих межах (вводиться клавішею **;**). Якщо *Nbegin* < *Nend*, то крок змінної дорівнює +1, інакше -1. У вікні документа MathCAD (рис. 3.2) наведено приклад означення дискретного аргументу.

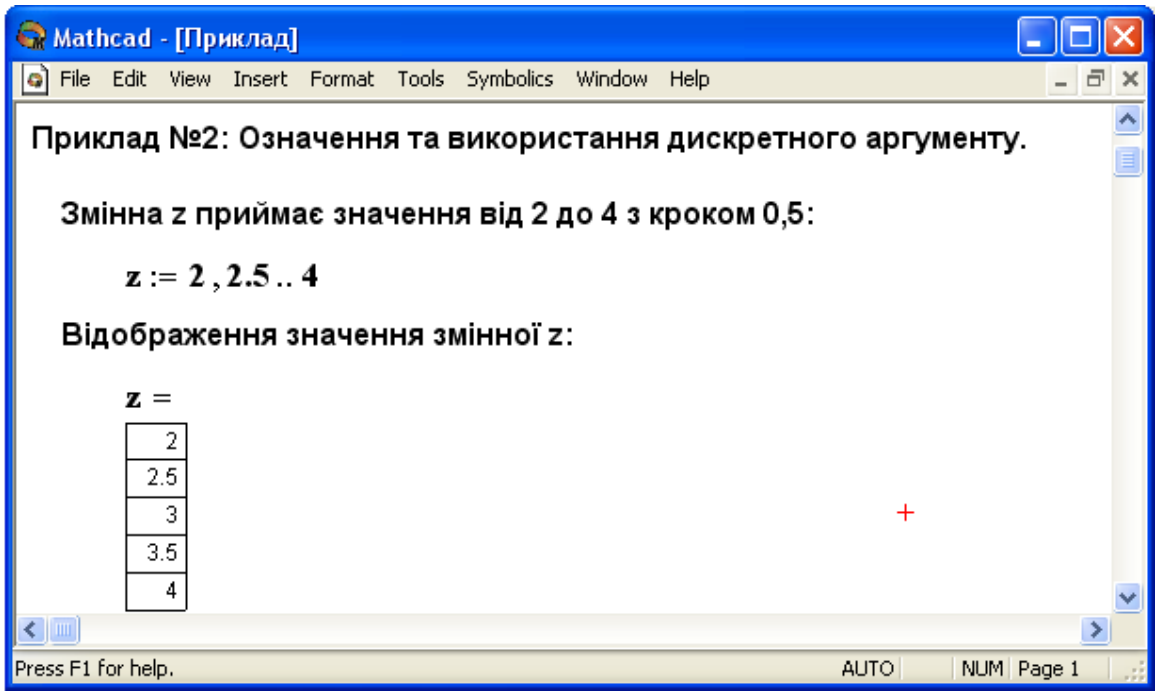


Рис. 3.2. Означення та використання дискретного аргументу у вікні документа MathCAD

2. *Name* := *Nbegin*, (*Nbegin* + *Step*) .. *Nend*

Тут *Step* – заданий крок змінної (він має бути додатнім, якщо *Nbegin* < *Nend*, або від'ємним у протилежному випадку).

Дискретні аргументи значно розширюють можливості MathCAD, дозволяючи виконувати багаторазові обчислення або цикли з повторюваними обчисленнями, формувати вектори й матриці.

Масив (у деяких мовах програмування також таблиця, ряд) - іменований набір однотипних змінних, розташованих у пам'яті безпосередньо один за одним (на відміну від списку), доступ до яких здійснюється через індекс.

У пакеті MathCAD використовують масиви двох найпоширеніших типів. У вікні документа MathCAD (рис. 3.3) наведено приклад означення та використання дискретного аргументу:

- одновимірні (вектори);
- двовимірні (матриці).

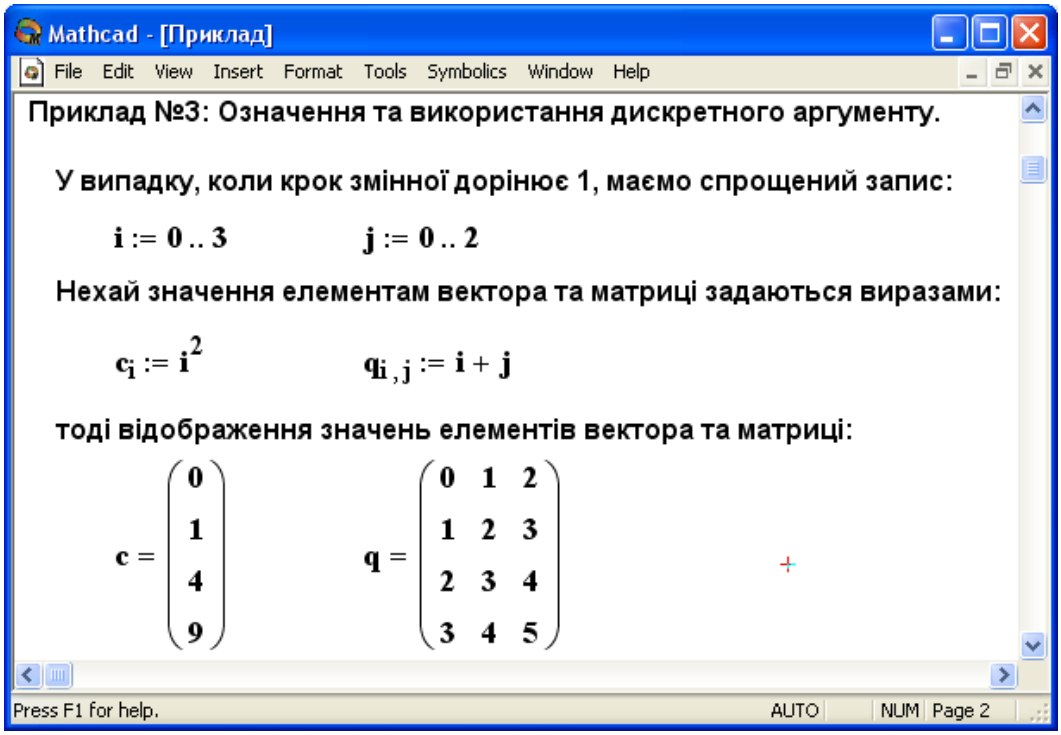


Рис. 3.3. Означення та використання дискретного аргументу у вікні документа MathCAD

Індекс масиву – ціле число, або значення типу, що зводиться до цілого, яке вказує на конкретний елемент масиву. Індекси можуть мати тільки цілочисельні значення. Вони можуть починатися з нуля або

одиниці відповідно до значень системної змінної ORIGIN (Додаток 2).

Вектори й матриці можна задавати різними способами:

- необхідно відкрити меню **Insert** та кликнути мишею на вкладку **Matrix**, або комбінацією клавіш Ctrl + M, або кликнути на кнопці  панелі **Matrix,** заповнивши масив порожніх полів;

- з використанням дискретного аргументу, коли є деяка залежність для обчислення елементів через їхні індекси.

Змінна величина називається функцією незалежних змінних, якщо кожній сукупності значень змінних із деякої області відповідає одне певне значення величини із множини.

Слід особливо зазначити різницю між *аргументами* й *параметрами* функції. Змінні, що зазначені в дужках після імені функції, називаються *параметрами* і замінюються у разі обчислення функції значеннями з дужок, тобто *аргументами*. У вікні документа MathCAD (рис. 3.4) наведено приклад означення та використання дискретного аргументу.

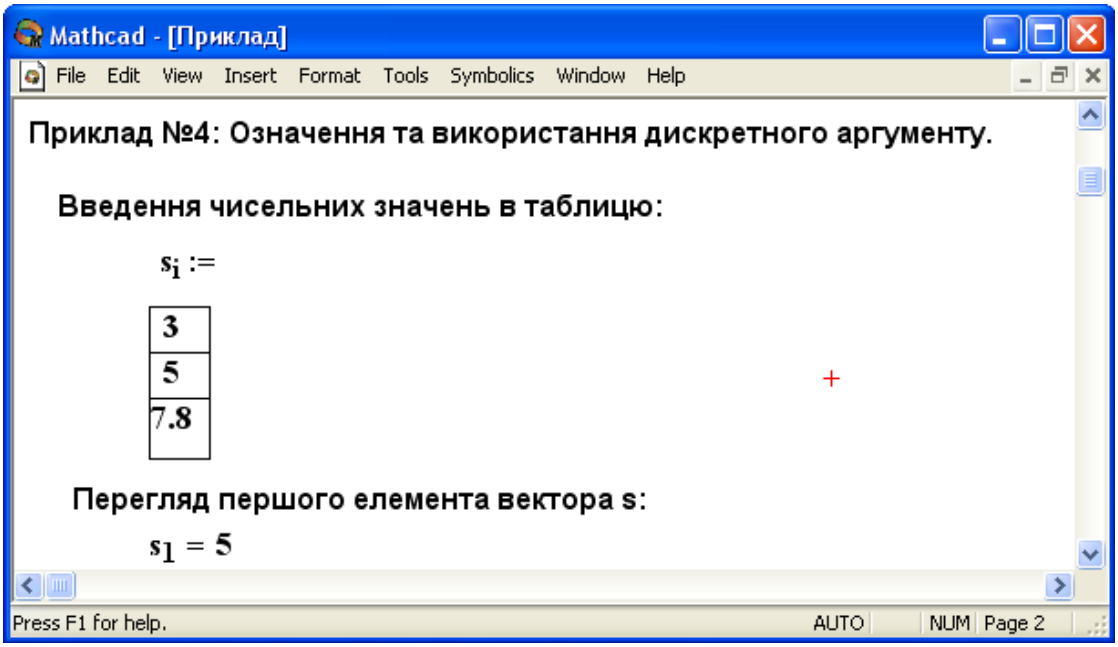


Рис. 3.4. Означення та використання дискретного аргументу у вікні документа MathCAD

Головною ознакою функції є *повернення значення*, тобто функція у відповідь на звертання до неї по імені із вказівкою її аргументів повертає своє значення. У вікні документа MathCAD (рис. 3.5) наведено приклад означення функції.

Функції в пакеті MathCAD можуть бути *вбудовані* (Додаток 2), тобто завчасно введені розроблювачами.

Існує три способи вставки вбудованої функції:

- необхідно відкрити меню **Insert** та кликнути мишею на вкладку **Function**;

- натисніть комбінацію клавіш Ctrl + E;

- кликніть кнопку  .

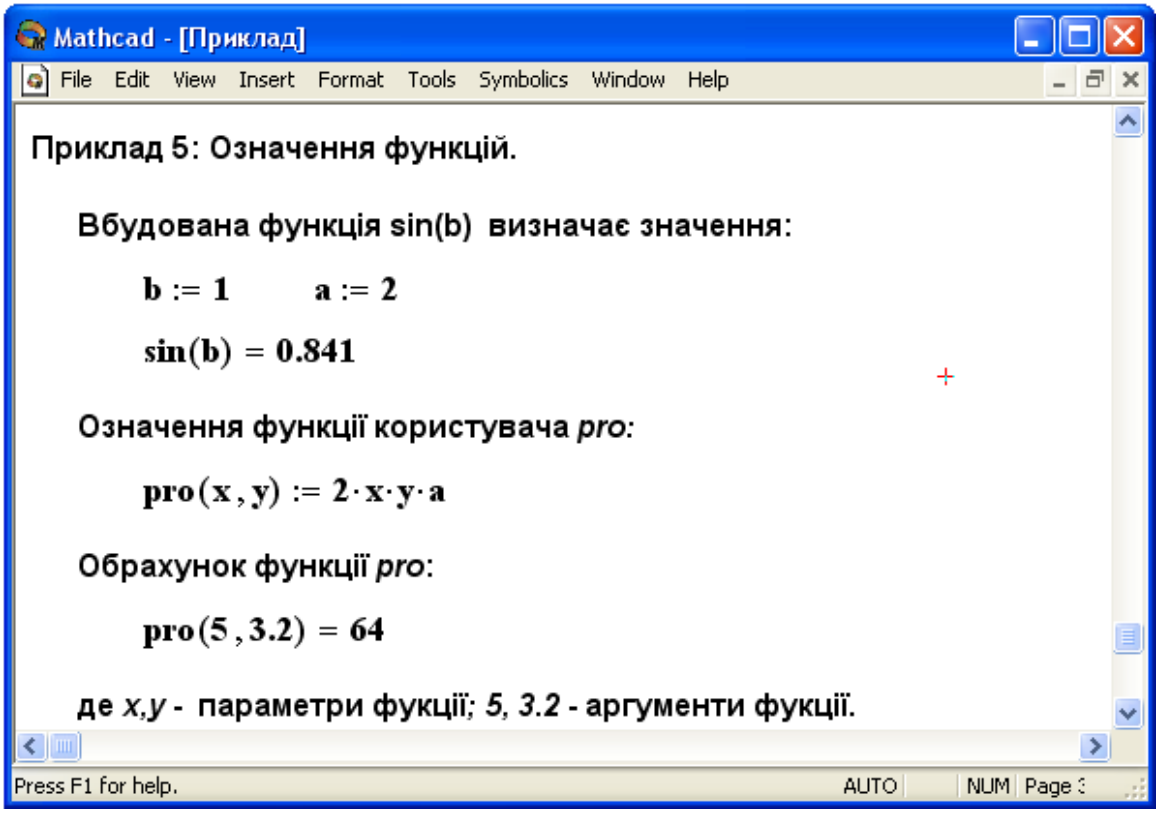


Рис. 3.5. Означення та використання дискретного аргументу у вікні документа MathCAD

Текстові фрагменти являють собою фрагменти тексту, які користувач хотів би бачити у своєму документі. Існують два види текстових фрагментів:

- *текстова область* призначена для невеликих фрагментів тексту – підписів, коментарів тощо. Вставляється за допомогою меню **Insert** та кликнути мишею на вкладку **Text Region** або комбінації клавіш Shift + " (подвійні лапки);

- *текстовий абзац* застосовують в тому випадку, якщо необхідно працювати з абзацами або сторінками. Вставляється за допомогою комбінації клавіш Shift + Enter.

Графічні області діляться на три основних типи – двовимірні графіки, тривимірні графіки й імпортовані графічні образи. Двовимірні й тривимірні графіки будуються самим MathCAD на підставі оброблених даних [13].

Для створення *декартового графіка* необхідно зробити наступні кроки:

- встановити курсор у порожньому місці робочого документа;

- відкрити меню **Insert**, кликнути мишею на вкладку **Graph** та вкладку **X-Y Plot,** або натиснути комбінацію клавіш Shift + @**,** або кликнути кнопку  панелі **Graph –** з'явиться шаблон декартового графіка;

- ввести у середній мітці під віссю *X* першу незалежну змінну, через кому – другу й так до 10, наприклад, *x*1, *x*2, …;

- ввести у середній мітці ліворуч від вертикальної осі *Y* першу незалежну змінну, через кому – другу й т.д., наприклад, *f*1(*x*1), *f*2(*x*2), …, або відповідні вирази;

- кликнути за межами області графіка, щоб побудувати графік.

Для форматування двовимірних графічних залежностей необхідно виконати наступні кроки:

- двічі кликнути лівою клавішею миші у полі графіка;

- у діалоговому вікні, що з'явиться, на закладці **X-Y Plot** задати тип координатних осей, наявність та щільність координатної сітки тощо;

- на закладці **Tracer** задати тип графіка, колір, товщину будь-якої з ліній, які побудовані на графіку, використовуючи списки системи.

Для отримання графічних залежностей між ознаками, значення яких задані як експериментальні точки, використовують форму запису значень ознак з нижнім індексом.

Розглянемо приклад, наведений на рис. 3.5**.** Необхідно визначити значення виразу $y = ax^2 + bx + c$ та побудувати графічну залежність y від x, коли сталі a, b та c приймають значення: $a = 3, b = -9, c = 2$. Змінна x набуває значень у діапазоні від 0 до 3 з кроком 0,5.

У MathCAD послідовність запису кроків алгоритму обчислення зазначеної функції залежить від способу запису значень змінної. Змінна може бути наведена у вигляді діапазону значень із певним кроком або у вигляді конкретних значень. Ту чи іншу форму алгоритму обчислення вибирають залежно від задачі, що розв'язують, та від доцільності наведення форми змінних.

Враховуючи зазначену вище послідовність алгоритму обчислення, запис розв'язку матиме такий вигляд:

1 Спосіб. У разі задавання конкретних значень змінної (рис. 3.6).

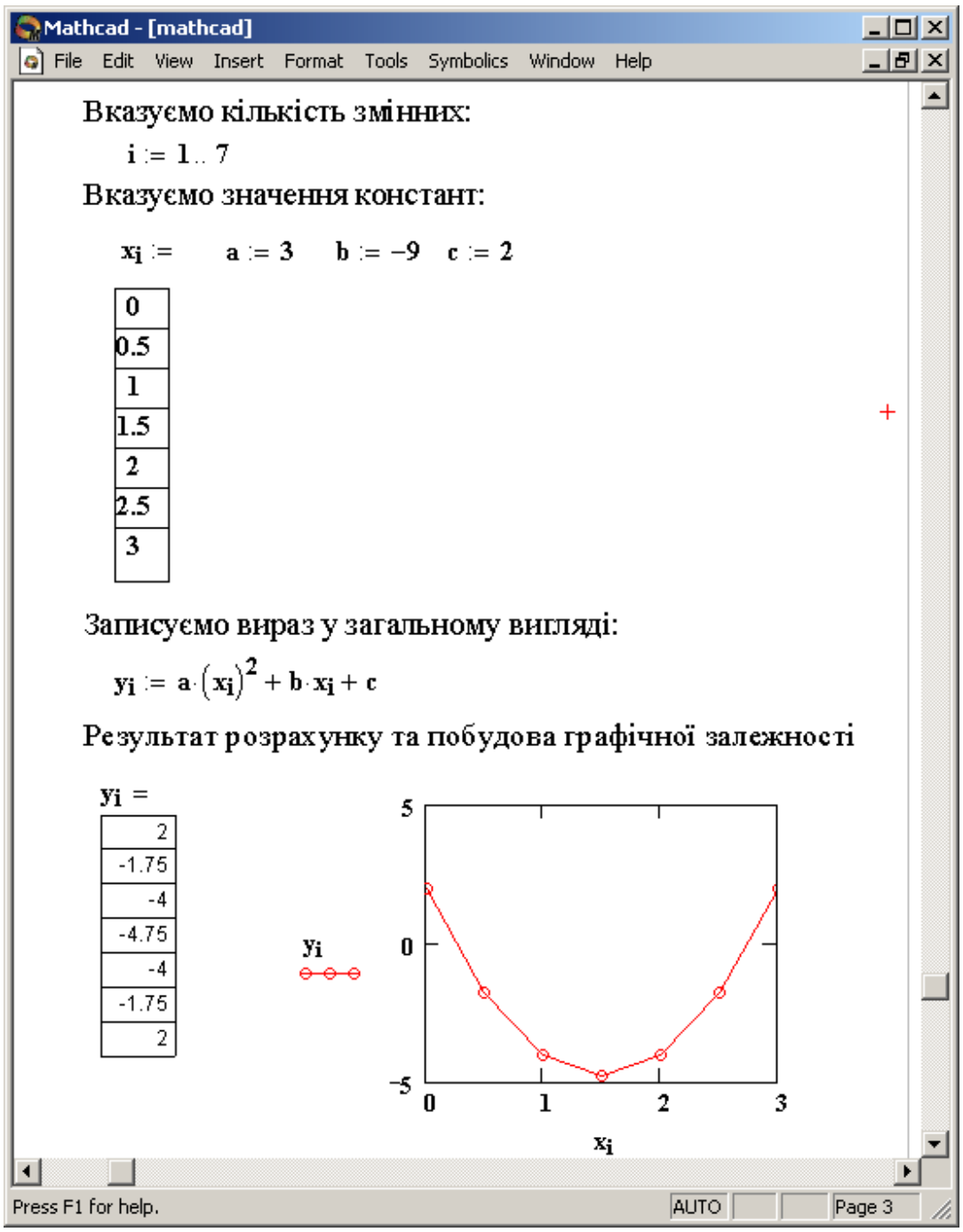


Рис 3.6. Задавання конкретних значень змінної аргументу
у вікні документа MathCAD

2 Спосіб. У разі задавання області змінних з вказаним кроком (рис. 3.7).

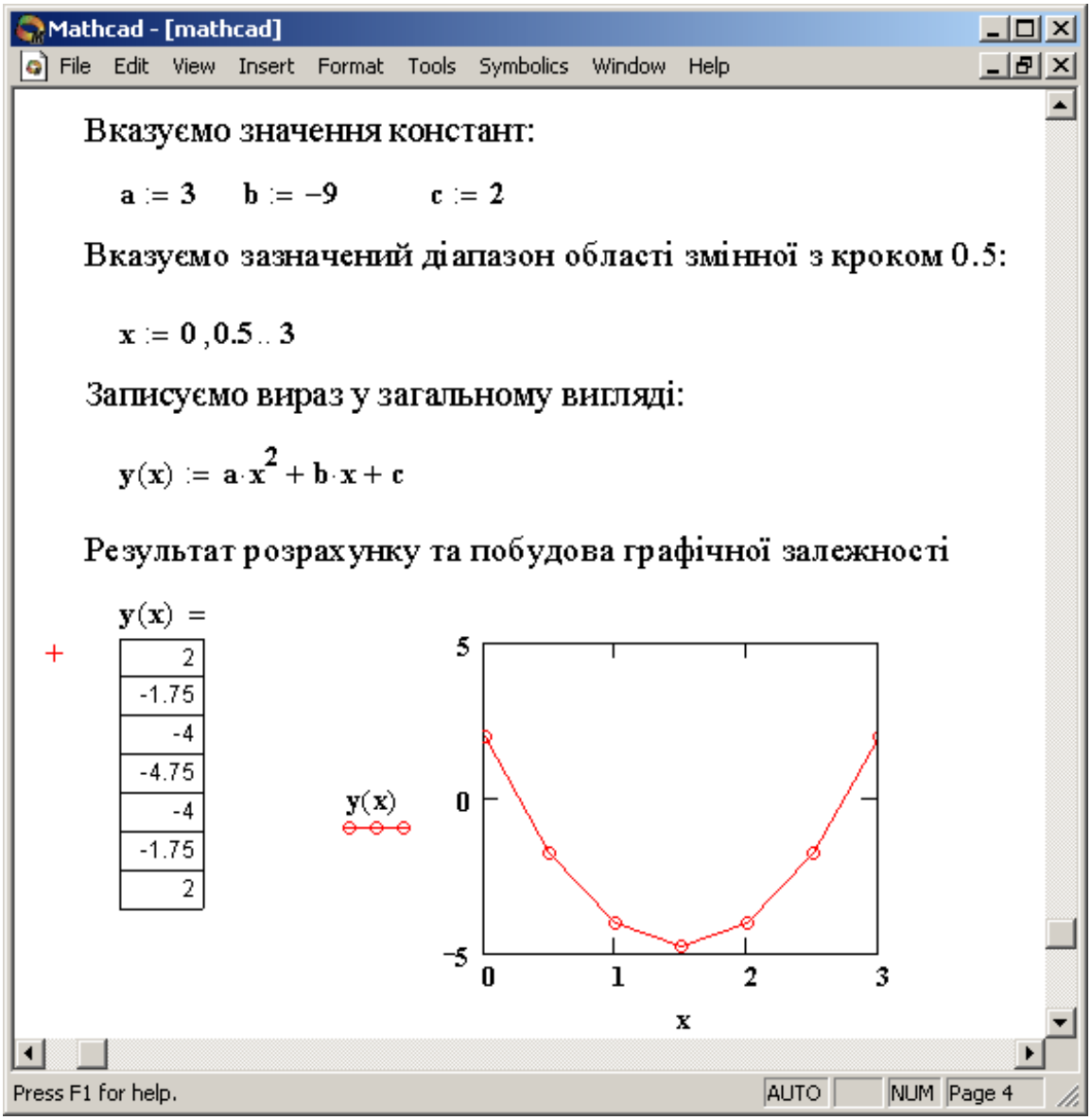


Рис 3.7. Задавання області змінних із вказаним кроком
у вікні документа MathCAD

Приклад побудови графічної залежності за експериментальними вимірами (i:=1..7) наведено на рис. 3.8. Значення ознак *y* та *x* записують як змінну з нижнім індексом.

Для побудови графічної залежності між точками, після виведення шаблону графіка, замість середніх вільних зон на осі абсцис записують символ ознаки x_i, а на осі ординат – символ ознаки y_i (див. рис. 3.8).

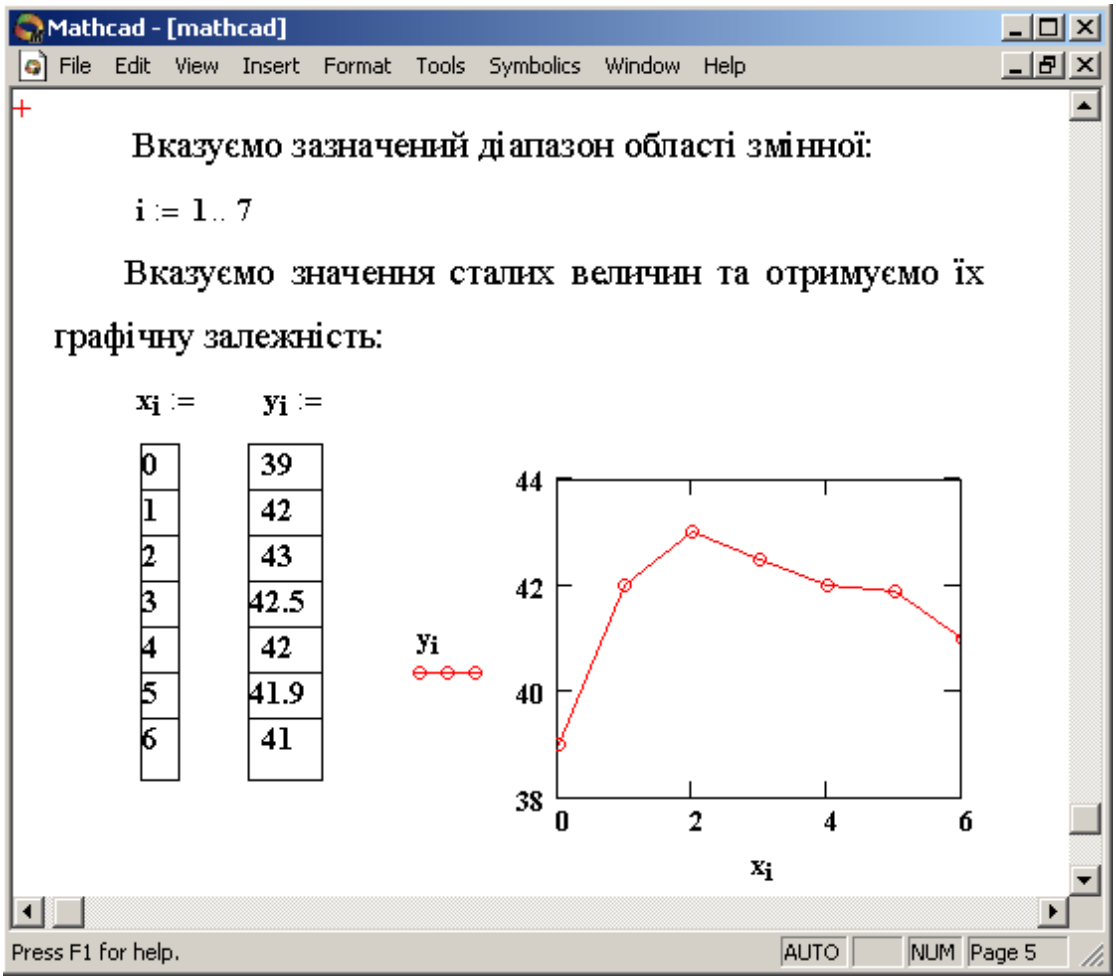


Рис 3.8. Побудова графічної залежності за експериментальними вимірами у вікні документа MathCAD

Тривимірні, або *3D-графіки*, відображають функції двох змінних виду *Z*(*X, Y*). Під час побудови тривимірних графіків у ранніх версіях MathCAD поверхню потрібно визначати математично (рис. 3.9, спосіб 1). Також застосовують функцію MathCAD *CreateMesh.*

CreateMesh(F (або G, або f1, f2, f3), x0, x1, y0, y1, xgrid, ygrid, fmap) створює сітку на поверхні, певною функцією *F*. *x*0, *x*1, *y*0, *y*1 – діапазон зміни змінних, *xgrid, ygrid* – розміри сітки змінних, *fmap* – функція відображення. Всі параметри, за винятком *F*, – факультативні [14]. Функція *CreateMesh* створює сітку на поверхні з діапазоном зміни змінних від –5 до 5 і із сіткою 20×20 крапок. Приклад на рис. 3.8 демонструє використання функції *CreateMesh* для побудови 3D-графіків у

вікні документа MathCAD. На рис. 3.9 наведено побудову 3D-графіка різними способами, з різним форматуванням. Така побудова здатна додати малюнку більшу наочність.

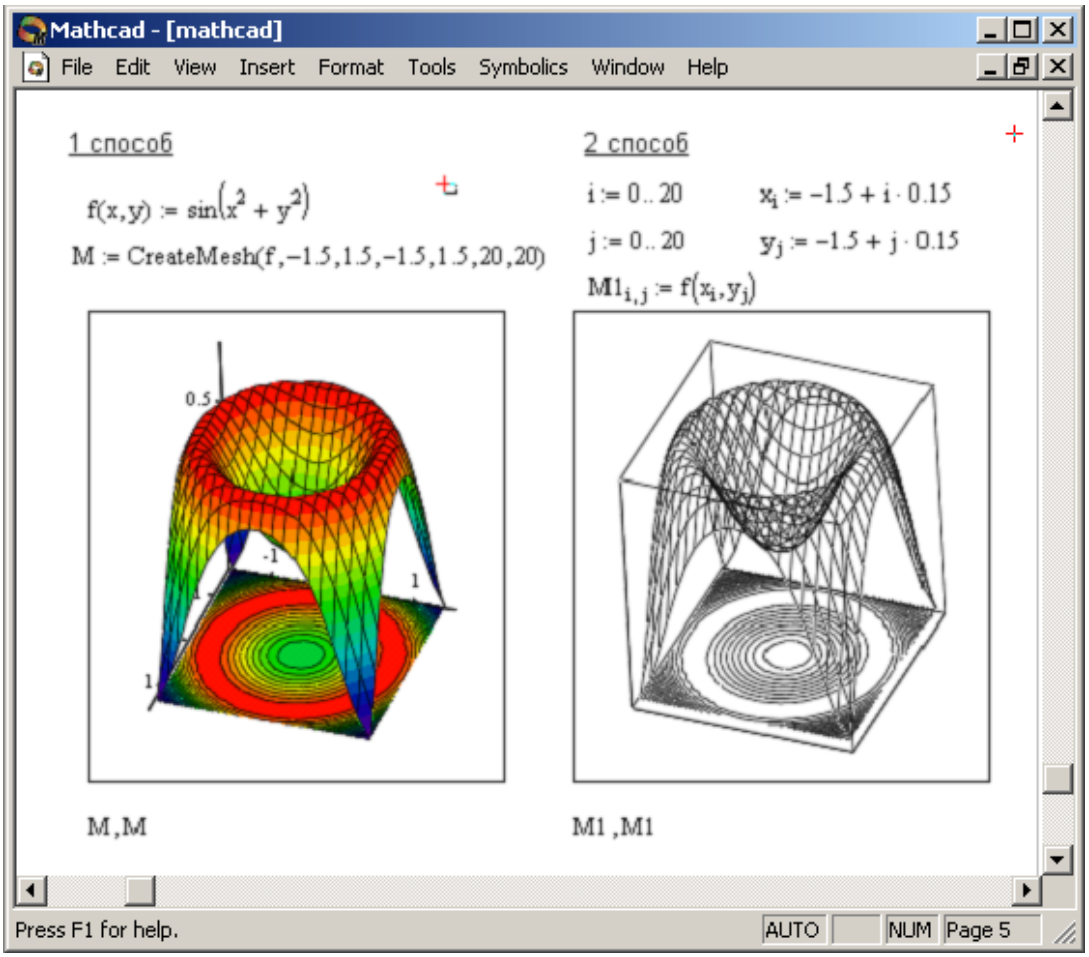


Рис. 3.9. Приклад побудови на одному малюнку двох 3D-графіків різного типу

3.1.2. Розв'язання рівнянь засобами MathCAD

Як відомо, багато рівнянь і системи рівнянь не мають аналітичних рішень. У першу чергу це стосується більшості трансцендентних рівнянь. Доведено також, що не можна побудувати формулу, за якою можна було б розв'язати довільне алгебраїчне рівняння вище четвертого ступеня. Однак такі рівняння можуть розв'язуватися чисельними методами із заданою точністю (не більше значення заданого системною змінною TOL).

Чисельне розв'язання нелінійного рівняння

Для найпростіших рівнянь виду *f(x) = 0* розв'язання в MathCAD здійснюється за допомогою функції *root*.

root(f(x1, x2, ...), x1, a, b),

де *f(x1, x2, ...)* – функція, визначена де-небудь у робочому документі або виразі. Вираз має повертати скалярні значення.

x1 – ім'я змінної, яка використовується у виразі. Цій змінній перед використанням функції *root* необхідно присвоїти числове значення. MathCAD використає його як початкове наближення під час пошуку кореня.

a, b – необов'язкові параметри, якщо використовуються, то мають бути дійсними числами, причому *a < b*.

Повертає значення *x1*, що належить відрізку *[a, b]*, при якому вираз або функція *f(x)* обертається в *0*. Обидва аргументи цієї функції мають бути скалярами. Функція повертає скаляр. Приклад (рис. 3.10) демонструє практичне застосування функції root .

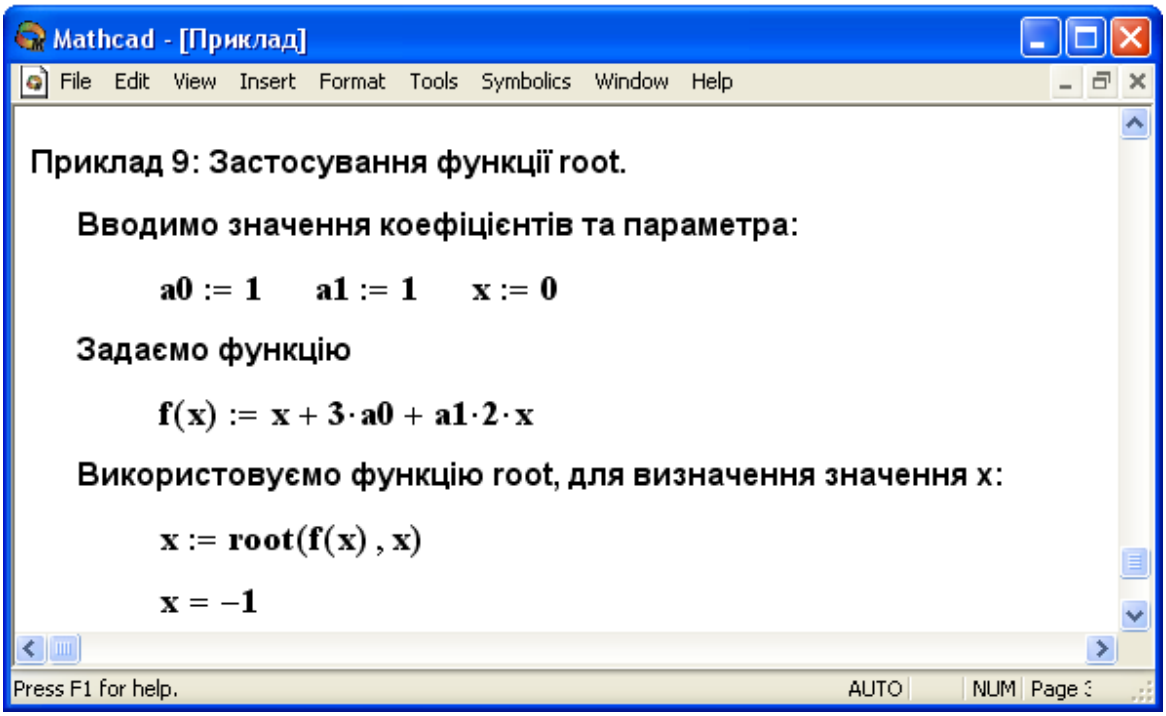


Рис 3.10. Застосування функції root у вікні документа MathCAD

Якщо після багатьох ітерацій MathCAD не знаходить відповідного наближення, це означає, що відсутня збіжність функції *root*. Ця помилка може бути викликана такими причинами [21]:

- рівняння не має коренів;
- корені рівняння розташовані далеко від початкового наближення;
- вираз має локальні *max* та *min* між початковим наближенням і коренями;
- вираз має розриви між початковими наближеннями та коренями;
- вираз має комплексний корінь, але початкове наближення було дійсним.

Щоб установити причину помилки, досліджуйте графік *f(x)*. Він допоможе з'ясувати наявність коренів рівняння *f(x) = 0* та, якщо вони є, визначити приблизно їхнє значення. Чим точніше обране початкове наближення кореня, тим швидше буде *root* сходитися.

Рекомендації з використання функції root

Для зміни точності, з якою функція *root* шукає корінь, потрібно змінити значення системної змінної TOL. Якщо значення TOL збільшується, функція *root* буде сходитися швидше, але відповідь буде менш точною. Якщо значення TOL зменшується, то функція *root* буде сходитися повільніше, але відповідь буде більш точною. Щоб змінити значення TOL у певному місці робочого документа, використайте визначення виду TOL:=0.01. Щоб змінити значення TOL для всього робочого документа, необхідно зайти в меню **Math**, кликнути мишею на вкладки **Options,** далі **Built-in variables** та вкладку **TOL.**

Якщо два корені розташовані близько один від одного, варто зменшити TOL, щоб розрізнити їх.

Якщо функція *f(x)* має малий кут нахилу біля шуканого кореня, функція *root(f(x), x)* може *сходитися* до значення *r*. У таких випадках для

знаходження більш точного значення кореня необхідно зменшити значення TOL.

Знаходження коренів полінома

Для знаходження коренів виразу, що має вид $v_n x^n + ... + v_2 x^2 + v_1 x + v_0$, краще використати функцію *polyroots*, ніж *root*. На відміну від функції *root*, функція *polyroots* не вимагає початкового наближення й повертає відразу всі корені, як дійсні, так і комплексні.

Polyroots(v),

де *v* – вектор, що містить коефіцієнти полінома.

Вектор *v* зручно створювати, використовуючи вкладку **Polynomial Coefficients** в меню **Symbolics**.

Функція повертає корені полінома степеня *n*. Коефіцієнти полінома перебувають у векторі *v* довжини *n + 1*. Функція повертає вектор довжини *n*, що складається з коренів полінома.

Розв'язання систем рівнянь

MathCAD дає можливість вирішувати також і системи рівнянь. Максимальне число рівнянь і змінних дорівнює 50. Результатом рішення системи буде чисельне значення шуканого кореня [22].

Для розв'язання системи рівнянь необхідно виконати наступні кроки:

- задати початкове наближення всіх невідомих, що входять у систему рівнянь. Це необхідно тому, що систему рівнянь MathCAD розв'язує за допомогою ітераційних методів;

- надрукувати ключове слово *Given*, воно вказує MathCAD, що далі буде введено систему рівнянь;

- ввести рівняння й нерівності в будь-якому порядку. Між лівими й правими частинами нерівностей може стояти кожний із символів <, >, ≥ і ≤;

- ввести будь-який вираз, що включає функцію *Find*, наприклад:

a:= Find(x, y).

Find(z1, z2, . . .)

Функція *Find* розв'язує систему рівнянь, у ній кількість аргументів має дорівнювати кількості невідомих. Ключове слово *Given*, рівняння й нерівності, які випливають за ним, вирази, що містять функцію *Find*, називають *блоком розв'язку рівнянь*.

У середині блоку розв'язку неприпустимі такі вирази:

- обмеження зі знаком ≠;
- дискретний аргумент або вираз, що містять дискретний аргумент у будь-якій формі;
- нерівності виду $a < b < c$.

Блоки розв'язку рівнянь не можуть бути вкладені один в одного, кожен блок може мати тільки одне ключове слово *Given* та ім'я функції *Find*.

Функція, що завершує блок розв'язання рівнянь, може бути використана аналогічно будь-якій іншій функції. Можна зробити з нею наступні три дії:

- вивести знайдений розв'язок, надрукувавши вираз виду:

Find(var1, var2,...) =.

- визначити змінну за допомогою функції *Find.* Це зручно зробити, якщо потрібно використати розв'язок системи рівнянь в іншому місці робочого документа:

a := Find(x) – скаляр,

var := Find(var1, var2,...) – вектор.

- визначити іншу функцію за допомогою *Find*:

f(a, b, c, ...) := Find(x, y, z, ...).

Ця конструкція зручна для багаторазового розв'язку системи рівнянь для різних значень деяких параметрів *a, b, c,...,* що безпосередньо

входять у систему рівнянь. Приклад (рис. 3.11) демонструє розв’язок системи рівнянь в MathCAD.

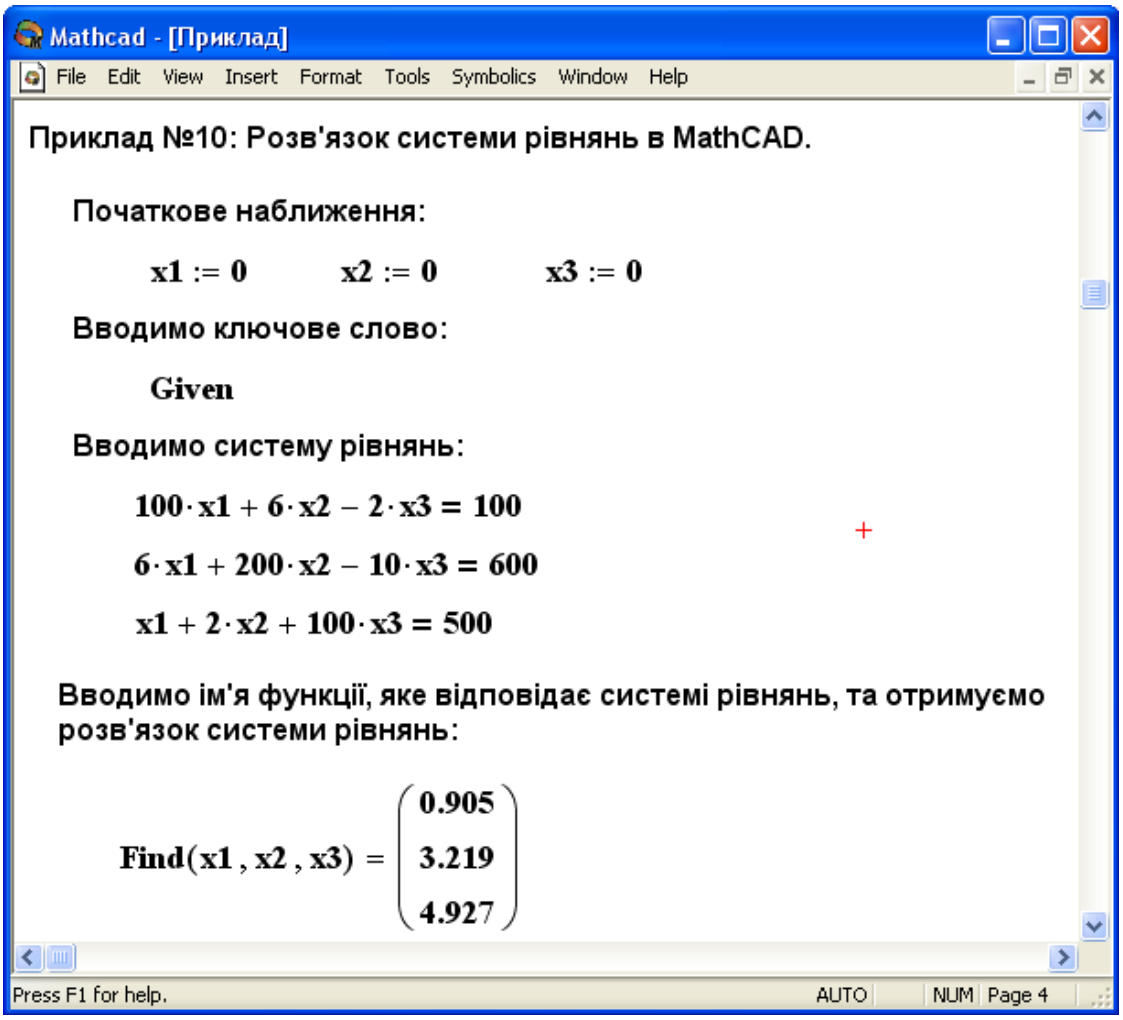


Рис 3.11. Розв’язок системи рівнянь у вікні документа MathCAD.

Повідомлення про помилку «Розв’язок не знайдено» під час розв’язання рівнянь з'являється, коли:

- поставлене завдання може не мати розв’язку;

- для рівняння, що не має розв’язку, як початкове наближення взяте дійсне число й навпаки;

- у процесі пошуку розв’язку послідовність наближень потрапила в область локального мінімуму нев’язок. Для пошуку шуканого розв’язку потрібно задати різні початкові наближення;

- можливо, поставлене завдання не може бути вирішене із заданою точністю. Спробуйте збільшити значення TOL.

Символьне розв'язання рівнянь

У MathCAD можна швидко й точно знайти чисельне значення кореня за допомогою функції *root*. Але є деякі завдання, для яких можливості MathCAD дозволяють знаходити розв'язок в символьному (аналітичному) вигляді.

Розв'язок рівнянь у символьному вигляді дозволяє знайти точні або наближені корені рівняння:

- якщо рівняння, яке розв'язуємо, має параметр, то рішення в символьному вигляді може виразити шуканий корінь безпосередньо через параметр. Тому замість того, щоб розв'язувати рівняння для кожного нового значення параметра, можна просто заміняти його значення в знайденому символьному розв'язку;

- якщо потрібно знайти всі комплексні корені полінома зі ступенем меншим або рівним 4, символьне рішення дає їх точні значення в одному векторі або в аналітичному чи цифровому вигляді.

За допомогою меню **Symbolics** почергово кликнути мишею на вкладку **Variable** та **Solve**, проводимо розв'язання рівняння щодо деякої змінної та виражаємо його корені через інші параметри рівняння. Приклад на рис. 3.12 демонструє символьний розв'язок системи рівнянь у вікні документа MathCAD.

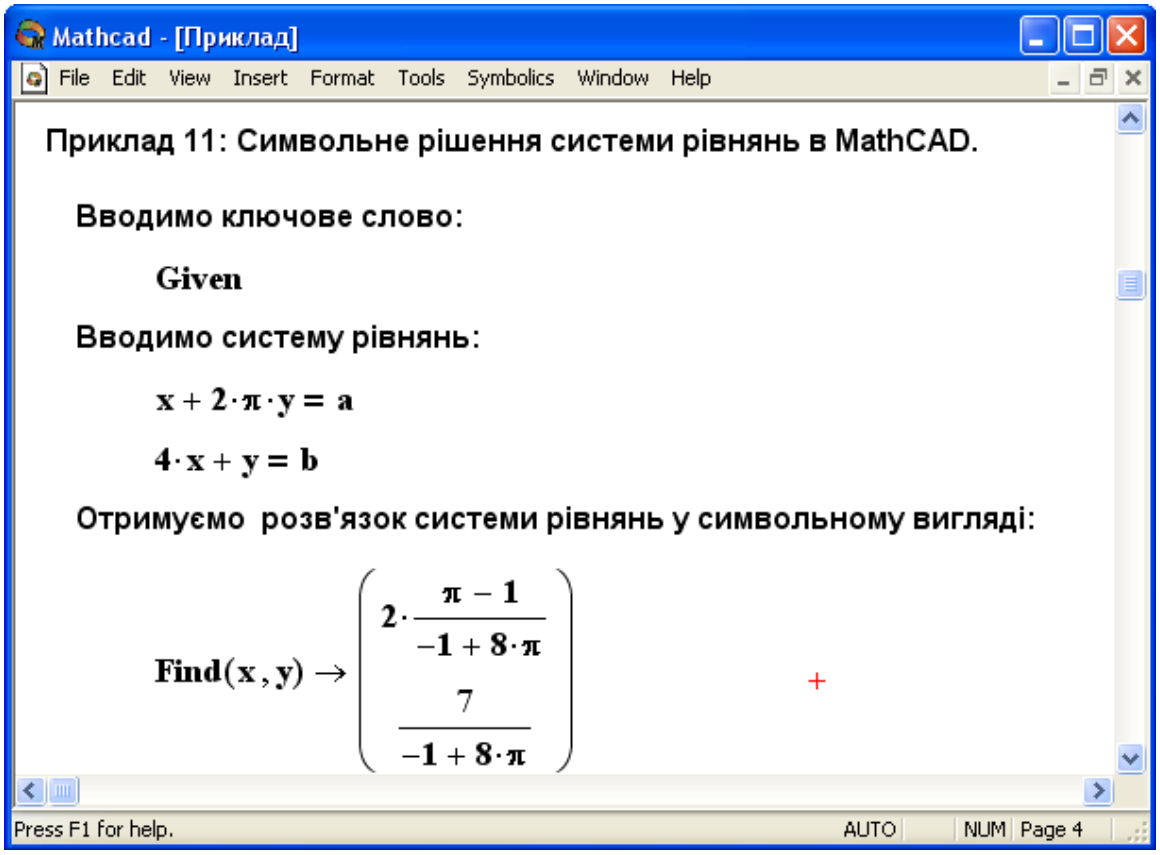


Рис 3.12. Символьний розв’язок системи рівнянь у вікні документа MathCAD

Наближений розв’язок рівняння

Функція *Minerr* дуже схожа на функцію *Find* (використовує той же алгоритм). Якщо в результаті пошуку не може бути отримане подальше уточнення поточного наближення до рішення, *Minerr* повертає це наближення. Функція *Find* у цьому випадку повертає повідомлення про помилку. Правила використання функції *Minerr* такі ж, як і функції *Find* [17]:

Minerr(z1, z2, . . .).

Якщо функція *Minerr* використовується в блоці рішення рівнянь, необхідно завжди включати додаткову перевірку вірогідності результатів. Приклад на рис. 3.13 демонструє наближений розв’язок нелінійних рівнянь у вікні документа MathCAD .

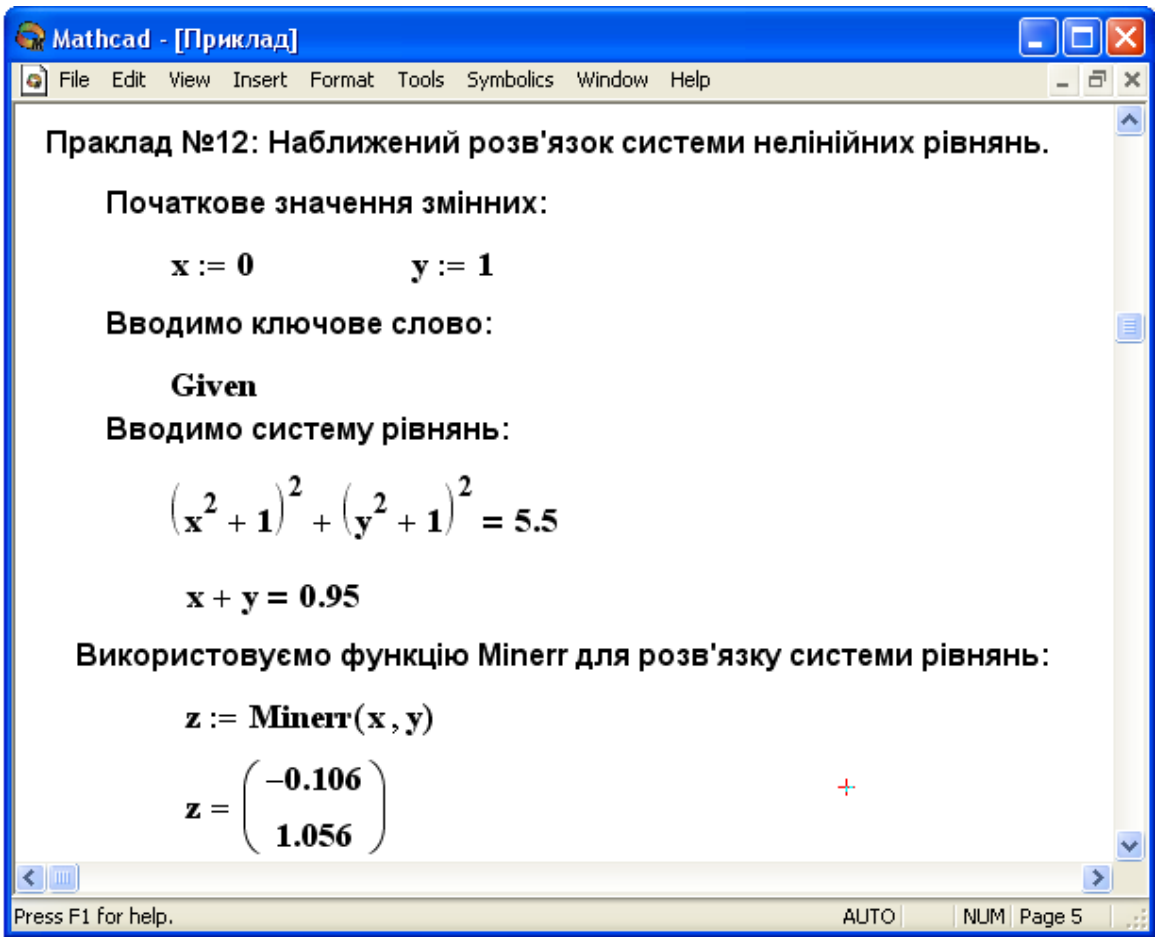


Рис 3.13. Наближений розв’язок нелінійних рівнянь у вікні документа MathCAD

Для символьного розв’язку системи рівнянь необхідно:

- ввести функцію;

- виділити змінну, відносно до якої потрібно розв’язати рівняння, кликнути на ній мишею;

- вибрати пункт меню **Symbolics,** почергово кликнути мишею на вкладку **Variable** та **Solve**.

Немає необхідності прирівнювати вираз до нуля. Якщо MathCAD не знаходить знака рівності, він припускає, що потрібно прирівняти вираз до нуля.

3.1.3. Логічні операції та використання виразів відношень у пакеті MathCAD

Вирази відношення використовують для порівняння двох арифметичних виразів між собою. Вирази відношення записують у вигляді:

< вираз А > < знак відношення > < вираз В> .

Як знак відношення виступають символи, наведені в таблиці 3.1. Якщо задане відношення виконується, то вираз відношення приймає значення рівне 1 («істина»), у протилежному випадку – 0 («хибне») [16].

Таблиця 3.1

Комбінації клавіш знаків відношень

Знак відношення	Комбінація клавіш
=	[Ctrl] + [=]
<	[<]
>	[>]
≥	[Ctrl] + [0]
≤	[Ctrl] + [9]
≠	[Ctrl] + [3]

Приклад на рис. 3.14 демонструє обчислення виразу відношення у вікні документа з однією умовою MathCAD.

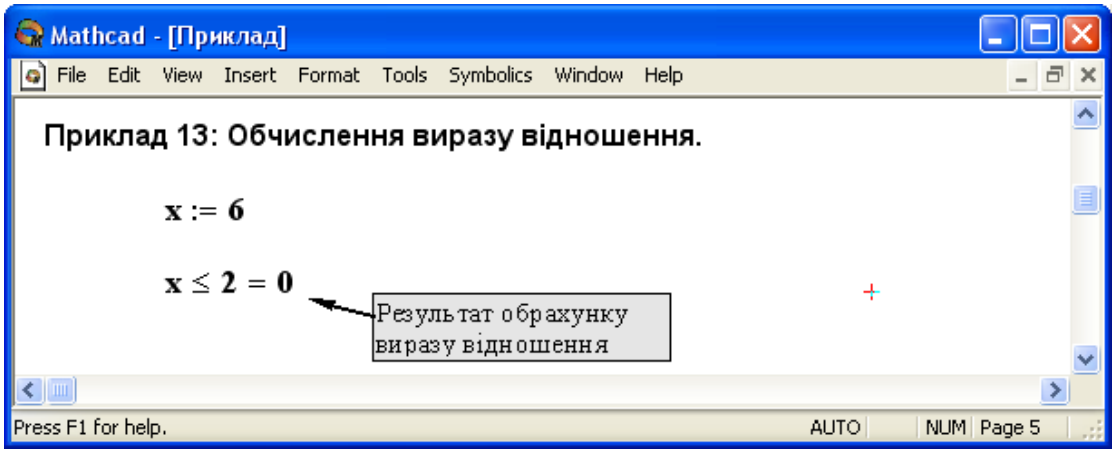


Рис. 3.14. Обчислення виразу відношення з однією умовою у вікні документа MathCAD

На відміну від мов програмування можемо відразу в одному виразі перевіряти кілька умов шляхом додавання знаків відношення та арифметичних виразів. Цю можливість демонструє приклад обчислення виразу відношення з кількома умовами у вікні документа MathCAD, наведений на рис. 3.15.

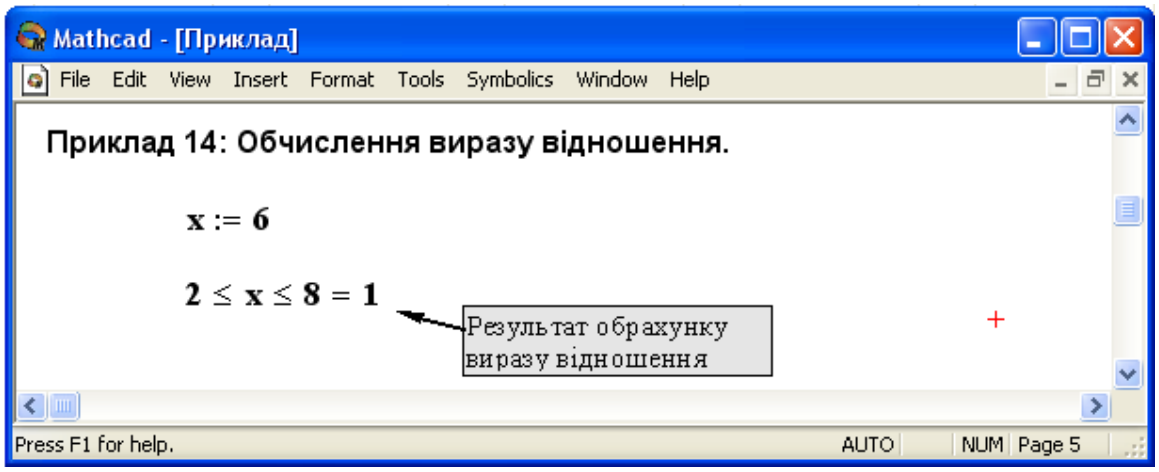


Рис 3.15. Обчислення виразу відношення з кількома умовами у вікні документа MathCAD

Логічні операції. Визначено дві логічні операції, які ставляться між виразами відношення.

Логічна операція **АБО**, позначається знаком «+» і записується у вигляді:

< логічний вираз 1 > + < логічний вираз 2 > = < результат операції >

Результат операції дорівнює 0, якщо обидва логічні вирази рівні 0, і дорівнюють 1 для всіх інших значень логічних виразів.

Логічна операція **И**, вводиться знаком «·» (на клавіатурі це *) і записується у вигляді:

< логічний вираз 1 > · < логічний вираз 2 > = < результат операції >

Результат дорівнює 1, якщо обидва логічні вирази рівні 1, і дорівнюють 0 для всіх інших значень логічних виразів (порівняйте з логічним оператором **АБО**).

Логічним виразом називають конструкцію, що складена з виразів відношення, знаків логічних операцій і круглих дужок. Значення логічного виразу обчислюється зліва направо з урахуванням відомого правила про пріоритет операцій [5]. Список пріоритетів (за їх зменшенням):

- круглі дужки;
- логічна операція **И**;
- логічна операція **АБО**.

Для однозначного обчислення логічного виразу використовуйте круглі дужки.

Умовна функція **if**. Цю функцію записують у вигляді (символи if вводять із клавіатури):

if (< логічний вираз > , < арифмет вираз 1> , < арифмет вираз 2 >)

Правило обчислення умовної функції **if**: якщо логічний вираз дорівнює 1, то функція приймає значення, що дорівнює значенню арифметичного виразу 1; якщо логічний вираз дорівнює 0, то функція приймає значення, що дорівнює значенню арифметичного виразу 2.

Умовна функція використовується в арифметичних виразах, що знаходяться в правій частині локального оператора присвоювання.

Приклад. Використовуючи умовну функцію **if**, реалізуємо алгоритм обчислення функції $y(x)=\begin{cases} x^2, x \le 0; \\ \sqrt{x}, x > 0, \end{cases}$.

На рис. 3.16 наведено реалізацію умовної функції **if** в зазначеному алгоритмі обчислення функції у вікні документа MathCAD.

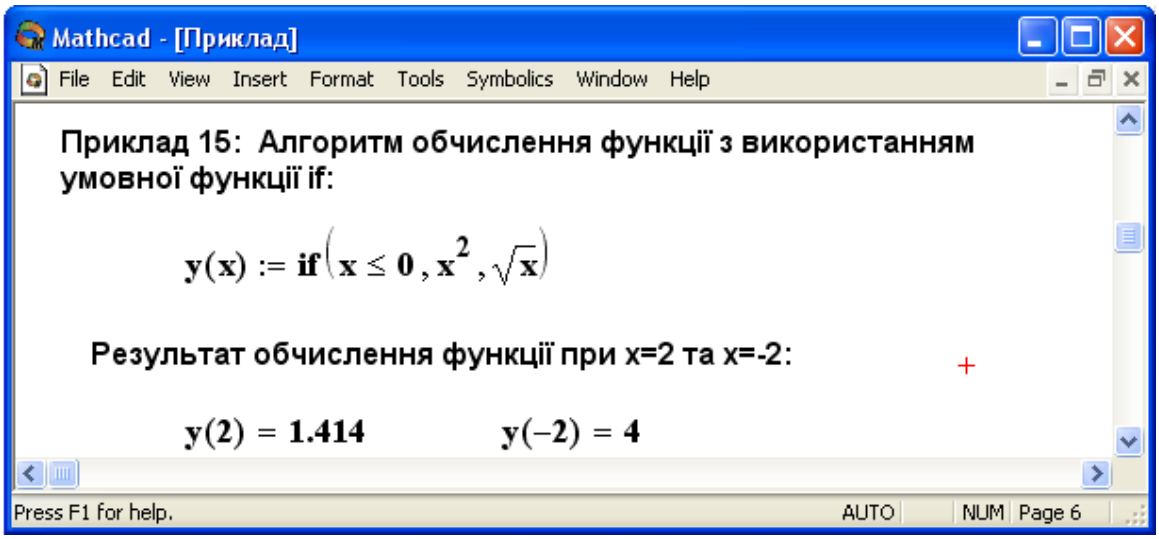


Рис 3.16. Умовна функція if в алгоритмі обчислення функції у вікні документа MathCAD

3.2. Основи програмування в математичному пакеті MathCAD

Процес створення комп'ютерних програм носить назву *програмування*, а людей, що займаються цим видом діяльності, називають *програмістами*. Програмування поєднує в собі елементи інженерії (існує навіть відповідна спеціальна галузь інженерії – програмна інженерія (англ. «software engineering»).

Під час роботи з математичним пакетом MathCAD завжди була потреба в програмуванні для розширення і вдосконалення базового набору математичних інструментів. У MathCAD вбудована нова мова програмування, не схожа на інші відомі та широко вживані мови програмування. По суті в MathCAD не вбудована мова програмування, а просто зняте обмеження на використання складових операторів у тілі

алгоритмічних конструкцій, що управляють вибором і повторенням. Крім того, додано цикл із параметром і оператором дострокового виходу та допоміжними операторами програмування.

До появи 7-ї версії системи MathCAD можливості програмування були вкрай обмеженими. Фактично MathCAD дозволяв реалізувати лише лінійні програми, в основі яких лежить поняття функції. Функція **if** і ранжирування змінної в окремих випадках могли замінити умовні вирази і цикли, але з серйозними обмеженнями. Була відсутня можливість завдання завершених програмних модулів. Ці можливості в розширеному варіанті з'явилися в MathCAD 7.0 Pro.

Відзначимо, що можливість складати програми реалізована тільки у версії Professional [13]. Всі MathCAD-програми з погляду програміста є програми-функції, які можуть повертати як результат число, вектор або матрицю. Функції можуть викликати самі себе (рекурсивно певні функції) або інші підпрограми-функції, визначені вище в MathCAD-документі.

3.2.1. Оператори Add line та локальне присвоювання в пакеті MathCAD

Перед тим, як використовувати програму-функцію, потрібно її описати. Опис програми-функції розміщено в робочому документі перед викликом програми-функції, він містить ім'я програми-функції, список формальних параметрів (який може бути відсутнім) і тіло програми-функції. Розглянемо ці поняття.

Кожна програма-функція MathCAD має *оригінальне ім'я*, використання якого здійснює звертання до цієї програми-функції. Через це ім'я (*і тільки через це ім'я*) повертається у робочий документ результат виконання програми-функції.

Після імені програми-функції йде *список формальних параметрів*, взятий у круглі дужки. Через формальні параметри усередину програми-функції потрапляють дані, необхідні для виконання обчислень всередині програми. Імена простих змінних, масивів та функцій можуть використовуватись як формальні параметри. Формальні параметри відокремлюються один від одного комами.

Програма-функція може не мати формальних параметрів, і тоді дані до функції передаються через імена змінних, визначених вище опису програми-функції.

Тіло програми-функції включає будь-яку кількість операторів: локальних операторів присвоювання, умовних операторів та операторів циклу, а також виклик інших програм-функцій і функцій користувача.

Порядок опису програми-функції в MathCAD. Для введення в робочий документ опису програми-функції необхідно виконати такі дії:

- ввести ім'я програми-функції та список формальних параметрів, взятих у круглі дужки;
- ввести символ «:» – на екрані відображається як «: =»;
- відкрити набірну панель **Programming** (див. Додаток 2) і клацнути кнопкою «**Add line**». На екрані з'явиться вертикальна риска та вертикальний стовпець із двома полями, для введення операторів, що утворять тіло програми-функції (рис. 3.17).

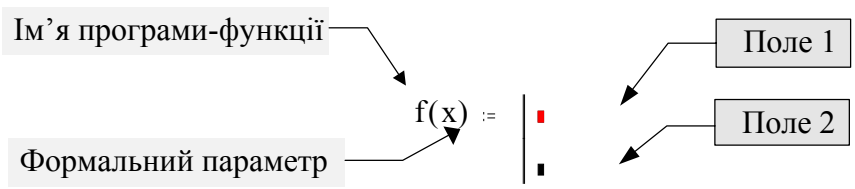


Рис. 3.17. Структура програми-функції

• перейти в поле 1, кликнувши на ньому мишею або нажавши клавішу [Tab], і ввести перший оператор тіла програми-функції. Нижнє поле завжди призначене для визначення значення, що повертається програмою. Поля введення для додаткових операторів відкриваються за допомогою натискання лівою кнопкою миші на кнопці «**Add line**» панелі програмування. При цьому поле введення з'являється після оператора, що є останнім до цього моменту оператора. Для видалення того або іншого оператора або поля введення з тіла програми-функції, потрібно виділити його та натиснути клавішу [Delete] (рис. 3.18);

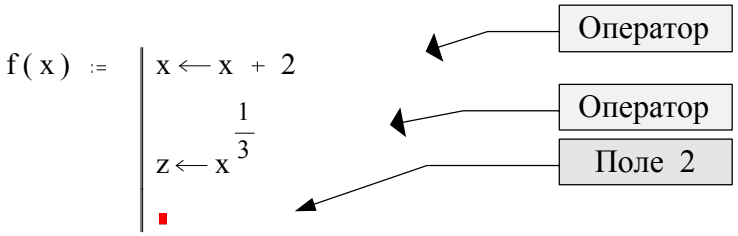


Рис. 3.18. Додавання операторів у тіло програми-функції

• заповнити нижнє поле введення (на рис. 3.17 позначене як поле 2), увівши вираз, що визначає значення результату виконання обрахунку шляхом виклику функції через її ім'я (рис. 3.19).

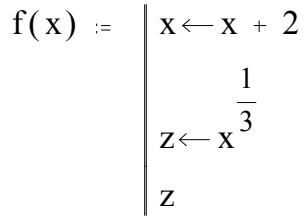


Рис. 3.19. Остаточна структура програми-функції

У наведеному прикладі формальним параметром є проста змінна *x*, тіло програми включає два локальних оператора присвоювання і значення

змінної *z* визначає результат виконання обрахунку через ім'я програми-функції.

Ця програма-функція відноситься до базової структури, яка має назву *послідовність*, блок-схему якої зображено на рис 3.20. Структура *послідовність* припускає послідовне виконання вхідних у неї інструкцій. Цю структуру розглядають як єдине ціле (на рис. 3.20 вона розміщена в пунктирному прямокутнику), має один вхід і один вихід.

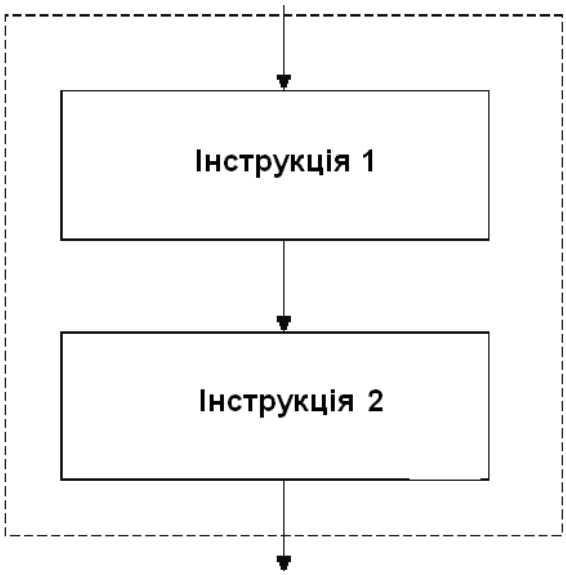


Рис. 3.20. Блок-схема базової структури *послідовність*

Для виконання програми-функції необхідно звернутися до її імені із вказівкою *списку фактичних параметрів*, якщо в описі програми присутній список формальних параметрів, тобто:

< ім'я – програми-функції > (список фактичних параметрів).

Фактичні параметри вказують, за яких конкретних значень здійснюються обчислення в тілі програми. Фактичні параметри відокремлюються один від одного комами.

Очевидно, що між фактичними й формальними параметрами має бути *відповідність щодо кількості, порядку розміщення й типу*. Відповідність типів параметрів:

- якщо формальним параметром є проста змінна, то як фактичний може використовуватися константа, змінна, арифметичний вираз;
- якщо формальним параметром є вектор або матриця, то фактичним має бути вектор або матриця;
- якщо формальним параметром є ім'я вбудованої функції або іншої програми, то й фактичним параметром має бути той самий об'єкт.

Локальний оператор присвоювання. Для завдання всередині програми значення якої-небудь змінної використовують локальний оператор присвоювання, що має вигляд:

< ім'я - змінної > ← < вираз >

Використання звичайного оператора присвоювання у тілі програми-функції приводить до синтаксичної помилки.

Приклад. Спростити вираз, розробити програму-функцію та побудувати її графічну залежність: $f(n) := \frac{\sqrt{n-1}\cdot(n-1)}{(n-1)\cdot\sin(n)}$. Зміна величини n знаходиться в межах [1], крок зміни значення n дорівнює 0,2.

На рис. 3.21 наведено приклад використання операторів Add line та локальне присвоювання у вікні документа MathCAD.

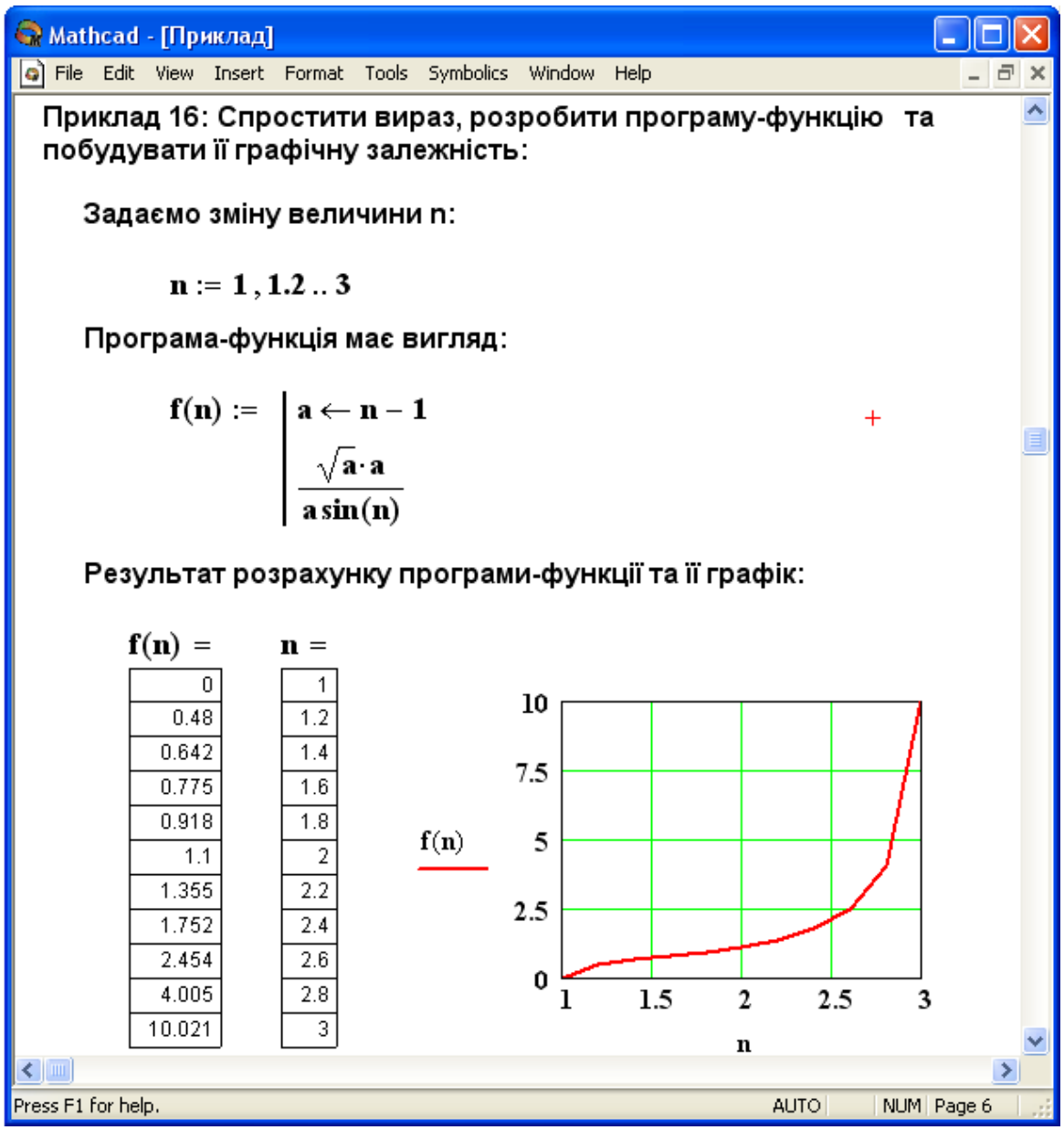


Рис. 3.21. Використання операторів Add line та локальне присвоювання у вікні документа MathCAD

3.2.2. Оператори if та otherwise в пакеті MathCAD

Цей оператор відноситься до базової структури, яка має назву *вибір*, блок-схему якої зображено на рис 3.22. Структура *вибір* припускає перевірку деякої умови. Залежно від того, виконується ця умова чи ні, виконується або одна інструкція, або інша.

Якщо на момент перевірки умова виконалась, то буде виконана інструкція 1, а інструкція 2 ігнорується. Якщо ж виявляється, що умова не виконана, то буде виконана інструкція 2, а інструкція 1 ігнорується. Структура *вибір* також має один вхід і один вихід.

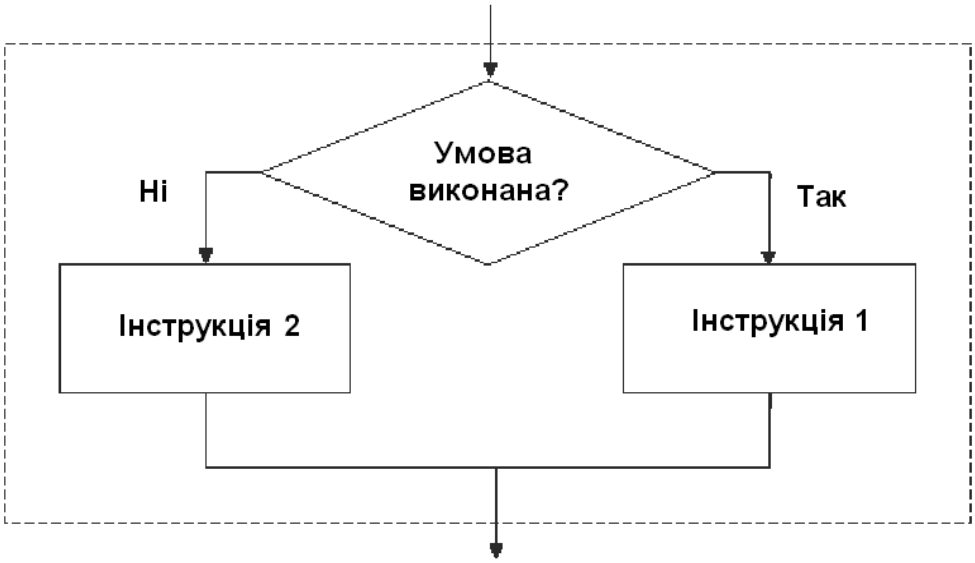


Рис. 3.22. Блок-схема базової структури *розгалуження*

Умовний оператор **if** використовують тільки в тілі програми-функції. Для його введення необхідно кликнути на кнопці if панелі програмування, на екрані з'являється конструкція із двома полями введення відповідно до зображення на рис. 3.23.

У поле 2 вводять логічний вираз (у найпростішому випадку це вираз відносин). В поле 1 вводять вираз (як правило, арифметичний), значення якого використовують, якщо логічний вираз, що перевіряється, набуває значення 1.

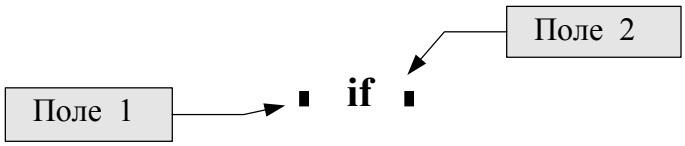


Рис. 3.23. Структура умовного оператора if

Умовний оператор може перебувати тільки всередині тіла програми-функції, наприклад, як це зображено на рис. 3.24:

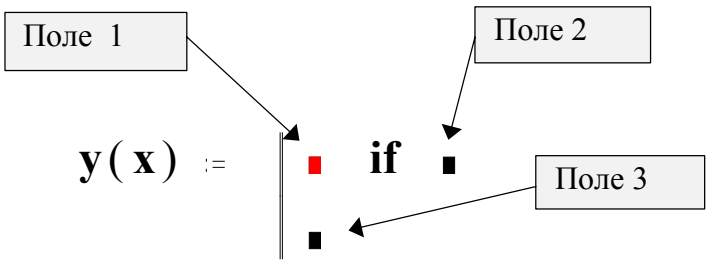


Рис. 3.24. Структура умовного оператора if в програмі-функції

У поле 3 вводять вираз, значення якого використовують, якщо логічний вираз дорівнює 0. Для введення в поле 3 необхідно:

- виділити це поле;
- кликнути на кнопці «otherwise» панелі програмування;
- в поле, що залишилося, ввести відповідний вираз.

Приклад. Використавши оператори if та otherwise в пакеті MathCAD, напишемо програму-функцію, що обчислює функцію:

$$y(x) = \begin{cases} x^2, x \le 0; \\ \sqrt{x}, x > 0. \end{cases}$$

На рис. 3.25 наведено програму-функцію, що обчислює зазначену функцію, яка має дві умови, тобто два інтервали зміни величини x, використавши оператори if та otherwise у вікні документа MathCAD.

Таким чином, вираз, розміщений перед словом **otherwise**, обчислюється тільки в тому випадку, якщо умова, що задана, перед ним не виконана.

У програмі можна використати послідовно умовні оператори з одним виразом перед оператором **otherwise**.

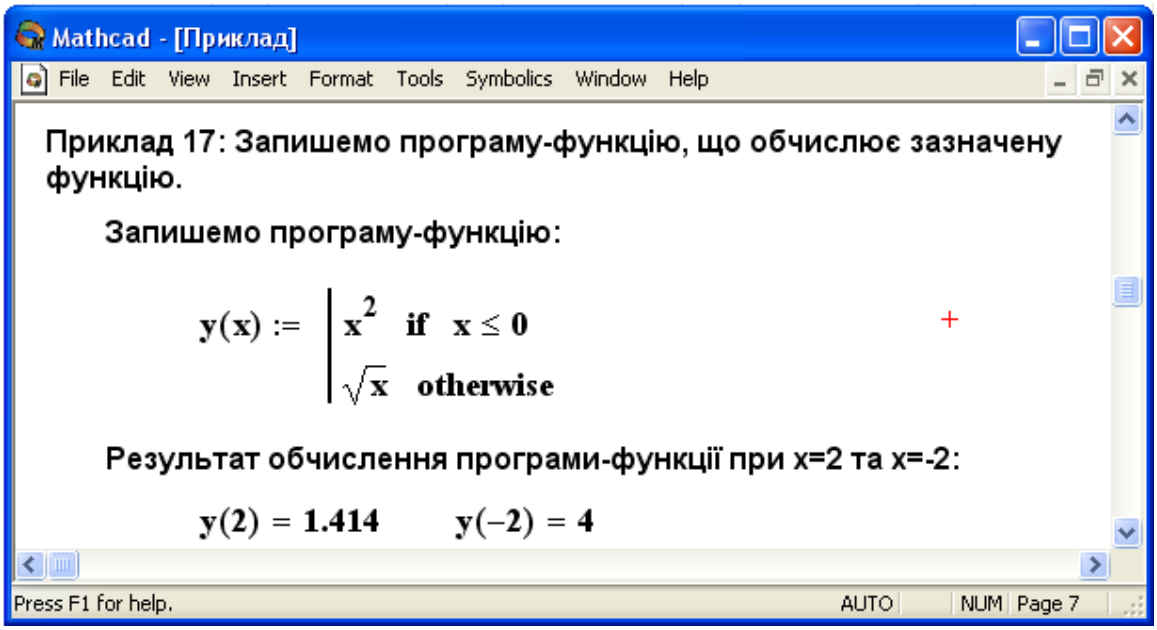


Рис. 3.25. Використання операторів if та otherwise за двох умов у вікні документа MathCAD

Приклад. Використавши оператори if та otherwise в пакеті MathCAD, напишемо програму-функцію для обчислення змінної z за функцією:

$$z(t) = \begin{cases} t^3, & t < -3; \\ t^2, & -3 \le t \le 4; \\ \ln(t), & t > 4. \end{cases}$$

На рис. 3.26 наведено зазначену програму-функцію для обчислення зазначеної функції, яка має три умови, тобто три інтервали зміни величини *t*, у вікні документа MathCAD. Відмітимо, що функція *z(t)* отримає значення *ln(t)* тільки тоді, коли не виконується умова, що записана в двох вищерозміщених рядках.

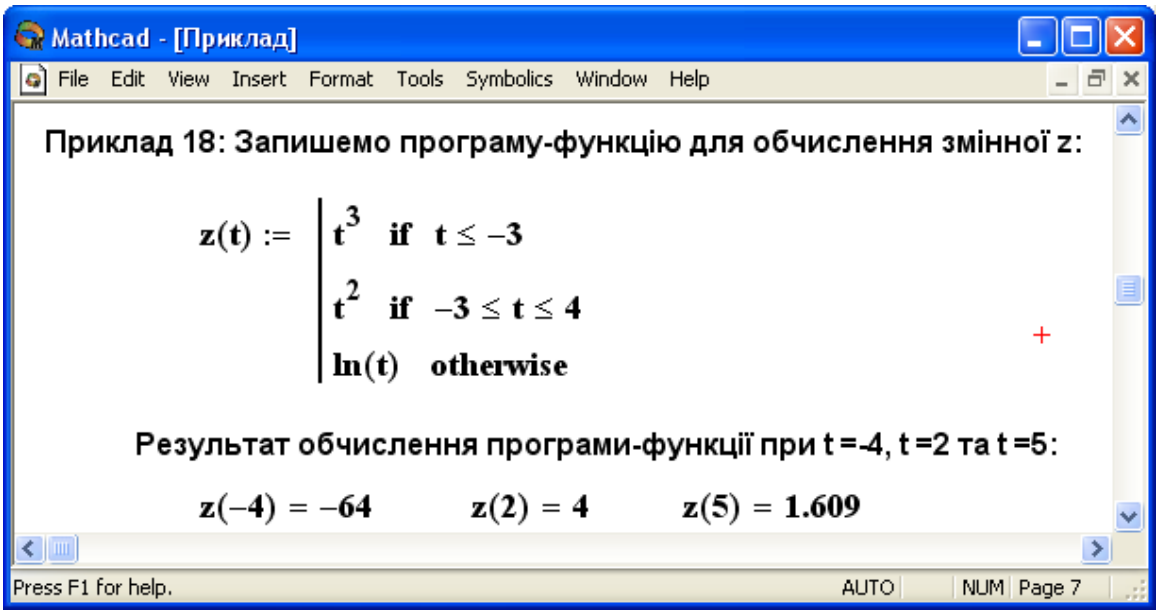


Рис. 3.26. Використання операторів if та otherwise за трьох умов у вікні документа MathCAD

Якщо в поле 3 ввести оператор без слова otherwise, то цей *оператор буде виконуватися завжди*, незалежно від виконання вище заданих умовних операторів.

3.2.3. Оператор for та циклічні алгоритми арифметичної прогресії в пакеті MathCAD

Програмування в програмі-функції циклічних алгоритмів. Циклічні алгоритми (або простіше цикли) містять обчислення, що повторюються певну кількість разів. Кількість повторень обчислень

залежить від деякої змінної. Така змінна називається *параметром циклу*, а самі обчислення, що повторюються, становлять *тіло циклу*.

Характерною рисою циклів типу арифметичної прогресії є те, що кількість повторень тіла циклу можна визначити до початку виконання програми, що реалізує цикл.

Для програмування циклу типу арифметичної прогресії використовують оператор циклу **for**.

Оператор **for** відносять до базової структури, яка має назву цикл. Цикл припускає повторення виконання деякої інструкції.

На рис. 3.27 зображено структуру *передумова*, блок-схему оператора циклу **for**. Якщо кількість повторень тіла циклу не досягає заданого значення, то інструкція буде виконана, що зображено стрілкою з написом Так. Якщо ж кількість повторень тіла циклу ще не вичерпана, то інструкція не буде виконана, і відбудеться вихід із циклу, що зображено стрілкою з написом Ні. Базова структура циклу має один вхід і один вихід.

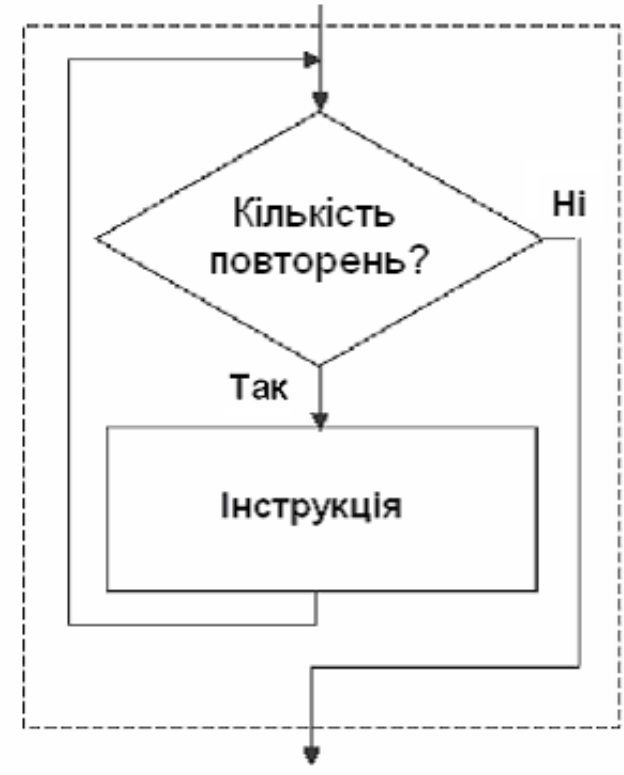


Рис. 3.27. Блок-схема базової структури оператора циклу for

Для введення такого оператора необхідно виконати такі дії:

- кликнути на кнопці **for** складальної панелі **Programming**. На екрані з'являться поля введення, зображені на рис. 3.28.
- у поле 1 ввести ім'я параметра циклу;
- у поле 2 ввести діапазон значень параметра циклу, використовуючи для цього дискретний аргумент;

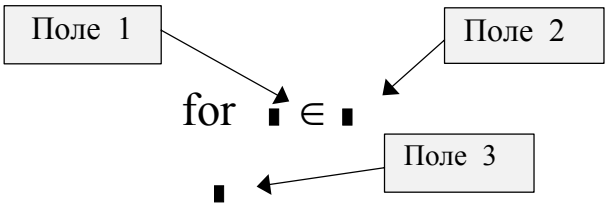


Рис. 3.28. Структура оператора циклу типу арифметичної прогресії for

- у поле 3 ввести оператори, що становлять тіло циклу. Якщо одного рядка недостатньо, то додаткові поля введення (додаткові рядки) створюються натисканням лівою клавішею миші на «Add line» в панелі програмування, і тоді ліворуч від тіла циклу з'явиться вертикальна риска.

Приклад. Використавши оператор for, розробити програму-функцію для визначення факторіала числа, тобто результатом обрахунку програми-функції має бути одне число.

На рис. 3.29 наведено зазначену програму-функцію у вікні документа MathCAD.

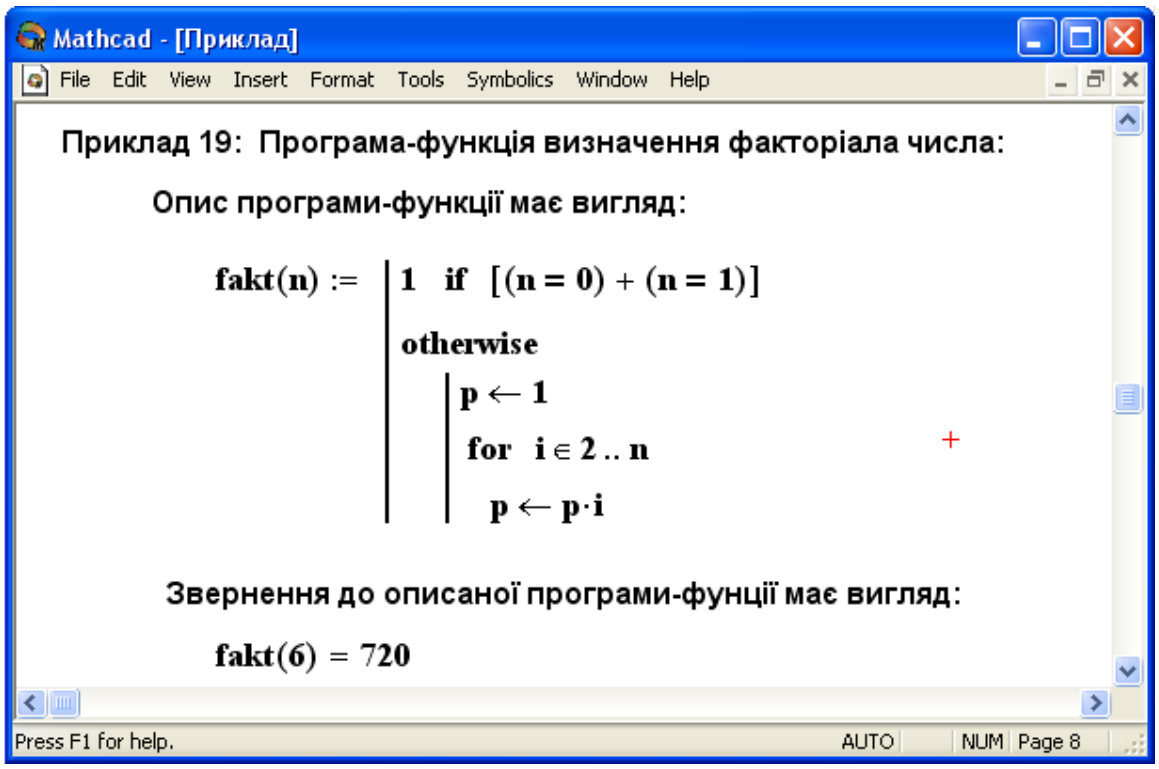


Рис. 3.29. Використання оператора for у вікні документа MathCAD

Приклад. Використавши оператор for, розробити програму-функцію для змінної x, що набуває значення на інтервалі від -2 до 2 із кроком 0,5, обчислити значення функції $f(x) = e^{-x} \cdot cos(2x)$ і сформувати із цих значень вектор y, тобто $y_1 = f(-2)$, $y_2 = f(-1.5)$. Результатом обрахунку програми-функції має бути масив чисел.

Опис програми-функції має вигляд, що наведено (рис. 3.30) у вікні документа MathCAD.

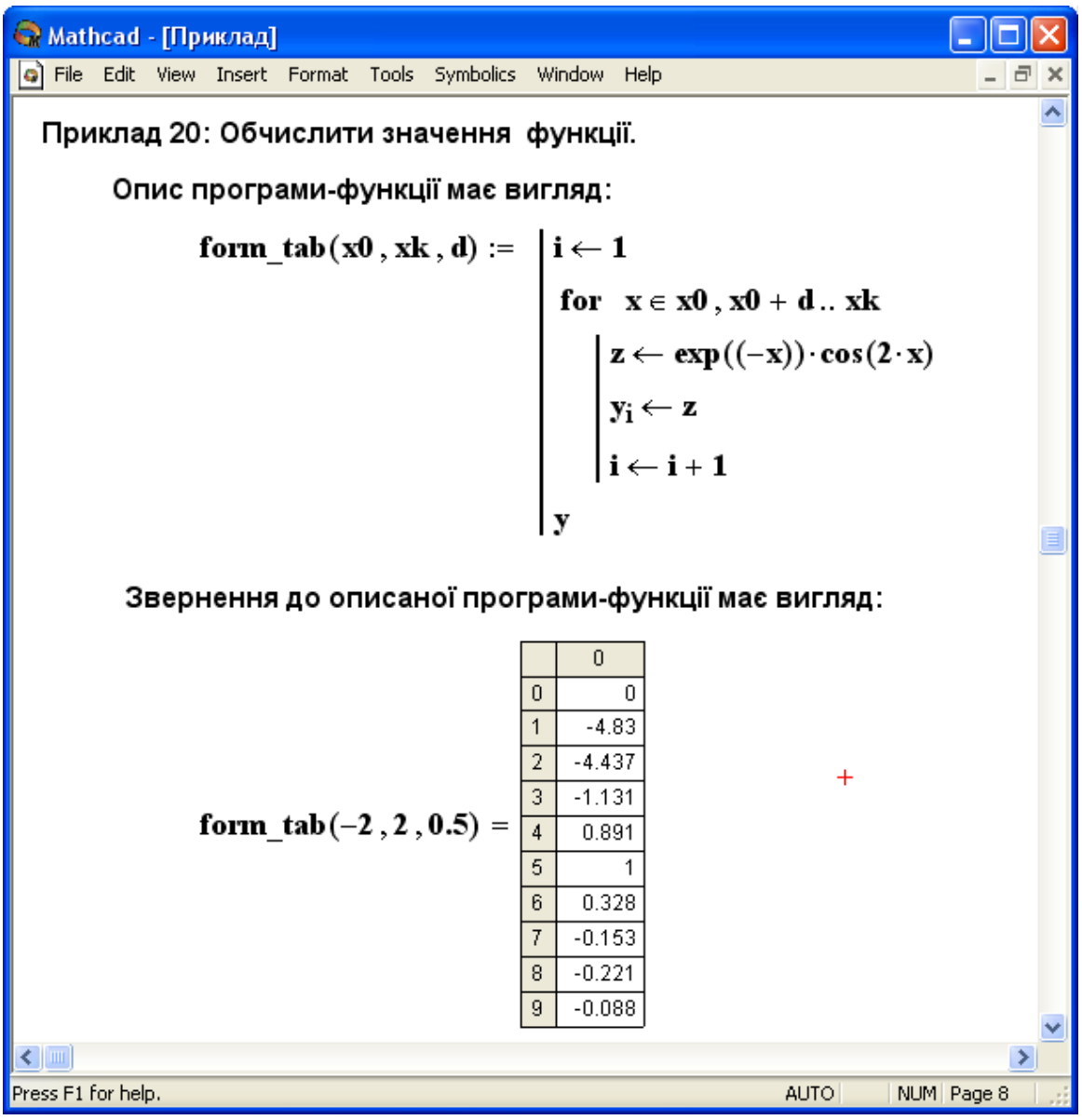


Рис. 3.30. Використання оператора for у вікні документа MathCAD

Кількість повторень визначають за формулою:

$$\left[\frac{x_k - x_0}{d}\right] + 1,$$

де x_k, x_0 – кінцеве та початкове значення параметра циклу;

d – крок його зміни.

Підставивши значення x_k, x_0 та d, одержимо кількість повторень циклу: (2 - (-2))/0.5+1=9.

У цьому варіанті опису програми-функції формальні параметри використовують для завдання діапазону зміни параметра циклу та кроку. Для зміни індексу елементів масиву *y* вводять змінну *i* цілого типу всередині програми-функції.

3.2.4. Оператор while та циклічні ітераційні алгоритми в пакеті MathCAD

Програмування ітераційних циклів. Для програмування таких циклів використовується оператор циклу **while**.

На рис. 3.31 зображено блок-схему оператора циклу **while**. Оператор **while** відносять до базової структури, яку називають циклічною. Кількість повторень тіла циклу залежить від умови виконання циклу. Якщо виконується умова виконання циклу, то інструкція буде виконана, що зображено стрілкою з написом «Так». Якщо не виконується умова виконання циклу, то інструкція не буде виконана і відбудеться вихід ыз циклу, що зображено стрілкою з написом «Ні». Базова структура циклу має один вхід і один вихід.

Для введення цього оператора необхідно виконати наступні дії:

- кликнути на кнопці **while** панелі **Programming.** На екрані з'являються елементи (рис. 3.32).

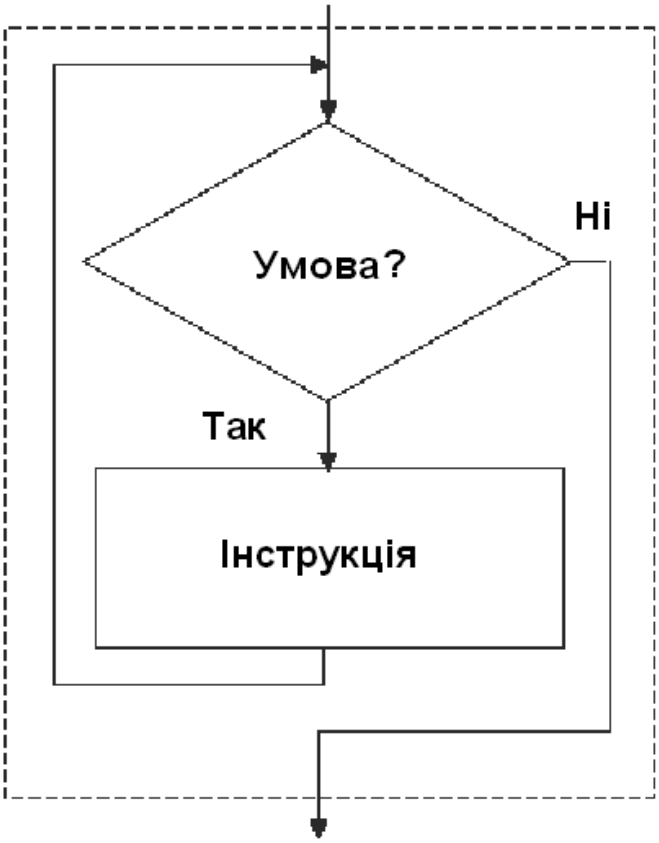


Рис. 3.31. Блок-схема базової структури оператора циклу while

- у поле 1 ввести умову виконання циклу;
- у поле 2 ввести оператори тіла циклу. У тілі циклу мають бути оператори, виконання яких змінює умову повторення циклу. Коли умова повторення циклу стає хибною, цикл завершується.

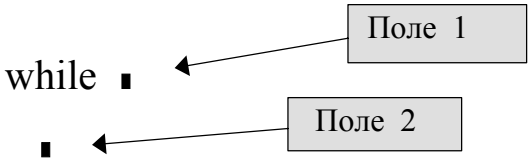


Рис. 3.32. Структура оператора циклу while

Для ітераційних циклів, до яких відноситься оператор **while**, не можна визначити кількість повторень тіла циклу. Це зумовлено тим, що закінчення таких циклів визначається не виходом параметра циклу за кінцеве значення, а більш складними умовами. Це ілюструє такий приклад.

Приклад. Використавши оператор while, розробити програму-функцію, що реалізує ітераційну процедуру наближеного обчислення кореня квадратного. На рис. 3.33 наведено зазначену програму-функцію у вікні документа MathCAD.

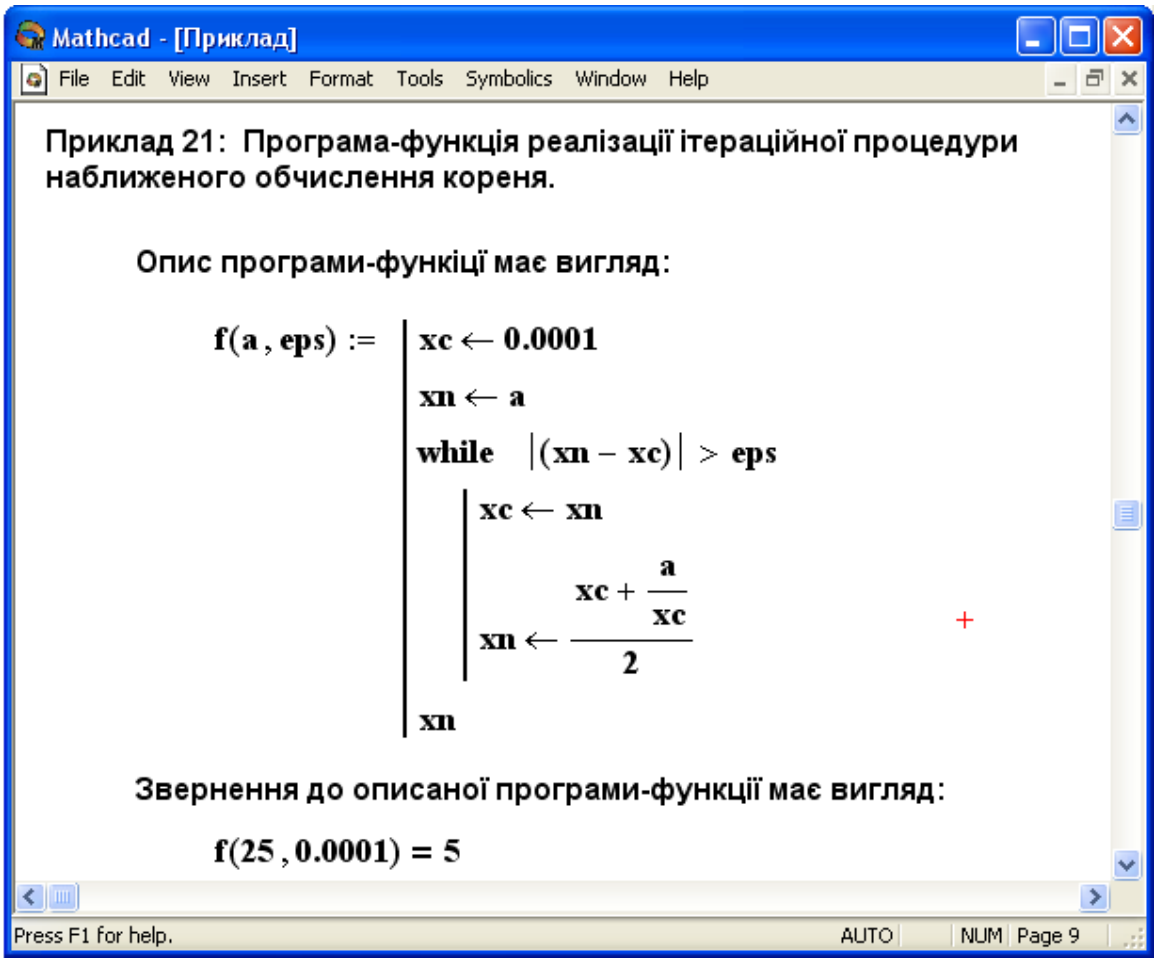


Рис. 3.33. Використання оператора while у вікні документа MathCAD

Використовують ітераційну процедуру наближеного обчислення кореня квадратного $x=\sqrt{a}$:

$$x_n=0.5\ (x_{n--1}+a/x_{n-1}),$$

де $n=1,2,3,..,\ x_0=a$.

Як наближене значення квадратного кореня беруть таке значення x_n, що задовольняє умову:

$$\left|x_n - x_{n-1}\right| \le \ \varepsilon ,$$

де ε – задана точність обчислення.

Очевидно, що задавши вихідні дані, наприклад, $a = 25$, $\varepsilon = 10^{-3}$, неможливо визначити кількість повторень тіла циклу, не виконуючи ітераційні обчислення.

Як видно з тексту програми-функції, немає необхідності зберігати в пам'яті всі наближені рішення $x_0,\ x_1,\ x_2\ ,\ ...$ і т.д. Досить зберігати попереднє значення *xc* і наступне значення *xn*.

Організація ітераційного циклу за допомогою оператора **while** без додаткових засобів контролю може призвести до зациклення. Наприклад, задавши у разі звернення до програми eps < 0 одержуємо зациклення програми. Для того, щоб вийти із зациклення, необхідно натиснути клавішу **Esc**.

3.2.5. Оператор break припинення циклу в пакеті MathCAD

У MathCAD є оператор **break**, що дозволяє вийти із циклу або призупинити роботу програми, перевіривши задану в операторі **break** умову. Для введення оператора **break** необхідно клацнути на кнопці **break** панелі **Programming** (не можна вводити цей оператор із клавіатури символами). Оператор **break** використовують у лівому полі введення умовного оператор **if**, а в правому розміщують умову, у разі виконання якої відбувається припинення роботи циклу або програми, у нижнім полі розміщують оператор, що виконується, якщо умова хибна. Тому спочатку вводяться оператор **if,** а потім заповнюють поля цього оператора.

Наступний приклад показує написання незацикленої програми з оператором **break**, який запобігає її зацикленню.

Приклад. Використавши оператор break, розробити програму-функцію, що реалізує ітераційну процедуру обчислення кореня квадратного $x=\sqrt{a}$, використовуючи ітераційну процедуру без зациклення. Опис такої програми-функції наведено нижче (3.34) у вікні документа MathCAD.

У цій програмі кількість повторень тіла циклу обмежено значенням 1000. Якщо за таку кількість ітерацій наближене значення кореня із заданою точністю не знайдено, то параметр *ierr* одержує значення 1, що свідчить про помилку обчислювального процесу. Оскільки через ім'я програми передається значення тільки однієї змінної, то для передачі двох значень *xn* та *ierr* використовують вектор, проекції якого формуються всередині програми.

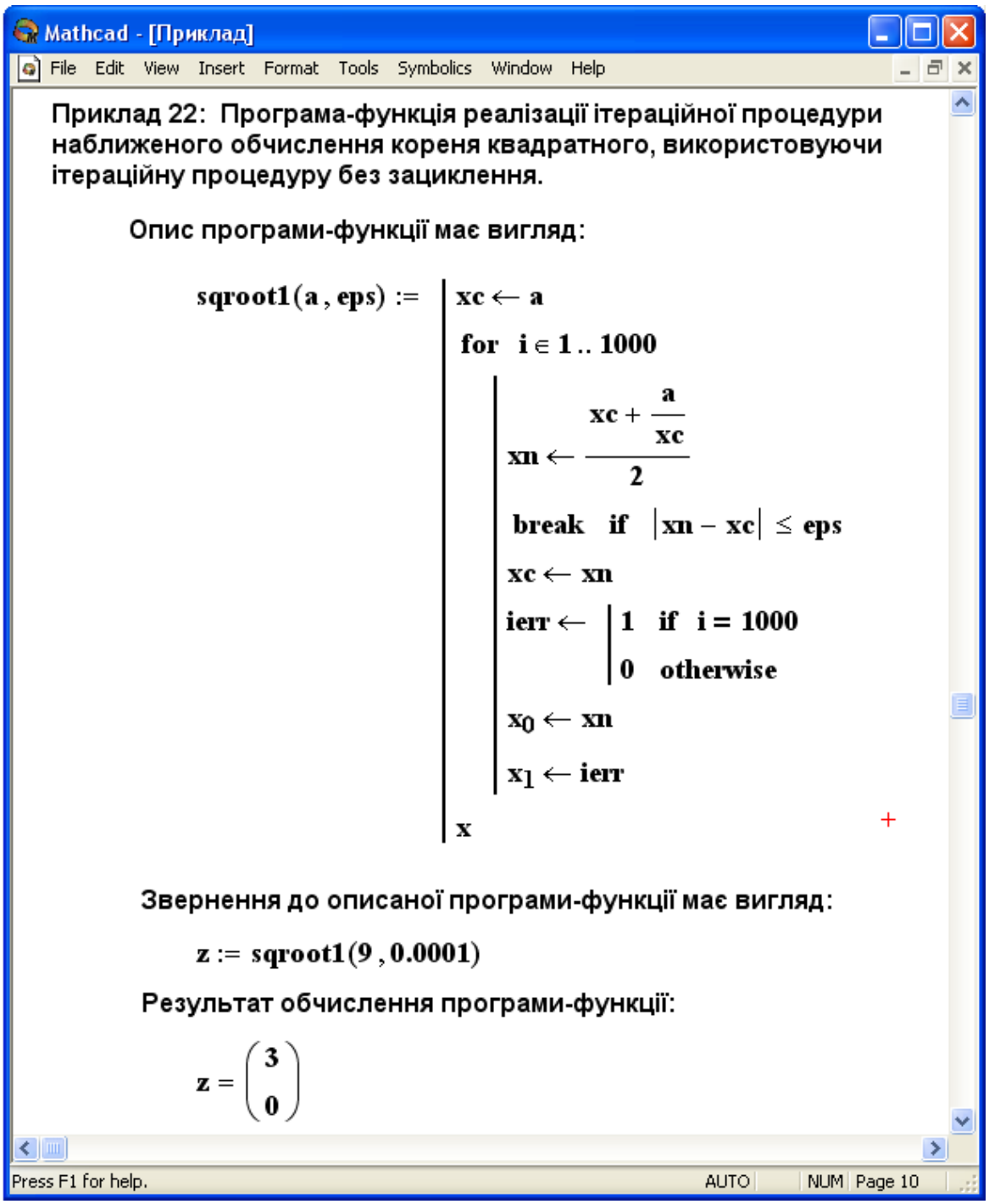


Рис. 3.34. Використання оператора break у вікні документа MathCAD

3.3. Додаткові оператори програмування в математичному пакеті MathCAD

Використання додаткових операторів допомагає програмісту побудувати більш складну програму-функцію і спростити її написання. Застосування додаткових операторів дозволить використати більш складну конструкцію програми, а саме ввести умову для продовження виконання циклу, переривання виконання програми-функції та повернення на початок, використати оператор обробки помилок та використовувати для висновку діагностичні повідомлення.

3.3.1. Оператор continue програмування циклів в пакеті MathCAD

Оператор **continue** зазвичай використовують для продовження виконання циклу шляхом повернення на початок тіла циклу. Наступний приклад пояснює роботу цього оператора.

Приклад. Використавши оператор continue, розробити програму-функцію, що формує новий вектор з додатніх чисел вхідного вектора. На рис. 3.35 наведено зазначену програму-функцію у вікні документа MathCAD.

У тілі програми-функції використовують функцію last(v), що визначає індекс останнього елемента масиву v.

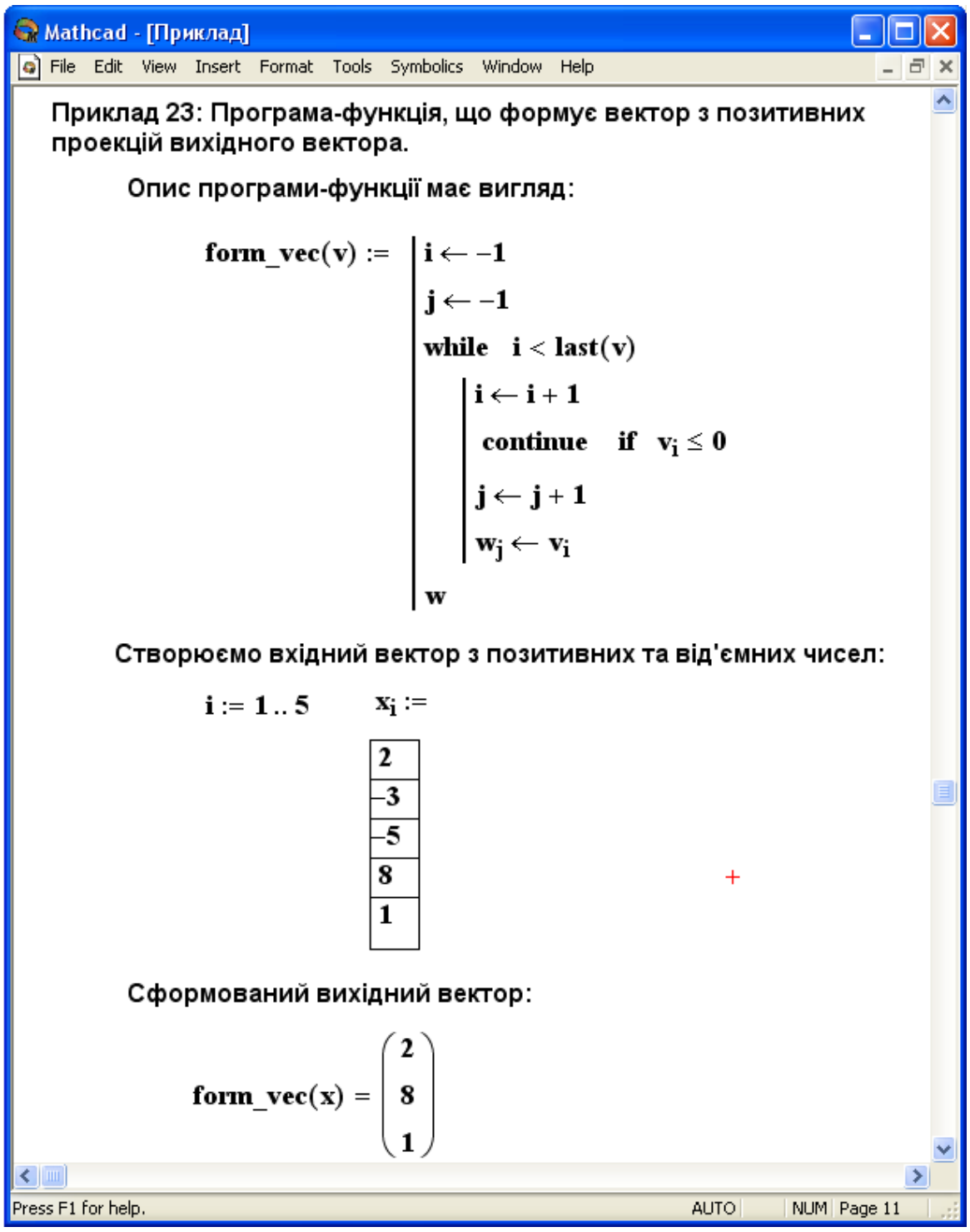


Рис. 3.35. Використання оператора continue у вікні документа MathCAD

Якщо черговий елемент вектора $v_i \leq 0$, то пропускаються всі оператори, що знаходяться нижче оператора **continue** (у нашому випадку – два оператори, що формують чергову проекцію вектора *w*), і тіло циклу повторюється за нового значення параметра циклу *i*.

3.3.2. Оператор return програмування циклів у пакеті MathCAD

Оператор **return** перериває виконання програми-функції та повертає значення операнда, що знаходиться за ним. Наступний приклад пояснює роботу цього оператора.

Приклад. Складемо програму-функцію, що знаходить першу додатню проекцію вихідного вектора. При складанні програми-функції зазначеної задачі можливі два варіанти рішення.

Перший – за допомогою оператора **while**. На рис. 3.36 наведено зазначену програму-функцію у вікні документа MathCAD.

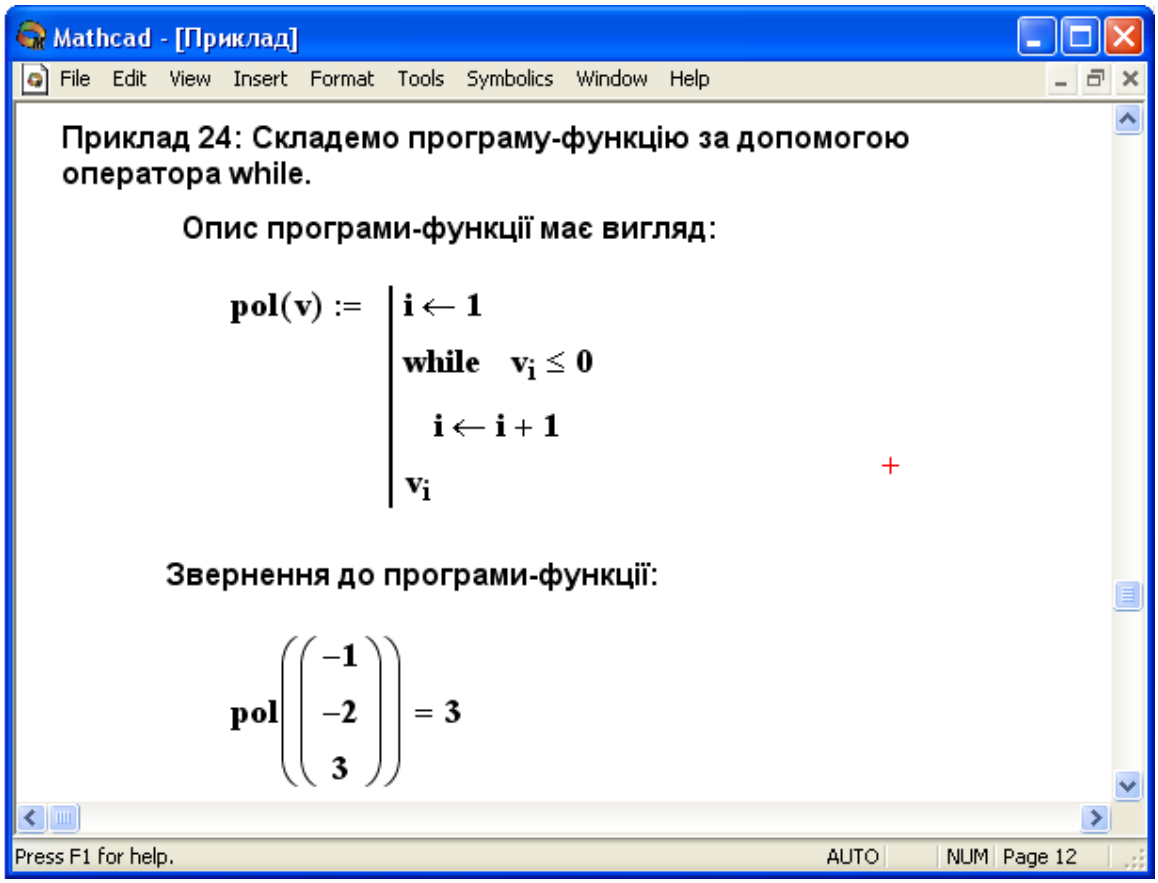


Рис. 3.36. Використання оператора while у вікні документа MathCAD

Другий – за допомогою оператора **for** та оператора **return**. На рис. 3.37 наведено зазначену програму-функцію у вікні документа MathCAD.

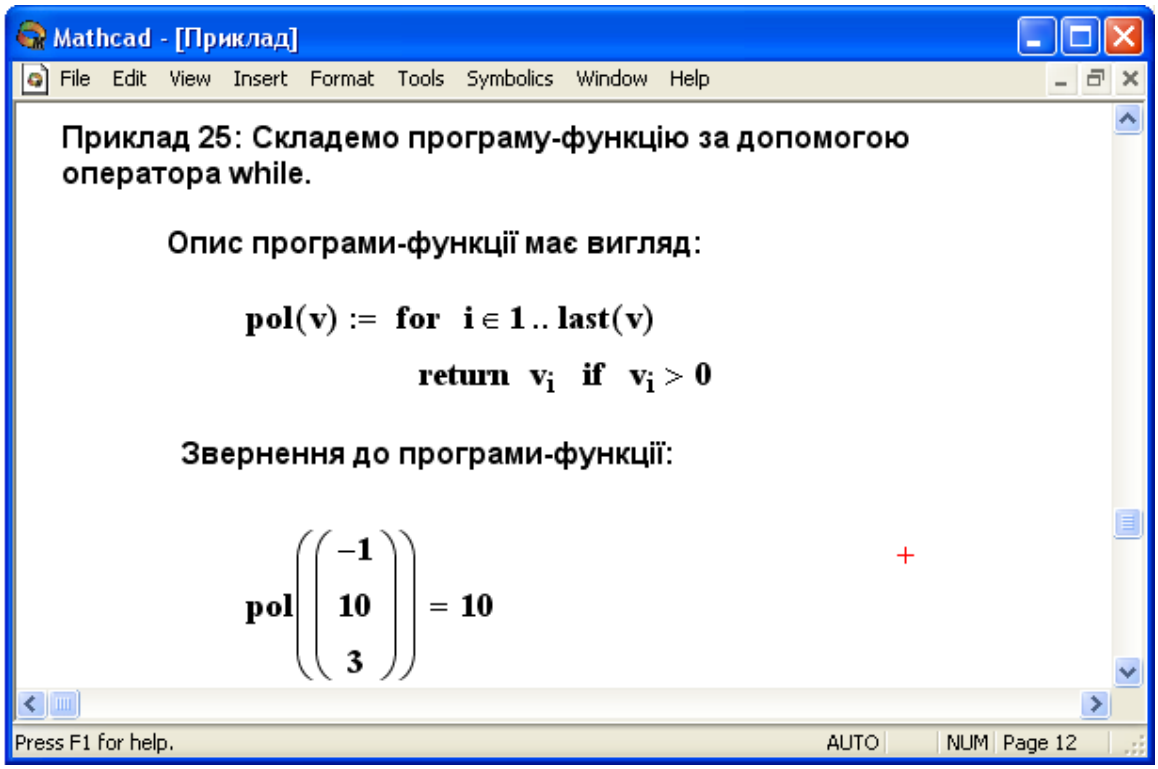


Рис. 3.37. Використання операторів for та return у вікні документа MathCAD

Другий варіант є більш простим порівняно з першим варіантом.

3.3.3. Оператор on error програмування циклів та функція error в пакеті MathCAD

Оператор **on error** обробляє помилки, що виникають під час виконання тих або інших обчислень, і записується у вигляді:

< вираз 1 > **on error** < вираз 2 >.

Виконується < вираз 1 >, якщо під час виконання < виразу 2 > виникає помилка. Якщо помилка не виникає, то виконується < вираз 2 >.

Приклад. Використаємо оператор **on error** для запобігання помилки, пов'язаної з діленням на нуль під час обчислення функції angl(x,y). Обчислення зазначеної функції наведено у вікні документа MathCAD (рис. 3.38).

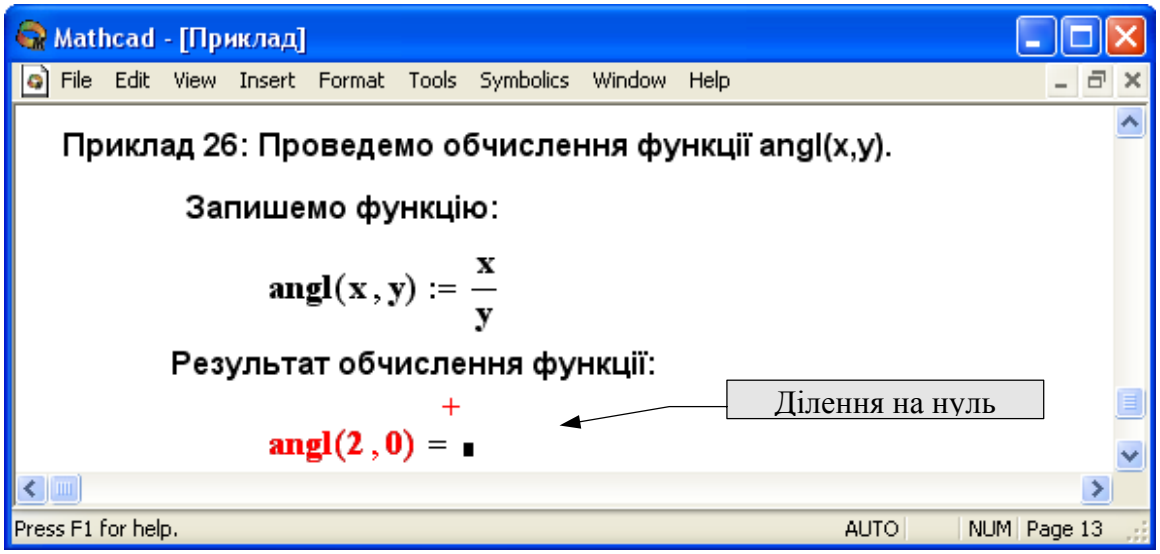


Рис. 3.38. Обчислення функції angl(x,y) у вікні документа MathCAD

Під час обчислення функції angl(x,y) виникла помилка, яка пов'язана з діленням на нуль. Для запобігання цієї помилки використаємо оператор **on error.** На рис. 3.39 наведено обчислення зазначеної функції у вікні документа MathCAD.

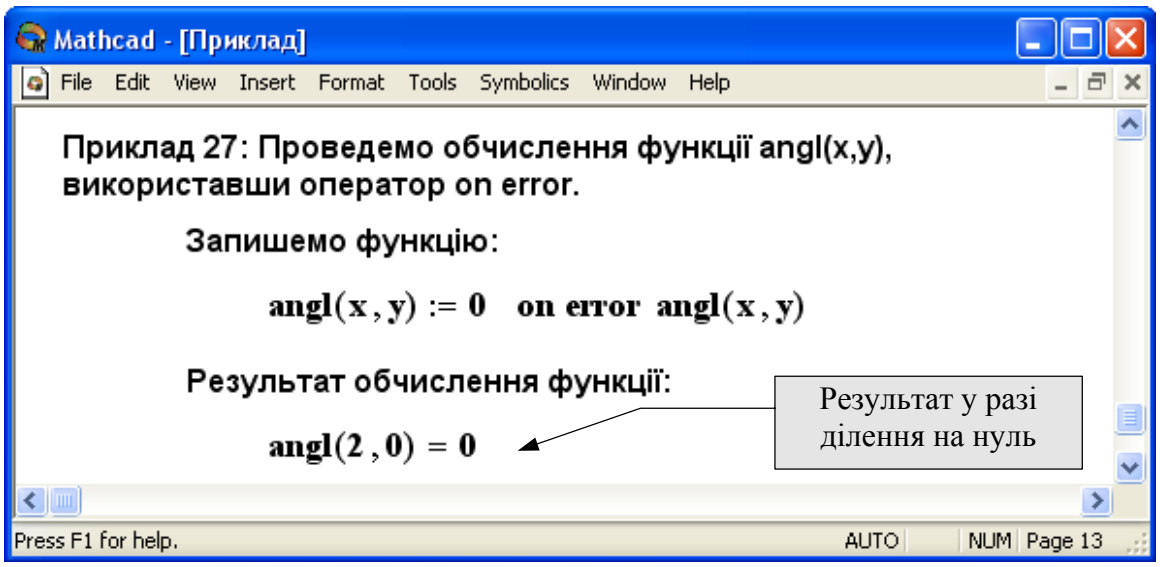


Рис. 3.39. Використання оператора on error у вікні документа MathCAD

Функцію **error** використовують для формування діагностичних повідомлень у разі виникнення в обчисленнях помилки і записують у вигляді:

error («< діагностичне повідомлення користувача >»).

Функція використовується в лівому полі умовного оператора **if,** як показано в такому прикладі.

Приклад. Використавши функцію error, вивести діагностичне повідомлення при спробі спроектувати вектор *v* на нульовий вектор *w*. На рис. 3.40 наведено реалізацію виведення діагностичного повідомлення у вікні документа MathCAD.

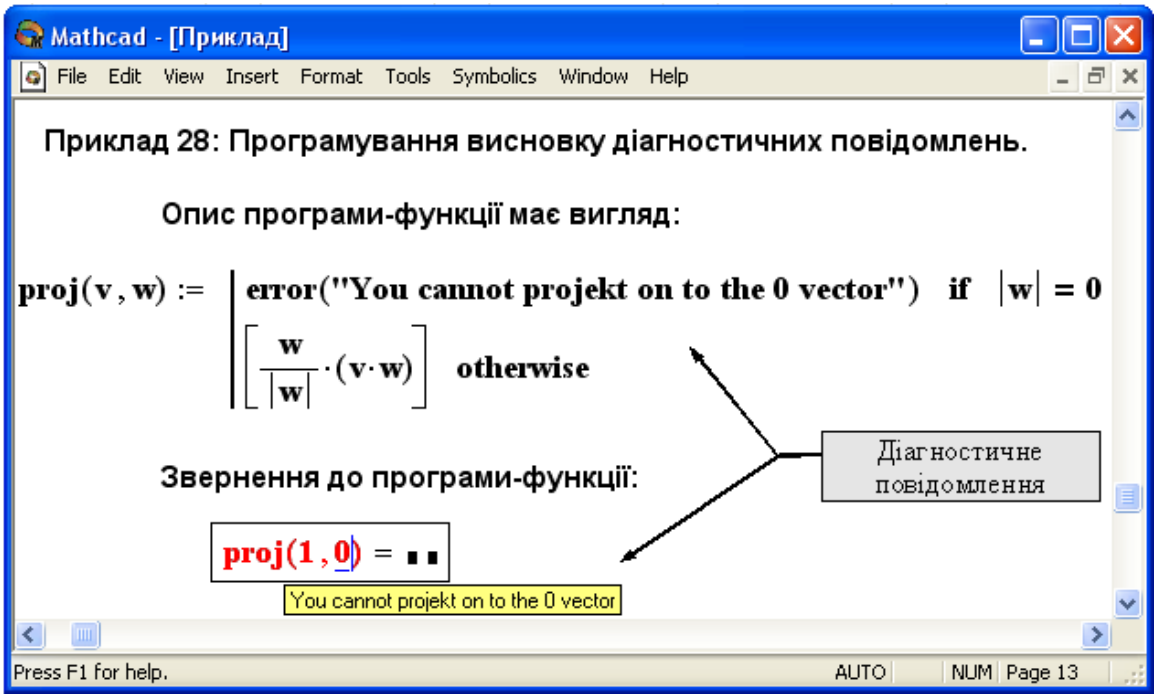


Рис. 3.40. Використання функції error у вікні документа MathCAD

Приклад. Використавши функцію error, визначити факторіал числа за умови, що число має бути ціле та додатнє, тобто потрібно врахувати три умови, за яких можемо визначити факторіал числа. На рис. 3.41 наведено реалізацію визначення факторіала числа у вікні документа MathCAD.

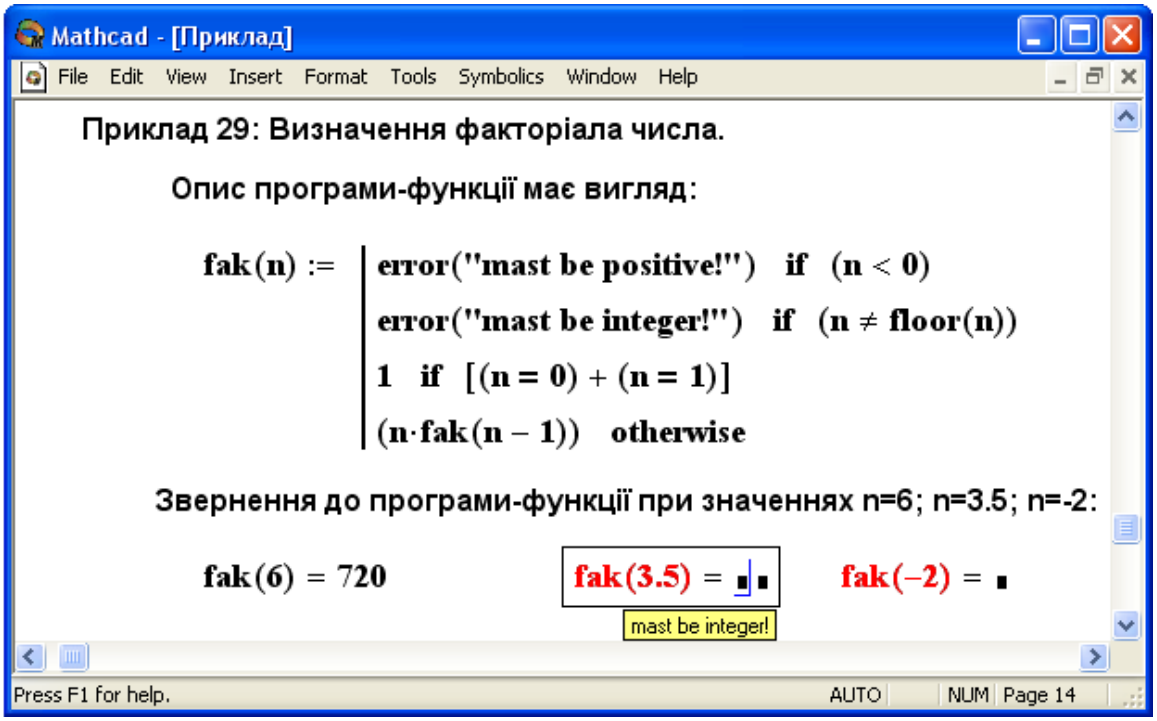


Рис. 3.41. Використання функції error за трьох умов у вікні документа MathCAD

3.4. Модульне програмування в математичному пакеті MathCAD

Загальна ідея модульного програмування передбачає:

- реалізацію обчислювальних процесів у вигляді окремих програмних одиниць – модулів;

- звертання до цих модулів в інших програмах із передачею даних, необхідних для обчислювального процесу.

Ідею модульного програмування широко використовують у сучасних алгоритмічних мовах програмування.

3.4.1. Модульне програмування в межах одного документа MathCAD

Модульне програмування в одному документі характеризується тим, що:

- для реалізації простих обчислень використовуються локальні функцій, а більш складних – програми-функції;

- опис локальних функцій, програм-функцій та їхній виклик (тобто звернення до них) перебувають у межах одного документа та зберігаються в одному файлі. При цьому часто всередині однієї програми-функції перебувають виклики локальних функцій, вбудованих функцій MathCAD та іншої програми-функції (рис. 3.42).

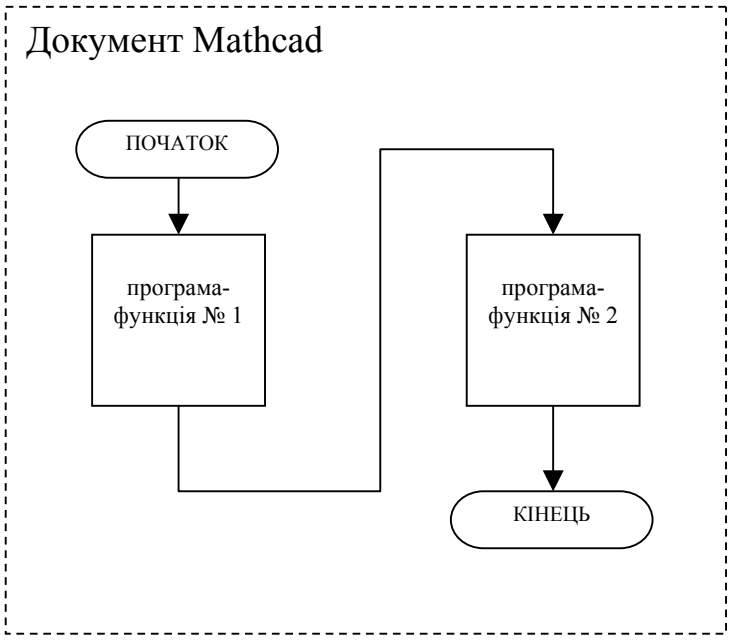


Рис. 3.42. Структурна схема модульного програмування
в межах одного документа

Приклад. У межах одного документа MathCAD у вигляді програми-функції обчислити інтеграл виду:

$$\int_a^b f(x)\,dx.$$

Використовуємо формулу Симпсона з автоматичним вибором кількості вузлів. При цьому програма-функція Simpson (f, a, b, N) обчислює певний інтеграл за формулою Симпсона за фіксованої кількості інтервалів N, а програма-функція Adapt (f, a, b) вибирає кількість інтервалів за заданої точності обчислення інтеграла, що дорівнює 10^{-8}.

Використовуючи ці програми-функції, обчислимо певний інтеграл від функції $f(x) = x^2$ на відрізку [0, 1]. Точне значення інтеграла дорівнює 1/3=0.33333333333333... На рис. 3.43 наведено зазначені програми-функції у вікні документа MathCAD.

Модульне програмування дозволяє зменшити об'єм вихідних текстів програм, зробити їх більш простими, прискорити написання та тестування програм, зменшити витрати на супровід (експлуатацію) програм [8].

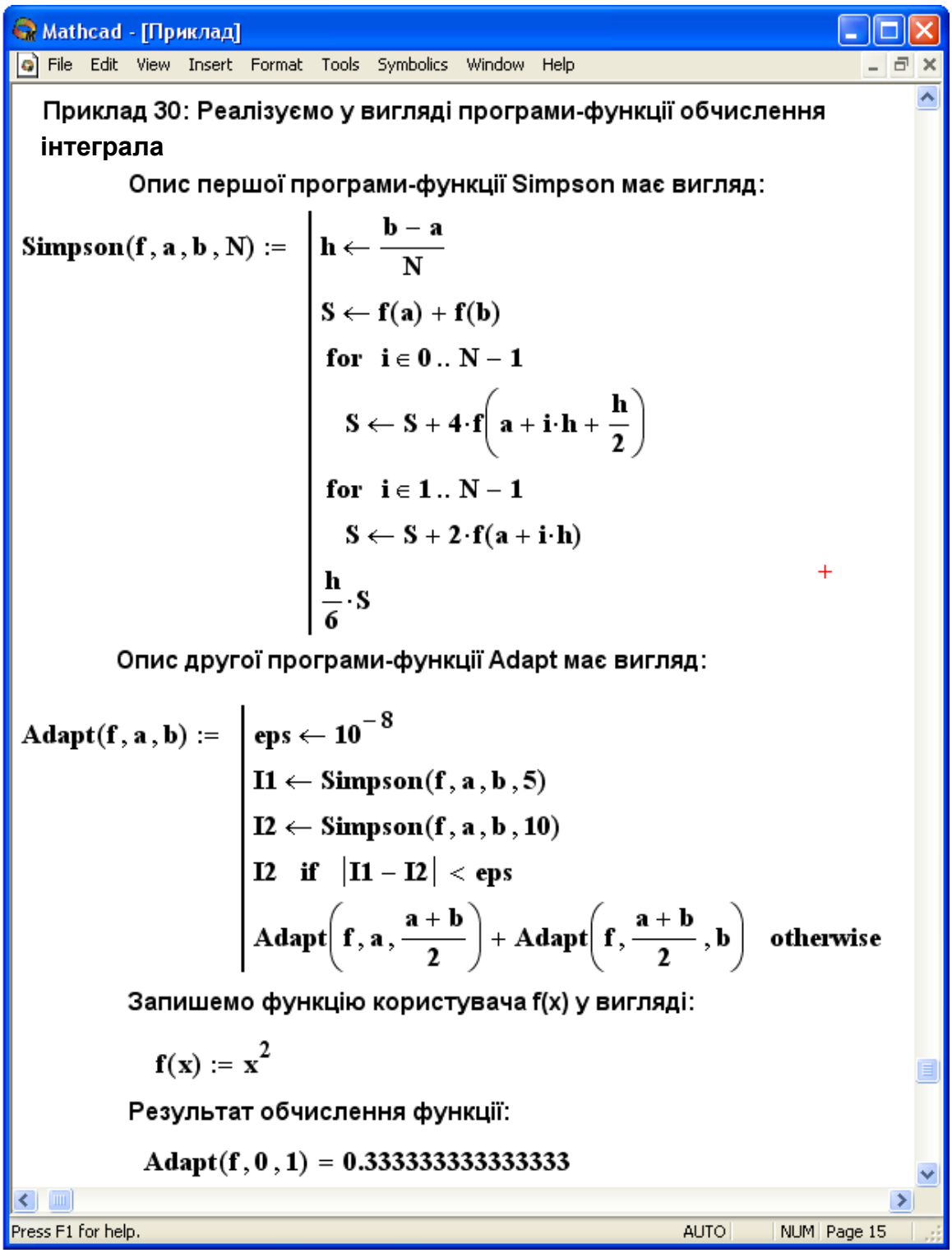


Рис. 3.43. Модульне програмування в межах одного документа MathCAD

3.4.2. Модульне програмування у декількох документах MathCAD

Реалізація модульного програмування, за яким опис модулів (функцій користувача та програм-функцій) і їх виклик перебуває в одному документі, має низку недоліків:

- неможливість паралельної розробки програм декількома розроблювачами;

- неможливість автономного налагодження програм-функцій та їхньої модифікації в процесі експлуатації програмного забезпечення;

- неможливість використання розробленої програми-функції в декількох документах без дублювання опису програми-функції.

Для подолання цих недоліків опис програми-функції виконують в одному документі MathCAD, а її виклик розміщують в іншому документі (рис. 3.44).

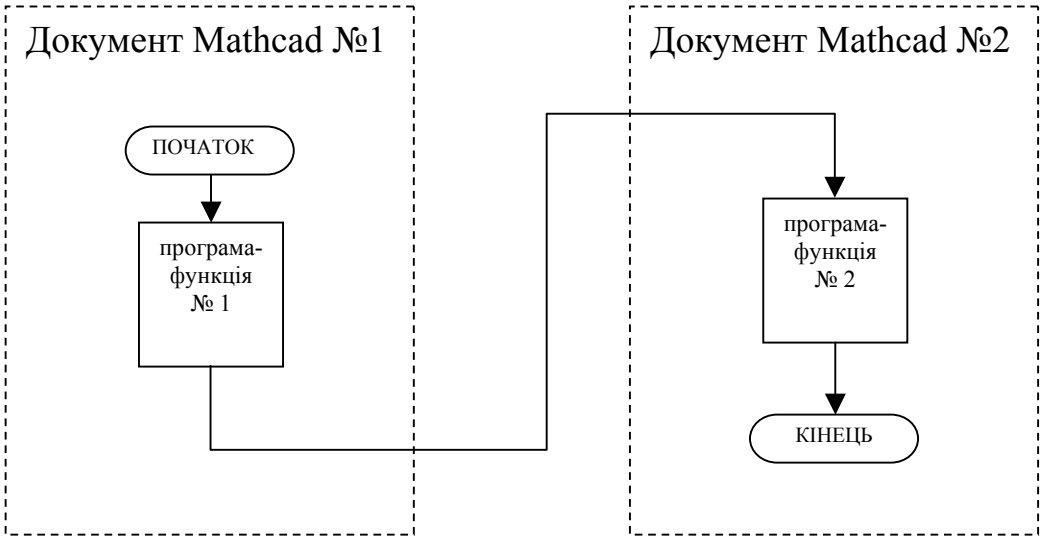


Рис. 3.44. Структурна схема модульного програмування у декількох документах

Однак при цьому виникає запитання: «Як під час виклику програми-функції в одному документі «приєднати» файл із іншим документом MathCAD, у якому перебуває опис програми-функції, що викликається?» Для такого приєднання існує спеціальний оператор **Reference,** що записується у вигляді, показаному на рис. 3.45.

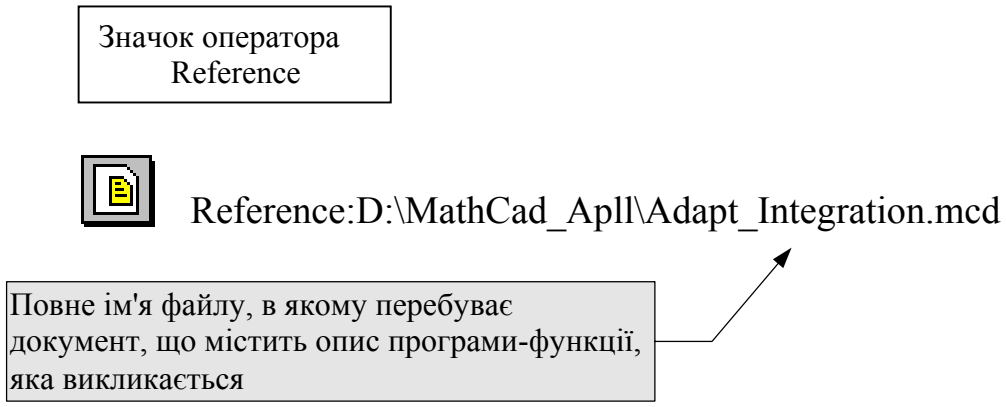


Рис. 3.45. Структура оператора Reference

Оператор **Reference** вставляється в текст документа, у якому викликається програма-функція перед її викликом. Для вставки цього оператора необхідно виконати наступні кроки:

Крок 1. Клацнути лівою кнопкою миші в тому місці, куди буде вставлений оператор **Reference.**

Крок 2. Звернутися до пункту меню **Insert** і виконати команду **Reference.**

Крок 3. У поле діалогового вікна, що з'явилося, ввести повне ім'я файлу, що містить документ із описом функції, що викликається. Для введення імені необхідно клацнути кнопку **Browse** і у діалоговому вікні, що з'явилося, указати диск, папку та ім'я файлу (у поле вводимо повне ім'я файлу).

Крок 4. Після виконаних установок натиснути кнопку **Ok**.

Після виконання цих кроків у документі з'явиться оператор **Reference** (рис. 3.45).

У такий спосіб реалізація модульного програмування в декількох документах MathCAD включає наступні етапи (які будемо ілюструвати на прикладі обчислення певного інтеграла з використанням програм-функцій):

- Опис у документі MathCAD необхідних програм-функцій і збереження цього документа у файлі в потрібній папці та під потрібним ім'ям (у нашому прикладі документ буде включати опис двох програм-функцій **Simpson** і **Adapt** і буде збережений на диску D: у папці MathCAD_Apll у файлі під ім'ям Adapt_Integration.mcd).
- Вставка оператора **Reference** у документі, у якому викликаються описані програми-функції шляхом виконання кроків 1-4, вставлений оператор **Reference** матиме вигляд:

Reference:D:\MathCad_Apll\Adapt_Integration.mcd

- Виклик потрібних програм-функцій. Виклик може мати такий вигляд:

$$f(x) := x^2$$

$$Adapt(f, 0, 1) = 0.33333333$$

Точне значення інтеграла
0.3333333333333333..

$$g(x) := \sin(x) + \cos(2 \cdot x)$$

$$Adapt\left(g, \pi, 3 \cdot \frac{\pi}{2}\right) = -1.0000000001$$

Точне значення інтеграла
- 1.00000000000000

Отже, описана реалізація модульного програмування дозволяє створювати бібліотеки програм-функцій, що реалізують обчислювальні алгоритми різної складності для різних предметних областей і використання бібліотеки програми-функції, розроблені іншими користувачами.

Оператор **Reference** дозволяє зчитати інформацію з файлу, що має розширення **.mcd*. Для зчитування та запису файлу з іншим розширенням використовують оператори **File Output** та **File Input**.

Оператор **File Output** вставляється в текст документа для зчитування інформації з файлу. Оператор **File Input** вставляється в текст документа для запису інформації у файл.

Для вставки цих операторів необхідно виконати такі кроки:

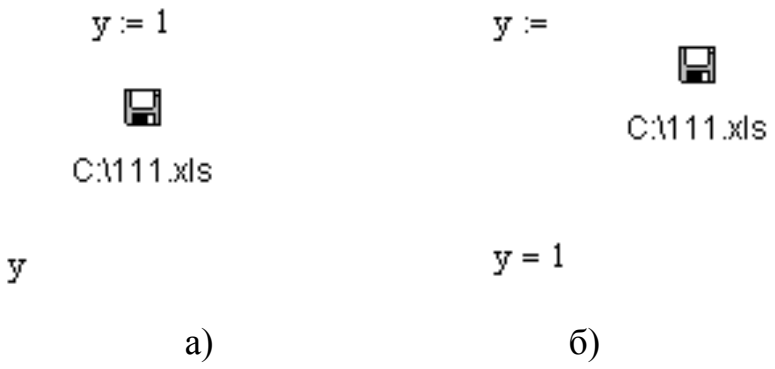
Крок 1. Клацнути лівою кнопкою миші в тому місці, куди буде вставлений оператор.

Крок 2. Звернутися до пункту меню **Insert**, кликнути мишею на вкладку **Data** та вкладку **File Output** чи **File Input.**

Крок 3. У поле діалогового вікна, що з'явилося, ввести повне ім'я файлу, що містить документ із даними, які зчитуються чи записуються. Необхідно вибрати формат файлу. Для введення шляху до файлу необхідно клацнути кнопку **Browse** і у діалоговому вікні, що з'явилося, указати диск, папку та ім'я файлу (у поле вводимо повне ім'я файлу).

Крок 4. Після виконаних установок натиснути кнопку **Готово**.

Після виконання цих кроків у документі з'явиться оператор **File Output** чи **File Input**, якому привласнюємо ім’я змінної (рис. 3.46).



а) б)

Рис. 3.46. Використання операторів:
а) для запису File Output; б) для зчитування File Input

Висновки

Математичне середовище MathCAD – це програмний пакет для роботи з формулами, числами, текстами та графіками. Він дозволяє отримувати числові розв'язки простих та складних функцій, які містять всі математичні дії, проводити символьне обчислення виразів, створювати програмні алгоритми.

Логічним виразом називається конструкція, що складена з виразів відношення, знаків логічних операцій і круглих дужок.

Програма-функція містить у собі ім'я програми-функції, список формальних параметрів та тіло програми-функції. Опис програми-функції розміщується в робочому документі MathCAD перед її викликом.

Умовний оператор if відноситься до базової структури, яка має назву вибір, та припускає перевірку деякої умови. Залежно від того, виконується ця умова чи ні, залежить виконання тієї чи іншої інструкції.

Циклічні алгоритми містять обчислення, що повторюються певну кількість разів. Кількість повторень обчислень залежить від деякої змінної. Така змінна називається параметром циклу, а самі обчислення,

що повторюються, становлять тіло циклу.

Характерною рисою циклів типу арифметичної прогресії є те, що кількість повторень тіла циклу можна визначити до початку виконання програми, що реалізує цикл. Для програмування циклу типу арифметичної прогресії використовують оператор циклу for.

Для програмування ітераційних циклів використовують оператор циклу while. Оператор while відносять до базової структури, яку називають циклічною. Кількість повторень тіла циклу залежить від умови виконання циклу. Якщо виконується умова виконання циклу, то інструкція буде виконана, якщо не виконується умова виконання циклу, то інструкція не буде виконана і відбудеться вихід із циклу.

Оператор break дозволяє запобігти зацикленню програми, тобто вийти із циклу або призупинити роботу програми, перевіривши задану в операторі break умову. Оператор break використовується в лівому полі введення умовного оператора if.

Оператор continue використовується для продовження виконання циклу шляхом повернення на початок тіла циклу. Оператор continue використовується в лівому полі введення умовного оператора if.

Оператор return перериває виконання програми-функції та повертає значення операнда, що знаходиться за ним. Оператор return використовується в лівому полі введення умовного оператора if.

Оператор on error обробляє помилки, що виникають при виконанні тих або інших обчислень.

Функцію error використовують для формування діагностичних повідомлень у разі виникнення в обчисленнях помилки. Функцію error використовують у лівому полі введення умовного оператора if.

Модульне програмування в математичному пакеті MathCAD передбачає: реалізацію обчислювальних процесів у вигляді окремих програмних одиниць – модулів; звертання до цих модулів в інших

програмах з передачею даних, необхідних для обчислювального процесу.

Контрольні запитання та завдання

1. Охарактеризуйте призначення виразів відношення.
2. Які існують логічні операції в MathCAD?
3. Що собою являють текстові фрагменти?
4. Яка функція називається умовною?
5. Поясніть необхідність друкування ключового слова Given в MathCAD?
6. Для чого використовують функцію Find?
7. Для чого використовують функцію Minerr?
8. Як створити тіло програми-функції?
9. Дайте визначення оператора for?
10. Порядок опису програми-функції в MathCAD.
11. Ім’я програми-функції. Як ввести список формальних параметрів?
12. Дайте визначення оператора if.
13. Дайте визначення оператора otherwise.
14. В якому випадку обчислюється вираз, що розташований перед словом otherwise?
15. Що таке параметр циклу?
16. Дайте визначення терміну «тіло циклу».
17. Який тип алгоритмів реалізує оператор for?
18. Дайте визначення оператора while.
19. Структура оператора while.
20. Який тип алгоритмів реалізує оператор while?
21. Коли необхідно використовувати оператор break?
22. Дайте визначення оператора continue.

23. Дайте визначення оператора return.

24. Дайте визначення оператора on error.

25. Функція error. З яким оператором використовують функцію error?

26. Загальна ідея модульного програмування.

27. Назвіть методи реалізації модульного програмування в пакеті MathCAD.

28. Назвіть недоліки модульного програмування в одному документі?

29. Чим характеризується модульне програмування в декількох документах MathCAD?

30. Назвіть призначення оператора Reference.

31. Назвіть призначення операторів File Output та File Input.

Вправи

1. Побудувати двохвимірні графіки функцій *y(x)* та *z(x)* за індивідуальним варіантом (табл. 3.2).

Таблиця 3.2

Вихідні дані для розрахунку за індивідуальним варіантом

Варіант		Функція 1	Функція2
1	16	$y = \sin(x)$	$z = \exp(x+3)/5000 - 1$
2	17	$y = \cos(x)$	$z = 0.00025e^{3-x} - 0.6$
3	18	$y = \lvert tg(x)\rvert + 1$	$z = (1+x)^6$
4	19	$y = (x^2 - 1)/15$	$z = 1 + \sin(x)$
5	20	$y = (x^3 - 2)/15$	$z = 5\cos(x)$
6	21	$y = (x^2 - 10)$	$z = 0.025e^{-1.2x}$
7	22	$y = \int_{-5}^{x} \sin(x)dx$	$z = 0.02x^3$
8	23	$y = \frac{d}{dx}(\sin(x) + 7 \times \ln(x))$	$z = 0.05x^2$
9	24	$y = \frac{d}{dx}(e^{1-0,2x\cdot\sin(x)})$	$z = 0.01x^3$
10	25	$y = \sqrt{2} + \cos(x)$	$z = -0.05(x2 + 10\cos(x))$
11	26	$y = \sin^2(x/3)$	$z = 0.01(x^2 - 40\sin(x))$
12	27	$y = \cos^3(x)$	$z = \sin(x) + \sin(2x)$
13	28	$y = 0.5x + \cos^2(x)$	$z = \sin^2(x) + \cos(x)$
14	29	$y = \sin(x) + \cos^2(2x)$	$z = x(0.5 + x)\exp(0.1x)$
15	30	$y = \lvert\sin(x)\rvert \exp(x/2)$	$z = 5x - x^{1.5+\sin(x)}$

2. Визначити струми (I_0, I_1, I_2, I_3, I_4) всіх віток електричного кола, схему якого наведено на рис. 3.46, якщо система рівнянь для цієї схеми складена на основі I та II законів Кірхгофа, матиме вигляд:

$$\begin{cases} I_0 + I_2 = I_4; \\ I_2 + I_3 = I_1; \\ R_1 \cdot I_1 + R_2 \cdot I_2 - R_0 \cdot I_0 = 0; \\ R_2 \cdot I_2 + R_4 \cdot I_4 - R_3 \cdot I_3 = 0; \\ R_0 \cdot I_0 + R_4 \cdot I_4 = E. \end{cases}$$

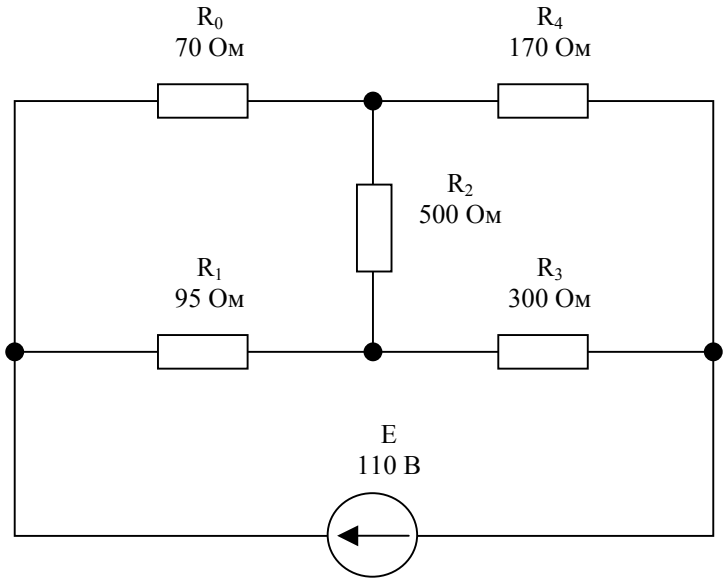


Рис. 3.46. Схема електричного кола.

3. За допомогою функції *root* розв'яжіть нелінійне рівняння $f1 = a3 \cdot x^3 + a2 \cdot x^2 + a1 \cdot x + a0$. Для цього потрібно за індивідуальним варіантом підставити значення коефіцієнтів полінома (табл. 3.3), задати функцію *f1(x)* та визначити *x*.

Таблиця 3.3

Значення коефіцієнтів полінома

№ Варіанта		**Коефіцієнти полінома *f1***				***f2***
		a0	**a1**	**a2**	**a3**	
1	16	5	-10	-2	a3= № варіанта	$x^4 - 2x^3 + x^2 - 12x + 20 = b$
2	17	7,5	-9	-2,5		$x^4 + 6x^3 + x^2 - 4x - 60 = b$
3	18	10	-8	-3		$x^4 - 14x^2 - 40x - 75 = b$
4	19	12,5	-7	-3,5		$x^4 - x^3 + x^2 - 11x + 10 = b$
5	20	15	-6	-4		$x^4 - x^3 - 29x^2 - 71x - 140 = b$
6	21	17,5	-5	-4,5		$x^4 + 7x^3 + 9x^2 + 13x - 30 = b$
7	22	20	-4	-5		$x^4 + 3x^3 - 23x^2 - 55x - 150 = b$
8	23	22,5	-3	-5,5		$x^4 - 6x^3 + 4x^2 + 10x + 75 = b$
9	24	25	-2	-6		$x^4 + x^3 - 17x^2 - 45x - 100 = b$

№ Варіанта		Коефіцієнти полінома *f1*				*f2*
		a0	**a1**	**a2**	**a3**	
10	25	27,5	-1	-6,5		$x^4 - 5x^3 + x^2 - 15x + 50 = b$
11	26	30	0	-7		$x^4 - 4x^3 - 2x^2 - 20x + 25 = b$
12	27	32,5	1	-7,5		$x^4 + 5x^3 + 7x^2 + 7x - 20 = b$
13	28	35	2	-8		$x^4 - 7x^3 + 7x^2 - 5x + 100 = b$
14	29	37,5	3	-8,5		$x^4 + 10x^3 + 36x^2 + 70x + 75 = b$
15	30	40	4	-9		$x^4 + 9x^3 + 31x^2 + 59x + 60 = b$

b – номер групи.

4. За допомогою функції *minerr* наближено розв'яжіть систему нелінійних рівнянь $\begin{cases} f1; \\ f2. \end{cases}$, де $f1 = a3 \cdot x^3 + a2 \cdot x^2 + a1 \cdot x + a0 = b$ та $f2$ за індивідуальним варіантом (табл. 3.3).

5. Запишіть функцію щільності заповнення пташника птицею та порівняйте отриманий результат з табличним (14-16 кг/м^2), вихідні дані за індивідуальним варіантом взяти з табл. 3.4.

Інструкція до виконання.

Щільність заповнення пташника птицею визначаємо за формулою:

$$M = \frac{n_i \cdot P}{a \cdot b}, \text{ кг/м}^2,$$

де n_i – кількість птиці, що знаходиться у пташнику (за варіантом), шт;

P – вага птиці (за варіантом), кг;

a та b – відповідно ширина та довжина пташника дорівнює 18х80, м.

Якщо у разі порівняння розрахункового та табличного значення щільності заповнення пташника птицею розрахункове значення знаходиться в межах табличного значення, то має з'явитись повідомлення «spivpalo», в іншому випадку – «ne spivpalo» .

Таблиця 3.4

Значення величин для розрахунку за варіантом

№ варіанта	n_i, *шт*	*P, кг*	№ варіанта	n_i, *шт*	*P, кг*
1	30000	1,1	16	22500	1,475
2	29500	1,125	17	22000	1,5
3	29000	1,15	18	21500	1,525
4	28500	1,175	19	21000	1,55
5	28000	1,2	20	20500	1,575
6	27500	1,225	21	20000	1,6
7	27000	1,25	22	19500	1,625
8	26500	1,275	23	19000	1,65
9	26000	1,3	24	18500	1,675
10	25500	1,325	25	18000	1,7
11	25000	1,35	26	17500	1,725
12	24500	1,375	27	17000	1,75
13	24000	1,4	28	16500	1,775
14	23500	1,425	29	16000	1,8
15	23000	1,45	30	15500	1,825

6. Запишіть функцію визначення більшого значення з двох чисел *a* та *b*, блок-схему якої наведено на рис. 3.47.

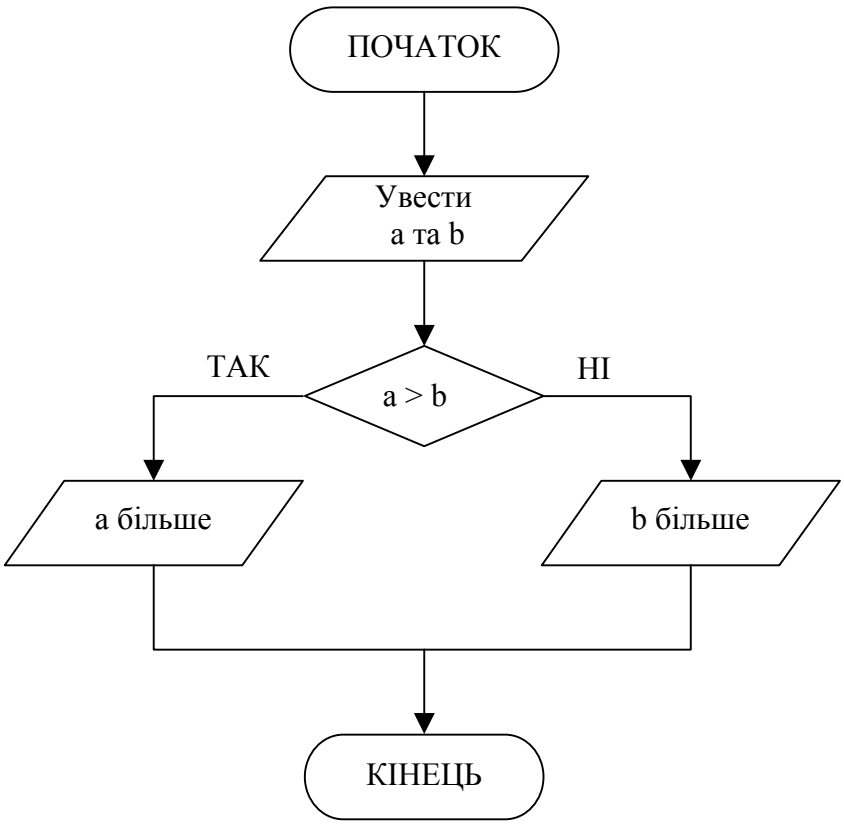


Рис. 3.47. Блок-схема визначення більшого значення числа з двох чисел *a* та *b*

7. Спростіть вираз $f(x,y) := \frac{\sin\left(\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}} + \sqrt{x^2+y^2}$ та запишіть його у вигляді програми-функції.

8. Розробіть алгоритм та програму-функцію визначення необхідної кількості теплоти від теплогенератора, яка відводиться на випаровування вологи в пташнику з напільним утриманням птиці за індивідуальним варіантом.

Інструкція до виконання.

Визначимо кількість вологи, що виділяється в пташнику, враховуючи водяні пари, що виділяються птицею і послідом:

$$W_{пос} = \frac{0.7 \cdot P_{пос} \cdot n_i}{24}, \text{ г/доб},$$

де $P_{пос}$ *=175* – середньодобовий вихід посліду від однієї птиці, г;

n_i – кількість птиці, що знаходиться у пташнику, шт.;

0,7 – коефіцієнт, який враховує усушку посліду.

Вологовиділення птицею визначаємо за виразом:

$$W_{пт} = K_w \cdot W_i \cdot n_i \cdot P, \text{ г/год},$$

де W_i *=4,5* – виділення вологи однією птицею, г/год·кг;

K_w – температурний коефіцієнт вологовиділень.

Залежність коефіцієнта K_w від температури Θ, наведено в табл. 3.5;

P – вага птиці, кг.

Таблиця 3.5

Залежність температурного коефіцієнта вологовиділень від температури

$\Theta, {}^oC$	4	8	12	16	20	24	28	32
K_W	0,85	0,9	0,95	1	1,02	1,05	1,22	1,32

Знаходимо сумарні вологовиділення у пташнику:

$W = W_{пт} + W_{пос}$, г/доб ,

де $W_{пт}$ – вологовиділення птицею, г/доб;

$W_{пос}$ – виділяється з посліду, г/доб.

Визначимо втрати теплоти $Q_{вип}$ на випаровування вологи з відкритих водних та мокрих поверхонь, а також у разі усушки посліду.

$Q_{вип} = 0.69 \cdot W$, Вт,

де W – сумарні вологовиділення у пташнику, г/год.

Необхідно записати програму-функцію з ім’ям tepl. Здійснити розрахунок, підставивши замість формальних параметрів їх фактичні значення, відповідно до свого варіанта (табл. 3.6).

Таблиц1я 3.6

Значення величин для розрахунку за варіантом

№ варіанта	n_i **, шт**	$\Theta, {}^oC$	**P, кг**	**№ варіанта**	n_i **, шт**	$\Theta, {}^oC$	**P, кг**
1	30000	4	1,1	16	22500	32	1,475
2	29500	8	1,125	17	22000	4	1,5
3	29000	12	1,15	18	21500	8	1,525
4	28500	16	1,175	19	21000	12	1,55
5	28000	20	1,2	20	20500	16	1,575
6	27500	24	1,225	21	20000	20	1,6

№ варіанта	n_i, *шт*	*Θ*, ^oC	*P, кг*	№ варіанта	n_i, *шт*	*Θ*, ^oC	*P, кг*
7	27000	28	1,25	22	19500	24	1,625
8	26500	32	1,275	23	19000	28	1,65
9	26000	4	1,3	24	18500	32	1,675
10	25500	8	1,325	25	18000	4	1,7
11	25000	12	1,35	26	17500	8	1,725
12	24500	16	1,375	27	17000	12	1,75
13	24000	20	1,4	28	16500	16	1,775
14	23500	24	1,425	29	16000	20	1,8
15	23000	28	1,45	30	15500	24	1,825

9. Оформити у вигляді програми-функції розв'язок квадратного рівняння $ax^2 + bx + c = 0$. Розробити програму-функцію kvadr розв'язку квадратного рівняння, використавши формальні параметри *a, b, c*.

Інструкція до виконання.

Дискримінант обраховують за формулою: $D = b^2 - 4ac$.

Розв'язок рівняння: якщо D>0, то $x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$, якщо D=0, то $x = \frac{-b}{2a}$, а якщо D<0, то розв'язків рівняння не існує.

10. Розробити алгоритм та програму-функцію визначення необхідного повітрообміну в пташнику. Визначення необхідної кількості повітря для системи вентиляції пташників проводиться за умов видалення з приміщення шкідливих газів, надлишкової вологи та надлишкового тепла.

Інструкція до виконання.

Розрахунок повітрообміну для пташників в холодну пору року враховує надлишок вологи та газів, а в теплу та перехідну пору року – надлишок теплоти та вологи. Вміст сірководню та аміаку суттєво не впливає на розрахунок повітрообміну в пташнику, достатньо врахувати повітрообмін для розчинення вуглекислого газу.

Повітрообмін, який потрібен для підтримання необхідної концентрації вуглекислого газу для пташника, визначаємо за формулою:

$$Lco_2 = \frac{n_i \cdot P \cdot C_{nm}}{C_1 - C_2}, \text{ м}^3\text{/год},$$

де n_i – кількість птиці у приміщенні, *гол*;

C_{nm} = 1,7 – кількість вуглекислого газу, що виділяє птиця, *л/год·кг;*

C_1 та C_2 – концентрація вуглекислого газу в припливному повітрі та гранично допустима норма у повітрі, що видаляється (C_1=*2,0 л/м*^{3} та C_1=*0,3 л/м*^{3});

P – жива маса однієї птиці, *кг*. Вага птиці від початку несучості до її закінчення постійно зростає залежно від породи птиці (табл. 3.7).

Таблиця 3.7

Залежність ваги птиці від віку

Вік птиці, днів	170	200	230	260	290	320	350	380	410	440
Вага несучок, кг	1,43	1,52	1,59	1,63	1,65	1,66	1,68	1,69	1,71	1,73
Вага м’ясної породи, кг	2,83	3,15	3,27	3,29	3,31	3,34	3,37	3,42	3,46	3,54

Повітрообмін приміщення за умови видалення теплових надлишків знаходимо за формулою:

$$L_q = \frac{Q_я}{c \cdot C_p \cdot (И_в - И_з)}, \text{ м}^3\text{/год},$$

де $Q_я$=*350 тис.* – виділення в приміщення явної теплоти, *Вт*;

$C_p = 1$ – питома теплоємність повітря, *кДж/(кг·°С);*

$И_в$, $И_з$ – температура внутрішнього та зовнішнього повітря, ^oC;

ρ – густина повітря (кг/ м^3) залежить від температури (табл. 3.8).

Таблиця 3.8

Залежність густини повітря від температури

Θ, °С	-20	-10	0	10	20	30
c , кг/ м^3	1,396	1,342	1,293	1,248	1,205	1,165

Кількість повітря, яку необхідно видалити з приміщення за надлишком вологи, визначаємо за формулою:

$$L_w = \frac{W}{\rho \cdot C_p \cdot (d_в - d_з)} \text{ , } м^3/год,$$

де $W=550$ *тис.* – сумарне надходження вологи в приміщення, *г/год*;

d_B , $d_З$ – відповідно вологовміст повітря в середині та зовні приміщення, *г/кг*.

Враховуючи, що в птахівничих приміщеннях одночасно доводиться коригувати якісний склад повітря за надлишковим виділенням теплоти, вологовиділенням та кількістю шкідливих газів, для розрахунку необхідно брати найбільший за значенням показник, тим самим задовольнивши вимоги всіх складових повітрообміну.

Записати програму-функцію з іменем tepl та як формальні параметри використати значення, що наведені в табл. 3.9 відповідно до варіанта індивідуального завдання.

Таблиця 3.9

Значення величин для розрахунку за варіантом

№ варіанта	n_i, *шт*	*Вік, днів*	*порода*	$\Theta_в$, ^oC	$\Theta_з$, ^oC	$d_В$, *г/кг*	$d_З$, *г/кг*
1	22500	170	несуча	12	-20	3	2
2	2200	200	м'ясна	13	-10	6	4
3	21500	230	несуча	14	0	8	6
4	2100	260	м'ясна	15	10	14	12
5	20500	290	несуча	16	20	20	18
6	2000	320	м'ясна	17	-20	4	2
7	19500	350	несуча	18	-10	7	4
8	1900	380	м'ясна	19	0	9	6
9	18500	410	несуча	12	10	15	12
10	1800	440	м'ясна	13	20	21	18
11	17500	170	несуча	14	-20	3	2
12	1700	200	м'ясна	15	-10	6	4
13	16500	230	несуча	16	0	8	6
14	1600	260	м'ясна	17	10	14	12
15	15500	290	несуча	18	20	20	18
16	2250	320	м'ясна	19	-20	4	2
17	22000	350	несуча	12	-10	7	4
18	2150	380	м'ясна	13	0	9	6
19	21000	410	несуча	14	10	15	12
20	2250	440	м'ясна	15	20	21	18
21	22000	170	несуча	16	-20	3	2
22	2150	200	м'ясна	17	-10	6	4
23	21000	230	несуча	18	0	8	6
24	2050	260	м'ясна	19	10	14	12
25	20000	290	несуча	12	20	20	18
26	1950	320	м'ясна	13	-20	4	2
27	19000	350	несуча	14	-10	7	4
28	1850	380	м'ясна	15	0	9	6
29	18000	410	несуча	16	10	15	12
30	1750	440	м'ясна	17	20	21	18

11. Використовуючи оператор **for**, напишіть програму-функцію для розрахунку суми всіх чисел на інтервалі від *ap* (початкове значення) до *ak* (кінцеве значення).

12. Використовуючи оператор **for**, напишіть програму-функцію визначення сумарної витрати теплоти на опалення та вентиляцію.

Інструкція до виконання.

Витрату теплоти на опалення та вентиляцію кожної будівлі визначимо за формулою:

$$Q = \left(q_o \cdot \left(t_в - t_{зо}\right) + q_в \cdot \left(t_в - t_{зв}\right)\right) \cdot V_{прим}\text{, Вт,}$$

де $q_o = 0.7$ та $q_в = 0.17$ – питомі теплові характеристики для опалення та вентиляції, $Вт/м^3 \cdot К$;

$t_{зо} = -18$ та $t_{зв} = -6$ – розрахункові температури зовнішнього повітря для проектування опалення та вентиляції, ^oC;

$V_{прим}$ – об'єм приміщення, $м^3$.

Визначаємо витрати теплоти на опалення та вентиляцію для п'яти різних приміщень, об'єми яких наведено в таблиці 3.10.

Необхідно записати програму-функцію з іменем op_ven та підставити як фактичні параметри значення, що наведені в таблиці 3.10.

Таблиця 3.10

Значення величини $V_{прим}$ для розрахунку за варіантом

№ варіанта	$V_{прим}$, $м^3$				
	1	**2**	**3**	**4**	**5**
1	3120	2830	2385	1705	640
2	3125	2835	2390	1710	650
3	3130	2840	2395	1715	660
4	3135	2845	2400	1720	670
5	3140	2850	2405	1725	680
6	3145	2855	2410	1730	690
7	3150	2860	2415	1735	700
8	3155	2865	2420	1740	710
9	3160	2870	2425	1745	720
10	3165	2875	2430	1750	730
11	3170	2880	2435	1755	740
12	3175	2885	2440	1760	750
13	3180	2890	2445	1765	760
14	3185	2895	2450	1770	770

№ варіанта	$V_{прим}$, $м^3$				
	1	**2**	**3**	**4**	**5**
15	3190	2900	2455	1775	780
16	3195	2905	2460	1780	790
17	3200	2910	2465	1785	800
18	3205	2915	2470	1790	810
19	3210	2920	2475	1795	820
20	3215	2925	2480	1800	830
21	3220	2930	2485	1805	840
22	3225	2935	2490	1810	850
23	3230	2940	2495	1815	860
24	3235	2945	2500	1820	870
25	3240	2950	2505	1825	880
26	3245	2955	2510	1830	890
27	3250	2960	2515	1835	900
28	3255	2965	2520	1840	910
29	3260	2970	2525	1845	920
30	3265	2975	2530	1850	930

13. Використовуючи оператор **while**, напишіть програму-функцію для обчислення суми всіх чисел на інтервалі від *ap* (початкове значення) до *ak* (кінцеве значення).

14. Розробити алгоритм та програму-функцію знаходження спільної точки двох функцій *f1(x)* та *f2(x)* відповідно до індивідуального варіанта (табл. 3.11).

Інструкція до виконання.

Записати програму-функцію знаходження спільної точки.

Побудувати графіки функцій $f1(x)=b+a\cdot x$ та $f2(x)=c\cdot b-a\cdot x$ в MathCAD (рис. 3.48).

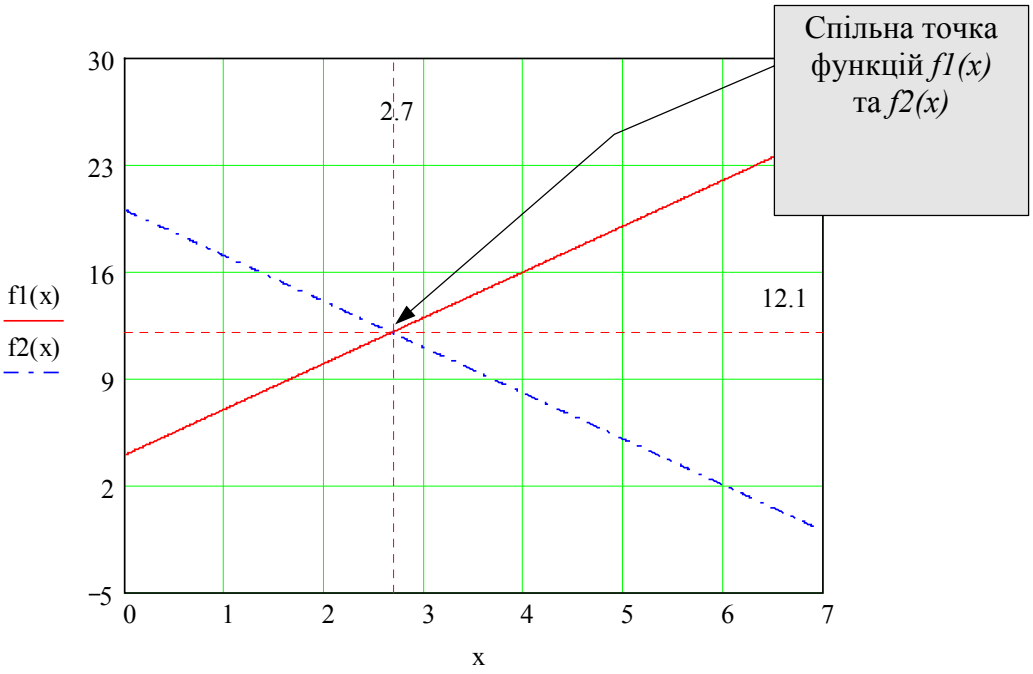


Рис. 3.48. Графічне зображення функцій *f1(x)*та *f2(x)*

Таблиця 3.11

Значення величин для розрахунку за варіантом

№ Варіанта		**Коефіцієнти функцій *f1(x)* та *f2(x)***		
		a	***b***	***c***
1	16	1	5	*c*=№ варіанта
2	17	1,2	5,2	
3	18	1,4	5,4	
4	19	1,6	5,6	
5	20	1,8	5,8	
6	21	2	6	
7	22	2,2	6,2	
8	23	2,4	6,4	
9	24	2,6	6,6	
10	25	2,8	6,8	
11	26	3	7	
12	27	3,2	7,2	
13	28	3,4	7,4	
14	29	3,6	7,6	
15	30	3,8	7,8	

15. Написати програму-функцію для розрахунку суми чисел від *ap* до *ak* та призупинити підрахунок суми на деякому числі *as*, використовуючи оператор **break,** де (*ap*<*as*<*ak*). Задати ім'я програми-функції sum_break.

16. Розробити алгоритм та програму-функцію визначення максимального значення щільності розподілу випадкової величини *f(x)* відповідно до індивідуального варіанта (табл. 3.12).

Інструкція до виконання.

Розробити алгоритм визначення максимального значення щільності розподілу випадкової величини за виразом $f(x)=\frac{1}{\sqrt{2\pi\sigma^2}}\cdot\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$.

Побудувати програму-функцію визначення максимального значення щільності розподілу випадкової величини та відобразити її на графіку.

Побудувати функцію розподілу випадкової величини в MathCAD, її графічне зображення на рис. 3.49.

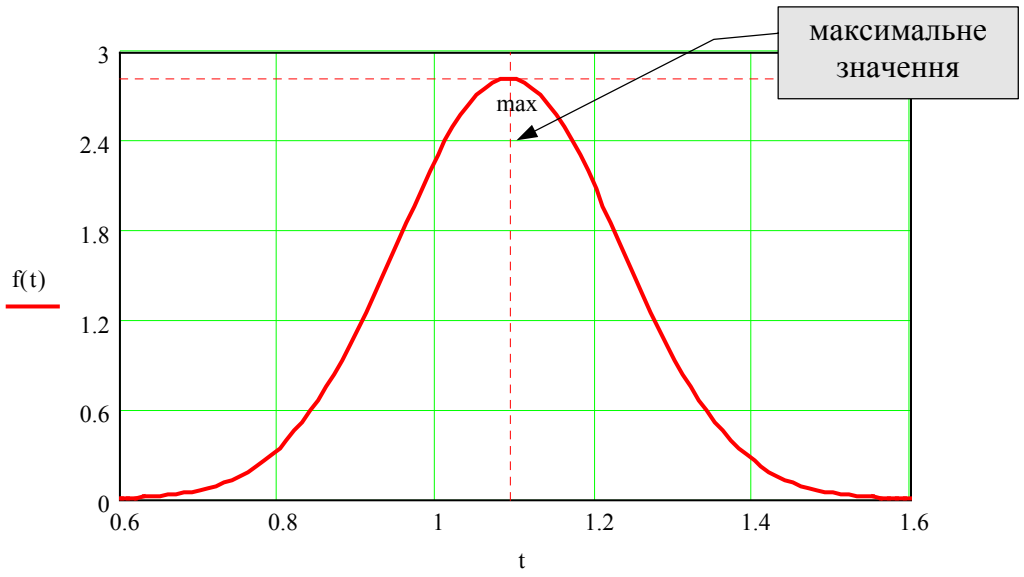


Рис. 3.49. Графічне зображення функцій *f(x)*

Таблиця 3.12

Значення величин для розрахунку за варіантом

№ варіанта	μ	σ	№ варіанта	μ	σ
1	0,5	0,1	16	1,25	0,4
2	0,55	0,12	17	1,3	0,42
3	0,6	0,14	18	1,35	0,44
4	0,65	0,16	19	1,4	0,46
5	0,7	0,18	20	1,45	0,48
6	0,75	0,2	21	1,5	0,5
7	0,8	0,22	22	1,55	0,52
8	0,85	0,24	23	1,6	0,54
9	0,9	0,26	24	1,65	0,56
10	0,95	0,28	25	1,7	0,58
11	1	0,3	26	1,75	0,6
12	1,05	0,32	27	1,8	0,62
13	1,1	0,34	28	1,85	0,64
14	1,15	0,36	29	1,9	0,66
15	1,2	0,38	30	1,95	0,68

17. Розробити алгоритм та програму-функцію визначення сумарної втрати теплоти через ґрунт у пташнику відповідно до індивідуального варіанта (табл. 3.13).

Інструкція до виконання.

Сумарні втрати теплоти через ґрунт у пташнику визначають за формулою:

$$Q_{гр} = \sum \frac{F_i}{R_i} \cdot (t_в - t_з), \text{ Вт},$$

де F_i – площа однієї зони; $м^2$;

R_i – опір теплопередачі однієї зони $(м^2 \cdot k)/Вт$;

$t_в$ та $t_з$ – температура повітря в середині пташника та зовні, ^oC.

Розмір пташника m х n за варіантом, розподіл площі на зони наведено на рис. 4.1: опір теплопередачі зон: $R_1=2,1$ $(м^2 \cdot k)/Вт$; $R_2=4,3$ $(м^2 \cdot k)/Вт$; $R_3=8,6$ $(м^2 \cdot k)/Вт$; $R_4=14,2$ $(м^2 \cdot k)/Вт$.

Побудувати програму-функцію визначення сумарної втрати теплоти через ґрунт у пташнику.

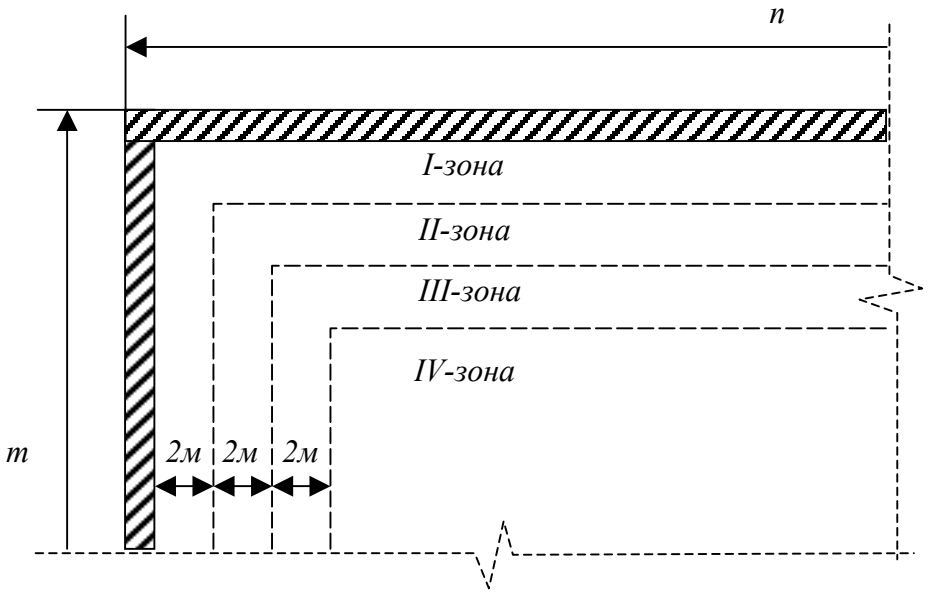


Рис. 3.50. Поділ площі в пташнику на зони

Таблиця 3.13

Значення величин для розрахунку за варіантом

№ варіанта	m	n	№ варіанта	m	n
1	10	48	16	17,5	78
2	10,5	50	17	18	80
3	11	52	18	18,5	82
4	11,5	54	19	19	84
5	12	56	20	19,5	86
6	12,5	58	21	20	88
7	13	60	22	20,5	90
8	13,5	62	23	21	92
9	14	64	24	21,5	94
10	14,5	66	25	22	96
11	15	68	26	22,5	98
12	15,5	70	27	23	100
13	16	72	28	23,5	102
14	16,5	74	29	24	104
15	17	76	30	24,5	106

18. Розробити програму-функцію визначення площини поверхні нагрівальних елементів водогрійного котла відповідно до індивідуального варіанта (табл. 3.14).

Інструкція до виконання.

Поверхню нагрівальних елементів визначають за формулою:

$$F_p = \frac{Q_p}{K \cdot \left(t_{пр} - t_в\right)}, м^2;$$

де Q_p – розрахункова тепловіддача нагрівальних приладів у приміщенні, *Вт*;

К=0,8 – коефіцієнт теплопередачі приладу, *Вт/(м²·К)*;

$t_в$ – температура в приміщенні, ^oC;

$t_{пр}$ – середня температура теплоносія в приладі:

$$t_{пр} = \frac{\left(t_н - t_o\right)}{2}, {}^oC;$$

де $t_н$ температура води, яка надходить до приладу, ^oC, розрахунок поверхні нагрівальних приладів виконується за умови, що $t_н > 70$, в іншому випадку розрахунок призупиняється;

t_o =70^o *C*, температура води на виході з приладу.

Таблиця 3.14

Значення величин для розрахунку за варіантом

№ варіанта	Q_p, **м²**	$t_в$, **°C**	**№ варіанта**	Q_p, **м²**	$t_в$, **°C**
1	22500	10	16	25500	17,5
2	22700	10,5	17	25700	18
3	22900	11	18	25900	18,5
4	23100	11,5	19	26100	19
5	23300	12	20	26300	19,5
6	23500	12,5	21	26500	20
7	23700	13	22	26700	20,5
8	23900	13,5	23	26900	21
9	24100	14	24	27100	21,5
10	24300	14,5	25	27300	22

№ варіанта	$Q_р$, $м^2$	$t_в$, ^oC	№ варіанта	$Q_р$, $м^2$	$t_в$, ^oC
11	24500	15	26	27500	22,5
12	24700	15,5	27	27700	23
13	24900	16	28	27900	23,5
14	25100	16,5	29	28100	24
15	25300	17	30	28300	24,5

19. Написати програму-функцію розв'язання рівняння *y(x)=(ax-b)/x*, за умови *x≠0*.

20. Розробити програму-функцію з іменем kol_vo для визначення кількості витяжних шахт, яка залежить від програми-функції plo площі витяжних шахт, відповідно до індивідуального варіанта (табл. 3.15).

Інструкція до виконання.

Розробити програму-функцію з іменем plo, площу витяжних шахт та їх кількість у приміщенні визначають за формулою:

$$F = \frac{L}{3600 \cdot \omega_ш}, м^2;$$

де $L=30000\ м^3$ – кількість припливного повітря;

$\omega_ш$ – швидкість руху у витяжній шахті:

$$\omega_ш = 2{,}2\sqrt{\frac{h \cdot (t_в - t_з)}{273}}, м/с.$$

де h – висота витяжної шахти, *м*,

$t_в=18\ ^oC$ та $t_з=5\ ^oC$ – температура повітря в середині приміщення та зовні.

Розробити програму-функцію з іменем kol_vo визначення кількості витяжних шахт за формулою:

$$n_ш = \frac{F}{f}, шт,$$

де f – переріз однієї шахти, $мм^2$.

Таблиця 3.15

Значення величин для розрахунку за варіантом

№ варіанта	*h,м*	*f, мм²*	№ варіанта	*h,м*	*f, мм²*
1	2	400	16	8	700
2	3	500	17	9	400
3	4	600	18	10	500
4	5	700	19	2	600
5	6	400	20	3	700
6	7	500	21	4	400
7	8	600	22	5	500
8	9	700	23	6	600
9	10	400	24	7	700
10	2	500	25	8	400
11	3	600	26	9	500
12	4	700	27	10	600
13	5	400	28	2	700
14	6	500	29	3	400
15	7	600	30	4	500

21. Розробити програму-функцію з іменем plo визначення необхідної витрати повітря в приміщенні за кратністю вентиляції.

Інструкція до виконання.

Необхідну подачу повітря в приміщення визначимо за кратністю вентиляції:

$$L = m \cdot V \text{ , } м^3;$$

де V – об'єм приміщення за зовнішнім обміром, $м^3$ визначається в одному документі та записується на диск за допомогою оператора **File Output**;

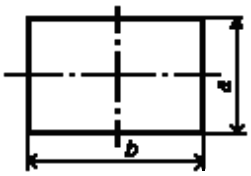
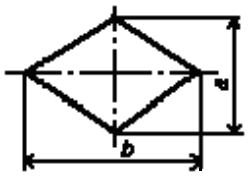
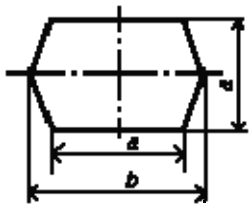
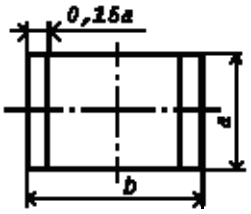
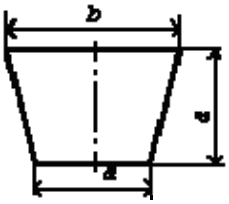
$m=10...12$ – кратність повітрообміну в пташнику, $год^{-1}$.

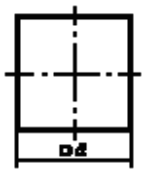
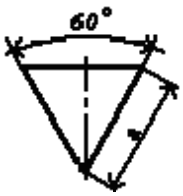
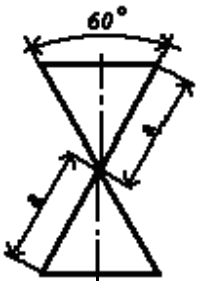
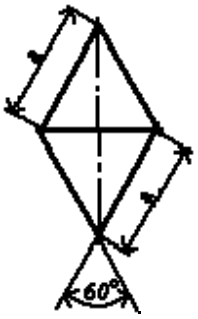
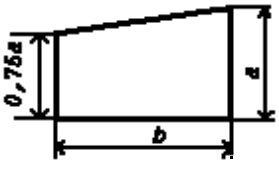
За допомогою оператора **File Input** викликаємо V в іншому документі та визначаємо необхідні витрати повітря за кратністю вентиляції.

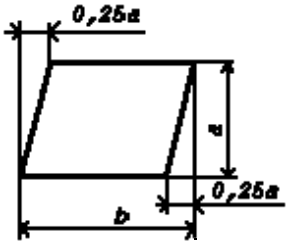
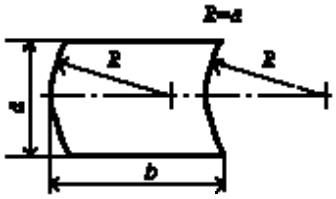
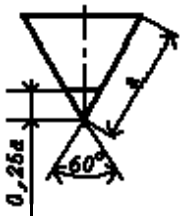
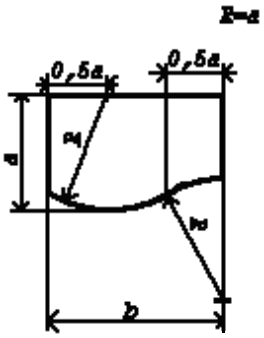
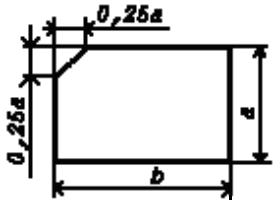
ДОДАТКИ

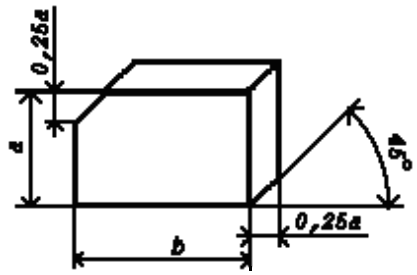
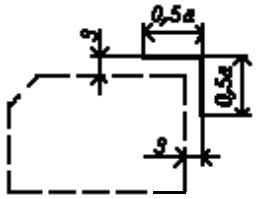
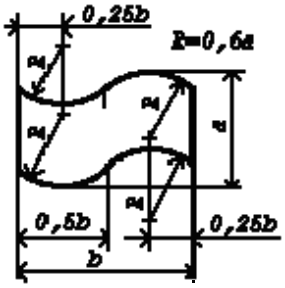
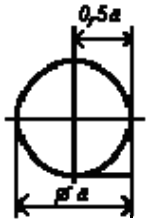
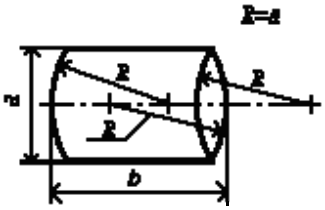
Додаток 1

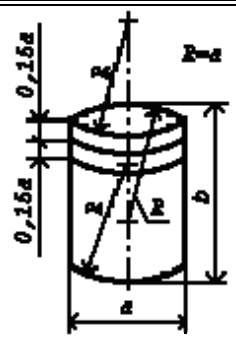
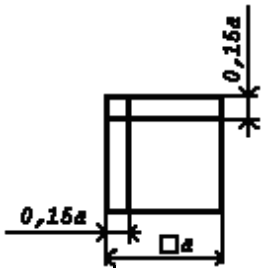
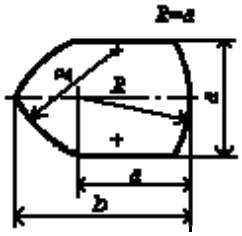
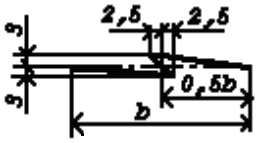
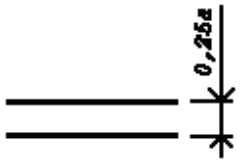
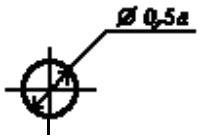
Перелік, найменування, позначення і розміри обов'язкових символів і функцій згідно з ГОСТ 19.701-90 (ИСО 5807-85) «Схемы алгоритмов, программ, данных и систем».

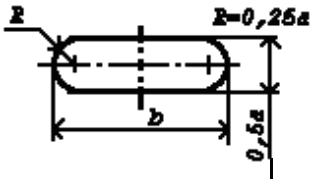
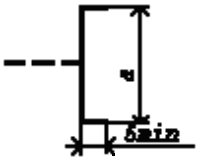
Найменування	Позначення і розміри в мм	Функція
1	2	3
1. Процес		Виконання операцій або групи операцій, в результаті яких змінюється значення, форма уявлення або розташування даних
2. Рішення		Вибір напряму виконання алгоритму або програми залежно від деяких змінних умов
3. Модифікація		Виконання операцій, що міняють команди або групу команд, що змінюють програму
4. Зумовлений процес		Використання раніше створених і окремо описаних алгоритмів або програм
5. Ручна операція		Автономний процес, що виконується вручну або за допомогою засобів, що діють неавтоматично

1	2	3
6. Допоміжна операція		Автономний процес, що виконується пристроєм, не керованим безпосередньо процесором
7. Злиття		Об'єднання двох або більше множин в єдину множину
8. Виділення		Видалення однієї або декількох множин з єдиної множини
9. Групування		Об'єднання двох або більше множин із виділенням декількох інших множин
10. Сортування		Упорядкування множини за даними ознаками
11. Ручне введення		Введення даних вручну за допомогою неавтономних пристроїв із клавіатурою, набором перемикачів, кнопок

1	2	3
12. Введення-виведення		Перетворення даних у форму, придатну для обробки (введення) або відображення результатів обробки (вивід)
13. Неавтономна пам'ять		Введення-виведення даних у разі використання пристрою, що запам'ятовує, керованого безпосередньо процесором
14. Автономна пам'ять		Введення-виведення даних у разі використання пристрою, що запам'ятовує, не керованого безпосередньо процесором
15. Документ		Введення-виведення даних, носієм яких є папір
16. Перфокарта		Введення-виведення даних, носієм яких є перфокарта

1	2	3
17. Колода перфокарт		Відображення набору перфокарт
18. Файл		Уявлення організованих на основі загальних ознак даних, що характеризують у сукупності деякий об'єкт обробки даних. Символ використовується у поєднанні з символами конкретних носіїв даних, що виконують функції вводу-виводу
19. Перфо-стрічка		Введення-виведення даних, носієм є перфо-стрічка
20. Магнітна стрічка		Введення-виведення даних, носієм яких є магнітна стрічка
21. Магнітний барабан		Введення-виведення даних, носієм яких є магнітний барабан

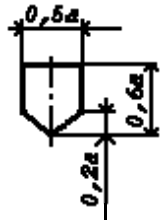
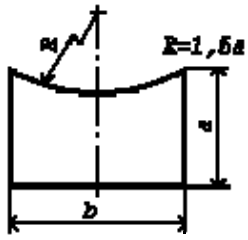
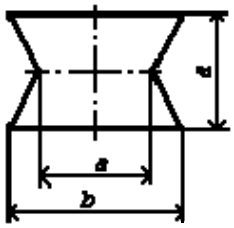
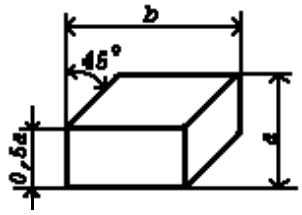
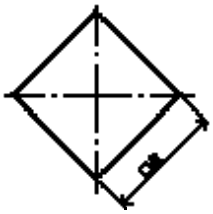
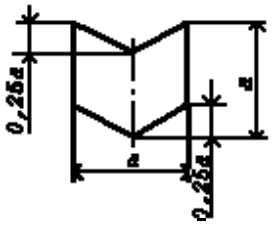
1	2	3
22. Магнітний диск		Введення-виведення даних, носієм яких є магнітний диск
23. Оперативна пам'ять		Введення-виведення даних, носієм яких є магнітний сердечник
24. Дисплей		Введення-виведення даних, якщо безносередньо підключений до процесу пристрій відтворює дані і дозволяє операторові ЕОМ вносити зміни в процесі їх обробки
25. Канал зв'язку		Передача даних каналами зв'язку
26. Лінія потоку		Вказівка послідовності між символами
27. Паралельні дії		Початок або закінчення двох і більше операцій, що одночасно виконуються
28. З'єднувач		Вказівка зв'язку між перерваними лініями потоку, зв'язувальними символами

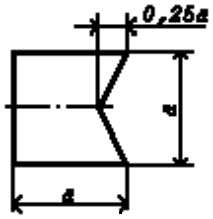
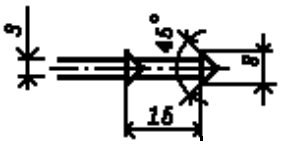
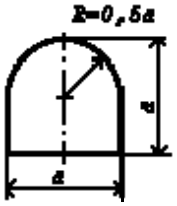
1	2	3
29. Початок – кінець		Початок, кінець, переривання процесу обробки даних або виконання програми
30. Коментар		Зв'язок між елементом схеми і поясненням

Співвідношення геометричних елементів-символів. Розмір *a* має вибиратися з ряду 10, 15, 20 мм. Допускається збільшення розміру *a* на число, кратне 5. Розмір *b* рівний 1,5*a*.

У разі ручного виконання схем алгоритмів і програм для обов'язкових символів 1-5, 11, 12, 16, 29 і символів, що рекомендуються, 3 і 4 допускається встановлювати *b* рівним 2*a*. Обов'язкові символи 7-10, 14 і символ, що рекомендується, 8 допускається у вигляді рівнобедреного прямокутного трикутника з катетом *a*.

Перелік, найменування, позначення і розміри символів, що рекомендуються, і функції згідно з ГОСТ 19.701-90 (ИСО 5807-85) «Схемы алгоритмов, программ, данных и систем», що відображаються ними, в алгоритмі і програмі обробки даних мають відповідати таблиці, наведеній нижче.

Найменування	Позначення і розміри в мм	Функція
1	2	3
1. Між сторінкове з'єднання		Вказівка зв'язку між роз'єднаними частинами схем алгоритмів і програм, розміщених на різних сторінках
2. Магнітна карта		Введення-виведення даних, носієм яких є магнітна карта
3. Ручний документ		Формування документа в результаті виконання ручних операцій
4. Архів		Зберігання комплекту впорядкованих носіїв даних з метою повторного застосування
5. Автономна обробка		Перетворення початкових даних у результаті виконання автономних операцій
6. Розшифровка		Зчитування з носія даних, перекодування і друк на тому ж або іншому носієві даних в результаті виконання автономної операції

1	2	3
7. Кодування		Нанесення кодованої інформації на носій у результаті виконання автономної операції
8. Копіювання		Утворення копії носія в результаті виконання автономної операції
9. Транспортування носіїв		Переміщення носіїв даних за допомогою транспортних засобів або кур'єром
10. Матеріальний потік		Вказівка послідовності операцій у технологічному процесі виготовлення предметів праці, напрямок їх переміщення
11. Джерело (приймач) даних		Відправник або одержувач даних

Додаток 2

Довідник з пакету MathCAD

Допоміжна інформація про деякі об'єкти й конструкції MathCAD, що використовуються під час програмування в цьому пакеті.

Складальні панелі MathCAD

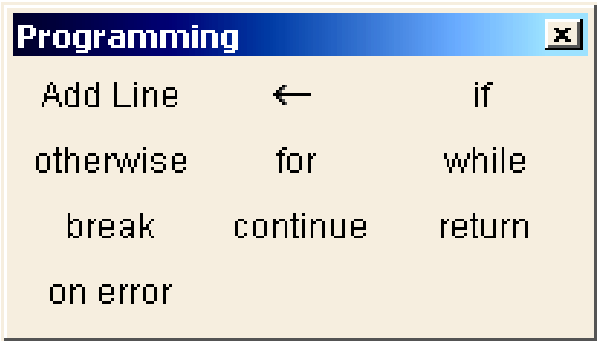
Для введення в текст документа MathCAD заготівок-шаблонів математичних знаків і конструкцій програмування (знаків арифметичних операцій, матриць, знаків інтегралів, похідних і т.д.) використовуються так звані складальні панелі. Призначення складальних панелей MathCAD пояснено в таблиці нижче.

Значок кнопки	Призначення складальної панелі
	Введення знаків арифметичних операцій, цифр
	Введення знаків відносин, використаних пыд час запису умов
	Побудова різних графіків
	Введення матричних операторів
	Введення операторів підсумовування, інтегрування та диференціювання
	Введення конструкцій програмування (панель **Програмування**)
	Введення операторів символьної математики
	Введення букв грецького алфавіту

Для виклику на екран потрібної складальної панелі досить кликнути лівою кнопкою миші на значку, а потім перетягнути і розмістити складальну панель, що розкрилася, у зручному для роботи місці програмного вікна MathCAD.

Панель Programming

Для введення конструкцій програмування необхідна складальна панель **Програмування,** в яку входять такі конструкції:



Складальна панель **Programming**

Для вставки потрібної конструкції в текст створюваної програми, досить клацнути мишею на відповідному значку складальної панелі **Programming.**

Клавіші для створення операторів

Наведені комбінації клавіш використовують для введення операторів. За єдиним виключенням (клавіша лапок – ") результат натискання цієї клавіші однаковий у математичній області й у порожньому місці.

Оператори, що знаходяться чисельно:

! Факторіал.

" У математичній області створює оператор комплексного сполучення.

" У порожньому місці створює текстову область.

Добуток за дискретним аргументом.

$ Підсумовування за дискретним аргументом.

& Інтеграл.

' Пари дужок.

(Ліва дужка.

) Права дужка.

* Множення.

+ Додавання.

, Розділяє аргументи функції.

, Розділяє вираз, що відкладається на одній осі графіка.

, Передує другому числу у разі завдання діапазону.

; Передує останньому числу у разі завдання діапазону.

- Заперечення або вирахування.

/ Розподіл.

< Менше.

> Більше.

? Похідна першого порядку.

[Нижній індекс.

\ Квадратний корінь.

^ Показник ступеня.

| Абсолютне значення.

Ctrl+1 Транспонування.

Ctrl+3 Не дорівнює.

Ctrl+4 Сума елементів вектора.

Ctrl+9 Менше або дорівнює.

Ctrl+0 Більше або дорівнює.

Ctrl+8 Векторний добуток.

Ctrl+- Оператор векторизації.

Ctrl+= Дорівнює.

Ctrl+6 Верхній індекс.

Ctrl+Shift+4 Підсумовування по індексу.

Ctrl+Shift+3 Добуток за індексом.

Ctrl+Shift+? Похідна *n*-го порядку.

Ctrl+\ Корінь n- го ступеня.

Ctrl+Enter Додавання з переносом.

Робота з вікнами та документами

Наступні комбінації клавіш використовуються для маніпуляції вікнами та робочими документами як цілим.

Alt+Esc Перейти до іншого завдання Windows.

Alt+Tab Перейти до іншого переліку завдань Windows

Ctrl+F4 Закрити робочий документ.

Ctrl+F6 Активізувати наступне вікно.

Ctrl+O Роздрукувати робочий документ.

Alt+F4 Вийти з MathCAD.

Ctrl+Q Вийти з MathCAD.

Ctrl+R Обновити екран.

F1 Відкрити вікно Довідки.

F5 Відкрити робочий документ.

F6 Зберегти робочий документ.

F7 Створити робочий документ.

F9 Перерахувати результати на екрані.

Shift+F1 Включити контекстно-чутливу Довідку.

Редагування

Ctrl+F9 Вставляє чистий рядок.

Ctrl+F10 Видаляє чистий рядок.

Ctrl+F5 Викликає діалогове вікно для пошуку фрагмента рядка.

Shift+F5 Викликає діалогове вікно для пошуку й заміни фрагмента рядка.

Alt+BkSp Скасовує останню дію редагування документа.

Ctrl+C Копіює виділення в буфер обміну.

Ctrl+V Вставляє в документ уміст буфера обміну.

Ctrl+X Вирізає виділення й поміщає його в буфер обміну.

Ctrl+U Викликає діалогове вікно для вставки вбудованих одиниць виміру.

Ins Перемикання режимів.

Системні змінні й константи MathCAD та їх значення

Змінні та константи	**Значення змінної**
$\pi = 3.14159$	Число π. Щоб надрукувати натисніть [Ctrl-P]
$e = 2.71828$	Основа натурального логарифма
∞	Нескінченність (10^{307}). Щоб надрукувати натисніть [Ctrl-Z]
%	Відсоток. Використайте його у вираженнях, подібних 10% або як масштабований множник
i	Уявна одиниця
j	Уявна одиниця
$TOL = 10^{-3}$	Допустима погрішність за різних алгоритмів апроксимації (інтегрування, рішення рівнянь). Змінити значення змінної TOL можна за допомогою команди **Математика⇒Параметри**
$CTOL = 10^{-3}$	Встановлює точність обмежень у вирішальному блоці, щоб рішення було допустимим
ORIGIN = 0	Визначає індекс першого елемента векторів і матриць
FRAME = 0	Використовується як лічильник під час створення анімацій
PRNPRECISION = 4	Кількість значущих цифр

Змінні та константи	Значення змінної
PRNCOLWIDTH = 8	Кількість позицій для числа
CWD	Поточний робочий каталог у формі рядка

Вбудовані оператори MathCAD та їх значення

Оператор	Клавіші	Призначення оператора
$X := Y$	X : Y	Локальне присвоювання X значення Y
$X \equiv Y$	$X \sim Y$	Глобальне присвоювання X значення Y
$X =$	$X =$	Вивід значення X
$X + Y$	$X + Y$	Додавання X з Y
$X + Y$	X [Ctrl][↵] Y	Те ж, що й додавання. Перенос суто косметичний
$X - Y$	X - Y	Вирахування з X значення Y
$X \cdot Y$	X * Y	Множення X на Y
$\frac{X}{z}$	X / z	Ділення X на z
z^w	z ^ w	Піднесення z до ступеня w
$\sqrt{z}$	z \	Обчислення квадратного кореня з z
$\sqrt[n]{z}$	n [Ctrl]\ z	Обчислення кореня n-го ступеня з z
n !	n !	Обчислення факторіала
B_n	B [n	Введення нижнього індексу n
$A_{n,m}$	A [n , m	Введення подвійного індексу
$A^{<n>}$	A [Ctrl]6 n	Введення верхнього індексу
$\sum_{i=m}^{n} X$	[Ctrl][Shift]4	Сумування X , на інтервалі $i = m, m + 1, \ldots n$
$\sum_{i} X$	$	Сумування X за дискретним аргументом i

Оператор	Клавіші	Призначення оператора
$\prod_{i=m}^{n} X$	[Ctrl][Shift]3	Множення *X* за *i = m, m + 1, . . . n*
$\prod_{i} X$	#	Множення *X* за дискретним аргументом *i*
$\int_a^b f(t)dt$	&	Обчислення певного інтеграла *f(t)* на інтервалі [a, b]
$\frac{d}{dt} f(t)$	?	Обчислення похідної *f(t)* за *t*
$\frac{d^n}{dt^n} f(t)$	[Ctrl]?	Обчислення похідної *n*-го порядку функції *f(t)* за *t*
(▪)	‘	Введення пари круглих дужок із шаблоном
$x > y$	$x > y$	Більше чим
$x < y$	$x < y$	Менше ніж
$x \geq y$	*x* [Ctrl]0 *y*	Більше або дорівнює
$x \leq y$	*x* [Ctrl]9 *y*	Менше або дорівнює
$z = w$	*z* [Ctrl]= *w*	Булева рівність повертає 1, якщо операнди рівні, інакше 0
$z \neq w$	*z* [Ctrl]3 *w*	Не дорівнює
$\|z\|$	| *z*	Обчислення комплексного модуля *z*

Вбудовані функції

Тригонометричні функції

sin(z) – синус

csc(z) – косеканс

cos(z) – косинус

sec(z) – секанс

tan(z) – тангенс

cot(z) – котангенс

Зворотні тригонометричні функції

asin(z) – зворотній тригонометричний синус

acos(z) – зворотній тригонометричний косинус

atan(z) – зворотній тригонометричний тангенс

Гіперболічні функції

sinh(z) – гіперболічний синус

tanh(z) – гіперболічний тангенс

csch(z) – гіперболічний косеканс

cosh(z) – гіперболічний косинус

sech(z) – гіперболічний секанс

coth(z) – гіперболічний котангенс

Показові та логарифмічні функції

exp(z) – експонентна функція (або e^z)

ln(z) – натуральний логарифм (за основою e)

log(z) – десятковий логарифм (за основою 10)

Функції роботи із частиною числа (округлення й ін.)

Re(z) – виділення дійсної частини z

Im(z) – виділення уявної частини z

arg(z) – обчислення аргументу (фази)

floor(x) – найбільше ціле, менше або рівне x

ceil(x) – найменше ціле, більше або рівне x

mod(x,y) – залишок від ділення x/y зі знаком x

angle(x,y) – позитивний кут із віссю x для точки з координатами (x,y)

Повідомлення про помилки

Під час виконання обчислень можливі помилки. Система виводить повідомлення про помилки, зафарбовуючи помилкові імена ідентифікаторів у яскраво-червоний колір. У колишніх версіях MathCAD повідомлення про помилки з'являлися в червоних прямокутниках із лінією, що відходить від місця помилки до прямокутника з повідомленням про неї. Однак це захаращувало документ, і від такого способу в новій версії відійшли. У новій версії MathCAD докладне повідомлення про помилку можна одержати також, установивши на вираз із помилкою курсор миші й нажавши її ліву клавішу.

Нижче представлено список основних повідомлень про помилки:

array size mismatch – невідповідність розміру масиву;

cannot be defined – не може бути визначено;

cannot take subscript – не містить верхніх (нижніх) індексів;

definition stack overflow – переповнення стека визначень;

did not find solution – рішення не знайдене;

dimension to non real power – розмірність масиву – не ціле число;

domain error – помилка області визначення;

duplicate – дублювання;

equation too large – занадто велике вираження;

error in constant – помилка в константі;

error in list – помилка в списку;

error in solve block – помилка в блоці;

file error – помилка у файлі;

file not found – файл не знайдений;

illegal array operation – неправильна операція з масивом;

illegal context – неправильний контекст;

illegal factor – неправильний множник;

illegal function name – неправильне ім'я функції;

illegal ORIGI – неправильне вживання ORIGI;

illegal range – неправильний діапазон;

illegal tolerance – некоректна точність апроксимації;

incompatible units – несумісні одиниці;

index out of bounds – індекс поза границями;

interrupted – рішення перерване;

invalid order – неправильний порядок;

list too long – довгий вхідний список;

misplased comma – недоречна кома;

missing operand – пропущений операнд;

missing operator – пропущений знак операції;

must be 3-vector – має бути тривимірний вектор;

must be array – має бути масив;

must be dimensionless – має бути безрозмірним;

must be increasing – має бути зростаючий;

must be integer – має бути цілим;

must be nonzero – має бути ненульовим;

must be positive – має бути позитивним;

must be range – має бути діапазон;

must be real – має бути дійсним;

must be scalar – має бути скаляром;

must be vector – має бути вектором;

nested solve block – наступний блок рішення;

no matching Given – немає відповідного Given;

no scalar value – не скалярна величина;

not a name – не є ім'ям;

not converging – не конвертується;

only one array allowed – допустимий тільки один масив;

overflow – переповнення;

significance lost – загублені значущі цифри;

singularity - розподіл на нуль;

stack overflow – переповнення стека;

subscript too large – занадто великий нижній індекс;

too few arguments – занадто мало аргументів;

too few constraints – занадто мало обмежень;

too few elements – занадто мало елементів;

too few subscripts – мало нижніх індексів;

too large to display – занадто велике, щоб бути відображеним;

too many arguments – занадто багато аргументів;

too many constraints – занадто багато обмежень;

too many points - занадто багато крапок;

too many subscripts – занадто багато індексів;

undefindet – не визначено;

unmatched parenthesis – дисбаланс дужок;

wrong size vector – неправильний розмір вектора.

Повідомлення про помилки короткі, але їх аналіз не викликає особливих труднощів. Зазначимо, що наведений список містить далеко не всі помилки, а лише найпоширеніші.

СПИСОК ЛІТЕРАТУРИ

1. Акишин Б.А. Прикладные математические пакеты / Б.А. Акишин, Н.Х. Эркенов. – РадиоСофт, 2009. – 132 с.
2. Баженов В.А. Інформатика. Комп'ютерна техніка. Комп'ютерні технології / Баженов В.А., Венгерський П.С., Горлач В.М. – К. : Каравела, 2004. – 464 с.
3. Глушаков С.В. Персональный комп'ютер / С.В. Глушаков, А.С. Сурядный. – [5-е изд., доп. и перераб.] – Харьков : Фолио, 2003. – 503 с.
4. Дейтел Г. Введение в операционные системы. В 2 т. / Г. Дейтел. – М. : Мир,1987. – 420 с.
5. Дьяконов В.П. MathCAD 11/12/13 в математике. Справочное пособие / В.П. Дьяконов – Горяч. Линия-Телеком, 2007. – 958 с.
6. Дьяконов В.П. Справочник по алгоритмам и программам на языке бейсик для персональных ЭВМ : справ очник / В.П. Дьяконов – М. : Наука, 1987. – 240 с. : ил.
7. Дьяконов В.П. MathCAD 8 PRO в математике, физике и Internet / В.П. Дьяконов, И.В. Абраменкова. – М. : Нолидж, 2000. - 512 с.: ил.
8. Калабеков Б.А. Применение ЭВМ в инженерных расчетах в технике святи / Б.А. Калабеков. – М. : Радио и связь, 1981. – 224 с., ил.
9. Кирьянов Д.Б. Самоучитель MathCAD 2001 / Д.Б. Кирьянов. – Спб. : BHV-Петербург, 2001. – 544 с.
10. Ковалюк Т.В. Основи програмування / Т.В. Ковалюк. – К. : Видав. група BHV, 2005.– 384 с. : іл.
11. Кравчук С.О., Шонін В.О. Основи комп'ютерної техніки: Компоненти, системи, мережі : навч. посіб. для вузів / С.О. Кравчук, В.О. Шонін – К. : ІВЦ «Вид-во «Політехніка» ; Каравела, 2005. – 344 с. : іл.

12. Кудрявцев Е.М. MathCAD 11: Полное руководство по русской версии [Электронный ресурс] / Е.М. Кудрявцев.
13. Кудрявцев Е.М. MathCAD 2000 Pro / Е.М. Кудрявцев – М. : ДМК Пресс, 2001. – 576 с. : ил.
14. Макаров Е. Инженерные расчеты в MathCAD. Учебный курс 2003 / Е. Макаров. – 448 с.
15. Макарова М.В. Інформатика та комп'ютерна техніка : навч. посіб. для студ. вищ. навч. закл. / Макарова М.В., Карнаухова Г.В., Запара С.В. – Суми : Університетська книга, 2003. – 642 с
16. Очков В.Ф. MathCAD 14 для студентов и инженеров / В.Ф. Очаков – Изд. группа BHV 2009. – 512 с.
17. Плис А.И. Н.А. MathCAD. Математический практикум для инженеров и экономистов : учеб. пособ. / А.И. Плис, Н.А.Сливина. – [2-е изд., перераб. и доп.]. – М. : Финансы и статистика, 2003. – 656 с.
18. Пройдаков Е.М. Англо-український тлумачний словник з обчислювальної техніки, Інтернету і програмування / Е.М. Пройдаков, Л.А. Теплицький. – К. : Видав. дім «СофтПрес», 2005. – 552 с.
19. Пушкар О.І. Інформатика. Комп'ютерна техніка. Комп'ютерні технології / О.І. Пушкар. – К. : Академія, 2002. – 704 с.
20. Таненбаум Э. Архитектура комп’ютера / Є. Таненбаум. – [4-е изд.]. – СПб. : Питер, 2005. – 699 с. : ил.
21. Фельдман Л. Чисельні методи в інформатиці / Фельдман Л., Петренко А., Дмитрієва О. – К.: Видав. група BHV, 2006. – 480 с. : іл.
22. Херхаргер М., Партолль Х. MathCAD 2000 : полное руководство / М. Херхаргер, Х. Партолль ; пер. с нем. – К. : Узд. группа BHV, 2000. – 416 с.
23. Шеховцов В.А. Операційні системи / В.А. Шеховцев. – К. : Видав. група BHV, 2008. – 576 с. : іл.

АЛФАВІТНИЙ ПОКАЖЧИК

Н

О

П

Ф

Ч

Ш

Навчальне видання

Лисенко Віталій Пилипович
Болбот Ігор Михайлович

КОМП'ЮТЕРИ ТА КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

Частина І

«Програмування в математичному пакеті MathCAD»

Навчальний посібник

Редагування	Л.М. Талюта
	Н.В. Крошко
Макетування	І.О. Сєрова

Підписано до друку 25.05. 2010. Формат 60х84/16.
Папір офсет. №1. Гарнітура Times New Roman. Друк офс.
Наклад 1000 примірників, Зам. №53

Редакційно-видавничий відділ
Науково-методичного центру аграрної освіти
Київ-151, вул. Смілянська,11
тел. 249-94-04

Фірма "Інтас"