

ПЕРЕДМОВА

На даному етапі розвитку суспільства майже не залишилося галузей, де не використовуються інформаційні технології. Тому інформатика, як фундаментальна наука, що забезпечує розвиток цих технологій, включається до навчальних програм усіх спеціальностей. Особливо це стосується технічних напрямків. Для майбутніх спеціалістів дуже важливе формування цілісного сприйняття і розуміння інформаційних процесів, що протікають в суспільстві.

Слід зауважити, що підготовка матеріалу для посібника викликала певні труднощі у зв'язку з дуже швидким і багатогранним розвитком інформаційних технологій. Технології, що вчора здавалися новітніми, вже увійшли в життя і широко використовуються. А деякі технології, що були перспективними, вже закінчують своє існування і замінюються більш ефективними або більш простими. Причому, завдяки проникненню інформаційних технологій в інші галузі — матеріалознавство, хімію, фізику, математику, електроніку, - швидкість змін ще більше зростає. Наприклад, від появи перших ЕОМ до персональних комп'ютерів пройшло біля 40 років, а розповсюдження ПК відбулося лише за 15-20 років. Таким чином, для забезпечення актуальності викладення матеріалу, автори посібника зробили наголос на теоретичних основах інформатики, а також основних алгоритмах, що можуть знадобитися студентам для засвоєння нових технологій.

Посібник охоплює весь матеріал відповідної дисципліни. В першій частині (розділи 1-5) викладається загальний курс - теоретичні основи інформатики, архітектура і можливості сучасних апаратних і програмних засобів, операційних систем і прикладного програмного забезпечення (ПЗ), описуються основи автоматизованої обробки даних за допомогою офісного ПЗ. Слід зауважити, що в якості об'єктів для детального розгляду вибиралося переважно ПЗ, що максимально підтримує відкриті стандарти і розповсюджується безкоштовно.

В другій частині (розділи 6-8) приділяється увага практичним питанням застосування комп'ютерних технологій при вирішенні загальнотехнічних задач, детально розглядаються базові алгоритми типових обчислювальних процесів, надається інформація про застосування алгоритмічної мови високого рівня PASCAL. Кожна тема супроводжується відповідними прикладами вирішення практичних завдань.

При підготовці посібника широко використовувалися матеріали всесвітньої вільної енциклопедії **wikipedia.org**, навчальні матеріали розробників прикладних програмних засобів, матеріали конференцій і форумів відповідної тематики.

Автори щиро дякують всім, хто брав участь в обговоренні і допомагав створювати цей посібник.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ІНФОРМАТИКУ

Інформатика - це наука, що вивчає закони, методи і засоби накопичення, передачі, обробки (перетворення) інформації.

Назва “Інформатика” затвердилася наприкінці 60-х років 20 століття і виникла із двох слів — **інформація** і **автоматика**, тобто автоматична обробка інформації. Інформатика швидко розвивається в основному завдяки інтенсивному розвитку електроніки і, у першу чергу, комп'ютерної техніки. Сучасний стан інформатики дозволяє забезпечити в повній мірі перехід від системи обробки даних - автоматизації допоміжних (рутинних) операцій розумової праці до системи обробки знань - використанню комп'ютерної техніки в творчих (інтелектуальних) процесах пізнання і управління.

1.1 СИСТЕМИ ЧИСЛЕННЯ

Система числення - це сукупність прийомів та правил найменування й позначення чисел.

Системи числення поділяються на позиційні і непозиційні. Система числення, в якій значення кожного знака (символу) визначається місцем в запису числа, називається **позиційною**. В **непозиційній** системі кількісне значення довільного знака не залежить від його місця в запису числа.

У непозиційній системі кожний знак у запису незалежно від місця означає одне й те саме число. У цій системі числення незручно й складно виконувати арифметичні операції. Добре відомим прикладом непозиційної системи числення є римська система, в якій роль цифр відіграють наступні букви алфавіту:

I - один,	V - п'ять,	
X - десять,	C - сто,	
Z - п'ятдесят,	D - п'ятсот,	M - тисяча.

Наприклад, $324 = \text{CCCXXIV}$.

У позиційній системі будь-яке дійсне число має вигляд

$$A = C, D \quad .$$

Тут $C = c_{n-1} \dots c_1 c_0$ - ціла частина, $D = d_{-1} d_{-2} \dots d_{-m}$ - дробова частина числа A . Позитивні нижні індекси $n-1, \dots, 1, 0$ визначають положення цифри в запису цілої частини C числа A (n розрядів), негативні $-1, -2, \dots, -m$ - положення цифри дробової частини D числа A (m розрядів).

У позиційній системі числення кількість p різних знаків (цифр), використовуваних для представлення числа називається основою системи числення.

В таблиці 1.1 наведені деякі позиційні системи числення, що використовуються в комп'ютерній техніці, та знаки (цифри), з яких утворюються в них числа.

Таблиця 1.1 Цифри в позиційних системах числення

Система числення	Основа p	Знаки (цифри)
Двійкова	2	0 1
Вісімкова	8	0 1 2 3 4 5 6 7
Десяткова	10	0 1 2 3 4 5 6 7 8 9
Шістнадцяткова	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

Загальноприйнятою є позиційна десяткова система числення.

1.1.1 Представлення чисел в різних системах числення

У загальному випадку запис будь-якого дійсного числа в системі числення з основою p представляється у вигляді

$$A_{(p)} = C_{(p)} + D_{(p)} \quad ,$$

$$C_{(p)} = c_{n-1} p^{n-1} + \dots + c_1 p^1 + c_0 p^0 \quad ,$$

$$D_{(p)} = d_{-1} p^{-1} + d_{-2} p^{-2} + \dots + d_{-m} p^{-m} \quad .$$

Приклад 1.1. Представлення чисел в різних системах числення (індекс справа знизу вказує основу системи числення):

$$1101_{(2)} = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0,$$

$$345,1_{(8)} = 3 \cdot 8^2 + 4 \cdot 8^1 + 5 \cdot 8^0 + 1 \cdot 8^{-1},$$

$$23,43_{(10)} = 2 \cdot 10^1 + 3 \cdot 10^0 + 4 \cdot 10^{-1} + 3 \cdot 10^{-2},$$

$$A1F,4_{(16)} = A \cdot 16^2 + 1 \cdot 16^1 + F \cdot 16^0 + 4 \cdot 16^{-1}$$

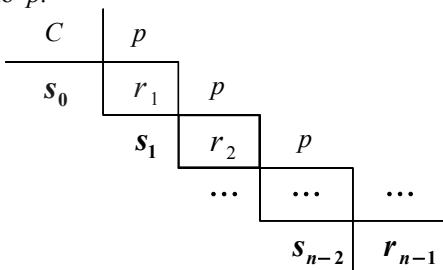
Таким чином, значення кожного знака в числі залежить від позиції, яку займає знак у запису числа. Саме тому такі системи числення називають позиційними.

1.1.2 Переклад чисел з однієї позиційної системи числення в іншу

При перекладі будь-якого числа $A=C, D$ з системи числення з основою q в систему з основою p зазвичай використовують наступний алгоритм перекладання цілої C і дробової D частин числа:

1. Основа нової системи числення p представляється цифрами початкової системи q .

2. Ціла частина C числа ділиться на p , після чого запам'ятовується залишок s_0 від ділення. Отримана при цьому частка r_1 і усі інші послідовно отримувані частки r_i ($i=2, \dots, n-1$) діляться на p , залишки s_i ($i=1, \dots, n-2$), запам'ятовуються. Ця процедура триває до останнього залишку ($s_{n-2}=0$ або $s_{n-2}<p$). Усі набуті значення s_i ($i=0, \dots, n-2$), r_{n-1} являються цифрами $c_i=s_i$ ($i=0, \dots, n-2$), $c_{n-1}=r_{n-1}$ цілої частини $C=c_{n-1}c_{n-2}\dots c_1c_0$ числа A у новій системі числення з основою p .



$$C=c_{n-1}\dots c_1c_0=r_{n-1}s_{n-2}\dots s_0$$

3. Дробова частина D числа A множиться на p , після чого ціла частина добутку g_1 записується цифрами нової системи числення з основою p . Отримана дробова частина h_1 добутку перемножується на p і т.д. Процедура послідовного множення триває до тих пір, поки дробова частина чергового добутку h_m не стане рівною нулю або не буде досягнута потрібна точність представлення числа. Отримані цілі частини добутків, які представлені цифрами нової системи числення з основою p , являються цифрами $d_{-i} = g_i$ ($i = 1, \dots, m$) дробової частини $D = d_{-1}d_{-2}\dots d_{-m}$ числа $A = C, D$.

$0,$	D $\times p$
g_1	h_1 $\times p$
g_2	h_2 $\times p$
\dots	\dots
g_{m-1}	h_{m-1} $\times p$
g_m	h_{-m}

$$D = d_{-1}d_{-2}\dots d_{-m} = g_1g_2 \dots g_m$$

4. Результатом перекладу числа A з системи числення з основою q в систему з основою p є

$$A = C, D = r_{n-1}s_{n-2}\dots s_0, g_1g_2 \dots g_m$$

Приклад 1.2. Перекласти десяткове число $1251_{(10)}$ в шістнадцяткову, вісімкову та двійкову системи числення:

1251	16	
3	78	16
	14 (E)	4

$$1251_{(10)} = 4E3_{(16)}$$

1251	8		
3	156	8	
	4	19	8
		3	2

$$1251_{(10)} = 2343_{(8)}$$

1251 2

1 625 2

1 312 2

0 156 2

0 78 2

0 39 2

1 19 2

1 9 2

1 4 2

0 2 2

0 1

$$1251_{(10)} = 10011100011_{(2)}$$

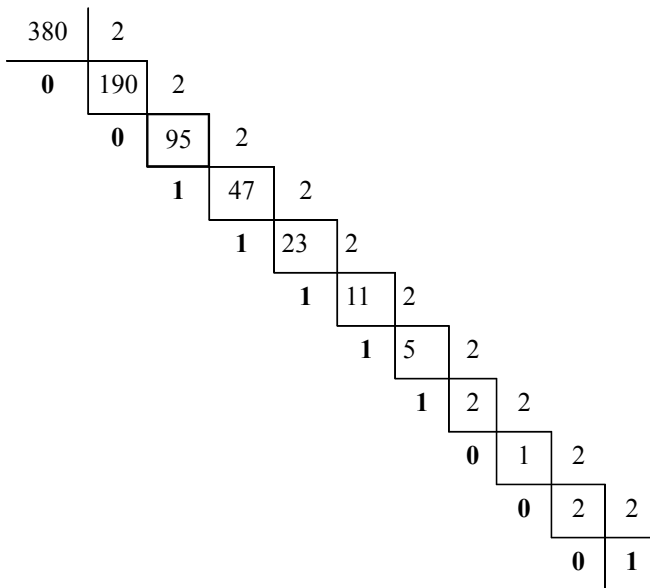
Приклад 1.3. Перекласти десяткове число $0,65534_{(10)}$ у шістнадцяткову.

0,	65534 ×16
10 (A)	48544 ×16
7	76704 ×16
12 (C)	27264 ×16
4	36224 ×16
5	79584 ×16
12 (C)	73344 ×16

$$0,65534(10)=0,A7C45C(16)$$

Приклад 1.4. Перекласти десяткове число $380,1875_{(10)}$ в двійкове число.

1) Переклад цілої частини $C=380$



2) Переклад дробової частини $D=0,1875$

0,	1875×2
0	3750×2
0	7500×2
1	5000×2
1	0000×2

3) Результат перекладу числа $380,1875$ з системи числення з основою 10 в систему з основою 2:

10001111100,0011

1.2 ІНФОРМАЦІЯ

Ключову роль в інформатиці грає інформація. Оскільки строгого і загально призаного визначення інформації до цих пір не існує, то замість визначення зазвичай використовують поняття про інформацію.

1.2.1.Поняття інформації

Будь-які відомості можна вважати інформацією, якщо вони знаходять споживача, тобто того, хто їх збирає, зберігає, переробляє, передає або просто використовує. Відомість - це форма представлення інформації у вигляді мови, тексту, зображення, графіків, таблиць, цифрових даних, електричних, магнітних, світлових сигналів і тому подібне. Інакше кажучи,

інформацією називається сукупність відомостей, що визначають міру наших знань про ті або інші об'єкти, події, явища і факти.

Поняття інформації є одним із найважливіших в сучасній науці і базовим для інформатики. Оскільки інформатика безпосередньо пов'язана з технічними засобами, що накопичують, зберігають, перетворюють і передають дані за допомогою відповідних методів, то для визначення інформації можна використовувати поняття, що більш повно відражає її фізичну суть:

інформація — це продукт (результат) взаємодії даних і адекватних їм методів.

Із наведеного визначення випливає:

- *Динамічний характер інформації.* Інформація не існує незалежно, і не є статичною — вона змінюється і існує тільки в момент поєднання даних і методів для їх обробки. Тобто інформація існує тільки під час протікання інформаційного процесу.
- *Вимогливість до адекватності методів.* Одні і ті ж дані в результаті інформаційного процесу можуть давати різну інформацію, якщо використовуються різні за адекватністю методи. Чим більш коректно підібрані методи, тим більше корисної інформації вдасться отримати із одного і того ж набору даних.
- *Діалектичний характер взаємодії даних і методів.* Дані в цьому сенсі є об'єктивними (як результат реєстрації об'єктивних процесів), а методи — суб'єктивними (як ті, що підготовлені людиною (суб'єктом)).

1.2.2 Властивості інформації

Інформація з'являється під час інформаційного процесу з участю даних і методів обробки. Як і кожний інший об'єкт, інформація має різноманітні властивості, які, завдяки залежності інформації від даних і методів, можуть істотно змінюватись. Для кожного випадку використання інформації можна виділити найбільш важливі властивості і домагатися покращення їх значень. З точки зору інформатики найбільш важливими властивостями є наступні:

- *Адекватність.* Ступінь відповідності реальному положенню діла. Неадекватні або неповні дані можуть призвести до неадекватної інформації. Також використання неадекватних методів для обробки повних і коректних даних призведе до появи невірної інформації.
- *Актуальність.* Ступінь відповідності інформації поточному моменту часу. Тому що інформаційний процес розтягнутий по часу, поява адекватної але не своєчасної інформації призведе до прийняття невірних рішень.

- *Доступність.* Наявність як повного об'єму даних, так і адекватних методів для обробки.
- *Достовірність.* Відношення корисних даних до шуму та поміх.
- *Повнота.* Достатність даних для отримання корисної інформації і прийняття на її основі рішень або отримання нових даних.
- *Об'єктивність.* Міра впливу суб'єктивних методів на оброблювані об'єктивні дані. Чим менше вплив методів, тим більш об'єктивна інформація з'являється на виході інформаційного процесу. Так, фотографія якогось об'єкту є більш об'єктивною, ніж малюнок того ж самого об'єкту, виконаний людиною.

1.2.3 Кількість інформації

Інформація може бути виражена математично і виміряна кількісно. Для визначення кількості інформації в основному використовується ймовірнісний підхід.

Ймовірнісний підхід був вперше застосований Клодом Шеноном — одним із засновників кібернетики. За Шеноном, інформація — величина, зворотня до ентропії, тобто хаосу (невизначеності), кількість інформації знаходиться за формулою

$$m = - \sum_{i=1}^N P_i \cdot \log_2 \left(\frac{1}{P_i} \right) \quad (1.1)$$

де m - середня ентропія повідомлення,

P_i - вірогідність з'явлення i -ої події,

N - кількість різноманітних можливих станів.

У випадку, коли вірогідності випадіння різних випадків рівні, формула (1.1) спрощується і має вигляд

$$m = \log_2 N \quad (1.2)$$

Формула (1.2) називається формулою Хартлі.

Розглянемо систему, що складається з символів 0 і 1. Хай ці символи пов'язані однаковою ймовірністю їх появи $P(0) = P(1) = 0,5$. Тоді кількість інформації на один знак при двійковому кодуванні дорівнює

$$m = \log_2 2 = 1 \quad (1.3)$$

Таким чином, кількість інформації у бітах, що укладена в слові з цих символів, дорівнює кількості символів в ньому.

У комп'ютерних системах два можливі результати кодуються двома цифрами: 0 і 1, які є символами двійкової системи числення. У кожній цифрі двійкового числа міститься 1 біт інформації.

Біт — *одиниця інформації, що отримується при проведенні дослідження, яке складається з отримання одного із рівновірогідних випадків.*

У комп'ютерній техніці біт є найменшою можливою одиницею інформації, але ж найменшою адресованою одиницею інформації є

$$1 \text{ байт} = 8 \text{ біт}$$

Для зручності введені і більші, ніж біт і байт, одиниці кількості інформації. Наглядно залежність одиниць вимірювання інформації представлена в таблиці: 1.2

Таблиця 1.2 Одиниці кількості інформації

Назва	В байтах	В бітах
Байт (Byte)	1	$2^3=8$
Слово (Word)	2	$2^4=16$
Кілобіт (Kbit)	$2^7=128$	$2^{10}=1\,024$
Кілобайт (KByte)	$2^{10}=1\,024$	$2^{13}=8\,192$
Мегабіт (Mbit)	$2^{17}=131\,072$	$2^{20}=1\,048\,576$
Мегабайт (MByte)	$2^{20}=1\,048\,576$	$2^{23}=8\,388\,608$
Гігабіт (Gbit)	$2^{27}=134\,217\,728$	$2^{30}=1\,073\,741\,824$
Гігабайт (GByte)	$2^{30}=1\,073\,741\,824$	$2^{33}=8\,589\,934\,592$
Терабіт (Tbit)	$2^{37}=137\,438\,953\,472$	$2^{40}=1\,099\,511\,627\,776$
Терабайт (TByte)	$2^{40}=1\,099\,511\,627\,776$	$2^{43}=8\,796\,093\,022\,208$
Петабіт (Pbit)	$2^{47}=140\,737\,488\,355\,328$	$2^{50}=1\,125\,899\,906\,842\,624$
Петабайт (PByte)	$2^{50}=1\,125\,899\,906\,842\,624$	$2^{53}=9\,007\,199\,254\,740\,992$

Зауваження:

- Такі одиниці, як кілобіт і мегабіт найчастіше використовуються для позначення швидкості передачі інформації (наприклад, Кбіт/с, Мбіт/с), тому що при передачі даних кількість бітів може не бути кратною 8.
- Зазвичай виробники пристроїв для накопичення інформації для спрощення вважають, що 1 Мбайт дорівнює 1 000 000 байт, а не 1 048 576. Це припущення зменшує реальну ємкість майже на 5%.

1.2.4 Кодування інформації

Для автоматизації роботи з даними, що відносяться до різних типів, дуже важливо уніфікувати їх форму представлення - для цього використовується прийом **кодування**, тобто вираз даних одного типу через дані іншого типу. Природні людські мови - це не що інше, як системи кодування понять для вираження думок за допомогою мови.

1.2.4.1. Двійкове кодування.

Для автоматичної обробки інформації в комп'ютерних системах використовується уніфікована форма представлення даних - система двійкового кодування.

Двійкове кодування - це процес перетворення інформації в комбінації нулів і одиниць.

Одним бітом можуть бути виражені два поняття: 0 або 1 (так чи ні, чорне або біле, істина чи брехня і т. п.). Якщо кількість бітів збільшити до двох, то вже можна виразити чотири різні поняття:

00 01 10 11

За допомогою трьох бітів можна записати вже вісім різних комбінацій:

000 001 010 011 100 101 110 111

Збільшуючи на одиницю кількість розрядів в системі двійкового кодування, ми збільшуємо в два рази кількість значень, що може бути виражено в даній системі, тобто загальна формула має вигляд:

$$N = 2^m \quad (1.4)$$

де m - кількість біт, необхідних для запису усіх можливих станів;

N - кількість різноманітних можливих станів.

Очевидно, що це формула Хартлі (1.2) і відображає вона у цьому вигляді зворотню інформацію - кількість різноманітних станів (комбінацій) N , які можливо отримати для m біт. Деякі значення кількості кодових комбінацій наведено у таблиці 1.3:

Таблиця 1.3 Кількість можливих кодових комбінацій

Назва	Кількість біт (байт) m	Кількість комбінацій N
bit	1	$2^1 = 2$
byte	$2^3 = 8$ (1)	$2^8 = 256$
word	$2^4 = 16$ (2)	$2^{16} = 16384$
Kbit	$2^{10} = 1\,024$ (128)	$2^{2^{10}} = 1.8 \cdot 10^{308}$
KByte	$2^{13} = 8\,192$ (1024)	$2^{2^{13}} = 10^{2466}$
Mbit	$2^{20} = 1\,048\,576$ (131072)	$2^{2^{20}} = 6.7 \cdot 10^{315652}$
MByte	$2^{23} = 8\,388\,608$ (1048576)	$2^{2^{23}} = 4 \cdot 10^{2525222}$

Неважко помітити, що формула (1.3) виводиться з формули Хартлі (1.2) і виконує зворотню операцію пошуку кількості різноманітних станів N , які можливо отримати із m біт.

1.2.4.2. Кодування цілих і дійсних чисел.

Для кодування цілих чисел від 0 до 255 достатньо мати 8 розрядів двійкового коду (8 біт). Шістнадцять біт дозволяють закодувати цілі числа до 65535, а 24 біта - вже більше 16,5 мільйонів різних значень. Для кодування дійсних чисел використовують 80-розрядне кодування. При цьому число попередньо перетворюється в нормалізовану форму:

$$3,1415926 = 0,3141592 \cdot 10^1 \quad 300\,000 = 0,3 \cdot 10^6$$

$$123\,456\,789 = 0,123456789 \cdot 10^9$$

1.2.4.3. Кодування текстової інформації.

Якщо кожному символу алфавіту зіставити певне ціле число (наприклад, порядковий номер), то за допомогою двійкового коду можна кодувати і текс-

тову інформацію. Восьми двійкових розрядів достатньо для кодування 256 різних символів. Цього вистачить, щоб висловити різними комбінаціями восьми бітів всі символи англійської та російської мов, як малі, так і прописні, а також знаки пунктуації, символи основних арифметичних дій і деякі загальноприйняті спеціальні символи.

Наприклад, слово “Інформатика” в двійковій системі буде виглядати як послідовність

```
10110110 11001110 11000110 11001111 11010010 11001101 11000001  
11010100 11001001 11001011 11000001
```

Для англійської мови, яка захопила де-факто нішу міжнародного засобу спілкування, протиріччя вже зняті. Інститут стандартизації США (ANSI - American National Standard Institute) ввів в дію систему кодування ASCII (American Standard Code for Information Interchange - стандартний код інформаційного обміну).

У системі ASCII закріплені дві таблиці кодування - базова і розширена. Базова таблиця закріплює значення кодів від 0 до 127, а розширена відноситься до символів з номерами від 128 до 255. Перші 32 коди базової таблиці, починаючи з нульового, віддані виробникам апаратних засобів (в першу чергу виробникам комп'ютерів і друкуючих пристроїв). У цій області розміщуються так звані керуючі коди, яким не відповідають ніякі символи мов, і, відповідно, ці коди не виводяться ні на екран, ні на пристрої друку, але ними можна керувати тим, як здійснюється вивід інших даних.

Починаючи з коду 32 по код 127 розміщені коди символів англійського алфавіту, розділових знаків, цифр, арифметичних дій і деяких допоміжних символів.

Аналогічні системи кодування текстових даних були розроблені і в інших країнах. Так, наприклад, в СРСР в цій області діяла система кодування КОІ-7 (код обміну інформацією, семизначний). Однак підтримка виробників обладнання та програм вивела американський код ASCII на рівень міжнародних стандартів, і національним системам кодування довелося «відступити» в другу, розширену частину системи кодування, що містить значення кодів з 128 по 255. Відсутність єдиного стандарту в цій галузі призвело до великої кількості одночасно діючих кодувань.

На просторах колишнього СРСР одночасно діє декілька систем кодування. Найбільш вживні - КОІ-8 (український варіант називається КОІ-8U), Windows-1251 (CP1251), UNICODE. Для систем на базі операційних систем MS Windows стандартом є Windows-1251, а для UNIX і Linux систем КОІ-8 і, в останній час UNICODE. Остання система кодування відрізняється кількістю біт на один символ. В цій системі використовується 16 біт на символ замість

8, що дозволяє розміщувати 65536 різних символів в одній таблиці замість 256 в попередніх системах кодування. Це дозволяє тримати в одній таблиці усі вживані символи всіх алфавітів світу. Розповсюдження цієї системи затримувалося лише неспроможністю апаратних та програмних засобів оброблювати вдвічі довші набори даних.

1.2.4.4. Кодування графічної інформації.

Якщо розглянути за допомогою лупи монохромне графічне зображення, надруковане в газеті чи книжці, то можна побачити, що воно складається з найдрібніших точок, що утворюють характерний візерунок, який називається **растром**. Оскільки лінійні координати й індивідуальні властивості кожної точки (яскравість) можна виразити за допомогою цілих чисел, то можна сказати, що растрове зображення дозволяє використовувати двійковий код для представлення графічних даних. Загальноприйнятим на сьогоднішній день вважається подання чорно-білих ілюстрацій у вигляді комбінації точок з 256 градаціями сірого кольору, і, таким чином, для кодування яскравості будь-якої точки звичайно досить восьмириздного двійкового числа.

Для представлення кольорових зображень використовують принцип декомпозиції, тобто кожну точку (піксел) будь-якого кольору можна замінити трьома точками червоного, синього і зеленого кольору відповідної яскравості. А кожний з цих кольорів вже можна замінити на двійковий код. Така схема (або ще кажуть, кольорове пространство) заміни називається RGB (**R**ed, **G**reen, **B**lue) і дозволяє представляти майже будь-який колір. Використовується пространство RGB найчастіше для відображення графічної інформації на пристроях, що світяться — моніторах, проекторах, телевізорах. Цей метод називають аддитивним, бо кольори додаються до чорного для отримання вихідного кольору.

Для поліграфічної продукції зазвичай використовується інша схема CMYK (**C**yan, **M**agenta, **Y**ellow, **b**lac**K** — голубий, пурпуровий, жовтий, чорний). Ця схема більше підходить для друку кольорових зображень на папері. Цю схему називають субтрактивною, тому що для отримання вихідного кольору від білого віднімаються послідовно всі компоненти.

1.2.4.5. Кодування аудіо- та відеоінформації.

Крім статичної в часі текстової та графічної інформації в сучасних інформаційних системах дуже широко приміняються такі види інформації, як аудіо та відео. Для цих видів інформації притаманні великі об'єми і складність обробки, що і затримувало їх розповсюдження. Зараз технічна база навіть не самого швидкодіючого обчислювального пристрою (наприклад, мобільного

телефону) дозволяє виконувати базові операції з цією інформацією — запис, відтворення, зберігання та транспортування.

Базові принципи кодування для аудіо і відео співпадають. Вхідний сигнал з приймача (мікрофон або матриця відеокамери) дискретизується за допомогою аналогово цифрового перетворювача (АЦП) - analog-digital converter (ADC) в двійковий код, який далі оброблюється таким чином, щоб займав якомога менше місця. Коли треба відтворити цю закодовану інформацію, викорисотвують пристрій, який виконує обратне перетворення цифро-аналоговим перетворювачем (ЦАП) - digital-analog converter (DAC), який передає вихідний сигнал або на монітор, або на динамік. В цьому сенсі кодування відео відрізняється від кодування аудіо лише тим, що для відео використовується двовимірна модель перетворення, а для аудіо — одновимірна, але по декількох каналах (у випадку стереосигналу).

Для автоматичної обробки інформації в комп'ютерних системах використовується уніфікована форма представлення даних - система двійкового кодування.

1.3 ІНФОРМАЦІЙНІ СИСТЕМИ

Сучасна інформаційна система передбачає використання комп'ютерної техніки і спеціалізованого програмного забезпечення, як основного засобу обробки інформації. Характерною особливістю інформаційної системи є обробка даних за активною участю людини.

У Державному Стандарті України ДСТУ 2874-94 дано таке визначення інформаційної системи:

Інформаційна система - *система, яка організовує пам'ять і маніпулювання інформацією щодо проблемної сфери.*

У роботі інформаційної системи можна виділити такі етапи:

- *Надходження даних* - одержання первинних повідомлень, що визначають результати певних операцій, властивості об'єктів і суб'єктів управління, параметри процесів, зміст нормативних та юридичних актів тощо.
- *Накопичення і систематизація даних* - організація розміщення даних, яка б забезпечувала швидкий пошук і відбір потрібних відомостей, методичне оновлення даних, захист їх від спотворень, втрати, деформування цілісності та ін.
- *Обробка даних* - організація інформаційних процесів, внаслідок яких

на підставі раніше накопичених даних формуються нові види даних: узагальнюючі, аналітичні, рекомендаційні, прогнозні і т.і. Похідні дані також можуть зазнавати подальшого оброблення, даючи більш результатну інформацію і т. д.

- *Зображення даних* - подання результатів інформаційного процесу у формі, придатній для сприйняття людиною: документів, графічних ілюстративних матеріалів (графіків, діаграм), у вигляді відео- аудіо- сигналів.

Повідомлення першого етапу можуть бути у вигляді звичайних паперових документів, у "машинному вигляді" або у тим й іншим одночасно. В сучасних інформаційних системах повідомлення масового характеру здебільшого мають "машинний вигляд". Технічні засоби, що використовується при цьому, мають назву *засоби реєстрації первинної інформації*.

Потреби другого і третього етапів задовольняються в сучасних інформаційних системах в основному засобами зображення інформації, компонентами комп'ютерної техніки.

Переважна більшість інформаційних систем працює в режимі діалогу з користувачем. Типові програмні компоненти інформаційних систем включають:

- діалогову підсистему введення-виведення,
- підсистему, яка реалізує логіку діалогу,
- підсистему прикладної логіки обробки даних,
- підсистему логіки управління даними.

Для мережевих інформаційних систем важливим елементом є комунікаційний сервіс, який забезпечує взаємодію вузлів мережі при спільному вирішенні задачі. Значна частина функціональних можливостей інформаційних систем забезпечується системним програмним забезпеченням: операційними системами, системними бібліотеками та інструментальними засобами розробки. Важливу роль відіграє інформаційна складова, яка тісно пов'язана з логікою управління даними і задає структуру, атрибутуку та типи даних.

Інформаційні системи класифікуються за наступними ознаками:

- За ступеню автоматизації: ручні, автоматизовані, автоматичні.
- За виконуваною функцією: інформаційно-пошукові, керуючі, моделюючі, навчальні і ін.

- За галуззю застосування: виробничі, фінансові, лінгвістичні, медичні і ін.

Будь-яку інформаційну систему можна представити сукупністю підсистем, які забезпечують функціонування системи в цілому. До таких підсистем можна віднести:

- технічне забезпечення;
- математичне забезпечення;
- програмне забезпечення;
- інформаційне забезпечення;
- організаційне забезпечення;
- правове забезпечення.

Впровадження інформаційної системи може сприяти

- прийняттю більш раціональних обґрунтованих рішень;
- звільненню персоналу від рутинної роботи;
- зменшенню витрат на виробництво;
- скороченню персоналу і т. ін.

1.4 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Інформаційні технології швидко розвиваються, охоплюючи всі види суспільної діяльності: виробництво, управління, науку, освіту, фінансово-банківські операції, медицину, побут та ін.

Інформаційною технологією називається комплекс засобів, методів і способів збору, обробки, накопичення, зберігання, транспортування і виводу інформації.

Сучасні інформаційні технології не можуть існувати без обчислювальної техніки і засобів зв'язку. Залежно від вирішуваних задач інформаційні технології можна поділити на наступні типи:

- технології обробки даних,
- технології керування,
- технології автоматизації,
- технології підтримки прийняття рішень,
- технології експертних систем і т. ін.

Контрольні питання та завдання

1. Наведіть визначення інформатики як науки.
2. Що таке система числення?
3. Які бувають системи числення?
4. Яка система числення використовується в інформаційних системах?
5. Перекладіть число $10011001_{(2)}$ в десяткову систему числення.
6. Перекладіть число $32767_{(10)}$ в двійкову і шістнадцяткову системи числення.
7. Перекладіть дійсне число $FE, EF_{(16)}$ в десяткову систему числення.
8. Наведіть визначення інформації.
9. Яким чином вимірюється об'єм (кількість) інформації?
10. Що таке біт?
11. Скільки треба біт для кодування всіх літер української абетки?
12. Наведіть приклади кратних одиниць кількості інформації.
13. В скільки разів 1Мбіт більше 1Кбайт?
14. Що більше: 2^{10} байт чи 10^3 байт?
15. Що таке кодування?
16. Які системи кодування текстової інформації ви знаєте?
17. Чим відрізняється кодування текстової та графічної інформації?
18. Наведіть визначення і основні етапи роботи інформаційної системи.
19. Що таке інформаційні технології?

2 АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ

Винайдення засобів і методів механізації і автоматизації робіт - одне з основних завдань технічних дисциплін. Автоматизація робіт з даними має свої особливості і відмінності від автоматизації інших типів робіт. Для цього класу завдань використовують особливі види пристроїв, більшість з яких є електронними приладами. Сукупність пристроїв, призначених для автоматичної або автоматизованої обробки даних в цифровій формі, називають обчислювальною технікою. Конкретний набір взаємодіючих між собою пристроїв і програм, призначений для обслуговування однієї робочої ділянки, називають обчислювальною системою. Центральним пристроєм більшості обчислювальних систем є *комп'ютер*.

Комп'ютер - це електронний прилад, призначений для автоматизації створення, зберігання, обробки і транспортування даних.

На даний момент майже усі обчислювальні прилади створені на базі електронних елементів і працюють з інформацією, представленою в бінарному вигляді, за попередньо складеними програмами.

2.1 ВИЗНАЧЕННЯ ЕОМ.

ЕОМ (електронно-обчислювальна машина, комп'ютер) — це програмно-апаратний комплекс, який призначається для накопичення, обробки, транспортування і представлення інформації у вигляді, потрібному користувачеві. Інформація в комп'ютері представляється в цифровому вигляді. Відповідно до заданої програми обчислень, комп'ютер виконує певний обчислювальний процес.

Як можна помітити, комп'ютер, це сукупність двох великих складових частин (апаратного та програмного забезпечення), взаємодія яких і становить інформаційний процес. Без апаратного забезпечення як виконавця неможливе ефективне оброблення інформації. Також неможливе воно і без методів (алгоритмів, програм), тобто, інструкцій для виконавця.

2.2 КЛАСИФІКАЦІЯ КОМП'ЮТЕРІВ

Класифікація обчислювальної техніки досить складна, тому що ця техніка знаходиться в постійному розвитку. Кожен рік з'являються нові і нові застосування для неї, випускаються нові пристрої, що розширюють сферу використання. У той же час, деякі тупікові відгалуження розвитку перестають існувати.

Однак, по деяких основних властивостях проаналізувати найбільш уживані типи комп'ютерної техніки можливо.

2.2.1 Класифікація комп'ютерів

2.2.1.1. За призначенням. Комп'ютери поділяються на універсальні і спеціалізовані

Універсальні застосовуються для рішення будь-яких задач — обчислювальних, розважальних, навчальних, медичних, інформаційних, тощо. Приклад — персональний комп'ютер.

Спеціалізовані застосовуються для рішення задач певного класу. Наприклад, система управління технологічним процесом, бортова ЕОМ літака і т. ін..

2.2.1.2. За характером розв'язуваних задач. Комп'ютери поділяються на обчислювальні, інформаційні та керуючі.

Обчислювальні використовуються для математичної обробки інформації.

Інформаційні використовуються для зберігання, пошуку і видачі інформації.

Керуючі використовуються для обробки інформації в системах керування.

2.2.1.2. По габаритах. Існує велика кількість типорозмірів комп'ютерів — від долей сантиметру (наприклад, RFID- мікросхеми для інвентаризації одиниць товарів) і до великих супер ЕОМ, займаючих цілі будинки.

По типорозмірах ЕОМ поділяють на супер-ЕОМ, міні-ЕОМ, мікро-ЕОМ, персональні ЕОМ.

Супер-ЕОМ — дуже великі і дуже швидкодіючі системи, на основі яких створюються обчислювальні центри (або центри обробки даних — ЦОД). Вони мають дуже багато окремих процесорних блоків, дуже багато оперативної пам'яті та дуже великі дискові масиви.

Супер-ЕОМ також і найдорожчі серед усіх типів ЕОМ. На даний момент

це не обов'язково один пристрій. Супер-ЕОМ можна побудувати за допомогою дешевих масових комп'ютерів, об'єднаних в мережу. Така система називається *класстером* і може посперечатися швидкістю роботи зі справжніми Супер-ЕОМ, а за ціною бути на багато порядків меншою. Сьогодні кожен, хто має необхідність і деяку (не дуже велику) кількість коштів, може створити для себе особистий суперкомп'ютер майже будь-якої потужності на основі серійних комп'ютерних компонентів.

Але, зазвичай, супер-ЕОМ встановлюються у великих корпораціях або наукових закладах, тобто там, де необхідно мати дуже велику обчислювальну потужність, а ціна відходить на другий план.

Міні-ЕОМ. Цей клас машин був широко розповсюджений в 60-80 роках 20 століття. Комп'ютери цього класу виконували задачі, пов'язані з обчисленнями для окремих навчальних або наукових закладів, або підприємств. На даний момент задачі, що вирішувалися на міні-ЕОМ, взяли на себе персональні комп'ютери, а справжніми міні-ЕОМ стали *системи керування виробництвом* і складними виробничими процесами в хімічній, металургійній, транспортній, енергетичній галузі. Ці системи представляють собою розгалужену мережу з персональних комп'ютерів, серверів, спеціалізованих промислових комп'ютерів, масивів даних, тощо

Мікро-ЕОМ. Сьогодні цей клас машин представляють різноманітні серверні системи. Середньостатистичний сервер — це окремий комп'ютер, що використовується для надання своїх ресурсів — обчислювальних, транспортних, ресурсів для зберігання даних. Найчастіше від персонального комп'ютера сервер майже нічим не відрізняється, окрім більшої кількості пам'яті або великого масиву даних. Але, до сервера часто пред'являються високі вимоги щодо надійності, безвідказної роботи на протязі років, тому виконують сервери із більш якісних і, звичайно, більш дорогих комплектуючих.

Персональні ЕОМ. Це найбільш масовий і найбільш різноманітний клас комп'ютерів. До нього можна включити комп'ютери від спеціалізованих робочих станцій і до мобільних телефонів. Відрізняються комп'ютери цього класу тим, що користується ними лише один користувач, хоча деякі ресурси свого ПК він може віддавати в суспільне використання.

По типорозмірах та виконанню ПК можна поділити на *стаціонарні* та *мобільні* (переносні).

Стаціонарні комп'ютери виконують у вигляді системного блоку, який встановлюється на стіл або на підлогу. До нього під'єднуються периферійні пристрої та пристрої вводу-виводу.

Мобільні комп'ютери виконуються у вигляді або книжки (ноутбуки),

або у вигляді моноблоку (кишенькові ПК, смартфони, планшетні ПК). Для вимог мобільності в цих комп'ютерах усі стандартні пристрої вводу-виводу (екран, клавіатура, тачпад) робляться вмонтованими.

Ця класифікація не є повною і вже встигла застаріти. Наприклад, майже зникли міні- і мікро-ЕОМ із-за збільшення потужності персональних ПК, а нові типи комп'ютерів з'явилися. Також, майже кожен комп'ютер завдяки великій обчислювальній потужності спроможний виконувати майже будь-яку задачу з певною мірою ефективності. Тобто, навіть на ноутбуку можна порахувати яку-небудь обчислювальну задачу, що її обчислення займало багато часу на супер-ЕОМ тридцятирічної давнини. Але ефективніше використовувати інструмент за призначенням.

2.3 АРХІТЕКТУРА КОМП'ЮТЕРА

Архітектура сучасного комп'ютера встигла устоятися і перебуває незмінною близько півстоліття. Якщо не брати до уваги швидкодію та кількість транзисторів на одному процесорі, кожен комп'ютер від мейнфрейму і до калькулятора складається з одних і тих самих вузлів, перелік яких обґрунтував видатний американський вчений угорського походження Джон фон Нейман.

Архітектура комп'ютера - *логічна організація і структура апаратних і програмних ресурсів обчислювальної системи. Архітектура містить в собі вимоги до функціональності і принципи організації основних вузлів ЕОМ.*

В даний час найбільшого поширення в ЕОМ отримали два типи архітектури: *прінстонська* (фон Неймана) і *гарвардська*. Обидві вони виділяють два основних вузла ЕОМ: центральний процесор і пам'ять комп'ютера. Відмінність полягає в структурі пам'яті: в прінстонській архітектурі програми і дані зберігаються в одному масиві пам'яті і передаються в процесор по одному каналу, тоді як гарвардська архітектура передбачає окремі сховища і потоки передачі для команд і даних.

Складовими компонентами архітектури є:

АЛП (арифметико-логічний пристрій). Цей пристрій за певною програмою автоматично виконує арифметичні або логічні операції по переробці даних. В сучасному комп'ютері за це відповідає центральний процесор — велика інтегральна схема, яка в своєму составі має багато (сотні мільйонів) електронних елементів — транзисторів, конденсаторів, резисторів, тощо.

ОЗП (оперативний запам'ятовувальний пристрій). Цей пристрій нази-

вають оперативною або внутрішньою пам'яттю. Пам'ять зберігає дані і команди, що необхідно виконати для вирішення певної задачі. Дані і команди довільно, згідно з програмою, вилучаються з пам'яті, обробляються процесором, Результати обробки даних процесор повертає в пам'ять. В сучасних комп'ютерах ОЗП так і називається — оперативна пам'ять. Дані і команди в пам'яті зберігаються тільки під час його роботи. Після виключення комп'ютера всі дані з оперативної пам'яті зникають.

Зовнішня пам'ять. Це пам'ять, де зберігаються програми, дані до обробки і результати обробки. Зовнішня пам'ять, зазвичай, більша за оперативну, але працює повільніше. Проте, зовнішня пам'ять може зберігати дані теоретично необмежену кількість часу, навіть при відсутньому живленні. Зараз існує безліч пристроїв зовнішньої пам'яті з різними характеристиками і принципом дії. Найбільш живі з них — накопичувачі на жорстких магнітних дисках, оптичні носії, накопичувачі на флеш-пам'яті.

Засоби вводу-виводу. Цей тип пристроїв забезпечує обмін інформацією для комп'ютера із зовнішнім світом — користувачем та іншими комп'ютерами. За допомогою приладів вводу-виводу здійснюється керування інформаційною системою і координується протікання інформаційного процесу. Їх можна з повною підставою вважати органами почуттів комп'ютера. До цих важливих елементів відносять монітори усіх видів, клавіатури, “миші”, принтери, сканери, сенсорні панелі, мережні пристрої, тощо.

Комп'ютер, який має подібну архітектуру, називається комп'ютером з **класичною архітектурою**. Звичайно, реальний комп'ютер може мати набагато складнішу і досконалішу архітектуру, може мати не один, а два або чотири процесори, зовсім не мати зовнішньої пам'яті. Але, за основними принципами, за якими виконується інформаційний процес, цей комп'ютер усе-таки належить до класичної архітектури. Ці принципи уперше використав в одному обчислювальному пристрої ще Конрад Цузе, але він не зумів їх повністю сформулювати. Лише в 1946 році Джон фон Нейман сформулював ці прості принципи, які і отримали його ім'я.

Отже, наведемо принципи фон Неймана:

Принцип програмного управління. Програма складається з набору команд, які виконуються одна за одною в певній послідовності.

Принцип адресності пам'яті. Оперативна пам'ять розбита на окремі комірки, кожна із них має свою унікальну адресу і доступна для процесора в будь-яку мить.

Принцип однорідності пам'яті. І команди, і дані, що обробляю-

ться, а також результати обробки знаходяться в одній пам'яті, над командами можливо виконувати ті самі операції, що і над даними.

2.4 КОНСТРУКТИВНЕ ВИКОНАННЯ СУЧАСНОЇ ЕОМ

Хоча майже всі сучасні комп'ютери мають класичну архітектуру, конструктивно вони можуть доволі сильно відрізнятися один від одного. Наприклад, на перший погляд, стійковий сервер і мобільний комп'ютер майже не мають спільних елементів, за виключенням, можливо, самих елементарних — транзисторів, резисторів і т. ін. Але і перший, і другий створені за допомогою так званого *магістрально-модульного принципу*.

Цей принцип заключається в тому, що кожний з окремих, логічно завершених елементів комп'ютера виконується у вигляді блоку, який під'єднується до інших блоків за допомогою так званої *магістралі* або *системної шини*. Шина виконує транспортні функції, функції керування і функції живлення для окремих блоків комп'ютера.

Цей принцип дозволяє користувачеві гнучко підлаштовувати апаратну конфігурацію комп'ютера під свої потреби, замінити окремі вузли, які вийшли з ладу, або на більш сучасні, виконуючи таким чином вдосконалення (чи, як зараз кажуть, *апгрейд*) комп'ютера.

Магістрально-модульний принцип вперше масово використала корпорація IBM в серії персональних комп'ютерів IBM PC, що вийшли в продаж на початку 80-х років минулого століття. Це рішення дозволило корпорації, яка ніколи раніше не займалася виробництвом персональних комп'ютерів, зайняти провідні позиції в цьому секторі ринку, і створити негласний стандарт IBM PC -сумісного комп'ютеру. До того ж, IBM відкрила специфікації на свої розробки, що дозволило безлічі електронних компаній створювати периферійні пристрої, сумісні з цими популярними системами.

Сучасні персональні комп'ютери виконуються в різних конструктивних варіантах в залежності від потужності, призначення та інших факторів, що впливають на зовнішній вигляд комп'ютера. Але найбільш поширеною конструктивною моделлю для персональних (десктопних) комп'ютерів є сукупність наступних елементів: системний блок, засоби вводу-виводу (монітор, принтер, клавіатура, “миша” тощо), кабелі для поєднання окремих компонентів і кабелі живлення для системного блока і периферії.

2.4.1 Системний блок

Системний блок — є головною складовою комп'ютера. Він містить в собі електронні та електричні (наприклад, блок живлення) компоненти, що потрібні для забезпечення роботи комп'ютера. Корпус системного блока виконується із різноманітних матеріалів (переважно зі сталі або алюмінію). Він повинен захистити від впливу зовнішніх факторів електронні компоненти і забезпечити стаке оточуюче середовище (температура, вологість). Передня панель корпусу використовується для виводу елементів керування: кнопки живлення, кнопки примусового перезавантаження, фронтальних панелей накопичувачів зі змінними носіями, периферійних портів тощо. На задній панелі корпусу виводяться роз'єми блока живлення, периферійні і комунікаційні порти а також вентиляційні отвори для відведення нагрітого повітря.

Існує безліч різноманітних системних блоків. Найбільш вживані фактори:

- Desktop (настільний, горизонтальне виконання, повна товщина);
- SlimLine (настільний, горизонтальне виконання, зменшена товщина);
- Tower (підлоговий або настільний, вертикальне виконання);
- Midi Tower (настільний, вертикальне виконання, зменшена висота);
- MiniITX (настільний, вертикальне або горизонтальне виконання, зменшені розміри).

Від розміру корпусу залежить максимальний розмір встановленої материнської плати і можливість розширення функцій комп'ютера шляхом додавання або оновлення складових системного блока.

Існує ще один розповсюджений і набираючий популярності вид корпусу - ноутбук (англ. *Notebook*, записничка). Состав ноутбука в цілому повторює десктопний ПК, тільки в більш компактному і легкому варіанті. Наприклад, для накопичувачів замість формату 3.5" використовується формат 2.5". Також в корпус ноутбука вмонтовується акумуляторна батарея для автономної роботи, скорочена і зменшена клавіатура і маніпулятор "тачпад". Вбудований екран на внутрішній поверхні кришки, що закриває клавіатуру. За більшістю показників ноутбуки не відрізняються від десктопних ПК, а за зручністю роботи перевершують.

Системний блок містить наступні елементи: материнська плата, блок живлення, накопичувачі на жорстких магнітних дисках, накопичувачі на опти-

чних дисках, систему охолодження електронних компонентів (повітряну або водяну) тощо.

Материнська плата - головний електронний компонент, що об'єднує інші електронні компоненти в єдиний пристрій. Вона представляє собою багатошаровий лист текстоліту з розпаяними мікросхемами і роз'ємами (слотами) для підключення інших елементів комп'ютера. Існує декілька стандартних розмірів материнських плат:

- ATX (повний настільний формат);
- MicroATX (зменшений настільний формат, менша кількість слотів розширення);
- MiniITX (компактний формат для мультимедійних центрів тощо).

Основні компоненти, які можна виділити при розгляді материнської плати:

центральный процесор, набір системної логіки (чипсет), підсистема BIOS, оперативна пам'ять (ОЗП), плати розширення (відео- та аудіоплати, різноманітні контролери вводу-виводу, порти вводу-виводу тощо).

Центральний процесор (*Central Processor Unit, CPU*) - головний виконувальний компонент комп'ютера. Він виконує логічні і математичні операції над даними по заданій програмі. Це інтегральна мікросхема, виконана в пластиковому, керамічному або текстолітовому корпусі. З однієї сторони до ядра процесору приєднується металічна пластина для ефективного відведення тепла, а з іншої сторони виводяться контакти, якими процесор приєднується до роз'єму на материнській платі (сокету). Зазвичай на процесор установлюється радіатор з вентилятором (кулером) для кращого охолодження.

Процесор містить в собі одне або декілька ядер, що виконують арифметично-логічні операції, багаторівневу систему кешування даних і команд ("кеш" від англійського cache - кошик, надоперативна пам'ять). В останній час в процесор вмонтовуються ще і контролер пам'яті, і відеопроцесор, що зазвичай входять до набору системної логіки (чипсету). Такі процесори називають system-on-chip - система на одній мікросхемі.

Чипсет (*chipset*) - набір мікросхем, що розташований на материнській платі і призначений для підключення центрального процесора до оперативної пам'яті і контролерів периферійних пристроїв системної шини. Зазвичай до чипсета входять дві великі мікросхеми: "північний" та "південний" мости.

"Північний" міст відповідає за організацію взаємодії процесора, оперативної пам'яті і графічної підсистеми.

"Південний" міст забезпечує роботу периферійних контролерів вводу-виводу, плат розширення і контролерів накопичувачів.

До чипсету можуть входити додаткові мікросхеми, що забезпечують функціональність, відсутню в "північному" та "південному" мостах: мережні інтерфейси, аудіо-контролери, додаткові контролери портів вводу-виводу тощо.

Базова система вводу-виводу (*Basic Input-Output System, BIOS*) - одна або декілька мікросхем з енергонезалежною пам'яттю, що запаяні або вставлені в спеціальні роз'єми материнської плати. BIOS містить декілька спеціальних програм для забезпечення запуску комп'ютера і підтримки стандартних пристроїв вводу-виводу (клавіатура, "миша", текстовий режим відеокarti тощо). BIOS також має невелику область пам'яті (зазвичай її називають CMOS), де зберігаються налаштування апаратного забезпечення комп'ютера у відсутність живлення (за допомогою батарейки).

Оперативна пам'ять (оперативно-запам'ятовувальний пристрій, ОЗП (*Random Access Memory, RAM*) пам'ять з довільним доступом - пристрій для тимчасового збереження даних і програм. Представляє собою набір мікросхем, запаяних на одній або декількох платах, що вставляються в спеціальні слоти на материнській платі. ОЗП є більш швидкодіюною пам'яттю, ніж інші види (наприклад, жорсткі диски), але вона потребує живлення. За його відсутності ОЗП втрачає інформацію.

Плати розширення - спеціалізовані контролери, що розширюють можливості комп'ютера з обробки інформації (наприклад, графічні плати, професійні аудіо-карти, плати захвату відео, аналого-цифрові перетворювачі), додають нові інтерфейси до вже існуючих (наприклад, контролери шини FireWare, SCSI, USB), розширюють комунікаційні можливості системи (наприклад, контролери бездротового зв'язку WI-Fi, мережні карти з підтримкою оптоволоконних мереж). Для плат, що зараз вважаються стандартними для кожного ПК застосовується назва "дискретні", для того, щоб підкреслити, що ці плати розроблялися окремо від материнської плати і мають кращі характеристики, ніж вбудовані (майже завжди так і є).

Графічна плата - спеціалізована плата розширення, що виконує операції виводу зображення на монітор, представляє собою набір мікросхем (один або декілька спеціалізованих графічних процесорів, графічної пам'яті, цифро-аналогових перетворювачів тощо), розташованих на окремому листі текстоліту. Графічна плата вставляється в роз'єм на материнській платі. В залежності від типу, графічна плата може підключатися до роз'єму шини PCI, AGP (за-

старілі) або PCI-Express (сучасні). Для отримання більшої графічної потужності можливе одночасне використання декількох однотипних графічних плат (якщо це дозволяє материнська плата), створених за однією із технологій - ATI CrossFire або NVIDIA SLI. Для систем, невибагливих до графічної потужності, застосовуються материнські плати з вбудованим графічним контролером. Такі системи часто використовують в якості офісних або навчальних станцій.

Накопичувачі - велика категорія пристроїв, що можуть зберігати великі об'єми інформації протягом тривалого часу у відсутності живлення. Слід зауважити, що для зміни збереженої інформації для будь-якого пристрою живлення усе одно необхідне.

Накопичувачі на жорстких магнітних дисках НЖМД (*Hard Disk Drive, HDD*) або "вінчестери" - пристрої, призначені для тривалого зберігання великих об'ємів інформації. Цей накопичувач представляє собою один або набір з декількох дисків з магнітним шаром, що одягаються на одну вісь. Вісь із дисками розкручується двигуном до швидкості в кілька тисяч обертів на хвилину (від 4200 - моделі для ноутбуків - до 15000 об./хв. - для високопродуктивних систем). Для запису та зчитування інформації до кожної поверхні кожного диску підводиться своя магнітна головка. Всі елементи накопичувача окрім електроніки управління пересуванням головок і обертанням шпинделя двигуна (контролера) розташовуються в міцному металевому корпусі (гермозоні) з повітряними фільтрами або тонкою мембраною. Мікросхеми контролера розташовуються зовні на приєднаній монтажній платі. Розміри сучасних жорстких дисків становлять 3.5" для звичайних застосувань, 2.5" - для ноутбуків і 1.8" - для вбудованих застосувань і мобільних пристроїв. Крім стаціонарних жорстких дисків існують переносні моделі, що підключаються до комп'ютера через шину USB або eSATA. Швидкість обміну для жорстких дисків сягає кількох десятків Мбайт/с. Ємкість сучасних моделей перевищує кілька Тбайт. Зараз жорсткі диски є найбільш оптимальним засобом зберігання великої кількості інформації з можливістю швидкого доступу до неї.

Накопичувачі на оптичних дисках (CD-ROM Drive, *Compact Disc Read-Only Memory Drive*) - пристрої, призначені для тривалого зберігання великих об'ємів інформації, яку переважно зчитують і рідко записують. Носій компакт-диск (*Compact Disc, CD*) представляє собою диск із полікарбонату розміром 12 см. На цьому диску створюється спеціальний відбиваючий світло шар із заглибинами і пагорбами розміром в один мікрометр (10^{-6} м.). За допомогою цих нерівностей кодується збережена інформація. Зчитати її можна за допомогою лазера. Накопичувач представляє собою пристрій з двигуном, який розкручує компакт-диск, каретку з лазером для зчитування і каретку для встановлення/вилучення компакт-диску із приводу. Компакт-диск

може вміщувати до 700 Мбайт, а швидкість зчитування - 7-8 Мбайт/с. Спочатку компакт диск розроблявся як місткий і якісний засіб збереження аудіозаписів. Тому зміна один раз записаної при створенні носія інформації не передбачалася. Пізніше з'явилися пристрої для одноразового (CD-R, *CD Recordable*) і багаторазового (CD-RW, *CD Rewritable*) запису інформації. Для збільшення об'єму збереженої інформації застосовуються двошарові і двосторонні компакт-диски. Компакт-диск для одноразового запису є найбільш оптимальним на даний момент засобом для тривалого збереження інформації. Більш місткі і швидкісні DVD приводи (*Digital Video (Versatile) Disc Read-Only Memory Drive*) - продовження розвитку оптичних носіїв. За рахунок використання меншої довжини хвилі синього лазера вдалося досягти об'єму в 4.7 Гбайт на один шар і одну сторону (17 Гбайт для двошарових двосторонніх дисків) при збереженні розмірів диску носія в 120 мм. Разом зі збільшенням ємкості збільшилася швидкість обміну до 20Мбайт/с, але зменшилася і надійність зберігання інформації.

На даний момент приводи DVD-RW спроможні багаторазово перезаписувати інформацію на DVD-RW дисках, а також зворотньо сумісні з усіма попередніми форматами звичайних CD, CD-R, CD-RW дисків.

Флеш-накопичувачі (flash — блискавка) - пристрої енергонезалежної твердотілої пам'яті, що може перезаписуватися. Представляє собою одну або декілька мікросхем, що зберігають інформацію за допомогою електричних зарядів. До мікросхем пам'яті зазвичай додається контролер, який керує записом та зчитуванням інформації. До комп'ютера такий накопичувач приєднується через шину USB або через спеціальний карт-рідер (*card-reader*). Використовуються накопичувачі з об'ємом пам'яті від декількох Кбайт і до декількох Тбайт. Мінімальні розміри і невибагливість до живлення дозволяють використовувати цей тип накопичувачів в мобільній та побутовій техніці, фотоапаратах і мультимедійних пристроях. Флеш-пам'ять також використовується для створення накопичувачів SSD (*Solid-State Drive*, твердотілий накопичувач), що застосовуються як основні пристрої зовнішньої пам'яті. Головною перевагою флеш-пам'яті над іншими видами енергонезалежної пам'яті є відсутність пересувних частин і захищеність майже від усіх видів зовнішнього впливу. Основний недолік полягає в невеликій кількості циклів перезапису (близько 100000 раз) у порівнянні з жорстким диском, менша швидкість запису (в 2-3 рази) і більша вартість. Тому флеш-накопичувачі застосовуються здебільшого не для зберігання, а для переносу інформації.

Блок живлення БЖ (*Power Supply Unit, PSU*) - пристрій для постачання живлення компонентам комп'ютера. Він перетворює зовнішнє живлення побутової електромережі (напруга 220В, змінний струм частотою 50Гц) в постійний струм напругою $\pm 3.3\text{В}$, $\pm 5\text{В}$, $\pm 12\text{В}$. Потужність сучасних блоків жи-

влення становить від 50 Вт (для вбудованої техніки) і до 1600 Вт (для високо навантажених систем). Зазвичай, БЖ поставляється разом із корпусом (для дешевих конфігурацій). В більш дорогих системах БЖ вибирається згідно з запланованим споживанням потужних компонентів комп'ютера з деяким запасом (10-20%). Для комп'ютерної техніки використовувалося два типи БЖ: АТ(застарілий) і АТХ(сучасний). Для кожного типу БЖ використовуються свої роз'єми на материнській платі. Деякий час існували системи з підтримкою як старого, так нового стандартів. В нових комп'ютерах використовується тільки стандарт АТХ. Слід зауважити, що від правильно підібраного БЖ залежить надійна і безперервна робота всього комп'ютера.

Система охолодження - частина системного блока, що відповідає за відведення надмірного тепла від електронних компонентів під час роботи комп'ютера. Вона може виконуватись на основі пасивних елементів (радіаторів, теплових трубок), повітряних вентиляторів (кулерів), водяних pomp тощо. Для створення оптимальних температурних умов для компонентів системного блока необхідно ретельно вибирати компоненти системи охолодження, починаючи з просторого і добре провітрюваного корпусу із зарезервованими отворами для додаткових вентиляторів, і закінчуючи якісним охолодженням процесору, графічної карти і накопичувачів. Слід пам'ятати, що надмірна потужність системи охолодження може доставляти акустичний дискомфорт на робочому місці і всмоктувати в корпус велику кількість пилу, що негативно впливає на охолодження і стабільність роботи комп'ютера. Тому, навіть для систем з пасивним охолодженням існує потреба періодичного видалення пилу із системного блока.

2.4.2 Периферія

До цього класу пристроїв можна віднести всі пристрої, що підключаються до зовнішніх портів системного блока. Розглянемо стислі характеристики кожного типу пристроїв.

Монітор (дисплей) - головний пристрій виводу, призначений для формування зображення текстової та графічної інформації.

Монітори можна розділити на текстові (виводять тільки текстові символи) і графічні (виводять довільну інформацію).

За засобом створення зображення графічні монітори можна розділити на

- растрові (зображення будується сукупністю малих елементів - пікселів)

- векторні (зображення будується як сукупність геометричних примітивів - ліній, кіл, кривих тощо).

Існує досить багато технологій створення зображення на екрані монітору. Перелічимо деякі із них:

- на основі електронно-променевої трубки (*Cathode Ray Tube, CRT*),
- на основі рідких кристалів (*Liquid Crystal Display, LCD*),
- на основі плазмових панелей,
- на основі OLED (*Organic Light Emitted Display*).
- Монітори можуть характеризуватися наступними параметрами:
- Розміром (діагоналі в дюймах),
- Співвідношенням сторін (наприклад 4:3 або 16:9),
- Розподільчою здатністю (в пікселях по горизонталі і вертикалі, наприклад 1280x1024),
- Частотою кадрової розгортки (частотою кадрів в герцах, наприклад 85 Гц).

Найбільш розповсюдженими є CRT- і LCD-монітори. Розглянемо детальніше їх принципи роботи.

CRT-монітор містить електронну вакуумну трубку з нанесеним на внутрішню поверхню екрана люмінофором (речовиною, що світиться при опроміненні потоком електронів). На другому кінці закріплюється електронна пушка з системою керування електронними променем, що за допомогою високої напруги (близько 20 кВ розганяє електрони. Система керування відхиляє промінь таким чином, щоб він проходив по всьому екрану порядково (малював кадр) кілька десятків разів за секунду. Вже при частоті 25 кадрів (стандарт для кінофільмів), картинка може передавати плавно рухи. Але для комфортної роботи з дрібними елементами зображення слід використовувати частоту кадрів починаючи з 85 Гц.

Для CRT-моніторів створювалися екрани з відношенням сторін 4:3 і розмірами до 24". Розподільна здатність сягала 2560x2048 пікселів.

Монітори на основі CRT були історично першими і до цього часу досить широко використовуються, особливо в професійних застосуваннях (дизайн,

обробка фотографій), тому що в них якісніша передача кольору зображення, ніж в LCD-моніторах.

Переважна більшість LCD-моніторів виконана за технологією TFT (тонкоплівкових транзисторів, *Thin Film Transistor*). Активний елемент LCD-монітора — матриця - створюється з сукупності окремих елементів (пікселів), розташованих на загальній підложці. Кожний піксель матриці створюється із трьох субпікселів зеленого, синього і червоного кольору. Кожний із них містить шар рідкокристалічних молекул між двома прозорими електродами, поляризаційних фільтрів і кольорового фільтру. При подаванні напруги на електроди рідкокристалічні молекули вишиковуються вздовж ліній електричного поля, що змінює кількість світла, що проходить крізь піксель. Для підсвічування матриці використовується газонаповнена лампа.

Блок керування виконує перетворення сигналу, що поступає з комп'ютера, у відповідне зображення на екрані, маніпулюючи напругою на всіх пікселях матриці. При цьому операція розгортки кадру виконується майже миттєво, а між кадрами зображення залишається довше, ніж на CRT-моніторах. Тому LCD-монітори більше підходять для тривалої обробки документів і читання з екрану. До того ж вони займають набагато менше місця і потребують менше енергії. До недоліків цього типу моніторів можна віднести дещо меншу контрастність, помилки в передаванні кольору зображення, невеликі кути огляду і змазування зображення в динамічних сценах.

Для сучасних моніторів на основі рідких кристалів випускаються екрани з співвідношенням сторін 4:3, 5:4, 16:9 і 16:10. Останні два типи називаються широкоформатними і більш зручні для перегляду фільмів і праці з документами, але мають дещо меншу площу зображення, ніж монітори з такою самою діагоналлю класичних пропорцій.

Монітор підключається до порта графічної карти. На даний час існує декілька різновидів цих портів:

- VGA (*Video Graphics Array*) - аналоговий стандарт для моніторів і відеокарт (використовується на дешевих моделях моніторів і вбудованих відеокартах);

- DVI (*Digital Video Interface*) - цифровий стандарт для підключення сучасних LCD-моніторів і проекторів;

- HDMI (*High Definition Multimedia Interface*)- мультимедійний інтерфейс з високим розподіленням (застосовується для передачі аудіо- і відеоінформації в цифровому виді)

- Display Port - новий вид інтерфейсу для передачі цифрового аудіо- і відеосигналу, альтернатива HDMI.

Принтер - пристрій для друкування текстової та графічної інформації на паперовий носій. За технологією друку принтери можна поділити на:

матричні, струменеві, лазерні, сублімаційні.

Матричні принтери (*Serial Impact Dot Matrix, SIDM*) є найстарішим типом принтерів. Функціонально вони подібні до друкарських машинок. Тільки в них лише одна головка, в якій вбудовано від 9 до 24 голок (матриця). Проходячи над місцем, де повинно бути символ, голки вдаряють по паперу через стрічку з сухими чорнилами. Із окремих точок формується зображення символу. Вони застосовувалися переважно для друку тексту. Друк на матричному принтері дуже повільний, але і самий дешевий.

Струменеві принтери схожі на матричні, але друк в них виконується за допомогою фарби, що випорскується на папір під тиском із спеціальних дюз. Для створення кольорового зображення застосовується декілька картриджів з фарбою - по одному на кожен головку. Роздрукований на струменевому принтері документ може мати якість, наближену до фотографічної, але це дуже збільшує витрати фарби. Друк на цьому типі принтерів швидший за матричний, але ціна і витратних матеріалів дуже висока.

Лазерні принтери - оптимальний варіант з точки зору якості і вартості відбитку. Принцип дії заснований на так званій електрографії. Металевий барабан заряджається статичною електрикою. Потім цей барабан опромінюється лазером в тих місцях, де потрібно створити зображення. На підготовлене, але ще не видиме зображення на барабані наноситься тонер. Він прилипає до тих місць, де заряду немає. Потім барабан прокатують по листу паперу і отриманий на ньому відбиток запікають в спеціальній пічці. Для отримання кольорового зображення застосовують декілька картриджів з основними кольорами. Відбиток отримують послідовним проходом паперу по всіх барабанах. Це найшвидший тип принтеру (до десятків сторінок за хвилину).

Сублімаційні принтери - досить нова технологія друку, що заснована на термосублімації - перетворенню твердого тіла на пар минаючи рідкий стан. В цьому принтері між друкарською головкою і папером протягується стрічка з нанесеним барвником. Головка дуже швидко нагріває мікроскопічні ділянки на стрічці і барвник переноситься на папір.

Отримуються досить якісні і стійкі до зовнішніх факторів відбитки (за умови застосування спеціального ламінованого шару). Але вартість таких відбитків досить велика.

Плотер - ще один варіант пристрою для перенесення зображення на папір. На відміну від принтерів, в плотері робочий елемент представляє собою каретку, яка може пересуватися над листом паперу в двох напрямках і

таким чином позиціюватися над будь-якою точкою листа. До каретки приєднується один або декілька олівців або фломастерів. Каретка за командами комп'ютера виконує певні рухи, що створюють зображення. Плотери використовуються для побудови зображень великих форматів (переважно конструкторської документації).

Для підключення принтерів до комп'ютера використовують наступні інтерфейси:

- послідовний порт COM (*Communication Port*) - застарілий повільний інтерфейс;
- паралельний порт LPT (*Local Printing Port*) - застарілий інтерфейс, але ще використовується;
- шина USB (*Universal Serial Bus*) - сучасний інтерфейс для периферійних пристроїв.

Аудіопристрої - сукупність периферійних пристроїв, що призначені для відтворення аудіо-інформації та введення її до інформаційної системи. До цієї групи можна віднести аудіоплати (як вбудовані, так і дискретні або навіть зовнішні), пристрої для підсилення звуку, аудіоколонки (гучномовці), мікрофони тощо. Сучасні стандарти для аудіо дозволяють отримувати якісне звучання з кількістю каналів від 2 (стерео) до 7.1 (7 середньочастотних і високочастотних каналів і один низькочастотний). Підключення аудіопристроїв до комп'ютера можливе як по аналогових каналах (лінійні і мікрофонні входи і виходи), так і по цифрових (SP-DIF або HDMI).

Клавіатура - пристрій для вводу символічної інформації за допомогою набору клавіш. Це головний "людино-орієнтований" пристрій, що використовується майже на всіх видах комп'ютерів. Клавіатура містить декілька наборів клавіш:

- алфавітно-цифрові клавіші (латинська та національна розкладки, цифри і додаткові символи);
- функціональні клавіші (F1-F12);
- блок цифрових клавіш;
- клавіші управління курсором;
- додаткові клавіші-модифікатори.

Клавіатура може підключатися до комп'ютера декількома шляхами

DIN - застарілий роз'єм, зараз не застосовується;

PS/2 - стандарт для підключення клавіатур і маніпуляторів "миша";

USB - сучасний інтерфейс для периферійних пристроїв.

Маніпулятори “Миша”, “Джойстик” - маніпулятори, що дозволяють людині вводити аналоговий сигнал для керування комп'ютером.

Підключаються маніпулятори до комп'ютера за допомогою одного із інтерфейсів: послідовний порт, PS/2, шина USB.

Сенсорні пристрої застосовуються для полегшення вводу даних. Зараз застосовується велика кількість пристроїв із сенсорними засобами вводу - мобільні телефони, ноутбуки, платіжні термінали тощо. Найбільш розповсюджені сенсорні пристрої: сенсорний екран і тачпад.

Сенсорний екран представляє собою звичайний LCD-монітор з нанесеною зверху прозорою чутливою плівкою. Ця плівка може реагувати або на тиск гострим предметом (стілусом), або на доторкання кінцівками пальців. Перша технологія більш проста і надійна, але менш зручна для людини (треба завжди мати стілус). Друга технологія більш зручна (не треба зайвих предметів), але погано працює на холоді і реагує тільки на кінцівки пальців. Треба зауважити, що для застосування сенсорного екрану необхідно істотно модифікувати інтерфейс програмного забезпечення (максимально збільшувати елементи керування, створювати зручні види екранних клавіатур тощо).

Тачпад більш розповсюджений пристрій. Він використовується як основний засіб керування курсором майже у всіх ноутбуках. Тачпад представляє собою невелику площу, чутливу до доторкання кінцівками пальців. Пересування пальця (або декількох) по тачпаду приводить до аналогічного пересування курсора на екрані ноутбука. До тачпада додаються одна або дві звичайні клавіші, що емітують клавіші "миші". В останніх версіях тачпадів з'явилася технологія "мультитач", що дозволяє виконувати розширені операції: масштабування, поворот скролінг документів.

Веб-камера - цифрова або аналогова відеокамера, що здатна передавати на комп'ютер зображення для подальшого передавання його в Internet. Звичай підключається через USB або FireWare шини. Використовується в основному для виконання відеодзвінків і відеоконференцій. Присутня майже на всіх сучасних моделях ноутбуків.

Джерело безперервного живлення, ДБП (*Uninterruptable Power Supply*,

UPS)- пристрій, що дозволяє підключеній до нього комп'ютерній техніці працювати без зовнішнього живлення деякий час.

Представляє собою окремий блок з акумуляторами і керуючу електроніку. При наявності живлення ДБП лише фільтрує скачки напруги. Коли живлення пропадає або напруга виходить за безпечні рамки, ДБЖ переходить на живлення від акумуляторів. Для офісних моделей час підтримки живлення вимірюється кількома хвилинами. За цей час можна зберегти роботу і коректно вимкнути комп'ютер. Більш потужні системи безперервного живлення можуть підтримувати роботу довший час. Головний параметр ДБЖ - потужність, що вимірюється в ватах (Вт) або вольт-амперах (ВА).

2.4.3 Засоби комунікації

До них можна віднести всі компоненти комп'ютера, що дозволяють підключати периферійне обладнання або інші комп'ютери і активні мережні пристрої.

Розглянемо спочатку локальні інтерфейси, що дозволяють підключати периферію до комп'ютера.

Послідовний порт (*Serial Port, Communication Port, COM Port*) - універсальний інтерфейс для двосторонньої передачі даних між комп'ютером і пристроєм (іноді, між двома комп'ютерами). Дані передаються послідовно (синхронно або асинхронно), по одному біту від джерела до приймача. Максимальна довжина лінії 30 м, а швидкість 115200 біт/с (при наявності екранованого кабелю). Найчастіше цей порт застосовувався для підключення маніпуляторів "миша", модемів і деяких принтерів. Був присутній на більшості комп'ютерів. В сучасних комп'ютерах може бути відсутній або не виведений на задню панель.

Паралельний порт (*Parallel Port*) - універсальний інтерфейс для двосторонньої передачі даних між комп'ютером і пристроєм. Дані передаються паралельно, по одному байту від джерела до приймача. Максимальна довжина лінії 3 м, а швидкість біля 2 Мбайт/с (в деяких режимах роботи). Найчастіше цей порт застосовувався для підключення принтерів, і сканерів. Був присутній на більшості комп'ютерів. В сучасних комп'ютерах може бути відсутній або не виведений на задню панель.

Порт PS/2 (*Personal System*) - порт для підключення "миші" і клавіатури. В персональних комп'ютерах прийшов на заміну роз'єму DIN (для клавіатури) і COM порту (для миші). В сучасних комп'ютерах (крім ноутбуків) встановлюється один універсальний (клавіатура або "миша") або два спеці-

алізованих порти PS/2. Потроху витісняється більш універсальною шиною USB.

Шина USB (*Universal Serial Bus*) - універсальний послідовний інтерфейс для периферійних пристроїв. За допомогою шини USB до одного контролера (на боці комп'ютера) може бути підключено до 127 пристроїв. Причому, серед них можуть бути як звичайні пристрої (клавіатура, флеш-накопичувач), так і спеціальні концентратори (хаби, *USB-Hub*), до яких також можна підключати пристрої. Глибина такої деревоподібної структури може досягати 5 рівнів. Підключення та відключення пристроїв USB може проводитися в будь-який момент (за виключенням накопичувачів, на які записується інформація). У зв'язку з тим, що USB шина не тільки передає дані, а і постачає живлення підключеним пристроям, вони можуть не мати власних блоків живлення (якщо вони в сумі не споживають більше 500мА струму при напрузі 5 В). Для більш потужних пристроїв (принтерів, сканерів тощо) потрібне власне живлення. Підключення до комп'ютера відбувається за допомогою спеціальних USB-кабелів, що можуть мати довжину не більше 5м. Швидкість шини USB змінювалася впродовж її розробки від 10-1500 Кбіт/с (версія USB 1.1) до 480 Мбіт/с (версія USB 2.0). Зараз набуває розповсюдження версія USB 3.0, що дозволить передавати інформацію зі швидкістю 4.8 Гбіт/с.

Шина FireWare (стандарт IEEE1394) - універсальна послідовна шина. Може використовуватися як для підключення периферійних пристроїв до комп'ютера, так і приєднання одного комп'ютера до другого. Пристрої також можуть утворювати деревоподібну структуру (до 64 пристроїв і до 16 рівнів) і отримувати живлення від комп'ютера. Ця шина є стандартним інтерфейсом в комп'ютерах Apple і дорогих версіях мобільних і мультимедійних пристроїв (фотоапаратів, відеокамер тощо). Довжина кабелю не повинна перевищувати 4.5м. Швидкість змінюється від 100 до 400 Мбіт/с. В останніх версіях стандарту максимальна довжина кабелю досягає 100 м (оптичний кабель), а швидкість до 3.2 Гбіт/с. Ця шина є більш дорогим і універсальним аналогом шини USB.

2.4.4 Внутрішні (локальні) інтерфейси

До внутрішніх інтерфейсів можна віднести шини, що застосовуються для підключення плат розширення і різноманітних накопичувачів всередині системного блока. Всі внутрішні інтерфейси сучасного комп'ютера можна умовно поділити на універсальні і спеціалізовані.

До універсальних внутрішніх інтерфейсів можна віднести наступні шини:

- Шина ISA (*Industry Standard Architecture*) - 8/16 розрядна шина

вводу-виводу для розширення IBM-PC сумісних комп'ютерів. Була широко розповсюджена на материнських платах формату AT. Зараз не використовується (за виключенням промислових комп'ютерів).

- Шина PCI (*Peripheral Component Interconnect*, шина для з'єднання периферійних компонентів) - універсальна 32/64-розрядна паралельна шина вводу-виводу для підключення плат розширення до материнської плати. Використовується на всіх сучасних комп'ютерах. Максимальна пропускна здатність 533 Мбайт/с.
- Шина PCI-E (*PCI-Express*) - високопродуктивна універсальна послідовна шина. Є розвитком шини PCI, але в послідовному варіанті. Використовується в основному для підключення високопродуктивних графічних карт, контролерів накопичувачів і мережних карт. Кожний пристрій на шині може використовувати від одного (PCI-E x1) до 32-х каналів (PCI-E x32) . Максимальна пропускна здатність 5Гбіт/с (на один канал).

До спеціалізованих внутрішніх інтерфейсів можна віднести наступні шини:

- Шина AGP - (*Accelerated Graphics Port*, прискорений графічний порт) - високопродуктивна 32-розрядна шина для підключення графічних плат. Застосовувалася замість шини PCI як більш швидка альтернатива. Зараз витіснена більш швидкою і універсальною шиною PCI-E. Максимальна пропускна здатність 2 Гбайт/с.
- Шина SCSI (*Small Computer System Interface*, інтерфейс малих комп'ютерних систем) - паралельний інтерфейс для підключення високопродуктивних накопичувачів. Застосовується в серверних системах. До одного порта може підключатись до 16 пристроїв. Максимальна пропускна здатність 320 Мбайт/с, довжина кабелю до 12м.
- Шина SAS (*Serial Attached SCSI*) - послідовна версія шини SCSI. Застосовується в серверних системах. До одного порта може підключатись до 4 пристроїв. Максимальна пропускна здатність 6 Гбіт/с.
- Шина ATA (*Advanced Technology Attachment*, підключення за передовою технологією) - паралельний інтерфейс для підключе-

ння накопичувачів. До недавнього часу був стандартом для ПК. Зараз витісняється шиною SATA. До одного порта може підключатись до 2 пристроїв. Максимальна пропускна здатність 133 Мбайт/с.

- Шина SATA (Serial ATA) - новий стандарт для підключення накопичувачів. На відміну від ATA використовує послідовну передачу. За рахунок зменшення кількості провідників і покращенню захисту кабелю, SATA дозволяє збільшити швидкість передавання даних до 300 Мбайт/с (версія SATA 2.0) і 600 Мбайт/с (версія SATA 3.0).
- Шина eSATA (*external SATA, зовнішній SATA*) - варіант інтерфейсу SATA, що дозволяє підключати зовнішні накопичувачі без розбирання корпусу комп'ютера.

2.4.5 Мережне обладнання

З точки зору мережних технологій комп'ютер є робочою станцією або хостом (*host*). Тобто це пристрій, що має унікальну адресу і приймає участь в мережному обміні даними. Для підключення комп'ютера до мережі використовують декілька технологій:

- мережні карти (*Network Card*) - плата розширення або вбудований в материнську плату інтерфейс, що дозволяє фізично під'єднатися до мережі. На даний момент в основному застосовуються мережні карти FastEthernet (100BASE-T, швидкість 100 Мбіт/с) і GigabitEthernet (1000BASE-T, швидкість 1 Гбіт/с). Довжина сегменту кабелю до 100 м.
- модеми (*MODdulator-DEModulator*) - пристрій для перетворення цифрового сигналу в аналоговий і навпаки для передачі інформації по аналогових лініях. Часто використовувалися для доступу до Internet за допомогою телефонної лінії. Максимальна швидкість 56 Кбіт/с. В останній час використовуються ADSL-модеми (*Asymmetric Digital Subscriber Line*, асиметрична абонентська цифрова лінія), що дозволяють отримувати швидкість до 24 Мбіт/с на прийом і до 1Мбіт/с на передачу.
- адаптери Wi-Fi (*Wireless Fidelity*) - адаптери бездротової комп'ютерної мережі. Дозволяють не прокладати виту пару для мережі FastEthernet в умовах невеликих помешкань і територій. Дозволяють отримати швидкість до 54Мбіт/с (стандарт 802.11g) і

480 Мбіт/с (стандарт 802.11n). Пристрої Wi-Fi можуть працювати в двох режимах: як точки доступу (*Access Point*) і в режимі точка-точка (*Ad-Hoc*). В першому режимі можна організувати обмін між декількома пристроями, а в другому - лише між двома. Обмін за допомогою цих пристроїв можливий на відстані до 45 м в приміщенні і до 100 м на відкритому просторі.

- адаптери BlueTooth - адаптери для мобільних пристроїв (телефонів, ноутбуків, бездротових гарнітур тощо), які дозволяють передавати інформацію з пристрою на пристрій на відстані до кілька десятків метрів.

Контрольні питання та завдання

1. Наведіть визначення комп'ютера (ЕОМ).
2. За якими ознаками можна класифікувати комп'ютери?
3. Що таке архітектура комп'ютера?
4. Які компоненти є складовими архітектури комп'ютера?
5. Наведіть основні функції арифметично-логічного пристрою (АЛП).
6. Наведіть основні функції оперативного запам'ятовувального пристрою (ОЗП).
7. Що таке зовнішня пам'ять?
8. Які пристрої можна віднести до пристроїв вводу-виводу?
9. Наведіть основні принципи побудови класичної архітектури комп'ютера (принципи фон Неймана).
10. Що таке магістрально-модульний принцип?
11. Наведіть основні конструктивні елементи сучасного комп'ютера.
12. Чи є системний блок обов'язковим елементом конструктивного виконання комп'ютера?

13. Скільки центральних процесорів може бути в системному блоці комп'ютера?
14. Що важливіше для продуктивності комп'ютера: тактова частота процесору чи кількість його ядер?
15. Розташуйте за швидкодією наступні компоненти: оптичний накопичувач, дисковод, жорсткий диск, флеш-накопичувач.
16. Який пристрій довше зберігає інформацію: жорсткий диск, дискета, компакт-диск, оперативна пам'ять?
17. Що таке надоперативна пам'ять (кеш)?
18. Чи може комп'ютер працювати без монітора?
19. Чи може комп'ютер працювати без зовнішнього живлення?
20. До якого порту (шини) підключається принтер?
21. Чи може монітор бути підключений до шини USB і чому?
22. Що таке плотер?
23. Чи може монітор виконувати ролі пристрою вводу і пристрою виводу одночасно?
24. Чи можна перенести великий об'єм інформації між комп'ютерами без мережного зв'язку і розбирання корпусу?
25. За який час можна заповнити жорсткий диск комп'ютера об'ємом 1Т-байт інформацією, що отримується по мережі зі швидкістю звичайного модему, ADSL модему, мережі FastEthernet?
26. Наведіть назву і опис технологій, за якими можна будувати комп'ютерні мережі масштабу кімнати, будинку, району.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНИХ ПРОЦЕСІВ

Поряд з апаратним забезпеченням комп'ютера існує друга сутність без якої неможливо виконати будь-який інформаційний процес. Ця сутність називається програмним забезпеченням (ПЗ). Вся сукупність програм, що існує зараз для безлічі діючих комп'ютерних систем, або тих, що вже не існують і тих, що тільки будуватимуться, виконує одну і ту саму задачу — обробку інформації за допомогою апаратного забезпечення, яке є в наявності. Тобто, будь-яка програма в кінцевому підсумку керує апаратним забезпеченням, навіть якщо вона на перший погляд і не взаємодіє з обладнанням. В свою чергу, програмне забезпечення не може існувати без апаратного забезпечення. Таким чином ці дві складові інформаційного процесу є нерозривними і невідокремлюваними.

3.1 ПОНЯТТЯ ПРОГРАМИ

Програмою називається певна послідовність окремих команд, що виконуються автоматично одна за одною для досягнення певного результату.

Майже завжди програма представляє собою виконуваний модуль, який зберігається на пристроях зовнішньої пам'яті в двійковому (бінарному, або скопійованому) вигляді. Для кожного типу виконуваного середовища (операційної системи, апаратної платформи) внутрішній формат програми може відрізнитися. Тому для різних середовищ одну і ту саму програму треба створювати окремо.

Перед виконанням програма завантажується в оперативну пам'ять і їй передається керування апаратним забезпеченням комп'ютера. Для створення програм використовуються спеціальні програмні системи, що називаються середовищами розробки (Developer Environment), що використовують як основу, одну або декілька алгоритмічних (тобто, заснованих на алгоритмах) мов програмування. Зараз найбільш поширені мови програмування C++, C#, Pascal, PHP, Python, Perl, Assembler. Метою будь-якої мови програмування є запис алгоритмів обробки інформації у вигляді, зрозумілому для машини.

Для спрощення і прискорення створення програм середовища розробки поєднують в собі текстовий редактор, модуль для розробки інтерфейсу кори-

стувача, модуль для перевірки і перекладу вихідного тексту програми в бінарний вид, модуль для налагодження (debugger). Ця об'єднана система називається інтегроване середовище розробки (IDE — integrated development environment).

Існує також різновид програм, що не потребують перекладу в бінарний вигляд. Це так звані інтерпретовані програми. Для їх виконання потрібне спеціальне середовище — інтерпретатор. Він читає вихідний код програми, перевіряє його на відсутність помилок і виконує операції, що записані в прочитаній частині програми. Цей вид програм виконується дещо повільніше, ніж скомпільований варіант, але простіший в підтримці і модифікації.

3.2 КЛАСИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Отже, програма є одним із важливих елементів в виконанні інформаційного процесу. І, хоча існує безліч різних програм для вирішення різноманітних задач, все програмне забезпечення можна поділити за призначенням на такі великі рівні:

- Базове ПЗ
- Системне ПЗ
- Службове ПЗ
- Прикладне ПЗ

Кожний рівень програм виконує певну функцію і пов'язаний з іншими видами програм за допомогою спеціальних *програмних інтерфейсів*. Ці інтерфейси допомагають програмі одного рівня користуватися функціональністю програми іншого рівня або надавати свої послуги іншим учасникам інформаційного процесу. Розглянемо більш детально функції кожного з цих рівнів програмного забезпечення.

Базове ПЗ. Це найбільш низький рівень програмного забезпечення. Він виконує функції керування і взаємодією з базовим (або стандартним) апаратним забезпеченням. Програми, що входять до цього рівня, поставляються виробниками апаратних засобів і зберігаються в постійному запам'ятовувальному пристрою - спеціальних мікросхемах ПЗУ. Ці програми зчитуються в оперативну пам'ять при включенні обчислювального пристрою, конфігурують параметри апаратних елементів, керують їх роботою і контролюють її на протязі всього часу роботи комп'ютера.

Якщо мова іде про комп'ютер, у першу чергу, до базового ПЗ відносять базову систему вводу-виводу BIOS (від Basic Input-Output System). Ця система поєднує в собі наступні елементи:

- Програму самотестування при включенні живлення (POST — Power-On Self Test).
- Програму для керування параметрами апаратних пристроїв (Setup)
- Програму для завантаження операційної системи.
- Програми для керування стандартними пристроями вводу-виводу.

Базове ПЗ дуже важливе для ефективності і надійності роботи всієї інформаційної системи, бо помилки і затримки в роботі цього рівня ПЗ катастрофічно відображаються на роботі інших рівнів ПЗ.

Системне ПЗ. Це більш високий, функціональний і складний рівень ПЗ. В першу чергу, до цього рівня відносять *операційні системи* і їх компоненти. Призначення цього рівня в зв'язку більш високих рівнів з базовим ПЗ і апаратним забезпеченням. Для обслуговування нестандартних пристроїв, що включаються до інформаційної системи, використовуються спеціальні програмні модулі, що називаються драйверами (drivers). Ці програми є специфічними для кожного пристрою і кожної операційної системи. Вони виконують посередницькі функції по керуванню пристроями, яких не може обслуговувати базовий рівень ПЗ. До цього рівня також відносять програмні компоненти, що реалізують інтерфейс користувача, тобто виконують взаємодію з користувачем за допомогою пристроїв вводу-виводу.

Якщо на комп'ютері встановлено системне ПЗ (операційна система), його можна використовувати для вирішення конкретних задач користувача.

Службове ПЗ. Головним призначенням програм цього рівня є організація більш зручного керування інформаційним процесом за допомогою автоматизації виконання робіт по налаштуванню, обслуговуванню та перевірці інформаційної системи. Цей вид ПЗ також називають утилітами (utility).

До службового ПО можна віднести:

- Оболонки (файлові менеджери, графічні оболонки).
- Антивірусні пакети.
- Архіватори.
- Пакетні фільтри (брандмауери).

- Дефрагментатори.
- Програми для захисту інформації від несанкційованого доступу та ін.

Утиліти можуть як тісно інтегруватися з ядром операційної системи (наприклад, як в MS Windows), або бути відносно незалежними і встановлюватися окремо за бажанням користувача (як в Unix і Linux).

Прикладне ПЗ. Цей рівень ПЗ призначений для виконання безпосередньо завдань користувача. Для ефективної і надійної роботи цього ПЗ і створюється інформаційна система.

Зазвичай, прикладне ПЗ оформлюється в так звані *пакети прикладних програм*. Це сукупність програмних засобів, що використовуються для вирішення завдань певного типу (або, предметної області). До цього типу ПЗ можна віднести наступні пакети програм:

- Офісні пакети (MS Office, Star Office, Open Office, Koffice, Lotus Notes)
- Видавничі пакети (DTP — Desktop Publishing, QuarkXPress, Adobe InDesign, Microsoft Publisher)
- Пакети автоматизованого проектування (CAD - Computer-Aided Design, AutoCAD, PCAD, ArchiCAD)
- Математичні пакети (MathCAD, MatLab)
- Фінансові пакети (1С:Бухгалтерія)
- Растрові графічні редактори (Adobe Photoshop)
- Векторні графічні редактори (Corel Draw, Microsoft Visio)
- Пакети для 3D моделювання (Maya 3D, 3DMax)
- Мультимедійні пакети та ін.

3.3 ОПЕРАЦІЙНІ СИСТЕМИ

Операційні системи мають великий вплив на проведення інформаційного процесу, визначають загальну ефективність системи, її функціональність та можливості розширення і масштабування. Тому необхідно детально розглянути цей вид ПЗ, його типи, функції і засоби роботи з ним.

3.3.1 Поняття ОС, основні функції

Операційною системою (ОС) *називається програмний комплекс, що організує активне використання апаратних ресурсів інформаційної системи і створює зручний програмний і користувальницький інтерфейс.*

Операційні системи виникли завдяки ускладненню прикладного ПЗ і апаратного забезпечення, на якому воно виконувалося. До виникнення ОС кожна програма повинна була самостійно керувати усіма потрібними апаратними засобами комп'ютера, створювати інтерфейс користувача і ще виконувати певну прикладну функцію. Якщо таких програм було декілька, частини, що не відносилися до прикладних функцій у цих програмах часто повторювалися. Ці програми було складно модифікувати і переносити на інші платформи. Тому ці функції винесли до окремої програми і створили спеціальний програмний інтерфейс, яким треба було користатися прикладним програмам.

Ці перетворення, а також використання таких ідей, як пакетний режим обробки інформації, розподілення часу, багатозадачність, розподілення повноважень і файлові системи дозволили створити сучасні операційні системи.

До базових функцій операційних систем можна віднести

- завантаження і виконання прикладних і службових програм;
- організація операцій вводу-виводу;
- керування розподіленням оперативної пам'яті;
- підтримка нестандартних пристроїв;
- зберігання інформації на енергонезалежних накопичувачах.

3.3.2 Класифікація ОС

Як і комп'ютери, операційні системи використовуються в різноманітних умовах і вимогах. Тому існує велика кількість різноманітних операційних систем, що відрізняються своїми властивостями і функціональністю.

За призначенням ОС можна поділити на

- універсальні,
- мережні,

- реального часу,
- для вбудованих рішень.

За кількістю задач, що виконуються одночасно розрізняють

- однозадачні,
- багатозадачні.

За кількістю одночасно працюючих користувачів

- з одним користувачем,
- з багатьма користувачами.

За типом інтерфейсу.

- з текстовим інтерфейсом,
- з графічним інтерфейсом

3.3.3 Сучасні ОС

Для сучасних ОС притаманна висока розповсюдженість і різноманітність. Вони використовуються на великих і потужних комп'ютерах, на персональних ПК, на мобільних телефонах, в побутовій техніці, виробничих і військових системах. Тобто там, де необхідно виконувати більше ніж одну програму одночасно або розподілено в часі. Існує багато сотень різних ОС, але найбільш уживаній на персональних комп'ютерах, з якими найчастіше стикаються звичайні користувачі, не більше десяти. Для стаціонарних комп'ютерів це системи Ms DOS, Ms Windows, UNIX, GNU/Linux, MacOS X. Для мобільних рішень використовуються більш різноманітні версії систем Ms Windows Mobile, Android, Symbian OS, Mobilin. Майже завжди мобільні версії мають модифіковане ядро від настільної системи і специфічний набір утиліт. Наприклад, ОС Android складається з ядра ОС Linux і великої кількості програм для комунікації за допомогою технології мобільних систем.

Наведемо огляд декілька найбільш розповсюджених і вживаних систем.

ОС Ms-DOS (англ. *Microsoft Disk Operating System*) — перша операційна система компанії Microsoft для персональних IBM-PC сумісних комп'ютерів. Перша версія (1.0) з'явилася в 1981. MS-DOS розвивалася і підтримувалася до 2000 року і надала компанії Microsoft можливість стати монополістом в секторі створення і продажу ОС для масових ПК (біля 90 відсотків ринку). До

цього часу існує багато систем, де використовується ця ОС — від банківського сектора і промислових комп'ютерів до вбудованих рішень і систем зв'язку.

Ms-DOS — це комерційна, однозадачна операційна система з базовим інтерфейсом командного рядка. Для розширення можливостей системи використовувалося декілька так званих *оболонки* — систем інтерфейсу користувача з можливістю більш простого керування інформацією на комп'ютері. Це системи Norton Commander, Volkov Commander, DOS Shell, DOS Navigator. Функціональні можливості і засоби роботи з цими програмами були схожі і вважалися майже стандартом. За основу в цих системах був взятий інтерфейс з двох панелей, на яких виводилася інформація про файли і каталоги, що знаходяться на комп'ютері. Керування виконувалося за допомогою комбінацій клавіш (в основному — функціональних F1-F10). Пізніше, цей інтерфейс знайшов продовження в таких популярних файлових менеджерах, як Far Manager і Windows Commander. Для Unix і Linux існує аналог цього ПЗ, який називається Midnight Commander. Взагалі, це дуже вдалий і зручний вид інтерфейсу як для текстового, так і для графічного режиму.

ОС Ms Windows — операційна система компанії Microsoft, що базується на графічному інтерфейсі користувача і є найрозповсюдженішою і найживішою системою на персональних комп'ютерах. Перша версія з'явилася в 1985 році і виконувала роль оболонки для ОС Ms DOS.

Подальші версії, починаючи з Windows 95, були вже більш повноцінними операційними системами. Вони були спроможні на запуск декілька застосунків одночасно, мали приємний і зручний для користувача графічний інтерфейс. У свій час ОС Windows представляли собою взірці впровадження новітніх технологій в спілкуванні користувача з комп'ютером і прикладної програми з операційною системою, апаратним забезпеченням.

Існує дві гілки в розробці систем сімейства Windows — професійна і для звичайних користувачів. Професійна гілка почалася з розробки ОС Windows NT (NetWork — мережа) і продовжується зараз продуктами з префіксом Server — Windows Server 2000, 2003, 2008. Ці системи орієнтовані на корпоративний сегмент і для використання в складі серверних систем з високими потребами в стабільності і продуктивності.

Інша, десктопна, гілка представлена більш простими і дешевими системами - Windows 95, 98, Me, 2000, XP, Vista, Seven. Починаючи з версій Windows 2000 в десктопних версіях використовується модифіковане ядро ОС Windows NT, що дозволило підвищити стабільність системи.

Сучасні версії ОС Windows є повноцінними ОС, які мають усі необхідні функціональні можливості. Особливістю цих систем є глибока інтеграція слу-

жбових і системних програм до ядра системи, що збільшує швидкодію, але зменшує надійність і універсальність.

Отже, сучасна ОС сімейства Windows — комерційна, багатозадачна, універсальна ОС з графічним інтерфейсом і розвиненими можливостями для виконання різноманітних прикладних програмних пакетів.

UNIX. Це багатозадачна система, що може обслуговувати багато користувачів одночасно. Вона була створена в 1969 році в дослідницькій лабораторії AT&T Bell Labs двома дослідниками — Кеном Томсоном і Денісом Рітчі. Основні ідеї, на основі яких було створено цю систему були дуже прості і базувалися всього на двох об'єктах — процес і файл. Процес виконував будь-яку корисну роботу, а файл надавав можливість зберігати інформацію і спілкуватися з різноманітними пристроями. В цих системах також використовується велика кількість спеціалізованих програм, кожна із яких виконує лише одну функцію, але виконує її максимально якісно і швидко. З таких програм збираються спеціальні конструкції — *конвейери*. В конвейерах програми обмінюються обробленою інформацією так, що вихідна інформація з однієї програми подається на вхід другої програми і так до кінця конвейера.

Існує багато Unix- подібних ОС. Деякі з них є комерційними (Sun Solaris, HP UX, IBM Irix), деякі є відкритими і розповсюджуються за різноманітними вільними ліцензіями (FreeBSD, OpenBSD, NetBSD, OpenSolaris). Всі ці системи відрізняються великою стабільністю, якісним захистом і розвинутими можливостями по масштабуванню і модифікації. UNIX системи ж основою глобальної мережі Інтернет, більшості корпоративних і територіально розподілених мереж, використовуються в ролі сервера і робочої станції.

Хоча графічний інтерфейс і не є обов'язковим елементом цих ОС, в останні роки він усе частіше використовується на робочих станціях з встановленою ОС UNIX.

GNU/Linux. Це сучасна популярна операційна система, виконана на основі ядра Linux і набору утиліт і системних компонентів, що розповсюджуються на основі вільної ліцензії GNU (GNU Is Not UNIX). Ядро Linux зробив в 1991 році студент Гельсінського університету Лінус Торвальдс. За основу він узяв принципи, що використовувалися в системах UNIX, але орієнтувався на персональні комп'ютери. Ядро комплектується великою кількістю утиліт, що разом складають дистрибутив — закінчену операційну систему з певними функціональними відмінностями.

Існує велика кількість дистрибутивів. Деякі з них розповсюджуються на комерційній основі (оплачується не сам код системи, а технічна підтримка), але більшість повністю безплатна.

Наведемо список найпопулярніших дистрибутивів Linux.

- *RedHat Enterprise Linux (RHEL)* — комерційний дистрибутив для корпоративного сектору створений компанією Red Hat.
- *Fedora Core* — вільний дистрибутив на основі RedHat Enterprise Linux.
- *SuSe Linux* — комерційний дистрибутив для корпоративного сектору, створений компанією Nowell.
- *OpenSuSe Linux* вільний дистрибутив на основі SuSe Linux
- *Ubuntu Linux* - вільний дистрибутив, створений компанією Canonical для звичайних користувачів.
- *Gentoo Linux* - вільний дистрибутив для ентузіастів.
- *Slakware Linux* - вільний дистрибутив для ентузіастів.

ОС Linux поєднують в собі надійність і гнучкість операційних систем UNIX з якісною підтримкою апаратної частини комп'ютера, яка притаманна системам на базі ОС Windows.

Слід зауважити, що для всіх сучасних дистрибутивів Linux також притаманне використання графічного інтерфейсу поряд з інтерфейсом командного рядка, що підходить як для починаючих користувачів, так і для професіоналів.

3.4 ТЕХНОЛОГІЇ ЗБЕРІГАННЯ ДАНИХ

Технологія зберігання даних має декілька рівнів:

- апаратне забезпечення;
- логічна структура розташування даних;
- прикладний рівень забезпечення збереження даних.

До апаратного рівня зберігання можна віднести:

- контролери накопичувачів і накопичувачі, виконані за технологією ATA, SATA, SCSI, SAS;
- пристрої мережного зберігання даних (NAS, SAN);

- пристрої і протоколи для сполучення мережних масивів з комп'ютером (Infiniband, FibreChannel, iSCSI).

Логічний рівень включає в себе:

- RAID-масиви;
- менеджери томів;
- файлові системи.

Прикладний рівень включає програмне забезпечення, що дозволяє оперувати даними на рівні файлової системи, керувати іншими рівнями збереження даних а також виконувати високорівневі сервісні функції по підтримці даних в несуперечній формі.

Накопичувачі описано в пп. 2.4.1

Розглянемо стисло інші елементи апаратного рівня.

NAS (*Network Attached Storage*, мережне сховище) - комп'ютер або пристрій, що містить певний дисковий масив і підключається до мережі. З комп'ютера користувача доступ до NAS можна отримати по мережних протоколах (FTP, HTTP, NFS, CIFS (SMB)). Швидкість обмежується швидкістю мережі, до якої підключений NAS і ПК користувача.

SAN (*Storage Area Network*, мережа зберігання даних) - сукупність апаратних, програмних засобів і протоколів, що дозволяє підключати зовнішні пристрої для збереження даних до серверних систем. З точки зору серверу масив SAN виглядає як локальний накопичувач. Доступ до масиву SAN для звичайних користувачів виконується за допомогою програмних засобів сервера.

Infiniband - швидкісна послідовна шина для передачі даних. Максимальна швидкість 96Гбіт/с.

Fibre Channel - сімейство швидкісних протоколів для передачі даних на великі відстані (до 50км). Максимальна швидкість 10 Гбіт/с.

iSCSI (*Internet SCSI*) - протокол для підключення систем зберігання даних через TCP/IP.

Логічний рівень забезпечує цілісність даних, що можуть перебувати на

різноманітних пристроях, підключених за різними протоколами тощо. Також в його задачу входить створення відказостійких систем, систем балансування навантаження і забезпечення якості сервісу доступу до даних (*Quality of Service, QoS*).

Як правило, велика частина логічного рівня реалізується в ядрі операційної системи і драйверах пристроїв зберігання даних. Деякі частини можуть мати апаратну реалізацію для збільшення швидкості і надійності (наприклад, RAID-масиви). На рівні ОС реалізується також і підтримка файлової системи.

RAID (*Redundant Array of Independent/Inexpensive Disks*, надлишковий масив незалежних/недорогих дисків) — це набір дискових пристроїв, що працюють разом, щоб підвищити швидкість і/або надійність системи вводу/виводу. Цим набором пристроїв управляє спеціальний RAID-контролер (або частина операційної системи), який забезпечує функції розміщення даних по масиву. Для решти системи масив представляється як один логічний пристрій вводу/виводу. За рахунок паралельного виконання операцій читання і запису на кількох дисках, масив забезпечує підвищену швидкість обміну в порівнянні з одним диском.

Масиви також можуть забезпечувати надмірне зберігання даних, з тим, щоб дані не були втрачені у разі виходу з ладу одного з дисків. Залежно від типу RAID, проводиться або дзеркалювання або розподіл даних по дисках.

Менеджер томів (англ. Logical Volume Manager, LVM) - проміжний компонент системи зберігання даних між фізичним носієм і файловою системою. Використовується в операційних системах Linux і OS/2. Дозволяє розбивати фізичний носій (або декілька носіїв) на блоки розміром від 4 до 32 Мбайт. Далі ці блоки можуть надаватися файловій системі як єдиний фізичний носій (група томів). Цей носій можна відформатувати і використовувати за призначенням. LVM дозволяє розподіляти довільним образом інформацію між всіма фізичними носіями, що знаходяться під керуванням менеджера томів.

Розглянемо більш детально поняття і основні функції файлової системи.

Визначення. *Файловою системою (ФС, file system) називається сукупність домовленостей про засоби організації, зберігання, іменування даних на носіях інформації. Файлова система визначає формат запису інформації на носії, правила доступу до неї.*

Кожна конкретна операційна система може підтримувати декілька різних файлових систем із різною ефективністю праці з ними.

Основними функціями файлової системи є:

- іменування об'єктів файлової системи;
- підтримка програмного інтерфейсу для доступу до інформації;
- відображення об'єктів файлової системи на фізичні носії;
- зберігання додаткової метайнформації для об'єктів файлової системи;
- забезпечення виправлення помилок при зберіганні даних;
- розмежування доступу до об'єктів файлової системи для різних категорій користувачів і процесів та ін.

Для кожної файлової системи визначається набір об'єктів та взаємозв'язок їх в середині файлової системи. Найчастіше для універсальних файлових систем застосовують такі об'єкти: розділи або томи, каталоги, файли, метадані.

Розділ (*partition*) - частина носія, що об'єднує сусідні області і, як правило, має окрему файлову систему. Операція розбиття на розділи виконується для всіх нових носіїв (жорстких дисків, флеш-дисків). Для дискет і оптичних носіїв ця операція не застосовується (тобто для них використовується один розділ). На сучасних комп'ютерах застосовуються дві системи розбиття на розділи:

- MBR (*Master Boot Record*, головний завантажувальний запис) - традиційна система розподілення розділів для IBM-PC сумісних комп'ютерів. Дозволяє мати на одному носії до 4 (первинних) розділів, один із котрих може бути розширеним. Розширений розділ може складатися з будь якої кількості логічних розділів (залежить від операційної системи). Таблиця існуючих розділів і головний завантажувальний запис знаходяться на самому початку носія. Це положення важливе для процесу завантаження комп'ютера, бо саме там BIOS шукає завантажувач операційної системи.
- GPT (*GUID Partition Table*) - нова система розподілення розділів.

Дозволяє мати на одному носії будь-яку кількість розділів. Підтримується всіма сучасними ОС.

Файл (*file*) - поіменована сукупність даних на носії. Основний параметр файлу - розмір. Він визначається як об'єм даних, що містяться в ньому. Для забезпечення систематизації зберігання файлів, їх об'єднують за якоюсь ознакою в каталоги. Слід зауважити, що поняття файлу для ОС UNIX/Linux є дещо ширшим. Для них файл - це будь-який пристрій (наприклад, дисковод), ресурс (наприклад, область пам'яті чи мережний ресурс) або звичайний файл в зовнішній пам'яті.

Каталог (директорія, тека) - спеціальний вид файлу, що містить інформацію про файли і підкаталоги, розташовані в ньому.

Файли і каталоги утворюють деревоподібну структуру з коренем, що називається кореневим каталогом (або просто коренем файлової системи). Вузлами в цьому дереві є каталоги і підкаталоги, а листям — файли. Метадані (*metadata*) - інформація, що забезпечує можливість впорядкованого зберігання корисних даних в файловій системі. Метадані ніяк не пов'язані зі змістом даних. До метаданих можна віднести

- ім'я файлу і каталогу, дата створення, останньої зміни, останнього доступу до файлу і каталогу;
- права доступу до файлу і каталогу (на читання, зміну, запуск, пошук тощо);
- розмір файлу, його стан (відкритий, закритий) редагується);
- інформація про квоти для окремих користувачів і процесів;
- журнали виконуваних операцій тощо.

Частина метаданих доступна (наприклад, імена файлів), а частина (наприклад журнали операцій) не доступна для користувача.

3.5 СУЧАСНІ ФАЙЛОВІ СИСТЕМИ

Розглянемо характеристики деяких сучасних файлових систем.

3.5.1.1 Файлові системи для носіїв з довільним доступом.

FAT16, FAT32 (*File Allocation Table-16/32bit*) - файлові системи ОС MS-DOS/MS Windows 9x. Дуже поширені і, до недавнього часу, майже єдині ФС для ПК. Зараз використовуються в більшості змінних носіїв на основі флеш-пам'яті.

Переваги:

- простота,
- розповсюдженість.

Недоліки:

- невелика швидкість,
- відсутність відновлення даних,
- розмежування доступу, квот.

Максимальний розмір файлу 2 Гбайт.
Максимальний розмір тому (розділу) 4 Тбайт.

NTFS - (NT File System) - сучасна файлова система для ОС MS Windows. Максимальний розмір файлу 16 ексабайт (1 ексабайт= 10^{18} байт). Максимальний розмір тому (розділу) 16 ексабайт.

EXT2, EXT3 — традиційні файлові системи ОС Linux. Мають високу продуктивність і засоби виправлення помилок. Максимальний розмір файлу 2 Тбайт. Максимальний розмір тому (розділу) 32 Тбайт.

EXT4 - сучасна журнальована файлова система ОС Linux. Максимальний розмір файлу 16 Тбайт. Максимальний розмір тому (розділу) 1000000 Тбайт.

UFS, UFS2 - файлова система ОС FreeBSD, OpenBSD, NetBSD. Відрізняється надійністю. Максимальний розмір файлу 8 зетабайт (1 зетабайт= 10^{21} байт). Максимальний розмір тому (розділу) 8 зетабайт.

ZFS - новітня файлова система ОС Sun Solaris і ОС FreeBSD. Включає в себе файлову систему і менеджер логічних дисків. Максимальний розмір файлу 16 ексабайт. Максимальний розмір тому (розділу) 16 ексабайт.

HFS+ - файлова система ОС Mac OS X (Apple). Максимальний розмір файлу 16 ексабайт. Максимальний розмір тому (розділу) 16 ексабайт.

3.5.1.2. Файлові системи для оптичних носіїв.

ISO9660 - файлова система для CD-ROM. Максимальний розмір файлу 2 Гбайт.

UDE (*Universal Disk Format*) - універсальна файлова система для оптичних носіїв. Підтримується більшістю операційних систем. Приходить на заміну ISO9660. Максимальний розмір файлу 16 ексабайт.

3.5.1.3. Файлові системи для розподілених застосувань.

SMB (*Server Message Block*) - стандарт для доступу до віддалених ресурсів в мережі ОС Windows.

NFS (*Network File System*, мережна файлова система) - стандартна розподілена система ОС UNIX/Linux.

Контрольні питання та завдання

1. Що таке програма?
2. Які рівні програмного забезпечення визнаєте?
3. Як між собою пов'язані базове ПЗ та системне ПЗ?
4. Що таке утиліта?
5. До якої з категорій (рівнів) ПЗ відносяться драйвери?
6. До якої категорії (рівня) ПЗ відноситься офісний пакет?
7. Наведіть визначення операційної системи.
8. Які основні функції виконує операційна система?
9. За якими ознаками можна класифікувати операційні системи?
10. Які сучасні ОС ви знаєте? Наведіть коротку характеристику кожної згаданої системи.

11. Чи є сумісними на рівні прикладного ПЗ системи UNIX та Windows?
12. Чи можна виконати програму Windows в операційній системі Linux? А навпаки?
13. Чим схожі і чим відрізняються системи сімейства Windows і UNIX/Linux?
14. Які рівні можна виділити в сучасних технологіях зберігання даних?
15. Що таке файлова система?
16. Які основні функції вона виконує?
17. Які основні об'єкти можна виділити в структурі файлової системи?
18. Чи може файлова система зберігати нескінчену кількість інформації і чому?
19. Скільки фільмів у форматі DVD5 (об'єм 4.7Гбайт) вміститься в файловій системі NTFS? Скільки важитиме така кількість DVD дисків, якщо кожний диск важить приблизно 16 грамів?

4 ОСНОВИ МЕРЕЖНИХ ТЕХНОЛОГІЙ

4.1 ВИЗНАЧЕННЯ КОМП'ЮТЕРНОЇ МЕРЕЖІ

Комп'ютерною мережею називається сукупність програмних і апаратних засобів, які застосовуються для транспортування інформації між двома або декілька інформаційними системами за допомогою різних фізичних середовищ.

Таким чином, комп'ютерна мережа зв'язує в єдине ціле розподілені територіально вузли обчислювальних ресурсів для транспортування інформації, об'єднання обчислювальних потужностей, керування окремими вузлами та для інших потреб.

4.1.1 Класифікація комп'ютерних мереж

Зараз комп'ютерні мережі є дуже різноманітними за призначенням, топологією, розмірами та іншими властивостями.

За територіальним принципом мережі поділяють на

- персональні (PAN, Personal Area Network),
- локальні (LAN, Local Area Network),
- місцеві (MAN, Metropolitan Area Network),
- національні (NAN, National Area Network),
- глобальні (Wide Area Network).

За типом взаємодії між об'єктами мережі поділяють на

- клієнт-серверні,
- змішані,
- однорангові,
- багаторангові.

За топологією (просторовим розташуванням) мережі бувають

- загальна шина (Ethernet),
- зірка (FastEthernet),
- кільце (TokenRing),
- решітка,
- змішана,
- повнозв'язана.

За операційними системами, що встановлені на комп'ютерах, що підключені до мережі поділяють на:

- гомогенні,
- гетерогенні.

4.1.2 Основні терміни сучасних мережних технологій

Розглянемо деякі найбільш вживані терміни мережних технологій

Домена адреса — зрозуміла для людини форма запису IP-адреси виду піддомен3.піддомен2.піддомен1.

IP адреса — унікальний чотирьохбайтний номер виду 111.222.333.444, що однозначно ідентифікує хост в мережі.

Комутатор (концентратор) — пристрій мережі Ethernet, що поєднує мережні об'єкти в єдине ціле за допомогою середовища передачі даних.

Маршрутизатор — комп'ютер або пристрій, що вирішує, яким маршрутом із наявних треба передавати інформацію від одного вузла мережі до іншого.

Мережна карта — карта розширення, що входить до складу комп'ютера, і призначена до під'єднання його до певного типу мереж — провідних, безпроводних, мобільних та ін.

Міст (гейт) - комп'ютер або пристрій, що під'єднує одну або декілька мереж в одне ціле.

Модем — пристрій для перекодування інформації з цифрового в аналоговий і в зворотньому напрямку.

Пакетний фільтр - програма, що виконує фільтрацію проходячих по мережі пакетів для запобігання несанкційованого доступу до мережних ресурсів

Провайдер — установа, що надає можливість користувачам під'єднуватись до глобальної мережі за допомогою однієї з розповсюджених мережних технологій — комутованого доступу, ADSL, витой пари, оптичної лінії, бездротового зв'язку та ін.

Робоча станція — окремий комп'ютер, під'єднаний до мережі і виконуючий певні завдання користувача.

Сервер — комп'ютер, що надає доступ до своїх інформаційних, обчислювальних, транспортних або інших видів ресурсів для об'єктів мережі.

Точка доступу — пристрій або комп'ютер, що має безпроводну мережну карту і призначений для надання доступу до провідного сегменту мережі для безпроводних пристроїв.

Хост - це будь-який мережний пристрій, що має окрему унікальну адресу і бере участь в мережному обміні інформацією.

Шлюз - комп'ютер або пристрій, що під'єднує локальну мережу до глобальної.

4.1.3 Локальні мережі

Локальні мережі є основою сучасного інформаційного середовища. Вони характеризуються використанням швидкісних ліній передачі даних (до десятків Гбіт/с), невеликою кількістю під'єднаних пристроїв (сотні і тисячі).

Найчастіше локальні мережі використовують технологію Ethernet для створення середовища передачі даних. Фізичним середовищем може слугувати вита пара, оптичне волокно, бездротова технологія. Територіально локальна мережа може об'єднувати від невеликої кімнати до цілого району міста. Для отримання доступу до мережі Інтернет в локальних мережах використовують шлюзи на основі апаратних пристроїв або виділених комп'ютерів.

4.1.4 Глобальна мережа Інтернет

Інтернетом називають сукупність апаратного і програмного забезпечення, протоколи передавання інформації і керування, що призначені для обміну інформацією в світовому масштабі.

Сучасний інтернет для користувача представляє собою глобальний інформаційний простір, сукупність різноманітних сервісів, що надають можливість вести пошук інформації, отримувати її на своєму комп'ютері, спілкуватися з іншими користувачами, ділитися своїми даними і інформацією про себе. Інтернет на сучасному етапі розвитку є рушійною силою багатьох процесів інформатизації суспільства і є зручним середовищем для переходу від маніпулювання матеріальними ресурсами до маніпулювання інформацією.

Виникнення Інтернет пов'язане з проектом дослідницької структури міністерства оборони США DARPA (англ. *Defense Advanced Research Projects Agency*, агенція з дослідження передових оборонних проектів), який повинен був дослідити можливість створення розподіленої комп'ютерної мережі, що могла працювати в умовах глобального ядерного конфлікту. В 1969 році мережа **ARPANET** об'єднала чотири університети, що займалися розробкою цієї теми — Каліфорнійський, Стенфордський, Університет штату Юта і Університет штату Каліфорнія в Санта-Барбарі.

29 жовтня 1969 року по цій мережі вдалося передати перше повідомлення, тому цю дату можна вважати **днем народження Інтернет**. Принцип побудови Інтернет заснований на децентралізації керування мережею і передаванні повідомлень у вигляді окремих пакетів. Якщо навіть більша частина інфраструктури цієї мережі буде виведена з ладу, передача даних буде виконуватися по тих каналах, що залишилися і система в цілому буде працювати, але, дещо повільніше.

В 70-х роках 20 століття продовжувалося вдосконалення і розширення цієї мережі. Але в 1984 році в ARPANET з'явився конкурент — наукова мережа NSFNet (англ. *National Science Foundation Network*), що об'єднала провідні наукові заклади США і інших країн. Починаючи з 1990 року назва Інтернет перейшла до цієї мережі, а мережа ARPANET перестала існувати. Пізніше, в 1995 році, керування маршрутизацією перейшло до комерційних структур — провайдерів.

Іншою важливою ступенню в розвитку Інтернет стало створення в 1988 році в ЦЕРНі (Європейська рада з ядерних досліджень) концепції всесвітньо-

го павутиння — World Wide Web. Цю концепцію запропонував Тім Беренс-Лі. Він також розробив протокол HTTP, мову HTML і адреси URL.

В наступні роки Інтернет об'єднав великі і маленькі мережі і зараз представляє собою загальний суспільний простір, що поєднує близько двох мільярдів користувачів і сотні мільйонів комп'ютерів.

4.2 ОСНОВНІ МЕРЕЖНІ СЕРВІСИ І ПРОТОКОЛИ

Протоколом називається сукупність домовленостей про засоби транспортування і інтерпретування інформації, що передається між двома об'єктами інформаційного процесу.

Розглянемо найбільш поширені і вживані протоколи прикладного рівня сучасних комп'ютерних мереж.

WWW (Workld Wide Web — всесвітнє павутиння). Це сукупність ресурсів в мережі, що зберігають і надають інформацію за допомогою мови гіпертекстової розмітки - *HTML (HyperText Markup Language)*. Технологія гіпертекстової розмітки припускає збереження корисної інформації у вигляді документів, що пов'язані гіпертекстовими посиланнями. За такими гіпертекстовими посиланнями можна пересуватися від одного документу до іншого пов'язаного з першим за змістом в межах одного ресурсу або між ресурсами. Крім текстової інформації в таких документах можна зберігати посилання на інші види інформації — графіку, аудіо та відеофайли, інтерактивні застосування та ін. Посилання представляють собою так званий *URL (Universal Resource Locator)* — універсальний локатор ресурсу. Це стандартизований спосіб запису розташування ресурсу в Інтернет. Для протоколу HTTP зазвичай він має наступний вигляд <http://www.google.com/>.

Документи зберігаються і обробляються для видачі користувачеві спеціальними серверами, що називаються Web-серверами. Для транспортування HTML документів використовується спеціальний протокол, що називається HTTP (HyperText Transfer Protocol). Клієнтське програмне забезпечення для цього виду ресурсів називається Web-броузером або просто броузером. Найбільш вживані броузери на сьогоднішній момент - Internet Explorer, Mozilla Firefox, Google Chrome, Apple Safari, Opera.

В Інтернет зараз існує велика кількість веб-ресурсів, і знайти потрібну інформацію буває досить складно. Для цього використовують так звані пошукові системи. Найбільша з них — Google. Сервери цих систем періодично

індексують доступні ресурси за багатьма ознаками і видають на запит користувача найбільш схожі.

E-Mail (Електрона пошта). Цей вид зв'язку був історично найпершим в Інтернет. Він дозволяє передавати текстові повідомлення адресату за допомогою системи поштових серверів. Ці сервери можуть працювати по двох протоколах:

- SMTP (Simple Mail Transfer Protocol) для передавання і зберігання вихідного листа.
- POP (Post Office Protocol) для прийому вхідного листа.

Для початку листування користувач повинен зареєструвати поштову скриньку на одному із існуючих серверів в Інтернет. При цьому користувачеві видається поштова адреса виду

`ім'я_користувача@доменне_ім'я_поштового_серверу`

Далі користувач за допомогою спеціального програмного забезпечення — поштового клієнта, створює листа і відсилає його на SMTP сервер, який за адресою одержувача вирішує, куди далі перенаправити цей лист.

Коли лист дістається до сервера одержувача (це також SMTP сервер), він перевіряється на відсутність вірусів і належність до *спаму* і, якщо ця перевірка пройдена, зберігається в поштовій скрині користувача. Одержувач за допомогою свого поштового клієнта перевіряє свою поштову скриньку і отримує всі останні надходження.

В останній час важливого значення набула проблема так званого спаму (SPiced hAM — гостра шинка), тобто небажаних поштових повідомлень рекламного характеру. За останніми дослідженнями спам досягає понад 90 відсотків всього поштового трафіку в Інтернет.

Найбільш часто вживане програмне забезпечення для обміну поштовими повідомленнями — поштовий клієнт та веб-браузер.

FTP (File Transfer Protocol). Це протокол передавання файлів. На відміну від HTTP протоколу він орієнтований на транспортування великої кількості інформації у вигляді окремих файлів. Файли зберігаються на спеціальних FTP серверах. Доступ до файлів відбувається за адресами у вигляді URL з вказанням протоколу :

`ftp://ftpl.freebsd.org/pub/FreeBSD/ISO-IMAGES-
i386/8.0/8.0-RELEASE-i386-disc1.iso.`

Для роботи з ftp серверами потрібно мати спеціальний ftp-клієнт або звичайний веб-браузер. Слід зауважити, що браузер може тільки викачувати файли з сервера, а для передачі на сервер потрібен окремий клієнт.

В останній час цей протокол витісняється протоколами на основі розподілених мережних технологій, наприклад протоколом **BitTorrent**.

4.3 БЕЗПЕКА ПРИ РОБОТІ В КОМП'ЮТЕРНИХ МЕРЕЖАХ

Для безпечної роботи в комп'ютерних мережах слід дотримуватися наступних правил поведінки:

1. Використовувати тільки ліцензійне або вільно-розповсюджуване програмне забезпечення і одержувати його із надійних джерел.
2. Встановлювати останні виправлення безпеки для своєї операційної системи.
3. Використовувати пакетні фільтри для запобігання атак на ваш комп'ютер.
4. Перевіряти всю отриману з мережі інформацію на наявність шкідливих додатків — вірусів, троянських коней, бекдорів та ін.
5. Періодично перевіряти інформацію, що зберігається на комп'ютері за допомогою антивіруса.
6. Використовувати для приватних облікових записів криптистійкі паролі.
7. Зберігати в таємниці параметри своїх приватних облікових записів.
8. Не відвідувати сайти з сумнівним змістом.
9. Виконувати періодичне резервне копіювання важливої інформації.

Контрольні питання та завдання

1. Що таке комп'ютерна мережа?
2. Як можна класифікувати мережі?
3. Що таке доменна адреса?
4. Що таке IP-адреса?
5. Чи є хостом робоча станція, сервер, веб-камера з мережним інтерфейсом?
6. Як, зазвичай, робочі станції можуть підключатися до комп'ютерної мережі?
7. Як називається пристрій, що поєднує локальну і глобальну мережі?
8. Що таке брандмауер (або пакетний фільтр)?
9. Що таке Інтернет?
10. Коли була створена мережа Інтернет?
11. Який протокол в Інтернет використовується в якості основного?
12. За яким протоколом працюють веб-сервери та веб-браузери?
13. Що можна передавати за допомогою електронної пошти?
14. Чим відрізняються WWW-сервер і FTP-сервер?
15. Яким чином можна знайти потрібні ресурси в Інтернет?
16. Наведіть декілька правил безпеки при роботі в комп'ютерній мережі.

5 ОФІСНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Обчислювальна техніка в наш час набула широкого розповсюдження. Вже давно комп'ютер не є лише приладом для обчислень. І, навіть більше того, найчастіше він використовується для обробки не числової, а текстової, графічної, або аудіо- та відеоінформації. Користувач, який зараз працює з комп'ютером, може не мати спеціальної технічної освіти, і взагалі бути неосвіченим. Достатньо лише навчити його користуватися стандартними засобами вводу-виводу і показати декілька прийомів роботи з графічним інтерфейсом. Велику роль в цьому процесі відіграє саме прикладне програмне забезпечення.

Найчастіше комп'ютер не як обчислювач використовують для обробки текстової інформації в процесі діловодства у сфері бізнесу, державного керування, приватних господарств. В цих випадках використовують програмне забезпечення, яке називають офісним.

5.1 ПРИЗНАЧЕННЯ І ФУНКЦІЇ ОФІСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для використання офісного програмного забезпечення існує безліч ситуацій — від створення невеличких оголошень і приватних записок і до створення великих документів з різноманітними додатками, таблицями, діаграмами, елементами презентацій та баз даних. Але, основні випадки застосування офісного ПЗ можна сформулювати таким чином.

Офісне ПЗ — це пакет прикладного програмного забезпечення, що вирішує задачі обробки текстової, числової та графічної інформації у вигляді, зрозумілому користувачеві.

У зв'язку з розвитком мережних технологій все більше користувачів переходять до застосування онлайн систем електронного документообігу (Enterprise Content Management System, ECMS), що можуть надавати можливість зручної сумісної роботи над документами, спілкування та транспортування в потрібному вигляді результатів цієї роботи. Також ці системи значно дешевші при використанні, ніж велика кількість локальних комплектів офісного ПЗ.

5.2 ОСНОВНІ КОМПОНЕНТИ

Основними компонентами сучасного офісного пакету можна вважати наступні:

- текстовий процесор,
- електронна таблиця,
- редактор презентацій,
- системи керування базами даних.

Цими видами ПЗ не вичерпується усе різноманіття офісних програм. Сюди можна також додати системи машинного перекладу і розпізнавання тексту, системи векторної та растрової графіки, аудіо- та відео-редактори та ін.

Розглянемо функції основних видів офісного ПЗ.

Текстовий процесор - це комплекс програмних засобів, що призначені для введення, редагування і виведення текстової інформації в потрібному користувачеві вигляді.

До складу функцій текстового процесора може входити:

- введення і форматування тексту;
- перевірка орфографії та пунктуації;
- підготовка тексту до друку;
- автоматизація створення змісту, бібліографічних та інших даних;
- додання зображень, табличних даних, діаграм, формул та інших видів змісту;
- створення і обробка шаблонів для типів документів, що використовуються найчастіше;
- імпорт та експорт документів в розповсюджені формати та ін.

Найчастіше текстовий процесор виконується у вигляді графічного застосування, що працює з документом за технологією WYSIWYG (What You See Is What You Get) — що ти бачиш, те і отримаєш на папері. Тобто, основним елементом інтерфейсу є лист паперу, де користувач за допомогою клавіатури вводить зміст документа, а потім редагує і форматує його за допомогою набору інструментів, що доступні через меню або панель інструментів. Зміст документа відображається на екрані у вигляді, що макси-

мально співпадає з роздрукованим документом. Ця технологія була вперше застосована в текстовому процесорі Microsoft Word 6.0 і стала в наш час найпоширенішою технологією обробки текстової інформації. Позитивні риси цієї технології включають в себе простоту для користувача, великий набір стандартних інструментів для форматування і редагування тексту, можливість одразу побачити результати своєї праці. Недоліками цієї технології є складність створення великих і складних документів, великі вимоги до апаратного забезпечення і неможливість внесення тонких змін до структури документу.

До WYSIWYG-редакторів відносяться

- Microsoft Word
- Microsoft FrontPage
- OpenOffice.org Writer
- KWord
- Abiword та ін.

Існує також інший підхід до обробки текстової інформації. Це так званий WYSIWM (What You See Is What You Mean) — бачиш те, що мав на увазі. При такому підході до обробки тексту документ розглядається як набір даних (тексту, зображень, тощо) і команд обробки (форматування) цих даних. Найбільш відома система цього типу створена Ніклаусом Віртом і називається LaTeX. Вона застосовується для створення наукових статей, монографій і т. ін.

До позитивних рис цієї системи можна віднести цілковитий контроль над виглядом результату обробки документу, легку автоматизацію дій користувача і невимогливість до апаратних ресурсів. Але ця система досить складна для початківця і не дуже пристосована до створення простих документів.

Табличний процесор - це програма, що виконує обробку числової інформації, яка представлена в табличному вигляді. Інтерфейс табличного процесора представляє собою робоче поле, що складається з окремих комірок. Кожна комірка може зберігати певні дані — число, текст або формулу і має унікальну адресу. Також на робочому полі можуть розміщуватися різноманітні діаграми, графіки, зведені таблиці.

Найчастіше ця програма використовується для нескладних економічних розрахунків, але може з успіхом застосовуватися в нескладних наукових і технічних розрахунках в якості калькулятора. Представники цього типу програм:

- Microsoft Excel
- OpenOffice.org Calc
- Gnumeric та ін.

Редактор презентацій - це програма, що використовується для створення документів, які включають в себе текстову, графічну та аудіо- і відео-інформацію. Ці документи застосовуються при проведенні публічних заходів — лекцій, семінарів, рекламних акцій, тощо. Найчастіше, програма поєднує інтерфейс текстового редактора з векторним графічним редактором. Результатом роботи програми є документ, який часто так і називається — презентація. Він складається з декількох слайдів, що демонструються один за одним за допомогою проекційного обладнання або великих моніторів. Демонстрація слайдів може відбуватися автоматично або за командою доповідача.

До цього типу програмних продуктів належить:

- Microsoft PowerPoint
- OpenOffice.org Impression

Система керування базами даних - це сукупність програм, що використовуються для збереження, пошуку і редагування взаємопов'язаних даних. Найбільш поширене сховище для СКБД — реляційна база даних. Вона складається з таблиць, які містять записи, що поділені на окремі поля. Кожне поле може мати той чи інший тип даних. Таблиці можуть бути пов'язані окремими полями, що допомагає уникнути зберігання повторюваних даних. Існує спеціальна мова SQL (Structured Query Language) для пошуку і редагування даних в цих таблицях. Крім самої СКБД, що лише зберігає дані, потрібно мати інтерфейс користувача. Зазвичай, інтерфейс створюється для кожної бази даних окремо.

Для офісних потреб використовують наступні СКБД:

- Microsoft Access (СКБД та оболонка для створення інтерфейсу керування базою даних)
- OpenOffice.org Base (СКБД та оболонка для створення інтерфейсу керування базою даних)
- MySQL (СКБД)

- Firebird (СКБД)

5.3 ОПИС ПОШИРЕНИХ ОФІСНИХ ПАКЕТІВ

Розглянемо найбільш вживані і розповсюджені на сьогоднішній час офісні пакети.

Microsoft Office - це комерційний продукт компанії Microsoft, що є найпоширенішим серед офісних пакетів середнього рівня. Цей пакет включає в себе

- текстовий процесор Microsoft Word,
- табличний процесор Microsoft Excel,
- редактор презентацій Microsoft PowerPoint,
- СКБД Microsoft Access,
- інтегровану систему розробки (IDE) для мови високого рівня VisualBasic,
- різноманітні майстри, шаблони, галереї, та інші необов'язкові розширення.

Інтерфейс кожної програми з цього пакету уніфікований, що допомагає засвоїти засоби керування обробкою інформації. Усі програми з Microsoft Office працюють з закритими і захищеними патентами форматами файлів.

Переваги цього пакету

- Стандарт для офісного ПЗ.
- Великий набір функцій.
- Великий ступінь інтеграцій з операційною системою.
- Велика швидкість роботи.

Недоліки

- Розповсюдження на комерційній основі.
- Закритий формат документів.
- Відсутність підтримки відкритих стандартів.
- Наявність версій лише під MS Windows і Apple MacOS X.

OpenOffice.org - це вільно розповсюджуваний безкоштовний

пакет, що за основними функціями співпадає з пакетом Microsoft Office версії 2000-2003. Підтримується спільнотою розробників і користувачів по всьому світі. Цей пакет почав існування в 2000 році з відкриття вихідних кодів пакету StarOffice з метою його подальшого розвитку на заставках вільного програмного забезпечення. Перші версії пакету не відрізнялися стабільністю, швидкістю роботи і сумісністю з пакетом Microsoft Office. Але зараз, для актуальної версії 3.2 майже усі проблеми вирішені або знаходяться на етапі вирішення, що дозволяє використовувати цей пакет в звичайному документообігу і приватному застосуванні.

До цього пакету входять:

- текстовий процесор Writer,
- табличний процесор Calc,
- векторний графічний редактор Draw,
- редактор презентацій Impression,
- редактор формул Math,
- СКБД Base.
- інтегровану систему розробки (IDE) для мови високого рівня OpenOffice.org Basic, JavaScript, BeanShell,
- різноманітні майстри, галереї малюнків, шаблони документів, тощо.

Переваги

- Вільне розповсюдження.
- Сумісність з Microsoft Office.
- Відкритий стандарт документу ODF.
- Підтримка імпорту та експорту документів в різноманітні формати документів (HTML, PDF, тощо).
- Наявність версій для всіх сучасних графічних ОС.

Недоліки

- Низька швидкість роботи.
- Наявність проблем зі складними документами Microsoft Office.

5.4 ТЕКСТОВИЙ ПРОЦЕСОР OPENOFFICE.ORG WRITER

За допомогою текстового процесора Writer можливо виконання підготовки текстових документів будь-якої складності з використанням зображень, таблиць, діаграм, тощо. За основними функціональними можливостями Writer наближається, а в деяких випадках і перевершує MS Word.

5.4.1 Інтерфейс. Основні функціональні можливості

Інтерфейс текстового процесора представляє собою графічне вікно, що за виглядом відповідає стандартам операційної системи. В його основу покладений принцип WYSIWYG, який дозволяє бачити результати обробки тексту одразу після її виконання. Це дозволяє користувачам-початківцям дуже швидко пристосуватися до умов використання текстового процесору і зосередитися на процесі створення змісту документу.

Основні елементи інтерфейсу Writer.

Головне меню - це набір елементів керування, що дозволяють використовувати всю функціональність текстового процесора.

Елементи головного меню розподілені за призначенням на окремі розділи:

- **Файл** - створення, зберігання, конвертація файлів.
- **Правка** - редагування змісту документа.
- **Вид** - налаштування вигляду текстового процесора.
- **Вставка** - додавання до змісту окремих об'єктів.
- **Формат** - зміна форматування окремих елементів змісту документа.
- **Таблиця** - робота з табличними даними.
- **Сервіс** - налаштування роботи текстового процесору і керування доповненнями.
- **Вікно** - керування вікнами різних документів, що редагуються одночасно.
- **Довідка** - виклик допомоги.

Панель інструментів. Цей елемент представляє собою сукупність

кнопок з найбільш часто використовуваними функціями. Кнопки групуються в окремі панелі, що можуть відокремлюватися від інструментальної панелі. Можливе також додавання і видалення кнопок з кожної панелі.

Робоче поле - це місце, де відображається процес створення документу в реальному часі. Представляє собою один або декілька аркушів паперу (в залежності від обсягу документа і режиму роботи процесора), на яких в реалістичному вигляді (тобто так, як буде виглядати роздрукований документ) виводиться зміст документу.

Рядок стану дозволяє отримати інформацію про стан документа і текстового процесора. Сюди виводиться кількість сторінок документа і номер поточної сторінки, мова елементу змісту або усього документа, режим відображення і масштаб документу.

5.4.2 Стилi оформлення тексту

Для стильового оформлення використовується менеджер стилів, що може бути викликаний клавішею **F11**.

В новому вікні **Стилi і форматування** виводиться список стилів, що доступні для користувача. Над списком знаходиться перемикач типу стилів, що може приймати наступні стани:

- Стилi абзацу.
- Стилi символу.
- Стилi врізки.
- Стилi сторінки.
- Стилi списку.

Внизу цього списку розташований фільтр, за допомогою якого можна виводити лише ті стилі, що потрібні користувачеві зараз. Список стилів також виводиться в панелі інструментів поряд з налаштуваннями шрифту. По замовчанню, для нового документа використовується стиль “Базовий”. Від цього стилю успадковуються усі інші стилі оформлення тексту. Тому, якщо змінити його властивості, ця зміна торкнеться усіх елементів документа. Для початку роботи зі стилями рекомендується створити власний стиль на базі стилю “Базовий” або “Основний текст”.

5.4.3 Введення тексту

Для введення тексту використовується клавіатура. Місце, де з'явиться новий символ, вказує спеціальний знак курсор.

Існує два режими введення символів тексту з клавіатури.

1. Введення з заміною. При цьому введений символ може замінити правий від курсора символ.
2. Введення з додаванням. При цьому введений символ додається після курсора, а символи, що стояли справа, здвигаються на одне знакомісце вправо.

По замовчанню, використовується режим введення з додаванням. Перемикання між режимами відбувається натисканням клавіші **Insert**.

Для видалення символу використовується клавіша **Delete**. Вона видаляє символ, що стоїть справа від курсора. Якщо потрібно видалити символ зліва від курсора, використовується клавіша **BackSpace**.

Разом з безпосереднім вводом тексту існує можливість використання буферу обміну, за допомогою якого можна вставляти фрагменти тексту з інших документів або з цього ж самого документу. Для цього використовують комбінації клавіш **Ctrl+C** для копіювання в буфер обміну і **Ctrl+V** для вставки тексту із буфера обміну в активний документ.

5.4.4 Форматування документу

Для форматування документу за бажанням користувача можна використовувати панель інструментів **Форматування** або пункт меню **Формат**. Операції форматування застосовуються для активного фрагменту документа. Це може бути символ, абзац, список, сторінка або інший елемент.

Хоча для швидкого форматування зручно використовувати панель інструментів або меню, коректнішим було б використання системи стилів.

5.4.5 Структура документу

За своєю структурою текстові документи можуть бути різноманітними — від невеличких листівок і до великих монографій. Для невеликих документів обсягом в 1-2 аркуші можна обійтися без складної структури. Але для більш складних документів з різними вимогами до форматування окремих елементів треба використовувати систему стильового форматування змісту. Для таких документів в якості структурних елементів слід використовувати розділи. Для цього можна вибрати пункт меню **Вставка** → **Розділ**.

5.4.6 Таблиці

Таблиця є дуже зручним засобом представлення великої кількості інформації - різноманітних звітів, прайсів, каталогів продукції, тощо. Таблиці також можуть використовуватися замість табуляції у випадку форматування документів зі складною структурою.

Для створення нової таблиці слід використати один із шляхів:

1. Вибрати пункт меню **Таблиця** → **Вставити** → **Таблиця** і у вікні діалогу **Створення таблиці** задати кількість стовпчиків і рядків нової таблиці.
2. Вибрати на стандартній панелі кнопку **Таблиця** і вибрати кількість стовпчиків і рядків для нової таблиці.

По замовчанню, для першого рядка використовується стиль **Заголовок таблиці**, а решта рядків мають стиль **Зміст таблиці**. За допомогою цих стилів дуже зручно змінювати оформлення всіх таблиць одночасно для всього документа.

Для числової інформації важливим є автоматичне розпізнавання чисел, яке можна включити і виключити за допомогою пункту меню **Сервіс** → **Налаштування** → **OpenOffice.org** → **Таблиця**.

5.4.7 Зображення

Текстовий процесор може використовувати зображення в якості складових текстового документу. Підтримуються зображення розповсюджених растрових форматів: jpeg, gif, png, bmp, tiff та ін. Також існує підтримка векторних форматів зображень dxf, wmf, eps та ін. До складу текстового процесора входить також майстер сканування, що дозволяє використовувати підключений до комп'ютера сканер напряму із оболонки процесора.

Для додавання зображень можна використовувати декілька варіантів дій:

1. Вибрати пункт меню **Вставка** → **Зображення** → **Із файлу**
2. Вибрати пункт меню **Вставка** → **Зображення** → **Сканування...**
3. Скопіювати зображення в буфер обміну і скористатися комбінацією **Ctrl+V**

5.4.8 Розширені операції з документом

Розглянемо деякі операції з текстом, що найчастіше зустрічаються при підготовці структурованих документів.

Створення змісту. Для автоматичного створення змісту документа і подальшого його використання при редагуванні потрібно застосовувати систему стилів для заголовків усіх рівнів, що повинні відображатися в змісті.

В OpenOffice Writer є набір стилів для заголовків починаючи з першого рівня і до десятого включно. Зазвичай, в документі не використовують більше ніж 2-3 рівні підзаголовків.

Для створення автозмісту слід виконати наступні дії:

1. Застосувати до відповідних заголовків стилі **Заголовок 1**, **Заголовок 2**, і т. ін.
2. Вибрати в меню **Вставка** → **Зміст і покажчики** → **Зміст і покажчики...**
3. У вікні діалогу **Вставити зміст або покажчики** вибрати стильове оформлення для автозмісту.
4. Для оновлення змісту після редагування основного тексту потрібно викликати контекстне меню автозмісту і вибрати пункт **Оновити зміст**.

Додавання колонтитулів. Колонтитул — частина сторінки, що відображається зверху або знизу від змісту сторінки. В колонтитулі може виводитися інформація, однакова для всіх сторінок документу (наприклад, назва), або динамічно змінювана в залежності від самої сторінки (наприклад, номер сторінки і назва розділу).

Для вставки колонтитула треба:

1. Вибрати **Вставка** → **Верхній колонтитул** → **Звичайний** для верхнього колонтитула.
2. Вибрати **Вставка** → **Нижній колонтитул** → **Звичайний** для нижнього колонтитула

Після цього в колонтитули можна вводити статичну інформацію, що буде відображена однаково для всіх сторінок розділу (або документу, якщо розділів немає). Для виводу динамічної інформації треба додавати спеціальні

поля (Вставка → Поля). Наприклад, можна додати ім'я автора документа, тему розділу, номер сторінки і загальну кількість сторінок, тощо.

Додавання номерів сторінок. Для додавання номерів сторінок спочатку треба виконати додавання колонтитулів (верхнього або нижнього), як показано в попередній главі. Далі треба помістити курсор в потрібний колонтитул і вибрати в меню пункт **Вставка → Поля → Номер сторінки**.

Якщо потрібно починати нумерацію не з одиниці, необхідно виконати наступну послідовність дій:

1. Додати верхній або нижній колонтитул
2. Додати в нього поле номеру сторінки
3. В основному тексті на сторінці ввести перший абзац або виділити вже існуючий.
4. Натиснути праву кнопку миші і вибрати пункт “Абзац...” контекстного меню
5. Перейти на закладку “Положення на сторінці”
6. Поставити галочку напроти пункту “Додати розрив” і “Зі стилем сторінки”
7. Вибрати стиль наступної сторінки і ввести в поле “Номер стор.” потрібне число.

Додавання виносок та приміток. Виноскою називається сукупність числа біля певного елемента тексту і пояснення цього елемента внизу сторінки.

Для додавання виноски треба:

Активувати потрібний елемент і вибрати пункт меню **Вставка → Виноска**. У вікні діалогу **Вставити виноску** вибрати тип: **Виноска** або **Кінцева виноска**

Звичайна виноска з'являється на активній сторінці, а кінцева додається після основного тексту документа. Місце розташування виноска може бути змінено у вікні діалогу налаштування сторінки (пункт меню **Формат → Сторінка...**) на вкладці **Виноска**.

Створення титульної сторінки. Для створення сторінки, що є ти-

тульною для документа потрібно використати окремий стиль сторінки “Перша сторінка”. Тобто, слід виконати наступні дії:

1. Зробити активною потрібну сторінку
2. Викликати менеджер стилів (клавіша **F11**)
3. Перейти на закладку Стилi сторінки
4. Клацнути два рази на пункті Перша сторінка

Таким же чином слід діяти, коли треба, наприклад, **змінити орієнтацію однієї або декілька сторінок з книжкової на альбомну або навпаки**. Але, при цьому слід використовувати стиль “Альбом” або власноруч створений стиль на основі стилю “Основний”.

Додавання спеціальних символів. В текстовий процесор вмонтовано каталог спеціальних символів, що можуть додаватися до тексту за допомогою вікна діалогу **Вибір символу**, яке з'являється після вибору пункту меню **Вставка → Спеціальні символи....** В залежності від встановлених в операційній системі шрифтів, набір доступних символів може істотно змінюватися.

Перевірка орфографії і граматики. В текстовому процесорі є можливість автоматичної та за потребою перевірки коректності написання слів. Слід зауважити, що для коректної перевірки необхідна наявність набору словників для мови, що використовується в документі. По замовчанню встановлюються лише словники для мови, що співпадає з мовою самого пакету OpenOffice.org. Для розширення можливостей перевірки орфографії можна встановлювати нові словники, які можна отримати на офіційному сайті <http://www.openoffice.org/> або його регіональному дзеркалі. Також важливо встановити для документу або виділеного фрагменту основну мову (пункт меню **Сервіс → Мова**).

Для запуску перевірки треба вибрати пункт меню **Сервіс → Орфографія і граматика** або вибрати кнопку на панелі інструментів. У вікні діалогу **Перевірка орфографії** для кожної помилки можна вибрати один із варіантів дії, що найбільш підходить в кожному окремому випадку.

Також, перевірка може виконуватися автоматично. В цьому випадку помилки автоматично підкреслюються червоною хвилястою лінією. При активації виділеного таким чином слова і виклику контекстного меню можна побачити декілька варіантів і вибрати для виправлення найбільш коректний.

Використання формул. В таблицях є можливість, як це робиться в табличному процесорі Calc, встановити клавішею **F2** режим редагування формули для виділених комірок. Комірки мають адреси, що складаються з імені стовпчика (A, B, C,...,Z) і номеру рядка (1,2,3,...). Для режиму формули можливе використання також деяких стандартних функцій.

5.5 ФОРМУЛИ

5.5.1 Програми для створення формул

Текстовий процесор часто використовується для створення математичних, фізичних, або інших наукових документів. Для цього застосовується спеціальне програмне забезпечення, що називається редактором формул. Це або окремий програмний засіб, або додаток до основного функціоналу текстового процесора.

Прикладами такого програмного засобу є

- Microsoft Equation, MathType (MS Office)
- FireMath (додаток для Firefox)
- OO Math (OO)
- TeX (Latex)

В склад офісного пакету OpenOffice.org входить редактор формул OO Math. Це окремий програмний засіб, що може викликатися із основного текстового процесора для швидкого створення і редагування формул.

5.5.2 Редактор формул OO Math

Для створення формули треба вибрати пункт меню **Вставка** → **Об'єкт** → **Формула**. Інтерфейс редактора формул представляє собою вікно, розділене на дві частини: **поле виводу формули** і **поле вводу тексту формули**. Користувач вводить текст формули в поле вводу, редактор інтерпретує ввід користувача і виводить формулу в основному вікні текстового редактора або у вікні виводу самого редактора формул. Ввід формули виконується за допомогою спеціальної мови, що є діалектом мови розмітки тексту в системі LaTeX. Наприклад, для того, щоб вивести формулу

$$E=mc^2$$

треба в полі вводу набрати

$$E=mc^2.$$

Або, для формули

$$V=\frac{4}{3}\cdot\pi R^2$$

треба ввести

$$V=\frac{4}{3}\cdot\pi R^2.$$

Такий підхід дозволяє дуже швидко створювати великі і складні формули.

В таблицях 5.1-5.9 наведено основні елементи оформлення математичних виразів, що можна використовувати при підготовці документу з а допомогою OpenOffice.org Math.

Таблиця 5.1 Елементи оформлення математичних виразів

Поле вводу	Поле виводу	Поле вводу	Поле виводу
a^b Лівий верхній індекс	a^b	a^b Правий нижній індекс	a^b
a^b Центральний верх- ній індекс	a^b	a_b Центральний нижній індекс	a_b
a_b Лівий нижній індекс	a_b	a_b Правий нижній індекс	a_b
bold 123 Товстий	123	<i>ital 123</i> Курсив	<i>123</i>
<u>underline 123</u> Підкреслений	<u>123</u>	size 8 123 Розмір	123
color red 123 Колір (червоний)	123	a phantom b c Невидимість елемента (b)	$a^b c$

Продовження таблиці 5.1

Поле вводу	Поле виводу	Поле вводу	Поле виводу
vec a Вектор	\vec{a}	bar a	\bar{a}
tilde a Тильда	\tilde{a}	hat a Циркумфлекс	\hat{a}
circle a Кружок	$\circ a$	breve a Дуга	\breve{a}
widevec abc Довгий вектор	\overrightarrow{abc}	overline abc Лінія зверху	\overline{abc}
widetilde abc Довга тильда	\widetilde{abc}	widehat abc Довгий циркумфлекс	\widehat{abc}
underline abc Підкреслювання	\underline{abc}	overstrike abc Закреслювання	\overline{abc}
{abc} underbrace {a} Нижня дужка	\underbrace{abc}_a	{abc} overbrace {a} Верхня дужка	\overbrace{abc}^a
{abc} underbrace {a} Нижня дужка	\underbrace{abc}_a	{abc} overbrace {a} Верхня дужка	\overbrace{abc}^a
binom {a}{b} Стовпець з двох елементів	$\begin{matrix} a \\ b \end{matrix}$	stack{a#b#c} Стовпець з трьох елементів	$\begin{matrix} a \\ b \\ c \end{matrix}$
matrix {a#b#c##d#e#f} Три стовпці в два рядка	$\begin{matrix} a & b & c \\ d & e & f \end{matrix}$	matrix {a#b##c#d##e#f} Два стовпці в три рядка	$\begin{matrix} a & b \\ c & d \\ e & f \end{matrix}$
matrix {a#"="b##{}#"="c} Вирівнювання за допомогою матриці	$\begin{matrix} a & " = " b \\ & " = " c \end{matrix}$	dotsvert Точки вертикально	\vdots
dotslow Точки знизу	\dots	dotsaxis Точки посередині	\dots

Продовження таблиці 5.1

Поле вводу	Поле виводу	Поле вводу	Поле виводу
<code>dotsup</code> Точки по діагоналі уверх	\ddots	<code>dottdown</code> Точки по діагоналі униз	\dotso
<code>Leftarrow</code> Стрілка вліво	\leftarrow	<code>Rightarrow</code> Стрілка вправо	\rightarrow
<code>uparrow</code> Стрілка уверх	\uparrow	<code>downarrow</code> Стрілка униз	\downarrow
<code>left lbrace</code> <code>stack {a#b#c}</code> <code>right none</code> Поодинокі ліва фігурна дужка, що масштабуються	$\left\{ \begin{matrix} a \\ b \\ c \end{matrix} \right.$	<code>left none</code> <code>stack {a#b#c}</code> <code>right rbrace</code> Поодинокі права фігурна дужка, що масштабуються	$\left. \begin{matrix} a \\ b \\ c \end{matrix} \right\}$

Таблиця 5.2 Елементи алгебри

Поле вводу	Поле виводу	Поле вводу	Поле виводу
<code>+a</code>	$+a$	<code>-a</code>	$-a$
<code>+-a</code>	$\pm a$	<code>a+-b</code>	$a \pm b$
<code>a+b</code>	$a+b$	<code>a-b</code>	$a-b$
<code>a \cdot b</code>	$a \cdot b$	<code>a times b</code>	$a \times b$
<code>a * b</code>	$a * b$	<code>a and b</code>	$a \wedge b$
<code>a div b</code>	$a \div b$	<code>a/b</code>	a/b
<code>a over b</code>	$\frac{a}{b}$	<code>c a over b</code>	$c \frac{a}{b}$
<code>neg a</code>	$\neg a$	<code>a or b</code>	$a \vee b$
<code>abs{x}</code>	$ x $	<code>a^m</code>	a^m
<code>sqrt x</code>	\sqrt{x}	<code>nroot m a</code>	$\sqrt[m]{a}$
<code>func e^x</code>	e^x	<code>exp x</code>	$\exp x$

Продовження таблиці 5.2

Поле вводу	Поле виводу	Поле вводу	Поле виводу
<code>ln x</code>	$\ln x$	<code>log_a b</code>	$\log_a b$
<code>"lg" x</code>	$\lg x$	<code>fact x</code>	$x!$
<code>(a+b)</code> Круглі дужки	$(a+b)$	<code>[a+b]</code> Квадратні дужки	$[a+b]$
<code>{a+b} over c+d</code> Об'єднуювальні дужки	$\frac{a+b}{c} + d$	<code>lbrace a+b</code> <code>rbrace</code> Фігурні дужки	$\{a+b\}$
<code>langle a+b</code> <code>rangle</code> Кутові дужки	$\langle a+b \rangle$	<code>lline a+b</code> <code>rline</code> Модуль	$ a+b $
<code>ldline a+b</code> <code>rdline</code>	$\ a+b\ $	<code>(a mline b)</code>	$(a b)$
<code>left(a over b</code> <code>right)</code> Круглі дужки, що масштабуються	$\left(\frac{a}{b}\right)$	<code>left [a over b</code> <code>right]</code> Квадратні дужки, що масштабуються	$\left[\frac{a}{b}\right]$
<code>left lbrace a+b</code> <code>right rbrace</code> Фігурні дужки, що масштабуються	$\{a+b\}$	<code>left langle a</code> <code>over b right</code> <code>rangle</code> Кутові дужки, що масштабуються	$\left\langle \frac{a}{b} \right\rangle$
<code>left lline a</code> <code>over b right</code> <code>rline</code> Вертикальні лінії, що масштабуються	$\left \frac{a}{b}\right $	<code>left ldline a</code> <code>over b right</code> <code>ldline</code> Подвійні вертикальні лінії, що масштабуються	$\left\ \frac{a}{b}\right\ $
<code>left [stack</code> <code>{a#b#c#dotslow#</code> <code>g} right]</code> Багатовимірний вектор	$\begin{bmatrix} a \\ b \\ c \\ \dots \\ g \end{bmatrix}$	<code>left (matrix</code> <code>{a#b##c#d##e#f</code> <code>} right)</code> Матриця розміром (з трьох рядків і двох стовпців)	$\begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix}$

Таблиця 5.3 Співвідношення величин

Поле вводу	Поле виводу	Поле вводу	Поле виводу
$a=b$ Дорівнює	$a=b$	$a \equiv b$ Тотожно дорівнює	$a \equiv b$
$a \neq b$ Не дорівнює	$a \neq b$	$a \approx b$ Наближено дорівнює	$a \approx b$
$a < b$ Менше	$a < b$	$a > b$ Більше	$a > b$
$a \leq b$ Менше або дорівнює	$a \leq b$	$a \geq b$ Більше або дорівнює	$a \geq b$
$a \mid b$ Кратно	$a \mid b$	$a \nmid b$ Не кратно	$a \nmid b$

Таблиця 5.4 Елементи геометрії

Поле вводу	Поле виводу	Поле вводу	Поле виводу
$a \parallel b$ Паралельно	$a \parallel b$	$a \perp b$ Перпендикулярно	$a \perp b$
$a \sim b$ Подібно	$a \sim b$	$a \propto b$	$a \propto b$
$a \simeq b$ Подібно або дорівнює	$a \simeq b$		

Таблиця 5.5 Тригонометричні та гіперболічні функції

Поле вводу	Поле виводу	Поле вводу	Поле виводу
$\sin\{x\}$ Сінус	$\sin x$	$\cos\{x\}$ Косінус	$\cos x$
$\tan\{x\}$ Тангенс (американський варіант)	$\tan x$	$\text{"tg"}\{x\}$ Тангенс (російський варіант)	$\text{tg } x$

Продовження таблиці 5.5

Поле вводу	Поле виводу	Поле вводу	Поле виводу
$\cot\{x\}$ Котангенс (американський варіант)	$\cot x$	"ctg"{x} Котангенс (російський варіант)	$\operatorname{ctg} x$
$\arcsin\{x\}$ Арксінус	$\arcsin x$	$\arccos\{x\}$ Арккосінус	$\arccos x$
$\arctan\{x\}$ Арктангенс (американський варіант)	$\arctan x$	"arctg"{x} Арктангенс (російський варіант)	$\operatorname{arctg} x$
$\operatorname{arccot}\{x\}$ Арккотангенс (американський варіант)	$\operatorname{arccot}\{x\}$	"arcctg"{x} Арккотангенс (російський варіант)	$\operatorname{arcctg} x$
$\sinh\{x\}$ Гіперболічний синус (американський варіант)	$\sinh\{x\}$	"sh"{x} Гіперболічний синус (російський варіант)	$\operatorname{sh} x$
$\cosh\{x\}$ Гіперболічний косинус (американський варіант)	$\cosh\{x\}$	"ch"{x} Гіперболічний косинус (російський варіант)	$\operatorname{ch} x$
$\tanh\{x\}$ Гіперболічний тангенс (американський варіант)	$\tanh x$	"th"{x} Гіперболічний тангенс (російський варіант)	$\operatorname{th} x$
$\operatorname{coth}\{x\}$ Гіперболічний котангенс (американський варіант)	$\operatorname{coth} x$	"cth"{x} Гіперболічний котангенс (російський варіант)	$\operatorname{cth} x$
$\operatorname{arsinh}\{x\}$ Гіперболічний арксинус (американський варіант)	$\operatorname{arsinh} x$	"arsh"{x} Гіперболічний арксинус (російський варіант)	$\operatorname{arsh} x$

Продовження таблиці 5.5

Поле вводу	Поле виводу	Поле вводу	Поле виводу
$\cot\{x\}$ Котангенс (американський варіант)	$\cot x$	"ctg"{x} Котангенс (російський варіант)	$\operatorname{ctg} x$
$\operatorname{arcosh}\{x\}$ Гіперболічний арккосинус (американський варіант)	$\operatorname{arcosh} x$	"arch"{x} Гіперболічний арккосинус (російський варіант)	$\operatorname{arch} x$
$\operatorname{artanh}\{x\}$ Гіперболічний арктангенс (американський варіант)	$\operatorname{artanh} x$	"arth"{x} Гіперболічний арктангенс (російський варіант)	$\operatorname{arth} x$
$\operatorname{arcoth}\{x\}$ Гіперболічний арккотангенс (американський варіант)	$\operatorname{arcoth} x$	"arcth"{x} Гіперболічний арккотангенс (російський варіант)	$\operatorname{arcth} x$

Таблиця 5.6 Елементи математичного аналізу

Поле вводу	Поле виводу	Поле вводу	Поле виводу
a toward b Прагне до	$a \rightarrow b$	a dlrarrow b еквівалентність	$a \Leftrightarrow b$
a dlarrow b	$a \Leftarrow b$	a drarrow b	$a \Rightarrow b$
$\lim \{a\}$ Границя	$\lim a$	$\lim \operatorname{csub} \{x \text{ toward } x_0\} \{y \text{ toward } y_0\}$ Границя	$\lim_{x \rightarrow y} a$
$\lim \operatorname{csub} \{\operatorname{binom} \{x \text{ toward } x_0\} \{y \text{ toward } y_0\}\}$ $\{\sin \{xy \text{ over } y\}\}$	$\lim_{\substack{x \rightarrow x_0 \\ y \rightarrow y_0}} \sin \frac{xy}{y}$	forall квантор загальності (будь-який)	\forall
exists квантор існування (існує)	\exists	sum {a} Сума	$\sum a$

Продовження таблиці 5.6

Поле вводу	Поле виводу	Поле вводу	Поле виводу
sum from k {a_k} Сума	$\sum_k a_k$	sum from k=1 to m {a_k} Сума	$\sum_{k=1}^m a_k$
prod {a} Добуток	$\prod a$	prod from k {a_k} Добуток	$\prod_k a_k$
prod from k=1 to m {a_k} Добуток	$\prod_{k=1}^m a_k$	coprod {a} Кодобуток	$\coprod a$
df over dx Похідна	$\frac{df}{dx}$	dot {vec r} Похідна від вектора по змін- ній t (t-час)	$\dot{\vec{r}}$
{d^2 f} over dx^2 Похідна другого порядку	$\frac{d^2 f}{dx^2}$	ddot {vec r} Похідна другого порядку від вектора по змін- ній t (t-час)	$\ddot{\vec{r}}$
{partial f} over {partial x} Часткова похідна	$\frac{\partial f}{\partial x}$	{partial^2 f} over {partial x^2} Часткова похідна другого порядку	$\frac{\partial^2 f}{\partial x^2}$

Таблиця 5.7 Множини

Поле вводу	Поле виводу	Поле вводу	Поле виводу
setN Множина натуральних чисел	\mathbb{N}	setZ Множина цілих чисел	\mathbb{Z}
setQ Множина ра- ціональних чисел	\mathbb{Q}	setR Множина дійсних чисел	\mathbb{R}
setC Множина компле- ксних чисел	\mathbb{C}	emptyset Множина порожня	\emptyset

Продовження таблиці 5.7

Поле вводу	Поле виводу	Поле вводу	Поле виводу
a in b Належить	$a \in b$	a notin b Не належить	$a \notin b$
a intersection b Переріз множин	$a \cap b$	a union b Об'єднання множин	$a \cup b$
a setminus b Віднімання множин	$a \setminus b$	a slash b Частка множин	a / b
a subset b Входить до	$a \subset b$	a subseteq b Входить до або дорівнює	$a \subseteq b$
a supset b Входить до	$a \supset b$	a supseteq b Входить до або дорівнює	$a \supseteq b$
a nsubset b Не входить	$a \not\subset b$	a nsubseteq b Не входить і не дорівнює	$a \not\subseteq b$
a nsupset b Не входить	$a \not\supset b$	a nsupseteq b Не входить і не дорівнює	$a \not\supseteq b$

Таблиця 5.8 Спеціальні символи

Поле вводу	Поле виводу	Поле вводу	Поле виводу
infinity Нескінченність	∞	nabla Диференційний оператор Гамільтона	∇
%DELTA% Оператор Лапласа	Δ	%infinite	∞
Re Дійсна частина	\Re	Im Мніма частина	\Im
%and	\wedge	%or	\vee
%element	\in	%noelement	\notin
%identical	\equiv	%notequal	\neq
%strictlylessthan	\ll	%strictlygreaterthan	\gg

Таблиця 5.9 Грецькі букви

%ALPHA	A	%BETA	B	%CHI	X
%DELTA	Δ	%EPSILON	E	%ETA	H
%GAMMA	Γ	%IOTA	I	%KAPPA	K
%LAMBDA	Λ	%MU	M	%NU	N
%OMEGA	Ω	%OMICRON	O	%PHI	Φ
%PI	Π	%PSI	Ψ	%RHO	P
%SIGMA	Σ	%THETA	Θ	%UPSILON	Y
%XI	Ξ	%ZETA	Z		
%alpha	α	%beta	β	%chi	χ
%delta	δ	%epsilon	ϵ	%eta	η
%gamma	γ	%iota	ι	%kappa	κ
%lambda	λ	%mu	μ	%nu	ν
%omega	ω	%omicron	o	%phi	ϕ
%pi	π	%psi	ψ	%rho	ρ
%sigma	σ	%theta	θ	%upsilon	υ
%xi	ξ	%zeta	ζ		

5.6 ОБРОБКА ГРАФІЧНОЇ ІНФОРМАЦІЇ

5.6.1 Види графічного програмного забезпечення

Графічна інформація є однією із самих розповсюджених видів інформації. Велика частка сучасного контенту, що зберігається на комп'ютерах користувачів і ресурсах мережі Інтернет представлена в графічному вигляді. Це зображення, малюнки, фотографії, креслення, схеми, діаграми тощо. Розповсюдження цього виду інформації пов'язане насамперед з легкістю і зручністю сприйняття цієї інформації користувачем, а також з розвитком засобів обробки і відображення графіки — потужними графічними прискорювачами, різноманітними системами візуалізації від звичайних моніторів і до систем тривимірного виводу. Від інших типів інформації графіка відрізняється

впорядкованою структурою. Зазвичай, графічна інформація зберігається або у вигляді набору окремих точок — пікселів (від англійської - pixel), або у вигляді графічних примітивів — геометричних фігур. Це може бути як двовимірна система (фотографія, малюнок), так і тривимірна (креслення).

Слід зазначити, що для відображення графічної інформації майже завжди (окрім деяких моніторів з векторною розгорткою) використовується растровий метод, тобто зображення створюється за допомогою великої кількості окремих елементів — растру. Усі інші види збереження графічної інформації перетворюються в растровий для відображення.

5.6.2 Растрові редактори

Растрова графіка — збереження графічної інформації у вигляді спеціальної структури растру, що представляє собою матрицю елементів - пікселів, кожний із них зберігає інформацію про одну точку зображення.

Для кожного пікселя задається колір. Растрове зображення створюється із пікселів, що розподіляються по рядках і стовпчиках.

Максимальна деталізація растрового зображення задається при його створенні і не може бути збільшена.

Важливими характеристикам для растрового зображення є

- кількість елементів (зазвичай вказується в пікселях по вертикалі і горизонталі, наприклад: 1280 на 1024);
- глибина кольору (16, 24 або 32 біта);
- кольорова модель (RGB або CMYK).

Кількість пікселів зображення впливає на якість відображення. Чим більше пікселів — тим якісніше виглядає зображений об'єкт. Для кожного пікселя можна задати колір довільною кількістю бітів. Чим більше бітів, тим краще наближення кольору пікселя до кольору реального об'єкту. Наприклад, для кольорової моделі RGB використання 8 біт для кожного кольору дає 24 біти на піксель і $2^{24}=16$ мільйонів відтінків. Для монохромного зображення можна використовувати меншу кількість біт. Вже 8 біт дадуть 256 відтінків сірого від білого до чорного.

Від кольорової моделі залежить вибір основних кольорів растру і зручність використання зображення при виводі на екран і на друк.

Розглянемо найбільш поширені кольорові моделі.

Кольорова модель RGB (Red, Green, Blue — червоний, зелений, блакитний) описує кожний піксель зображення трьома компонентами основних кольорів. Ця модель використовується, в основному, для виводу зображення на пристроях, що випромінюють світло, тобто на моніторах. Ця модель заснована на аддитивному законі додавання відтінків основних кольорів для створення кольору пікселя. Так, для червоного кольору буде використовуватися комбінація R:255, G:0, B:0. Для білого R:255, G:255, B:255, для жовтого R:255, G:255, B:0.

Кольорова модель CMYK (Cyan, Magenta, Yellow, Black — світло-блакитний, пурпуровий, жовтий, чорний) описує кожний піксель зображення трьома компонентами додаткових кольорів. Ця модель використовується, в основному, для виводу зображення на папір. При цьому використовується субстрактивний закон додавання відтінків. Тобто, якщо узяти всі три компоненти максимального значення, отримаємо чорний колір. Додатковий окремий чорний колір використовується для поліграфічного друку і дозволяє отримувати більш контрастне і якісне зображення.

Переваги растрової графіки

- Відтворення майже довільного зображення з заданою якістю.
- Поширеність.
- Швидкість обробки великих зображень.
- Наближеність до засобів вводу-виводу.

Недоліки

- Великий розмір файлів.
- Неможливість масштабування без втрат якості.

Існує велика кількість форматів збереження растрової графіки. Наведемо найбільш розповсюджені:

- BMP (Bitmap Picture) простий формат без стиску (Windows).
- GIF (Graphics Interchange Format) — простий формат зі стиском (256 кольорів одночасно).
- JPEG (Joint Photographics Experts Group) — поширений формат зі стиском з втратою інформації.
- PNG (Portable Network Graphics) — поширений формат зі стиском з втратою інформації (є можливість задавати прозорість).

- TIFF (Tagged Image File Format) - формат зі стиском без втрати інформації.
- RAW формат даних для збереження фотографій в цифрових фотокамерах.

Існує безліч векторних редакторів, як спеціалізованих, так і універсальних. Наведемо список найбільш розповсюджених.

Вільно розповсюджувані:

GIMP

Krita

Paint.NET

Комерційні редактори

Adobe PhotoShop

Corel Photo Paint

MS Paint

5.6.3 Векторні редактори

Одним із засобів збереження графічної інформації - векторний.

Векторна графіка — це сукупність засобів і геометричних примітивів для створення зображення методами комп'ютерної графіки.

Основні графічні примітиви: точки, лінії, полілінії, криві Без'є, сплайни, кола та еліпси, багатокутники, текст.

Роздивимося приклад використання векторної графіки для кола. Його можна уявити як сукупність наступних елементів:

- Координати центру кола.
- Радіус кола.
- Товщина лінії.
- Колір лінії.
- Тип лінії (безперервна, переривчаста, подвійна тощо).
- Тип заповнення внутрішнього простору кола (без заповнення, штрих, колір).

Цю структуру можна зберігати в спеціальному файлі і відтворювати зображення кола коли виникає потреба.

Переваги цього засобу зберігання графічної інформації

- Мінімальний розмір файла .
- Масштабування без втрати якості зображення.
- Легке і зручне редагування створених об'єктів.

Недоліки

- Не кожний об'єкт можна представити у вигляді векторного зображення.
- Великі вимоги до апаратного забезпечення для відтворення векторного зображення.
- Залежність швидкості виводу зображення від кількості об'єктів.
- Відсутність методу перетворення растрового зображення у векторне без втрати якості.

Векторні графічні редактори, типово, дозволяють обертати, переміщати, відбивати, розтягувати, скошувати, виконувати основні афінні перетворення над об'єктами, комбінувати примітиви в більш складні об'єкти. Також можна використовувати булеві операції для об'єктів: об'єднання, доповнення, перетин та ін.

Найбільш використовувані формати збереження графічних даних:

PDF (portable Document Format),

EPS (Encapsulated Postscript),

SVG (Scalable Vector Graphics),

WMF (Windows Meta File).

Також використовуються специфічні формати для кожного векторного редактору, наприклад:

CDR (Corel DRAW),

DXF (AutoCAD),

AI (Adobe Illustrator).

Існує безліч векторних редакторів, як спеціалізованих, так і універсальних. Наведемо список найбільш розповсюджених.

Вільно розповсюджувані:

Inkscape,

OpenOffice.org Draw,

Xara Xtreme.

Комерційні

Corel DRAW,

Adobe Illustrator,

Maya (система тривимірної графіки).

5.7 БАЗИ ДАНИХ

5.7.1 Визначення бази даних

Базою даних є представлена в об'єктивній формі сукупність самостійних даних, систематизованих таким чином, щоб ці дані могли бути знайдені і оброблені за допомогою комп'ютерної техніки.

База даних має відмінні особливості:

- База даних зберігається і обробляється в інформаційній системі.
- Дані в базі даних добре структуровані (є можливість виділення окремих елементів, зв'язків між ними, існує типізація для даних і зв'язків).

Бази даних можна класифікувати за наступними ознаками:

За моделлю даних

- ієрархічні,
- мережні,
- реляційні,
- багатовимірні,
- об'єктні та ін.

За призначенням

- географічні,
- історичні,

- наукові,
- мультимедійні,

За ступенем розподілення

- централізовані,
- розподілені.

Визначення СКБД

Система керування базами даних (СКБД) — це сукупність програмних компонентів, призначених для внесення, зберігання, пошуку, редагування і аналізу великої кількості даних.

Основні функції СКБД

- керування даними у зовнішній пам'яті (на дисках);
- керування даними в оперативній пам'яті з використанням дискового кеша;
- журналізація змін, резервне копіювання і відновлення бази даних після збоїв;
- підтримка мов БД (мова визначення даних (DDL- Data Definition Language), мова маніпулювання даними (DML- Data Manipulation Language)).

СКБД можуть застосовувати різні засоби доступу до бази даних:

- Файл-серверні (дані розташовані на файл-сервері, доступ до них організується засобами операційної системи).
- Клієнт серверні (дані розташовані на сервері бази даних, існує спеціальне програмне забезпечення для доступу до даних).
- Вбудовані (дані зберігаються на локальному комп'ютері, доступ організується за допомогою вбудованих функцій конкретного застосування).

Найбільш поширеними і зручними для повсякденного користування є локальні або мережні (клієнт-серверні або файл-серверні) реляційні бази даних.

5.7.2 Основні елементи СКБД

Розглянемо ближче структуру звичайної реляційної бази даних на прикладі OpenOffice.org Base.

Дані всередині бази даних зберігаються у вигляді таблиць (table). Кожна таблиця складається з рядків і стовпців. Дані, що містяться в одному рядку, називаються записом (record). Кожен запис складається з одного або більше полів (field), а кожне поле має свій тип. Наприклад, тип поля INTEGER (або INT) використовується для зберігання цілочислових значень, у той час як тип VARCHAR підходить для роботи з текстовими рядками. Тип поля DATE, як випливає з назви, використовується для зберігання дат, а двійковий (BINARY) тип створений для зберігання двійкових даних, наприклад, зображень.

Запити (queries) використовуються для здобування, перегляду і маніпулювання даними. Запити є інструментами для сортування, фільтрації, зміни налаштувань і аналізу даних. У Base (як і в більшості СУБД), запити є створеними SQL-скриптами. Дозволяючи працювати з командами і скриптами SQL, Base також має графічні інструменти, що допомагають створювати досить складні запити, не заглиблюючись в SQL-програмування.

Форми (forms) дозволяють переглядати та редагувати дані з таблиць. Можна розглядати форми як графічний інтерфейс бази даних: в той час як таблиці служать для зберігання даних, форми служать для їх відображення і маніпулювання ними в таблицях.

Звіти (reports) використовуються для виводу на принтер змісту таблиці. Наприклад, якщо у вас є таблиця бази даних, що містить інформацію про адреси, ви можете створити звіт, який виводить адреси у вигляді акуратно відформатованої електронної таблиці. Більш складні звіти можуть видавати дані не тільки безпосередньо з таблиць, але також і з запитів. Конструктор звітів (Report Builder), що водить до Base, також дозволяє створювати звіти, використовуючи візуальні інструменти.

5.7.3 Поширені СКБД

Розглянемо найбільш поширені СКБД і наведемо кратку характеристику для кожної з них.

- MS Access (СКБД початкового рівня, файл- серверна, комерційна, входить до складу MS Office, ОС Windows).
- OO Base (СКБД початкового рівня, файл- серверна або клієнт- серверна, вільно розповсюджувана, входить до складу OpenOffice.org, кросплатформенна).

- Interbase (СКБД середнього рівня, клієнт- серверна, комерційна, ОС Windows).
- Firebird (СКБД середнього рівня, клієнт- серверна, вільно розповсюджувана, кросплатформенна).
- MySQL (СКБД середнього рівня, клієнт- серверна, вільно розповсюджувана, кросплатформенна).
- PostgreSQL (СКБД середнього рівня, клієнт- серверна, вільно розповсюджувана, кросплатформенна).
- MS SQL Server (СКБД професійного рівня, клієнт- серверна, комерційна, ОС Windows).
- Oracle (СКБД професійного рівня, клієнт- серверна, комерційна, кросплатформенна).

Для локальних та невеликих мережних баз даних часто застосовується СКБД MS Access. В Інтернет-проектах дуже часто використовують СКБД MySQL або PostgreSQL. Для середнього бізнесу дуже вживаною є СКБД MS SQL Server або Firebird.

5.8 ПРЕЗЕНТАЦІЇ

5.8.1 Поняття про презентацію

Презентація — це поєднання комп'ютерної анімації, графіки, відео, музики та звукового ряду, які зберігаються в одному документі. Як правило, презентація має сюжет, сценарій і структуру, організовану для зручного сприйняття інформації.

Презентація — це, зазвичай, рекламний або інформаційний інструмент, що дозволяє користувачеві активно взаємодіяти з ним через меню управління. Презентація зазвичай містить в собі текст, ілюстрації до нього і витримана в єдиному графічному стилі. Сьогодні інформаційні технології дозволяють створювати презентації з використанням аудіо- і відеовставок, робити презентації динамічними і інтерактивними, використовувати в них гіпертекстові посилання.

Презентація створюється з окремих елементів, які називаються слайдами. Процес демонстрації матеріалу презентації полягає в послідовній зміні слайдів у відповідній до сценарію послідовності.

Кожний кадр може містити довільну кількість об'єктів — текст, зображення, гіперпосилання тощо. Перехід з одного кадру до іншого може викону-

ватись як автоматично, так і по команді користувача. Перехід може супроводжуватися анімаційними ефектами.

5.8.2 Презентаційне програмне забезпечення

Існує досить багато технологій створення презентацій і відповідного програмного забезпечення. Для звичайних користувачів достатню функціональність забезпечують програмні засоби, що входять до складу офісного пакету. Наведемо поширені редактори презентацій:

Microsoft PowerPoint (комерційна ліцензія),

OpenOffice.org Impress (вільна ліцензія),

Mediaslance Multimedia Builder (обмежена ліцензія).

Найбільш відомі перші два з них. При чому, OpenOffice.org Impress може відтворювати з деякими обмеженнями документи, створені в Microsoft PowerPoint і має усі необхідні функції для створення сучасної презентації.

Розглянемо типове створення презентації за допомогою OpenOffice.org Impress.

5.8.3 Створення нової презентації

Створення нової презентації починається з запуску Майстра презентацій

Для цього треба виконати наступні дії:

1. Запустіть OpenOffice.org (OOo) Impress. З'явиться вікно **Майстра Презентацій**.

2. В секції **Тип** встановіть один з наступних перемикачів:

- **Порожня презентація** - створення презентації «з нуля».
- **З шаблону** - дозволяє використовувати один зі стандартних шаблонів як основи для нової презентації. Буде відображений список доступних шаблонів.
- **Відкрити існуючу презентацію** - дозволяє продовжити роботу над будь-якою вже створеною презентацією. Буде відображений список існуючих презентацій.

Не знімайте прапорець **Попередній перегляд**, щоб шаблони, стилі слайдів і переходи між слайдами автоматично відображалися в області попереднього перегляду. Якщо Ви не хочете, щоб Майстер Презентацій відкривався при кожному запуску

Impress, встановіть прапорець **Більше не показувати цей діалог**.

3. Натисніть **Далі**, щоб перейти до другого кроку **Майстра Презентацій**. Якщо ви встановили другий перемикач, то в області попереднього перегляду відобразиться зразок слайда.

4. Встановіть бажаний стиль слайда у секції вибору фону і стилю слайда. Виберіть **Оригінал**, якщо не бажаєте використовувати жодної з наявних стилів (порожній стиль). 5. Залежно від подальшого використання презентації, встановіть потрібний перемикач у секції **Спосіб відображення презентації**. У більшості випадків потрібен вигляд презентації на екрані монітора.

6. Натисніть **Далі**, щоб перейти до кроку **Майстра Презентацій - Вибір ефекту і швидкості при зміні слайдів**

7. Виберіть бажаний ефект зі списку **Ефект**.

8. Виберіть бажану швидкість переходу між слайдами зі списку **Швидкість**.

9. Натисніть на кнопку **Готово**.

Буде створена нова презентація.

Бажано зберегти презентацію відразу після її створення. Не забувайте також регулярно зберігати вашу презентацію в процесі роботи з нею, щоб уникнути випадкової втрати інформації.

Якщо на першому кроці **Майстра Презентацій** ви обираєте варіант 3 шаблону, то на третьому кроці буде доступна кнопка **Далі** і наступні кроки **Майстра Презентацій**. Ці кроки тут не розглядаються.

5.8.4 Форматування презентацій

5.8.4.1. **Вибір розмітки першого слайда.** У правій частині робочого простору Impress Ви побачите панель **Макети**. На панелі **Макети** відображаються ескізи слайдів, які допоможуть вам під час форматування презентації. Кожен ескіз відповідає певній розмітці слайда. По кліку на ескізі буде встановлена відповідна розмітка слайда. Зверніть увагу на спливаючі підказки до ескізів.

5.8.4.2. **Вставка нових слайдів.** Нові слайди завжди вставляються після активного (виділеного) слайда.

1. З меню виберіть **Вставити** → **Слайд**. У робочій області з'явиться порожній слайд.

2. Виберіть **Формат** → **Макети слайдів**. У правій частині робочого простору розгорнеться панель **Макети**.

3. Виберіть потрібну розмітку слайда.

4. Натисніть **ОК**.

Вставити слайд можна і за допомогою панелі інструментів **Презентація**. Ця панель дозволяє швидко отримати доступ до функцій, що служать для роботи зі слайдами. Якщо ця панель інструментів не відображається, виберіть **Вигляд** → **Панелі інструментів** → **Презентація**.

5.8.5 Робота зі слайдами

5.8.5.1. Зміна слайдів.

1. Виділіть слайд, який хочете змінити, зі списку слайдів, що знаходиться в лівій частині робочої області.

2. Змінійте слайд за допомогою вибору іншої розмітки з панелі **Макети**, що знаходиться в правій частині робочої області або за допомогою меню: **Формат** → **Макети слайдів** → **Видалення слайдів**.

3. Клацніть мишею на слайді, який ви хочете видалити.

4. Виберіть **Правка** → **Видалити слайд**.

5.8.5.2. Перейменування слайда

1. Клацніть правою кнопкою миші по ескізу слайда на панелі в лівій частині робочої області і виберіть з контекстного меню **Перейменувати слайд**. З'явиться діалогове вікно **Перейменувати слайд**.

2. У полі **Ім'я** введіть нове ім'я слайда, і натисніть **ОК**.

5.8.5.3. Зміна порядку слайдів.

1. У робочій області клацніть на вкладці **Режим слайдів**.

2. Для своєї зручності можете змінити кількість слайдів, які відображаються в одному рядку.

3. Перетягувати слайди мишею. У місці, де буде вставлений слайд, з'явиться чорна вертикальна лінія.

4. Для того, щоб виділити кілька слайдів, натисніть ліву кнопку миші і,

не відпускаючи її, виділіть групу слайдів, після чого можна перетягувати всю виділену групу слайдів.

Множинне виділення можна також здійснити, утримуючи клавіші **Ctrl** або **Shift**.

5.8.6 Режими робочого простору

Область роботи зі слайдом розташована по центру робочого простору, ліворуч від слайда розташовані ескізи слайдів, праворуч - панель задач.

У Impress є п'ять режимів робочого простору. Кожне з них служить для вирішення певного кола завдань.

Режим малювання використовується в більшості випадків. Він слугує для форматування та розмітки слайда, додавання тексту, графіки та ефектів анімації.

Режим структури слугує для відображення загальної структури презентації та роботи з цілими слайдами як з об'єктами. Цей режим надає можливість зміни порядку слайдів і додати нові.

Режим приміток дозволяє додати до кожного слайда примітку, на яку можна буде посилатися при необхідності. Можна змінити розмір і місце розташування поля примітки за допомогою миші

Режим тез виводить слайди у зменшеному вигляді і в оптимальному для друку вигляді.

Режим слайдів представляє слайди в зменшеному вигляді, розташовуючи їх у звичайному порядку. Цей режим використовується для зміни порядку слайдів, демонстрації певного фрагмента презентації або додавання переходів між слайдами.

5.8.7 Показ слайдів

1. Для запуску демонстрації слайдів виберіть з меню **Демонстрація** → **Демонстрація**, натисніть на кнопку **Демонстрація** або натисніть **F5**.

2. Для переміщення між слайдами можна використовувати стрілки на клавіатурі, клавішу **Пробіл** і мишу.

3. Коли ви закінчите перегляд останнього слайда, з'явиться повідомлення: «Для виходу з презентації клацніть ...». Клацніть мишею або натисніть будь-яку клавішу для завершення демонстрації.

5.9 ЕЛЕКТРОНІ ТАБЛИЦІ

5.9.1 Призначення електронних таблиць

Електронна таблиця (табличний процесор) — програмний засіб, що призначений для обробки та аналізу числових, текстових та інших типів даних, що впорядковані у вигляді таблиці.

Основними функціями табличного процесору є

- ввід, редагування і зберігання табличних даних,
- обробка даних за допомогою математичних виразів,
- побудова графіків функцій і діаграм із застосуванням табличних даних,
- зручне форматування табличних даних (сортування, фільтрація),
- створення невеликих баз даних за допомогою зведених таблиць та ін.

Найбільш розповсюдженими табличними процесорам на цей час є

- MS Excel (MS Office, комерційний, ОС Windows).
- OO Calc (OO.org, вільно розповсюджуваний, кросплатформений).
- Gnumeric (вільно розповсюджуваний, кросплатформений).

За основними можливостями і принципами побудови інтерфейсу ці програми майже однакові.

Розглянемо більш ретельно інтерфейс табличного процесора у випадку OO Calc.

5.9.2 Інтерфейс табличного процесора

Табличний процесор представляє собою графічне середовище, що має наступні основні елементи

- **Головне меню** (об'єднує усі доступні в табличному процесорі функції).
- **Панель інструментів** (містить найбільш вживані функції, які доступні у вигляді кнопок).

- **Панель введення формул** (для введення формул, навігації, швидкого доступу до вживаних функцій).
- **Робоче поле** (служить для введення формул, зберігання та форматування даних).
- **Панель аркушів** (містить елементи перемикання активного аркуша).
- **Рядок стану** (відображає стан програми).

Розглянемо деякі найбільш важливі елементи табличного процесора.

5.9.2.1. Панель формул.

Панель формул представляє собою сукупність наступних елементів

- **Випадаючий список “Область аркуша”** (дозволяє переміщатися по робочому полю).
- **Кнопка Майстер функцій** (викликає діалог майстра створення функцій).
- **Кнопка Сума** (виконує автоматичне сумування виділених даних).
- **Кнопка =** переводить панель і активну комірку в стан введення формул.
- **Вікно введення функції або даних** (дозволяє редагувати дані або функцію в активній комірці).

5.9.2.2. Робоча книга. Це найбільш важлива частина табличного процесора. За допомогою цього елемента виконуються усі основні функції по обробці даних.

Робоча книга представляє собою сукупність декількох аркушів, кожний з котрих має матричну структуру. Тобто, кожний лист розбивається на сукупність рядків і стовпчиків.

Ця сукупність організує структуру із комірок. Кожна комірка має декілька власних атрибутів:

- адреса,
- тип даних,
- формат,
- оформлення та ін.

Рядки позначаються за допомогою арабських чисел від 1 до 65536.

Стовпчики позначаються латинськими літерами від A до AMG

5.9.2.3. Типи адресації. Кожна комірка робочого поля має власну унікальну адресу на аркуші. Наприклад, перша комірка в першому рядку буде мати адресу A1, друга - B1 і так далі до AMG1. В другому рядку комірки будуть мати адреси A2, B2, ..., AMG2. Таким чином, для кожної комірки можна задати адресу у вигляді: Лист.СтовпчикРядок.

Існує декілька типів адресації комірок робочого поля

- Відносна адресація.
- Абсолютна адресація.
- Змішана адресація.

Роглянемо детальніше кожний з них.

5.9.2.4. Відносна адресація. Ця адресація застосовується по-умовчанню. Адреса комірки в загальному випадку виглядає як Лист.СтовпчикРядок. Цей тип адресації можливо використовувати при операціях автозаповнення, переносу формул і вставки додаткових стовпчиків і рядків перед формулою. При цьому виконується автоматична корекція адрес комірок в формулі. Приклади:

A1, B2, Лист1.A10, F10:G20

5.9.2.5 Абсолютна адресація. Ця адресація використовується в тих випадках, коли необхідно запобігти змін адрес комірок (і рядка, і стовпчика) при операціях автозаповнення та переносу формул. В загальному випадку змішана адресація виглядає як Лист.\$Стовпчик\$Рядок. Тобто, перед номером рядка і стовпчика додається знак "\$".

\$A\$1, Лист1.\$A\$10, \$F\$10:\$G\$20

Слід зазначити, що корекція адрес комірок зі змішаною адресацією все-таки проводиться, коли додаються і видаляються стовпчики або рядки перед комітками, що містять формули.

5.9.2.6. Змішана адресація. Ця адресація використовується в тих випадках, коли необхідно запобігти змін компонентів адрес комірок (або рядка, або стовпчика) при операціях автозаповнення та переносу формул. В загальному випадку змішана адресація виглядає як Лист.\$СтовпчикРядок або

Лист.Стовпчик\$Рядок". Тобто, перед номером рядка або стовпчика додається знак "\$".

\$A1,A\$1, Лист1.\$A10, F\$10:\$G20

Слід зазначити, що корекція адрес комірок зі змішаною адресацією все-таки проводиться, коли додаються і видаляються стовпчики або рядки перед комітками, що містять формули.

5.9.2.7. Діапазони комірок. Для запису адрес декількох комірок в одному виразі використовують діапазони. Вони можуть бути зв'язані та незв'язані.

Зв'язані діапазони представляють прямокутних. Адреса цього діапазону записується як адреса лівого верхнього кута і, через двокрапку, адреса правого нижнього кута.

A6:H8, H2:G4, Лист1.\$A10:Лист1.\$B12

Незв'язані діапазони можуть мати довільну форму. Для виділення незв'язаних діапазонів можна використовувати ліву кнопку "миші" і одночасно натиснену клавішу **Ctrl**. Адреса такого діапазону записується як сукупність зв'язаних діапазонів через крапку з комою:

A1:C3;B7:G8, A1;C2;H7:G9

5.9.2.8. Формати даних. Кожна комірка в табличному процесорі може мати один із наступних типів даних

- **Числовий** - для основного відображення чисел. У його налаштуваннях можна задати кількість зображуваних знаків після коми, застосування роздільник груп розрядів, а також спосіб відображення негативних чисел.
- **Відсотковий** - для відображення числа зі знаком відсотка.
- **Грошовий** - для позначення грошових значень. При цьому порядок із значенням відображається символ грошової одиниці. Грошовий формат за замовчуванням визначається регіональними параметрами операційної системи.
- **Дати** - для відображення дати у вигляді числа (згідно з типом і мови - місцю розташування).

- **Часу** - для відображення часу у вигляді числа (згідно з типом і мови - місцю розташування).
- **Науковий** - для відображення числа в експоненційному виді.
- **Дробовий** - для відображення числа у вигляді дробу в відповідно з заданим типом дробу.
- **Логічний** - для використання логічних функцій.
- **Текстовий** - для відображення введених у клітинку даних у вигляді тексту. При цьому і текст, і числа відображаються так само, як були введені.

5.9.2.9. Створення формул. Розглянемо деякі визначення математичних термінів в контексті табличних процесорів.

Формула - сукупність операндів і дій над ними (операторів), що обчислюється в деякій комірі.

Операнд - окремий елемент даних (число, функція, посилання).

Функція - заздалегідь створена (вбудована) або користувацька формула, що виконує обчислення для заданого аргументу.

Оператор - позначка операції, яку необхідно виконати над операндом (операндами).

До списку операторів, що дозволено використовувати в табличному процесорі входять такі:

Арифметичні ("+", "-", "*", "/", "^");

Логічні ("=", "<", ">", "<=", ">=");

Текстові ("&"(об'єднання текстових фрагментів));

Посилання (":", ";")

Формули можна вводити як вручну, так і з допомогою майстра функцій.

При ручному створенні функцій необхідно

1. Виділити комірку, де буде знаходитися формула.
2. Ввести знак "=".

3. Ввести операнди і оператори.

4. Натиснути <Enter>.

Після цього на місці формули буде значення, що є результатом обчислення формули. Відредагувати формулу можна знову виділивши потрібну комірку і натиснувши **F2** або подвійним кліком "миші".

Майстер функцій дозволяє спростити створення складних формул. Для використання майстра функцій треба:

1. Виділити комірку, де буде знаходитися формула

2. Натиснути кнопку "Майстер функцій".

3. У вікні "Майстра функцій" виконати введення формули, вибираючи вбудовані функції в лівій частині вікна і заповнюючі їх аргументи в правій частині.

4. В процесі створення формули буде підраховуватися (якщо це можливо) результат обчислення.

5. Після введення формули треба натиснути кнопку "ОК".

Після цього на місці формули буде значення, що є результатом обчислення формули. Відредагувати формулу можна знову виділивши потрібну комірку і натиснувши **F2**, подвійним кліком "миші", або знову викликавши "Майстер функцій".

5.9.2.10. Створення графіків і діаграм.

Діаграма - наглядне представлення числової інформації в потрібному користувачеві вигляді.

Діаграми в табличному процесорі можуть складатися з багатьох елементів. Наведемо призначення основних елементів діаграми.

Ряд даних - набір пов'язаних між собою даних, що розміщені на діаграмі. Кожному ряду може відповідати свій колір і геометрична фігура. На діаграмі може бути декілька рядів даних одночасно.

Вісі координат - вісь даних (Y) і вісь категорій (X).

Легенда - перелік рядів даних з позначенням їх кольорів.

Область побудови - область обмежена осями координат

Область діаграми - область, що включає в себе всі елементи діаграми.

Маркер - позначка однієї точки на діаграмі однією із геометричних фігур.

Підписи даних - додаткова інформація про дані.

Підписи поділів - підписи значень на осях даних і категорій.

Існує можливість створювати наступні види діаграм:

Гістограма - лінійчата діаграма з вертикальними рядами, прямо пропорційними значенням даних. Вісь X використовується для відображення категорій, а вісь Y - для значень кожної категорії.

Лінійчата - діаграма з горизонтальними рядами, прямо пропорційними значенням даних. Вісь Y використовується для відображення категорій.

Кругова діаграма - діаграма з циклічними секторами загального кола. Довжина дуги або площа кожного сектора пропорційна значенням даних.

Область - діаграма з відображенням значень у вигляді точок на осі Y, з'єднаних лінією. Область між кожними двома лініями виділена окремим кольором. Вісь X використовується для відображення категорій.

Лінія - діаграма з відображенням значень у вигляді точок на осі Y. Вісь X використовується для відображення категорій. Значення Y кожного ряду даних можуть бути з'єднані лінією.

Діаграма XY - діаграма з відображенням значень у вигляді точки в двовимірній системі координат. У своїй базовій формі ґрунтується на одному ряді даних, що складається з імені, списку значень *x* і списку значень *y*. Ця діаграма відображає відносини чисельних значень у декількох рядах даних, або ж показує дві числові групи як один ряд координат X і Y. На діаграмі XY значення двох осей об'єднуються в єдину точку даних і поміщаються з нерівними інтервалами, або кластерами.

Сітчаста - діаграма з відображенням значень у вигляді точок, з'єднаних лініями в сітку.

Біржова діаграма - діаграма із заданим розташуванням рядів. Дані для відображення тенденції ринку відповідно до ціни відкриття, закриття, мінімуму і максимуму.

Стовпці та лінії - комбінована діаграма на основі гістограми і лінійчатої діаграми.

Побудова діаграм виконується за допомогою "Майстра діаграм".

В ньому покроково заповнюється необхідна інформація для побудови діаграми.

На першому кроці вибирається тип діаграми у відповідності до потрібного результату. На другому кроці задаються діапазони комірок, звідки буде братися інформація для побудови рядів даних. На третьому кроці формуються ряди даних і налаштовуються їхні властивості. На четвертому кроці визначаються додаткові елементи діаграми, які треба відображати разом з даними (легенду, підписи даних тощо).

Після додавання діаграми в документ, її можна вільно переміщати "мишею" за область діаграми, змінювати параметри діаграми, додавати або видаляти

Контрольні питання і завдання

1. Що таке офісне програмне забезпечення?
2. Які основні функції має офісне ПЗ?
3. Чи входять до офісного ПЗ файлові менеджери, архіватори, антивірусні пакети?
4. Що таке текстовий процесор?
5. Чи може текстовий процесор обробляти графічну, аудіо- та відеоінформацію?
6. Чому табличний процесор так називається?
7. Чим відрізняється редактор презентацій від графічного редактора?
8. Які основні офісні пакети ви знаєте? В чому їх недоліки і переваги?
9. Чи може текстовий процесор OpenOffice.org Writer працювати з форматами документів MS Office Word? А навпаки?
10. Що ми отримаємо, якщо введемо в редакторі формул OpenOffice.org Math наступний рядок: $S = \sqrt[3]{x^2_1 + y^2_1 - 10}$
11. $\sqrt{\frac{x+y}{x-y}}$ right none?
12. Що означає запис $S = \sum_{i=1}^{\infty} \{1/i\}$ в редакторі формул OpenOffice.org Math?
13. Який основний елемент презентації?
14. Які основні типи адресації в електронних таблицях ви знаєте?
15. Як зміниться формула $=A\$1^2 + \$B2^2$, якщо перед коміркою з формулою вставити стовпчик?
16. Які формати даних можуть зберігатися в клітках електронної таблиці?

17. Як побудувати графік функції на заданому проміжку із заданим кроком в електронній таблиці?

6 РОЗВ'ЯЗАННЯ ПРИКЛАДНИХ ЗАДАЧ З ВИКОРИСТАННЯМ ЕОМ

6.1 ТЕХНОЛОГІЯ ПІДГОТОВКИ І РОЗВ'ЯЗАННЯ ЗАДАЧ НА ЕОМ

Весь процес підготовки і розв'язання задач з використанням комп'ютерної техніки умовно можна поділити на наступні етапи:

1. Постановка задачі.
2. Формалізація задачі.
3. Вибір методу розв'язання задачі.
4. Розробка алгоритму.
5. Програмування.
6. Тестування і налагоджування програми.
7. Обчислення, обробка й аналіз результатів.

Розглянемо коротко зміст кожного з цих етапів.

6.1.1 Постановка задачі

На цьому етапі розкривається й формулюється основна суть задачі, вибирається загальний підхід, з'ясовуються умови (початкові, крайові й ін.), приймаються допущення, розрахункові схеми, визначаються вихідні дані, конкретизуються дані, які підлягають визначенню. В одних випадках постановка задачі виконується просто, в інших - на це можуть піти місяці роботи. Іноді на цьому етапі можуть допомогти деякі питання, якщо на них спробувати чітко сформулювати відповіді. Наприклад:

- Що дано?
- Яких даних не вистачає?
- Чи є якісь дані марними?
- Що потрібно знайти?

- Як одержати рішення?
- Які результати і в якому вигляді повинні бути отримані?
- Які зроблені недогляди?

Можливо, що відповівши на якусь частину питань, для більш повної й коректної постановки задачі деякі питання доводиться ставити повторно.

6.1.2 Формалізація задачі

Звичайно на цьому етапі задача описується у вигляді рівнянь, математичних виразів, формул, умов і т. ін. Це важливий процес у рішенні задачі, тому що він істотно впливає на всі інші етапи. Приступаючи до побудови математичної моделі, варто спробувати знайти відповіді на два основних питання: які математичні структури найбільше підходять для побудови математичної моделі й чи існують рішення аналогічних задач.

Зробивши вибір математичної структури, варто сформулювати задачу в термінах відповідних математичних об'єктів і встановити логіко-математичний зв'язок між всіма видами розглянутих у задачі даних. Формалізований опис задачі повинен бути як можна більше простим і в той же час у рамках необхідної точності правильно передавати властивості досліджуваного об'єкта.

Друге питання є основним, тому що при відповіді на нього часто перший виявляється зайвим. Більшість розв'язуваних на практиці задач є модифікаціями раніше вирішених.

На закінчення необхідно відзначити, що математична модель дозволяє звести дослідження реального об'єкта до пошуку рішення математичної задачі, тобто дозволяє використати для його вивчення добре розроблений математичний апарат із застосуванням засобів обчислювальної техніки.

6.1.3 Вибір методу розв'язання задачі

Рішення задачі можливо, як правило, декількома методами. Аналітичні методи пошуку рішень мало придатні для прикладних задач, розв'язуваних на ЕОМ, тому в основному використовуються чисельні методи одержання наближеного рішення. Застосовуваний метод повинен бути повністю схематизований (однозначно визначений кожний крок його чисельної реалізації) і бажано, щоб він хоча б частково, задовольняв наступним вимогам:

- необхідна точність обчислень;

- швидкість одержання результату;
- програмна підтримка (можливість використання раніше розроблених або стандартних програм);
- ступінь універсальності;
- обмеження пам'яті.

Найчастіше заставою вдалого вибору методу є досвід, а основним принципом - принцип безпосереднього рішення.

6.1.4 Розробка алгоритму

Чисельний метод в основному не є програмованим алгоритмом. Комп'ютер може виконувати тільки арифметичні дії й приймати найпростіші кількісні рішення, тому весь обчислювальний процес повинен містити лише такі операції, які є здійсненними на обчислювальних пристроях. Для чисельної реалізації математичної моделі варто створити алгоритм, тобто чітку послідовність приписів, що визначають рішення поставленої задачі за допомогою кінцевого числа операцій ЕОМ. Алгоритм повинен по можливості будуватися рекурсивно, тобто з відносно невеликих складових частин, які неодноразово виконуються для різних значень аргументів; бути змінюваним в залежності від набору варіантів вихідних даних, оскільки встановити границі зміни цих даних не завжди представляється можливим.

6.1.5 Програмування для ЕОМ

Програмування полягає у викладі розробленого алгоритму мовою, що може бути зрозумілою ЕОМ. Для розв'язання прикладних інженерних задач звичайно використовують алгоритмічні мови високого рівня.

6.1.6 Тестування і налагоджування програми

Допущені в процесі алгоритмізації та програмування помилки на етапі налагоджування програми повинні бути виявлені і виправлені, сама програма ретельно перевірена й випробувана на тестових завданнях. Цей етап носить послідовний характер доведення програми і є одним з найбільш трудомістких етапів.

6.1.7 Обчислення, обробка й аналіз результатів

На цьому етапі безпосередньо виконуються обчислення на ЕОМ при різних варіантах вихідних даних, приводяться обробка й аналіз результатів обчислень, їхня інтерпретація.

Контрольні питання і завдання

1. Наведіть основні етапи розв'язання задач за допомогою ЕОМ.
2. Навіщо застосовується математична модель при рішенні задачі?
3. Що відбувається на етапі побудови алгоритму?
4. Що відбувається, коли знайдені логічні помилки в алгоритмі?
5. Як називається процес доведення програми до робочого стану?
6. Що відбувається на останньому етапі вирішення задачі?

7 ОСНОВИ АЛГОРИТМІЗАЦІЇ

7.1 ВИЗНАЧЕННЯ АЛГОРИТМУ

Алгоритмом називається певна послідовність дій, що однозначно приводить до рішення поставленої задачі.

Опис алгоритму повинний бути досить повним, урахувати всі можливі ситуації, які можуть виникнути при його реалізації. Запис алгоритмів може виконуватись різними способами. Найпоширенішими є наступні три способи: словесний, графічний і псевдокод.

При словесному способі запис алгоритму не формалізується, представляється у вигляді тексту - кінцевого набору приписів, правил, команд і т.д. В алгоритмі крім звичайних слів можуть використатися спеціальні символи, формули. Словесний спосіб звичайно використовується для нескладних алгоритмів, орієнтованих в основному на виконавця - людину.

Приклад 7.1 Записати словесним способом алгоритм обчислення дійсних коренів квадратного рівняння

$$Ax^2 + Bx + C = 0.$$

Алгоритм

1. Задати значення коефіцієнтів A, B, C .
2. Обчислити значення дискримінанта $D = B^2 - 4AC$.
3. Якщо дискримінант негативний перейти до пункту 9.
4. $D := \sqrt{D}$.
5. Обчислити $x_1 = \frac{-B + D}{2A}$.
6. Обчислити $x_2 = \frac{-B - D}{2A}$.

7. Вивести результат обчислення коренів x_1 і x_2 .
8. Перейти до пункту 10.
9. Вивести повідомлення “Дійсних коренів немає”.
10. Закінчити роботу.

Псевдокод є частково формалізованою мовою, що представляє собою систему позначень і правил, призначену для запису алгоритмів. Використовувані в псевдокоді конструктивні елементи звичайно властиві мовам програмування, що спрощує перехід від алгоритму до комп'ютерної програми. Однак відсутність строгих правил запису команд, єдиного визначення псевдокоду допускають деяке свавілля в описі алгоритмів.

Приклад 7.2 Записати псевдокодом алгоритм обчислення дійсних коренів квадратного рівняння

Алгоритм

алгоритм *корінь_рівняння*;

початок

ввод (A, B, C);

$D := B^2 - 4AC$;

якщо $D < 0$

то вивід ('Дійсних коренів немає')

інакше

початок

$D := \sqrt{D}$;

$x_1 := (-B + D) / (2A)$;

$x_2 := (-B - D) / (2A)$;

вивід (x_1, x_2)

кінець

все

кінець

Для подання алгоритмів рішення наукових і інженерних задач найчастіше використається найбільш простий і наочний спосіб - графічний. У цьому випадку алгоритм оформляється у вигляді схем, які являють собою послі-

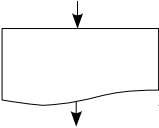
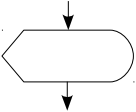
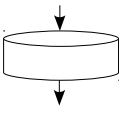

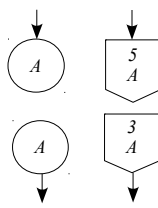

довність блоків. Ці блоки називаються символами дій. Символи дій з'єднуються лініями потоку інформації. У випадку неоднозначного сприйняття лінії забезпечуються стрілками, що уточнюють напрямок потоку інформації. Лінії потоку визначають черговість виконання блоків і зв'язок між ними.

Кожний символ дії являє собою геометричну фігуру, у контур якої вписана відповідна дія або кілька дій. Перелік і найменування символів, їхня форма й розміри, правила оформлення блок-схем алгоритмів регламентуються міжнародними стандартами ІСО 2636-73 і ІСО 1028-73. Деякі найбільш часто використовувані символи дій для опису алгоритмів наведені в таблиці 7.1.

Таблиця 7.1 Символи і їхнє графічне подання в блок-схемах алгоритмів

Найменування	Позначення	Призначення
Процес		Дія, обчислювальна операція або група операцій
Рішення		Розгалуження в алгоритмі за умовою
Модифікація		Операція повторення - заголовок циклу i – параметр циклу (ПЦ); $k1, k2$ – відповідно початкове та кінцеве значення ПЦ; $k3$ — величина зміни ПЦ
Зумовлений процес		Підключення окремо сформованих алгоритмів-програм, стандартних підпрограм
Введення-виведення		Загальне позначення вводу або виводу

Продовження таблиці 7.1

Найменування	Позначення	Призначення
Документ		Вивід результатів на папір
Дисплей		Ввід з клавіатури й вивід на екран
Магнітний диск		Операції вводу-виводу з магнітним диском
Пуск-зупин		Початок або кінець алгоритму, вхід у підпрограму або вихід з неї
З'єднувачі		Розрив ліній потоку на сторінці, на різних сторінках відповідно
Коментар		Коментар

Приклад 7.3. Записати графічним способом алгоритм обчислення дійсних коренів квадратного рівняння

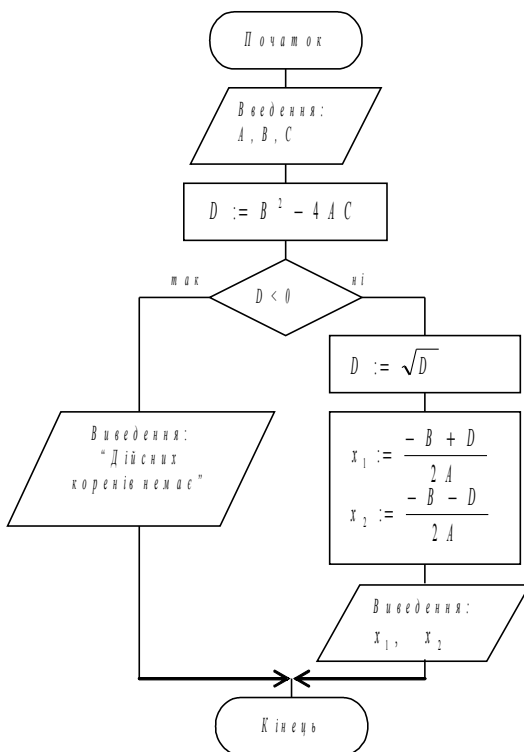


Рисунок 7.1. Схема алгоритму обчислення коренів квадратного рівняння.

7.2 ВЛАСТИВОСТІ АЛГОРИТМУ

Кожний алгоритм розробляється під конкретного виконавця. Виконавцем алгоритму можуть бути людина, автомат, комп'ютер і т. ін. Характерним для алгоритму є наявність наступних властивостей:

- зрозумілість - для виконавця алгоритму;
- детермінованість - однозначність результату при заданих вихідних даних;

- дискретність - можливість подання алгоритму у вигляді послідовності окремих виконуваних, логічно завершених дій-кроків;
- результативність - після виконання кінцевого числа кроків алгоритму одержання конкретного результату - або рішення задачі, або неможливість подальшого продовження алгоритму;
- точність - точно визначений порядок проходження дій-кроків від початку до завершення виконання алгоритму;
- масовість - можливість рішення безлічі однотипних задач шляхом варіювання вихідних даних у певних межах.

Розробка алгоритму - складна творча робота, що звичайно припускає наявність у розроблювача високої кваліфікації, глибоких знань і навичок. Відзначені вище властивості дозволяють процес рішення часом дуже складних задач звести до послідовності дій, виконання яких стає посильним практично для будь-якого навіть такого, що не володіє знаннями у відповідній області виконавця. Алгоритм повинен будуватися по можливості рекурсивно, тобто з відносно невеликих складових частин, які неодноразово реалізуються для різних наборів значень і бути змінюваним залежно від набору вихідних даних. Знання границь зміни цих даних не завжди можливо.

7.3 СТРУКТУРА АЛГОРИТМУ

Будь-якої складності алгоритм можна представити у вигляді послідовності функціональних блоків, що позначають певні дії по обробці інформації. Схематично блоки зображуються у вигляді прямокутників, що мають один вхід і один вихід. Усередині блоків вказуються їхні умовні імена, що звичайно визначають їхнє призначення, або записуються відповідні дії (Рисунок 7.2).

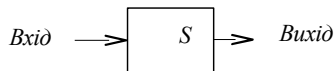


Рисунок 7.2. Функціональний блок.

В основу структурного програмування покладені принципи системного

підходу при розробці, тестуванні, оформленні й експлуатації алгоритмів і програм. Суть структурного програмування в основному визначається теоремою про структурування. У ній затверджується, що якої би складності не була задача алгоритм її рішення в остаточному підсумку завжди може бути представлений сукупністю базових функціональних блоків, кожний з яких визначає одну із трьох елементарних керуючих структур. До елементарних базових структур відносяться наступні структури: проходження (лінійна), вибору (розгалуження) і повторення (циклічна).

Однією з основних ідей структурного програмування є поетапна система розробки алгоритмів і програм з різним ступенем деталізації на кожному етапі: від самої загальної, у вигляді сукупності блоків - окремих підзадач до конкретної, з можливістю безпосередньої реалізації на ЕОМ.

7.4 ЛІНІЙНА СТРУКТУРА

Лінійною називається структура, у якій передача керування здійснюється послідовно по ланцюжку від одного функціонального блоку до наступного (Рисунок 7.3).

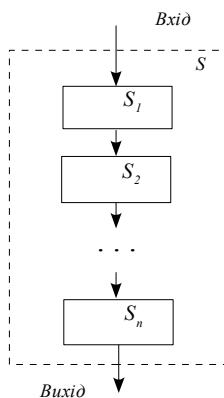


Рисунок 7.3. Схема лінійної структури.

Кожний блок, залежно від ступеня деталізації, може являти собою послідовність більше простих дій і навпаки, послідовність блоків завжди може бути замінена одним функціональним блоком.

7.5 СТРУКТУРА РОЗГАЛУЖЕННЯ

Структурою розгалуження називається структура, що забезпечує можливість вибору функціонального блоку, якому повинне бути передане керування залежно від виконання, або не виконання деякої умови (Рисунок 7.4).

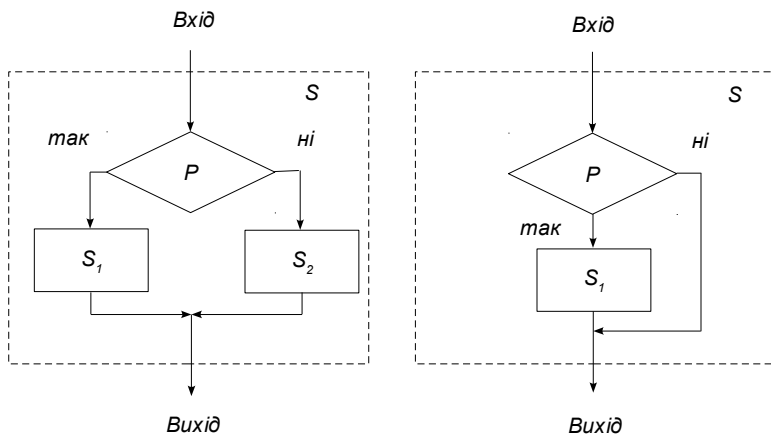


Рисунок 7.4. Схема структури розгалуження.

При вході в блок даної структури аналізується логічна умова P , значенням якого може бути або істина, або неправда. У випадку істинного значення (напрямок так), керування передається блоку S_1 , у протилежному випадку (напрямок ні) – блоку S_2 , якщо розгалуження повне й ніякої дії, якщо розгалуження неповне.

7.6 ЦИКЛІЧНА СТРУКТУРА

Циклічна структура використовується для позначення багаторазово повторюваної дії - циклу. Умова виходу із циклу може перебувати спочатку циклу (цикл із передумовою) або наприкінці його (цикл із післяумовою) (Рисунок 7.5).

Цикл виконується в такий спосіб. При вході в цикл керування передається функціональному блоку S_1 (цикл із параметром і цикл із післяумовою), потім аналізується логічна умова. Якщо вона не виконується, здійснюється повернення до блоку S_1 . Передача керування цьому блоку відбувається бага-

торазово, поки умова не стане істиною. Таким чином, у циклах з параметром i з післяумовою функціональний блок S_i виконається хоча б один раз незалежно від того, чи є умова істинною або помилковою.

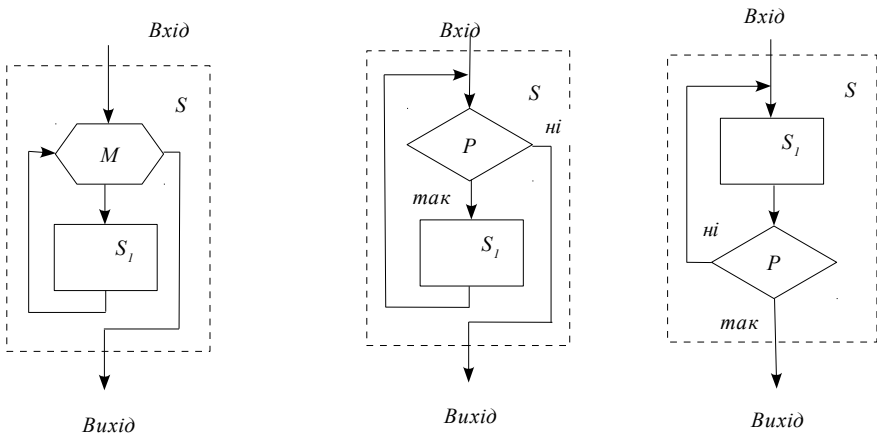


Рисунок 7.5. Схема циклічної структури.

У циклі із передумовою спочатку перевіряється умова і якщо вона істинна, то керування передається функціональному блоку S_i , після цього здійснюється повернення до умови, якщо вона не виконується – відбувається вихід із циклу. Тому в циклі із передумовою функціональний блок S_i виконуватися не буде (жодного разу), якщо при першій же перевірці умова не виконується.

7.7 ТИПОВІ ПРИЙОМИ АЛГОРИТМІЗАЦІЇ

Розглянемо типові прийоми побудови алгоритмів, що використовуються при вирішенні різних практичних задач.

7.8.1 Цикл з цілим кроком

Введемо наступні позначення:

i – параметр циклу;

k_1 – початкове значення параметра циклу;

k_2 – кінцеве значення параметра циклу;

k_3 – крок зміни параметра циклу (якщо $k_3 = 1$, то його можна в записі циклу опустити);

T_c – функціональний блок - тіло циклу, що містить одне або кілька дій.

Схема циклу з цілим кроком показана на рисунку 7.7.

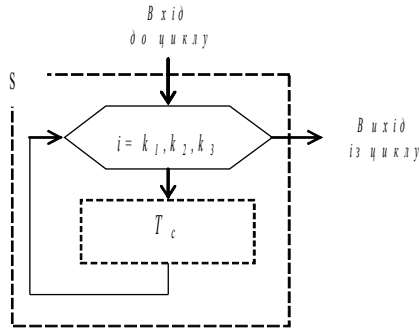


Рисунок 7.7. Цикл із цілим кроком.

Приклад 7.4. Вводиться натуральне число n . Визначити, чи є воно простим.

Зауваження: просте число – це натуральне число, більше одиниці, яке ділиться тільки саме на себе та на одиницю. Простими числами із інтервалу $(0 - 100)$ – є наступні числа: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.

1. Постановка задачі.

Дано: n – натуральне число.

Визначити: n – просте число?

Позначення: f – допоміжний прапор

$(f=0 \dots n$ – число просте,

$f=1 \dots n$ – число не просте);

$[]$ – виділення цілої частини числа;

$n \bmod i$ – залишок від ділення n на i .

2. Математична модель (умовна)

$$f := \begin{cases} 1, & \text{якщо } n \bmod i = 0; \\ 0, & \text{якщо } n \bmod i \neq 0. \end{cases}$$

3. Алгоритм

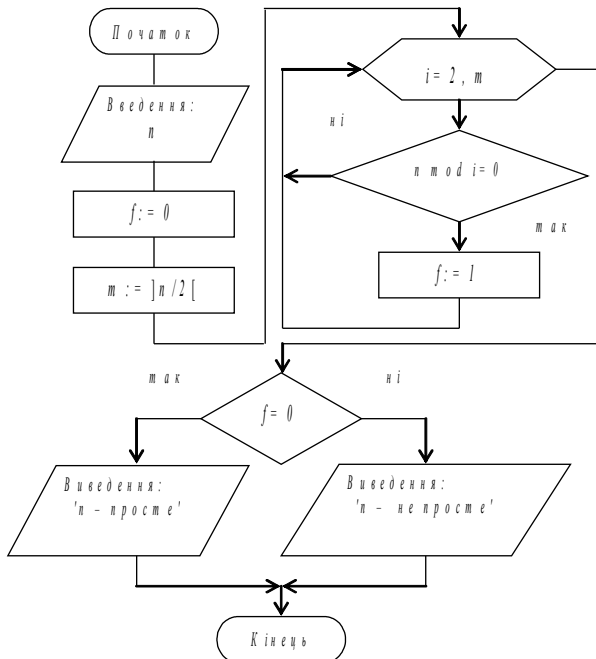


Рисунок 7.8. Схема алгоритму визначення простого числа.

7.8.2 Цикл з довільним кроком

Прийняті позначення:

- x - параметр циклу;
- x_n - початкове значення параметра циклу;
- x_k - кінцеве значення параметра циклу;
- Δx - збільшення параметра циклу;
- T_c - функціональний блок – тіло циклу, що містить одне або кілька дій.

Зміна параметра циклу - перехід від точки до точки заданого відрізка $[x_n, x_k]$, здійснюється за допомогою блоку $x := x + \Delta x$.

На рисунку 7.9 представлено два варіанти організації циклу: перший - із передумовою, другий - з післяумовою. Основне розходження цих циклів полягає в тому, що перший цикл виконується хоча б один раз незалежно від того, чи є умова істиною або помилковою.

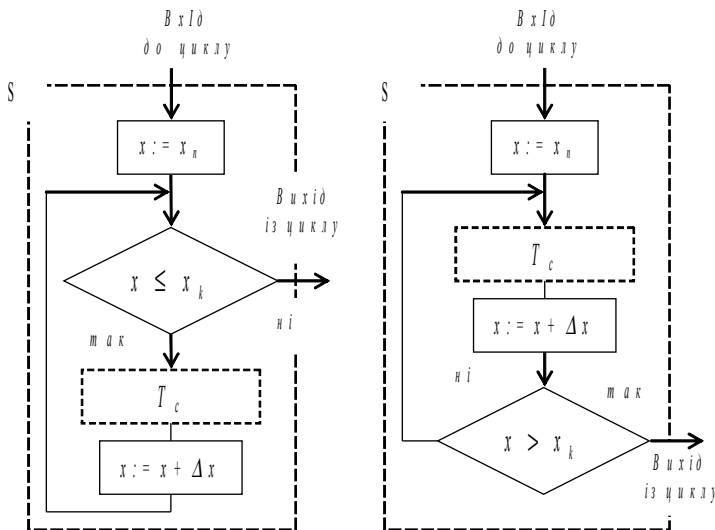


Рисунок 7.9. Схема циклів із передумовою та з післяумовою.

Приклад 7.5. Для заданого $x \in [-3, 2; 1, 5]$, $\Delta x = 0,1$ обчислити значення функції

$$z = x^3 - \frac{\operatorname{tg} x}{e^x + 1}.$$

1. Постановка задачі.

Дано: $x_n, x_k, \Delta x$.

Знайти: z .

2. Математична модель

$$z = x^3 - \frac{\operatorname{tg} x}{e^x + 1}.$$

3. Алгоритм

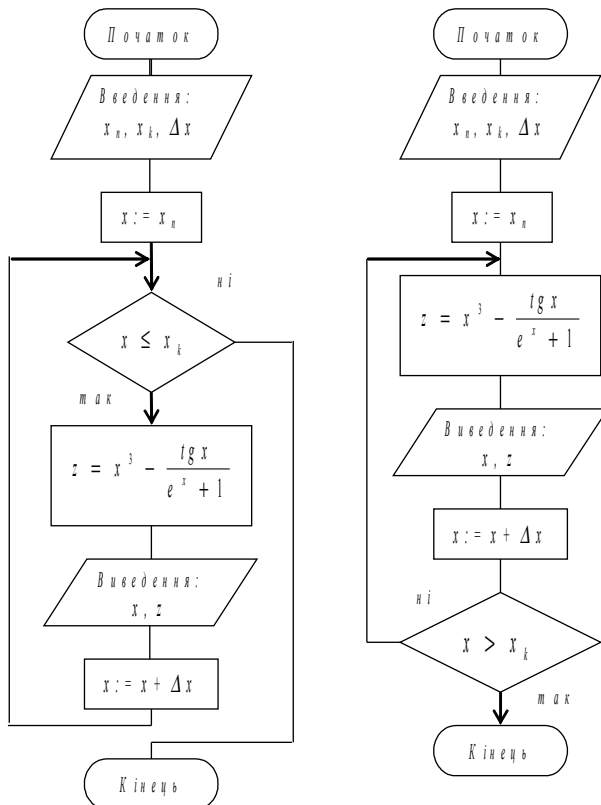


Рисунок 7.10. Схеми алгоритму із передумовою та з післяумовою.

7.8.3 Цикл зі збереженням результатів

У розглянутому вище випадку по закінченню обчислень у пам'яті комп'ютера зберігається лише останній результат. Для запам'ятовування значень у всіх точках відрізка $[x_n, x_k]$ необхідно кожний результат обчислювати в циклі як змінну з індексом (елемент масиву). Число елементів масиву визначається по формулі

$$n = \left\lceil \frac{x_k - x_n}{\Delta x} \right\rceil + 1 .$$

Приклад 7.6 . Для заданого $x \in [-3, 2; 1, 5]$, $\Delta x = 0,1$ обчислити значення функції

$$z = x^3 - \frac{\operatorname{tg} x}{e^x + 1}$$

з запам'ятовуванням результатів.

1. Постановка задачі.

Дано: $x_n, x_k, \Delta x$.

Знайти й запам'ятати: z .

2. Математична модель

$$z_i = x^3 - \frac{\operatorname{tg} x}{e^x + 1}, \quad (i = 1, 2, \dots, m), \quad \text{де} \quad m = \left\lceil \frac{x_k - x_n}{\Delta x} \right\rceil + 1;$$

3. Алгоритм

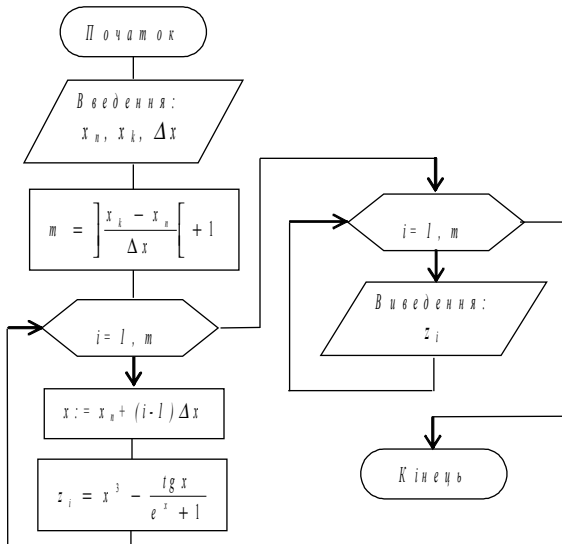


Рисунок 7.11. Схема алгоритму обчислення функції із запам'ятовуванням результатів.

7.8.4 Ітераційний цикл

Ітераційним називається цикл із невідомим числом повторень, необхідний результат у якому досягається послідовним наближенням або уточненням.

Приклад 7.7. Визначити найбільше ціле значення $k=1,2,\dots$, при якому виконується умова $\frac{e^k}{k} \leq a$, для будь-якого заданого a .

1. Постановка задачі

Дано: a .

Знайти: $\max k$.

2. Математична модель

$$y = \frac{e^k}{k} \leq a.$$

3. Алгоритм

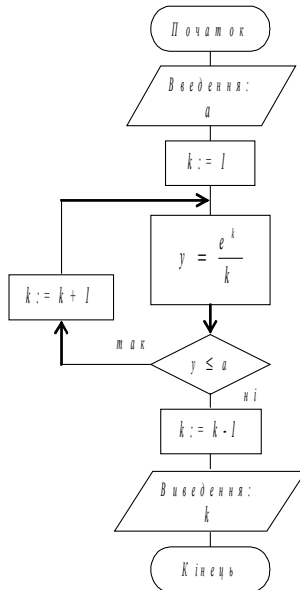


Рисунок 7.12. Схема алгоритму ітераційного циклу.

7.8.5 Вкладені цикли

Будь-який цикл усередині себе може містити один або кілька інших, внутрішніх циклів. Така структура з декількох, розташованих один в іншому, циклів називається вкладеними циклами. Цикл, що містить усередині себе інші цикли, називається зовнішнім. Параметри зовнішнього й внутрішнього циклів повинні бути різними. Змінюються вони не одночасно, тобто при кожному значенні параметра зовнішнього циклу параметр внутрішнього циклу приймає по черзі всі свої значення. Можливі схеми вкладення циклів представлені на рисунку 7.13.

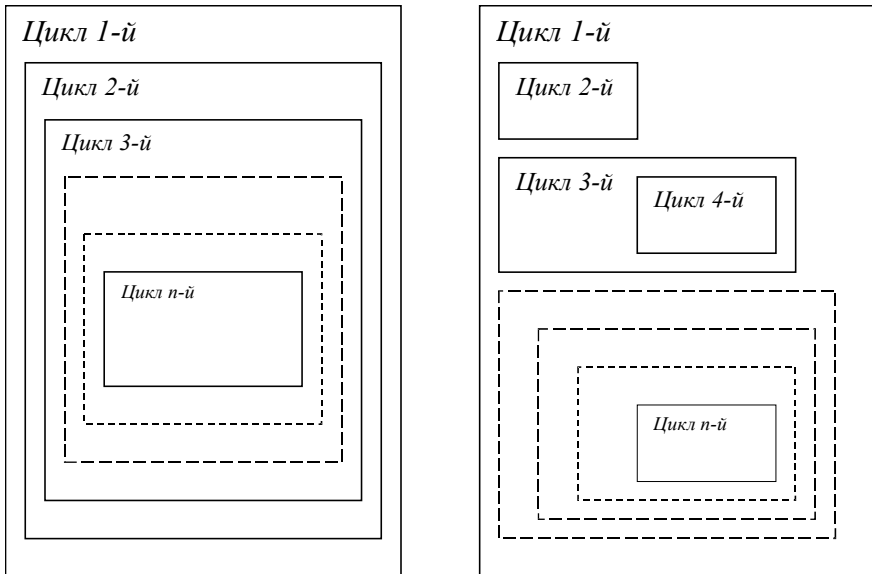


Рисунок 7.13. Можливі схеми вкладення циклів.

Приклад 7.8. Обчислити функцію $w = 2 \sin^3(a_i x) - 3 \cos(b_j x)$ при кожному значенні

$$a_i \in [a_1, a_2, \dots, a_m], \quad b_j \in [b_1, b_2, \dots, b_s],$$

для всіх $x \in [x_n, x_k]$, що змінюються з кроком Δx .

1. Постановка задачі

Дано: $m, s, a(m), b(s), x_n, x_k, \Delta x$.

Знайти: w .

2. Математична модель

$$w = 2 \sin^3(a_i x) - 3 \cos(b_j x) \quad (i=1, 2, \dots, m; \quad j=1, 2, \dots, s).$$

3. Алгоритм

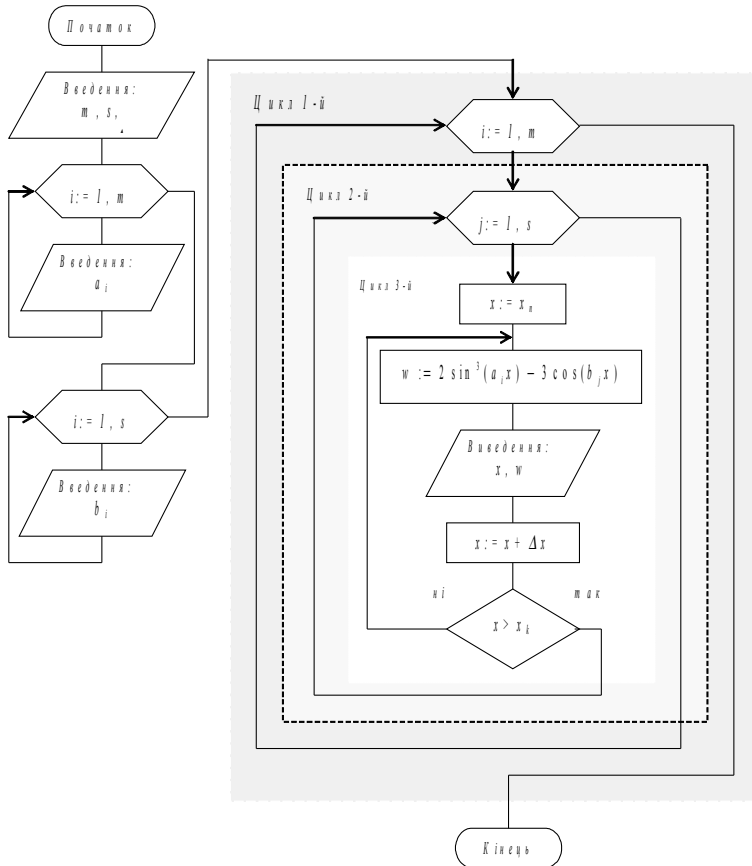


Рисунок 7.14. Схема алгоритму з використанням вкладених циклів.

7.8.6 Підрахунок кількості

Для підрахунку кількості об'єктів по заданій властивості (умові), необхідно ввести лічильник (k), установити його у початковий стан ($k=0$) і далі в циклі щораз при виконанні поставленої умови присвоювати йому нове значення ($k:=k+1$).

Приклад 7.9. Підрахувати кількість негативних значень функції

$$y = \sin^3 x - \cos x$$

для усіх $x \in [x_n, x_k]$, що змінюються з кроком Δx .

1. Постановка задачі.

Дано: $x_n, x_k, \Delta x$.

Знайти: k .

2. Математична модель $y = \sin^3 x - \cos x$.

3. Алгоритм

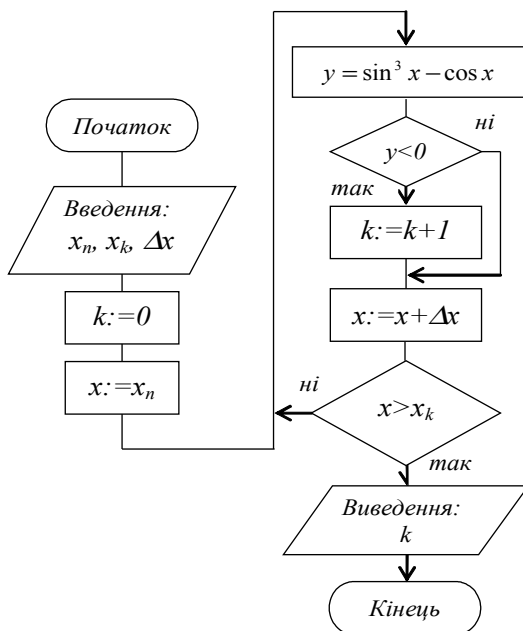


Рисунок 7.15. Схема алгоритм у підрахунку кількості негативних значень функції.

7.8.7 Обчислення добутку

Для обчислення добутку необхідно перед циклом задати початкове значення добутку рівним одиниці. Обчислення добутку здійснюється в циклі шляхом накопичення у відповідності до умовної математичної формули:

$$P := P \cdot b_i \quad (i=1,2,\dots,n),$$

де P – проміжне значення добутку, після закінчення циклу шукане;

b_i – співмножник.

Приклад 7.10. Обчислити середнє геометричне всіх позитивних елементів матриці $A_{m \times n}$.

1. Постановка задачі

Дано: $m, n, A(m, n)$.

Знайти: G .

2. Математична модель

$$G = \sqrt[k]{g_1 \cdot g_2 \cdot \dots \cdot g_k} = \sqrt[k]{\prod_{i=1}^k g_i}, \quad g_i = a_{i,j} > 0 \quad (i=1,2,\dots,k).$$

3. Алгоритм

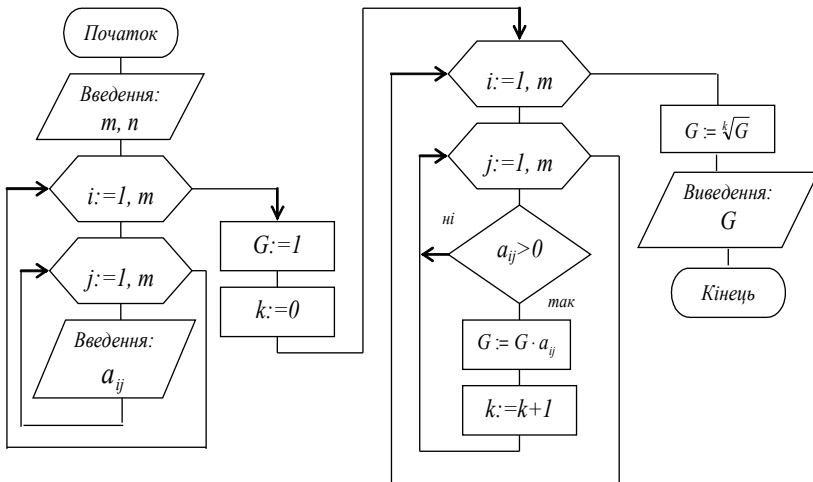


Рисунок 7.16. Схема алгоритму обчислення середнього геометричного.

7.8.8. Обчислення факторіалу

Алгоритм обчислення факторіалу $F = n!$ аналогічний алгоритму обчислення добутку. Відмінність полягає в рекурентній формулі

$$F := F \cdot i, \quad (i=1, 2, \dots, n),$$

Приклад 7.11. Обчислити число сполучень із n по m по формулі

$$C_n^m = \frac{n!}{m!(n-m)!}$$

1. Постановка задачі

Дано: m, n .

Знайти: C_n^m .

2. Математична модель

$$C_n^m = \frac{n!}{m!(n-m)!}.$$

3. Алгоритм

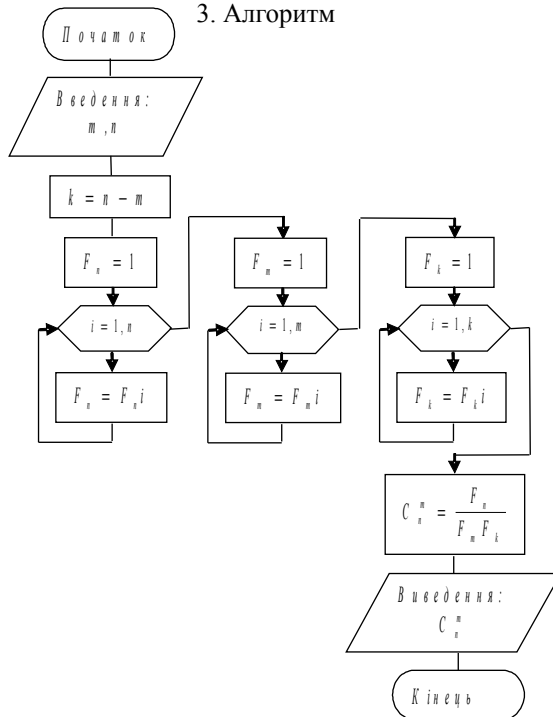


Рисунок 7.17. Схема алгоритму обчислення числа сполучень із n по m .

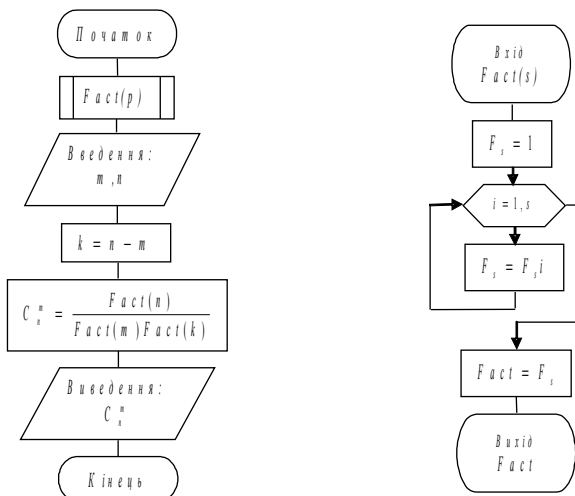


Рисунок 7.18. Схема алгоритму обчислення числа сполучень із n по m з використанням алгоритма-функції.

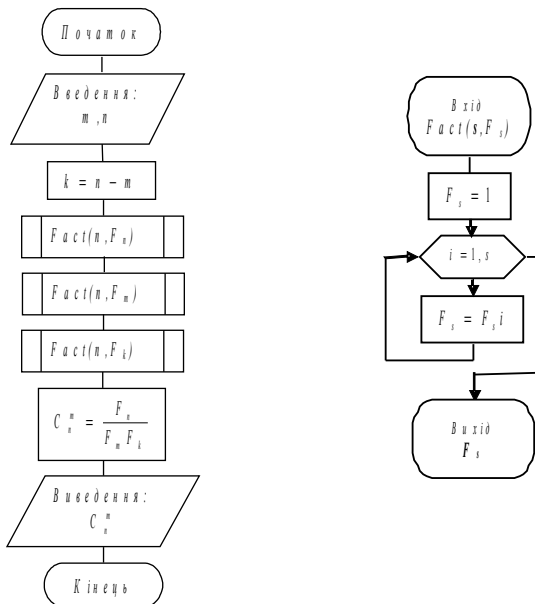


Рисунок 7.19. Схема алгоритму обчислення числа сполучень із n по m з використанням алгоритма-процедури.

Приклад 7.12. Скласти алгоритм обчислення біноміальних коефіцієнтів бінома Ньютона

$$(x+a)^n$$

1. Постановка завдання.

Дано: n .

Знайти: C_n^i ($i=0,1,\dots,n$).

2. Математична модель

$$(x+a)^n = C_n^0 x^n a^0 + C_n^1 x^{n-1} a^1 + \dots + C_n^n x^0 a^n$$

де $C_n^i = \frac{n!}{i!(n-i)!}$ ($i=0,1,\dots,n$).

3. Алгоритм

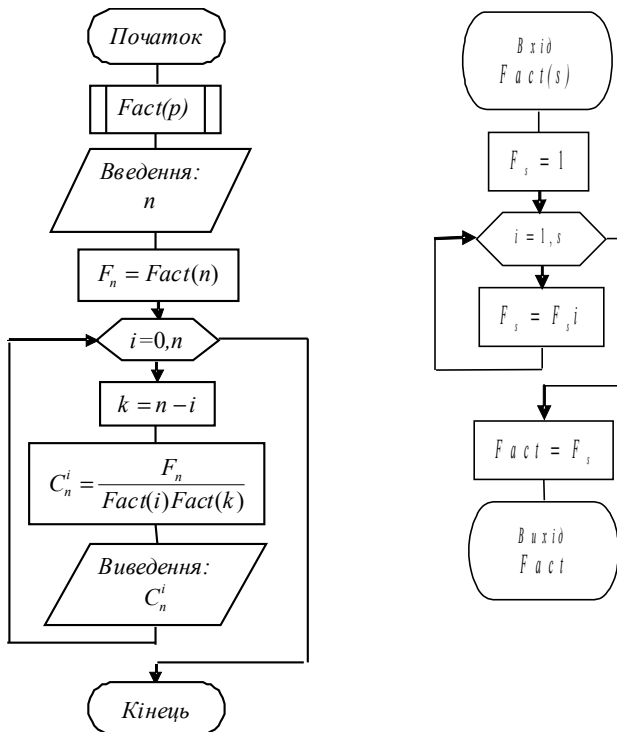


Рисунок 7.20. Схема алгоритму обчислення біноміальних коефіцієнтів з використанням алгоритма функції.

7.8.9 Обчислення суми

7.8.9.1 Сума кінцевого числа доданків.

Для обчислення суми необхідно перед циклом задати початкове значення суми, звичайно рівне нулю. Обчислення суми в циклі виконується шляхом накопичення відповідно до умовної формули

$$S := S + a_i \quad (i = 1, 2, \dots, n),$$

де S - проміжна сума, після закінчення циклу шукана;
 a_i - доданок.

Приклад 7.13. Обчислити суму позитивних елементів вектора

$$a = \{a_1, a_2, \dots, a_n\}$$

1. Постановка задачі.

Дано: $n, a(n)$.

Знайти: S .

2. Математична модель

$$S = a_1 + a_2 + \dots + a_n = \sum_{i=1}^n a_i.$$

3. Алгоритм

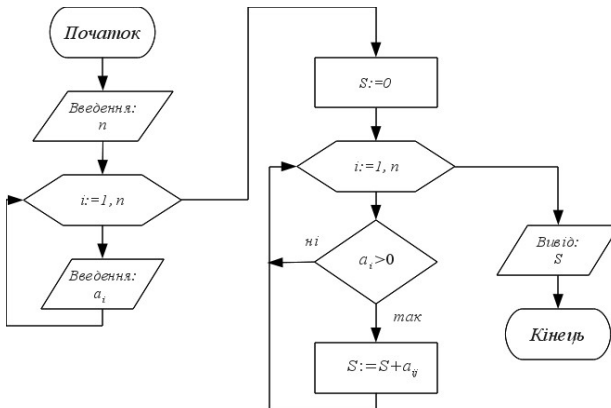


Рисунок 7.21 Схема алгоритму обчислення суми позитивних елементів вектора.

7.8.9.2 Сума ряду

Для обчислення суми членів нескінченного ряду використовується розглянутий раніше прийом накопичення суми в циклі, число повторень якого заздалегідь невідомо. Умовою виходу із циклу є досягнення необхідної точності

$$|u_n| \leq \varepsilon,$$

де u_n – значення n -го члена ряду, ε – задана точність.

Приклад 7.14. Обчислити суму ряду

$$z = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!} + \dots$$

з точністю до ε .

1. Постановка задачі

Дано: x, ε .

Знайти: z .

2. Математична модель

$$z = \sum_{n=0}^{\infty} \frac{x^n}{n!}.$$

3. Алгоритм

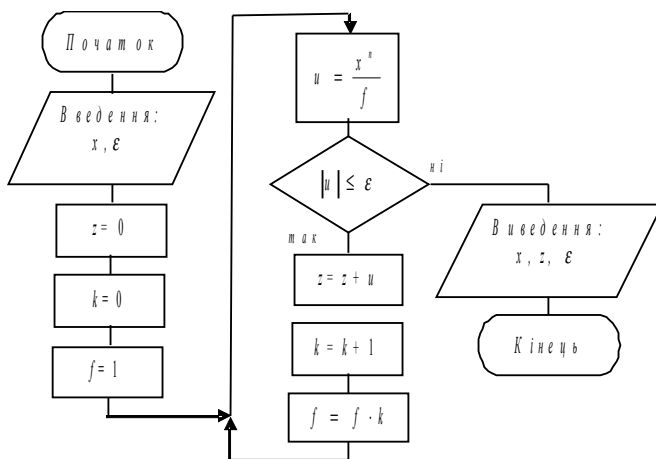


Рисунок 7.22 Схема алгоритму обчислення суми ряду.

7.8.10 Знаходження найменшого й найбільшого значень функції

Для знаходження найменшого значення функції необхідно перед циклом задати його початкове значення, рівне дуже великому числу. У циклі слід передбачити обчислення поточного значення функції й порівняння його з найменшим з попередніх значень. Якщо поточне значення виявляється менше, то його треба вважати новим найменшим значенням. У протилежному випадку найменше значення залишається колишнім. Ці дії можна описати наступною умовною математичною формулою

$$z_{\min} = \begin{cases} z, & \text{якщо } z < z_{\min}; \\ z_{\min}, & \text{якщо } z \geq z_{\min} \end{cases}$$

При знаходженні найбільшого значення його початкову величину треба брати рівної дуже малому числу. У циклі за допомогою умовної математичної формули

$$z_{\max} = \begin{cases} z, & \text{якщо } z > z_{\max}; \\ z_{\max}, & \text{якщо } z \leq z_{\max} \end{cases}$$

знаходимо найбільше значення.

Приклад 7.15. Знайти найменше значення функції

$$z = x^4 - x^3 - \sqrt{|x|}$$

для всіх $x \in [x_n, x_k]$, що змінюються з кроком Δx .

1. Постановка задачі

Дано: $x_n, x_k, \Delta x$.

Знайти: z_{\min} .

2. Математична модель

$$z = x^4 - x^3 - \sqrt{|x|}.$$

3. Алгоритм

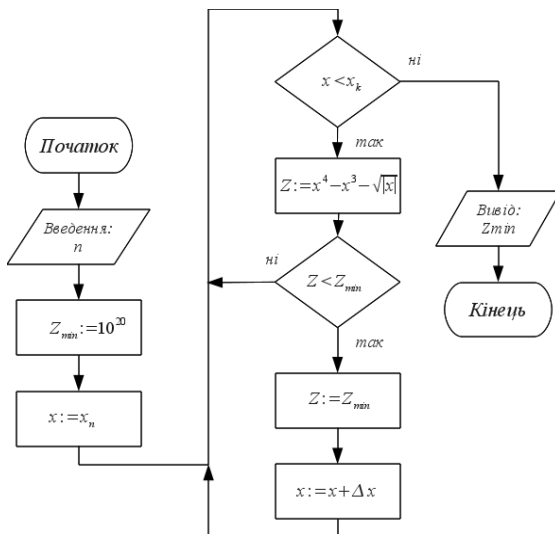


Рисунок 7.23. Схема алгоритму знаходження найменшого значення обчислюваної функції.

7.8.11 Обчислення значення полінома

Для обчислення полінома n -го ступеня

$$y = a_1 x^n + a_2 x^{n-1} + \dots + a_n x + a_{n+1}$$

зручно використати формулу Горнера

$$y = (\dots((a_1 x + a_2)x + a_3)x + \dots + x^n) + a_{n+1}.$$

При цьому, якщо прийняти наступні позначення

$$y_1 = a_1;$$

$$y_2 = a_1 x + a_2 = y_1 x + a_2;$$

$$y_3 = (a_1 x + a_2)x + a_3 = y_2 x + a_3;$$

тобто можна записати

$$y_{i+1} = y_i x + a_{i+1}.$$

Представляючи коефіцієнти полінома у вигляді масиву

$$A = \{a_1, a_2, \dots, a_{n+1}\},$$

складемо схему алгоритму обчислення значення полінома заданого ступеня.

Алгоритм:

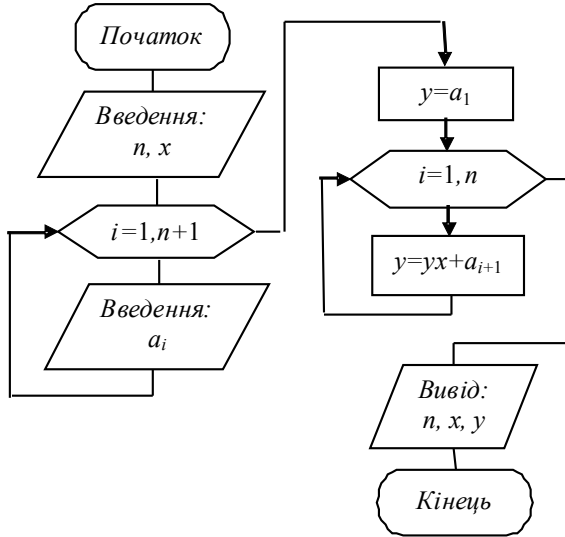


Рисунок 7.24. Схема алгоритму обчислення полінома.

Зауваження. Якщо доданок, що містить x у якому-небудь ступені відсутній, то це означає, що коефіцієнт при ньому дорівнює нулю.

Приклад 7.16. Використовуючи алгоритм обчислення полінома (рисунок 7.24) скласти алгоритм, за допомогою якого можна обчислити

$$U_v(t) = t^4 + 2t^3 + 1, \quad w(x) = \frac{P_n(x)}{Q_m(x)}, \quad z(x, y) = \frac{P_n(x)}{R_s(y)},$$

де $P_n(x) = x^7 - 4x^5 + 3x^2 + 6x - 2$;

$Q_m(x) = x^9 + 7x^6 - 4x^3 + 2x - 5$;

$R_s(y) = y^8 - 7y^5 + 6y^3 - 2$

$t=0,87$; $x=0,36$; $y=1,27$.

1. Постановка задачі.

Дано: $v=4$; $A_U=\{1, 2, 0, 0, 1\}$;
 $n=7$; $A_P=\{1, 0, -4, 0, 0, 3, 6, -2\}$;
 $m=9$; $A_Q=\{1, 0, 0, 7, 0, 0, -4, 0, 2, -5\}$;
 $s=8$; $A_R=\{1, 0, 0, -7, 0, 6, 0, 0, -2\}$;
 $t=0,87$; $x=0,36$; $y=1,27$.

Знайти: U, w, z .

2. Алгоритм

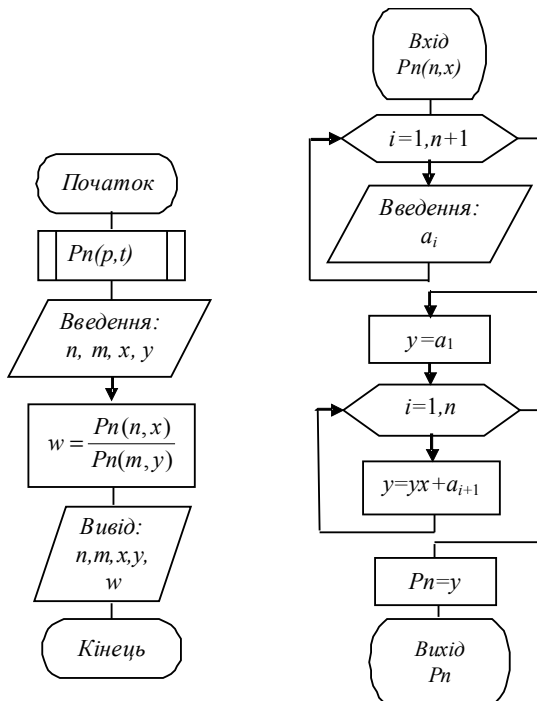


Рисунок 7.25. Схема алгоритму функцій $w(x)$ і $z(x,y)$ з використанням алгоритма функції.

3. Обчислення.

1) Обчислення значення полінома $U(t)$:

Ввести значення фактичних параметрів: $v, 1, t, t$.

Ввести значення: A_U, A_1 , де $A_1=\{0, 1\}$.

1) Обчислення значення функції $w(x)$:

Ввести значення фактичних параметрів: n, m, x, x .

Ввести значення: A_P, A_Q .

2) Обчислення значення функції $z(x, y)$:

Ввести значення фактичних параметрів: n, s, x, y .

Ввести значення: A_P, A_R .

7.8.12 Обробка масивів

До алгоритмів, за допомогою яких обробляють одномірні масиви, відносяться алгоритми визначення найбільшого або найменшого елемента, його порядкового номера (індексу), перестановка елементів у масиві, сортування елементів по убутанню значень, зростанню або по якій-небудь іншій ознаці.

7.8.12.1 Визначення мінімального й максимального елементів масиву. Для визначення мінімального елемента масиву в якості його початкового значення береться значення першого елемента. Далі в циклі за допомогою умовної математичної формули

$$\min = \begin{cases} a_i, & \text{якщо } a_i < \min; \\ \min, & \text{якщо } a_i \geq \min \end{cases}$$

визначається мінімальний елемент.

При знаходженні максимального елемента його початкову величину також слід прийняти рівною значенню першого елемента. У циклі за допомогою умовної математичної формули

$$\max = \begin{cases} a_i, & \text{якщо } a_i > \max; \\ \max, & \text{якщо } a_i \leq \max \end{cases}$$

знаходимо максимальний елемент.

Приклад 7.17. Визначити найменший елемент масиву

$$x = \{x_1, x_2, \dots, x_n\}$$

і його порядковий номер.

1. Постановка задачі.

Дано: $n, x(n)$.

Знайти: \min .

2 Алгоритм

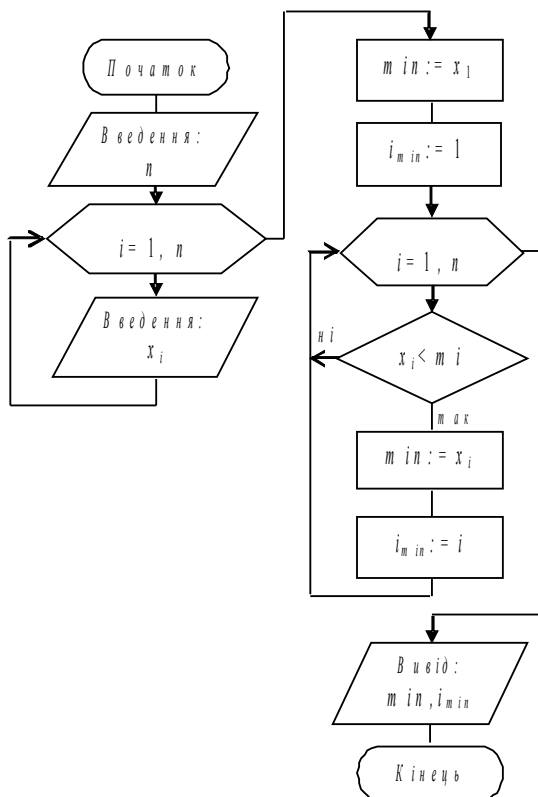


Рисунок 7.26. Схема алгоритму визначення мінімального елемента масиву.

7.8.12.2 Перестановка елементів масиву

Щоб поміняти місцями два елементи масиву необхідно:

1. Присвоїти значення першого елемента, що буде переставлятися, якій-небудь змінній, наприклад p .

2. Першому елементу, що, буде переставлятися, присвоїти значення другого елемента, що переставляється.

3. Другому елементу, що переставляється, присвоїти значення змінної p .

Приклад 7.18. У заданому одномірному масиві

$$a = \{a_1, a_2, \dots, a_n\}$$

поміняти місцями елементи з індексом t і s .

1. Постановка задачі.

Дано: $n, a(n)$.

Поміняти місцями a_t і a_s .

2. Алгоритм

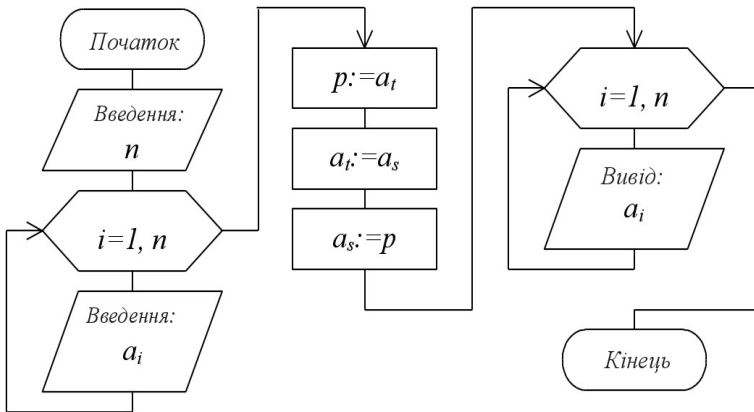


Рисунок 7.27. Схема алгоритму перестановки елементів масиву.

7.8.12.3 Упорядкування елементів одномірного масиву

Існує багато алгоритмів сортування (упорядкування) масивів. Розглянемо найбільш відомий з них - сортування за допомогою прямого вибору. Суть його полягає в наступному: Вибирається елемент заданого масиву: найменший - при впорядкуванні по зростанню й найбільший - при впорядкуванні по убутанню й шляхом перестановки з першим елементом розміщується на крайню ліву позицію. На наступному кроці в частині, що залишилася від

масиву, здійснюється пошук іншого відповідно найменшого або найбільшого елемента, що розміщується на другій позиції, і т.д.

Приклад 7.19. У заданому одномірному масиві

$$a = \{a_1, a_2, \dots, a_n\}$$

розташувати всі елементи в порядку зростання.

Алгоритм

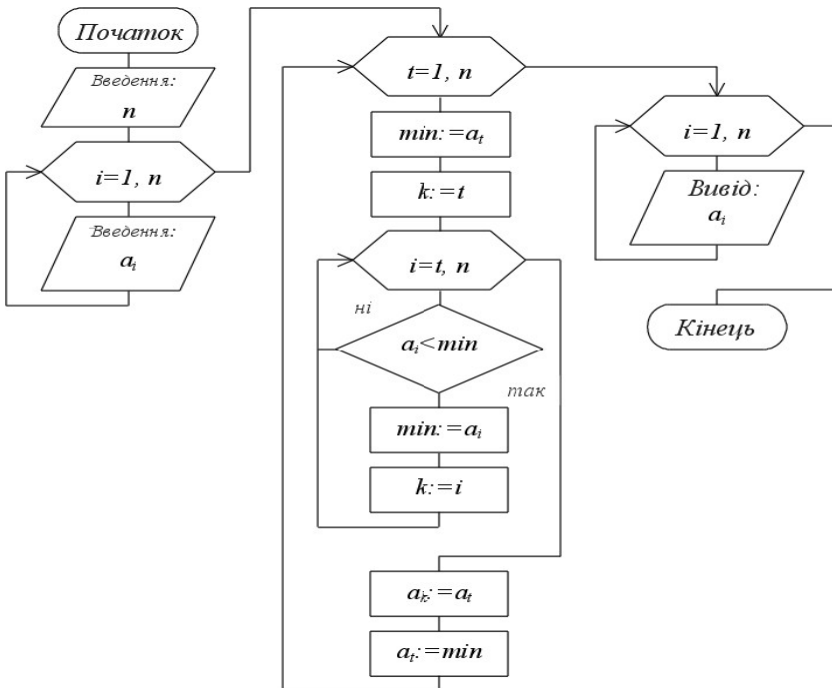


Рисунок 7.28. Схема алгоритму впорядкування елементів масиву по зростанню.

Контрольні питання та завдання

1. Наведіть визначення алгоритму.
2. Яким чином можна записати алгоритм?
3. Які основні властивості алгоритму ви знаєте?
4. Чим відрізняється детермінованість від результативності?
5. Що означає масовість алгоритму?
6. Які основні структури алгоритмів ви знаєте?
7. Які дві структури алгоритму застосовуються для розгалуження процесу виконання?
8. Які структури можна застосовувати для зациклювання виконання окремих частин алгоритмів?
9. Чим відрізняються цикл з передумовою і цикл з післяумовою?
10. Наведіть алгоритм для обчислення значень функції на заданому проміжку з вказаним кроком.
11. Яка формула застосовується для розрахування кількості кроків циклу для попереднього алгоритму?
12. Як розрахувати суму ряду, що збігається?
13. Наведіть алгоритм розрахування факторіалу числа.
14. Знайдіть середнє квадратичне від'ємних елементів і середнє геометричне додатних елементів заданого масиву.
15. Знайдіть кількість елементів матриці розміром 3×3 , що більші за її визначник.
16. Знайдіть для заданої матриці транспоновану матрицю.

8 МОВА ПРОГРАМУВАННЯ TURBO PASCAL 7.0

Алгоритмічна мова Turbo Pascal, що входить до складу професійного пакету розробки програм Borland Pascal with Objects 7.0 (BPO), являється мовою високого рівня (MBP). Реалізація програм, написаних на цій мові, здійснюється за допомогою компілятора, який має наступні основні версії:

- версія компілятора, що працює під управлінням MS DOS в реальному режимі процесора (TURBO.EXE);
- версія компілятора, що працює під управлінням MS DOS в захищеному режимі процесора (BP.EXE);
- версія компілятора, що працює під управлінням Windows (BP-W.EXE).

8.1 ОСНОВНІ ЕЛЕМЕНТИ МОВИ

Основу мови Turbo Pascal 7.0, як і будь-якої іншої алгоритмічної мови програмування, складають алфавіт, синтаксис і семантика.

8.1.1 Алфавіт мови

Безліч символів таблиці кодів ASCII, що використовуються в програмах сприймаються компілятором, утворюють алфавіт мови. Враховуючи відмінності у застосуванні символів, алфавіт мови Turbo Pascal умовно можна розділити на наступні групи:

- алфавітно-цифрову;
- спеціальних символів;
- складених символів;
- керувальних символів;
- символів обмеженого застосування.

1. Алфавітно-цифрова група

До алфавітно-цифрової групи входять рядкові (малі) і прописні (великі) літери латинського алфавіту, символ підкреслення, арабські десяткові цифри (Таблиця 8.1)

Таблиця 8.1 Алфавітно-цифрова група

Символ	Код	Символ	Код	Символ	Код	Символ	Код	Символ	Код
A	65	N	78	a	97	n	110	—	95
B	66	O	79	b	98	o	111		
C	67	P	80	c	99	p	112		
D	68	Q	81	d	100	q	113	0	48
E	69	R	82	e	101	r	114	1	49
F	70	S	83	f	102	s	115	2	50
G	71	T	84	g	103	t	116	3	51
H	72	U	85	h	104	u	117	4	52
S	73	V	86	s	105	v	118	5	53
J	74	W	87	j	106	w	119	6	54
K	75	X	88	k	107	x	120	7	55
L	76	Y	89	l	108	y	121	8	56
M	77	Z	90	m	109	z	122	9	57

Символи алфавітно-цифрової групи використовуються в основному для складання ідентифікаторів. У програмі прописні і рядкові літери не розрізняються

2. Група спеціальних символів

Ця група об'єднує символи, що виконують певні функції при побудові конструкцій мови

Таблиця 8.2 Група спеціальних символів

Символ	Код	Символ	Код	Символ	Код	Символ	Код	Символ	Код
#	35	*	42	/	47	>	62	{	123
\$	36	+	43	:	58	@	64	}	125
`	39	,	44	;	59	[91		
(40	-	45	<	60]	93	пропуск	32
)	41	.	46	=	61	^	94		

3. Група складених символів

Деякі поєднання спеціальних символів, що виконують певні функції і сприймаються компілятором як єдине ціле, утворюють групу складених символів

<= <> >= := (* *) (. .)

4. Група керувальних символів

До цієї групи входять символи, що розташовані спочатку таблиці кодів ASCII і мають коди від 0 до 31. Керувальні символи в програмах можуть використовуватися при описі рядкових і символьних констант, а також як роздільники.

5. Група символів обмеженого застосування

Обмежене застосування у мові Turbo Pascal має решта символів основної таблиці кодів ASCII

! “ % & ? \ | ~

І всі символи розширеної таблиці ASCII, що мають коди від 128 до 255 (зокрема символи російського алфавіту і символи псевдографіки). Використання цих символів можливо тільки як значення символьних або рядкових констант, а також в коментарях.

8.1.2 Синтаксис

Синтаксис – це система правил, що визначають допустимі структури і послідовності з символів алфавіту мови. За допомогою цих структур на мові програмування подаються окремі компоненти алгоритму і алгоритм в цілому.

8.1.3 Семантика

Семантика – це система правил і угод, що визначає смислове тлумачення внутрішніх структур мови, або простіше кажучи, семантикою називається зміст або сенс програми.

На основі синтаксичних і семантичних правил мови розробляються тексти програм, перевіряється коректність їх запису, визначається спроможність переведення їх в послідовність внутрішніх кодів компілятора і виконання комп'ютером.

8.1.4 Лексеми

Елементарною структурою будь-якої програми є слово – лексема.

Лексемою називається мінімальна значима одиниця тексту (можливо окремих символ) у програмі, що має певний сенс.

Лексеми умовно діляться на декілька категорій. Розглянемо деякі з них.

1. Зарезервовані (ключові слова)

Використовувані у мові Turbo Pascal англійські слова або словосполучення для позначення яких-небудь дій, операторів, функцій, властивостей тощо, називаються зарезервованими (ключовими) словами і складають основу мови Turbo Pascal 7.0

AND	ASM	ARRAY
BEGIN	CASE	CONST
CONSTRUCTOR	DESTRUCTOR	DIV
DO	DOWNT0	ELSE
END	EXPORTS	FILE
FOR	FUNCTION	GOTO

IF	IMPLEMENTATION	IN
INHERITED	INLINE	INTERFACE
LABEL	LIBRARY	MOD
NIL	NOT	OBJECT
OF	OR	PACKED
PROCEDURE	PROGRAM	RECORD
REPEAT	SET	SHL
SHR	STRING	THEN
TO	TYPE	UNIT
UNTIL	USES	VAR
WHILE	WITH	XOR

Зауваження: не допускається використання зарезервованих слів у якості ідентифікаторів користувача.

2. Ідентифікатори

Імена констант, типів, змінних процедур, функцій, модулів, програм і полів записів називаються ідентифікаторами. Ідентифікатор повинен починатись з букви або символу "_" і може складатися з букв, цифр і "_", мати довжину до 126 символів, проте значущими являються тільки перші 63 символи.

У мові Turbo Pascal 7.0 розрізняють два види ідентифікаторів: стандартні (зумовлені) і ті, що вводяться користувачем.

До стандартних ідентифікаторів відносяться:

- імена усіх вбудованих процедур та функцій (Read, Write, Sqr, Sqrt, Abs, Sin, Cos і т. д.);
- імена типів (Integer, Byte, Word, Real, Boolean, Char і т. д.);
- імена директив

Absolute	Assembler	Export	External	Far
Forward	Index	Interrupt	Near	Private
Public	Rezident	Virtual		

Перевизначення стандартних ідентифікаторів допускається, але не рекомендується. Нижче наведені синтаксичні діаграми запису ідентифікаторів

Приклад 8.1.

Неправильні ідентифікатори	Правильні ідентифікатори
#3 - використовується недопустимий символ #	<i>Number_3</i>
<i>Lab Rab 5</i> - присутні пропуски	<i>LabRab_5</i>
<i>2_e_zadanie</i> - починається з цифри	<i>Zadanie_2e</i>
<i>Y(2,1)</i> - використовуються круглі дужки та кома	<i>Y21</i>

3. Мітки

Мітки можуть бути цілочисловими (від 0 до 9999) або ідентифікаторами. Мітка відділяється від оператора символом ":".

Приклад 8.2.

Неправильні мітки	Правильні мітки
<i>Блок_1</i> - використовується кирилиця	<i>335</i>
<i>Main part</i> - присутній пропуск	<i>Vivod_Rez</i>
<i>_*</i> - містить символ "*"	<i>Metka_1</i>
<i>A.7</i> - містить символ "."	<i>Start</i>

4. Числа

У Turbo Pascal використовуються числа десяткові цілі і дійсні, а також цілі шістнадцяткові. Для позначення шістнадцяткових чисел використовується символ \$, який записується спереду числа.

Число вважається дійсним, якщо воно містить десяткову точку або символ E(e). Всі інші числа вважаються цілими. Символ E(e) в записі дійсного числа замінює основу ступеня 10, за ним безпосередньо слідує порядок.

Приклад 8.3.

Звичайна форма числа	Форма числа у Turbo Pascal
<i>3,14</i>	<i>3.14</i>
<i>7,8·10⁶</i>	<i>7.8E6</i>
<i>0,0035</i>	<i>-0.35E-02</i>
<i>D128F</i>	<i>\$D128F</i>

5. Рядки

Рядки - це послідовність символів, укладених в одиночні лапки (апострофи). Керувальні символи представляються в рядку за допомогою символа # і слідує за ним кодом ASCII керувального символу. Таким чином, в рядку можуть бути представлені і інші символи алфавіту мови. Максимальна довжина рядка - 255 символів.

Приклад 8.4.

```
`Мова програмування Turbo Pascal`  
`Звуковий сигнал`#7  
`Перехід на наступний рядок`#10
```

6. Коментарі

Коментарі представляють собою будь-який текст, довільну послідовність символів, укладену у фігурні дужки { } або (* *). Коментарі можуть розташовуватися в будь-якому місці програми і не впливають на її роботу.

Приклад 8.5.

```
(* Коментар  
    може  
        займати  
            декілька  
                рядків *)  
{ В процесі відладки програми під виглядом коментаря
```

можна зробити не виконуваними окремий оператор, блок операторів, ділянку програми }

7. Директиви компілятора

Директиви компілятора записуються аналогічно коментарям та в місцях, де допустимі коментарі. Починаються вони з символу \$ і беруться у фігурні дужки.

Є три типи директив:

- Директиви перемикання - включають (+) або вимикають (-) певні режими компілятора.
- Параметричні директиви - задають параметри, що впливають на компіляцію.
- Умовні директиви - управляють умовною компіляцією частин вихідного тексту в залежності від визначених користувачем умовних символів.

Приклад 8.6.

{ \$D+ } - цією директивою при компіляції програми відладчик Turbo Pascal дозволяє виконувати цей модуль програми по кроках і встановлювати в ньому точки зупину.

8.1.5 Роздільники

Як роздільники лексем використовуються пропуски (код 32), а також будь-які керувальні символи таблиці ASCII з кодами від 0 до 31. Крім того, як роздільники можуть виступати і деякі лексеми — коментарі, спеціальні або складені символи, наприклад “:”, “:=”, “(”, “)”, “,” і т.д., відокремлення яких від інших лексем символами-роздільниками не обов'язково.

Приклад 8.7. Рядок тексту програми

```
If z>0 then writeln(`z - додатне число');
```

містить такі лексеми

```
If, z, >, 0, then, writeln, (, `z - додатне число', ), ;
```

Введення додаткових роздільників між лексемами не змінює сенсу фрагменту тексту програми:

```
If  z  >  0
    then  writeln  (
        'z - додатне число'  )      ;
```

Між двома суміжними лексемами допускається довільна кількість символів-роздільників.

8.1.6 Дані

Дані в Pascal-програмі представляються константами та змінними. Константи та змінні визначаються ідентифікаторами, за якими здійснюється звернення до них. Константами називаються елементи мови, які при виконанні програми зазвичай не змінюються. Змінні на відміну від констант можуть приймати різні значення, але єдині в даний момент часу. Тип констант розпізнається компілятором автоматично, в той час як тип змінних повинен бути описаний попередньо, тобто до виконання над ними будь-яких дій.

8.1.7 Типи даних

Кожен тип визначає безліч значень, які можуть приймати дані цього типу, а також сукупність операцій над ними. У Turbo Pascal визначені наступні групи типів:

- скалярні (прості) типи;
- структуровані (складені) типи;
- покажчики;
- процедурні типи;
- об'єкти.

Скалярні типи поділяються на стандартні (зумовлені) і ті, що введені користувачем. До стандартних типів відносяться цілі, дійсні, логічний і символний типи. Усі скалярні типи, крім дійсних, називаються порядковими (дискретними) типами.

8.2 СТРУКТУРА ПРОГРАМИ

Загальна структура програми може бути подана наступною синтаксичною діаграмою:



Програма починається заголовком, який, в принципі, не є обов'язковим. Першим в заголовку записується службове слово *PROGRAM*, за яким вказується ім'я програми.

До не обов'язкових елементів програми відносяться пропозиція *USES* яка повинна бути присутньою у програмі тільки у тому випадку, коли в ній використовуються стандартні модулі Turbo Pascal (окремо трансльовані програмні одиниці).

Основною частиною будь-якої програми є блок, що складається з двох розділів - опису даних і послідовності дій, які необхідно виконати над даними. У першому розділі блоку у відповідних підрозділах розташовується опис та визначення об'єктів (міток - *Label*, типів - *Type*, констант - *Const*, змінних - *Var*, процедур - *Procedure* і функцій - *Function*), у другому розділі - оператори. Порядок розташування підрозділів довільний.

Розділ операторів являється основним розділом і обов'язково повинен бути присутнім у кожній програмі. Послідовність запису операторів визначає порядок їх виконання і повинна строго відповідати синтаксису та правилам пунктуації.

Блок називається глобальним, якщо його виконавчою частиною є головна програма. Взагалі, глобальним блоком можна вважати будь-який блок, що містить усередині себе один або декілька інших, локальних, блоків (процедур та/або функцій). Об'єкти, визначені в локальних блоках - локальні.

Структуру програми в загальному випадку можна представити наступним чином:

Program < Ім'я програми >;

Uses < Ім'я модуля 1, ім'я модуля 2, ... >;

Label		
	. . . ;	Опис міток
Const		
	. . . ;	Опис констант
Type		
	. . . ;	Опис типів
Var		
	. . . ;	Опис змінних
Function	< Ім`я > (Параметри); <Тип результату >;	
	Label . . . ;	Опис міток
	Const . . . ;	Опис констант
	Type . . . ;	Опис типів
	Var . . . ;	Опис змінних
	begin	
	< Оператори функції >	
	end;	
Procedure	< Ім`я > (Параметри);	
	Label . . . ;	Опис міток
	Const . . . ;	Опис констант
	Type . . . ;	Опис типів
	Var . . . ;	Опис змінних
	begin	
	< Оператори процедури >	
	end;	
BEGIN		
	< Оператори програми >	
END.		

8.3 ОГОЛОШЕННЯ І ОПИСИ

8.3.1. Опис міток

За допомогою міток можна зазначити (вказати) ті оператори, на які може бути передано управління з інших точок програми. Всі мітки повинні бути описані.

Приклад 8.8. Опис міток.

```
Label A1, 227, Start, max.
```

8.3.2 Опис типів

Безліч використовуваних у Turbo Pascal типів ділиться на стандартні типи та типи, що визначені користувачем. Стандартні типи даних не потребують опису, тоді як визначені користувачем типи повинні бути заздалегідь описані.

Приклад 8.9. Опис типів.

```
Type    Ext = Extended;  
Index = Byte;  
Logic = Boolean;
```

8.3.3 Опис констант

Константою в Pascal-програмі може бути ідентифікатор константи, ціле або дійсне число, рядок символів. Цілі константи в представляються як в десятковій, так і шістнадцятковій формі.

При опису константи можна використовувати тільки раніше визначені константи, з'єднані знаками операцій, і наступні стандартні функції:

<i>Abs</i>	<i>Chr</i>	<i>Hi</i>	<i>Length</i>	<i>Lo</i>
<i>Odd</i>	<i>Ord</i>	<i>Pred</i>	<i>Ptr</i>	<i>Round</i>
<i>Size</i>	<i>Of</i>	<i>Succ</i>	<i>Swap</i>	<i>Trunc</i>

Приклад 8.10. Опис констант.

```
Const  Max = 256;  
Min = 16;  
SizeR = (Max-Min)/2;
```



```

Err = `Помилка введення!`;
ErrStr = Err + `Повторіть введення.`;
ESC = #27;
CH =Ord(`Z`)-Ord(`A`);

```

Окрім звичайних констант широко використовуються типізовані константи, які можуть міняти своє значення як змінні, можуть використовуватись як змінні відповідного типа, знаходитися ліворуч від знаку привласнення.

Приклад 8.11. Опис типізованих констант.

```

Const  X: Integer=100;
        Y: Byte=16;

```

8.3.4 Опис змінних

Усі змінні, що використовуються в програмі, повинні бути описані. При цьому указується тип, що зазвичай визначає усі можливі значення змінної і операції допустимі над нею.

Приклад 8.12. Опис змінних.

```

Var      X, Y: Real;
        Num:  Byte;
        Err:  Boolean;

```

8.3.5 Опис процедур та функцій

Процедури та функції дозволяють включати в основний програмний блок додаткові блоки, окремо сформовані програмні одиниці.

8.4 СТАНДАРТНІ ТИПИ

До простих стандартних типів відносяться наступні типи:

- цілі;
- дійсні;

- булеві;
- символьний;
- рядковий;
- вказівний;
- текстовий.

8.4.1 Цілі типи

В Turbo Pascal є п'ять типів, що розрізняються допустимим діапазоном значень цілих чисел і розміром потрібної для їх розміщення області пам'яті.

Таблиця 8.1. Цілі типи.

Тип	Діапазон значень	Розмір пам'яті
Shortint	-128 .. 127	1 байт
Integer	-32768 .. 32767	2 байта
Longint	-2147483648 .. 2147483647	4 байта
Byte	0 .. 255	1 байт
Word	0 .. 65535	2 байта

Приклад 8.13. Опис змінних цілих типів.

```

Var   Step, Parametr: Byte;
      Kol_vo:          Longint;
      N_Test, N_Mod:   Word;
      Res:             Integer;

```

8.4.2 Дійсні типи

Turbo Pascal підтримує п'ять дійсних типів, які розрізняються діапазоном і точністю значень.

Таблиця 8.2. Дійсні типи.

Тип	Діапазон значень	Число цифр мантиси	Розмір пам'яті
Real	2.9e – 39 .. 1.7e38	11 — 12	6 байт
Single	1.5e – 45 .. 3.4e38	7 — 8	4 байта
Double	5.0e – 324 .. 1.7e308	15 — 16	8 байт
Extended	3.4e – 4932 .. 1.1e4932	19 - 20	10 байт
Comp	-9.2e18 .. 9.2e18	19 - 20	8 байт

Всі дійсні типи, окрім **Real**, можуть використовуватися в програмі тільки за наявності в комп'ютері математичного співпроцесора і при встановленій директиві **{N+}**.

Приклад 8.14. Опис змінних дійсних типів.

```
Var    X, Y, Z:      Extended;
      B, H, Del:    Real;
      Omega:        Double;
```

Тип **Comp** хоча й відноситься до дійсного типа, містить тільки ціло-числові значення в діапазоні

$-2^{63+1} (-9.2e18) \dots 2^{63-1} (9.2e18)$

8.4.3 Булеві типи

Булеві типи широко застосовуються в логічних виразах і виразах відношення. Мова Turbo Pascal 7.0 містить наступні булеві типи:

- Boolean;
- ByteBool;
- WordBool;
- LongBool.

Всі ці типи подані двома значеннями: **True** (істина) і **False** (брехня). Останні три булеві типа були введені у сьомій версії Turbo Pascal тільки для того, щоб забезпечити сумісність програм, що розробляються, з операційною системою Windows, в якій значенню **False** відповідає число 0, а значенню **True** — будь-яке відмінне від нуля число.

Таблиця 8.3. Булеві типи.

Тип	Розмір пам'яті	True	False
Boolean	1 байт	будь-яке відмінне від нуля число	0
ByteBool	1 байт		0
WordBool	2 байти		0 (у всіх байтах)
LongBool	4 байти		0 (у всіх байтах)

Приклад 8.15. Опис змінних булевих типів.

```
Var    Lx, Ly, Err:      Boolean;
      Lwb:              WordBool;
      Prm1_bb, Prm2_bb: ByteBool;
```

8.4.4 Символьний тип

Значенням символьного типу може бути будь-який з 256 символів розширеного коду ASCII. Ці значення займають один байт. Символьне значення зображується відповідним знаком, укладеним в апострофи. Якщо символьне значення не має графічного представлення, то його можна ввести за допомогою символу # і коду ASCII. Деякі керуючі символи ASCII можна ввести в формі ^C, де C - умовне позначення керуючого символу. (Таблиця 8.4)

Таблиця 8.4. Приклади представлення символьних значень.

Код ASCII	Представлення за допомогою			Мнемокод	Призначення символу
	' '	#	^		
3		#3	^C	ETX	Кінець тексту
7		#7	^G	BEL	Звуковий сигнал

8		#8	^H	BS	Повернення на крок
10		#10	^J	LF	Переклад рядка
13		#13	^M	CR	Повернення каретки
27		#27	^[ESC	Вихід, перехід, перемикання
32	' '	#32			Пропуск
82	'R'	#82			Буква R

Приклад 8.16. Опис змінних символьного типу.

```
Var    Ch1, Ch2, Ch3: Char;
```

8.4.5 Рядкові типи

Рядкові типи на відміну від інших стандартних типів відносяться до структурних типів. Turbo Pascal 7.0 надає можливість працювати з двома рядковими типами - String і PChar.

Значенням строкового типу String є послідовність символів, кількість яких визначає довжину рядка і може динамічно змінюватися від 0 до 255. Визначення даних строкового типу здійснюється за допомогою ідентифікатора string, за яким слідує у квадратних дужках покажчик допустимої довжини рядка даного типу. Якщо покажчик довжини відсутній, то довжина рядка приймається за замовчуванням рівною 255.

Для доступу до окремого символу рядка необхідно вказати відразу за ідентифікатором строкової змінної в квадратних дужках порядковий номер (індекс) символу в рядку. Доступ до інформації про поточну довжину рядка здійснюється за допомогою нульового значення індексу або стандартної функції Length.

Рядковий тип PChar введений у Turbo Pascal 7.0 для підтримки рядкових структур даних, використовуваних в Windows, з метою полегшити створення програм, що працюють в цьому середовищі.

Приклад 8.17. Опис змінних рядкових типів.

```

Type      Str = String[80];

Var      Intr : Pchar;
          St1, St2 : Str;
          Nazv: String[30];
          Adres: String[120];

```

8.4.6 Вказівний тип

Стандартний вказівний тип **Pointer** в Turbo Pascal 7.0 є універсальним типом покажчика. Цей тип дозволяє оголошувати вказівник не пов'язуючи його з будь-яким конкретним типом даних. Використовуючи тип **Pointer** як проміжний, можна присвоїти значення одного покажчика іншому навіть при неспівпаданні їх типів.

Значеннями констант і змінних вказівного типу є адреси оперативної пам'яті. Адреси задаються двома шістнадцятковий словами - сегментом і зміщенням. Сегмент і зсув займають по 2 байти оперативної пам'яті.

Приклад 8.18. Опис змінних вказівного типу.

```

Var      Pntr, Up_1, Up_2 : Pointer;

```

8.4.7 Текстовий тип

Текстовий тип **Text** застосовується для опису текстових файлів, тому буде розглянуто пізніше при роботі з файлами.

8.5 ВВЕДЕННЯ-ВИВЕДЕННЯ

У мові Turbo Pascal для введення і виведення даних використовуються відповідно процедури **Read**, **Readln** і **Write**, **Writeln**.

8.5.1 Пристрій CON

Це пристрій означає консоль, за допомогою якої інформація, що виводиться, пересилається на екран дисплея, а що вводиться - зчитується з клавіатури.

8.5.2 Паралельні адаптери LPT

Якщо підключено один пристрій друку, то на нього посилаються як на LPT, LPT1 або PRN. Стандартний модуль PRINTER містить стандартну змінну з і'ям LST і з її допомогою встановлює зв'язок із пристроєм LPT. Тому для забезпечення виведення інформації на принтер необхідно в програмі за допомогою USES модуля PRINTER в процедурі виведення вказати LST.

8.5.3 Процедура читання Read

Процедура Read забезпечує введення числових цілих і дійсних даних, символів, рядків. Загальна форма оператора звернення до процедури:

$$Read(f_v, x_1, x_2, \dots, x_n);$$

де f_v - і'мя пристрою (файлова змінна), звідки вводяться дані x_i ($i=1,2,\dots,n$). За умовчанням f_v дорівнює CON, тому при введенні з клавіатури f_v можна не указувати. Значення, що вводяться x_1, x_2, \dots, x_n набираються через один пропуск і більш. Після набору даних введення їх здійснюється клавішею **ENTER**.

8.5.4 Процедура читання Readln

Ця процедура аналогічна Read, єдина відмінність полягає в тому, що після зчитування значень всіх змінних для однієї процедури Readln дані для наступної процедури будуть зчитуватись з початку нового рядка.

8.5.5 Процедура читання Write

Процедура Write використовується для виведення числових цілих і дійсних даних, символів, рядків і логічних даних. Загальна форма оператора звернення до процедури:

$$Write(f_v, x_1, x_2, \dots, x_n);$$

де f_v - і'мя пристрою (файлова змінна), на який здійснюється вивід. За умовчанням f_v дорівнює CON, x_i ($i=1,2,\dots,n$) - вирази, змінні, константи, значення яких повинні бути виведені.

8.5.6 Процедура читання Writeln

Ця процедура аналогічна процедурі `Write`, але після виконання її здійснюється перехід до початку наступного рядка.

8.5.7 Форматоване виведення

При виводі на екран і друк є можливість вказати у вигляді константи, змінної або виразу величину поля виводу. Вона записується через двокрапку після імені або виразу значення, що виводиться.

При виводі дійсних величин з фіксованою точкою після величини поля через двокрапку може бути вказано число символів дробової частки.

Приклад 8.19. Ввести з клавіатури і вивести на екран в один рядок і на друк в два рядка наступні дані:

$A=5,186;$ $B=0,837 \cdot 10^5;$ $C=456,24;$ $I=12;$ $N=85.$

```
Program Primer;  
Uses Printer;  
Var  
    A, B, C: Real;  
    I, N: Byte;  
    Lst: Text;  
BEGIN  
    Assign(Lst, 'Lpt1');  
    Rewrite(Lst);  
    Writeln('Ввести значення  
            A, B, C, I, N');  
    Read(A, B, C, I, N);  
    Writeln('Виведення даних на екран');  
    Writeln('A=', A:7:3, ' ', 'B=', B:9,  
' ', 'C=', C:8:3, ' ', 'I=', I:4,  
' ', 'N=', N);  
    Writeln(Lst,  
            'Виведення даних на принтер');  
    Writeln(Lst, 'A=', A:7:3, ' ',
```



```

        'B=',B:9,' ','C=',C:8:3);
Writeln (Lst,'I=',I,' ','N=',N);
Close (Lst)
END.

```

Робота з програмою:

Виведення повідомлення:

Ввести значення

Введення даних з клавіатури

-5,186 _ 0,837e5 _ 456,24 _ 12 _ 85

Виведення повідомлення:

Виведення даних на екран

A=-5.186_B=8.370E+04_C=456.240_I=_12_N=85

Виведення повідомлення:

Виведення даних на принтер

A=-5.186_B=8.370E+04_C=456.240

I=_12_N=85

8.6 СТАНДАРТНІ ПРОЦЕДУРИ І ФУНКЦІЇ

Стандартні процедури і функції заздалегідь визначені і викликаються для виконання по імені.

Для спрощення запису введемо наступні позначення:

X - константа, змінна або вираз, може бути як цілого, так і дійсного типу;

I - константа, змінна або вираз тільки цілого типу.

S - константа, змінна або вираз тільки порядкового типу.

Таблиця 8.5. Математичні функції.

Функція	Призначення	Тип результату
---------	-------------	----------------

$Abs(X)$	$ X $	Співпадає з X
$ArcTan(X)$	$arctg(X)$	Дійсний
$Cos(X)$	$\cos(X)$	Дійсний
$Exp(X)$	$\exp(X)$	Дійсний
$Frac(X)$	Дробова частина числа	Дійсний
$Int(X)$	Ціла частина числа	Дійсний
$Ln(X)$	$\ln(X)$	Дійсний
Pi	Число π	$\approx 3,1415926535897932385$
$Sin(X)$	$\sin(X)$	Дійсний
$Sqr(X)$	X^2	Співпадає з X
$Sqrt(X)$	\sqrt{X}	Дійсний
$Random$	Значення випадкового числа з діапазону 0 ... 0,99	Дійсний
$Random(I)$	Значення випадкового числа з діапазону 0 ... 1	Цілий

Таблиця 8.6. Функції перетворення типів.

Функція	Призначення	Приклади
$Chr(I)$	Перетворення ASCII коду у відповідний символ	$Chr(65) \Rightarrow A$ $Chr(122) \Rightarrow z$
$High(X)$	Отримання максимального значення	
$Low(X)$	Отримання мінімального значення	
$Ord(S)$	Перетворення порядкового типу в цілий	$Ord('A') \Rightarrow 65$ $Ord(\#4) \Rightarrow 4$ $Ord(^G) \Rightarrow 7$
$Round(X)$	Округлення дійсного числа до найближчого цілого	$Round(15.7) \Rightarrow 16$

Таблиця 8.7. Скалярні процедури і функції.

Функція	Призначення	Приклади
---------	-------------	----------

$Dec(I[n])$	Зменшення змінної I на n . За відсутності n на 1	$I:=2; \quad Dec(I,2) \Rightarrow 6$ $I:=16; \quad Dec(I) \Rightarrow 15$
$Inc(I[n])$	Збільшення змінної I на n . За відсутності n на 1	$I:=10; \quad Inc(I,4) \Rightarrow 14$ $I:=16; \quad Inc(I) \Rightarrow 17$
$Odd(I)$	Перевірка величини I на непарність. Результат <i>True</i> , якщо I непарне і <i>False</i> , якщо I парне.	$Odd(5) \Rightarrow True$ $Odd(12) \Rightarrow False$
$Pred(S)$	Визначення попереднього значення величини S порядкового типа.	$Pred(100) \Rightarrow 99$ $Pred('Z') \Rightarrow 'Y'$ $Pred(True) \Rightarrow False$
$Succ(S)$	Визначення наступного значення величини S порядкового типа.	$Succ(15) \Rightarrow 16$ $Succ('A') \Rightarrow 'B'$ $Succ(True) \Rightarrow False$

8.7 Вирази

Вирази будуються з операндів, знаків операцій і круглих дужок.

Операнди — це константи, змінні, звернення до функцій.

Операції — це дії, які повинні бути виконані над операндами.

Унарні операції пов'язані тільки з одним операндом, перед яким указується символ операції, бінарні — з двома.

Найпростішим виразом є константа або змінна.

Залежно від характеру виконуваних дій операції діляться на наступні групи:

- арифметичні операції (унарні та бінарні);
- операції відношення;
- булеві (логічні) операції;
- рядкова операція;
- операція над множинами;
- операція @.

8.7.1 Арифметичні операції

Арифметичні операції виконують арифметичні дії у виразах над значеннями операндів цілочисельних і дійсних типів Унарні арифметичні операції представлені в таблиці 8.8, бінарні — в таблиці 8.9.

Таблиця 8.8 Арифметичні унарні операції.

Знак	Операція	Тип операнда	Тип результату
+	Збереження знаку	Цілий Дійсний	Цілий Дійсний
-	Заперечення знаку	Цілий Дійсний	Цілий Дійсний
<i>not</i>	Арифметичне заперечення	Цілий	Цілий

Унарна операція збереження знаку + залишає поточний знак операнда без змін, операція заперечення знаку — відновлює значення операнда з протилежним знаком. Операція *not* виконує побітну інверсію операнда, який представляється в двійковій формі.

Приклад 8.20. Унарна арифметична операція *not* ($A=13$).

Форма подання	Операція	A	Результат	
Десяткова	<i>not</i>	13	242	
Двійкова	<i>not</i>	0	=	1
	<i>not</i>	0	=	1
	<i>not</i>	0	=	1
	<i>not</i>	0	=	1
	<i>not</i>	1	=	0
	<i>not</i>	1	=	0
	<i>not</i>	0	=	1
	<i>not</i>	1	=	0
<i>not</i> 13 \Rightarrow 242				

Таблиця 8.9 Арифметичні бінарні операції.

Знак	Операція	Тип операндів	Тип результату
------	----------	---------------	----------------

+	Складання	Цілий Дійсний	Цілий Дійсний
-	Віднімання	Цілий Дійсний	Цілий Дійсний
*	Множення	Цілий Дійсний	Цілий Дійсний
/	Ділення	Цілий Дійсний	Дійсний Дійсний
<i>div</i>	Цілочислове ділення	Цілий	Цілий
<i>mod</i>	Залишок від ділення	Цілий	Цілий
<i>and</i>	Арифметичне І	Цілий	Цілий
<i>shl</i>	Зсув вліво	Цілий	Цілий
<i>shr</i>	Зсув вправо	Цілий	Цілий
<i>or</i>	Арифметичне АБО	Цілий	Цілий
<i>xor</i>	Арифметичне, що виключає АБО	Цілий	Цілий

Бінарні операції $+$, $-$, $*$, $/$ виконуються як операції у звичайних арифметичних виразах. Цілочислове ділення *div* повертає цілу частину частки. Перед виконанням операції *div* обидва операнди округляються до цілих значень. Операція *mod* повертає залишок від цілочислового ділення.

Приклад 8.20.

$$11 \text{ div } 5 \Rightarrow 2;$$

$$14 \text{ div } 3 \Rightarrow 4;$$

$$3 \text{ mod } 2 \Rightarrow 1;$$

$$14 \text{ mod } 5 \Rightarrow 4.$$

Операції *and* (логічне множення), *or* и *xor* (логічне складання) виконують дії, у відповідності з таблицею 8.10. Операнди і результат представляються у десятковій формі, но під час виконання операнди переводяться в двоїчні числа.

Таблиця 8.10 Арифметичні бінарні операції.

Операція	Дія	Вираз	Значення операндів		Результат
			A	B	
<i>and</i>	Логічне множення	$A \text{ and } B$	1	1	1
			1	0	0
			0	1	0
			0	0	0
<i>or</i>	Логічне складання	$A \text{ or } B$	1	1	1
			1	0	1
			0	1	1
			0	0	0
<i>xor</i>	Логічне складання	$A \text{ xor } B$	1	1	0
			1	0	1
			0	1	1
			0	0	0

Для більшої наочності дію операцій *and*, *or*, *xor*, *shl*, *shr* розглянемо на конкретних прикладах для значень *A* і *B* типу Byte.

Приклад 8.21. Бінарна арифметична операція *and* ($A=15, B=21$).

Форма подання	<i>A</i>	Операція	<i>B</i>	Результат	
Десяткова	15	<i>and</i>	21	5	
Двійкова	0	<i>and</i>	0	=	0
	0	<i>and</i>	0	=	0
	0	<i>and</i>	0	=	0
	0	<i>and</i>	1	=	0
	1	<i>and</i>	0	=	0
	1	<i>and</i>	1	=	1
	1	<i>and</i>	0	=	0
	1	<i>and</i>	1	=	1
$15 \text{ and } 21 \Rightarrow 5$					

Приклад 8.22. Бінарна арифметична операція *or* ($A=17, B=12$).

Форма подання	A	Операція	B	Результат	
Десяткова	17	<i>or</i>	12	29	
Двійкова	0	<i>or</i>	0	=	0
	0	<i>or</i>	0	=	0
	0	<i>or</i>	0	=	0
	1	<i>or</i>	0	=	1
	0	<i>or</i>	1	=	1
	0	<i>or</i>	1	=	1
	0	<i>or</i>	0	=	0
	1	<i>or</i>	0	=	1
$17 \text{ or } 12 \Rightarrow 29$					

Приклад 8.23. Бінарна арифметична операція *xor* ($A=19, B=23$).

Форма подання	A	<i>xor</i>	B	Результат	
Десяткова	19	<i>xor</i>	23	4	
Двійкова	0	<i>xor</i>	0	=	0
	0	<i>xor</i>	0	=	0
	0	<i>xor</i>	0	=	0
	1	<i>xor</i>	1	=	
	0	<i>xor</i>	0	=	
	0	<i>xor</i>	1	=	1
	1	<i>xor</i>	1	=	0
	1	<i>xor</i>	1	=	
$19 \text{ xor } 23 \Rightarrow 4$					

Приклад 8.24. Бінарна арифметична операція *shl* для ($A=2, B=5$).

Форма подання	A	Операція	B	Результат
---------------	-----	----------	-----	-----------

Десяткова				2	<i>shl</i>	5	64		
Пояснення:									
A	0	0	0	0	0	0	1	0	
	Зсув на В (5) позицій вліво ←								
	0	1	0	0	0	0	0	0	
$2\ shl\ 5\ \Rightarrow\ 64$									

Приклад 8.25. Бінарна арифметична операція *shr* для ($A=25, B=3$).

Форма подання				A	Операція	B	Результат		
Десяткова				25	shr	3	3		
Пояснення:									
A	0	0	0	1	1	0	0	1	
	→ Зсув на B (3) позиції вправо								
	0	0	0	0	0	0	1	1	3

Результат бінарних операцій завжди буде дійсним, якщо хоч би один операнд виявиться дійсним

8.7.2 Операції відношення

Операція відношення виконує порівняння значень двох операндів. Результат виконання операція відношення є булеве значення *True* або *False*. Операції відношення наведені в таблиці 8.11

Для рядкового типу порівняння рядків виконується зліва направо посимвольно відповідно до кодів ASCII. Всі рядки незалежно від їх довжини вважаються сумісними. Допускається порівнювати значення рядкового і символьного типів. При цьому значення символьного типу обробляється як рядки завдовжки в один символ.

При порівнянні значень вказівного типу можливо використання тільки

двох операцій: $=$ і \diamond . Показники вважаються рівними, якщо вони посилаються на один і той же об'єкт.

Таблиця 8.11 Операції відношення.

Знак	Операція	Тип операндів	Тип результату
$=$	Рівно	Сумісний простий Рядковий Вказівний	Булевий
\diamond	Не рівно	Сумісний простий Рядковий Вказівний	
$<$	Менше	Сумісний простий Рядковий	
$>$	Більше	Сумісний простий Рядковий	
\leq	Менше або рівно	Сумісний простий Рядковий	
\geq	Більше або рівно	Сумісний простий Рядковий	

Приклад 8.26.

$$2 * 2 = 5 \quad \Rightarrow \text{False} ;$$

$$'A + B' < 'a + b' \quad \Rightarrow \text{True}.$$

8.7.3 Булеві (логічні) операції

Результати булевих операцій мають логічне значення *True* або *False* і відповідають звичайній булевій алгебрі. В якості операндів логічних виразів можуть виступати тільки дані булевого типу.

Таблиця 8.12 Булева унарна операція.

Операція	Дія	Вираз	Значення операнда Λ	Результат
----------	-----	-------	--------------------------------	-----------

not	Логічне заперечення	$not\ A$	True False	False True
------------	---------------------	----------	---------------	---------------

Таблиця 8.13. Булеві бінарні операції.

Операція	Дія	Вираз	Значення операндів		Результат
			A	B	
and	Логічне <i>I</i>	$A\ and\ B$	True True False False	True False True False	True False False False
or	Логічне <i>АБО</i>	$A\ or\ B$	True True False False	True False True False	True True True False
xor	Що виключає <i>АБО</i>	$A\ xor\ B$	True True False False	True False True False	False True True False

8.7.4 Рядкова операція

Операція конкатенація (зчеплення) використовується для з'єднання декількох символів або рядків в один результуючий рядок. Якщо довжина результуючого рядка перевищує 255 символів, то всі зайві символи справа відкидаються.

Таблиця 8.14. Рядкова операція.

Знак	Операція	Тип операндів	Тип результату
+	Конкатенація (зчеплення)	Символьний Рядковий	Рядковий

Приклад 8.27.

Вираз	Результат
'I'+ 'B'+ 'M'+ ' '+ 'PC'+ ' ' + 'computer'	IBM PC computer

8.7.5 Операції над множинами

Операції над множинами виконуються відповідно до правил теорії множин. У Turbo Pascal 7.0 допускається використання тільки скінченних множин.

При роботі з множинами застосовуються бінарні операції об'єднання, різниці, перетин, а також деякі операції відношення. Операндами цих операцій є множинні константи, змінні-множини або вирази, що набувають значення множинного типу. Результатом операцій відношення є булеве значення True або False

Таблиця 8.15. Операції над множинами.

Операція	Запис в математиці	Запис в Turbo Pascal	Результат
Об'єднання	$A \cup B$	$A + B$	<i>Множина</i> , елементи якої входять хоч би в одну з множин A або B
Різниця	$A \setminus B$	$A - B$	<i>Множина</i> , до якої входять тільки ті елементи множини A , що не входять в множину B
Перетин	$A \cap B$	$A * B$	<i>Множина</i> , елементи якої одночасно входять і в множину A і в множину B
Рівність	$A = B$	$A = B$	<i>True</i> , якщо множини A і B співпадають
Не рівність	$A \neq B$	$A <> B$	<i>True</i> , якщо множини A і B не співпадають
Є підмножиною	$A \subseteq B$	$A \leq B$	<i>True</i> , якщо всі елементи множини A належать множині B
Є надмножиною	$A \supseteq B$	$A \geq B$	<i>True</i> , якщо всі елементи множини B належать множині A
Приналежність	$a \in A$	$A \text{ in } B$	<i>True</i> , якщо елемент a входить в множину A

Приклад 8.28. Операції над множинами.

$$[1, 3, 5, 7, 9] + [1, 2, 3, 4, 5] \Rightarrow [1, 2, 3, 4, 5, 7, 9];$$

$$[1, 3, 5, 7, 9] - [1, 2, 3, 4, 5] \Rightarrow [7, 9];$$

`[1, 3, 5, 7, 9] * [1, 2, 3, 4, 5] ⇒ [1, 3, 5];`

`[1, 3, 5, 7, 9] = [1, 2, 3, 4, 5] ⇒ False;`

`[1, 3, 5, 7, 9] <> [1, 2, 3, 4, 5] ⇒ True;`

`[1, 3, 5, 7, 9] <= [1, 2, 3, 4, 5] ⇒ False;`

`[1, 3, 5, 7, 9] >= [1, 2, 3, 4, 5] ⇒ False;`

`5 in [1, 2, 3, 4, 5] ⇒ True.`

8.7.6 Операція @ (створення покажчика)

Операція @ є унарною операцією, за допомогою якої можна створити покажчик на змінну, процедуру, ідентифікатор функції. Тип результату являється таким же, як тип покажчика *nil*, що дозволяє привласнити результат будь-якому покажчику змінної.

Таблиця 8.16. Операція @.

Знак	Операція	Тип операнда	Тип результату
@	Отримання покажчика	Посилання на змінну, процедуру, ідентифікатор функції або методу	Покажчик (сумісний з <i>nil</i>)

8.7.7 Пріоритет операцій

Порядок виконання операцій у виразі визначається пріоритетом операцій. Значення пріоритетів наведені у таблиці 8.17.

Таблиця 8.17. Пріоритети операцій

Операції	Пріоритет	Вид операції
()	Перший (вищий)	Виконання дій в дужках
Функції	Другий	Виконання функцій

@ not	Третій	Унарні
* / div mod and	Четвертий	Типа множення
shl shr + - or xor	П'ятий	Типа складення
= <> < > <= >= in	Шостий (нижчий)	Відношення

Виконання операцій здійснюється за наступними правилами:

- Якщо ліворуч й праворуч операнда операції різного пріоритету, то він у першу чергу зв'язується з операцією, що має більш високий пріоритет.
- Вираз в дужках обчислюється як окремий операнд.
- Операції з рівними пріоритетами виконуються зліва направо.

Приклад 8.29. Записати засобами мови Turbo Pascal вираз $a \leq x \leq b$

$(x \leq a) \text{ and } (x \leq b)$

8.8 ОПЕРАТОРИ

Оператори служать для опису дій, які повинні бути виконані над описаними даними при реалізації алгоритму. Оператори Turbo Pascal поділяються на дві групи:

- прості оператори,
- структурні оператори.

8.8.1 Прості оператори

Простими називаються ті оператори, які не містять у собі інших операторів. До простих операторам відносяться:

- оператор присвоювання;

- оператор процедури;
- оператор безумовного переходу;
- порожній оператор.

1. Оператор присвоювання

Загальна форма оператора:

$$a := b \text{ ,}$$

де a - ідентифікатор змінної або функції, b - вираз. Змінна і вираз повинні бути сумісні за типом.

2. Оператор процедури

Загальна форма оператора:

$$a[(c_1, c_2, \dots, c_n)] \text{ ,}$$

де a - ідентифікатор процедури, c_1, c_2, \dots, c_n - список фактичних параметрів. Оператор служить для виклику стандартної або користувацької процедури. Виконання оператора активізує описані у процедурі дії. Зазначені в операторі процедури фактичні параметри підставляються замість описаних у заголовку процедури формальних параметрів, тому списки фактичних і формальних параметрів повинні бути узгоджені за типом, кількістю і по порядку проходження параметрів у списках.

3. Оператор безумовного переходу

Загальна форма оператора:

$$GOTO n \text{ ,}$$

де n - мітка оператора, якому має бути передане керування. Область дії оператора - блок. Відповідно до принципів структурного програмування застосування оператора GOTO в програмах має бути обмежене або виключено взагалі.

4. Порожній оператор

Цей оператор не містить ніяких символів і не виконує ніяких дій. Зазвичай порожній оператор використовується для організації переходу по мітці порожнього рядка.

8.8.2 Структурні оператори

Структурні оператори будуються з зарезервованих слів, логічних виразів та інших операторів. До структурних операторів відносяться:

- складений оператор;
- умовний оператор *if*;
- оператор вибору *case*;
- оператори повторення *for*, *repeat*, *while*;
- оператор приєднання *with*.

1. Складений оператор

Складений оператор представляє собою групу з довільного числа операторів, відділених один від одного крапкою з комою і обмежену операторними дужками *Begin* і *End*. Складений оператор використовується в тих випадках, коли необхідно кілька операторів представити у програмі як один.

2. Умовний оператор

У мові Turbo Pascal умовний оператор використовується в повній і неповній формах.

Неповна форма оператора:

$$IF\ u\ THEN\ d$$

Якщо вираз *u* має значення *True*, то виконується оператор *d*, якщо *False* - виконується оператор, безпосередньо наступний за оператором *IF*.

Повна форма оператора:

$$IF\ u\ THEN\ d_1 \\ ELSE\ d_2$$

Якщо вираз *u* має значення *True*, то виконується оператор *d1*, якщо *False* - виконується оператор *d2*.

Якщо оператори *IF* потрібно розташувати один в іншому, то структура вкладеності операторів повинна мати наступний вигляд:

$$IF\ u'\ THEN\ d \\ IF\ u''\ THEN\ d_1 \\ ELSE\ d_2$$

тобто Else має зв'язуватися з найближчим попереду розташованим IF.

Приклад 8.30. Скласти програму обчислення виразу

$$w = \begin{cases} \frac{x^{n+1}}{A(n+1)} & \text{нпу } n \neq -1; \\ \frac{\ln x}{A} & \text{нпу } n = -1. \end{cases}$$

```
Program Primer;  
Var   n:           Integer;  
      A, x, w:     Real;  
  
BEGIN  
  Read(n,A,x) ;  
  IF n<>1  
    THEN w:=Exp((n+1)*ln(x))/(A*(n+1))  
    ELSE w:=ln(x)/A;  
  Writeln(n,A,x) ;  
  Writeln(w)  
END.
```

3. Оператор вибору

Цей оператор дозволяє зробити вибір з довільного числа наявних варіантів і може використовуватися як в повній, так і в неповній формах

Неповна форма оператора:

CASE u_s *OF*

c_1 : d_1 ;

c_2 : d_2 ;

...

Повна форма оператора:

CASE u_s *OF*

c_1 : d_1 ;

c_2 : d_2 ;

...

$c_n: d_n$
END

$c_n: d_n$
ELSE d
END

Якщо вираз-селектор u_s збігається з одним зі значень списку c_i , то виконується оператор d_i ($i=1,2,\dots,n$), якщо u_s не збігається ні з одним зі значень списку c_1, c_2, \dots, c_n , то оператори d_1, d_2, \dots, d_n не виконуються.

Якщо вираз-селектор u_s збігається з одним зі значень списку c_i , то виконується оператор d_i ($i=1,2,\dots,n$), якщо u_s не збігається ні з одним зі значень списку то в c_1, c_2, \dots, c_n , виконується оператор d .

Вираз-селектор може приймати лише значення порядкового типу, загальна кількість елементів якого не більше 65535.

Приклад 8.31. Скласти програму обчислення площі фігури

$$S = \begin{cases} a^2 & \text{при } k = 1; \\ ab & \text{при } k = 2; \\ \frac{ah}{2} & \text{при } k = 3; \\ \frac{(a+b)h}{2} & \text{при } k = 4; \\ \pi R^2 & \text{при } k = 5. \end{cases}$$

Program Primer;

Var k: Byte;

A, B, H, R, S: Real;

BEGIN

Writeln('Введіть значення k');

```

Read(k);
Case k Of
1:Begin
    Writeln('Введіть значення A');
    Read(A);
    Writeln('Площа квадрата S=',
            Sqr(A))
End;
2:Begin
    Writeln('Введіть значення A,B');
    Read(A,B);
    Writeln('Площа прямокутника S=',
            A*B)
End;
3:Begin
    Writeln('Введіть значення A,H');
    Read(A,H);
    Writeln('Площа трикутника S=',
            A*H/2)
End;
4:Begin
    Writeln('Введіть значення A,B,H');
    Read(A,B,H);
    Writeln('Площа трапеції S=',
            (A+B) *H/2)
End;
5:Begin
    Writeln('Введіть значення R');

```

```

Read (R) ;
Writeln ('Площа круга S=' ,
          Pi * Sqr (R) )

End;
END.

```

4. Оператор цикла *FOR*

Оператор циклу *FOR* є циклом із заданим числом повторень, з цілим кроком. Загальна форма оператора:

Цикл зі зростаючим параметром:

```

FOR i := k1
  TO k2 DO d

```

де *i* - параметр циклу,
 k_1 і k_2 - відповідно початкове і кінцеве значення параметра циклу ($k_1 \leq k_2$).

Крок зміни параметра циклу завжди дорівнює +1.

Цикл з убуючим параметром:

```

FOR i := k1
  DOWNTO k2 DO d

```

де *i* - параметр циклу,
 k_1 і k_2 - відповідно початкове і кінцеве значення параметра циклу ($k_1 \geq k_2$).

Крок зміни параметра циклу завжди дорівнює +1.

Параметр циклу, його початкове і кінцеве значення повинні належати до одного й того ж порядковому типу даних.

Приклад 8.32. Скласти програму обчислення виразу

$$C_n^m = \frac{n!}{m!(n-m)!}$$

```

Program Primer;

```

```

Var      m,n,k,I,Fn,Fm,Fk:   Byte;
          C:                   Real;

BEGIN

  Writeln('Введіть значення: m, n');
  Read(m,n);
  k:=n-m;
  Fn:=1;
  For I:=1 to n do
    Fn:=Fn*I;
  Fm:=1;
  For I:=1 to m do
    Fm:=Fm*I;
  Fk:=1;
  For I:=1 to k do
    Fk:=Fk*I;
  C:=Fn/(Fm*Fk);
  Writeln('C=',C)

END.

```

5. Оператор цикла *REPEAT*

Оператор циклу *REPEAT* відноситься до циклів з довільним кроком і з невідомим числом повторень.

Загальна форма оператора:

```

REPEAT
  d1;
  d2;

```

\dots
 $d_n;$
 UNTIL u

Оператори d_i ($i=1,2,\dots,n$), що розташовані між службовими словами *Repeat* і *Until*, утворюють тіло циклу. Серед операторів повинен бути хоча б один оператор, що впливає на умову закінчення циклу. Цикл, організовуваний за допомогою оператора *Repeat*, є циклом з післяумовою, тому оператори тіла циклу виконуються як мінімум один раз за будь-яких умов. Цикл триває поки логічний вираз u приймає значення *False* і закінчується, коли u стає рівним *True*.

Приклад 8.33. Скласти програму обчислення виразу

$$z = \ln^2 \frac{2 - \cos x}{\sqrt{e^{x^2} - \sin^2 2x}} \quad \text{при } x \in [a, b], \quad \Delta x = h.$$

Введемо позначення:

$$p = 2 - \cos x;$$

$$q = e^{x^2} - \sin^2 2x; \quad q > 0.$$

Математична модель прийме наступний вигляд:

$$z = \ln^2 \frac{p}{\sqrt{q}}; \quad q > 0.$$

```

Program Primer;
  Var  x,a,b,h,z,p,q: Real;
BEGIN
  Writeln('Введіть значення a,b,h');
  Read(a,b,h);
  x:=a;

```

```

Repeat
  p:=2-Cos(x);
  q:= Exp(Sqr(x))-Sqr(Sin(2*x));
  If q>0 Then
    Begin
      z:=Sqr(Ln(p/sqrt(q)));
      Writeln('x=',x,'  'z=',z)
    End;
  Else Writeln('При x=',x,4#32,
    'Рішень немає, q=',q);
  x:=x+h;
Until x>b;
END.

```

6. Оператор цикла *WHILE*

Оператор циклу *WHILE*, так само як і оператор *REPEAT*, відноситься до циклів з довільним кроком і з невідомим числом повторень.

Загальна форма оператора:

WHILE u DO d

При вході в цикл аналізується булевий вираз *u*: якщо він має значення *True* - оператор в циклі *d* виконується, якщо *False* - здійснюється перехід до наступного за циклом оператору. Цикл, організовуваний за допомогою оператора *WHILE*, є циклом з передумовою, так як перевірка умови *u* проводиться до початку чергового повторення. Тому оператор *d* може не виконуватися жодного разу.

Приклад 8.34. Скласти програму обчислення значень функції

$$f = x^3 - \frac{e^{2x}}{2+x^2} \quad \text{при} \quad x \in [a, b], \quad \Delta x = h.$$

Program Primer;

Var x,a,b,h,f: Real;

```

BEGIN
  Writeln('Введіть значення a,b,h');
  Read(a,b,h);
  x:=a;
  While x<=b do
  begin
    f:=Exp(3*Ln(x))-Exp(2*x)/(2+Sqr(x));
    x:=x+h;
    Writeln('x=',x,' f=',f)
  end;
END.

```

7. Оператор приєднання *WITH*

Оператор *WITH* використовується для спрощення форми звернення до полів запису.

Загальна форма оператора:

WITH a DO d

де *a* - змінна типу запис, *d* - оператор. Оператор приєднання *WITH* дозволяє скоротити імена полів запису наступним чином: один раз вказавши змінну типу запис *a* в операторі *WITH*, можна працювати з іменами полів як зі звичайними змінними.

8.9 СТАНДАРТНІ ПРОЦЕДУРИ ПЕРЕХОДІВ

У Turbo Pascal 7.0 програмування переходів з однієї точки програми в іншу можливо за допомогою оператора безумовного переходу *GOTO*. Проте використання цього оператора в програмах не рекомендується, оскільки суперечить принципам структурного програмування.

У версії 7.0 мови Turbo Pascal є спеціальні засоби конструювання переходів у програмах, що реалізовані у вигляді стандартних процедур *BREAK*, *CONTINUE*, *EXIT*, які добре узгоджуються з принципами структурного програмування.

8.9.1 Процедура виходу із циклу *BREAK*

Ця процедура дозволяє по умові *u* передати управління оператору, безпосередньо наступному за циклом, в якому розташовується оператор процедури *BREAK*. Оператор процедури *BREAK* зазвичай є виконавчою частиною умовного оператора і записується в наступному вигляді:

IF u THEN BREAK

8.9.2 Процедура продовження циклу *CONTINUE*

Процедура *CONTINUE* перериває по умові *u* чергову ітерацію циклу і здійснює перехід до нової ітерації. Форма запису звернення до процедури *CONTINUE* зазвичай має наступний вигляд:

IF u THEN CONTINUE

8.9.3 Процедура виходу із блоку *EXIT*

Процедура *EXIT* перериває по умові *u* виконання поточного блоку програми і передає управління у точку виклику. Форма запису звернення до процедури *EXIT* аналогічна *CONTINUE* і має наступний вигляд:

IF u THEN EXIT

8.10 ПРОЦЕДУРИ І ФУНКЦІЇ

Підпрограми (процедури і функції) є важливим засобом представлення програм складних задач у вигляді функціонально закінчених незалежних складових частин. У процедури і функції звичайно виділяються неодноразово повторювані аналогічні фрагменти тексту програми, певні структури програми, що дозволяють зробити програму більш наочною, краще уявити її роботу.

8.10.1 Процедури

Процедура - це незалежна поименована частина програми, що допускає звернення по імені, яка отримує вихідні дані та повертає результати роботи через список формальних параметрів.

8.10.2 Функції

Функція - це підпрограма, результат роботи якої повертається через її ім'я. Тому при оголошенні функції їй у відповідність ставлять певний тип даних, тобто тип, повертаемого функцією значення. Загальна структура опису функцій аналогічна структурі опису процедур.

Функція активізується шляхом зазначення її імені зі списком фактичних параметрів у відповідному виразі. Фактичні параметри обов'язково повинні бути узгоджені з формальними.

Приклад 8.35. Скласти програму обчислення значення функції

$$z = \frac{\log_a x + \log_a y \log_b x + \log_b y}{\log_{a+b}(x+y)}.$$

```

Program Primer;
Var
    x,y,z,a,b: Real;
Function Fl(u,v:Real):Real;
Begin
    F:=ln(v)/ln(u);
    Fl:=F
End;
BEGIN
    Writeln('Введить значення a,b,x,y');
    Read(a,b,x,y);
    z:=(Fl(a,x)+Fl(a,y)*Fl(b,x)+Fl(b,y))/
        Fl(a+b,x+y);
    Writeln('a=',a,' ','b=',b);
    Writeln('x=',x,' ','y=',y,' ','z=',z);
END.

```

8.10.3 Область дії ідентифікаторів

Областю дії ідентифікатора називається та частина програми, де можливо його використання. Якщо ідентифікатор оголошений в основній програмі, і

тільки в ній, тоді він є глобальним і дія його поширюється на всі процедури і функції програми. Якщо ідентифікатор оголошений тільки в процедурі, то він є локальним у цій процедурі і глобальним у всіх вкладених процедурах та функціях. Глобальність або локальність ідентифікаторів визначається неоднозначно і може встановлюватися тільки по відношенню до конкретної процедури або функції. Розглянемо область дії ідентифікаторів в залежності від місця їх оголошення на наступному прикладі:

Приклад 8.36. Записати програму визначення області дії змінних, наведених у таблиці 8.18.

Таблиця 8.18 Таблиця визначення змінних A_0 , A_1 , A_2 , B , C , D у різних блоках програми.

Основна програма (Блок 0)		Процедура Proc_1 (Блок 1)		Процедура Proc_2 (Блок 3)	
Ідентифікатор	Тип	Ідентифікатор	Тип	Ідентифікатор	Тип
A_0	Byte	A_1	Real	A_2	Char
B	Byte				
C	Byte	C	Real		
D	Byte	D	Real	D	Char

```

Program Prog;                                     Блок 0
  Var    A0,B,C,D: Byte;

  Procedure Proc_1;                               Блок 1
    Var    A1,C,D: Real;

    Procedure Proc_2;                             Блок 2
      Var    A2,D: Char;
      begin
        Readln(A2,D);
        Writeln(A2,D,'- локальні (тип Char)');
        Writeln(A1,C,'- глобальні (тип Real)');
        Writeln(A0,B,'- глобальні (тип Byte)')
      end;

  Begin
    Readln(A1,C,D);

```

```

        Writeln(A1,C,D,' - локальні (тип Real)');
        Writeln(A0,B,' - глобальні (тип Byte)')
    End;

BEGIN
    Readln(A0,B,C,D);
    Writeln(C,D,' - локальні (тип Byte)');
    Writeln(A0,B,' - глобальні (тип Byte)')
END.

```

8.10.4 Параметри

Параметри в Turbo Pascal можуть передаватися як в програму (*in*), так і з неї (*out*) за значенням (*value*) або за адресою-посиланням (*addr*). Всі параметри і ті, що передаються в підпрограму, так і ті, що передаються з неї, можна розбити на три категорії:

- параметри-значення (*value-in*);
- параметри-змінні (*addr-inout*);
- параметри-константи (*addr-in*).

Всі розглянуті нижче способи передачі параметрів можуть використовуватися одночасно.

8.10.4.1. Параметри-значення.

У цьому випадку здійснюється передача параметрів за значенням. Це забезпечує збереження величини переданого параметра, тобто всі зміни параметра всередині процедури ніяк не позначаються на значенні переданого аргументу. У якості фактичного параметра можуть використовуватися змінні, константи та вирази відповідних типів. Не допускаються тільки файлові типи і похідні від них.

Приклад 8.37. Скласти програму обчислення значення

$$C_n^m = \frac{n!}{m!(n-m)!}$$

з використанням підпрограми-функції.

```

Program Primer;

Var
    m,n,k: Byte;
    Fn,Fm,Fk: Integer;
    C: Real;

Function Fact(p:Byte):Integer;
    Var I:Byte;
    Begin
        F:=1;
        For I:=1 to p do
            F:=F*I;
        Fact:=F
    End;

BEGIN
    Writeln('Введіть значення: m, n');
    Read(m,n);
    k:=n-m;
    C:=Fact(n)/(Fact(m)*Fact(k));
    Writeln('C=',C)

END.

```

8.10.4.2. Параметри-змінні.

Передача параметрів в цьому випадку виконується за посиланням, що означає передачу в процедуру адреси фактичного параметра, вказаного при зверненні до процедури. При цьому будь-які зміни параметра всередині процедури є зміна значення фактичного аргументу. В якості такого параметру при зверненні до процедури або функції не можуть стояти константи (звичайні) і вирази. Параметри, які повертаються процедурою або функцією, обов'язково повинні передаватися за посиланням, тобто мати описувач *Var*.

Приклад 8.38. Скласти програму обчислення значення

$$C_n^m = \frac{n!}{m!(n-m)!}$$

з використанням підпрограми-процедури.

```

Program Primer;
Var
  m,n,k: Byte;
  Fn,Fm,Fk: Integer;
  C: Real;
Procedure Fact(p:Byte; Var F:Integer);
Var
  I:Byte;
Begin
  F:=1;
  For I:=1 to p do
    F:=F*I
  End;
BEGIN
  Writeln('Введіть значення: m, n');
  Read(m,n);
  k:=n-m;
  Fact(n,Fn);
  Fact(m,Fm);
  Fact(r,Fk);
  C:=Fn/(Fm*Fk);
  Writeln('C=',C)
END.

```

8.10.4.3. Параметри-константи.

Параметри-константи доцільно використовувати, коли необхідно передавати в підпрограму дані, що займають великий обсяг пам'яті, при чому зміна значень переданих даних у процесі виконання підпрограми неприпустимо. У цьому випадку в підпрограму передаються адреси фактичних параметрів, за якими дозволяється тільки брати значення цих параметрів, не змінюючи їх.

Параметри-константи не можуть передаватися в інші підпрограми у вигляді фактичних параметрів. В якості параметрів-констант можуть використовуватися як змінні, так і константи будь-якого типу за винятком файлового.

8.10.4.4. Параметри без типу.

Параметр без вказівки типу може передаватися тільки за адресою як параметр-змінна або як параметр-константа. У цьому випадку фактичний параметр може бути будь-яким посиланням на змінну або константу незалежно від їх типу. У підпрограмі формально не типізований параметр не може бути використаний, поки не буде виконано його приведення до якогонебудь типу. Відповідальність за коректне приведення до типу покладається на програміста, оскільки компілятор не може перевірити сумісність типів.

8.10.4.5. Відкриті параметри-масиви.

Turbo Pascal 7.0 надає можливість працювати з відкритими масивами при передачі даних у функції та процедури. У підпрограмі такі масиви обов'язково повинні використовуватися як масиви з нульовою границею індексу. Верхня межа індексу формального параметра-масиву може бути визначена стандартною функцією *High*. За допомогою функції *SizeOf* встановлюється розмір фактичного параметра-масиву. Передані в підпрограму фактичні параметри повинні збігатися з описаними формальними параметрами за типом, збіг параметрів по розмірності не обов'язково.

8.11 КОРИСТУВАЦЬКІ ТИПИ

До користувацьких відносяться наступні типи:

- порядкові типи (та інтервальний);
- структуровані типи.

8.11.1 Порядкові типи

Порядкові користувацькі типи включають в себе:

- перераховуваний тип;
- інтервальний тип.

8.11.1.1. Перераховувані типи.

Перераховувані типи визначають впорядковані набори значень. Опис перераховуваного типу містить у собі список елементів-ідентифікаторів, укладених в круглі дужки.

Ідентифікатори з упорядкованого набору значень вважаються константами відповідного перераховуваного типу, але не рядковими і не символьними константами, тому в лапки не беруться. Повторення констант в різних перераховуваних типах в межах одного блоку вважається не припустимим. Порядковий номер перераховуваної константи визначається її позицією у списку ідентифікаторів при оголошенні. Перша перераховувана константа в списку має порядковий номер 0. Над значеннями перераховуваних типів визначені тільки операції порівняння. До них застосовні стандартні функції *Ord*, *Pred* і *Succ*.

Приклад. 8.39.

Type

```
Color = (Red, Yellow, Blue, Green) ;
WeekDay = (Monday, Tuesday, Wednesday,
            Thursday, Friday, Saturday, Sunday)
```

Для вводу-виводу значень перераховуваного типу не можуть бути використані стандартні процедури *Read*, *Readln*, *Write*, *Writeln*.

8.11.1.2. Інтервальні типи.

Інтервальний тип являє собою інтервал (діапазон) значень порядкового (базового) типу. В описі базового типу вказується дві константи - відповідно найменше та найбільше значення інтервалу. Обидві константи повинні належати одному з скалярних типів. Значення першої константи має бути обов'язково менше значення другої. Інтервальний тип успадковує всі властивості основного типу.

Зауваження. Опис інтервального типу не повинен починатися з круглої дужки.

Приклад 8.40.

Type

```

Int = 0..99;
Ind = -128..127;
Alf = 'A'..'Z';
Workday = Monday..Friday;

```

8.11.2 Структуровані типи

До структурованих типів відносяться наступні типи:

- масиви (array);
- множини (set);
- записи (record);
- файли (file).

8.11.2.1. Масиви

Масив це визначена за розміром сукупність однотипних елементів. Припустимими типами індексу є всі дискретні типи, перераховувані та інтервальні, за винятком Longint та інтервальних типів, побудованих на основі типу Longint. Елементами масиву можуть бути значення будь-якого типу. Число елементів у масиві дорівнює кількості значень у кожному типі індексу. Розмірність (число індексів) масиву не обмежена.

Приклад 8.41. Опис масивів.

```

array[1..100] of Real - одновимірний масив;
array[Boolean] of array[1..10] of Real або
array[Boolean,1..10] of Real - тривимірний масив.

```

Ім'я масиву є єдиним для всіх його елементів. Для доступу до елементів масиву необхідно вказати ім'я масиву з індексом або для багатовимірних масивів зі списком індексів у дужках. В якості індексів можуть виступати довільні вирази того ж типу, що й індекси. Одномірні масиви, що складаються з елементів символьного типу (Char) та які мають в якості індексу обмежений цілий тип, називаються рядками array [1 .. 3] of char. Строковим масивам можна присвоювати конкретні значення, що утворені з ланцюжка символів, укладених у апострофи. Допускається формування рядків з використанням десяткових кодів символів. Слід пам'ятати, що присвоювання рядків символьним масивам передбачає точну відповідність довжин рядків і розмірів масивів.

Приклад 8.42. Скласти програму, що виконує перевірку рівності заданих змінних або масивів. Протестувати програму при наступних умовах:

Дано: $A=\{1,2,3,4,5,4,3,2,1,0\};$

$B=\{1,2,3,4,5,1,2,3,4,5\}.$

Порівняти:

1. $A \neq B$.
2. Перші $k=5$ елементів A з першими $k=5$ елементами B .
3. Перші $k=7$ елементів A з першими $k=7$ елементами B .
4. $A=1$ і $B=1$ як прості змінні ($m=1, k=1$)

Program Primer;

Type

Massive=array[1..100] of byte;

Var

A,B:Massive; m,k,i:Byte; Rez:Boolean;

Function Comp(Var Ms1,Ms2;

Sz:Byte):Boolean;

Type Pr=array[1..Maxint] of Byte;

Var n:Byte;

Begin

n:=0;

While (n<=Sz) and

(Pr(Ms1)[n]=Pr(Ms2)[n])

do Inc(n);

Comp:= n-1=Sz;

End;

BEGIN

```

Writeln('Введіть розмір масива m');
Read(m);
Writeln('Введіть значення A');
For i:=1 to m do
    Read(A[i]);
Writeln('Введіть значення B');
For i:=1 to m do
    Read(B[i]);
If m=1 Then k:=1
Else
    begin
        Writeln('Введіть кількість елементів, що
                порівнюються k');
        Read(k)
    end;
Rez:=Comp(A,B,k);
If Rez and (k=m) and (k<>1)
Then Writeln('Масиви A і B рівні')
Else
    If Rez and (k<>m)
    Then Writeln(k, ' перших елемента
                A і B рівні')
Else
    If Rez and (k=1)
    Then Writeln('A рівне B')
    Else
        If (k=m)
        Then Writeln('A не рівне B')
        Else Writeln(k, ' перших

```

елемента А и В не рівні')

END.

Приклад 8.43. Скласти програму сортування в алфавітному порядку кожного рядка заданої матриці A розміром $m \times n$, елементами якої є рядкові латинські літери.

```
Program Primer;
Const
    R=10; C=10;
Type
    Massive=array[1..R,1..C] of Char;
Var
    A: Massive;
    m,n,i,j,k: Byte;
    Pt: Char;

BEGIN
    Writeln('Введіть кількість рядків m
            і стовпців n');
    Readln(m,n);
    Writeln('Введіть значення A');
    For i:=1 to m do
        For j:=1 to n do
            Read(A[i,j]);
    For i:=1 to m do
        begin
            For k:=1 to n-1 do
                For j:=k+1 to n do
                    If A[i,k]>=A[i,j] Then
                        begin
```

```

        Pt:=A[i,k];
        A[i,k]:=A[i,j];
        A[i,j]:=Pt
    end;

end;

For i:=1 to m do
begin
    For j:=1 to n do
        Write(A[i,j], ' ', ' ');
        Writeln
    end
end.

```

8.11.2.2. Множини

Множиною в Turbo Pascal називається кінцева сукупність різних об'єктів одного типу, який називається базовим. До базового типу відносяться всі скалярні типи, крім дійсного. Максимальна кількість значень базового типу множини називається його потужністю

Базовий тип не повинен мати більше 256 можливих значень і порядкові значення верхньої і нижньої межі базового типу не повинні перевищувати діапазону від 0 до 255. Такі типи, як *ShortInt*, *Integer*, *LongInt*, *Word*, перераховувані та інтервальні типи, утворені з них, використовувати в якості базових типів множин не допускається. Звичайно в якості базових типів множин використовуються перераховувані типи, *byte*, *char*, або інтервальні типи на їх основі. У Pascal-програмі множина задається у вигляді списку елементів, вкладених у квадратні дужки. У дужках може бути жодного елементу (порожня множина), один або більше елементів. В якості елементів можуть використовуватися константи, змінні, вирази, значення яких належать базового типу.

Приклад 8.44. Скласти програму, що візуально демонструє результати виконання основних операцій над множинами. За основу взяти наступну сукупність кольорів: *чорний, синій, зелений, блакитний, червоний, фіолетовий, коричневий, світло-сірий, темно-сірий, світло-синій, світло-зелений, світло-блакитний, світло-червоний, світло-фіолетовий, світло-коричневий, білий.*

Program Primer;

Uses Crt;

Type

*Color = (Black,Blue,Green,Cyan,Red,
Magenta,Brown,LightGray,
Darkgray,LighBlue,LightGreen,
LightCyan,LightRed,
LightMagenta,Yellow,White);*

Palette = set of Color;

Var

A,B,P,R,S,U: Palette;

Clr: Color;

Procedure Rez (M:Palette;Ch:Char);

Var

k: 0..15;

Begin

Write(' ',Ch,' ',#196,' ');

For Clr:=Black to White do цикл по всіх кольорах

If Clr in M Then

begin

k:=Ord(Clr); перетворення в цілий тип

TextColor(k); завдання кольору символу

Write(' ',Ch) виведення символу на екран

end;

TextColor(15); завдання білого кольору символу

Writeln; пропуск рядка

End;

BEGIN

```

A:=[Brown,LightGray,DarkGray,White,
LightBlue,LightCyan,Red,Magenta,
LightGreen];      потужність множини A дорівнює 9
Rez (A, 'A' );
B:=[Blue,Cyan,Green,Magenta,LightBlue,
LightGreen,LightRed,Yellow];
Rez (B, 'B' );      потужність множини B дорівнює 8
S:=A+B;            об'єднання, потужність множини S дорівнює 14
Rez (S, 'S' );
R:=A-B;            різниця, потужність множини R дорівнює 6
Rez (R, 'R' );
U:=A*B;            перетин, потужність множини U дорівнює 3
Rez (U, 'U' );
writeln
END.

```

8.11.2.3. Записи

Запис містить певне число компонент (полів), але на відміну від масиву, ці компоненти можуть мати різні типи. Кожному полю запису присвоюється своє ім'я і визначається тип значення цього поля. Звернення до поля здійснюється за допомогою ідентифікатора змінної та ідентифікатора поля, розділених крапкою. Такий ідентифікатор називається складеним. Різновидом записів є "записи з варіантами" (варіантна частина), в яких можна задавати тип, що містить визначення декількох варіантів. Варіанти можуть відрізнятися як числом компонент, так і їх типами. Запис може містити лише одну варіантних частину.

Приклад 8.45. Написати програму обліку засобів обчислювальної техніки (ЗОТ) підприємства, що перебуває в користуванні п'яти відділів (підрозділів). Кожен запис має містити назву (Nazv), рік випуску (GodVip), термін гарантії (GrntSrok). Залежно від виду техніки (VidTehn) повинні бути зазначені наступні параметри:

- для комп'ютера - процесор (CPU), материнська плата (MB), обсяг

оперативної пам'яті (RAM), обсяг вінчестера (HDD), модель дисплея (ModelDsp), розмір екрану (RazmerEkr);

- для принтера - фірма-виробник принтера (FirmaPrn), тип принтера (TipPrn), формат аркуша паперу (Format), можливість кольорового друку (Color).

Необхідно вивести на екран всі дані по кожному відділу по комп'ютерах, що мають тактову частоту не нижче 300 МГц і об'єм оперативної пам'яті 64 Мбайт і більше, а також по принтерам, що працюють з папером формату А1.

```
Program Primer;
```

```
Const m=5;
```

```
Type
```

```
Str = string[25];
```

```
PersonalComp = record
```

```
    CPU : word;
```

```
    MB : str;
```

```
    RAM : word;
```

```
    HDD : word;
```

```
    ModelDsp : str;
```

```
    RazmerEkr : byte
```

```
end;
```

```
Printer = record
```

```
    FirmaPrn : Str;
```

```
    TipPrn : Str;
```

```
    Format : byte;
```

```
    Color : byte
```

```
end;
```

```
Svt = record
```

```

    Nazv          : Str;
    GodVip        : word;
    GrntSrok      : Byte;
    Case VidTehn  : byte of
        1: ( PC    : PersonalComp );
        2: ( Prnt  : Printer );
    end;

Var
    i,VidTehn:Byte;
    Otdel: array [1..10] of SVT;

BEGIN
    For i:=1 to m do
        with Otdel[i] do begin
            Writeln('Введіть вид техніки:');
            Writeln('    1 - комп'ютер');
            Writeln('    2 - принтер');
            Readln(VidTehn);
            Writeln('Введіть Nazv');
            Readln(Nazv);
            Writeln('Введіть GodVip');
            Readln(GodVip);
            Writeln('Введіть GrntSrok');
            Readln(GrntSrok);
            Case VidTehn of
                1: beginwith PC do begin
                    writeln('Введіть тактову частоту процесора');
                    Readln(CPU);

```



```

Writeln('Введіть материнську плату');
Readln(MB);
Writeln('Введіть об'єм пам'яті в Мбайтах');
Readln(RAM);
Writeln('Введіть об'єм вінчестера в Мбайтах');
Readln(HDD);
Writeln('Введіть модель дисплея');
Readln(ModelDsp);
Writeln('Введіть розмір екрану в дюймах');
Readln(RazmerEkr);

end;

end;

2: begin      with Prnt do begin
Writeln('Введіть фірму-виробника принтера');
Readln(FirmaPrn);
Writeln('Введіть тип принтера:');
Writeln('          1 - матричний');
Writeln('          2 - струменевий');
Writeln('          3 - лазерний');
Readln(TipPrn);
Writeln('Введіть формат паперу:');
Writeln('          1 - A1');
Writeln('          2 - A2');
Writeln('          3 - A3');
Writeln('          4 - A4');
Readln(Format);
Writeln('Введіть 1, якщо принтер чорно-білий');
Writeln('          2, якщо принтер кольоровий');

```

```

        Readln(Color);
    end;
end;
end;

end;

writeln;
For i:=1 to m do
    with otdel[i] do begin
        Writeln('Биддил №',i);
        with PC do
            if (CPU>=300) and (Ram>=64)
            then
                Writeln(Nazv,' ',
                    GodVip,' ',
                    GrntSrok,' ',
                    CPU,' ',
                    RAM,' ',
                    HDD,' ',
                    ModelDsp,' ',
                    RazmerEkr);
        with Prnt do
            if Format =1
            then
                Writeln(Nazv,' ',
                    FirmaPrn,' ',
                    TipPrn,' ',
                    Format,' ',
                    Color)

```

end;

END.

8.11.2.4. Файли

Поняття файлу в мові Turbo Pascal пов'язано з одного боку з певною областю зовнішньої пам'яті, яка містить будь-яку інформацію (фізичні файли), з іншого боку - з файловими змінними певного типу (логічні файли).

Використовувані в Pascal-програмі файлові змінні повинні відповідати фізичним файлам або логічним пристроям, пов'язаним з введенням-виведенням інформації. Так клавіатурі комп'ютера за замовчуванням ставиться у відповідність файл зі стандартним ім'ям *Input*, а екрану дисплея - *Output*. Ці пристрої автоматично відкриваються при запуску програми і, також автоматично закриваються по закінченню її роботи.

Фізичний файл являє собою розташовані в певній послідовності байти пам'яті на магнітному, магнітооптичному або іншого виду зовнішньому носії інформації. Логічний файл - це послідовність однотипних елементів, число яких визначає довжину файлу. Елементи файлу розташовуються зліва направо і нумеруються починаючи з нуля. Після останнього елемента файлу автоматично записується спеціальний символ ASCII # 26 (Ctrl + Z) - маркер кінця файлу (EOF).

Основною особливістю файлів є змінність їх довжини в процесі роботи з ними. Робота з файлами полягає у зчитуванні що містяться в них елементів і запису нових. Для визначення елемента послідовності, з яким виконується операція зчитування або запису, вводиться поняття покажчика поточної позиції у файлі. У Turbo Pascal є три категорії файлів:

- типізовані;
- нетипізовані;
- текстові.

Типізовані файли містять елементи певного типу. Найчастіше ці файли складаються із записів і застосовуються для створення різних баз даних. Нетипізовані файли представляються як сукупність елементів довільного типу з обговореним розміром цих елементів. При визначенні нетипізованих файлів слово *of* і тип опускаються. Використання нетипізованих файлів в Turbo Pascal дозволяє здійснювати доступ до будь-яких дискових файлів незалежно від їхньої структури.

Файли, що містять символи, об'єднані в рядки (довжиною від 0 до 256 символів), називаються текстовими. Кожен рядок текстового файлу закінчується спеціальними керуючими символами ASCII # 13 (CR - повернення каретки) і # 10 (LF - переклад рядка). Оскільки цей тип файлу часто використовується в програмах, то він прийнятий в якості стандартного типу *Text* і, отже, описувати його не треба. При цьому слід мати на увазі, що тип *Text* не еквівалентний типу *File of Char*, оскільки *File of Char* представляє собою неподіляему на рядки послідовність символів.

У Turbo Pascal розрізняють два види файлів: послідовного та довільного доступу. Зазвичай всі файли вважаються файлами послідовного доступу. Файлами ж довільного доступу можуть бути тільки типізовані і нетипізовані файли.

8.12 ОПИС ТИПІЗОВАНОЇ КОНСТАНТИ

Крім звичайних констант в Turbo Pascal широко використовуються типізовані константи. В описі типізованої константи вказується як значення константи, так і її тип. Типізовані константи можуть використовуватися в програмі як змінні, але не можуть застосовуватися в оголошеннях інших констант або типів. Типізовані константи можуть бути як стандартних, так і структурованих типів (масив, множина, запис та ін.).

8.12.1 Константи стандартних типів

До констант стандартних типів відносяться наступні константи:

- Числові константи (цілі і дійсні).
- Логічні константи, які приймають два значення *True* і *False*.
- Символьні константи - це якийсь один символ таблиці кодів ASCII, укладений в апострофи.
- Рядкові константи - це послідовність символів, укладена в апострофи. Максимальна довжина строкової константи - 255 символів. Рядкові константи можуть бути записані як послідовність відповідних символів кодів. При цьому перед кожним кодом записується символ #.

Приклад 8.46. Опис констант стандартних типів.

Const

```

Last_Ind  : Word = 1000;
Nach_Usl  : Real = 0.002;
Zvuk_Sign : Char = #7;
Str_Koment: String[15] = 'Результати:';

```

8.12.2 Константи-масиви

При описі констант цього типу значення елементів масиву по кожному індексу задаються в окремих дужках і розділяються комами. Елементи, що визначаються крайнім (правим) індексом, задаються в самих внутрішніх дужках.

Приклад 8.47. Опис констант-масивів.

```

Const
Mas_Ch1: array [1..12] of Char = 'Кінець задачі';
Mas_Ch2: array [1..6] of Char = ('д', 'а', 'н', 'і');
Mas_D   : array [1..5] of real = (1.12, 1.25, 3.8,
                                4.76, 2.4);
Mas_2D: array [1..3, 1..4] of Byte =
      ( (1, 2, 3, 4),
        (2, 3, 4, 5),
        (3, 4, 5, 6) );
Mas_3D: array [1..2, 1..3, 1..4] of Word =
      ( ( (1, 2, 3, 4),
          (2, 3, 4, 5),
          (3, 4, 5, 6) )
        ( (4, 5, 6, 7),
          (5, 6, 7, 8),
          (6, 7, 8, 9) ) ) );

```

8.12.3 Константи-множини

Кожен елемент константи цього типу є або окремою константою відповід-

ного типу, або діапазон значень, що визначається двома константами, розділеними двома точками.

Приклад 8.48. Опис констант-множин.

Type

Symbol = set of Char;

Digit = set of 0..9;

WeekDay = (Mon, Tue, Wed, Thu, Fri, Sat, Sun);

Const

Prostoe : *Digit* = [2, 3, 5, 7];

Operate : *Symbol* = ['+', '-', '*', '/'];

ChetDay : *WeekDay* = [Tue, Thu, Sat];

WorkDay : *WeekDay* = [Mon..Fri];

Data : set of 1..31 = [1..11, 21..31];

LatAlf : set of Char = ['a'..'z', 'A'..'Z'];

8.12.4 Константи-записи

В описі константи-запис вказуються значення всіх полів запису та їх ідентифікатори. Поля вказуються в тому ж порядку, в якому вони слідуєть у описі типу.

Приклад 8.49. Опис констант-записів.

Type

Zap = record

R : Real;

B : Boolean

C : Char;

end;

MasZap = array [1..2] of *Zap*;

```

ZapZap = record;
  M : array {1..4} of Char;
  S : String;
  Z : Zap;
end;

```

Const

```

CZp    : Zap    =
  ( R: 1.45; B: True; C: '$' );

```

```

CMas   : MasZap =
  ( (R: 0.48; B: True; C: 'Y');
    (R: 0.0; B: False; C: 'N') );

```

```

CZpZp  : ZapZap =
  ( M : ('+', '-', '*', '/');
    S : 'Turbo Pascal Program';
    Z : (R: 2.74; B: True; C: 'Y') );

```

8.13 РОБОТА З РЯДКАМИ

Рядки займають проміжне місце між простими і структурованими типами даних. Значеннями цього типу є послідовності будь-яких символів, укладених в одинарні лапки.

8.13.1 Операції з рядками

Рядки можна присвоювати, порівнювати, поєднувати, вводити і виводити за допомогою стандартних процедур введення-виведення. До кожного окремого символу рядка застосовні ті ж операції, що й до змінної типу *Char*. Для

доступу до будь-якого символу рядка необхідно вказати ім'я рядкової змінної і в квадратних дужках номер позиції символу в рядку.

Приклад 8.50. Скласти програму, в результаті роботи якої має бути складена пропозиція «Операция присваивания.» (рос.м.) з окремих слів «Операция» (рос.м.) та «Присваивание» (рос.м.).

```
Program Primer;
  var Str_1,Str_2,Str_3,Str_4 :string;
      k: byte;
BEGIN
  Str_1   := 'Операция';
  Str_2   := 'Присваивание';
  Str_3   := Str_2;
  Str_3[1] := 'п';
  writeln(Str_2 < Str_3); {Виведення: True}
  k       := length(Str_3);
  Str_3[k] := 'я';
  Str_4   := Str_1 + #32 + str_3 + ' .';
  Writeln(Str_4)
      { Виведення: Операция присваивания. }
END.
```

8.13.2 Засоби обробки рядків

Turbo Pascal для обробки рядків надає цілий ряд функцій і процедур. Розглянемо деякі з них.

8.13.2.1. Визначення довжини рядка

Визначення фактичної довжини рядка здійснюється за допомогою наступної функції:


```
function Length(St: String): Integer
```

Ця функція повертає кількість всіх вхідних в рядок *St* символів, включаючи пропуски.

8.13.2.2. Копіювання фрагмента рядка

Для копіювання фрагмента однієї строкової змінної в іншу використовується наступна функція:

```
function Copy(St:String;Poz,N:Integer):String
```

Ця функція виділяє з рядка *St* підрядок довжиною в *N* символів, починаючи з позиції *Poz*. Якщо значення *Poz* або *Poz + N* перевищує довжину рядка, то результатом виконання функції *Copy* в першому випадку буде рядок нульової довжини, у другому - останні символи рядка *St*, починаючи з символу *Poz*.

Приклад 8.51. Скласти програму, в результаті роботи якої повинно бути отримано декілька слів шляхом копіювання окремих частин заданого слова «БАЛЮСТРАДА» (*рос.м.*).

```
Program Primer;
```

```
Const
```

```
Str: string[10] = 'БАЛЮСТРАДА';
```

```
Var
```

```
Str_1,Str_2,Str_3,Str_4 : string[10];
```

```
BEGIN
```

```
Str_1 := Copy(Str,1,3);
```

```
Str_2 := Copy(Str,3,6);
```

```
Str_3 := Copy(Str,5,6);
```

```
Str_4 := Str_1+Copy(Str,3,1)+Copy(Str,8,3);
```

```
writeln(Str_1,' ',Str_2,' ',Str_3,' ',Str_4);
```

```
{ Виведення: БАЛ ЛЮСТРА СТРАДА БАЛЛАДА }
```

```
END.
```

8.13.2.3. Перетворення символу з рядкового в прописний

Перетворення символу будь-якої букви з рядкового в прописний здійснюється за допомогою функції *UpCase*. Ця функція має наступний вигляд:

```
Function UpCase (Ch:Char):Char
```

Функція *UpCase* перетворює тільки букви латинського алфавіту.

8.13.2.4. Визначення позиції заданого фрагменту

Для виявлення входження певної послідовності символів (підрядка) в заданий рядок використовується наступна функція:

```
Function Pos (St1,St2:String):Byte
```

Якщо зазначена послідовність символів *St1* є фрагментом вказаного рядка *St2*, то імені функції *Pos* присвоюється значення, рівне номеру символу, з якого починається перший фрагмент (якщо він у рядку не один). В іншому випадку *Pos* присвоюється значення 0.

8.13.2.5. Вставка та видалення символів.

Для вставки (видалення) певного фрагмента символів в заданий рядок використовується процедура *Insert (Delete)*. Процедури *Insert* і *Delete* мають наступний вигляд:

```
Procedure Insert (St1:String; var St2:String,  
  
Poz:Integer)
```

```
Procedure Delete (var St:String; Poz,N:Integer)
```

Після виконання процедури *Insert* послідовність символів рядка *St1* буде вставлена в рядок *St2*, починаючи з позиції *Poz*. Виконання процедури *Delete* видаляє *N* символів рядка *St*, починаючи з символу з номером *Poz*.

Приклад 8.52. Скласти програму перетворення слова «Балюстрада» (рос.м.) в слово «Балласт» (рос.м.) із застосуванням функції *Pos*, процедур *Insert* і *Delete*.

```
Program Primer;  
Const  
  Str:string[10]='Балюстрада';  
Var  
  k:byte;  
BEGIN  
  k:=Pos('пада',Str);  
  Delete(Str,k,4);  
  writeln(Str);          { Виведення: Балюст }  
  k:=Pos('ю',Str);  
  Delete(Str,k,1);  
  writeln(Str);          { Виведення: Балст  }  
  Insert('ла',Str,4);  
  writeln(Str);          { Виведення: Балласт }  
END.
```

8.13.2.6. Перетворення числа в рядок і назад

Для перетворення числового значення в рядковий застосовується процедура *Str*. Ця процедура має наступний вигляд:

```
Procedure Str(X[:w[:d]]; var St:String)
```

При виконанні процедури *Str* число *X* перетворюється в рядок *St*. Кваліфікатори *w* і *d* є необов'язковими параметрами та позначають відповідно довжину поля і число цифр після коми.

Процедура *Val* перетворює рядок у величину цілочисельного або дійсного типу. Процедура *Val* має наступний вигляд:

```
Procedure Val(St:String; var V;  
              var Code: Integer)
```

Виконання процедури *Val* полягає в перетворенні значення *St* у величину цілого або дійсного типу *V*. При цьому пропуски в *St* допускаються тільки на початку рядка. Цілочисельна змінна *Code* зберігає нульове значення, якщо при перетворенні помилки не виявлено, в іншому випадку *Code* приймає значення, рівне номеру позиції першого помилкового символу. Значення *V* в цьому випадку буде не визначено.

Приклад 8.53. Написати програму перетворення числових значень

A=124; B=257,89; C=3,1·10³

в рядковій та строковій

R='10101'; S='54237'; T='12.45'

в числовій.

```

Program Primer;
Var
    A,Cod                      : Integer;
    B,C,V_1,V_2,V_3           : Real;
    R,S,T,St_1,St_2,St_3      : string;
BEGIN
    A := 124;                  R := '10101';
    B := 257.89;               S := '5 4237';
    C := 3.1e3;                T := '12.45';
    Str(A,St_1);               Val(R,V_1,cod);
    writeln(St_1,' ',V_1);
    { Виведення: 124           1.0101000000E+04}
    Str(B,St_2);               Val(S,V_2,cod);
    writeln(St_2,' ',V_2);
    { Виведення: 2.5789000000E+02   5.4237000000E+04}
    Str(C:8:3,St_3);           Val(T,V_3,cod);
    writeln(St_3,' ',V_3);
    {Виведення: 3100.000           1.2450000000E+01}

```

writeln

END.

8.14 РОБОТА З ФАЙЛАМИ

Робота з файлами починається з установки зв'язку файлової змінної *f* з ім'ям файлу на диску за допомогою такої процедури:

Procedure Assign (Var f, String)

Тут назва файлу вказано у рядковій константі *String*.

Наприклад,

Assign (PascalNameFile,

'G:\BUD_10-1\Antonov\program1')

або

Assign (PascalNameFile, 'prn'),

де *PascalNameFile* - ім'я файлової змінної в Pascal-програмі, *G:* - ім'я диска, *BUD_10-1* - ім'я каталогу групи, *Antonov* - особистого підкаталогу, *program1* - ім'я файлу, *prn* - принтер (замість *prn* можливо *lpt1*).

Підготовка файлу до роботи (запис або читання) здійснюється процедурою

Procedure Rewrite (Var f).

Ця процедура створює на диску новий (порожній) файл, ім'я якого задано процедурою *Assign*. Показчик поточної позиції встановлюється в початок файлу. Якщо файл з такою назвою вже існував, то він знищується. Після виконання процедури *Rewrite* файл стає доступним як для запису, так і для читання.

Функція

Function IOResult Integer

пов'язана з результатом останньої виконаної операції введення-виведення (читання з файлу або запису в нього) і повертає число 0, якщо операція вводу-виводу завершилася благополучно, і інше число - в іншому випадку.

Підготовку створеного раніше файлу для читання або запису виконує процедура

Procedure Reset (Var f)

Ця процедура знаходить вже існуючий на диску файл і відкриває його для роботи, розміщуючи покажчик поточної позиції у початок файлу. Якщо файл у вказаному на диску каталозі не знаходиться, то при встановленій директиві компілятора `{SI+}` видається повідомлення про помилку введення.

При завершенні програми всі відкриті файли обов'язково повинні бути закриті за допомогою процедури

Procedure Close (Var f).

Ця процедура закриває відкритий файл, пов'язаний з файловою змінною *f*. Після останнього елемента файлу автоматично записується ознака кінця файлу.

Функція

Function Eof (Var f)

приймає значення *True*, якщо вказівник поточної позиції знаходиться на маркері кінця файлу (*End Of File*).

8.14.1 Текстові файли

При роботі з текстовими файлами в Turbo Pascal додатково до розглянутих вище використовуються наступні процедури і функції:

Procedure Append (Var f:Text)

- відкриває існуючий файл, пов'язаний з файловою змінною *f* для поповнення його новою інформацією.

Procedure Read (Var f:Text; x₁, x₂, ..., x_n)

- зчитує значення з файлу, пов'язаного з файловою змінною *f*, і присвоює їх відповідно змінним *x₁, x₂, ..., x_n*.

Procedure Readln (Var f:Text; x₁, x₂, ..., x_n)

- зчитує рядок значень з файлу, пов'язаного з файловою змінною *f*, і присвоює їх відповідно змінним *x₁, x₂, ..., x_n*.

Procedure Write (Var f:Text; x₁, x₂, ..., x_n)

— записує значення змінних *x₁, x₂, ..., x_n* в файл, пов'язаний з файло-

вою змінною f . Змінні можуть бути цілого, дійсного, символьного, рядкового або булевих типів.

Procedure Writeln (Var f :Text; x_1, x_2, \dots, x_n)

- записує у вигляді рядка значення змінних x_1, x_2, \dots, x_n в файл, пов'язаний з файловою змінною f . Змінні можуть бути цілого, дійсного, символьного, рядкового або булевих типів.

Function Eoln (Var f :Text):Boolean

- повертає значення *True*, якщо поточна позиція у файлі знаходиться на мітці кінця рядка.

Function SeekEoln (Var f :Text):Boolean

- повертає значення *True*, якщо поточна позиція у файлі знаходиться на мітці кінця рядка (при цьому пропускаються в кінці рядка символи пропуску і табуляції).

Function SeekEof (Var f :Text):Boolean

— повертає значення *True*, якщо поточна позиція у файлі знаходиться на маркері кінця файлу.

— При виконанні введення або виведення стандартними засобами генерується код, перевіряючий результат звернення до процедури введення-виведення. У разі виникнення помилок програма припиняє роботу і виводить на екран повідомлення. Щоб уникнути зупинки програми при виявленні помилок слід до звернення до процедури введення-виведення скасувати генерацію коду, за допомогою директиви компілятора $\{SI-\}$.

Приклад 8.54. На диску F : є текстовий файл $\#A.dat$, що містить цілі числа. Написати програму, що дозволяє прочитати з файлу $\#A.dat$ всі дані, перетворити їх в дійсні і записати по n чисел у кожному рядку в форматі $V: 10:2$ до новоствореного на цьому ж диску текстовий файл $\#B.rez$.

Program Primer;

Var Nf_A, Nf_B : Text;

V : Real;

i, k, n : byte;

BEGIN

{SI-}

Assign (Nf_A, '#A.dat');

```

Assign(Nf_B, '#B.rez');
Reset(Nf_A);
{$I+}
if IOResult <> 0 then
    Writeln('Файл #A.dat не знайдено');
    else
begin
    Writeln('Введіть значення n');
    Readln(n);
    Rewrite(Nf_B);
    k:=0;
    While k=0 do
        begin
            For i:=1 to n do
                if not SeekEof(Nf_A) then
                    begin
                        Read(Nf_A,V);
                        Write(Nf_B,V:10:2)
                    end
                else
                    k:=1;
            Writeln
        end
    end;
    Close(Nf_A);
    Close(Nf_B)
END.

```

8.14.2 Типізовані файли

При роботі з типізованими файлами можна використовувати процедури *Assign*, *Reset* і *Rewrite*. На відміну від текстових типізовані файли, відкриті процедурою *Reset*, доступні не лише для читання, але і для запису. Нижче наводяться додаткові процедури і функції, які використовуються при роботі з типізованими файлами.

Procedure Read(f, x₁, x₂, ..., x_n) - зчитує з файлу, пов'язаного з файловою змінною *f*, компонентів x_1, x_2, \dots, x_n зазначеного типу.

Procedure Write(f, x₁, x₂, ..., x_n) - записує в файл, пов'язаний з файловою змінною *f*, компонентів x_1, x_2, \dots, x_n зазначеного типу.

Procedure Seek(Var f; N: Longint) - встановлює поточну позицію у файлі, пов'язаному з файловою змінною *f*, на елемент з номером *N* (*N* - вираз цілого типу).

Procedure Truncate(Var f) - видаляє частину файлу, пов'язаного з файловою змінною *f*, починаючи з поточної позиції, після чого поточна позиція стає кінцем файлу.

Function FilePos(Var f): Longint - повертає поточну позицію у файлі, пов'язаному з файловою змінною *f*.

Function FileSize(Var f): Longint - повертає поточний розмір файлу, пов'язаного з файловою змінною *f*.

Типізовані файли набагато коротше аналогічних текстових файлів і, відповідно, займають менше місця на диску. Типізовані файли в основному використовуються для зберігання числових даних.

Приклад 8.55. Скласти програму обчислення значень функцій синуса і косинуса в *n* точках відрізка $x \in [x_n, x_k]$ із записом отриманих результатів для порівняння в текстовий і типізований файли.

```
Program primer;  
  Const n=10;  
  Var
```

```

Nf_A   : Text;
Nf_B   : file of real;
x,xn,xk,dx,Sn,Cs: real;
i       : byte;

BEGIN

Assign(Nf_A, 'Rez_A.txt');

Assign(Nf_B, 'Rez_B.dig');

Rewrite(Nf_A);

Rewrite(Nf_B);

xn:=0.0;   xk:=PI;   dx:=(xk-xn)/n;

for i:=0 to n do
begin
    x:=xn+i*dx;

    sn:=sin(x);   Cs:=cos(x);

    Writeln(Nf_A,x,Sn,Cs);

    Write(Nf_B,Sn,Cs)

end;

Close(Nf_A);

Close(Nf_B)

END.

```

Після виконання даної програми на поточному диску будуть створені текстовий файл *Rez_A.txt* і типізований *Rez_B.dig*. Типізований файл *Rez_B.dig* набагато коротше текстового файлу *Rez_A.txt*. Обидва файли містять однакову інформацію, однак при перегляді з'ясовується, що в текстовому файлі результати відображаються в зручному для сприйняття вигляді, в типізованому - у вигляді хаотичного набору символів, відповідному машинному поданню інформації.

8.14.3 Файли без типу

Для роботи з нетипізованими файлами використовуються такі додаткові процедури і функції:

Procedure BlockRead(Var f:File; Var Buf; N:Word)

- зчитує N записів в змінну *Buf* з файлу, пов'язаного з файловою змінною *f*.

Procedure Seek(Var f; N:Longint)

- встановлює поточну позицію у файлі, пов'язаному з файловою змінною *f*, на елемент з номером N (N - вираз цілого типу).

Procedure Truncate(Var f)

- видаляє частину файлу, пов'язаного з файловою змінною *f*, починаючи з поточної позиції, після чого поточна позиція стає кінцем файлу.

Procedure BlockWrite(Var f:File; Var Buf; N:Word)

- записує N записів зі змінної *Buf* в файл, пов'язаний з файловою змінною *f*.

Function FilePos(Var f):Longint

- повертає поточну позицію у файлі, пов'язаному з файловою змінною *f*.

Function FileSize(Var f):Longint

- повертає поточний розмір файлу, пов'язаного з файловою змінною *f*.

Для нетипізованих файлів основним параметром є довжина запису в байтах. Наприклад, якщо зазначений тип *Char* або *Byte*, великого значення це не має, оскільки важливим є лише те, який обсяг займають оголошені дані. Ця обставина дозволяє забезпечити максимально можливу швидкість доступу до даних, що містяться в цих файлах.

8.15 ПРАВИЛА ПУНКТУАЦІЇ

При оформленні програм слід пам'ятати, що:

1. Крапка з комою не ставиться на розділах описів після службових слів *Unit*, *Uses*, *Label*, *Type*, *Const*, *Var* і ставиться після завершення кожного розділу опису.
2. Крапка з комою не ставиться після *Begin* і перед *End*, так як ці службові слова є операторні дужки, а не операторами.
3. Крапка з комою є розмежувачами операторів, її відсутність між операторами викликає помилку компіляції.

4. У операторах циклу крапка з комою не ставиться після *While*, *Repeat*, *Do* і перед *Until*.
5. В умовних операторах крапка з комою не ставиться після *Then* і перед *Else*.

8.16 СТАНДАРТНІ МОДУЛІ

Всі стандартні засоби входять до складу мови і розташовуються в спеціалізованих бібліотечних модулях. Кожен модуль трансліюється окремо і може використовуватися в програмі користувача. Turbo Pascal 7.0 має 10 стандартних модулів, що містяться у файлі TURBO.TPL

System - забезпечує роботу всіх інших модулів системи;

Dos - включає засоби, що дозволяють реалізувати різні функції Dos;

CRT - містить засоби керування дисплеєм і клавіатурою комп'ютера;

OVERLAY - містить засоби організації оверлейних програм;

PRINTER - забезпечує швидкий доступ до друкуючого пристрою;

і в окремих файлах з розширенням TPU

GRAPH - містить пакет графічних засобів;

STRINGS - дає можливість використовувати в програмах, сумісних з Windows-додатками, рядки з завершальним нулем;

WINDOS - підтримує найбільш часто використовувані функції операційної системи;

TURBO3 - забезпечує сумісність з версією Turbo Pascal 3.0

GRAPH3 - підтримує використання стандартних графічних підпрограм Turbo Pascal 3.0;

Крім того Turbo Pascal 7.0 містить модулі бібліотеки об'єктно-орієнтованих програм Turbo Vision для розробки користувальницьких інтерфейсів. Модуль SYSTEM підключається за замовчуванням, всі інші під-

ключаються за допомогою службового слова *Uses*. Розглянемо коротко призначення деяких модулів.

8.16.1 Модуль SYSTEM

Модуль SYSTEM є фактично основною бібліотекою середовища Turbo Pascal. В нього входять всі зумовлені процедури і функції стандарту мови Turbo Pascal, а також додаткові підпрограми, що дозволяють виконувати різноманітні дії загального призначення (керування вводом-виводом, робота з рядками, статичної та динамічної пам'яттю і т.д.).

Модуль SYSTEM включає в себе наступні процедури і функції:

- арифметичні функції;
- процедури і функції для величин порядкового типу;
- процедури та функції роботи з рядками;
- функції перетворення типів;
- процедури і функції управління вводом-виводом;
- процедури і функції управління динамічною пам'яттю;
- адресні функції і управління програмою;
- функції управління програмою і т . ін.

8.16.2 Модуль CRT

Модуль CRT містить константи, змінні та підпрограми, призначені для роботи з консоллю. На відміну від стандартного вводу-виводу, коли він здійснюється через операційну систему, підпрограми модуля CRT працюють з BIOS, і навіть безпосередньо з пам'яттю.

8.16.2.1. Вікна

Для роботи з вікнами в Turbo Pascal використовується процедура модуля CRT - Window. Положення вікон на екрані і їх розмір визначається за допомогою координат (X1, Y1) лівого верхнього і (X2, Y2) правого нижнього кутів. Координата X визначає позицію в рядку. Нумерація починається з 1 і йде зліва направо. В якості координати Y - номер рядка. Нумерація рядків починається з 1 і здійснюється зверху вниз. У вікні координати відлічуються від ліво-

го верхнього кута вікна. Для отримання координат поточного вікна використовуються зумовлені змінні WindMin і WindMax типу Word, що надають інформацію про розміри поточного вікна.

8.16.2.2. Установка кольорів

При роботі з модулем CRT весь екран представляється сукупністю прямокутних елементів, для кожного з яких може бути заданий колір фону, колір і ефект мерехтіння символу на ньому. Вся ця інформація зберігається в одному байті пам'яті

7	6	5	4	3	2	1	0
М	Ф	Ф	Ф	С	С	С	С

Тут **М** - біт мерехтіння;

Ф - біти кольору фону;

С - біти кольору символу.

Таблиця 8.19. Коды кольорів

Black	0	Чорний	LightBlue	9	Світло-синій
Blue	1	Синій	LightGreen	10	Світло-зелений
Green	2	Зелений	LightCyan	11	Світло-голубий
Cyan	3	Голубий	LightRed	12	Рожевий
Red	4	Червоний	LightMagenta	13	Світло-фіолетовий
Magenta	5	Фіолетовий	Yellow	14	Жовтий
Brown	6	Коричневий	White	15	Білий
LightGray	7	Світло-сірий			
DarkGray	8	Темно-сірий	Blink	128	Мерехтіння символа

Кольори з кодами від 0 до 7 включно можна використовувати як для символів, так і для фону. Інші кольори і код мерехтіння можна використовувати тільки для символів.

Установка кольорів символів, що виводяться на екран, виконується процедурою *TextColor*, кольору фону - процедурою *TextBackGround* .. Установка яскравості символів здійснюється процедурами *LowVideo*, *HighVideo* і

NormVideo. Для управління кольорами в текстовому режимі можна використати зумовлену змінну *TextAttr* модуля CRT, яка має наступний вигляд:

*TextAttr:=16*ЦветФона+ЦветСимволов+МерцаниеСимволов*

Тут установка кольору і мерехтіння вказується або відповідною константою (*Black, Blue, Green, ...*) або кодом. Наприклад,

*TextAttr:=16*Cyan+Red+Blink*

або

*TextAttr:=16*3+4+128*

8.16.2.3. Процедури і функції

У модулі Crt реалізовані наступні процедури і функції:

Procedure ClrEol

- видалення всіх символів від курсору (включно) до кінця рядка.

Procedure ClrScr

- очищення поточного вікна. Курсор встановлюється у верхній лівий кут вікна.

Procedure Delay(Msec:Word)

- затримка виконання програми на *Msec* мілісекунд.

Procedure DelLine

- видалення рядка, в якому знаходиться курсор.

Procedure GoToXY(X,Y:Byte)

- переміщення курсору в позицію з координатами X, Y.

Procedure HighVideo

- установка підвищеної яскравості символів (кольори 0 - 7 замінюються на кольори 8 - 15).

Procedure InsLine

- вставка порожнього рядка в місці розташування курсору.

Function KeyPressed:Boolean

- повернення значення *True*, якщо на клавіатурі натиснута клавіша і *False* в іншому випадку. Введений символ залишається в буфері клавіатури і може бути зчитаний за допомогою функції *ReadKey*.

Procedure LowVideo

- установка зниженої яскравості символів (кольори 8 - 15 замінюються на кольори 0 - 7).

Procedure NormVideo

- установка первісної яскравості символів.

Procedure NoSound

- відключення звуку.

Function ReadKey:Char

- зчитування символу з клавіатури.

Procedure Sound(Herz:Word)

- генерування звукового сигналу з частотою Herz (герц).

Procedure TextBackGround(Color:Byte)

- установка кольору фону параметром *Color*.

Procedure TextColor(Color:Byte)

- установка кольору символів параметром *Color*.

Procedure TextMode (Mode:Word)

- установка текстового режиму параметром *Mode*.

Function WhereX:Byte

- повернення координати X положення курсору щодо поточного вікна.

Function WhereY:Byte

- повернення координати Y положення курсору щодо поточного вікна.

Procedure Window (X1,Y1,X2,Y2:Byte)

- визначення на екрані текстового вікна з координатами X1, Y1 лівого верхнього кута і координатами X2, Y2 - нижнього правого кута.

Контрольні питання та завдання

1. Наведіть коротку характеристику мови TURBO PASCAL.
2. Якими елементами описується мова TURBO PASCAL?
3. Які групи символів входять до алфавіту мови TURBO PASCAL?
4. Які символи не можна використовувати в програмі на мові TURBO PASCAL і чому?
5. Що таке ідентифікатор?
6. Наведіть правила, за якими створюється ім'я ідентифікатора.
7. Що таке зарезервовані слова? Наведіть декілька прикладів.
8. Які групи операторів існують в мові TURBO PASCAL?
9. Які типи даних можна застосовувати в програмі на мові TURBO PASCAL?
10. Наведіть загальну структуру програми на мові TURBO PASCAL.
11. Які елементи програми на мові TURBO PASCAL не є обов'язковими?

12. Наведіть мінімальну програму, коректну з точки зору TURBO PASCAL.
13. Чим відрізняється змінна від константи?
14. Як описуються константи і змінні?
15. Наведіть декілька прикладів опису змінних цілого, рядкового, логічного типу.
16. Які оператори слугують для введення і виведення інформації в програмі на мові TURBO PASCAL?
17. Чим відрізняється оператор Write() і WriteLn()?
18. Як вивести дійсне число з 3 знаками після коми?
19. Як записати за допомогою стандартних функцій мови TURBO PASCAL такий вираз: $y = \sin\{2x\} + 1 \cdot \sqrt{\ln\{x^2\} + \cos\{x\}}$?
20. Які операції відношення ви знаєте?
21. Які логічні функції існують в мові TURBO PASCAL?
22. Який результат поверне наступний вираз: (True or not True) and (False xor (True or False))?
23. Що таке операція привласнення?
24. Як використовуються в програмі операторні дужки?
25. Як в мові TURBO PASCAL виконується розгалуження?
26. Які дві форми умовного оператора існують в мові TURBO PASCAL?
27. Чим відрізняється умовний оператор IF від оператору вибору CASE?
28. Що виконує оператор FOR?
29. Наведіть дві форми оператору FOR.

30. Чим відрізняються оператор WHILE..DO і оператор REPEAT..UNTIL?
31. Скільки разів як мінімум може виконатися оператор REPEAT..UNTIL? А оператор WHILE..DO?
32. Скільки разів виведеться слово 'Hello' в даному випадку: for i:=1 to 4 do for j:=4 downto i do Write('Hello');?
33. Чим відрізняються параметри-значення від параметрів-змінних?
34. Якщо в процедуру треба передати великий масив даних краще використовувати параметри-змінні чи параметри-константи?
35. Чим відрізняються процедури від функцій?
36. Що таке масив?
37. Які значення може приймати індекс масиву?
38. Як оголошується масив?
39. Чи можна в масив записувати різні типи даних?
40. Скільки елементів в оголошеному масиві: A:array[1..10] of array[1..2,1..3] of array[-1..1] of byte;
41. Як привласнити комірці попереднього масиву A певне значення?
42. Скільки займатиме місця в пам'яті наступний масив: A:array[-127..128,1..2] of array[1..2] of array[1..10] of word;?
43. Чи можна поєднувати в одній змінній мови TURBO PASCAL різні типи даних? Який тип буде мати така змінна?
44. Які типи файлів існують в мові TURBO PASCAL? Для чого використовується кожний із них?
45. Які дії треба виконати, щоб скопіювати дані з одного текстового файлу в інший?
46. Що таке модуль?
47. Які модулі ви знаєте? Наведіть їх коротку характеристику.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Информатика. Конспект лекцій. Частина 1,2. Для студентів всіх спеціальностей заочної форми навчання /Укл. М. М. Пруденко/. — Запоріжжя, вид-во ЗДІА, 2006 — 100с.
2. Программирование в среде Turbo Pascal 7.0/ Марченко А.И., Марченко Л.А.: Под ред. Тарасенко В.П. — 5-е изд., доп. И перераб. - К.: ВЕК+, 1999. — 464 с.
3. Розробка алгоритмів та програмування мовою Turbo Pascal. Навчальний посібник для технічних ВНЗ. / В. Я. Сердюченко/. — ВКУ "Парітет" ЛТД, — 1995. — 352 с.
4. Архитектура компьютера./Э. Таненбаум/ — 5-е изд. — СПб.: Питер, 2007. — 844 с.
5. Информатика. Базовый курс. 2-е издание / Под ред. С. В. Симоновича /. — СПб.: Питер, 2005. — 640 с: ил.
6. Операционные системы. Разработка и реализация. /Э. Таненбаум, А. Вудхалл/. — 3-е изд. — СПб.: Питер, 2007. — 704 с: ил.
7. Практикум по информатике: Учеб. пособие для студ. высш. учеб. заведений. / А.В.Могилев, Н.И.Пак, Е.К.Хеннер; Под ред. Е.К.Хеннера/. — 2-е изд., стер. — М.: Издательский центр «Академия», 2005. — 608 с.
8. История вычислительной техники. [http://ru.wikipedia.org/wiki/История вычислительной техники](http://ru.wikipedia.org/wiki/История_вычислительной_техники)
9. Интернет. <http://ru.wikipedia.org/wiki/Интернет>
10. Компьютерная память [http://ru.wikipedia.org/wiki/ Компьютерная память](http://ru.wikipedia.org/wiki/Компьютерная_память)
11. Сложная разметка в OpenOffice.org Math. / Д.В.Смирнов/. <http://myooo.ru/content/view/70/54/>
12. Список операционных систем. [http://ru.wikipedia.org/wiki/Список операционных систем](http://ru.wikipedia.org/wiki/Список_операционных_систем)
13. Список файловых систем. [http://ru.wikipedia.org/wiki/Список файловых систем](http://ru.wikipedia.org/wiki/Список_файловых_систем)
14. OpenOffice.org 3 User Guides. http://wiki.services.openoffice.org/wiki/Documentation/OOo3_User_Guides/Chapters

ЗМІСТ

ПЕРЕДМОВА	3
1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ІНФОРМАТИКУ	4
1.1 СИТЕМИ ЧИСЛЕННЯ	4
1.1.1 Представлення чисел в різних системах числення	5
1.1.2 Переклад чисел з однієї позиційної системи числення в іншу	6
1.2 ІНФОРМАЦІЯ	10
1.2.1 Поняття інформації	10
1.2.2 Властивості інформації	11
1.2.3 Кількість інформації	12
1.2.4 Кодування інформації	14
1.3 ІНФОРМАЦІЙНІ СИТЕМИ	18
1.4 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ	20
2 АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ	22
2.1 ВИЗНАЧЕННЯ ЕОМ	22
2.2 КЛАСИФІКАЦІЯ КОМП'ЮТЕРІВ	23
2.2.1 Класифікація за призначенням	23
2.2.2 Класифікація за характером розв'язуваних задач	23
2.2.3 Класифікація за габаритами	23
2.3 АРХІТЕКТУРА КОМП'ЮТЕРА	25
2.4 КОНСТРУКТИВНЕ ВИКОНАННЯ СУЧАСНОЇ ЕОМ	27
2.4.1 Системний блок	28
2.4.2 Периферія	23
2.4.3 Засоби комунікації	39

2.4.4 Внутрішні (локальні) інтерфейси	40
2.4.5 Мережне обладнання	42
3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ	46
3.1 ПОНЯТТЯ ПРОГРАМИ	46
3.2 КЛАСИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	46

3.3 ОПЕРАЦІЙНІ СИСТЕМИ	49
3.3.1 Поняття ОС, основні функції	50
3.3.2 Класифікація ОС	50
3.3.3 Сучасні ОС	51
3.4 ТЕХНОЛОГІЇ ЗБЕРЕГАННЯ ДАНИХ	54
3.5 СУЧАСНІ ФАЙЛОВІ СИСТЕМИ	59
4 ОСНОВИ МЕРЕЖНИХ ТЕХНОЛОГІЙ	62
4.1 ВИЗНАЧЕННЯ КОМП'ЮТЕРНОЇ МЕРЕЖІ	62
4.1.1 Класифікація комп'ютерних мереж	62
4.1.2 Основні терміни сучасних мережних технологій	63
4.1.3 Локальні мережі	64
4.1.4 Глобальна мережа Internet	65
4.2 ОСНОВНІ МЕРЕЖНІ СЕРВІСИ І ПРОТОКОЛИ	66
4.3 БЕЗПЕКА ПРИ РОБОТІ В КОМП'ЮТЕРНИХ МЕРЕЖАХ	68
5 ОФІСНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ	70
5.1 ПРИЗНАЧЕННЯ І ФУНКЦІЇ ОФІСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	70
5.2 ОСНОВНІ КОМПОНЕНТИ	71
5.3 ОПИС ПОШИРЕНИХ ОФІСНИХ ПАКЕТІВ	74
5.4 ТЕКСТОВИЙ ПРОЦЕСОР OPENOFFICE.ORG WRITER	76
5.4.1 Інтерфейс. Основні функціональні можливості	76
5.4.2 Стили оформлення тексту	77
5.4.3 Введення тексту	78
5.4.4 Форматування документа	78
5.4.5 Структура документа	78
5.4.6 Таблиці	79
5.4.7 Зображення	79
5.4.8 Розширені операції з документом	80
5.5 ФОРМУЛИ	83
5.5.1 Програми для створення формул	83
5.5.2 Редактор формул OO Math	83

5.6 ОБРОБКА ГРАФІЧНОЇ ІНФОРМАЦІЇ	93
5.6.1 Види графічного програмного забезпечення	93
5.6.2 Растрові редактори	94
5.6.3 Векторні редактори	96
5.7 БАЗИ ДАНИХ	98
5.7.1 Визначення бази даних	98
5.7.2 Основні елементи СКБД	100
5.7.3 Поширені СКБД	100
5.8 ПРЕЗЕНТАЦІЇ	101
5.8.1 Поняття про презентацію	101
5.8.2 Презентаційне програмне забезпечення	102
5.8.3 Створення нової презентації	102
5.8.4 Форматування презентацій	103
5.8.5 Робота зі слайдами	104
5.8.6 Режими робочого простору	105
5.8.7 Показ слайдів	105
5.9 ЕЛЕКТРОННІ ТАБЛИЦІ	106
5.9.1 Призначення електронних таблиць	106
5.9.2 Інтерфейс табличного процесора	106
6 РОЗВ'ЯЗАННЯ ПРИКЛАДНИХ ЗАДАЧ З ВИКОРИСТАННЯМ ЕОМ	114
6.1 ТЕХНОЛОГІЯ ПІДГОТОВКИ І РОЗВ'ЯЗАННЯ ЗАДАЧ НА ЕОМ	114
6.1.1 Постановка задачі	114
6.1.2 Формалізація задачі	115
6.1.3 Вибір методу розв'язання задачі	115
6.1.4 Розробка алгоритму	116
6.1.5 Програмування для ЕОМ	116
6.1.6 Тестування і налагоджування програми	116
6.1.7 Обчислення, обробка, й аналіз результатів	117
7. ОСНОВИ АЛГОРИТМІЗАЦІЇ	118

7.1 ВИЗНАЧЕННЯ АЛГОРИТМУ	118
7.2 ВЛАСТИВОСТІ АЛГОРИТМУ	122
7.3 СТРУКТУРА АЛГОРИТМУ	123
7.4 ЛІНІЙНА СТРУКТУРА	124
7.5 СТРУКТУРА РОЗГАЛУДЖЕННЯ	125
7.6 ЦИКЛІЧНА СТРУКТУРА	125
7.7 ТИПОВІ ПРИЙОМИ АЛГОРИТМІЗАЦІЇ	126
7.7.1 Цикл з цілим кроком	126
7.7.2 Цикл з довільним кроком	128
7.7.3 Цикл зі збереженням результатів	130
7.7.4 Ітераційний цикл	132
7.7.5 Вкладені цикли	133
7.7.6 Підрахунок кількості	135
7.7.7 Обчислення добутку	136
7.7.8 Обчислення факторіалу	137
7.7.9 Обчислення суми	140
7.7.10 Знаходження найменшого й найбільшого значень функції	142
7.7.11 Обчислення значення полінома	143
7.7.12 Обробка масивів	146
8. МОВА ПРОГРАМУВАННЯ TURBO PASCAL 7.0	151
8.1 ОСНОВНІ ЕЛЕМЕНТИ МОВИ	151
8.1.1 Алфавіт мови	151
8.1.2 Синтаксис	154
8.1.3 Семантика	154
8.1.4 Лексеми	154
8.1.5 Розподільники	158
8.1.6 Дані	159
8.1.7 Типи даних	159
8.2 СТРУКТУРА ПРОГРАМИ	160
8.3 ОГолошення і описи	162

8.3.1	Опис міток	162
8.3.2	Опис типів	162
8.3.3	Опис констант	162
8.3.4	Опис змінних	163
8.3.5	Опис процедур та функцій	163
8.4	СТАНДАРТНІ ТИПИ	164
8.4.1	Цілі типи	164
8.4.2	Дійсні типи	165
8.4.3	Булеві типи	165
8.4.4	Символьний тип	166
8.4.5	Рядкові типи	167
8.4.6	Вказівний тип	168
8.4.7	Текстовий тип	168
8.5	ВВЕДЕННЯ-ВИВЕДЕННЯ	168
8.5.1	Пристрій CON	169
8.5.2	Паралельні адаптери LPT	169
8.5.3	Процедура читання Read	169
8.5.4	Процедура читання Readln	169
8.5.5	Процедура читання Write	169
8.5.6	Процедура читання Writeln	170
8.5.7	Форматоване введення	170
8.6	СТАНДАРТНІ ПРОЦЕДУРИ І ФУНКЦІЇ	171
8.7	ВИРАЗИ	173
8.7.1	Арифметичні операції	174
8.7.2	Операції відношення	178
8.7.3	Булеві (логічні) операції	179
8.7.4	Рядкова операція	180
8.7.5	Операції над множинами	181
8.7.6	Операція @ (створення покажчика)	182
8.7.7	Пріоритет операцій	182
8.8	ОПЕРАТОРИ	183

8.8.1 Прості оператори	184
8.8.2 Структурні оператори	185
8.9 СТАНДАРТНІ ПРОЦЕДУРИ ПЕРЕХОДІВ	193
8.9.1 Процедура виходу із циклу BREAK	194
8.9.2 Процедура продовження циклу CONTINUE	194
8.9.3 Процедура виходу із блоку EXIT	194
8.10 ПРОЦЕДУРИ І ФУНКЦІЇ	194
8.10.1 Процедури	195
8.10.2 Функції	195
8.10.3 Область дії ідентифікаторів	196
8.10.4 Параметри	197
8.11 КОРИСТУВАЦЬКІ ТИПИ	201
8.11.1 Порядкові типи	201
8.11.2 Структуровані типи	202
8.12 ОПИС ТИПІЗОВАНОЇ КОНСТАНТИ	214
8.12.1 Константи стандартних типів	214
8.12.2 Константи-масиви	215
8.12.3 Константи-множини	216
8.12.4 Константи-записи	216
8.13 РОБОТА З РЯДКАМИ	217
8.13.1 Операції з рядками	218
8.13.2 Засоби обробки рядків	218
8.14 РОБОТА З ФАЙЛАМИ	223
8.14.1 Текстові файли	224
8.14.2 Типізовані файли	227
8.14.3 Файли без типу	229
8.15 ПРАВИЛА ПУНКТУАЦІЇ	229
8.16 СТАНДАРТНІ МОДУЛІ	230
8.16.1 Модуль SYSTEM	231
8.16.2 Модуль CRT	231