МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
“НАЦІОНАЛЬНИЙ ГІРНИЧИЙ УНІВЕРСИТЕТ”

Л.І. Цвіркун,
А.А. Євстігнєєва,
Я.В. Панферова

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ПРОГРАМУВАННЯ

Навчальний посібник

Видання третє, виправлене

Під загальною редакцією професора Л.І. Цвіркуна

Дніпропетровськ
НГУ
2016

УДК 004.415.2=93С++(075.8)
ББК 32.973-018я73
Ц 28

Рекомендовано вченою радою як навчальний посібник для бакалаврів спеціальності 123 Комп'ютерна інженерія (протокол № 3 від 11 березня 2016).

Рецензенти:

І.В. Жуковицький, доктор технічних наук, професор, завідувач кафедри обчислювальних машин (Дніпропетровський національний університет залізничного транспорту ім. В. Лазаряна);

О.Д. Шамровський, доктор технічних наук, професор, завідувач кафедри програмного забезпечення (Запорізька державна інженерна академія).

Цвіркун Л.І.

Ц 28 Розробка програмного забезпечення комп'ютерних систем. Програмування: навч. посіб. / Л.І. Цвіркун, А.А. Євстігнєєва, Я.В. Панферова; під заг. ред. Л.І. Цвіркуна; М-во освіти і науки України, Нац. гірн. ун-т. – 3-тє вид., випр. – Дніпропетровськ: НГУ, 2016. – 223 с.

ISBN 978–966–350–595–4

Наведено основні поняття мови програмування С++, а також методи розробки програм. Подано лабораторний практикум з основ програмування мовою С++. Сформульовано вимоги до складових курсового проекту з програмування та наведено рекомендації щодо його оформлення, подані завдання до навчальної практики.

Для студентів вищих навчальних закладів спеціальності "Комп'ютерна інженерія".

УДК 004.415.2=93С++(075.8)
ББК 32.973-018я73

ISBN 978–966–350–595–4

ЗМІСТ

Присвячується 15-річчю
початку підготовки
фахівців за спеціальністю
„Комп'ютерна інженерія"

ПЕРЕДМОВА

Навчальна дисципліна „Програмування" належить до професійно-практичного циклу і посідає важливе місце у підготовці студентів в галузі знань 12 Інформаційні технології спеціальності 123 Комп'ютерна інженерія.

Зміст посібника відповідає навчальним планам дисципліни „Програмування", яка більше двадцяти років викладається авторами у Державному ВНЗ "Національний гірничий університет" (Державний ВНЗ "НГУ", м. Дніпропетровськ, Україна).

У даному виданні разом з теорією програмування зібрані лабораторний практикум та методичні поради до курсового проекту і першої навчальної практики, а також наведено багато прикладів з реальними доробками програм.

Як мова програмування у даній роботі застосовується C++, а середовищем програмування є Microsoft Visual Studio 2010.

Даний навчальний посібник стане у пригоді студентам під час виконання курсового проекту з дисципліни „Програмування" та завдань першої навчальної практики. Також його можна використовувати для самостійного навчання.

Посібник, написаний викладачами НГУ Л.І. Цвіркуном, кандидатом технічних наук, професором кафедри автоматизації та комп'ютерних систем (АКС) та асистентами цієї кафедри А.А. Євстігнєєвою та Я.В. Панферовою, складається з 5 частин і 21 розділу. Розділи 1, 2, 9, 12 і 16–20 та загальне редагування виконано Л.І. Цвіркуном, 3–5, 10, 11, 13–15 і 18–21 написані А.А. Євстігнєєвою, 6– 8, 18, 19 і 21 – Я.В. Панферовою.

Критичні зауваження, пропозиції та побажання будуть із вдячністю прийняті за адресою: кафедра автоматизації та комп'ютерних систем, просп. К. Маркса, 19, Державний ВНЗ "Національний гірничий університет", м. Дніпропетровськ, Україна, 49005, або за e-mail: TsvirkunL@gmail.com.

ВСТУП

Сучасне програмування виникло в 40-х роках минулого століття, коли з'явилися перші обчислювальні машини. Проте поняття алгоритму або опис порядку виконання дій для досягнення певної мети, пов'язане з програмуванням, відоме давно.

Так, ще в стародавні часи були одержані правила (тобто фактично алгоритми) знаходження площ і об'ємів геометричних фігур. Математики Стародавнього Сходу винайшли десяткову систему і склали правила обчислень у цій системі.

Сам термін „алгоритм" теж має стародавнє походження, будучи латинізованою транскрипцією імені великого середньоазіатського вченого IX століття Мухаммеда аль-Хорезми (де Хорезм – назва стародавньої держави на території Узбекистану). У цього математичному трактаті формулювалися також і правила різноманітних обчислень.

Правила запису алгоритму для виконання його обчислювальною машиною повинні бути строгими, тобто розумітися однозначно. Адже ЕОМ не повинна нічого додумувати за людину.

Сукупність засобів і правил подання алгоритму у вигляді, придатному для сприйняття ЕОМ, називається мовою програмування. Кожен алгоритм, записаний на деякій мові програмування, називається програмою.

У 1978 році з'явилося повідомлення про розробку мови програмування С. До моменту її появи вже широко використовувалися різні версії мов фортран, алгол-60, мови для обробки комерційної інформації кобол, ЛІСП, універсальні мови ПЛ/1, симула-67 і алгол-68, мови структурного програмування паскаль, BCPL та ін.

Однією з найбільш поширених мов третього тисячоліття є С++, яка дає подальший розвиток мови програмування С.

Автором мови програмування С++ є датський вчений, нині професор Техаського університету Бьєрн Страуструп.

Ранні версії мови С++, широко відомі під назвою «С із класами», з'явилися у 1980 році. Її було розроблено для написання високоефективних програм і вона замінила мову симула-67 для таких застосувань.

«С із класами» була успішно перевірена на проектах, які вимагали мінімальний об'єм пам'яті і мінімум часу виконання. У ній не було перевантаження операторів, посилань, віртуальних функцій, шаблонів, виключень і т.д.

С++ почала використовуватися програмістами з літа 1983 року. Назва С++ вимовляється «сі плюс плюс» або на англійський зразок «сі плас плас».

Спочатку С++ створювалась для того, щоб звільнити автора і його друзів від програмування на асемблері. Пізніше інтенсивний розвиток С++ викликало деякі зміни і в 1987 році почалися роботи із створення стандарту С++.

Первісний проект стандарту був випущений для публічного обговорення в квітні 1985-го. Міжнародний стандарт ISO (ISO/IEC 14882) по С++ був прийнятий у 1998 році.

C++ використовується сотнями тисяч програмістів практично у всіх прикладних застосуваннях. Мова підтримується десятком незалежних реалізацій, сотнями бібліотек, керівництв, декількома технічними журналами, численними конференціями і великою кількістю консультантів.

Ефективність C++ дозволяє її застосовувати для написання драйверів та інших програм, призначених для безпосереднього керування апаратурою в режимі реального часу.

Найсильніша сторона C++ – це можливість ефективного використання в програмах, призначених для широкого кола прикладних застосувань. Крім того, C++ здатна працювати з фрагментами кодів, написаними на інших мовах.

Таким чином, мова програмування C++ достатньо:

– зрозуміла для навчання основним концепціям;

– змістовна і може бути інструментом навчання складнішим концепціям і методам;

– реалістична, ефективна і гнучка для критичних проектів;

– доступна для різних середовищ розробки і виконання;

– універсальна для використання придбаних знань при вивченні інших мов і середовищ.

Отже C++ – мова програмування, з якою ви можете рости.

При спрощеному розгляді (без урахування етапів моделювання і тестування) програма розробляється в три етапи: аналіз, проектування та програмування.

Спочатку знайомляться з умовами поставленого завдання (аналіз), потім визначаються з напрямами вирішення і розробляють алгоритм, власне, його розв’язання у вигляді схеми програми (проектування), далі втілюють алгоритм у програмі (програмування).

У цьому навчальному посібнику фактично теоретичний матеріал і практичні завдання підібрані так, що в більшій мірі приділяється увага (як і виходить з назви книги і навчальної дисципліни, що вивчається) програмуванню.

Перший і другий етапи, а також моделювання й тестування програмного забезпечення детальніше розглядаються в інших дисциплінах і відповідно до інших навчальних посібників і книг.

ЧАСТИНА 1. ОСНОВИ МОВИ ПРОГРАМУВАННЯ С++

1. ОСНОВНІ ПОНЯТТЯ ПРО ФОРМИ ПОДАННЯ ІНФОРМАЦІЇ

Навчальною метою розділу є ознайомлення студентів з формами подання інформації.

У результаті вивчення даного розділу студенти повинні знати:

– визначення системи числення;
  ознаки, за якими класифікуються системи числення;
  визначення основи позиційної системи числення;
  вміти:
  навести приклади позиційних систем числення;
  виконати переведення чисел з однієї системи числення в іншу.

1.1. Системи числення

Будь-яке число можна подати набором спеціальних знаків (цифр). Спосіб подання чисел характеризує вид системи числення.

Системою числення (СЧс) називається сукупність прийомів і правил написання і читання чисел за допомогою спеціальних знаків (символів, цифр).

Існують позиційні і непозиційні системи числення.

У непозиційних системах числення вага спеціального знака (цифри) не залежить від позиції, яку він займає в числі. Стародавній варіант римської системи числення може служити прикладом непозиційної системи. Як спеціальні знаки в римській системі числення застосовувались букви, які позначали: *I* – 1, *V* – 5, *X* – 10, *L* – 50, *C* – 100, *D* – 500, *M* – 1000. Величина числа визначалась як сума цифр, що відповідають спеціальним знакам у числі.

Наприклад:

MCCCXXIII=1323.

1.2. Позиційні СЧс

Найбільшого поширення набули позиційні системи числення, в яких вага кожного спеціального знака (цифри) визначається як його величиною, так і його положенням (позицією) серед інших символів.

Позиційна система числення (ПСЧс) – це така система, в якій один і той же спеціальний знак (цифра) має різну вагу залежно від його позиції (місця розташування) в числі.

У ПСЧс кількість різних спеціальних знаків (цифр), використовуваних для зображення чисел, називається *основою* СЧс. Основою системи числення може бути будь-яке число більше за одиницю.

У найбільш поширених ПСЧс вага кожного розряду (ключове число) зростає у геометричній прогресії, тобто його вага в основу раз більша за вагу сусіднього справа розряду. До таких систем належить сучасна десяткова СЧс з основою, що дорівнює 10. У десятковій СЧс послідовність ключових чисел, або базис має вигляд:

$$q_0 = 10^0, q_1 = 10^1, q_2 = 10^2, ..., q_n = 10^n \ .$$

А числа в ПСЧс у цілому набувають вигляду:

$$N = a_n \cdot q_n + a_{n-1} \cdot q_{n-1} + ... + a_2 \cdot q_2 + a_1 \cdot q_1 + a_0 \cdot q_0 = a_n a_{n-1} ... a_2 a_1 a_0 ,$$

тобто, наприклад, число 4537 запишемо у десятковій ПСЧс:

$$4537_{10} = 4 \cdot 10^3 + 5 \cdot 10^2 + 3 \cdot 10^1 + 7 \cdot 10^0 \ .$$

Таким чином, застосовуючи позиційний принцип, ми маємо можливість зобразити будь-яке дійсне число в десятковій системі числення за допомогою всього десяти цифр в їх різних комбінаціях.

За основу ПСЧс можна прийняти будь-яке натуральне число: два, три, чотири, ... , шістнадцять і т.д. Отже, можлива нескінченна безліч позиційних систем числення. Прикладами позиційних систем числення, що використовуються найчастіше, є: двійкова, десяткова, вісімкова, шістнадцяткова СЧс.

При одночасній роботі з декількома системами числення для їх розрізнення основа системи вказується у вигляді нижнього індексу, який записується в десятковій системі:

2784_{10}, – це число 2784 в десятковій системі числення;

1101011_2, – це число 1101011 в двійковій системі числення.

У деяких спеціальних галузях застосовуються особливі правила вказівки основи. Наприклад, у програмуванні шістнадцяткова система позначається так:

– у асемблері і записах загального вигляду, не прив’язаних до конкретної мови, буквою *h* (від hexadecimal) в кінці числа;

– у паскалі знаком „*$*” на початку числа;

– у С++ та інших мовах програмування комбінацією нуля і маленькою або великою літерою х (*0x* або *0X*) на початку числа.

У **двійковій системі числення,** де основою системи є число 2, використовується всього дві цифри – 0 і 1. Тож рахунок ведеться у такій послідовності:

0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111 і т. д.

Двійкові числа складаються, віднімаються, множаться і діляться за тими ж правилами, що і в десятковій системі числення (табл. 1.1).

Таблиця 1.1

Правила виконання дій додавання та віднімання з двійковими числами

Значення *i*-го розряду 1-го числа	Значення *i*-го розряду 2-го числа	Результат додавання		Результат віднімання		Значення *i*-го розряду добутку
		Значення *i*-го розряду	Перенесення у старший розряд	Позичан-ня у старшому розряді	Значен-ня *i*-го розряду	
0	0	0	0	0	0	0
0	1	1	0	1	1	0
1	0	1	0	0	1	0
1	1	0	1	0	0	1

Наприклад:

```
 1011,101        1011,110        1101,10        -100011 | 111
+1001,001        - 1001,001      * 101,01          111   | 101
 10100,110         10,101          110110          00111
                                  110110            111
                                110110                0
                               1000110,1110
```

У вісімковій системі числення, де основою системи є число 8**,** використовується вісім цифр: 0, 1, 2, 3, 4, 5, 6, 7. Тому в цій системі числення рахунок ведеться в такій послідовності:

0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, ... 77, 100, 101, … .

У шістнадцятковій системі числення, де основою системи є число 16, використовується десять арабських цифр від 0 до 9 і шість букв латинського алфавіту: A, B, C, D, E, F. Тож в шістнадцятковій системі числення рахунок ведеться в такій послідовності:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 21, ... 99, 9A, 9B, 9C, 9D, 9E, 9F, A1, A2, ..., AF, B1,..., FF, 100, 101, … .

1.3. Переведення чисел з однієї СЧс в іншу

Переведення дійсного числа з десяткової системи числення в двійкову або в будь-яку іншу здійснюється таким чином: спочатку виконується переведення цілої частини числа, потім дробової.

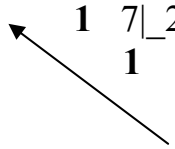
Для переведення цілої частини числа використовується такий алгоритм:

– ціла частина і цілі частки, що утворюються, послідовно діляться на основу нової СЧс, до тих пір, доки частка від ділення є більшою ніж основа нової системи числення. В результаті ділення остачі є цифрами цілої частини числа в новій СЧс. Остання частка від ділення є цифрою найстаршого розряду переведеного числа.

Наприклад. Необхідно перевести число з десяткової системи числення в двійкову (10 → 2). Десяткове число A_{10}=30,6. A_2=?

Переводимо спочатку цілу частину числа:

30|_2_
0 15|_2_
1 7|_2_
1 3|_2_
1 **1**



Результат: A_2=11110.

Для переведення дробової частини числа з десяткової системи числення в двійкову або в будь-яку іншу використовується такий алгоритм:

– дробова частина та дробові частини множення, що виходять, послідовно множаться на основу нової системи числення. Цілі частини, що виходять при кожному множенні, не беруть участі в подальших множеннях. Такі частини являють собою цифри дробової частини числа в новій СЧс. Значення першої цілої частини є першою цифрою після коми наведеного числа. Множення необхідно виконувати до набуття необхідної точності (кількості знаків після коми).

Наприклад. Необхідно перевести дробову частину числа з десяткової системи числення в двійкову. Десяткове число A_{10}=30,6, A_2=?

Переводимо окремо дробову частину числа:

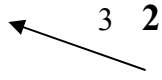
0,6·2=**1**,2
0,2·2=**0**,4
0,4·2=**0**,8
0,8·2=**1**,6
0,6·2=**1**,2

Результат: A_2=11110,10011.

Наприклад. Переведемо $B_{10}=153,7_{10}$ у вісімкову СЧс (10 → 8). B_8=?

Переведемо окремо цілу та дробову частини:

153|_8_
1 19|_8_
3 **2**



0,7·8=**5**,6
0,6·8=**4**,8
0,8·8=**6**,4

Результат: B_8=231,546.

Для переведення числа N з двійкової або будь-якої іншої системи числення в десяткову необхідно зобразити число у такому вигляді:

$$N_{10} = \sum_{i=0}^{n-1} a_i \cdot q^i ,$$

де a − розряд числа, q − основа системи числення даного числа, n − кількість розрядів у числі.

Нумерація розрядів у числі починається справа наліво.

Наприклад. Необхідно перевести число A_2=10011 у десяткову СЧс.

$$A_{10} = \sum_{i=0}^{4} a_i q^i = 1\cdot 2^0 + 1\cdot 2^1 + 0\cdot 2^2 + 0\cdot 2^3 + 1\cdot 2^4 = 19_{10},$$

де a_i − розряди числа, тобто a_0=1, a_1=1, a_2=0, a_3=0, a_4=1; q=2 − основа СЧс числа.

Якщо необхідно перевести дійсне число, то треба використовувати таку формулу:

$$N_{10} = \sum_{i=0}^{n-1} a_i q^i + \sum_{i=-k}^{-1} b_i q^i,$$

де a − розряд цілої частини числа, b − розряд дробової частини числа, q − основа системи числення числа, n − кількість розрядів цілої частини числа, k − кількість розрядів дробової частини числа.

У двійковому числі крайня права цифра показує кількість одиниць, наступна цифра – кількість двійок, наступна – кількість четвірок і т. д.

Наприклад:

$$110{,}101_2 = 1\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 + 1\cdot 2^{-1} + 0\cdot 2^{-2} + 1\cdot 2^{-3} = 4 + 2 + 0.5 + 0.125 = 6.625_{10}.$$

У вісімковому числі цифри, залежно від розряду, мають вагу: 1, 8, 64, 512, 4096 і т. д.

Наприклад:

$$2735_8 = 2\cdot 8^3 + 7\cdot 8^2 + 3\cdot 8^1 + 5\cdot 8^0 = 2\cdot 512 + 7\cdot 64 + 3\cdot 8 + 5 = 1501_{10}.$$

У шістнадцятковому числі цифри (або знаки), залежно від розряду, мають вагу: 1, 16, 256, 4096, 65536 і т. д.

Наприклад:

$$3BF7_{16} = 3\cdot 16^3 + 11\cdot 16^2 + 15\cdot 16^1 + 7\cdot 16^0 = 3\cdot 4096 + 11\cdot 256 + 15\cdot 16 + 7 = 15351_{10}.$$

При переведенні з десяткової системи в двійкову і навпаки можна використовувати метод переведення з використанням проміжної системи числення, тобто систему з основою 2^k.

Як проміжні СЧс широко використовуються вісімкова ($8=2^3$) і шістнадцяткова ($16=2^4$) системи.

При переведенні з десяткової в двійкову систему спочатку можна здійснювати переведення в проміжну СЧс за допомогою описаних раніше методів, а потім замість цифр (букв) підставляють двійкові еквіваленти.

Для переведення з двійкової системи в десяткову спочатку розбивають двійковий код на групи по *K* розрядів і кожну групу замінюють відповідною цифрою (буквою), а потім переходять від проміжної до десяткової системи числення за допомогою будь-якого з методів.

Наприклад, оскільки основа вісімкової СЧс $N=8=2^3$, то для переведення 8 → 2 достатньо кожен вісімковий розряд подати трьома двійковими (тріадами).

Якщо число дійсне, то розбиття на тріади виконується окремо вправо та вліво від коми.

Наприклад:

$$456{,}31_8 \rightarrow X_2$$

$$\overbrace{100}^{4}\ \overbrace{101}^{5}\ \overbrace{110}^{6},\ \overbrace{011}^{3}\ \overbrace{001}^{1}$$

Результат: $X_2 = 100101110{,}011001$.

Для переведення числа $1011110{,}1011_2$ → 8 необхідно вручну розбити його на тріади від коми (вліво і вправо), а потім кожну тріаду замінити вісімковою цифрою. Якщо крайня ліва або права тріади будуть неповними, то у двійкове число потрібно відповідно зліва або справа додати 0.

Наприклад:

$$1011110{,}1011_2 \rightarrow X_8$$

$$\underbrace{001}_{1}\ \underbrace{011}_{3}\ \underbrace{110}_{6},\ \underbrace{101}_{5}\ \underbrace{100}_{4}$$

Результат: $X_8 = 136{,}54$.

Оскільки основа шістнадцяткової системи $N=16=2^4$, то перехід від шістнадцяткового запису чисел до двійкового здійснюється описаним далі способом. Кожна шістнадцяткова цифра виражається чотирма двійковими – тетрадами.

Наприклад:

$$456{,}31_{16} \rightarrow X_2$$

$$\overbrace{0100}^{4}\ \overbrace{0101}^{5}\ \overbrace{0110}^{6},\ \overbrace{0011}^{3}\ \overbrace{0001}^{1}$$

Результат: $X_2 = 10001010110{,}0011001$.

Для переведення числа $1011110{,}1011_2$ → 16 необхідно вручну розбити його на тетради від коми (вліво і вправо), а потім кожну тетраду замінити шістнадцятковою цифрою. Якщо крайня права або ліва тетради будуть неповними, то до двійкового числа потрібно відповідно справа або зліва додати цифру 0 або декілька 0.

Наприклад:

$$1011110{,}1011_2 \rightarrow X_{16}$$

$$\underbrace{0101}_{5}\ \underbrace{1110}_{\mathrm{E}},\ \underbrace{1011}_{\mathrm{B}}$$

Результат: $X_{16} = 5\mathrm{E{,}B}$.

1.4. Обробка інформації в комп'ютері

В електронних обчислювальних машинах застосовують ПСЧс з недесятковою основою: двійкові, вісімкові та шістнадцяткові. Найбільше розповсюдження в обчислювальних машинах має двійкова система числення.

Це обумовлено:

– простішою реалізацією алгоритмів виконання арифметичних і логічних операцій;

– надійнішою фізичною реалізацією основних функцій, оскільки вони мають усього два стани (0 і 1);

– економічністю апаратурної реалізації всіх схем ЕОМ.

Двійкове зображення числа вимагає більшої (для багаторозрядного числа приблизно в 3,3 рази) кількості розрядів, ніж його десяткове уявлення. Проте застосування двійкової системи допомагає у проектуванні цифрових обчислювальних машин з простих елементів, оскільки для втілення в машині одного розряду двійкового числа може бути використаний будь-який елемент, що має всього два стійкі стани. Такими елементами, наприклад, є реле, тригерні схеми і т. ін.

Один такий елемент в ЕОМ може бути застосований для запам'ятовування одиниці інформації в 1 біт.

1 байт = 8 бітам, 1 Кб (кілобайт) = 1024 байтам.

1 Мб (мегабайт) = 1024 кілобайтам, 1 Гб (гігабайт) = 1024 мегабайтам.

Комп'ютер може обробляти тільки інформацію, яка подана в числовій формі. Вся інша інформація (наприклад, звуки, зображення, показання приладів тощо) для обробки на комп'ютері повинна бути перетворена в числову форму.

Таким чином, на комп'ютері обробляється і текстова інформація. При введенні у комп'ютер кожна буква кодується певним числом, а при виведенні на зовнішні пристрої (екран або друк) для сприйняття людиною будуються відповідні зображення букв за цими числами. Відповідність між набором букв і числами називається *кодуванням символів*.

У восьми розрядах, наприклад, можна записати $2^8 = 256$ різних цілих двійкових чисел – від 00000000 до 11111111. Це цілком достатньо для того, щоб дати унікальне (що не повторюється) 8-бітове позначення кожній заголовній і рядковій букві українського та англійського алфавітів, усім арабським цифрам, розділовим знакам, деяким іншим необхідним символам, а також службовим кодам для передачі інформації (тобто всім символам, які застосовуються в комп'ютері).

Таблиця кодування символів 8-бітовими числами називається **кодовою таблицею символів ASCII** (American Standard Code for Information Interchange – американський стандартний код для обміну інформацією). Перша її половина (коди 0 − 127), що містить розділові знаки, арабські цифри і символи англійського алфавіту, є загальноприйнятою в усьому світі. Коди 128−255 (розширені ASCII-коди) використовуються для національних алфавітів і символів для рисування ліній (псевдографіки).

При обробці інформації в комп'ютері, якщо $\{a_i\}$ – числова послідовність, то послідовність $\{S_n\}$ називають *послідовністю часткових сум* ряду (нескінченного), тобто

$$S_1 = a_1;\ \ S_2 = a_1 + a_2; \qquad (1.1)$$

$$S_n = \sum_{i=1}^{n} a_1 + a_2 + \ldots + a_n \qquad (1.2)$$

Обчислення часткової суми доцільно виконувати методом накопичення за формулами:

$$S_0 = 0;$$

$$S_i = S_{i-1} + a_i,\ i = 1 \ldots n,$$

де a_i – елемент підсумовуваної послідовності (функціонального ряду), що називається загальним членом ряду.

Ряд називається *знакопереміжним*, якщо його члени є (по черзі) додатними і від'ємними. Такий ряд можна записати у вигляді

$$\sum_{i=0}^{\infty} (-1)^{i-1} c_i = c_1 - c_2 + c_3 - c_4 + \ldots + (-1)^{i-1} c_i + \ldots,$$

де $c_i > 0$ для будь-якого i.

Формули (1.1) і (1.2) є рекурентними.

Рекурентним називається співвідношення, в якому подальші значення членів ряду визначаються відповідно до попередніх.

Відображення множини натуральних чисел N у множину дійсних функцій однієї змінної x, *визначених* на інтервалі I, *називається* функціональною *послідовністю* і позначається $\{f_n(x)\}$ або $f_1(x), f_2(x), f_3(x), \ldots,$ де функції $f_n(x)$ називаються членами послідовності.

Кожне значення $x \in I$, для якого послідовність має деякий остаточний результат, належить області збіжності цієї послідовності. Таким чином, послідовність визначає в області своєї збіжності деяку функцію

$$f(x) = \lim_{n \to \infty} f_n(x),$$

яка називається *граничною функцією* (або границею) послідовності. Надалі припускатимемо, що область збіжності сбігається з областю визначення I.

Для характеристики граничної функції використовується поняття *рівномірної збіжності*. Функціональна послідовність збігається до граничної функції $f(x)$ рівномірно в I, якщо для будь-якого $\varepsilon > 0$ знайдеться таке $N(\varepsilon)$, не залежне від x, що для всіх $n > N(\varepsilon)$ і для всіх $x \in I$ виконується нерівність

$$\left| f_n(x) - f(x) \right| < \varepsilon.$$

Функціональні ряди. Нескінченний ряд, побудований з функціональної послідовності

$$\sum_{i=1}^{\infty} f_1(x) + f_2(x) + \ldots + f_n(x) + \ldots,$$

називається функціональним рядом.

Функціональна послідовність часткових сум визначається за формулою:

$$S_n(x) = \sum_{i=1}^{n} f_i(x) .$$

Висновки

У даному розділі розглянуті наведені нижче основні питання:

- визначення систем числення та функціональних рядів;
- особливості позиційних систем числення;
- правила переведення чисел з однієї СЧс в іншу;
- застосування СЧс для подання інформації у комп'ютері.

Контрольні питання

1. У якому вигляді подаються дані в комп'ютері?
2. Що таке система числення та які СЧс використовуються в ЕОМ?
3. Що визначає позиція числа?
4. Що визначає основу СЧс?
5. За яким правилом виконується переведення цілої частини числа з десяткової СЧс в будь-яку іншу?
6. За яким правилом виконується переведення дробової частини числа з десяткової СЧс в будь-яку іншу?
7. Якими символами зображуються числа в двійковій, вісімковій, десятковій і шістнадцятковій системах числення?
8. Що таке тріади і тетради і яке їх призначення?
9. За якими правилами виконуються арифметичні операції над числами, поданими в двійковій СЧс?
10. За яким правилом виконується переведення числа з будь-якої системи числення в десяткову?

2. ПРОГРАМИ ТА ТЕХНОЛОГІЇ ЇХ СТВОРЕННЯ

Навчальною метою розділу є ознайомлення студентів із схемами програм та процесом їх створення.

У результаті вивчення даного розділу студенти повинні знати:

- різновиди програм;
- правила відображення алгоритму програми за допомогою схеми;
- приклади символів, які часто використовуються в схемах програм;
- базові конструкції групування символів у схемах програм;
- порядок створення програм у середовищі Microsoft Visual Studio;

уміти:

- створити проект консольної прикладної програми у середовищі Microsoft Visual Studio;
- запустити програму на виконання.

2.1. Початкові відомості про типи програм

Для того щоб комп'ютер мав можливість виконувати будь-які дії з обробки інформації, необхідно скласти для комп'ютера на зрозумілій для нього мові точну та детальну послідовність інструкцій, тобто програму.

Програми можна розділити на 3 категорії:

– *прикладні*, що забезпечують виконання необхідних користувачу робіт: редагування текстів, рисування картинок і т. д.;

– *системні,* що виконують різні допоміжні функції, наприклад, створення копій використовуваної інформації, видання довідкової інформації про операційну систему, перевірку працездатності пристроїв комп'ютера тощо;

– *системи програмування,* що забезпечують створення нових програм для комп'ютера.

Будь-яка програма реалізує деяку сукупність дій, тобто алгоритм.

2.2. Відображення алгоритму програми за допомогою схеми

Алгоритм – основа вирішення будь-якої задачі. Під *алгоритмом* прийнято розуміти точне розпорядження, яке визначає обчислювальний процес – від початкових даних до шуканого результату. Таким чином, алгоритм повинен містити кінцеву послідовність кроків або операцій, що однозначно описує процес переробки початкових і проміжних даних у кінцевий результат.

При складанні алгоритмів слід враховувати ряд вимог, виконання яких приводить до формування необхідних властивостей, а саме:

– алгоритм повинен бути однозначним, тобто таким, що виключає довільність тлумачення будь-якого з розпоряджень і заданого порядку виконання. Ця властивість називається *означеністю*;

– реалізація обчислювального процесу, передбаченого алгоритмом, повинна через певну кількість кроків привести до видачі результату або повідомлення про неможливість вирішення задачі. Ця властивість називається *результативністю*;

– вирішувати однотипні задачі з різними початковими даними можна за одним і тим самим алгоритмом, що дає можливість створювати типові програми для вирішення завдань при різних варіантах значень початкових даних. Ця властивість називається *масовістю*;

– зумовлений алгоритмом обчислювальний процес можна розчленувати на окремі етапи, елементарні операції. Ця властивість називається *дискретністю.*

Короткі алгоритми з декількома командами тримаються в пам’яті, хоча це не ефективно з погляду тиражування. Необхідність запису алгоритмів не викликає сумніву, особливо якщо вони містять десятки або сотні команд, часто використовуються і передаються від одного спеціаліста іншому.

Широке застосування одержав спосіб запису алгоритмів у вигляді схем програм. При цьому способі певні команди відображаються відповідними символами процесу, які з’єднуються певними символами ліній.

Зображення схем програм виконується за стандартом ГОСТ 19.701-90, який також регламентує зображення форми символів та їх розташування.

Відповідно до цього стандарту схеми програм описують послідовність операцій у програмі та включають у себе (табл. 2.1):

– символи процесу, що відображають фактичні операції обробки даних (включаючи символи, що визначають шлях, якого слід дотримуватися з урахуванням логічних умов);
– символ даних;
– символи ліній, що показують потік керування;
– спеціальні символи для полегшення написання і читання схеми;
– специфічні символи;
– короткий пояснювальний текст.

Символи по можливості мають бути одного розміру. Не можна змінювати кути та інші параметри, що впливають на форму символів.

Символи у схемі повинні розташовуватися рівномірно, а лінії з'єднань бути розумної довжини.

Лінії в схемі треба, щоб підходили до символу зліва або зверху, виходили вправо або вниз та спрямовані були до центру символу. Якщо напрямок потоку відрізняється від стандартного, тобто вліво або вверх, то його необхідно вказувати за допомогою стрілки.

У схемах слід уникати ліній, що перетинаються.

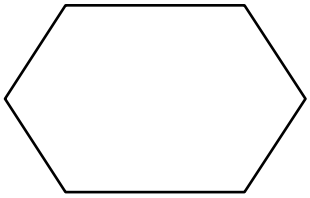
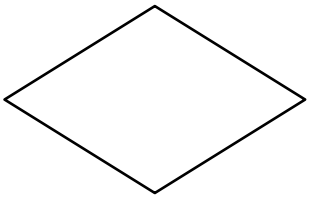
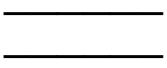
На даний час основна тенденція застосування схем програм полягає в групуванні символів у вигляді *базових конструкцій.*

До них належать проходження, розгалуження і повторення.

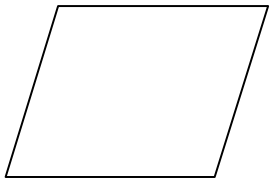
На рис. 2.1 наведені відповідно схеми цих конструкцій. Кожна конструкція має один вхід і один вихід, причому компоненти, що їх складають, зображені вертикально.

Таблиця 2.1

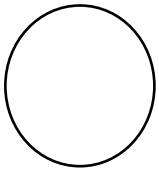
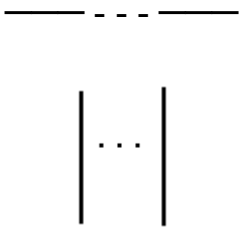
Символи, які часто використовуються в схемах алгоритмів

Символ	Найменування символу	Призначення символу
Символи процесу (основні)		
	Процес	Відображає функцію обробки даних будь-якого вигляду (виконання певної операції або групи операцій, що приводить до зміни значення, форми або розміщення інформації або до визначення, за яким з декількох напрямів потоку керування слід рухатися)
Символи процесу (специфічні)		
	Зумовлений процес	Відображає зумовлений процес, що складається з однієї або декількох операцій або кроків програми, які визначені в іншому місці (підпрограмі, модулі, функції)
	Підготовка	Відображає модифікацію команди або групи команд з метою дії на деяку подальшу функцію (установка перемикача, моди-фікація індексного регістра або ініціалізація програми)
	Розв’язок	Відображає розв’язок або функцію типу перемикача, що має один вхід і ряд альтернативних виходів, один і лише один з яких може бути активізований після обчислення умов, визначених усередині цього символу. Відповідні результати обчислення можуть бути записані поряд з лініями, що показують ці шляхи
	Паралельні дії	Відображає сінхронізацію двух або більшої кількості паралельних операцій

Продовження табл. 2.1

Символ	Найменування символу	Призначення символу
	Межа циклу	Символ складається з двох частин і відображає початок та кінець циклу. Обидві частини символу мають один і той же ідентифікатор. Умови для ініціалізації, приріст, завершення і т. п. розміщаються всередині символу, на початку або у кінці залежно від розміщення операції, що перевіряє умову
Символи даних (основні)		
	Дані	Відображає дані, носій яких не визначений
Символи ліній (основні)		
———————	Лінія	Відображає потік даних або керування. При необхідності можуть бути додані стрілки-покажчики
Символи ліній (специфічні)		
– – – – – – – – – –	Пунктирна лінія	Відображає альтернативний зв’язок між двома або більшою кількістю символів. Крім того, символ застосовується для обведення анотованої ділянки

Закінчення табл. 2.1

Символ	Найменування символу	Призначення символу
Спеціальні символи		
	З’єднувач	Відображає вихід у частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії і продовження її в іншому місці. Відповідні блоки-з’єднувачі повинні містити одне і те саме унікальне позначення
	Термінатор	Відображає вихід у зовнішнє середовище і вхід із нього (початок або кінець схеми програми, зовнішнє використання і джерело або пункт призначення даних)
	Коментар	Використовують для додавання описових коментарів або записів у вигляді пояснень або приміток. Пунктирні лінії у символі коментарю зв’язані з відповідним символом або можуть обводити групу символів. Текст коментарю або приміток повинен бути написаний біля обмежуючої фігури
	Пропуск	Символ (три крапки) застосовують у схемах для відображення пропуску символа чи групи символів, у яких не визначено ні тип, ні кількість символів, а також у символах лінії або між ними, головним чином у схемах, які зображають загальні вирішення з невідомою кількістю повторень

Алгоритм лінійної структури (проходження). Символи в цій структурі розташовуються в схемі у тому самому порядку, в якому повинні бути виконані дії, що ними продиктовані (рис. 2.1, а). Такий порядок виконання називається природним.

Алгоритм розгалуженої структури (розгалуження). Це така схема, в якій передбачене розгалуження вказаної послідовності дій на два напрямки залежно від підсумку перевірки заданої умови (рис. 2.1, б).

Алгоритм циклічної структури (повторення). Це схема, в якій передбачене неодноразове виконання однієї і тієї самої послідовності дій. Такий порядок виконання називається циклом (рис. 2.1, в).

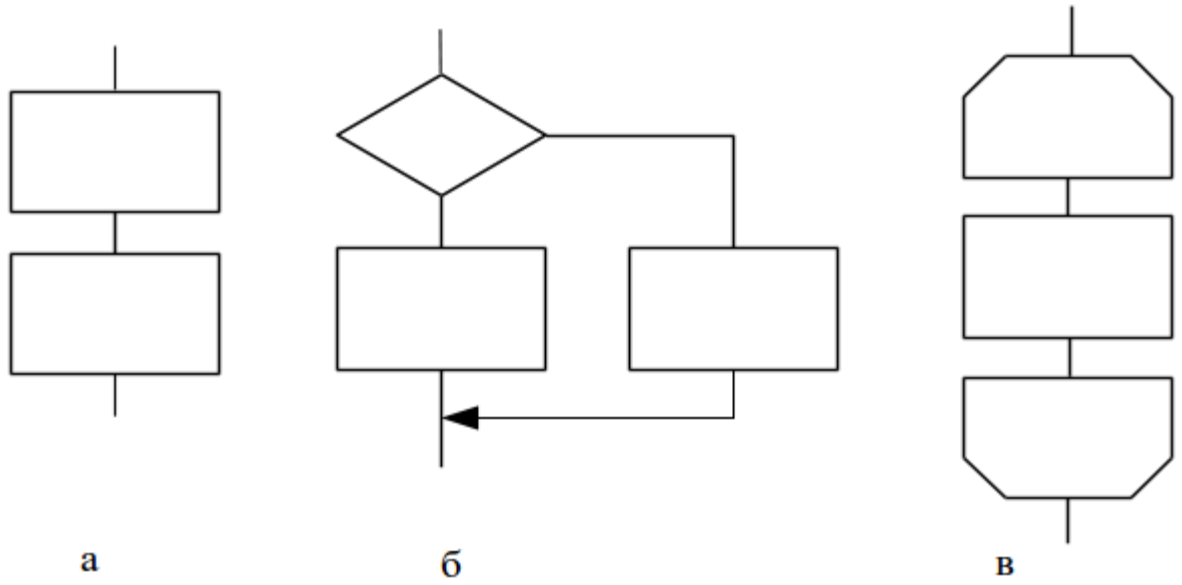


Рис. 2.1. Базові конструкції схем програм: проходження (а),розгалуження (б) і повторення (в)

Приклад 2.1. Скласти схему програми для вирішення такого завдання: дано три дійсні числа. Піднести до квадрата ті з них, значення яких позитивні, і до четвертого ступеня – негативні.

Схема програми прикладу наведена на рис. 2.2.

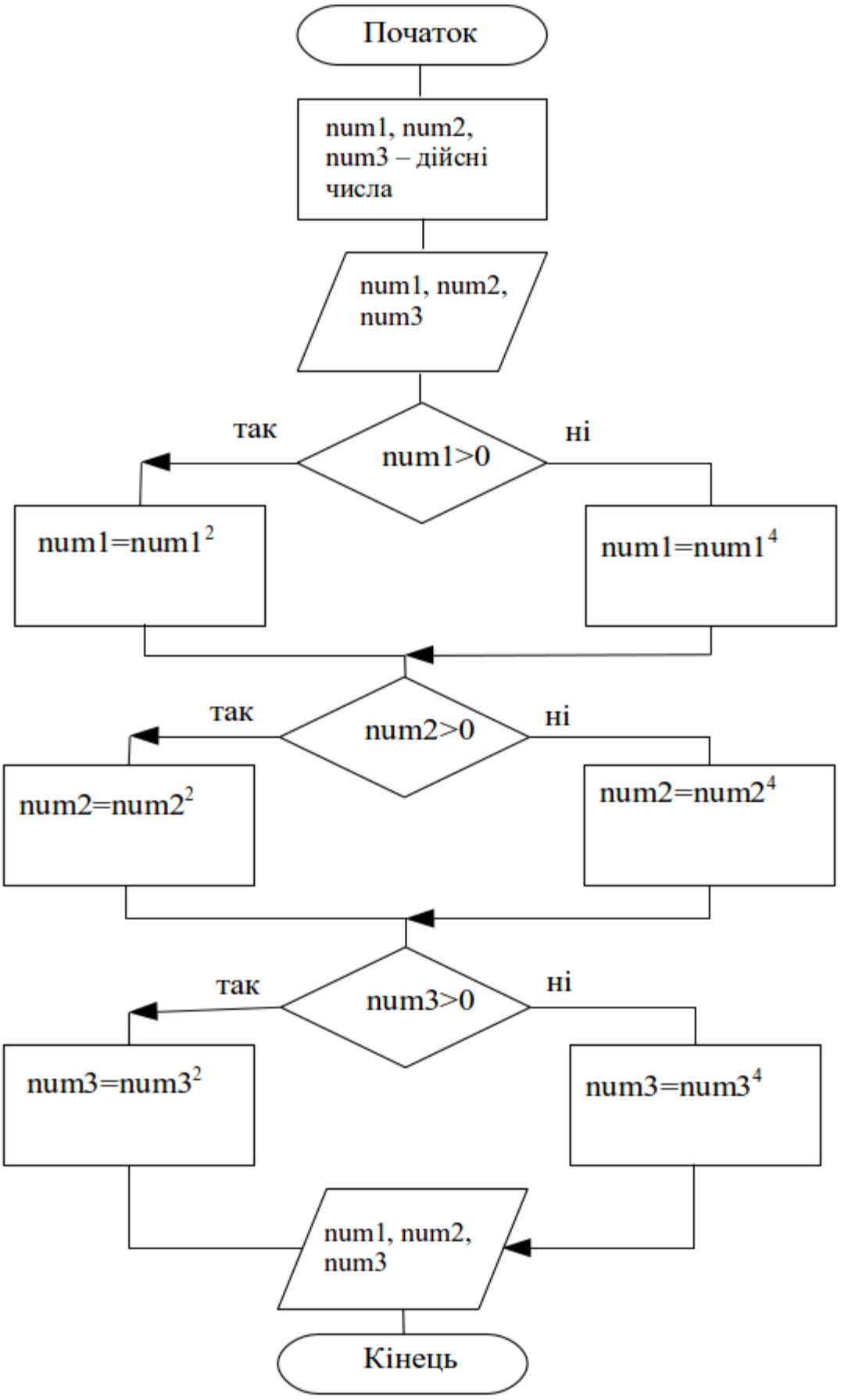


Рис. 2.2. Схема програми прикладу

2.3. Технологія розробки програм за допомогою системи програмування

Для розробки програм мовою С++ найчастіше використовують так звані системи програмування, що дозволяють створювати програми за допомогою певної реалізації мови програмування.

Створення різних версій системи програмування навіть одним і тим же виробником обумовлене розвитком технологій і відображає еволюцію середовища для створення програм.

Прикладом такої системи є Microsoft Visual Studio. Вона включає в себе широкий набір різноманітних інструментів для проектування програм, у тому числі програм мовою С++.

Процес створення програм у системі програмування включає 4 етапи:

1. Написання та редагування вихідного тексту програми за допомогою *редактора (Editor)* та зберігання у вигляді вихідного файла або модуля.

Файли, створювані за допомогою редактора, називаються файлами джерел. Вони звичайно мають розширення *.cpp*, *.cp* або *.c* (для програми на мові С++).

2. Компіляція програми та отримання її на певній проміжній мові зі зберіганням у вигляді об'єктного файлу або модуля.

Файл з вихідним текстом програми – це ще не програма, і її не можна виконати або запустити. Щоб перетворити вихідний текст на програму, використовується *компілятор (Compiler)*. Він переводить програму в машинний код (званий також об’єктним кодом).

У системі С++ перед початком етапу компіляції виконується програма попередньої обробки. Ця програма підкоряється спеціальним командам, званим директивами препроцесора, які вказують, що в програмі перед її компіляцією потрібно виконати певні перетворення. Звичайно ці перетворення полягають у включенні інших текстових файлів у файл, що підлягає компіляції, і виконанні різних текстових замін. Після завершення компіляції вихідного коду створюється об’єктний файл. Цей файл звичайно має розширення *.obj*. Але це ще не виконувана програма.

3. Побудування виконуваного файлу або модуля шляхом об'єднання (компоновки) отриманого об'єктного модуля програми з іншими об'єктними модулями стандартних та спеціальних бібліотек за допомогою *компонувальника (Linker)*.

Програми мовою С++ звичайно створюються шляхом компонування одного або декількох об’єктних файлів з однією або декількома бібліотеками.

Бібліотекою *(Library)* називається колекція компонованих файлів, які або поставляються разом з компілятором, або отримуються окремо, або створюються і компілюються самим програмістом. Бібліотеки підтримують заздалегідь написані модулі, які можна застосовувати в призначених для користувача програмах.

4. Відлагодження програми (покрокове виконання), яке можна проводити за допомогою спеціального компонента (відладчика), що полегшує виявлення помилок.

2.4. Інтерфейс Microsoft Visual Studio

Інтерфейс Microsoft Visual Studio наочний і його використовують під час проглядання компонентів проекту – ресурсів, класів, файлів або діалогової документації (рис. 2.3). У Visual Studio можна працювати з прикладною програмою, як з проектом.

Проект – це набір файлів: заголовків, текстів програм, ресурсів, установок, конфігурацій. Microsoft Visual Studio дає можливість працювати з усіма компонентами проекту одночасно. Для роботи із прикладною програмою необхідно відкрити проект (файл з розширенням .sln для Microsoft Visual Studio 2003 і вище) за допомогою команди File/ Open Solution.

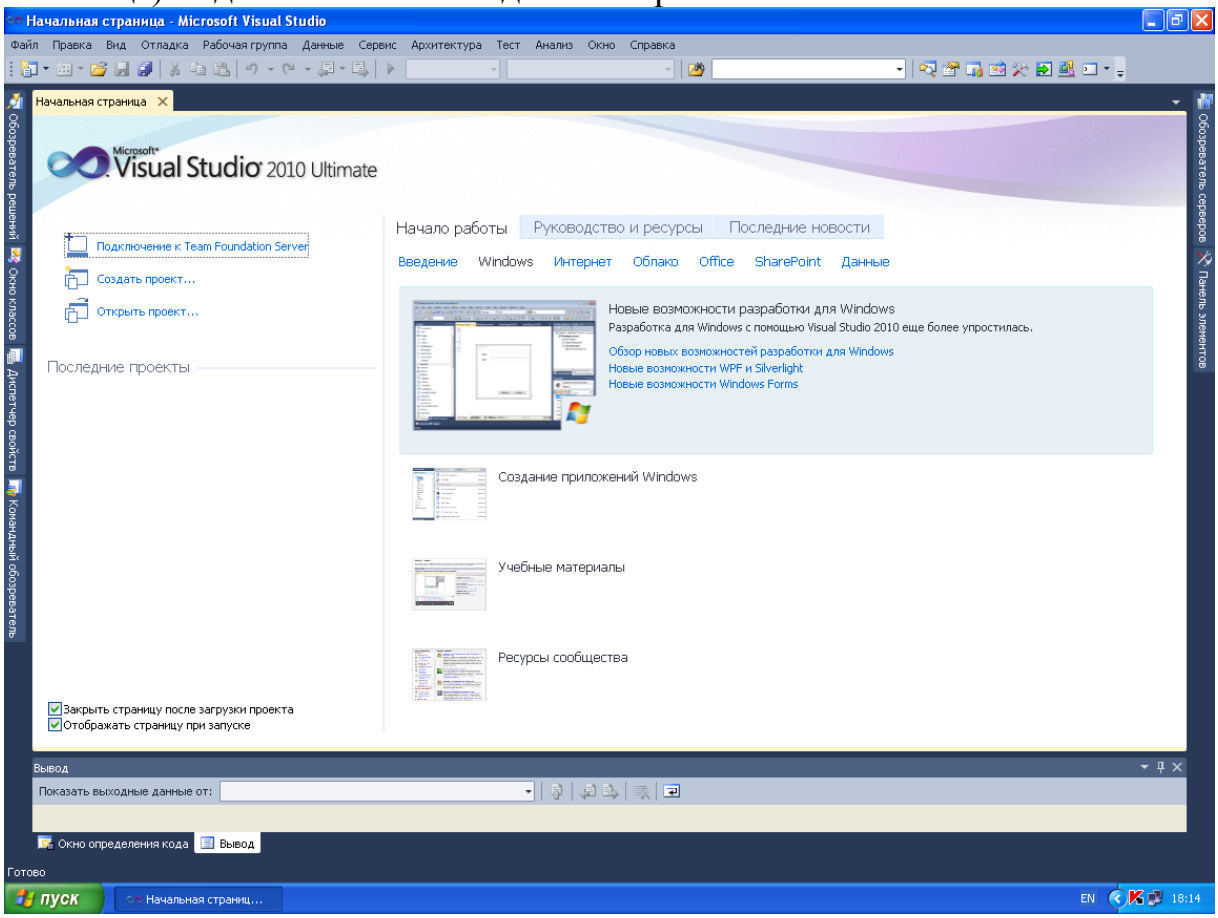


Рис. 2.3. Головна сторінка Microsoft Visual Studio

Екран Microsoft Visual Studio розділений на такі зони (рис. 2.4):

– зверху меню і панелі інструментів;

– вікно *Solution Explorer* (*Обозреватель решений*) призначене для надання допомоги при керуванні великими програмами, що включають безліч файлів вихідного коду, а також визначає, з яким компонентом проекту виконується робота у даний момент;

– *вікно Editor* (*Код*) з'являється праворуч від вікна *Solution Explorer* і використовується для введення і перевірки вихідного коду;

– *вікно Output* (*Вывод*) з'являється в нижній частині екрана, у ньому відображаються повідомлення про хід виконання, синтаксичні помилки і резюме, що належать до виконуваних команд.

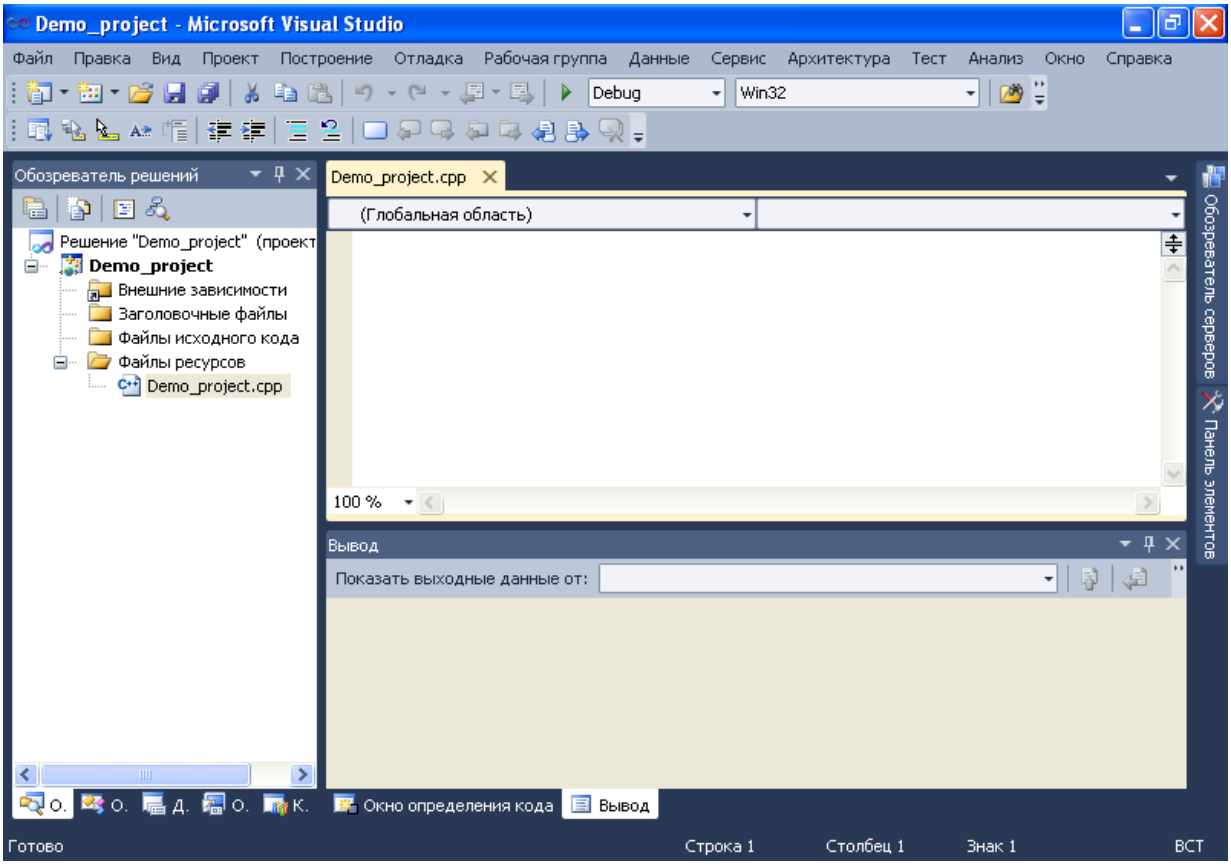


Рис. 2.4. Інтерфейс Microsoft Visual Studio для роботи з прикладною програмою

2.5. Створення проекту консольної прикладної програми мовою C++ у Visual Studio

Для створення консольної прикладної програми мовою C++ необхідно виконати послідовно такі дії:

– запустити Visual Studio і виконати команду *File->New*. При цьому відкриється вікно діалогу *New Project* (рис. 2.5);

– у вікні *Project types* вибрати: *Visual C++/Win32,* у вікні *Templates* – прикладну програму *Win32 Console Application*. У поле *Name* ввести ім'я проекту, а в полі *Location* вказати шляхи каталогів, у яких планується зберігати файли проекту. Натиснути на кнопку *OK*;

– у діалоговому вікні, що з'явилося, вибрати вкладку *Application Settings* (рис. 2.6). Далі вибрати *Application type*: *Console application*, *Additional options*: *Empty project* і натиснути кнопку *Finish.* У результаті створюється порожній проект, до якого потрібно додати файл джерела;

– відкрити вікно *Solution Explorer* (меню *View*) для проглядання вмісту проекту.

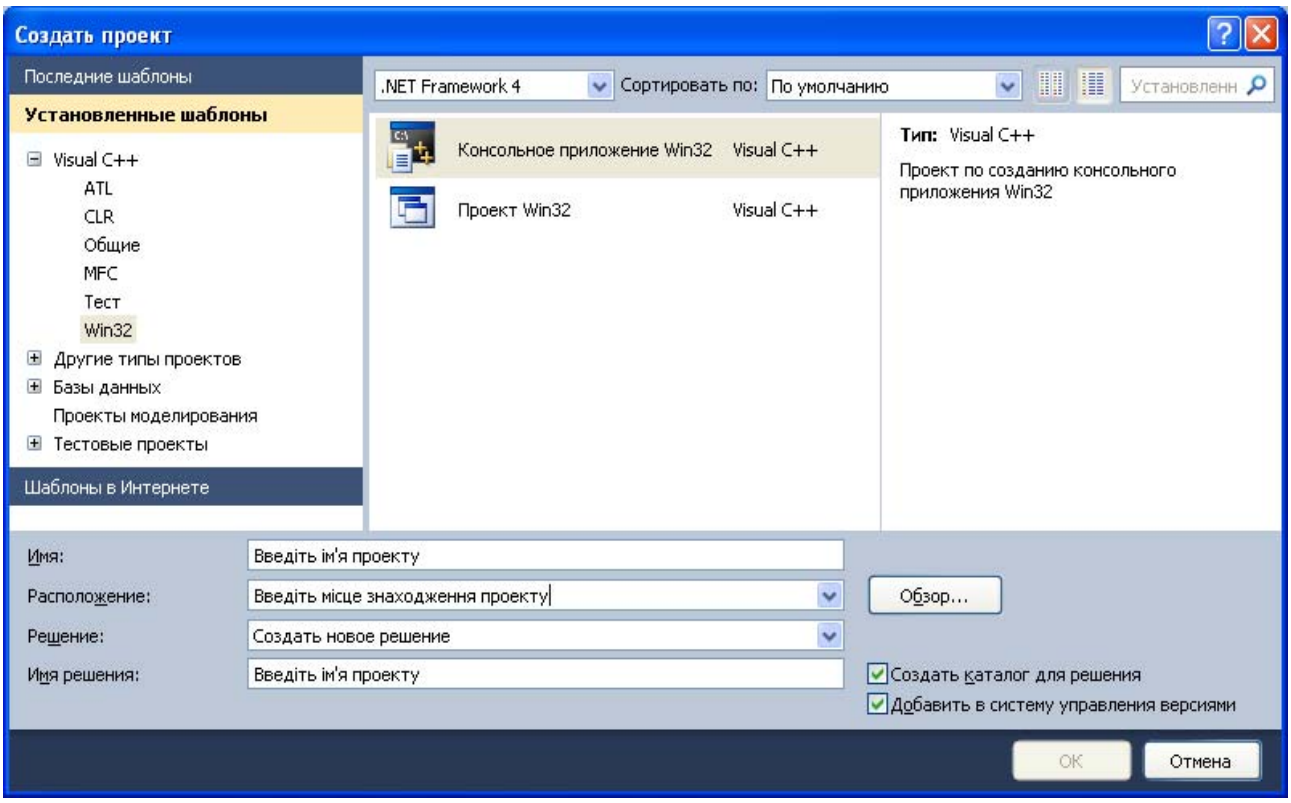


Рис. 2.5. Створення проекту

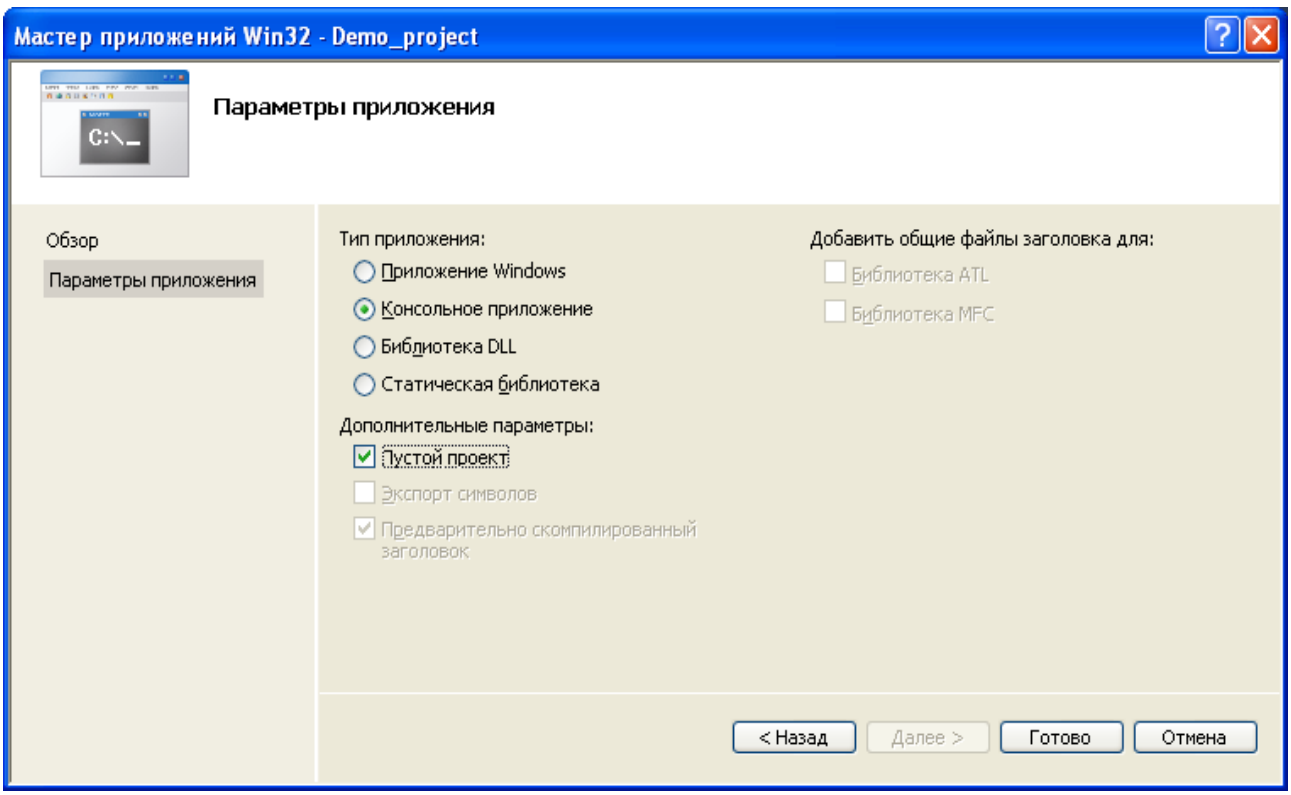


Рис. 2.6. Вибір типу проекту

Для того, щоб у проект додати файл джерела, необхідно виконати послідовно такі дії (рис. 2.7):

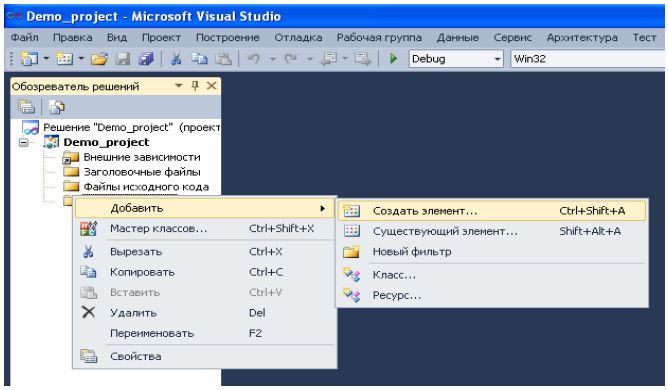


Рис. 2.7. Додавання файлу у проект

– вибрати у вікні *Solution Explorer* папку *Source Files* і з контекстного меню вибрати команду *Add/Add New Item* (або використовуючи команду *Add New Item* з меню *Project*);

– з'явиться діалогове вікно *Add New Item*. У вікні *Categories* вибрати *Code*, у вікні *Templates – C++ File(.cpp)*, у полі *Name* написати ім'я файлу джерела (рис. 2.8). Натиснути кнопку *Add*.

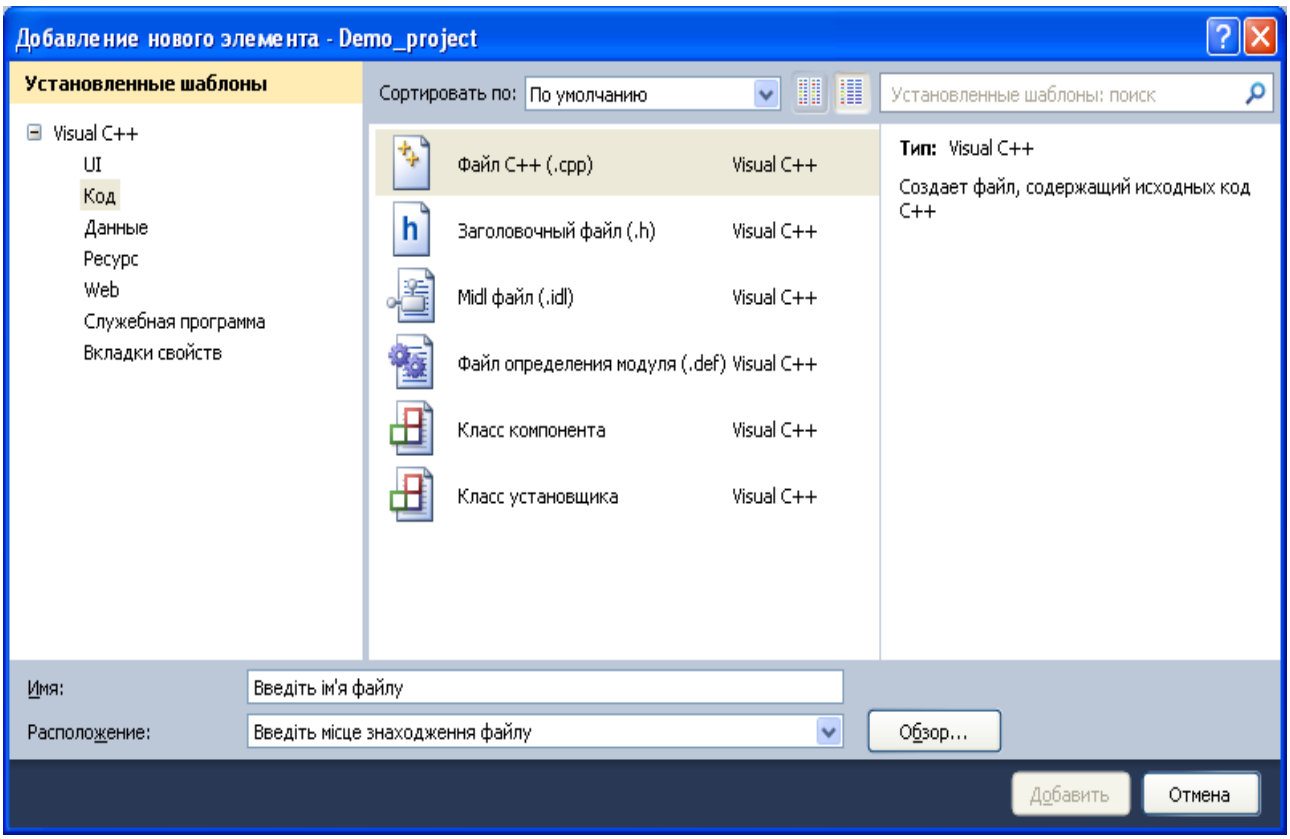


Рис. 2.8. Введення ім'я файла джерела

2.6. Запуск програми

Для запуску програми спочатку потрібно вибрати з меню *Build* команду *Build Solution (F5)*. Ця команда запускає компілятор та компонувальник. А потім, якщо немає помилок компіляції, з меню *Debug* вибрати команду *Execute або Start Without Debugging (Ctrl+F5)*, щоб запустити програму.

У результаті компіляції можуть виявитися помилки. Помилки бувають двох типів: синтаксичні та логічні.

Синтаксичні помилки виявляються при складанні проекту і відображаються у вікні Output.

Висновки

У даному розділі розглянуті наведені нижче основні питання:

- визначення алгоритму;
- правила застосування схем програм;
- символи, які часто використовуються в схемах програм;
- базові конструкції групування символів у схемах програм;
- компоненти Microsoft Visual Studio;
- інтерфейс Microsoft Visual Studio;
- запуск програми у середовищі Microsoft Visual Studio.

Контрольні питання

1. Дати визначення схеми програми.
2. За яким стандартом виконується зображення схем програм, а також регламентується форма символів та їх розташування?
3. Які властивості алгоритму і в чому його суть?
4. Яким символом позначають переходи керування за умовою?
5. Які символи використовуються для організації циклу?
6. З яких компонентів складається Microsoft Visual Studio?
7. Що таке виконуваний файл і як його створити?
8. Що таке проект і з чого він складається?
9. Як у проект додати файл джерела?
10. Як запустити програму на виконання?

3. СТРУКТУРА ТА БАЗОВІ СКЛАДОВІ ПРОГРАМ

Навчальною метою розділу є ознайомлення студентів із структурою та базовими складовими програми, написаної мовою С++.

У результаті вивчення даного розділу студенти повинні знати:

- структуру програми, написаної мовою С++;
- визначення та використання змінних і констант;

визначення виразу та операції;

уміти:

- застосовувати вирази для виконання арифметичних операцій;
- використовувати операції вводу-виводу в консоль.

3.1. Структура програм, написаних мовою С++

У мові С++ дії визначаються *виразами* та *операторами*. Кожен оператор обов'язково закінчується крапкою з комою (;).

Оператори групуються в іменовані блоки, що називаються функціями.

Програма мовою С++ складається з функцій.

У кожній програмі мовою С++ повинна бути функція *main()*, без якої запуск програми неможливий. Виконання програми мовою С++ починається з першого оператора функції main() і закінчується виконанням останнього оператора функції main(). Всі оператори функції знаходяться всередині фігурних дужок {} – це тіло функції.

Найменша програма на С++ виглядає так:

```
main ()
{ }
```

Функція *main* у данному випадку нічого не робить.

Звичайно програма має видавати якісь результати. Ось програма, яка виводить у консольне вікно вітання Hello, Ukraine!:

```
#include<iostream>
using namespace std;
int main ()
{
    cout << "Hello, Ukraine!\n";
    return 0;
}
```

Ввід та вивід у мові С++ не належать безпосередньо до мови С++. Вони забезпечуються стандартними бібліотеками. Для С++ така бібліотека має назву *iostream*.

При створенні виконуваного коду програм С++ використовується *препроцесор*. Це програма, яка обробляє початковий файл перед основною компіляцією. Препроцесор обробляє директиви, що починаються із символу #, і запускається автоматично перед компіляцією.

Директива *#include* приводить до того, що препроцесор додає в програму копію вказаного в директиві файлу (у даному прикладі файлу *iostream*, тобто вміст файлу *iostream* замінює в програмі рядок *#include <iostream >*.

Таким чином, рядок *#include <iostream >* – це директива препроцесора, що підключає заголовний файл *iostream*, необхідний для роботи із стандартними потоками вводу-виводу. Цей файл повинен бути включений для всіх програм, які виводять дані на екран або вводять дані з клавіатури.

Для того, щоб визначення у файлі *iostream* були доступні програмі, необхідно використовувати директиву препроцесора для зони імен *using*:

using namespace std;

Операція << (помістити в потік) записує свій другий параметр у перший. У даному випадку рядок "Hello, Ukraine!\n" записується в стандартний вихідний потік *cout* – визначений об'єкт для відображення на екрані різноманітних даних, включаючи рядки, числа, окремі символи. Рядок – це послідовність символів, узята в подвійні лапки.

Два символи *\n* − спеціальний символ, який є символом кінця рядка, тому він видається у прикладі після символів *Hello, Ukraine!*

У рядках, що виводяться за допомогою об'єкта *cout*, можна використовувати комбінації спеціальних символів, першим символом яких є зворотна коса риска "**" (її також називають лівим або зворотним слешем):

– *\n,* новий рядок;
– *\b,* повернення на один символ;
– *\r,* повернення на початок рядка;
– **, вивід зворотного слеша;
– *\'*, вивід одинарної лапки;
– *\"*, вивід подвійної лапки;
– *\?*, вивід знаку питання;
– *\t*, символ табуляції;
– *\a*, звуковий сигнал.

В одному операторі виводу можна об'єднувати декілька операцій << (помістити в потік).

Наприклад:

cout << "Мені " << 17 <<" років";

На екрані ми побачимо строку: Мені 17 років.

Рядок *return 0* включається в кінці кожної функції *main*. Ключове слово *return* – один з декількох способів виходу з функції. Коли оператор *return* використовується в кінці *main()*, значення 0 свідчить про те, що програма завершена успішно. Ціле значення, повернуте функцією *main()*, якщо тільки воно є, вважається повернутим для системи значенням програми.

Якщо нічого не повертається, то для системи значення програми набуде якогось випадкового значення.

Приклад 3.1. Створити програму, що виводить на екран геометричну фігуру за допомогою заданого символу (табл. 3.1).

Таблиця 3.1.

Завдання для прикладу

Геометрична фігура	Символ виводу
Ромб	*

```
#include <iostream>
using namespace std;
int main()
{
   cout<<"\t\t\t\tLaboratory work No.3\n";
   cout<<"\t\t\t\tgroup: KI – 11 – 1\n";
        cout<<"\t\t\t\tPetrenko S.I.\n\n\n";
        cout<<"     *\n";
        cout<<"  *    *\n";
        cout<<"*         *\n";
        cout<<"  *    *\n";
        cout<<"     *\n";
}
```

3.2. Змінні та базові типи даних

Змінна – це область у пам'яті комп'ютера, де може зберігатися деяке значення для використання його в програмі. Кожна змінна має ім'я, тип, розмір і значення.

Ім'я змінної – це будь-який допустимий ідентифікатор.

Ідентифікатор – це послідовність символів, що містить букви, цифри і символи підкреслення, але не починається з цифри. У С++ допускаються ідентифікатори будь-якої довжини. Мова С++ чутлива до регістра, тобто великі та малі букви розрізняються (a1 і A1 – різні ідентифікатори).

Під *типом даних* розуміють безліч допустимих значень цих даних і безліч дозволених операцій над ними. Одночасно тип даних визначає і розмір пам'яті, який займають змінні й константи даного типу (табл. 3.2).

Таблиця 3.2

Базові типи даних

Тип мовою С++	Тип даних	Розмір, байт	Діапазон значень
bool	Логічні	1	true або false
void	Порожній		Не має значення
int	Цілі	*	Залежить від системи
short	Цілі	2	Від -32768 до 32767
long	Цілі	4	Від -2 147 483 648 до 2 147 483 647
char	Символьні	1	Від –128 до 127
float	Речовинні	4	3.4E +/- 38 (7 знаків)
double	Речовинні	8	1.7E +/- 308 (15 знаків)
long double	Речовинні	8	1.7E +/- 308 (15 знаків)

*Тип int має розмір системного слова. Це два байти в MS-DOS і 16-розрядній версії Windows та 4 байти в 32-розрядних операційних системах.

Усі змінні повинні бути описані (оголошені) з вказівкою імені й типу даних, перш ніж вони використовуватимуться в програмі.

Декілька змінних одного типу можуть бути записані в одному оголошені:

```
int a, b, c;
```

або в декількох:

```
int a;
int b;
int c;
```

Компілятору необхідно вказати тип змінної, для того щоб він знав, скільки місця зарезервувати для змінної і якого роду значення зберігатимуть у ній.

Щоб визначити розмір пам'яті, виділеної для зберігання змінної, слід використовувати функцію sizeof().

Наприклад:

```
cout<<" Розмір базового типу int = "<<sizeof(int)<<endl;
```

Змінні необхідно *ініціалізувати*, тобто привласнити їм яке-небудь значення.

Ініціалізувати змінні можна при оголошенні: int A=3;

Таким чином, ми оголосили змінну *A* цілого типу і привласнили цій змінній значення 3.

При оголошенні змінної для неї виділяється пам'ять. Резервування пам'яті не очищає чарунки від значень, які раніше в них зберігалися. Тому, якщо після оголошення змінної не відбувається її ініціалізація, то поточне значення змінної буде випадковим, а не нульовим.

Цілочислові змінні можуть бути *знаковими* (*signed*) і *беззнаковими* (*unsigned*). Знакові змінні можуть бути додатними або від'ємними, беззнакові – тільки додатними.

Якщо беззнаковій змінній привласнити значення, що перевищує його граничний максимум, тобто при переповненні, значення змінної скидається в нуль і відлік починається спочатку. При переповненні знакової змінної, при виході за межі максимального позитивного значення змінна приймає мінімальне негативне значення.

3.3. Константи і способи їх завдання

За способом завдання константи можна розділити на *символічні* і *літеральні*.

Звичайно константи задаються у виразах як конкретні значення операндів і значень параметрів функцій. Сама константа є простим виразом.

Символічними константами (константними змінними) називаються змінні, які одержують як початкові значення постійні вирази. Значення символічної константи не може бути надалі модифіковано.

Для оголошення символічних констант використовують ключове слово *const*.

const float pi=3.14, e=2.72;

Літеральна константа є константою, тип і значення якої визначаються її зовнішнім виглядом. У необхідних випадках пам'ять під константу виділяється автоматично, але може бути не виділена зовсім, наприклад, для літералів у складі константного виразу. Вміст виділеної пам'яті не змінюється і визначається типом і величиною константи.

Розрізняють такі види *літеральних констант:*

– *числові*: 3.14, -2.5, 254, 0x00A7, 056. Літерал, який починається з нуля і символу '*x*' (*0x*), є шістнадцятковим числом (з основою 16). Літерал, який починається з нуля, є вісімковим числом (з основою 8);

– *строкові* – послідовність символів, узятих у подвійні лапки: "*This string constant*", "*ERROR!!!*";

– *символьні*: '*f*', '*1*', '*G*', '*n*', '*t*' – символ, узятий в одиночні лапки.

3.4. Вирази

Вираз – це правило для обчислення нового значення. Вирази в С++ можуть бути арифметичні, логічні, умовні, над символами і рядками і над адресами. Вирази формуються з операндів, операторів і круглих дужок. Як операнди можуть бути константи, змінні й результати функцій. Тип результату виразу визначається як типом операндів, так і типом результату, тобто значенням, яке формується при виконанні останньої операції виразу.

Арифметичні вирази – це аналог виразів алгебри і математики.

У табл. 3.3 наведені арифметичні операції, типи операндів, до яких вони застосовні, і типи результатів.

3.5. Операції

Усе, внаслідок чого з'являється деяке значення, у мові С++ називається операцією. Про операції говорять, що вони повертають значення. Так, операція 3+2 повертає значення 5. Всі операції є разом з тим і виразами.

Вираз *x=a+в* не тільки складає значення змінних *a* і *в*, але і присвоює результат змінної *x*, тобто повертає результат підсумовування змінної *x*. Тому даний вираз цілком можна назвати операцією. Операції завжди розташовуються праворуч від оператора присвоювання.

3.6. Ввід даних з консолі

Консо́ль − це пристрій, який забезпечує взаємодію оператора ПК з операційною системою. Як правило, як консоль використовується дисплей і клавіатура. Ввід, що виконується з клавіатури користувачем, називається стандартним *вхідним потоком* або стандартним вводом (вводом з консолі). Він зв'язується із зумовленим в iostream.h потоком cin.

Операція вводу >> читає значення із стандартного вхідного потоку, наприклад

cin >> index;

де index, змінна, в яку буде записано значення, введене з клавіатури користувачем.

Такі операції також можна з'єднувати в одному операторі. Наприклад, якщо в програмі зустрінеться такий оператор:

cin >> i1 >> i2;

то програма буде чекати ввода з клавіатури двох величин, першу з них вона помістить в змінну i1, а другу − в змінну i2. Ці дві введених величини можна розділяти пробілом або табуляцією, а можна кожну з них вводити з нового рядка. Операція вводу працюватиме правильно.

Таблиця 3.3

Арифметичні операції

Операція	Призначення операції	Типи операндів	Тип результату
+	Унарний плюс	Цілі, дійсні	Збігається з типом операнда
–	Унарний мінус	Цілі, дійсні	Збігається з типом операнда
*	Множення	Цілі, дійсні	Відповідно до ієрархії типів
/	Ділення	Цілі, дійсні	Відповідно до ієрархії типів
%	Остача від ділення	Цілі	Цілий
оп =	Змінити і замінити, де <оп> (оператор) можливо +, - *, / або % (для цілих), <<, >>, &, ^, |	Цілі, дійсні	Відповідно до ієрархії типів
++	Інкремент (збільшити на 1)	Цілі, дійсні	Збігається з типом операнда
– –	Декремент (зменшити на 1)	Цілі, дійсні	Збігається з типом операнда

Висновки

У даному розділі розглянуті наведені нижче основні питання:

- визначення змінних і базових типів даних;
- вирази та операції;
- застосування виразів для виконання арифметичних операцій, типи операндів, з якими вони працюють, і типи результатів;
- ввід даних з консолі.

Контрольні питання

1. Що таке змінна і які параметри вона має?
2. Що розуміється під типом даних?
3. Яке значення набуває неініціалізована змінна?
4. Яке значення набуває ціла знакова змінна при переповнюванні?
5. Що таке константа і які існують способи завдання констант?
6. Що таке вираз і чим визначається тип виразу?
7. Які арифметичні операції використовуються в мові С++?

4. ОПЕРАТОРИ

Навчальною метою розділу є ознайомлення студентів із операторами мови програмування С++.

У результаті вивчення даного розділу студенти повинні знати:

– визначення оператора;
  види математичних операторів;
  види операторів присвоювання;
  визначення пріоритету операторів;
  уміти:
  застосовувати у програмах операції інкременту і декременту.

Оператор – це літерал, який змушує компілятор виконувати деяку дію. Оператори впливають на операнди. Операндами в С++ можуть бути як окремі літерали, так і цілі вирази. Мова С++ має в своєму розпорядженні два види операторів:

– математичні оператори;
– оператори присвоювання.

4.1. Математичні оператори та функції

У С++ використовується 5 математичних операторів: складання (+), віднімання (-), множення (*), ділення (/), ділення по модулю (%).

Ділення дещо відрізняється від звичайного. При діленні цілого числа на ціле результатом буде ціле число.

Наприклад, при діленні 21 на 4 відповідь виходить 5.

Для того щоб отримати дійсний результат, необхідно, аби хоча б один з операндів був дійсний (див. далі підрозділ 4.5. Перетворення типів).

Щоб одержати остачу від ділення, потрібно розділити число 21 по модулю на 4, у результаті одержимо залишок 1. Ділення по модулю часто використовується для знаходження чисел, кратних заданому.

У мові С++, окрім математичних операторів, можна використовувати також і математичні функції. Для застосування математичних функцій необхідно підключити бібліотеку з математичними функціями, тобто в програму включити рядок:

#include <math.h>.

Приклади вбудованих математичних функцій, опис яких знаходиться в заголовному файлі <*math.h*>, наведені в табл. 4.1.

Майже всі математичні функції приймають аргументи тільки дійсного типу: *float* або *double*.

Тригонометричні функції набувають аргументи, задані в радіанах.

Наприклад:

```
float A=sin(1.570796);
```

4.2. Оператори присвоювання

У мові С++ запис значення в змінну, тобто присвоювання змінній нового значення може проводитися за допомогою операторів "=" та "оп =", де <оп> (оператор) можливо +, -, *, /, %, <<, >>, &, ^, |, а також за допомогою префіксних і постфіксних операцій "++" і "- -".

Наприклад:

A = b * 5 + 28;

Тут спочатку розраховується значення виразу b*5+28, а потім результат виразу записується в змінну A, тобто присвоюється змінній A.

Таблиця 4.1

Перелік вбудованих математичних функцій файлу *<math.h>*

Вбудовані функції	Повернутий результат
abs(x)	Модуль аргументу *x*,де *x* – ціле
fabs(x)	Модуль аргументу *x,* де *x* – дійсне
atan(x)	Арктангенс (кут у радіанах)
sin(x)	Синус (кут у радіанах)
sinh(x)	Синус гіперболічний
cos(x)	Косинус (кут у радіанах)
cosh(x)	Косинус гіперболічний
tan(x)	Тангенс (кут у радіанах)
tanh(x)	Тангенс гіперболічний
exp(x)	Експонента e^x
log(x)	Логарифм натуральний
log10(x)	Логарифм десятковий
sqrt(x)	Корінь квадратний аргументу
pow(x,y)	Значення *x* у степені *y*
fmod(x,y)	Остача від ділення двох чисел: *x* на *y*
floor(x)	Найближче менше ціле <=*x*
ceil(x)	Найближче більше ціле >=*x*

Дуже часто буває, що необхідно збільшити значення змінної, а потім результат присвоїти цій змінній. Наприклад:

```
int MyAge=5;
int temp;
temp=MyAge+2;
MyAge=temp;
```

У мові С++ можна помістити одну і ту саму змінну по обидві сторони оператора присвоювання. Таким чином, попередній блок зведеться до одного виразу:

```
MyAge= MyAge+2;
```

Існує ще простіший варіант попереднього запису:

```
MyAge+=2;
```

Ця скорочена форма оператора присвоювання із сумою (+=) спочатку додає до значення змінної MyAge 2, а потім результат додавання присвоює змінній MyAge.

Скорочена форма оператора присвоювання (оп=) застосовується, коли змінна лівої частини оператора присвоювання використовується і в правій частині оператора присвоювання. У загальному вигляді скорочена форма оператора присвоювання може бути подана так:

A оп= B, що еквівалентно *A = A оп (B),* де

A – ім'я змінної, яка набуває нового значення;

B – вираз для отримання нового значення, дужки позначають обчислення виразу *B* до виконання операції *оп;*

оп – один з операторів: +, -, *, /, %, <<, >>, &, ^, |.

Наприклад, оператор a *= c +1 еквівалентний оператору a = a * (c + 1).

4.3. Інкремент і декремент

Дуже часто в програмах до змінних додається або віднімається одиниця. У мові C++ збільшення змінної на 1 називається інкрементом, а зменшення – декрементом. Причому ці операції можуть бути:

– префіксні: ++*a* і - - *a*;

– постфіксні: *a*++ і *a* - -,

де *a* – ідентифікатор змінної.

У разі використання префіксної форми операції інкремента / декремента у виразі, значення змінної спочатку збільшується на 1 / зменшується на 1, а потім використовується в виразі.

У разі використання постфіксної форми операції інкремента / декремента у виразі, значення змінної спочатку використовується в виразі, а потім збільшується на 1 / зменшується на 1.

Наприклад:

a=2; в=3; c=1.

a=в++c*; це еквівалентно *c=c+1, a=в*c*. В результаті *a=6*;

*a=в*c*++; це еквівалентно *a=в*c, c=c+1*. В результаті *a=3*,

оскільки *c* збільшиться тільки після його використання у виразі для обчислення значення *a*.

4.4. Пріоритети операторів

Пріоритет оператора визначає послідовність, у якій програма виконує оператори у виразі або формулі. Якщо один оператор має пріоритет над іншим, то він виконується першим.

У табл. 4.2 наведені різні оператори мови C++ у порядку спадання пріоритету. Їх пріоритети для кожної групи однакові.

Таблиця 4.2

Операції C++ у порядку спадання пріоритету

Операції	Призначення операцій
( )	Виклик функції
[ ]	Виділення елементу масиву
.	Виділення елементу запису
->	Виділення елементу запису
!	Логічне заперечення
~	Порозрядне заперечення
-	Зміна знаку
++	Збільшення на одиницю
--	Зменшення на одиницю
&	Узяття адреси
*	Звернення за адресою
(тип)	Перетворення типу (тобто (float) a)
sizeof( )	Визначення розміру в байтах
*	Множення
/	Ділення
%	Визначення остачі від ділення
+	Складання
-	Віднімання
<<	Зсув уліво
>>	Зсув управо
<	Менше ніж
<=	Менше або дорівнює
>	Більше ніж
>=	Більше або дорівнює
= =	Дорівнює
!=	Не дорівнює
&	Порозрядне логічне "І"
^	Виключальне "АБО"
|	Порозрядне логічне "АБО"
&&	Логічне "І"
||	Логічне "АБО"
?:	Умовна (тернарна) операція
=	Присвоювання
+=, - =, *=, /= %=, <<=, >>=, &=, |=, ^=	Бінарні операції (наприклад, a *= b (тобто a = a * b) і т.д.)

Порядок виконання оператора може регулюватися за допомогою круглих дужок. Якщо є вкладені дужки, то обчислення виразу починається з внутрішніх дужок. Якщо у виразі не використовуються круглі дужки, що задають послідовність виконання операцій, усередині пари дужок ця послідовність визначається з урахуванням відносного пріоритету операцій.

4.5. Перетворення типів

Мова C++ підтримує змішані вирази з різними типами даних. Тобто компілятори допускають можливість виконання арифметичних операцій з операндами різних типів.

Існує явне і неявне перетворення типів.

4.5.1. Неявне перетворення типів

Неявне перетворення виконується компілятором автоматично в двох випадках, якщо:

– у виразі присутні операнди різних типів;
– змінній одного типу присвоюється значення іншого типу.

У першому випадку перетворення типів виконується згідно з ієрархією типів. Кожен тип даних можна умовно визнати «вище» або «нижче» по відношенню до інших типів.

Ієрархія типів наведена у табл.4.3.

Таблиця 4.3

Ієрархія типів даних

Тип даних	Старшинство
long double	Ще вище
double	Найвищий
float	Вищий
long	Середній
int	Нижчий
short	Найнижчий
char	Ще нижче

Наприклад:

```
int a = 5;
float b = 2.5;
float c = a + b;
```

У даному випадку при розрахунку значення виразу $a + b$ спочатку тип значення змінної a перетворюється на float (тобто $5 \rightarrow 5.0$), і отримане значення зберігається у тимчасовій змінній. Тільки після цього перетворення виконується операція додавання. Результатом виразу $a + b$ буде значення типу float.

У випадку, коли змінній одного типу присвоюється значення іншого типу, тип збережуваного значення автоматично змінюється на тип змінної.

Наприклад:

```
int a = 5;
float b = 2.5;
int c = a + b;
```

Результатом виразу *a + b* буде значення типу float. Але значення типу float не можливо зберегти у змінній типу int, тому тип результату виразу змінюється на int, а отримане значення записується в змінну. В результаті отримуємо змінну *c* із значенням 7.

4.5.2. Явне перетворення типів

Явні перетворення типів робляться програмістом. Такі перетворення необхідні, якщо компілятор не здатен безпомилково перетворити типи автоматично.

У C++ існує декілька різних операцій перетворення типів:

– static_cast<тип даних>(ім’я змінної);
– (тип даних) ім’я змінної;
– тип даних (ім’я змінної);

Наприклад:

```
char aCharVar = static_cast<char>(anIntvar);
```

Змінна, тип якої змінюється, міститься в круглих дужках, а тип, у який перетворюється змінна – в кутових.

```
aCharVar = (char)anIntVar;
```

або

```
aCharVar = char(anIntVar);
```

Тип значення змінної *anIntVar* змінюється до типу *char*.

Приклад 4.1. Вивести на екран у вигляді таблиці інформацію про дійсні типи даних – ім’я типу даних, розмір, діапазон значень.

```
#include<iostream>
using namespace std;
int main( )
{
  cout<<”-------------------------------------------------\n”;
      cout<<”|   Type      | Size  |       Diapazon       |\n”;
      cout<<”-------------------------------------------------\n”;
      cout<<”|    float    |   4   |    3.4E +/- 38       |\n”;
      cout<<”-------------------------------------------------\n”;
      cout<<”|   double    |   8   |    1.7E +/- 308      |\n”;
      cout<<”-------------------------------------------------\n”;
      cout<<”| long double |   8   |    1.7E +/- 308      |\n”;
      cout<<”-------------------------------------------------\n”;
}
```

Приклад 4.2. Увести з клавіатури значення змінних *a, b* (дійсні) та *x* (ціле). Обчислити і вивести на екран значення функції:

$$k = \frac{a \lg x^{|a-b|} - 2\sin(b^2)}{e^{3x}}$$

```
#include<iostream>
using namespace std;
#include<math.h>
int main( )
{
        cout<<" k=(a*log(pow(x,abs(a-b))) – 2*sin(b*b)) / exp(3*x) = ?\n";
        cout<<"Enter 2 double numbers (a and b)\n";
        double a, b;
        cin>>a>>b;
        cout<<"Enter integer number (x)\n";
       int x;
       cin>>x;
       double k=(a*log(pow((double)x,fabs(a-b)))-
       2*sin(b*b))/exp((double)3*x);
       cout<<"k = "<<k<<'\n';
}
```

Висновки

У даному розділі розглянуті наведені нижче основні питання:

- математичні оператори;
- оператори присвоювання;
- особливості операцій інкремента і декременту;
- пріоритети операцій.

Контрольні питання

1. Назвіть види математичних операторів.
2. Назвіть види операторів присвоювання.
3. Яка відмінність спостерігається між префіксними і постфікс ними операціями?
4. Які відомі чотири різні способи збільшення одиниці до цілої змінної x? Напишіть їх.
5. У чому різниця між операціями *y++* і *y+1*?
6. Як пріоритети операцій застосовуються у програмах?

5. ЛОГІЧНІ ОПЕРАЦІЇ

Навчальною метою розділу є ознайомлення студентів з логічними операціями мови програмування С++.

У результаті вивчення даного розділу студенти повинні знати:

– визначення логічного виразу;
  логічні операції;
– визначення виразів порівняння;
  уміти:
– навести приклади побітових логічних операцій;
– описати застосування форматованого виводу даних.

Логічний вираз – це один із засобів алгебри логіки. Логічні вирази застосовуються для аналізу, тобто для визначення істинності або хибності певних ситуацій. Логічний вираз – це два операнди, сполучені логічною операцією або операцією відношення. Тип логічної операції цілий: якщо результат операції істинний, то мовою С++ значення результату дорівнює 1, а якщо хибний, то значення результату – 0.

Логічні вирази використовуються:

– в умовних операторах і умовних виразах;
– в операторах присвоювання;
– у заголовках циклів;
– у списках фактичних параметрів функцій, наприклад, функцій вводу даних.

До логічних належать операції:

– порівняння: <, <=, >, >=, = =, !=;
– логічні над даними будь-якого типу: !, &&, ||;
– порозрядні (побітові): ~, &, |, ^;
– зсуву: <<, >>.

5.1. Вирази порівняння

Вирази порівняння (відношення) – це два операнди, зв’язані операцією порівняння (табл. 5.1). Операндами відношень можуть бути вирази будь-якого типу. Різні типи операндів перед виконанням операцій відношення зводяться до одного типу відповідно до ієрархії типів. Тип результату операції відношення цілий: якщо відношення істинно, то результат дорівнює 1, а якщо хибно, то – 0.

5.2. Логічні вирази для даних будь-якого типу

Для даних будь-якого типу можуть застосовуватися логічні операції: ! (НІ), && (І), || (АБО). Операндами цих логічних операцій можуть бути результати виразів будь-якого типу. Але перед виконанням логічних операцій значення результатів виразів-операндів перетвориться в логічний вираз істина – 1 (true) або хибність – 0 (false). Усі значення, відмінні від 0, інтерпретуються як істинні – 1, а якщо значення дорівнює 0, то воно хибне.

Таблиця 5.1

Операції порівняння

Операція	Призначення операції
<	Менше
<=	Менше або дорівнює
>	Більше
>=	Більше або дорівнює
= =	Дорівнює
! =	Не дорівнює

Результат логічної операції дорівнює 1, якщо логічний вираз істинний, і результат дорівнює 0, якщо логічний вираз хибний.

Логічні операції над логічними даними дають результат, наведений в табл. 5.2. Операція '!' – унарна, виконується над одним операндом (операнд-1 в табл. 5.2).

Таблиця 5.2

Таблиця істинності логічних операцій

Операнд-1	Операнд-2	Результат операції		
		!	&&	||
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

5.3. Побітові логічні операції

Логічні вирази з бітовими операндами призначені для обробки значень на машинному рівні. Звичайно побітові операції використовуються в програмах, що реалізують доступ до апаратури, для логічної обробки бітових даних, що вимагає операцій над бітами: їх виділення із значення, аналізу шляхом порівняння, заміни, зсуву і т.п.

Подібно до всіх інших, операторам відношення і логічним операторам властивий деякий рівень пріоритету, який визначає порядок обчислення цих операторів. Побітові логічні операції у порядку спадання їх пріоритету наведені в табл. 5.3.

Щоб правильно використовувати ці операції, треба знати машинне (бітове) подання оброблюваних значень.

Операнди побітових операцій повинні бути цілого типу. Результати побітових операцій подані в табл. 5.3 і 5.4.

5.3.1. Операція обернення

Оператор '~' інвертує біти значення операнда, замінюючи кожен 0 на 1, а 1 на 0. При цьому набуте значення прийнято називати доповненням до 1.

Таблиця 5.3

Побітові логічні операції

Операція	Найменування операції	Призначення операцій
~	Обернення	Формування оберненого коду числа порозрядно: 1 → 0, 0 → 1
<<, >>	Зсув уліво і вправо	Еквівалентно множенню (діленню) на 2^n, де n – кількість зсувів
&	“І”	Порозрядне логічне множення
^	“Виключаюче АБО”	Складання розрядів по модулю 2
|	“АБО”	Порозрядне логічне складання

Таблиця 5.4

Таблиця істинності побітових операцій

Операнд-1	Операнд-2	Результати операцій		
		^	&	|
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	1

5.3.2. Операції зсуву

Операції зсуву ‘<<’ і ‘>>’ виконують зсув лівого операнда вліво або вправо на кількість розрядів, задану правим операндом.

Результат невизначений, якщо правий операнд від’ємний або більше або дорівнює довжині об’єкта в бітах.

При зсуві всі біти, які виходять за край розрядності значення, втрачаються. Місце, яке звільнилося з іншої сторони, заповнюється нулями завжди при зсувах уліво і вправо знакових цілих з позитивним значенням.

Значенням x<<n є x, зсунуте вліво на n бітів. Значенням x>>n є x, зсунуте вправо на n бітових позицій.

Наприклад:

$x = 00000010_2 = 2$; $y = 00011000_2 = 24$;

$x<<1 = 00000100_2 = 4 = 2*2^1$; $y>>1 = 00001100_2 = 12 = 24/2^1$;

$x<<2 = 00001000_2 = 8 = 2*2^2$. $y>>2 = 00000110_2 = 6 = 24/2^2$.

З прикладів видно, що операції зсуву можуть застосовуватися для ділення або множення цілого значення на число, що дорівнює степені 2, тобто:

x<<n – еквівалентно множенню x на 2^n.

x>>n – еквівалентно діленню x на 2^n.

5.3.3. Виключаюче “АБО” (порозрядне складання по модулю 2)

Оператор ‘^’ – це порозрядне складання по модулю (основі системи числення) 2. Цю операцію використовують, якщо як результат треба зберегти лише ті розряди, в яких є тільки одна із складових одиниць операндів.

Наприклад, x = 000011, y = 010110, тоді:

```
  000011 – перший доданок – значення x;
^ 010110 – другий доданок – значення y;
  010101 – результат операції ‘^’.
```

При чому x ^ x = 0.

Ця операція використовується також для перемикання значень деяких бітів на протилежні. Для установки використовується вага розряду біта, який перемикається.

Наприклад, необхідно перемкнути значення третього біта на протилежне. Вага третього біта 2^2=4. Тоді при виконанні операції “виключаюче АБО” третій біт буде перемкнутий на протилежне значення.

Наприклад:

```
  000010     000110
^ 000100   ^ 000100
  000110     000010
```

5.3.4. Побітове “АБО”

Оператор ‘|’ використовують, якщо треба сформувати значення результату як сукупності всіх двійкових одиниць операндів.

Наприклад, x = 001001, y = 000010.

Тоді:

```
  001001 – перший операнд – значення x;
| 000010 – другий операнд – значення y;
  001011 – результат операції ‘|’.
```

Ця операція використовується також для установки окремих бітів в одиницю. Для установки застосовується вага розряду, що перевіряється.

Наприклад, необхідно встановити в 1 п’ятий біт змінної. Вага п’ятого біта 2^4=16. Тоді при виконанні операції побітове “АБО” п’ятий біт буде встановлений в 1, незалежно від того, яке він мав значення до операції:

```
  000010     010010
| 010000   | 010000
  010010     010010
```

5.3.5. Побітове “І”

Оператор ‘&’ використовується при перевірці одиничних бітів і установки нульових значень бітів. Для перевірки одиничного значення необхідно знати вагу біта, що перевіряється.

За допомогою операції ‘&’ можна виділити (замаскувати) значення необхідних розрядів для подальшої їх обробки. Для виділення розрядів треба використовувати одиниці, для їх маскування – нулі.

Наприклад, x=016_8 (001110_2). Для виділення трьох молодших розрядів можна використовувати такий вираз: x & 07_8. Результатом буде 06_8. У двійковій СЧс:

001110 – перший операнд, значення змінної $x = 016_8$;
& 000111 – другий операнд, константа виділення значення 07_8;
000110 – результат операції & = 06_8.

5.4. Обчислення за скороченою схемою

Припустімо, що компілятору зустрівся такий вираз:

(x == 5) && (y == 3);

У цьому разі компілятор спочатку оцінить перший вираз (x == 5) і, якщо він поверне false, то не стане обчислювати другий (y == 3), оскільки для істинності всього виразу з оператором “І” потрібне, щоб обидві його складові були істинними.

Аналогічно, якщо компілятору зустрінеться вираз з оператором “АБО”:

(x == 5) | | (y == 3);

і перший вираз виявиться істинним (x == 5), то компілятор не стане обчислювати другий вираз (y == 3), оскільки йому достатньо одного дійсного результату, щоб визнати істинним увесь вираз.

5.5. Форматований вивід даних

З оператором *cout* можна використовувати маніпулятори. Маніпулятори – це спеціальні функції, які дозволяють змінювати стан потоку. Для використання більшості маніпуляторів потрібно включати в програму файл *iomanip.h.*

Далі наведений список маніпуляторів, що не вимагають включення в програму файлу *iomanip.h.*

– *flush* – очищає буфер виводу;
– *endl* – вставляє символ нового рядка й очищає буфер виводу;
– *oct* – встановлює вісімкову основу для чисел, що виводяться;
– *dec* – встановлює десяткову основу для чисел, що виводяться;
– *hex* – встановлює шістнадцяткову основу для чисел, що виводяться.

Нижче наведений набір маніпуляторів, для яких необхідно включати файл *iomanip.h:*

– *setw (ширина)* – встановлює мінімальну ширину поля виводу;
– *setfill (символ)* – встановлює символ заповнення незайнятих позицій поля виводу;
– *setprecision (точність)* – встановлює кількість знаків після плаваючої коми;
– *setiosflags (прапор)* – встановлює один або декілька прапорів *ios*;
– *resetiosflags (прапор)* – скидає один або декілька прапорів *ios.*

Наприклад:

cout<<setw(16)<<setfill(‘#’)<<hex<<x<<endl;

– встановлюється ширина поля виводу в 16 знаків, символ заповнення незайнятих позицій поля виводу ‘#’ та шістнадцяткова основа чисел, що виводяться, після чого виводиться значення змінної x, додається символ нового рядка й очищається буфер виводу.

Всі оператори, за винятком *flush*, *endl, setw,* залишаються включеними впродовж усієї роботи програми, якщо, звичайно, не будуть зроблені інші установки. Установка маніпулятора *setw* відміняється відразу ж після виводу із застосуванням об’єкта *cout.*

Нижче наведені прапори, вживані спільно з маніпуляторами *setiosflags* і *resetiosflags:*

– *ios::left* – вирівнює дані по лівому краю поля виводу;

– *ios::right* – вирівнює дані по правому краю поля виводу;

– *ios::internal* – вирівнює дані по ширині поля виводу;

– *ios::dec* – виводить дані в десятковому форматі;

– *ios::oct* – виводить дані у вісімковому форматі;

– *ios::hex* – виводить дані в шістнадцятковому форматі;

– *ios::showbase* – додає префікс 0x до шістнадцяткових значень і 0 до вісімкових;

– *ios::showpoint* – заповнює нулями відсутні знаки в значеннях заданої довжини;

– *ios::uppercase* – відображає шістнадцяткові та експоненціальні значення у верхньому регістрі;

– *ios::showpos* – додає знак ‘+’ перед додатними числами;

– *ios::scientific* – відображає числа з плаваючою комою в експоненціальному вигляді.

Наприклад:

cout<<setiosflags(ios::left | ios::hex) – встановлюється прапор ios::left, що приводить до вирівнювання даних, що виводяться, по лівому краю поля виводу, і прапор ios::hex, що приводить до виводу даних у шістнадцятковому вигляді.

Додаткову інформацію можна знайти у файлі *iostream.h* або в довідковій системі Visual Studio.

Приклад 5.1. Над цілими змінними *x, y* виконати порозрядне логічне додавання по модулю 2. Результат перевести в шістнадцяткову СЧс.

```
#include<iostream>
using namespace std;
int main( )
{
  int x, y;
      cout<<"Enter 2 integer numbers\n";
      cin>>x>>y;
      cout<<"x^y="<<(x^y)<<'\n';
      cout<<"in hex:\n";
      cout<<"x^y="<<hex<<(x^y)<<'\n';
```

```
}
```

Приклад 5.2. Знайти значення виразу: (x != 4) | | !(y <= 15) && (z= =y).

```
#include<iostream>
using namespace std;
int main( )
{
  int x, y, z;
       cout<<"Enter 3 integer numbers (x, y, z)\n";
       cin>>x>>y>>z;
       cout<<"(x != 4) | | !(y <= 15) && (z= =y)  =  "
            << ((x!=4)||!(y<=15)&&(z==y))<<'\n';
}
```

Висновки

У даному розділі розглянуті наведені нижче основні питання:

- визначення виразів;
- побітові логічні операції;
- обчислення за скороченою схемою;
- форматований вивід даних.

Контрольні питання

1. У яких випадках використовуються логічні операції?
2. У яких випадках використовуються побітові логічні операції?
3. Якого типу формується результат логічної операції і чим він визначається?
4. Які логічні операції використовуються в мові С++?
5. Яким чином відбувається обчислення за скороченою схемою?
6. Істинним або хибним є значення -1?
7. Чи істинно таке твердження: операції відношення мають вищий пріоритет, ніж арифметичні операції?
8. Яка різниця між виразами *x=3* і *x = =3*?
9. Чи рівнозначні в мові С++ вирази !!x і x?
10. Які логічні вирази відповідають таким умовам:
    - weight більше або дорівнює 115, але менше 125;
    - *x* є парним, але не 26;
    - *x* знаходиться в діапазоні 1000 – 2000 або *y* є 1?

Напишіть їх.

6. РОЗГАЛУЖЕННЯ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Навчальною метою розділу є ознайомлення студентів з розгалуженням обчислювальних процесів.

У результаті вивчення даного розділу студенти повинні знати:

- формат конструкції *if-else*;
- формат конструкції *switch*;
- формат оператора *break*;

Звичайно всі оператори у програмі виконуються один за одним, але якщо необхідно змінювати порядок виконання операторів, використовують конструкції розгалуження та оператори безумовного переходу. У мові С++ існує дві конструкції розгалуження:

- if-else;
- switch,

а також оператори безумовного переходу:

- break;
- continue.

6.1. Конструкція if-else

Конструкція if-else використовується у випадках, коли необхідно виконувати різні дії залежно від умови.

Формат конструкції:

if (вираз) оператор-1;
[else оператор-2;]

Виконання конструкції *if* починається з обчислення *виразу*, далі здійснюється за такою схемою:

– якщо *вираз* істинний (тобто відмінний від 0), то виконується *оператор-1*;

– якщо *вираз* хибний (тобто дорівнює 0), то виконується *оператор-2*;

– якщо *вираз* хибний і відсутній *оператор-2* (у квадратні дужки поміщена необов'язкова конструкція), то виконується наступний за *if* оператор.

Приклад:

```
if (i < j)   i++;
else { j = i - 3;   i++; }
```

Допускається використання вкладених конструкцій *if*. Конструкція *if* може бути включена в конструкцію *if* або в конструкцію *else* іншої конструкції *if*. Щоб зробити програму більш читабельною, рекомендується групувати оператори і конструкції у вкладених конструкціях if, використовуючи фігурні дужки. Якщо ж фігурні дужки опущені, то компілятор пов'язує кожне ключове слово else з найбільш близьким if, для якого немає else.

Наприклад:

```
int t=2, b=7, r=3;
 if (t>b)
{     if (b < r)
            r=b;
  }
else r=t;
```

У результаті виконання цієї програми *r* буде дорівнювати 2.

Якщо ж у програмі опустити фігурні дужки, що стоять після конструкції *if*, то програма матиме такий вигляд:

```
int t=2,b=7,r=3;
if ( t>b )
 if ( b < r )     t=b;
        else    r=t;
```

У цьому випадку *r* набуде значення, що дорівнює 3, оскільки ключове слово *else* належить до іншої конструкції if*,* яка не виконується, тому що не виконується умова, що перевіряється у першої конструкції if.

6.2. Конструкція switch

Конструкція *switch* призначена для організації вибору з безлічі різних варіантів. Формат конструкції такий:

```
switch ( вираз )
{
[ case константний вираз 1]: [ список операторів 1]
[ case константний вираз 2]: [ список операторів 2]
                :
[ default: [ список операторів ]]
}
```

Вираз, що йде за ключовим словом *switch* в круглих дужках, повинен бути цілого типу або типу покажчика. Значення цього виразу є ключовим для вибору з декількох варіантів. Тіло конструкції *switch* складається з декількох *операторів*, позначених ключовим словом *case* з подальшим *константним виразом*. Як константний вираз використовуються цілі або символьні константи.

Усі *константні вирази* в конструкції *switch* повинні бути унікальні. Окрім *операторів*, позначених ключовим словом *case,* можливий, але обов'язково один, фрагмент, позначених ключовим словом *default.*

Список операторів може бути порожнім або містити один або більш операторів. Причому в конструкції *switch* не потрібно брати послідовність операторів у фігурні дужки.

Схема виконання конструкції *switch* така:

– обчислюється вираз у круглих дужках;

– обчислене значення послідовно порівнюються з константними виразами, що йдуть за ключовими словами case;

– якщо один з константних виразів збігається із значенням виразу, то керування передається на оператора, позначеного відповідним ключовим словом case;

– якщо жоден з константних виразів не дорівнює виразу, то керування передається на оператора, позначеного ключовим словом *default*, а у разі його відсутності керування передається на наступного після *switch* оператора.

Для негайного виходу з *case*, якщо це необхідно, використовується оператор *break*.

Для того, щоб виконати однакові дії для різних значень виразу, можна позначити одного і того самого оператора декількома ключовими словами *case*.

Наприклад:

```
int i=2;
switch (i)
{
      case 1: i += 2;
      case 2: i *= 3;
      case 0: i /= 2;
      case 4: i -= 5;
      default:  ;
}
```

Виконання конструкції *switch* починається з оператора, позначених *case 2*. Таким чином, змінна *i* набуває значення, що дорівнює 6, далі виконується оператор, позначений ключовим словом *case 0*, а потім *case* 4, змінна *i* набуде значення 3, а потім значення -2. Оператор, позначений ключовим словом *default*, не змінює значення змінної.

6.3. Оператор break

Оператор *break* забезпечує припинення виконання самої внутрішньої з обхоплюючих його конструкцій *switch, do-while, for, while*. Після виконання оператора *break* керування передається оператору, що йде за перерваною конструкцією.

Наприклад:

```
int i=2;
switch (i)
{ case 1: i += 2;
        case 2: i *= 3; break;
        case 0: i /= 2;
        case 4: i -= 5;
        default:     ;
}
```

У цьому випадку виконання конструкції *switch* починається з оператора, позначеного *case 2*, змінна *i* набуває значення, що дорівнює 6, далі виконується оператор *break* і здійснюється вихід зі *switch.*

Приклад 6.1. Написати програму, яка виводить на екран лінію із символів. Користувач вказує після запитів програми, який символ і яку лінію (вертикальну або горизонтальну) виводити на екран.

Схема програми для цього прикладу матиме вигляд, як показано на рис. 6.1.

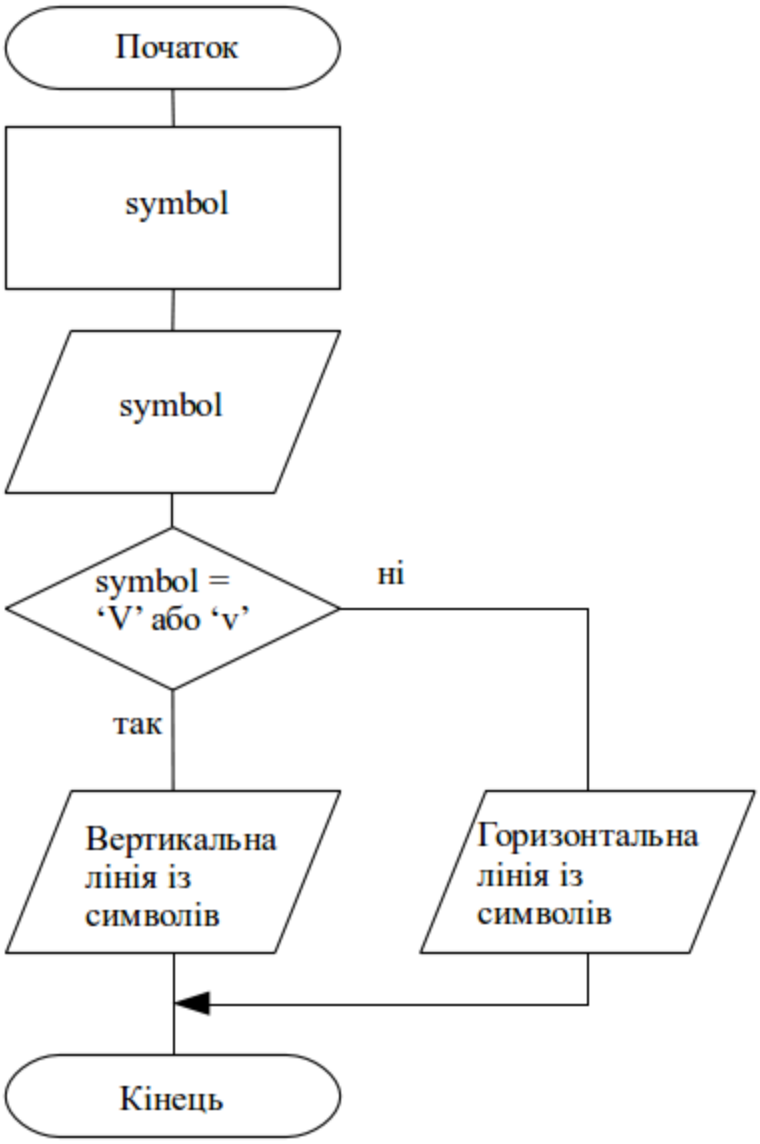


Рис. 6.1. Схема програми прикладу

```
#include<iostream>
using namespace std;
int main( )
{
  cout<<"Enter symbol\n";
  char symbol;
  cin>>symbol;
  cout<<"Enter v (vertical) or h (horizontal) line\n";
  char type;
  cin>>type;
  if(type == 'v' || type == 'V')
      cout<<symbol<<'\n'<<symbol<<'\n'<<symbol<<'\n'<<symbol<<'\n';
  else
       cout<<symbol<<symbol<<symbol<<symbol<<'\n';
}
```

Приклад 6.2. Скласти програму, що залежно від порядкового номера місяця (1, 2, ..., 12), який указує користувач, виводить на екран його назву (січень, лютий, ... грудень).

Схема програми для цього прикладу матиме вигляд, як показано на рис. 6.2.

```
#include<iostream>
using namespace std;
int main( )
{
  cout<<”Enter number\n”;
  int number;
  cin>>number;
  switch(number)
       {
            case 1: cout<<”January\n”; break;
            case 2: cout<<”Febuary\n”; break;
            case 3: cout<<”March\n”; break;
            case 4: cout<<”April\n”; break;
            case 5: cout<<”May\n”; break;
            case 6: cout<<”June\n”; break;
            case 7: cout<<”July\n”; break;
            case 8: cout<<”August\n”; break;
            case 9: cout<<” September \n”; break;
            case 10: cout<<”October \n”; break;
            case 11: cout<<”November\n”; break;
            case 12: cout<<”December\n”; break;
       }
  }
```

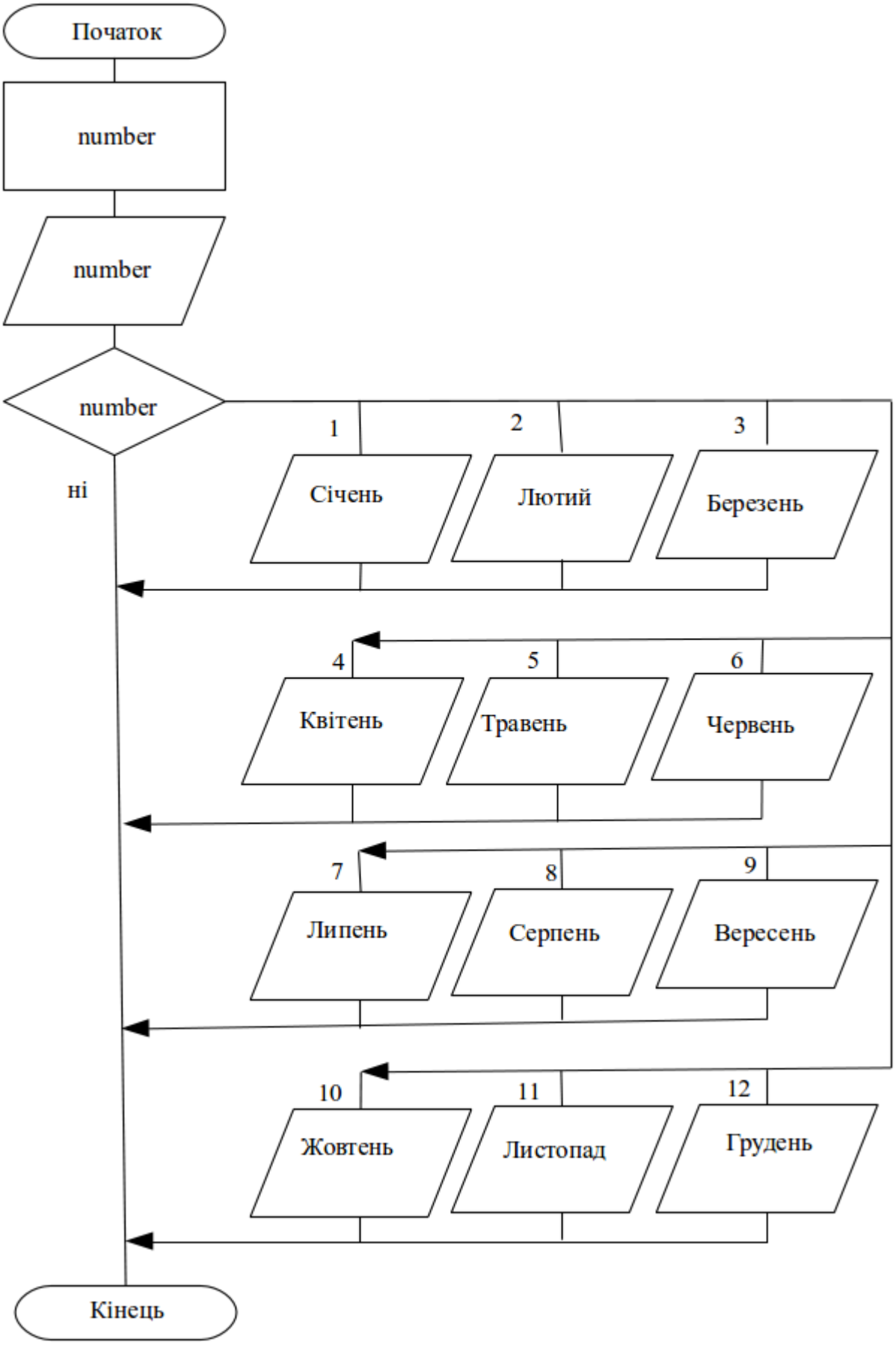


Рис. 6.2. Схема програми прикладу

6.4. Виявлення логічних помилок у середовищі Microsoft Visual Studio

Оскільки під час використання конструкції розгалуження задається не послідовний порядок виконання операторів, з'являється вірогідність появи логічних помилок у програмі. Такі помилки виявити набагато важче. Зазвичай вони виявляються у ході виконання програми.

Відладкою програми називають процес покрокового виконання програми та пошуку в ній логічних помилок.

При наявності логічних помилок програма працює, але поводиться якось дивно, припустімо, видає неправильний результат. Ось тут і виникає ряд специфічних завдань, наприклад, зупинити виконання програми і подивитися зміст змінних, проглянути програму в покроковому режимі й т. ін.

При розмові про відлагодження за допомогою Microsoft Visual Studio потрібно чітко розуміти, що є дві версії зібраного проекту: налагоджувальна (debug) і кінцева (release). Налагоджувальна версія характеризується великим розміром, який може в десятки разів перевищувати розмір кінцевої версії.

Це пов’язано з тим, що в зібраний проект поміщається вся необхідна інформація для відладчика (таблиці символів і т. д.).

Крім того, при складанні налагоджувального проекту не використовується будь-яка оптимізація, саме з цим пов’язано багато проблем отримання кінцевої версії, коли налагоджувальна працює, а кінцева ні. Це частково є наслідком застосування оптимізації. Але у будь-який момент можливо перемикання з кінцевої версії на налагоджувальну для пошуку помилки.

За цей процес відповідає пункт меню *Build/Configuration Manager*. За умовчанням проект створюється в налагоджувальній версії. Після його створення в теці проекту з’являється відповідна тека з результатами створення, в якій і лежатиме готовий exe-файл.

Імена відповідають іменам конфігурації (рис. 6.3).

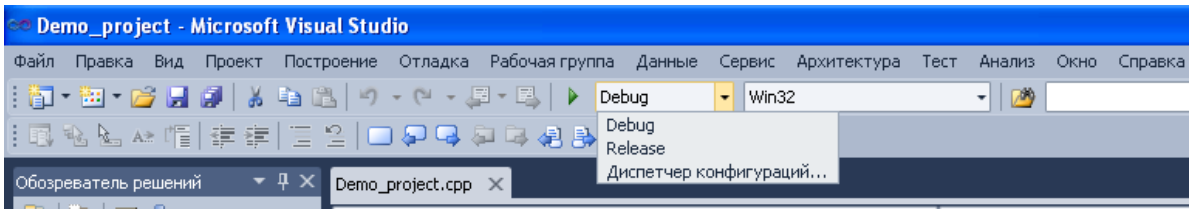


Рис. 6.3. Вибір версії проекту

Панель інструментів відлагодження можна викликати через контекстне меню: правою кнопкою миші клацніть в області головного меню програми та виберіть меню *Debug (Отладка)* (рис. 6.4).

Після цього з’явиться панель відлагодження (рис. 6.5).

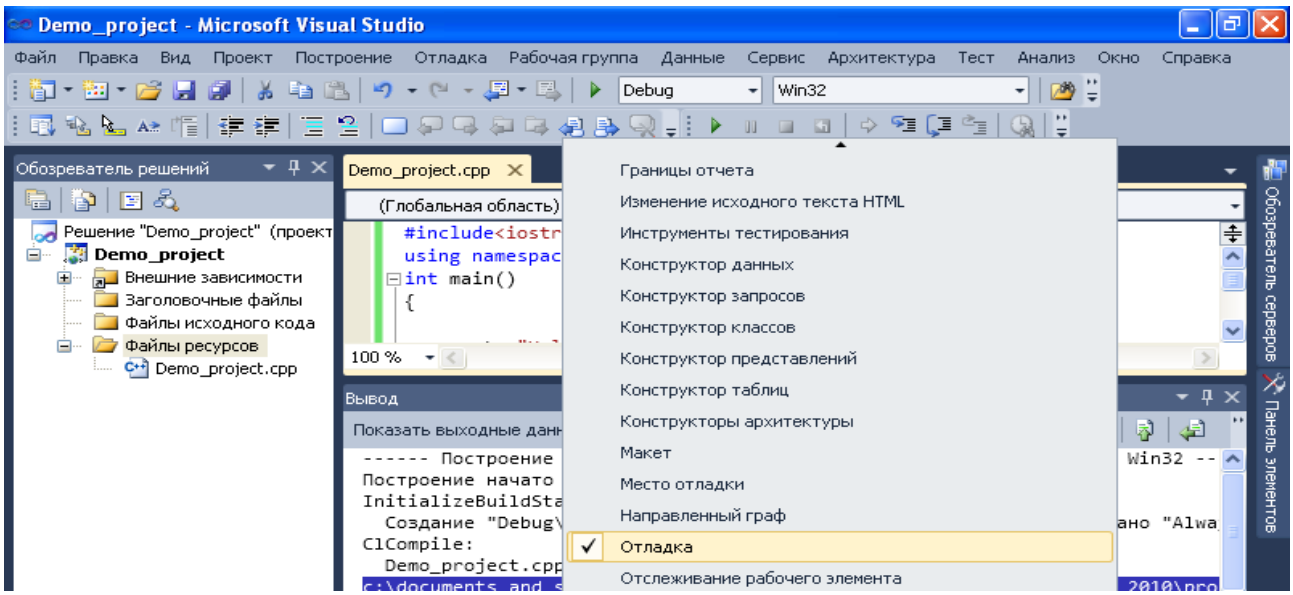


Рис. 6.4. Вибір панелі інструментів Debug

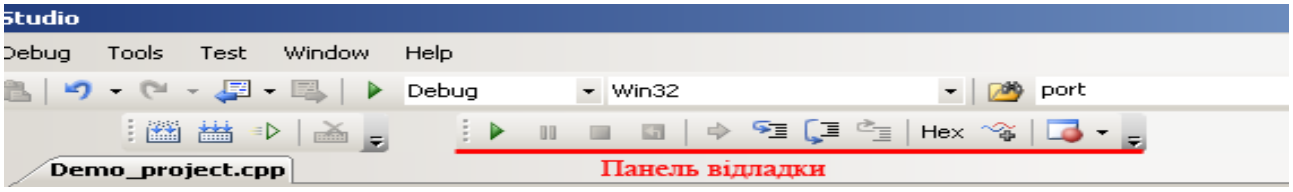


Рис. 6.5. Панель інструментів Debug

Для того, щоб запустити програму в режимі налагодження, необхідно вибрати команду *Debug/Start Debugging* або натиснути клавішу *F5*.

У будь-який момент можна зупинити налагоджувану програму, обравши команду *Debug/Stop Debugging* або *Shift+F5*.

Існують так звані точки зупинки.

Вони потрібні для того, щоб зупинити виконання програми в заданому місці й далі виконувати програму покроково або до наступної точки зупинки.

Щоб установити точку зупинки, необхідно вибрати місце зупинки і натиснути клавішу *F9*.

Покрокове виконання програми буває двох типів:

– із заходом у функції – команда *Debug/Step Into* або *F11*;

– без заходу у функції – команда *Debug/Step Over* або *F10*.

Для проглядання значень поточних змінних можна відкрити вікно *Locals* (меню *Debug/Windows/Locals*). Змінні, які були змінені, підсвічуються червоним кольором. У вікні *Locals (Локальные)* відображаються тільки ті змінні, які використовуються всередині функції (рис. 6.6).

Для швидкого перегляду значення змінної існує два способи.

Перший – підвести курсор на середину імені змінної.

Другий – клацнути на імені змінної правою кнопкою миші та викликати з контекстного меню команду *Add Watch (Контрольные значения)*.

Таким чином, змінна буде додана у вікно перегляду змінних *Watch (Контрольные значения)* (рис. 6.7) та з'явиться можливість слідкувати за зміненням її значення у будь-який момент.

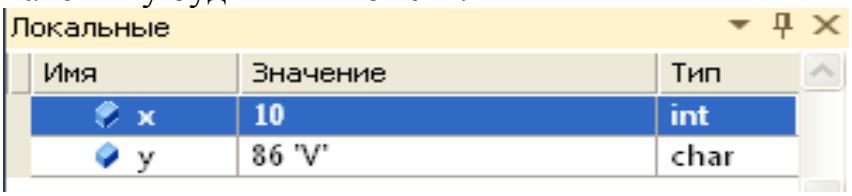


Рис. 6.6. Вікно Locals

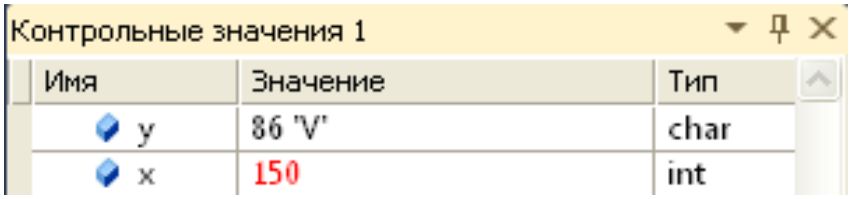


Рис. 6.7. Вікно Watch

Тут можна писати вирази й обчислювати їх значення.

Наприклад, можна написати *type+4* і натиснути *Enter*, у результаті буде розраховане значення.

У ході налагодження можна побачити, що значення змінної явно не правильне, а бажання піти далі є. Виправити значення змінної можна прямо у вікні перегляду *Watch*. Для цього двічі клацаємо правою кнопкою миші на значенні змінної. У полі *Value (Значение)* з'явиться курсор і далі можна правити. Після закінчення редагування потрібно натиснути *Enter*.

Висновки

У даному розділі розглянуті наведені нижче основні питання:

– організація обчислювальних процесів, що розгалужуються;
– оператори, що застосовуються в таких процесах;
– необхідність та процедура відладки програми.

Контрольні питання

1. Який формат запису має конструкція *if-else* і яке його призначення?
2. Як відбувається виконання конструкції *if-else*?
3. Який формат запису має конструкція switch і яке його призначення?
4. Як відбувається виконання конструкції *switch*?
5. Яке призначення має оператор *break*?
6. Як сформувати розгалуження *if-else*, що друкує слово „*Yes*" у випадку, якщо значення змінної *age* більше ніж 21? Напишіть його.
7. Яке розгалуження *if-else* друкує слово „*Yes*" у випадку, якщо значення змінної *age* більше ніж 21, і слово „*No*" у решті випадків? Напишіть його.
8. Яке розгалуження *switch* друкує слово „*Yes*" у випадку, якщо значення змінної *ch* дорівнює '*y*', і слово „*No*" у випадку, якщо „*ch*" дорівнює '*n*', а також „*unknown*" – у решті випадків? Напишіть його.

7. ЦИКЛІЧНІ ОБЧИСЛЮВАЛЬНІ ПРОЦЕСИ

Навчальною метою розділу є ознайомлення студентів з циклічними обчислювальними процесами.

У результаті вивчення даного розділу студенти повинні вміти:

- застосовувати конструкції для організації циклів;

знати:

- формати конструкцій *for, while*, *do-while*.

7.1. Види конструкцій для організації циклів

При виконанні програми нерідко виникає необхідність неодноразового виконання однотипних обчислень. Для цього використовуються так звані цикли.

Цикл є ділянкою програми, в якій однакові обчислення виконуються неодноразово для різних значень одних і тих самих змінних (об'єктів).

Для організації циклів в С++ використовуються такі три конструкції: *for, while, do-while.*

7.1.1. Конструкція *for*

Конструкції *for* є циклом з параметрами і звичайно використовується у разі, коли відома точна кількість повторів обчислень. При цьому здійснюються три операції: ініціалізація лічильника циклу, порівняння його величини з деяким граничним значенням, зміна значення лічильника при кожному проходженні тіла конструкції.

Конструкції *for* має таку форму запису:

for (ініціалізація циклу; умова продовження циклу; оновлення змінної циклу)
тіло циклу;

Ініціалізація циклу виконується тільки один раз. Як правило, цей вираз застосовується для завдання початкового значення змінної циклу, після чого ця змінна може використовуватися для підрахунку кількості *ітерацій* (перевірка умови і тіло циклу) циклу (в цьому випадку її називають *лічильником циклу*).

Умова продовження циклу визначає, чи слід завершити виконання циклу. Якщо результат перевірки умови продовження циклу істинний (тобто відмінний від 0), тоді тіло циклу виконується.

Конструкції *for* є циклом з вхідною умовою (або з передумовою). Це означає, що умова продовження циклу перевіряється перед виконанням кожної ітерації циклу. При цьому тіло циклу взагалі не виконується, якщо не виконується умова продовження циклу (рис. 7.1).

Оновлення змінної циклу відбувається в кінці циклу після виконання його тіла. Звичайно це приводить до зменшення або збільшення значення змінної циклу на величину, яка називається *кроком циклу.*

Тіло циклу може складатися з одного оператора або сукупності операторів, поміщених у блок з використанням фігурних дужок {}.

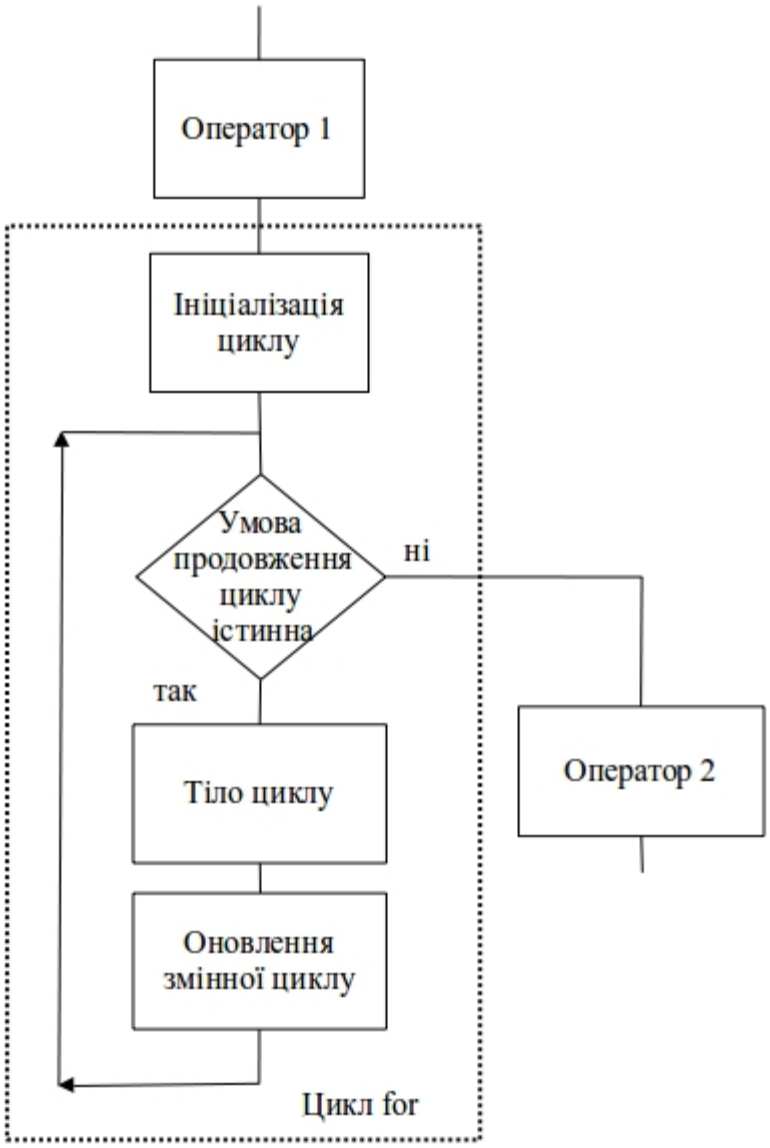


Рис. 7.1. Принцип роботи конструкції циклу for

Як вирази, що керують циклом, може використовуватися будь-який дійсний вираз C++.

7.1.2. Конструкція *while*

Конструкція *while* є циклом з передумовою. Вона використовується у разі, коли, по-перше, невідома точна кількість повторів і, по-друге, при цьому немає необхідності, щоб цикл був виконаний хоча б один раз. Конструкція *while* має таку форму запису:

while (умова продовження циклу)
тіло циклу;

Якщо результат перевірки *умови продовження циклу* істинний (тобто відмінний від 0), тоді *тіло циклу* виконується один раз, а потім *умова продовження циклу* перевіряється наново. Ітерації здійснюються доти, поки *умова продовження циклу* не стане хибною.

При організації конструкції *while* в його тіло повинні бути включені оператори, що змінюють умову продовження циклу так, щоб врешті-решт вона стала хибною. Інакше виконання циклу ніколи не закінчиться.

7.1.3. Конструкція *do-while*

Конструкція *do-while* є циклом з постумовою і використовується в тих випадках, коли не відома точна кількість повторів, але в той же час цикл необхідно виконати щонайменше один раз.

Конструкція *do-while* дуже схожа на конструкцію *while.* Різниця полягає в тому, що перевірка істинності умови продовження циклу в конструкції *do-while* відбувається після виконання тіла циклу.

Конструкція *do-while* має таку форму запису:

do

тіло циклу;

while (умова продовження циклу);

7.2. Оператори, які керують у конструкціях

У конструкціях усіх трьох типів може використовуватися оператор *break.* Виконання оператора *break* приводить до виходу з циклу, в якому він міститься, і переходу до наступного за циклом оператора. Якщо оператор *break* знаходиться всередині вкладених циклів, то його дія розповсюджується тільки на той цикл, безпосередньо в якому він знаходиться.

Оператор *continue* може використовуватися тільки серед операторів тіла циклу. Цей оператор викликає пропускання частини ітерації, що залишилася, усередині циклу і перехід до наступної ітерації.

Приклад 7.1. Увести з клавіатури число і перевернути його «фізично» (наприклад, число 2356 стане числом 6532).

Схема програми для цього прикладу матиме вигляд, як показано на рис. 7.2.

Програма

```
#include<iostream>
using namespace std;
int main( )
{
    float number,reverse=0;
    do
    {
        cout<<"Enter number (0 to 2147483647)\n";
        cin>>number;
    }while(number<0 || number>2147483647);
    while(number>0)
    {
        reverse*=10;
```

```
        reverse+=(int)number%10;
        number=(int)number/10;
    }
    cout<<"reverse number: "<<reverse<<'\n';
    return 0;
}
```

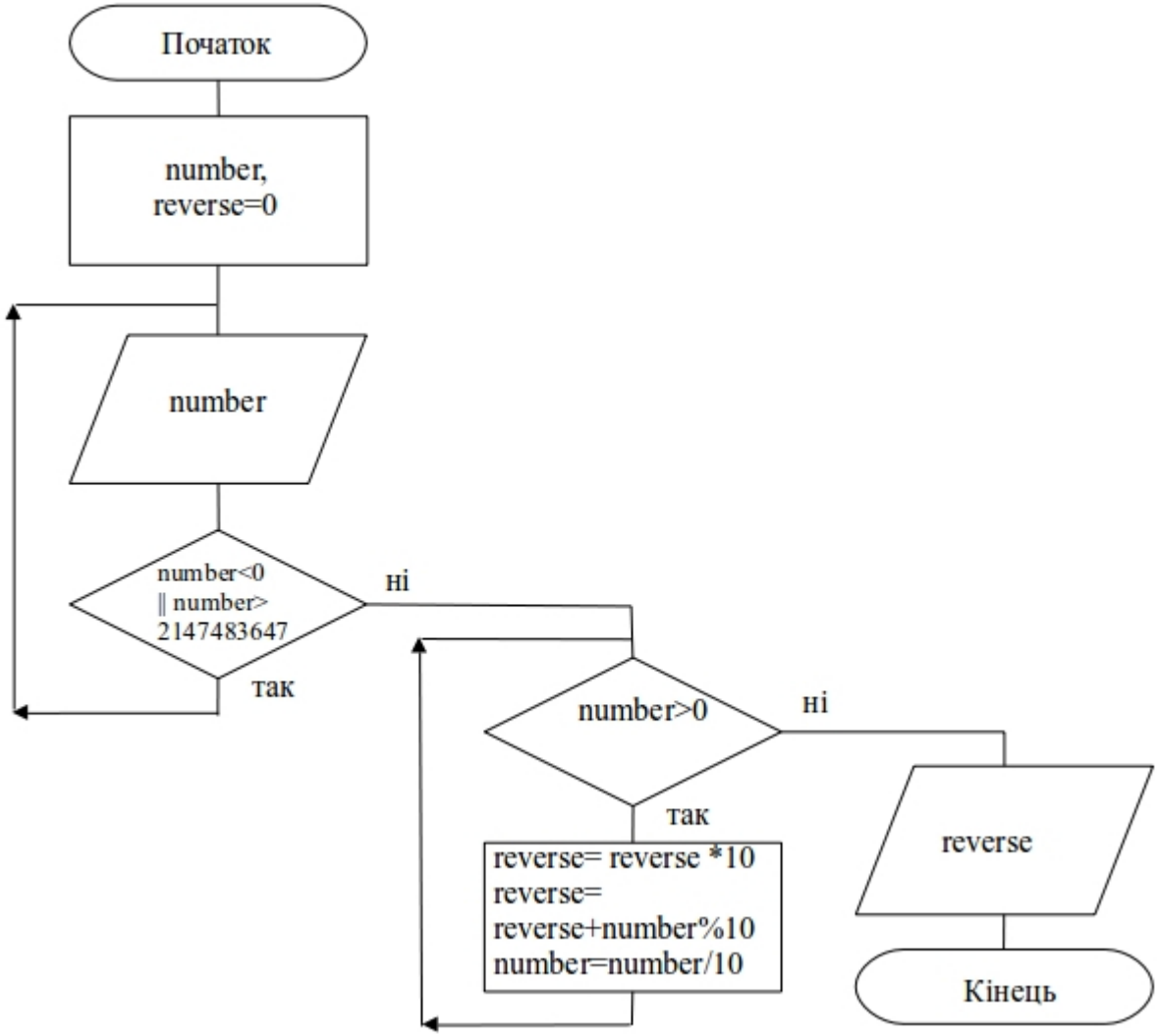


Рис. 7.2. Схема програми прикладу

Висновки

У даному розділі розглянуті наведені нижче основні питання:

– організація циклічних обчислювальних процесів;

– конструкції *for, while*, *do-while*, що застосовуються в обчислювальних процесах;

– оператори, які керують у конструкціях.

Контрольні питання

1. Яке основне призначення кожного з трьох виразів, що входять до складу конструкції *for*?

2. Яка конструкція виводитиме на екран числа від 100 до 110? Створіть її.

3. Скільки разів виконується тіло конструкції *do-while* для організації циклу?

4. У чому відмінність конструкції з передумовою від конструкції для організації циклу з постумовою? До якого з вказаних видів можна віднести кожну конструкцію для організації циклів у мові С++?

5. Що виводить на друк фрагмент коду, що наведений нижче?

```
for( int j=0; j<11; j+=3)
    cout<<j;
    cout<<”\n”<<j<<”\n”;
```

6. Що виводить на друк фрагмент коду, що наведений нижче?

```
int j=5;
while( ++j < 9 )
cout<<j++<<”\n”;
```

7. Що виводить на друк фрагмент коду, що наведений нижче?

```
int j=8;
do
cout<<” j = ”<<j<<”\n”;
while(j ++ < 5 )
```

8. Яким чином для організації циклу можна включити в тіло конструкції декілька операторів?

9. Яка конструкція *for* виведе на друк значення 1, 2, 4, 8, 16, 32, 64 при збільшенні на 2 значення лічильника на кожному кроці циклу? Створіть конструкцію.

8. МАСИВИ

Навчальною метою розділу є ознайомлення студентів з масивами мови програмування С++.

У результаті вивчення даного розділу студенти повинні знати:

– визначення масиву;

– визначення двовимірного масиву;

уміти:

– використовувати одно- та двовимірні масиви.

8.1. Оголошення та ініціалізація одновимірного масиву

Часто при розробці програми з'являється необхідність зберігати та обробляти дані одного типу та призначення.

Наприклад, треба зберігати дані про температуру зовнішнього середовища, що вимірювалась протягом місяця. Щоб мати можливість обробляти ці дані, треба зберегти їх у змінних. Так, для 30 даних необхідно 30 змінних. Наявність великої кількості змінних ускладнює обробку даних, тим більше що з'являється необхідність повторювати ті самі операції з усіма змінними. Наприклад, якщо необхідно просто вивести дані температури на екран.

Для того, щоб спростити обробку однотипних даних, існує такий тип змінної, як масив.

Масив – це множина елементів одного типу даних. Масив, як і звичайна змінна, перед використанням повинен бути оголошений. Оголошення масиву складається із специфікації типу, ідентифікатора і розмірності (в квадратних дужках [ ]). Наприклад, оператор

```
int A[10];
```

описує масив з іменем А, що складається з 10 об'єктів типу int.

Кожен об'єкт називається *елементом* масиву А. Імен у цих об'єктів немає, але кожний з них має свій порядковий номер. Доступ до елементів здійснюється відповідно до його положенню в масиві, при цьому вказується ім'я масиву та номер елементу в квадратних дужках []. Ця форма доступу називається *індексацією*.

Доступ до елементу масиву має вигляд:

A[2]= 5; – записуємо число 5, як 3-й елемент масиву.

Одна з основних особливостей масивів у мові С++ пов'язана з тим, що елементи масиву нумеруються, починаючи з 0. Для масиву з 10 елементів індекс набуває значення від 0 до 9, а не від 1 до 10.

Таким чином, А[2] дає нам насправді третій елемент масиву А, а для масиву з 10 елементів звертання А[10] буде помилкою (11-й елемент), про яку транслятор нічого не скаже, а всі наслідки виявляться при запуску і можуть бути дуже неприємними. Тому треба добре запам'ятати, що індексація масивів іде від 0.

Розмірність описує кількість елементів у масиві і вказується в квадратних дужках.

Розмірність масиву повинна бути більше або дорівнювати 1. Значення розмірності має бути константним виразом, тобто вона повинна бути відома на етапі трансляції. Це означає, що змінна не може використовуватися для завдання розмірності масиву.

Так, задавати розмірність масиву можна таким чином:

```
const int buf_size = 512;
char buffer[buf_size];
```

А так не можна:

```
int buf_size = 512;
char buffer[buf_size];
```

Масив, як і змінну, можна ініціалізувати (задавати значення елементам масиву) при оголошенні або в ході виконання програми.

Масив можна явно ініціалізувати при визначенні за допомогою списку значень, розділеного комами. Список беруть у фігурні дужки. Після таких фігурних дужок ставиться крапка з комою (це визначення змінної).

```
int A[3]= { 0, 1, 2 };
```

При явній ініціалізації масиву немає необхідності указувати його розмірність. Транслятор визначить її за кількістю значень у списку ініціалізації:

```
int A[] = { 0, 1, 2 };  // масив з розмірністю 3
```

Якщо розмірність задана, то кількість значень у списку ініціалізації не повинна її перевищувати, бо

```
int A[2]= {0, 1, 2};
```

приведе до помилки трансляції.

Якщо розмірність масиву більша за кількість значень у списку, то елементи масиву, що не ініціалізували явно, будуть встановлені в 0:

```
int A[5]= {0, 1, 2};  // A буде дорівнювати { 0, 1, 2, 0, 0 }
```

Масив символів (масив типу char) можна ініціалізувати як списком з окремих символьних констант, так і рядком символів. При цьому у разі ініціалізації рядком символів масив одержить у кінці і нульовий символ, що обмежує рядок:

```
char ca1[] = { 'C', '+', '+' };
char ca2[] = „C++";
```

Масив ca1 буде з розмірністю 3, а ca2 – з розмірністю 4 (і містити { 'C', '+', '+', '\0' }).

У цьому прикладі з масивом ca3 буде видана помилка:

```
char ca3[3] = „C++";   // потрібен масив з 4 елементів
```

Працюючи з масивами, майже завжди використовують цикл для обробки елементів масиву.

Ось правильний цикл для обходу та ініціалізації масиву:

```
const int a_size = 10;
int A[a_size];
for (int i = 0; i < a_size ; i++)
        A[i]= i;
```

Один масив не може ініціалізуватися іншим масивом і не може бути привласнений іншому масиву.

```
int a1[] = {0, 1, 2};
int a2[] = a1;   // помилка
int a3[3];
a3 = a1;     // помилка
```

Щоб скопіювати один масив в інший, необхідно скопіювати кожен елемент по черзі:

```
const int a_size = 7;
int a1[] = {0, 1, 2, 3, 4, 5, 6};
int a2[a_size];
for(int i = 0; i < a_size; i++)
a2[i] = a1[i];
```

Як індекс масиву (в прикладі змінна i) може використовуватися будь-який вираз, який зводиться до цілочислового значення.

У C++ немає контролю виходу індексу за межі. Вся відповідальність покладена на програміста.

Приклад 8.1. В одновимірному масиві, що складається з 20 дійсних чисел, обчислити добуток елементів, що знаходяться між мінімальним (min) і максимальним (max) елементами.

Схема програми прикладу наведена на рис. 8.1.

Програма

```
#include<iostream>
using namespace std;
int main( )
{
    const int size=20; // розмір масиву
    float A[size];      // оголошуємо масив розміром 20 елементів
    cout<<"Enter 20 elements of array\n";
    for(int i=0; i<size; i++)  // введення елементів масиву з клавіатури
    {
        cout<<"A["<<i<<"]=";
        cin>>A[i];
    }
    cout<<"Your array\n";
    for(int i=0; i<size; i++)     // вивід масиву на екран
    {
        cout<<"A["<<i<<"]="<<A[i]<<'\n';
    }
    float min=A[0], max=A[0];
    int index_min=0, index_max=0;
    for(int i=1; i<size; i++)
    {
        if(A[i]<min)     // знаходимо мінімальний елемент
        {
            min=A[i];
```

```
            index_min=i;  // зберігаємо індекс мінімального елементу
        }
        if(A[i]>max)          // знаходимо максимальний елемент
        {
            max=A[i];
            index_max=i; // зберігаємо індекс максимального елементу
        }
    }
      float multiple=1;  // множення елементів
    if(index_min>index_max)     //в index_min записуємо менший індекс;
    {                           //а в index_max – більший
        int temp=index_max;
        index_max=index_min;
        index_min=temp;
    }
    for(int i=index_min+1; i<index_max; i++)       // множення елементів між
    {                                  // min і max
        multiple*=A[i];
    }
    // вивід результатів на екран
    cout<<"Min element of array = "<<min<<"\tindex = "<<index_min<<'\n';
    cout<<"Max element of array = "<<max<<"\tindex = "<<index_max<<'\n';
    cout<<"Multiple of elements = "<<multiple<<'\n';
}
```

8.2. Заповнення одновимірного масиву випадковими значеннями

Масив можна ініціалізувати випадковими значеннями. Для цього використовується функція генерації випадкових чисел rand(). Функція rand() генерує ціле число від 0 до RAND_MAX (символічна константа, визначена в заголовному файлі stdlib.h). Значення RAND_MAX повинне щонайменше дорівнювати 32767 – максимальне позитивне значення двобайтового цілого числа.

Для того, щоб отримати цілі значення в потрібному діапазоні з функцією rand(), використовується операція ділення по модулю. Наприклад:

```
int Value=rand();              // змінна Value буде дорівнювати
                               // випадковому значенню
                               // у діапазоні від 0 до 32767
int Value2=rand()%6;           // змінна Value2 буде дорівнювати випадковому
                               // значенню в діапазоні від 0 до 5
int Value3=rand()%10+1;        // змінна Value3 буде дорівнювати
                               // випадковому значенню від 1 до 10
```

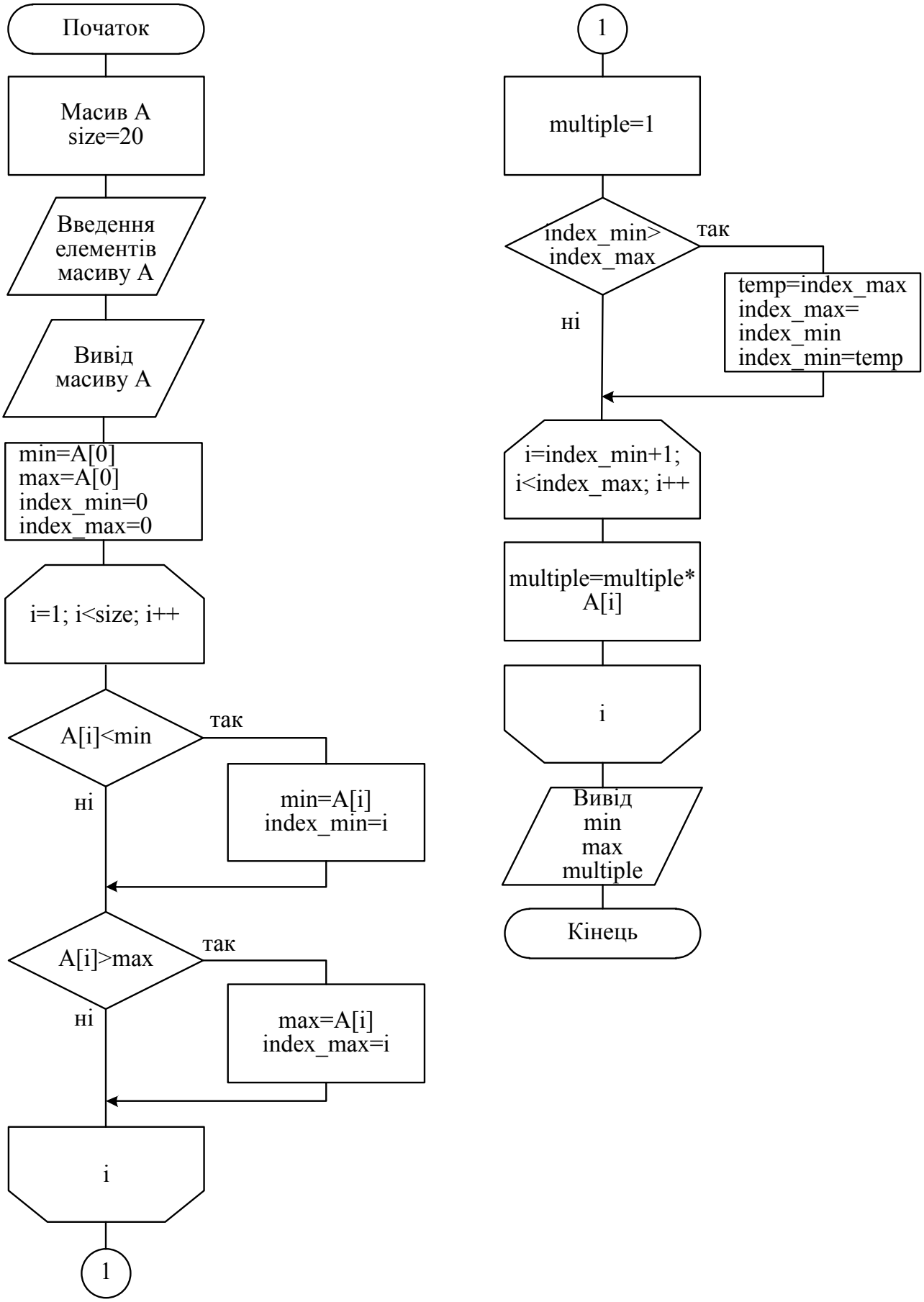


Рис. 8.1. Схема програми прикладу

8.3. Визначення і використання двовимірного масиву

Аналогічно одновимірному масиву можна визначити і багатовимірний (аналог двовимірного масиву – таблиця, в якої є 2 вимірювання: рядки та стовпці). Кожне вимірювання описується власною парою квадратних дужок.

```
int m[ 4 ][ 3 ];
```

Тут визначений двовимірний масив. Перша розмірність – це розмірність по рядках, а друга – по стовпцях, тобто m є двовимірним масивом (матрицею) 4 на 3 (4 рядки і 3 стовпці).

Загалом багатовимірний масив розглядається як одновимірний масив, кожний з елементів якого – теж масив.

Багатовимірний масив можна також ініціалізувати явно:

```
int m [2][3] = {{ 0, 1, 2 },{ 3, 4, 5 }};
```

Внутрішні дужки можна не писати, але це ускладнює читання програми:

```
int m [2][3] = { 0, 1, 2, 3, 4, 5, 6 };
```

Не завжди дужки не важливі. Вони показують, що елемент масиву ініціалізувався масивом.

Тут будуть проініціалізовані перші елементи кожного рядка, а елементи, що залишилися, ініціалізувалися нулем:

```
int m [2][3] = { {0}, {1} };
```

Тут будуть проініціалізовані два елементи першого рядка, а елементи, що залишилися, дорівнюють нулю:

```
int m [2][3] = { 0, 1 };
```

Для індексування в багатовимірному масиві потрібна пара дужок для кожного вимірювання.

Ось обхід та ініціалізація двовимірного масиву:

```
const int size_str = 2;   // визначення константи для зберігання кількості
                          // рядків
const int size_stb = 3;   // визначення константи для зберігання кількості
                          // стовпців
int m[size_str][size_stb];     //визначення масиву
for (int i = 0; i < size_str; i++)
        for (int j = 0; j < size_stb; j++)
                m[i][j]= i+j;     // ініціалізація масиву
```

Приклад 8.2. Вивести на екран елементи двовимірного масиву розміром 5x5, що знаходяться в заштрихованій області (рис. 8.2).

Схема програми прикладу наведена на рис. 8.3.

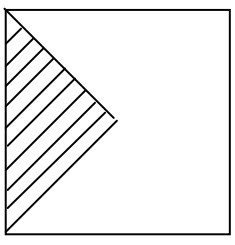


Рис. 8.2. Схема масиву

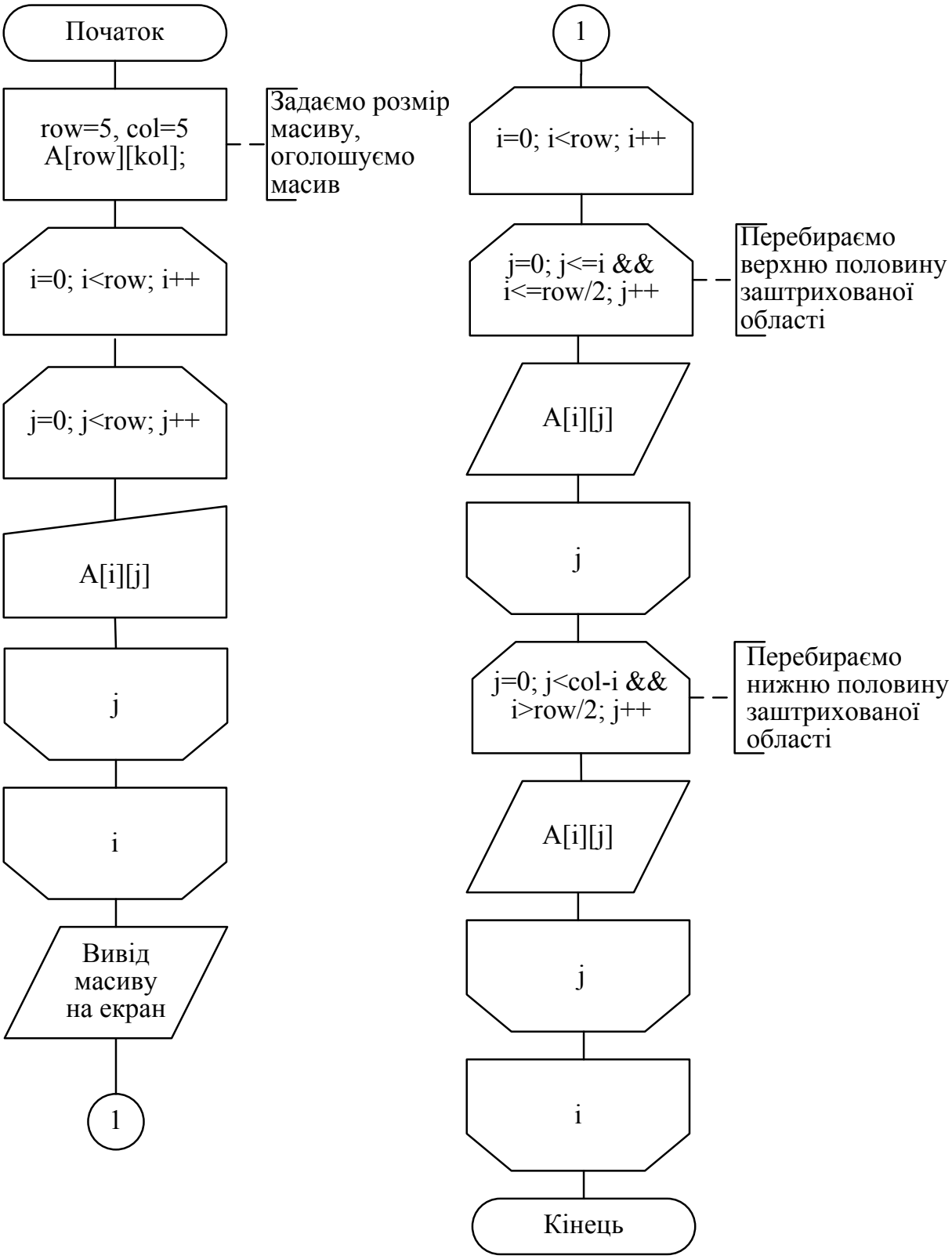


Рис. 8.3. Схема програми прикладу

Програма

```
#include<iostream>
using namespace std;
int main( )
{
    const int row=5, col=5; // розмір масиву
    int A[row][col];      // оголошуємо масив розміром 25 елементів
    // введення елементів масиву з клавіатури
    cout<<"Enter 25 elements of array\n";
    for(int i=0; i<row; i++)
        for(int j=0; j<col; j++)
            {
            cout<<"A["<<i<<"]["<<j<<"]=";
            cin>>A[i][j];
            }
    // вивід масиву на екран
      cout<<"Your array\n";
        for(int i=0; i<row; i++)
        {
            for(int j=0; j<col; j++)
            {
                cout<<A[i][j]<<'\t';
            }
        cout<<'\n';
      }
      // вивід заданої частини масиву на екран
      for(int i=0; i<row; i++)
      {
            for(int j=0; j<=i && i<=row/2; j++)    // верхня половина
            {                                      // заштрихованої області
            cout<<A[i][j]<<'\t';
            }
            for(int j=0; j<col-i && i>row/2; j++)  // нижня половина
            {                                      // заштрихованої області
            cout<<A[i][j]<<'\t';
            }
            cout<<'\n';
      }
}
```

8.4. Робота з діагональними елементами

Для квадратного двовимірного масиву *головну діагональ* утворюють елементи, розташовані між елементом у верхньому лівому та елементом у нижньому правому кутах масиву, включаючи самі ці елементи (рис 8.4).

Індекси елементів головної діагоналі завжди збігаються один з одним.

m[0][0]				
	m[1][1]			
		m[2][2]		
			m[3][3]	
				m[4][4]

Рис. 8.4. Головна діагональ масиву

Тому для вибору елементів, що знаходяться на головній діагоналі, достатньо одного циклу.

Програма

```
#include<iostream>
using namespace std;
int main( )
{
    const int row=5, col=5;    // розмір масиву
    int A[row][col];        // оголошуємо масив розміром 25 елементів
    // введення елементів масиву з клавіатури
    cout<<"Enter 25 elements of array\n";
    for(int i=0; i<row; i++)
        for(int j=0; j<col; j++)
            {
            cout<<"A["<<i<<"]["<<j<<"]=";
            cin>>A[i][j];
            }
      // вивід масиву на екран
      cout<<"Your array\n";
      for(int i=0; i<row; i++)
      {
            for(int j=0; j<col; j++)
            {
            cout<<A[i][j]<<'\t';
            }
            cout<<'\n';
      }
      // вивід на екран елементів масиву,
      // що знаходяться на головній діагоналі
      cout<<"Main diagonal:\n";
      for(int i=0; i<row; i++)
      {
      cout<<A[i][i]<<'\t';      //індекс рядка та стовпчика збігаються
      }
}
```

Побічна діагональ квадратного двовимірного масиву показана на рис 8.5.

				m[0][4]
			m[1][3]	
		m[2][2]		
	m[3][1]			
m[4][0]				

Рис. 8.5. Побічна діагональ масиву

Її утворюють елементи, розташовані між елементом у верхньому правому та елементом у нижньому лівому кутах масиву, включаючи самі ці елементи.

Взаємозв'язок індексів елементу побічної діагоналі, що стоїть на перетині і-го рядка і j-го стовпця, виражається співвідношенням $i + j = n - 1$.

Висновки

У даному розділі розглянуті наведені нижче основні питання:

– визначення і використання одновимірного масиву;

– визначення і використання двовимірного масиву;

– робота з діагональними елементами двовимірного масиву.

Контрольні питання

1. Що таке масив? Як визначити масив?
2. Які існують способи ініціалізації масиву?
3. Дайте визначення розмірності масиву. Яким значенням повинна дорівнювати розмірність масиву?
4. Як отримати доступ до елементів масиву?
5. Як можна скопіювати один масив в інший? Наведіть приклад.
6. Який оператор визначає одновимірний дійсний масив, що містить 100 елементів?
7. Яким чином можна ініціалізувати масив нульовими значеннями?
8. Скільки елементів у масиві?
   int Mas[]={2,3,6,4};
9. Як оголосити масив цілих чисел розміром 10, задати 5-му елементу значення 3, а інші обнулити?
10. Скільки елементів у кожному масиві? Які помилки є в наведених оголошеннях, якщо вони є?
    - int Mas[3]={2,3,6,4};
    - int Mas[5]={2,3,6,4};
11. Як визначити двовимірний масив?
12. Які існують способи ініціалізації багатовимірного масиву?
13. Як отримати доступ до елементів двовимірного масиву?
14. Який оператор визначає двовимірний дійсний масив, що містить 100 елементів? Напишіть його.

15. Яким чином можна ініціалізувати двовимірний масив нульовими значеннями?

16. Скільки елементів у масиві?

```
int Mas[3][4]={2,3,6,4};
```

17. Як оголосити двовимірний масив цілих чисел розміром 10, задати кожному елементу нульового рядка і другого стовпця значення 3, а інші обнулити?

18. Який вираз буде мати доступ до 4-го елементу двовимірного масиву?

19. Яка послідовність операторів надрукує двовимірний масив у табульованому форматі? Напишіть її.

20. Як присвоїти нульові значення першим 10-ти елементам двовимірного масиву?

9. ФУНКЦІЇ

Навчальною метою розділу є ознайомлення студентів з функціями мови програмування С++.

У результаті вивчення даного розділу студенти повинні знати:

– визначення функції;
– визначення прототипу функції;
– визначення файлу;
– функції, які використовуються при роботі з файлами;
– приклади функцій для роботи з файлами;

уміти:

– використовувати виклики функції.

9.1. Визначення функції

Функції дозволяють розбити великі обчислювальні завдання на дрібніші й структурувати їх. Вони можуть розглядатися як визначені користувачем операції.

Розрізняють оголошення і визначення функції. Оголошення функції складається з типу поверненого нею значення, імені функції і списку параметрів або тільки з їх типів без імен. Ці три елементи називаються прототипом функції.

Наприклад:

```
int min(int, int);
```

Для того, щоб використовувати функцію, її потрібно спочатку визначити. Визначення функції складається із заголовка і тіла функції.

Заголовок включає тип результату функції, її ім’я і список параметрів. Параметри функції розділяються комами, і їх список розташовується в дужках, причому дужки не можна опускати. Результат функції називається поверненим значенням. Його тип записується зліва від імені функції.

Дії, які виконує функція, записуються в тілі функції. Тіло функції розташовується у фігурних дужках.

Визначення функції має вигляд:

```
тип_результату ім’я_функції (список параметрів)
{
тіло функції
}
```

Ось простий приклад визначення функції.

```
// мінімум двох чисел
int min (int v1, int v2)
{
        if(v1>=v2)
                return v1;
        else
                return v2;
}
```

Пам’ять для функції виділяється у так званому робочому стеку програми. Ця пам’ять залишається в стеку до закінчення функції. Потім пам’ять автоматично звільняється.

Список параметрів функції описує її формальні параметри. Кожному формальному параметру відводиться пам’ять у робочому стеку. Розмір пам’яті для параметра визначається його специфікацією типу.

Для того, щоб викликати функцію, треба написати ім’я функції і в круглих дужках – параметри функції. Якщо функції необхідні параметри, то вони розташовуються всередині дужок. Це називається передачею параметрів функції. Вирази, які знаходяться між дужками у виклику функції, називаються фактичними параметрами виклику. Кожен параметр відділяється комою. При виклику функції її формальним параметрам присвоюються значення фактичних.

У наступному прикладі функція main() викликає min():

```
#include<iostream >
using namespace std;
int min (int v1, int v2);
 int main()
 {
  int i,j;
  cin >> i >> j;
  cout << min(i,j)<< endl;
  return 0;
 }
```

9.2. Параметри за умовчанням

Часто функції потрібно більше параметрів, ніж у найпростішому і найбільш поширенішому випадку. Значення за умовчанням підходить для більшості випадків. Функція може розраховувати значення за умовчанням для одного або декількох своїх параметрів.

Для цього всередині списку параметрів (в оголошенні або у визначенні) необхідно використовувати ініціалізацію параметра потрібного значення. Наприклад:

```
char fun(int v1 = 10, int v2 = 80, char v3 = ' ');
```

Тут задаються значення за умовчанням для всіх трьох параметрів функції fun.

Функція, для якої задані значення параметрів за умовчанням, може викликатися як з відповідними фактичними параметрами, так і без них.

Якщо фактичний параметр присутній, то значенням за умовчанням нехтують.

Кожний з подальших викликів коректний:

```
char c;
// еквівалентно виклику fun(10,80,' ');
c = fun();                    // 10, 80 і ' ' за умовчанням
```

```
// еквівалентно виклику fun(66,80,' ');
c = fun(66);                // 80 і ' ' за умовчанням
// еквівалентно виклику fun(66,256,' ');
c = fun(66,256);            // ' ' за умовчанням
c = fun(66,256,'#');        // немає значень за умовчанням
```

Функція може задати значення за умовчанням для всіх або тільки для частини своїх параметрів. Але при цьому всі параметри із значеннями за умовчанням повинні йти після параметрів без цих значень:

```
// друк цілої із завданням основи, за умовчанням в десятковому вигляді
void print(int value, int base = 10);   // правильно
void print(int base = 10, int value);   // помилка
```

9.3. Передача параметрів у функцію за значенням

Передача параметрів – це процес ініціалізації пам'яті формальних параметрів значеннями фактичних параметрів.

Стандартним способом ініціалізації при передачі параметрів в C++ є копіювання значень фактичних параметрів у пам'ять формальних. Це називається передачею за значенням.

При передачі за значенням функція не отримує доступу до фактичних параметрів виклику. Значення, з якими працює функція, – це її власні локальні копії параметрів. Вони запам'ятовуються в стеку.

Взагалі зміни цих значень не відбиваються на значеннях фактичних параметрів. При закінченні функції і звільненні стека від пам'яті, виділеної для неї, ці локальні значення втрачаються.

Таким чином, при передачі за значенням вміст фактичних параметрів не змінюється, наприклад:

```
// хочемо міняти місцями v1 і v2
void swap(int v1, int v2)
{
int tmp = v1;
v1 = v2;
v2 = tmp;
}
```

У такому вигляді swap() міняє значення тільки локальних копій своїх параметрів. Значення змінних, переданих swap(), не зміняться (рис.9.1):

```
int a = 10, b = 20;
swap(a,b);
// тут a = 10, b = 20
```

Приклад 9.1. Знайти всі прості числа із заданого діапазону [n, m] (визначити функцію, що дозволяє розпізнавати прості числа).

Схема програми прикладу наведена на рис. 9.2.

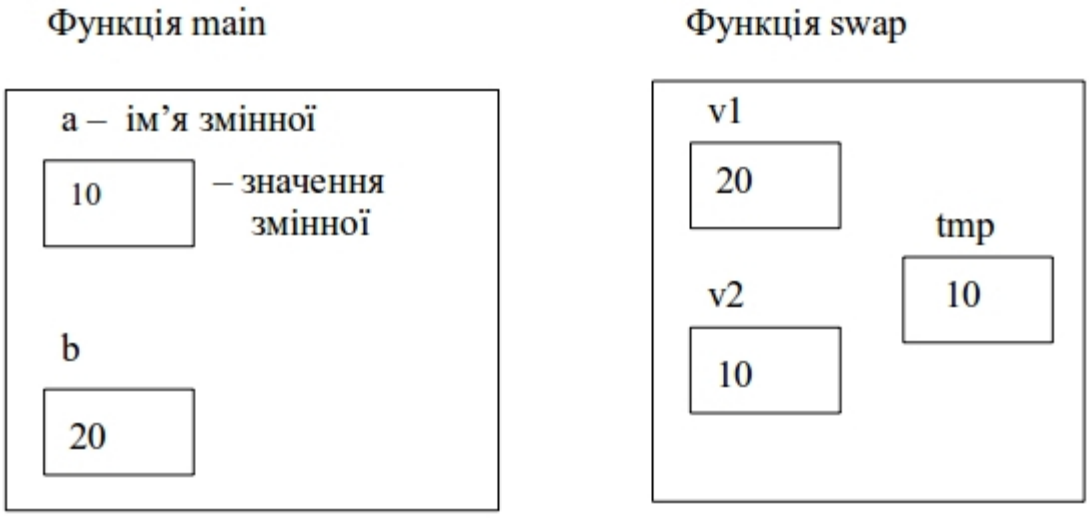


Рис. 9.1. Результат роботи функцій

Програма

```
#include<iostream>
using namespace std;
bool PrimeNumbers(int a);          // прототип функції
void main()
{
     int n=0, m=33;                // задаємо діапазон
     cout<<"Prime numbers:\n";
     for(int i=n; i<=m; i++)       // перебираємо діапазон чисел
     {
          bool rezult= PrimeNumbers(i);     // викликаємо функцію
          if(rezult)                        // якщо функція повернула true,
               cout<<i<<'\t';               // то число просте
     }
}
bool PrimeNumbers (int a)
{
     if(a<=0)
               return false;
     for(int f=a-1; f>1; f--)      // перевіряємо, чи є число простим
          if(a%f==0)
               return false;
     return true;
}
```

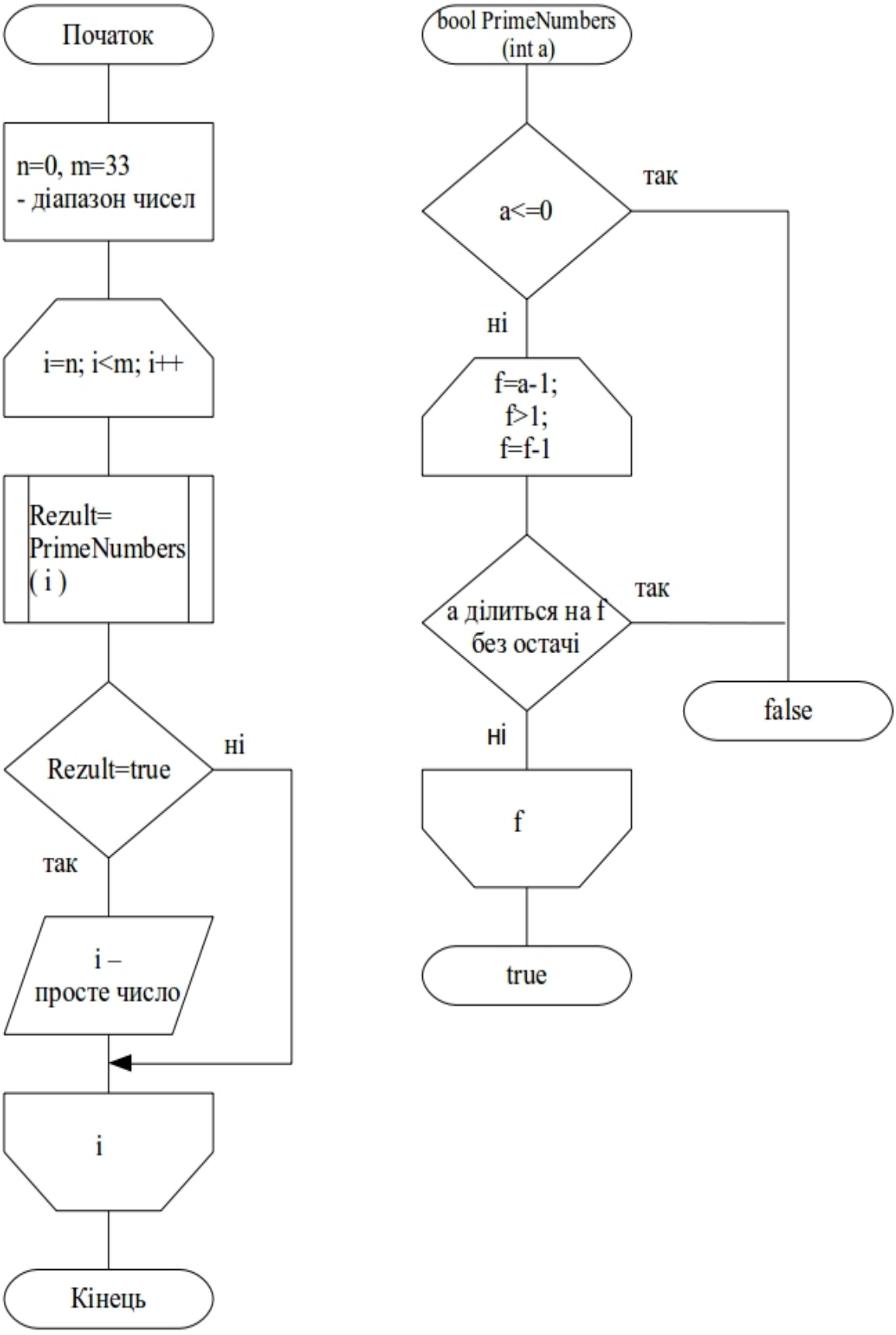


Рис. 9.2. Схема програми прикладу

9.4. Використання функцій для роботи з файлами

Для зберігання текстової інформації, як правило, використовуються файли. Для того, щоб файл був доступний, його потрібно відкрити (функція fopen), зазначивши, для виконання яких дій він буде потрібен. При цьому можливі такі види операцій: читання, записи, оновлення даних, а також тип файлу – двійковий або текстовий.

При роботі з файлами можливі помилки, тому рекомендується, використовуючи функцію ferror, перевіряти результат виконання операцій.

Читання даних з текстового файлу можна виконувати за допомогою функції fscanf, а запис – fprint.

Після завершення роботи з файлом потрібно його обов'язково закрити (функція fclose).

Інформація щодо функцій роботи з файлами наведена в додатку А.

Висновки

У даному розділі розглянуті такі основні питання:

– визначення функції;
– визначення прототипу функції;
– передача параметрів у функцію;
– визначення файлу;
– види файлів;
– опис функцій для виконання операцій з файлами: читання, запису та оновлення даних.

Контрольні питання

1. Що таке функція?
2. Як визначити й оголосити функцію?
3. Що таке прототип функції?
4. Для чого використовуються функції?
5. Як викликати функцію?
6. Чим відрізняються формальні параметри функції від фактичних?
7. Які параметри називаються параметрами за умовчанням?
8. Як здійснюється передача параметрів у функцію за значенням?
9. Що є процесом передачі параметрів у функцію?
10. Скільки можливих способів виклику функції, у якої два з трьох параметрів задані за умовчанням?
11. Що таке файл?
12. Які види файлів бувають?
13. Яка функція застосовується для відкриття файлу?
14. Яка функція застосовується для читання файлу?
15. Які функції застосовуються для запису файлів?
16. Які функції застосовуються для оновлення файлів?
17. Які функції застосовуються для закриття файлів?

10. РЯДКИ І ПОКАЖЧИКИ

Навчальною метою розділу є ознайомлення студентів з рядками і покажчиками мови програмування С++.

У результаті вивчення даного розділу студенти повинні знати:

– визначення рядків;

– визначення покажчиків;

уміти:

– використовувати рядки і покажчики.

10.1. Оголошення і використання рядків

У мові С++ немає спеціального строкового типу даних. Символьні рядки організовуються як масиви символів, останнім з яких є символ ‘\0’, внутрішній код якого дорівнює нулю.

Рядок описується як символьний масив. Наприклад:

```
char STR[20];
```

Одночасно з описом рядок може ініціалізуватися. Можливі два способи ініціалізації рядка – за допомогою строкової константи та у вигляді списку символів:

```
char S[10] = "рядок";
char S[] = "рядок";
char S[10] = {‘р’,’я’,’д’, ’о’, ’к’, ’\0’};
```

Відповідно до результату першого опису під рядок S буде виділено 10 байт пам’яті, з них перші 5 набудуть значення при ініціалізації (6 – нульовий символ). Другий опис сформує рядок з 6 символів. Третій опис рівнозначний першому. Звичайно, можна визначити символьний масив і так:

```
char S[10]={‘р’,’я’,’д’, ’о’, ’к’ };
```

тобто без нульового символа в кінці. Але це приведе до проблем з обробкою такого рядка, оскільки буде відсутній орієнтир на його закінчення.

Окремі символи рядка можуть ідентифікуватися індексованими іменами. Наприклад, в описаному вище рядку S[0]=’р’, S[4]= ‘к’.

Обробка рядків звичайно пов’язана з перебором усіх символів від початку до кінця. Ознакою кінця такого перебору є виявлення нульового символу. Наприклад, підрахунок довжини рядка:

```
#include <iostream>
using namespace std;
int main()
{
        char S[]="string1";
        int i=0;
   for( ; S[i]!='\0'; i++) ;
   cout<<"Length of string "<<i<<'\n';
}
```

Рядки використовують як параметр функції аналогічно масивам інших типів. Необхідно пам’ятати, що ім’я масиву є покажчик його початку.

```
void func (int * mas, int size);
int main()
{
        int Mas[5]={1,2,3,4,5};
        func(Mas, 5);
}
```

У функцію передається адреса і розмір масиву.

При передачі рядка у функцію не потрібно передавати довжину рядка, кінець рядка завжди можна визначити відповідно до завершуючого символа ‘\0’.

10.2. Змінні-покажчики

Будь-яка змінна, окрім типу, імені, розміру і значення, має ще й адресу, за якою в пам’яті зберігається значення цієї змінної.

Наприклад, якщо оголошена змінна

int A=10;

то для неї у пам’яті виділяється 4 байти для зберігання її значення.

І ця зона пам’яті асоціюється зі змінною А (рис. 10.1). При звертанні до змінної А, використовуючи її ім’я, звертаємось до чарунки, де зберігається значення змінної, тобто за заданою адресою.

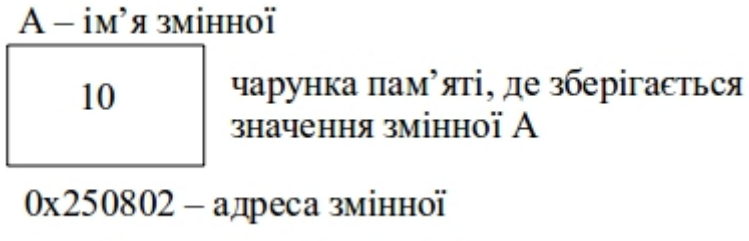


Рис. 10.1. Адреса змінної

Операція отримання адреси & – це унарна операція, застосована до об’єкта. Наприклад:

```
int i = 1024;
cout<<”Address of i = ”<<&i;          //виводиться на екран адреса змінної і.
```

Операція & застосовується тільки до об’єктів, розташованих у пам’яті, – змінних та елементів масивів.

Змінна-покажчик зберігає значення, яке є адресою об’єкта в пам’яті.

Для визначення покажчика необхідно перед його ідентифікатором поставити знак *операції розіменування* *. Цю операцію ще називають операцією *побічності* або *операцією розкриття посилання*. Її опис аналогічний простому визначенню змінної заданого типу.

Приклади визначення змінних-покажчиків:

```
char *pc;              // pc – покажчик на char
int *pi1, *pi2;        // pi1 і pi2 – покажчики на об’єкти типу int
double *pd;            // pd – покажчик на double
```

Покажчик ініціалізується адресою об’єкта даних того самого типу, розташованого в пам’яті, або іншим покажчиком того самого типу (рис. 10.2).

Наприклад:

int i = 1024;

int *p=&i; //покажчик p тепер містить адресу i.

int *pi = p;

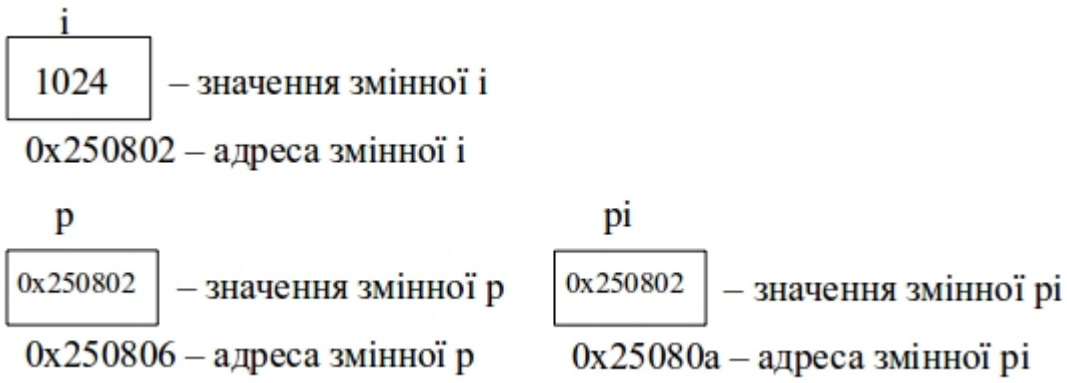


Рис. 10.2. Ініціалізація покажчиків

Кожен покажчик може посилатися тільки на об’єкти відповідного типу. Покажчик має тип. Покажчик типу int указує на об’єкт типу int і т.п. Пов’язаний з покажчиком тип описує, як повинен інтерпретуватися вміст пам’яті, що адресується їм, і який розмір цієї пам’яті. Для зберігання покажчика теж необхідна пам’ять. Розмір її звичайно достатній для зберігання адреси пам’яті та однаковий для всіх (для 32-розрядних систем це 4 байти).

Через покажчик можна звертатися до об’єкта побічно.

Операція * (розіменування), застосована до покажчика, видає об’єкт, на який даний покажчик посилається.

Наприклад, в нашому випадку можна написати:

int j = *pi;

і значення j буде дорівнювати 1024.

10.3. Покажчики і масиви

У C++ існує зв’язок між масивами і покажчиками. Будь-який доступ до елементу масиву, здійснюваний за допомогою операції індексування, може бути виконаний за допомогою покажчика.

Ідентифікатор масиву є адресою першого елементу, що входить у цей масив. Усі елементи масиву розташовані в пам’яті один за одним (рис. 10.3).

int mas[5]={1,2,3,4,5};

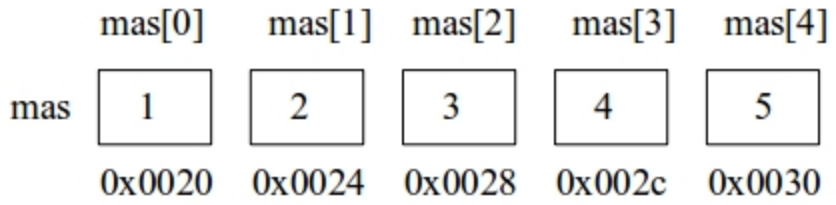


Рис. 10.3. Зображення масиву у пам’яті

Тут ім’я масиву – це покажчик на його перший елемент, тобто
cout<<”Адреса масиву і першого елементу: ”<<mas<<endl;
char *str=”Рядок”; // str – покажчик на рядок
cout<<”Друга буква рядка: ”<<str[1]<<’n’;
cout<<”Адреса рядка: ”<<&str[0]<<’n’;
cout<<”Весь рядок: ”<<str<<’n’;

Можна додавати до покажчиків ціле число, віднімати від покажчиків ціле число або інші покажчики, порівнювати покажчики, а також присвоювати їм значення 0. Не можна додавати, ділити й множити покажчики, змінювати на величину дійсного типу.

int *pMas=mas; //зберігаємо адресу масиву в покажчику
cout<<”Адреса другого елементу масиву:”<<pMas+1<<’n’;
cout<<”Значення другого елементу масиву:”<<*(pMas+1)<<’n’;

Тому, збільшуючи значення покажчика, можна переміщатися по масиву. Якщо значення адреси збільшується на 3 (pMas+3), насправді результатом виразу буде значення адреси, збільшене на 3*розмір_змінної (адреса_pMas + 3 * розмір_типу_int).

10.4. Параметри-покажчики

При цьому за значенням передається сам покажчик, але об’єкт, на який він указує, можна змінити, і ці зміни будуть доступні функції, що робила виклик. У функцію потрібно передавати адресу того об’єкта, який необхідно змінити, а всередині функції розіменовувати параметр-покажчик.

Нашу swap() можна переписати так (рис. 10.4):

```
void swap(int *v1, int *v2)
{
int tmp = *v1;
*v1 = *v2;
*v2 = tmp;
}
```

А її виклик матиме такий вигляд:

```
int a = 10, b = 20;
swap(&a,&b);  // передаються адреси
// тут a = 20, b = 10
```

10.5. Посилання

Посилання є альтернативним ім'ям об'єкта. Визначається специфікацією типу та операцією &.

Посилання повинно ініціалізуватись при визначенні вже існуючим об'єктом такого самого типу.

```
int value = 10;        //визначення змінної
int& ref = value;          //визначення посилання
```

Усі дії, що застосовуються до посилання, відбуваються з об'єктом, з яким пов'язане посилання.

```
ref += 2;              //змінна value збільшується на 2
```

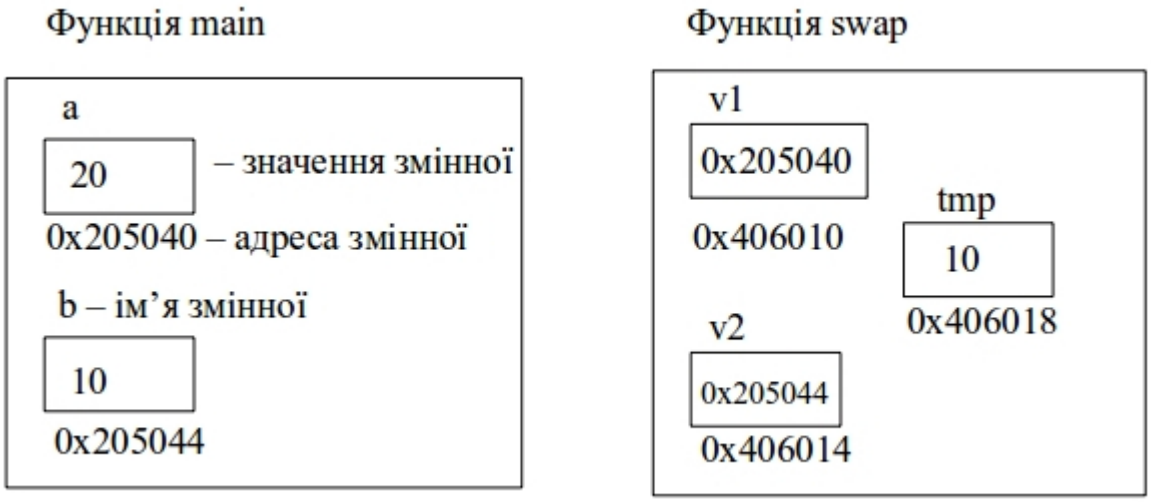


Рис. 10.4. Результат роботи функцій

Не можна визначати масив посилань, посилання на посилання, покажчиків на посилання.

Посилання використовують для передавання параметрів у функцію та повернення посилання з функції.

10.6. Параметри-посилання

Мова C++ дає інший, зручніший спосіб зміни значень параметрів. Для цього кожен формальний параметр, який необхідно змінити, описують з типом посилання. Так і говорять, що відбувається передача параметрів за посиланням.

Оскільки посилання – це інше ім'я об'єкта і всі дії, які проводяться над посиланням, є діями над самим об'єктом, то при передачі параметра посилання всередині функції ми маємо справу із самим об'єктом, а не з його локальною копією.

Наша swap() тепер буде такою:

```
void swap(int &v1, int &v2)
{
int tmp = v1;    // маємо справу з посиланнями, тобто дії проводяться
v1 = v2;         // над самими об'єктами – фактичними параметрами
v2 = tmp;
}
```

А її виклик матиме такий вигляд (рис. 10.5):

```
int a = 10, b = 20;
swap(a,b);   // передаються самі об’єкти
// тут a = 20, b = 10
```

Приклад 10.1. Дані два слова. Надрукувати тільки ті букви слів, які зустрічаються в обох словах тільки один раз. Наприклад, якщо задані слова «процесор» та «інформація», то відповіддю повинно бути: «пеінфмая».
Схема програми прикладу наведена на рис. 10.6.

Рис. 10.5. Результат роботи функцій

Програма

```
#include <iostream>
using namespace std;
void func(char *word1, char *word2);
void main()
{
    char word1[20];
    char word2[20];
    cout<<"Enter 2 word to compare:\n";
    cin>>word1;
    cin>>word2;
    cout<<"Characters are used in words once:"<<endl;
    func(word1,word2);                  // пошук схожих букв у першому слові
    func(word2,word1);                  // пошук схожих букв у другому слові
}
void func(char *word1,char *word2)
{
    bool flagok;
    for(int i=0; *(word1+i)!='\0'; i++)          // перебираємо всі букви
    {                                            // першого слова
        flagok=true;
```

```
            for(int z=0; word1[z]!='\0'; z++)
            {
                if(z!=i&&(*(word1+i)==*(word1+z)))      // визначаємо чи
                {                                       // повторюється
                    flagok=false;                       // будь яка буква
                    break;                              // в першому слові
                }
            }
            if(!flagok) continue;
            for(int j=0; *(word2+j)!='\0'; j++)         // перебираємо всі букви
            {                                           // другого слова
                if(*(word1+i)==*(word2+j))              // порівнюємо букви першого
                {                                       // слова з буквами другого
                    flagok=false;
                    break;
                }
            }
            if(flagok)
                cout<<*(word1+i)<<endl;
        }
}
```

Висновки

У даному розділі розглянуті наведені нижче основні питання:

– визначення рядків і покажчиків;

– покажчики і масиви;

– параметри як покажчики або посилання.

Контрольні питання

1. Що таке рядок?
2. Як оголосити рядок?
3. Що таке покажчик?
4. Як оголосити та ініціалізувати покажчик?
5. Чим відрізняється передача параметрів у функцію за значенням і за посиланням?
6. Чим відрізняється передача параметрів у функцію за посиланням і за покажчиком?
7. Чим відрізняється передача у функцію цілочисельного масиву від рядка символів?
8. Як визначається кінець рядка?
9. Який зв’язок між масивами і покажчиками?
10. Чи можна збільшити значення покажчика на ціле число?

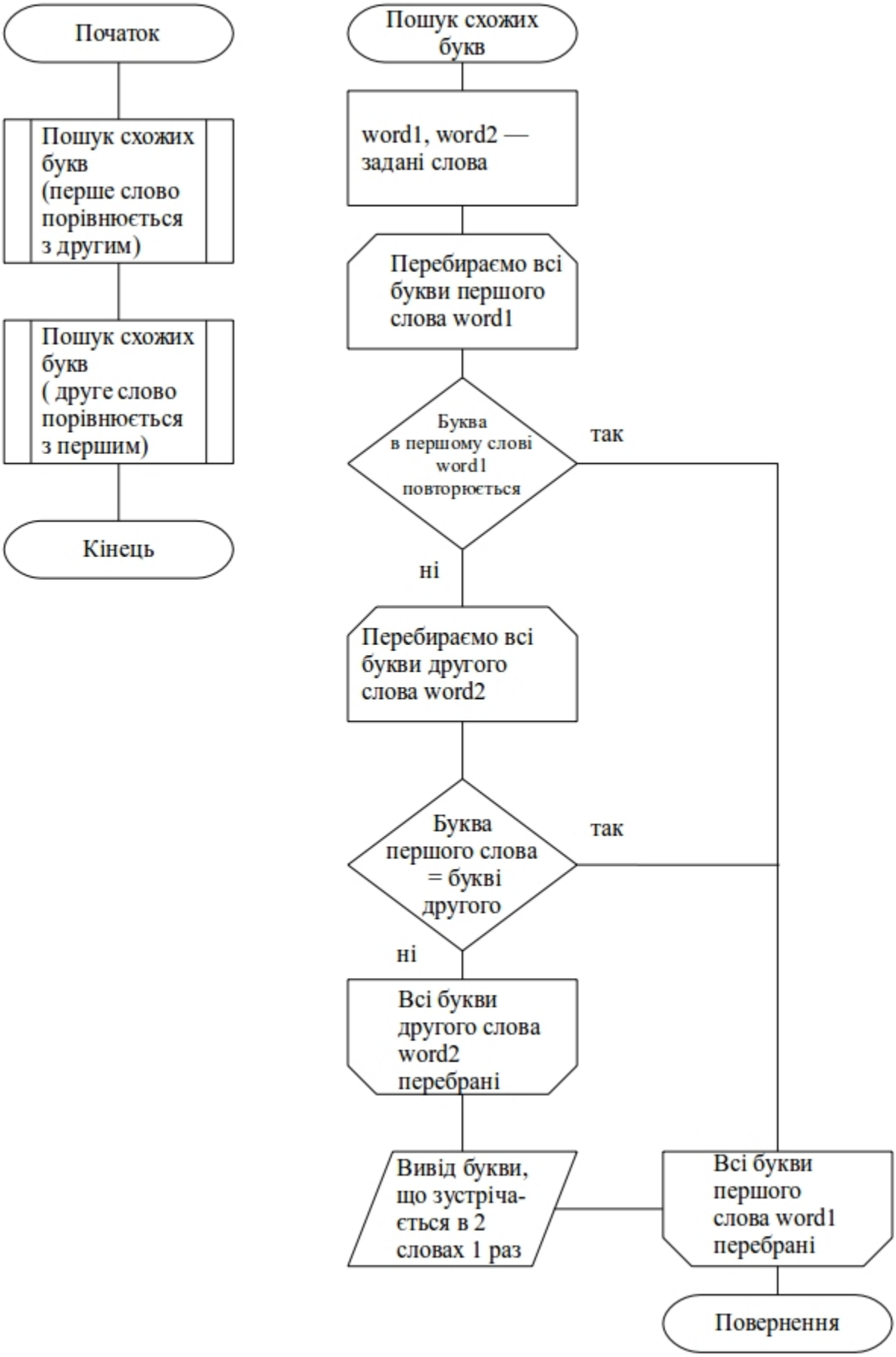


Рис. 10.6. Схема програми прикладу

11. ДИНАМІЧНІ МАСИВИ

Навчальною метою розділу є ознайомлення студентів з динамічними масивами мови програмування С++.

У результаті вивчення даного розділу студенти повинні знати:

– визначення динамічного розподілу пам’яті;

– визначення масиву покажчиків;

уміти:

– використовувати багатовимірні динамічні масиви.

11.1. Динамічний розподіл пам’яті

У мові програмування С++ об’єкти можуть бути розміщені або статично, або динамічно.

Статичними називаються такі величини, пам’ять під які виділяється при компіляції і зберігається протягом усієї роботи програми.

Існує й інший спосіб виділення пам’яті під дані, який називається динамічним. У цьому випадку пам’ять під величини відводиться під час виконання програми. Такі величини називаються динамічними. Розділ оперативної пам’яті, що розподіляється статично, називається статичною пам’яттю; динамічно розподілюваний розділ пам’яті називається динамічною пам’яттю.

Статичне розміщення ефективніше, оскільки виділення пам’яті відбувається до виконання програми, проте воно набагато менш гнучке, тому що ми повинні наперед знати тип і розмір розміщуваного об’єкта.

Використання динамічних величин надає програмісту ряд додаткових можливостей. По-перше, підключення динамічної пам’яті дозволяє збільшити об’єм оброблюваних даних. По-друге, якщо потреба в якихось даних відпала до закінчення програми, то цю пам’ять можна звільнити для іншої інформації. По-третє, використання динамічної пам’яті дозволяє створювати структури даних змінного розміру.

Робота з динамічними величинами пов’язана з використанням покажчиків.

Покажчик містить адресу поля в динамічній пам’яті, де зберігається величина певного типу. Сам покажчик розташовується в статичній пам’яті.

Адреса величини – це номер першого байта поля пам’яті, в якому розташовується величина. Розмір поля пам’яті визначається типом.

Основні відмінності між статичним і динамічним виділенням пам’яті такі:

– статичні об’єкти позначаються іменованими змінними, і дії над цими об’єктами проводяться безпосередньо з використанням їх імен. Динамічні об’єкти не мають власних імен, і дії над ними здійснюються побічно, за допомогою покажчиків;

– виділення і звільнення пам’яті під статичні об’єкти проводиться компілятором автоматично. Програмісту не потрібно самому піклуватися про це. Виділення і звільнення пам’яті під динамічні об’єкти цілком і повністю покладається на програміста.

11.2. Виділення і звільнення динамічної пам’яті

Для маніпуляції пам’яттю, що динамічно виділяється, служать оператори new і delete.

Оператор new має дві форми. Перша форма виділяє пам’ять під одиничний об’єкт певного типу:

nt *pint1 = new int(1024);

int *pint2 = new int;

Тут оператор new виділяє пам’ять під безіменний об’єкт типу int, ініціалізує його значенням 1024 і повертає адресу створеного об’єкта. Ця адреса використовується для ініціалізації покажчика pint1. Всі дії над таким безіменним об’єктом проводяться шляхом розіменування даного покажчика, оскільки явно маніпулювати динамічним об’єктом неможливо. Для ініціалізації покажчика pint2 використовується адреса створеного в динамічній пам’яті об’єкта, але ця область пам’яті залишається не ініціалізованою (рис. 11.1).

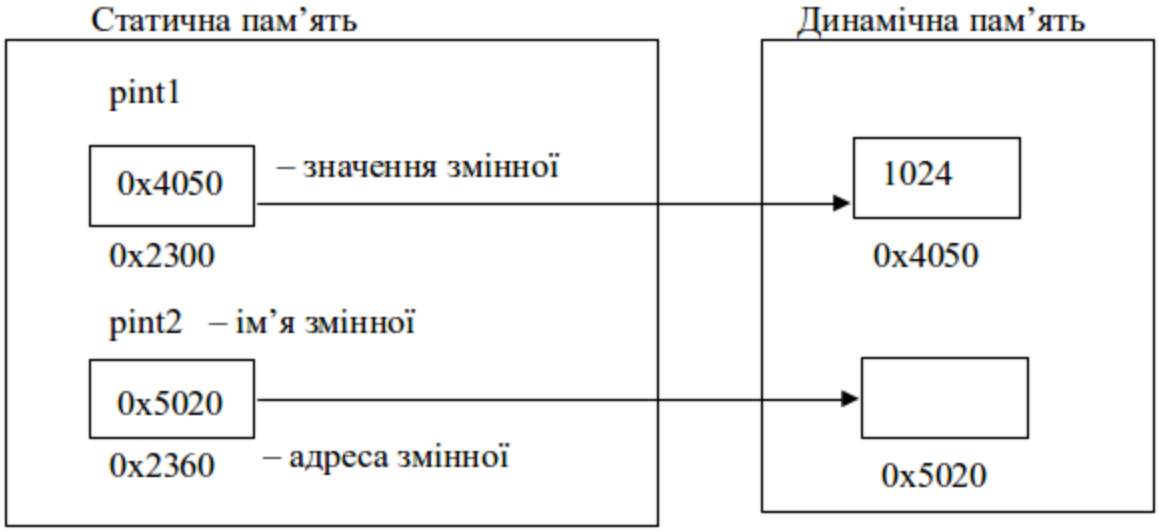


Рис. 11.1. Зв’язок покажчиків та об’єктів динамічної і статичної пам’яті

Друга форма оператора new виділяє пам’ять під масив заданого розміру, що складається з елементів певного типу:

int *pia = new int[4];

У даному прикладі пам’ять виділяється під масив з чотирьох елементів типу int. Дана форма оператора new не дозволяє ініціалізувати елементи масиву.

Обидві форми оператора new повертають однаковий покажчик, у нашому прикладі це покажчик на ціле. І pint1, і pia оголошені абсолютно однаково, проте pint1 указує на єдиний об’єкт типу int, а pia – на перший елемент масиву з чотирьох об’єктів типу int.

Коли динамічний об’єкт більше не потрібен, необхідно звільнити відведену під нього пам’ять. Це робиться за допомогою оператора delete, що має, як і new, дві форми – для одиничного об’єкта і для масиву:

delete[] pia; // звільнення масиву

delete pint1; // звільнення одиничного об’єкта

Що трапиться, якщо не звільнити виділену пам’ять?

Пам’ять витрачатиметься даремно і залишиться невживаною. Проте повернути її системі не можна, оскільки у нас не буде на неї покажчика.

Таке явище одержало спеціальну назву *втрата пам’яті*. Врешті-решт програма аварійно завершиться із-за браку пам’яті (якщо, звичайно, вона працюватиме достатньо довго). Невелика втрата важко піддається виявленню, але існують утиліти, що допомагають це зробити.

11.3. Покажчик на покажчик

Коли ми застосовуємо операцію отримання адреси (&) до об’єкта типу int, то одержуємо результат типу int*

```
int ival = 1024;
int *pi = &ival;
```

Якщо ту ж операцію застосувати до об’єкта типу int* (покажчик на int), ми отримаємо покажчик на покажчик на int, тобто int**.

Покажчик на покажчик – це змінна, значенням якої є адреса покажчика (рис. 11.2).

```
int **ppi = &pi;
cout << "Значення ival\n"
<< "явне значення: " << ival << "\n"
<< "непряма адресація: " << *pi << "\n"
<< "двічі непряма адресація: " << **ppi << "\n";
```

Шляхом розіменування ppi ми одержуємо об’єкт типу int*, що містить адресу ival. Щоб отримати сам об’єкт ival, операцію розіменування до ppi необхідно застосувати двічі.

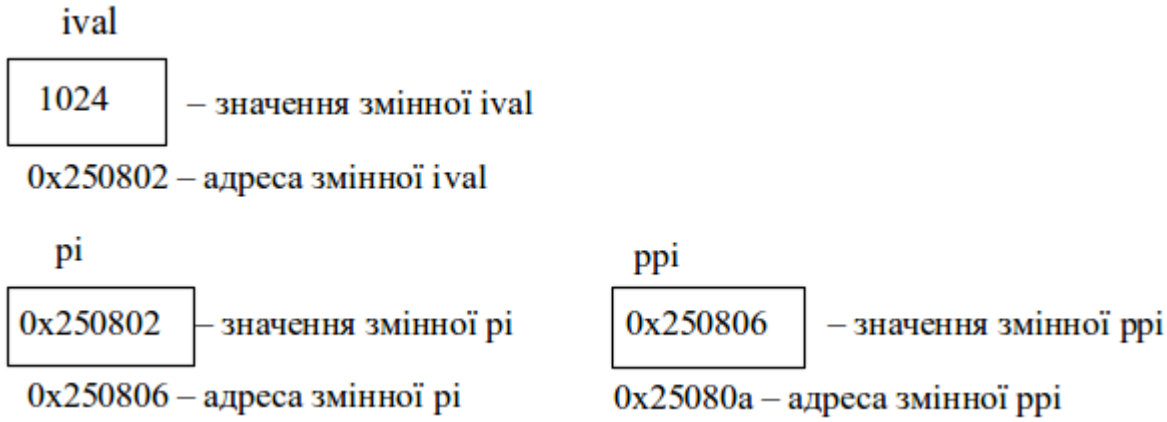


Рис. 11.2. Покажчик на покажчик

11.4. Масив покажчиків

Масиви можуть містити покажчики. Кожним елементом такого масиву є адреса деякого об’єкта в пам’яті. Об’єкти можуть бути як статичними, так і динамічними (рис. 11.3).

Наприклад:

```
int a=5, b=8, c=10;
int* pArrSt[3];
pArrSt[0]=&a;                    // a, b, c – статичні об’єкти
pArrSt[1]=&b;
```

```
pArrSt[2]=&c;
int *pArrD[3];
pArrD[0]=new int;          // динамічний об’єкт
pArrD[1]=new int;
pArrD[2]=new int;
```

pArrSt і pArrD – одновимірні масиви, що складаються з 3 елементів. Значеннями елементів масивів є адреси об’єктів, уже розміщених у пам’яті. В массиві pArrSt зберігаються адреси статичних об'єктів, а в масиві pArrD − динамічних.

Така структура даних часто використовується для формування масиву рядків або для формування багатовимірних динамічних масивів.

Можна зберегти адресу масиву покажчиків. Оскільки ім’я масиву – це адреса першого елементу масиву, то ім’я масиву – це змінна, значенням якої є адреса, тобто це покажчик. Якщо масив зберігає адреси інших об’єктів, то ім’я масиву – це покажчик на покажчик.

Наприклад:

```
int ** pArr=pArrSt;                                    // зберігаємо адресу
cout<<”Адреса масиву покажчиків: ”<<pArr[0];           //масиву
покажчиків
```

Використовуючи покажчик на масив можна звертатися до елементів масиву, використовуючи індекс або арифметику покажчиків.

```
cout<<”Адрес змінної a: ”<<pArr[0]<<’\n’;
cout<<”Адрес змінної в:”<<(pArr+1)<<’\n’;
cout<<”Значення змінної a:”<<*pArr[0]<<’\n’;
cout<<”Значення змінної c:”<<*(pArr+2)<<’\n’;
```

11.5. Багатовимірні динамічні масиви

Багатовимірні динамічні масиви будуються за принципами, викладеними вище. Двовимірний динамічний масив – це одновимірний масив, елементами якого є одновимірні масиви (адреси цих масивів). Спочатку створюється динамічний масив покажчиків, а потім виділяється пам’ять під декілька одновимірних масивів, адреси яких зберігаються в масиві покажчиків. Тому виходять «масиви в масиві» (рис.11.3).

Доступ до елементів одержаного двовимірного масиву здійснюється так само, як і в статичному масиві, тобто за допомогою індексів рядка і стовпця або за допомогою адреси елементу, наприклад:

```
mas[1][3] = 25;
або
*(mas[1]+3) = 25;
```

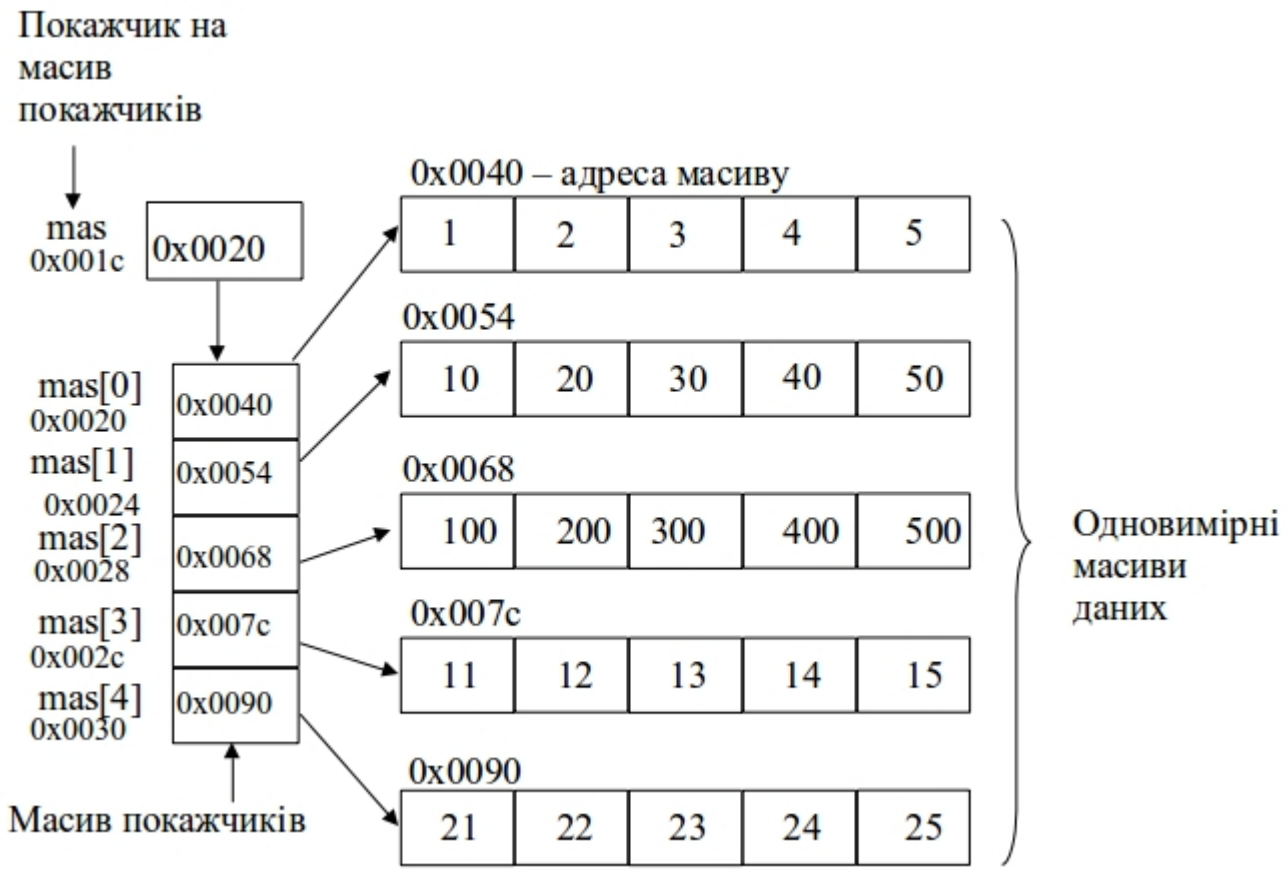


Рис. 11.3. Зображення динамічного масиву у пам’яті

Приклад 11.1. У двовимірний масив повинні були записати оцінки кожного з 23 учнів класу за дванадцятьма предметами (у першому стовпці – по першому предмету, в другому – по другому і т. д.), але помилково забули вписати в масив оцінки ще за одним предметом, який повинен бути в переліку в s-м стовпці. Змінити масив так, щоб він був заповнений як треба. Оцінки по новому предмету вводяться з клавіатури і в додатковий масив записуватися не повинні.

Програма

```
#include <iostream>
using namespace std;
// заповнюємо масив випадковими значеннями
void FullArray(int **Array,const int row, const int column)
{
    for(int i=0; i<row; i++)
        for(int j=0; j<column; j++)
        {
            Array[i][j]=rand()%5+1;
            if(j==column-1)       // останній стовпець заповнюємо
            Array[i][j]=0;        // нулями
        }
}
```

```
// виводимо масив на екран
void OutArray(int **Array,const int row, const int column)
{
	for(int i=0; i<row; i++)
	{
		for(int j=0; j<column; j++)
		{
			cout<<Array[i][j]<<'\t';
		}
		cout<<'\n';
	}
}
// видаляємо масив
void DeleteArray(int **Array,const int row)
{
	for(int y=0; y<row; y++)
	{
		delete[] Array[y];
	}
	delete[] Array;
}
// зрушуємо елементи в рядку вправо на 1 позицію,
// закінчуючи позицією вставки стовпця
void MoveElementsOfArray(int *StringOfArray, const int size, int move)
{
	for(int i=size-2;i>=move;i--)
	{
		StringOfArray[i+1]=StringOfArray[i];
	}
}
void main()
{
	const int row=5;
	const int column=4;
	int **p=new int *[row];		// створюємо масив покажчиків
					// і зберігаємо його адресу
	for(int i=0; i<row; i++)	// створюємо row масивів розміром column
	p[i]=new int [column];
	FullArray(p,row,column);	// заповнюємо масив
	OutArray(p,row,column);		// виводимо масив
	int move;
	cout<<"Enter number of column you want to paste"<<endl;
	cin>>move;
	cout<<"Enter marks:"<<endl;
	for(int i=0; i<row; i++)
```

```
    {
        // зрушуємо елементи рядка вправо
        MoveElementsOfArray(p[i],column,move);
        // вписуємо оцінки відповідно до нового предмета
        cin>>p[i][move];
    }
    OutArray(p,row,column);          // виводимо на екран масив
    DeleteArray(p,row);              // видаляємо масив
}
```

Схема програми прикладу наведена на рис. 11.4.

Висновки

У даному розділі розглянуті наведені нижче основні питання:

– динамічний розподіл пам’яті;

– покажчик на покажчик;

– масив покажчиків;

– багатовимірні динамічні масиви.

Контрольні питання

1. Чим відрізняється статичне виділення пам’яті від динамічного?
2. Як виділити ділянку пам’яті в динамічній області?
3. Як звільнити пам’ять, виділену в динамічній області?
4. Що таке покажчик на покажчик?
5. У яких випадках використовується покажчик на покажчик?
6. Як розіменовувати покажчик на покажчик?
7. Як визначити масив покажчиків?
8. Як визначити двовимірний динамічний масив?
9. Як ініціалізувати двовимірний динамічний масив?
10. Як отримати доступ до елементів двовимірного динамічного масиву?

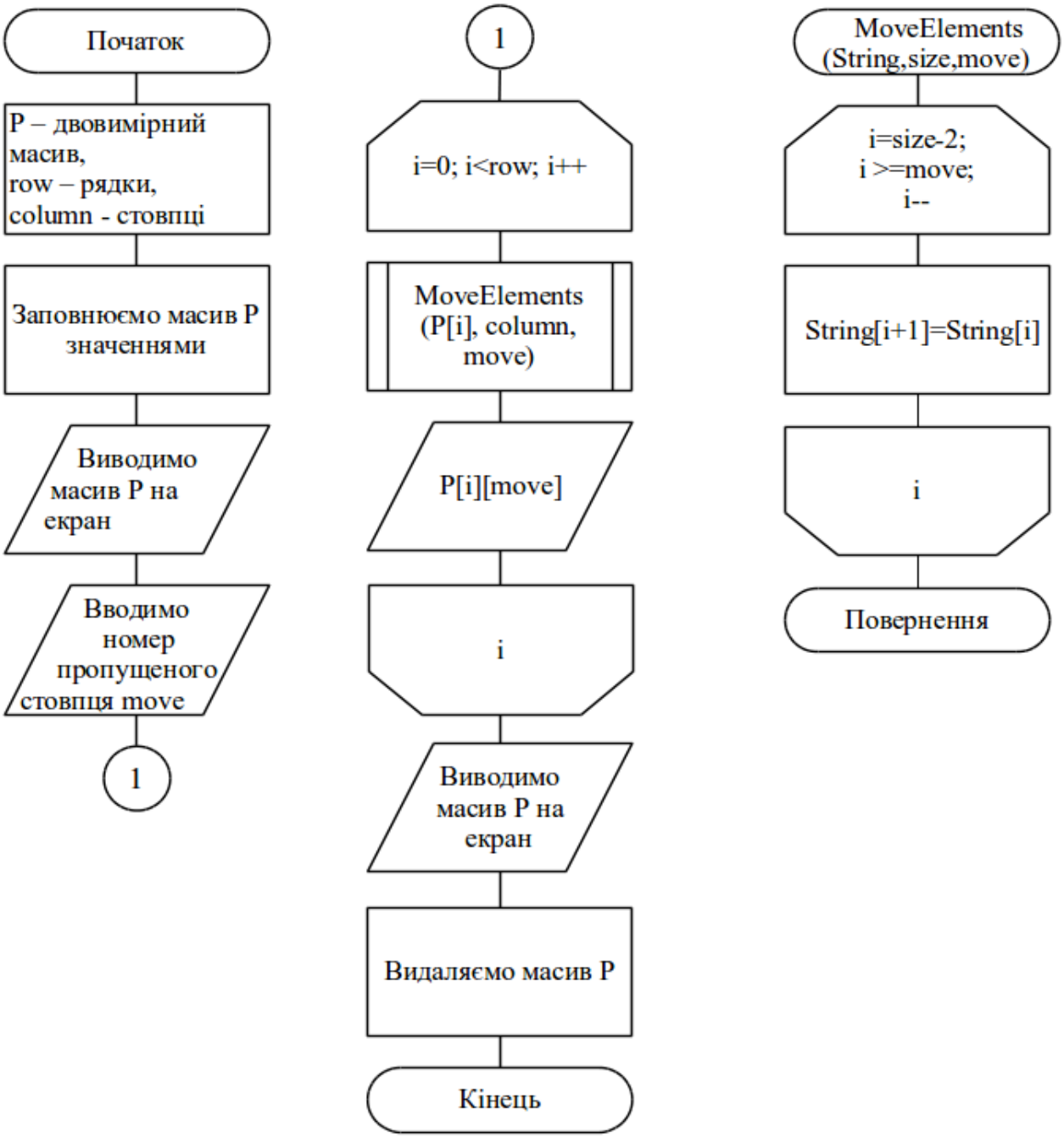


Рис. 11.4. Схема програми прикладу

12. СТРУКТУРИ

Навчальною метою розділу є ознайомлення студентів із структурами мови програмування С++.

У результаті вивчення даного розділу студенти повинні знати:

– визначення структур;

– визначення масивів структур;

уміти:

– використовувати структури;

– створювати багатофайлові проекти.

12.1. Оголошення і визначення структур

Структура – це одна або декілька змінних (можливо, різних типів), які для зручності роботи з ними згруповані під одним ім'ям.

Оголошення структури складається з ключового слова struct і містить список описів, узятих у фігурні дужки:

```
struct point {
        int x;
        int y;
        };
```

За словом struct йде ім'я структури. Це ім'я стає ім'ям нового типу даних, що відповідають цій структурі і які включають описані всередині структури змінні (у нашому випадку це *x* і *y* типу int). Ніяка пам'ять при цьому не виділяється.

Перераховані в структурі змінні називаються членами або елементами структури.

Крапка з комою в кінці оголошення структурного типу обов'язкова.

Тепер point – це тип даних. Він ще називається визначеним користувачем типом даних.

До визначених користувачем типів даних належать, окрім структур, також об'єднання і класи. Можна визначати змінні такого типу.

Змінні типу point можуть визначатися точно так, як і змінні інших типів:

```
point p1;               // описує змінну p1 типу point
point *pp;              // описує покажчик на структуру типу point
```

Можна відразу при оголошенні типу структури визначити і змінну такого типу. Ідентифікатор змінної при цьому потрібно помістити між фігурною дужкою, що закриває, і крапкою з комою:

```
struct point {
        int x,y;
        } p1;       // визначення змінної p1 типу point
```

Можна описати тільки змінну, не оголошуючи тип. Для цього в такому визначенні потрібно опустити ім'я структури:

```
struct {
      int x,y;        // визначення змінної p1
      } p1;           // без оголошення типу структури
```

Використовувати таку змінну можна також.

Доступ до окремих елементів структури можна одержати за допомогою конструкції

им’я_змінної.елемент,

де им’я_змінної – це ім’я змінної структурного типу.

Наприклад:

```
int i = 100;
p1.y = i;              // елемент у змінній p1
int j = p1.x;
```

Структури можуть бути вкладені одна в одну. Наприклад:

```
struct rect {
struct point pt1;
struct point pt2;
};
```

Доступ має вигляд:

```
rect r;
r.pt1.x = 20;
```

Для ініціалізації простих структур може використовуватися такий самий запис, як і для ініціалізації масивів.

Наприклад:

```
struct address {
char *name;            // ім’я
char *town;            // місто
char *street;          // вулиця
long number;           // номер будинку
};
address ad = { "Дмитренко І.І.", "Харків", "вул. Пушкінська", 79 };
```

Якщо елементів у дужках більше, ніж у структурі, то буде помилка, а якщо менше, то елементи, яких не вистачає будуть ініціалізуватися нулями.

12.2. Структури і покажчики

Дуже часто об’єкти-структури розміщують у динамічній пам’яті і працюють з ними через покажчики:

```
address *pad1 = new address;   // відводиться пам’ять під
                               // об’єкт типу address
```

Можна ініціалізувати покажчик на структуру та адресу вже існуючого об’єкта:

```
address *pad2 = &ad;           // pad2 указує на ad
```

Для доступу до елементу структури, на яку вказує покажчик, в C++ існує особлива форма запису. Якщо p – покажчик на структуру, то

p->елемент_структури

є її окремий елемент.

Наприклад:

```
int i = pad2->number;          // i = 79
```

Це зручніший запис для

```
int i = (*pad2).number;      // i = 79
```

Операції доступу до елементів структури **.** і **->** мають дуже високий пріоритет і виконуються раніше всіх інших операцій. Наприклад:

```
++pad2->number;
```

сприймається як ++(pad2->number) і на одиницю збільшиться елемент структури number, а не покажчик pad2.

12.3. Масиви структур

Можна описувати і масиви структур. Наприклад, опис

```
address ara[3];
```

дасть нам масив з 3-х змінних типу address.

Ініціалізація такого масиву схожа з ініціалізацією багатовимірного масиву – кожен елемент береться у фігурні дужки:

```
address ara[3]= {
        { "Дмитренко І.І.", "Харків", "вул. Пушкінська", 79 },
        { "Петренко П.П.", "Харків", "вул. Сумська", 150 },
        { "Сидоренко С.С.", "Харків", "вул. Свободи", 12 }
        };
```

Якщо для доступу до такого масиву використовувати покажчик, наприклад

```
address *para = ara;
```

то робота з таким покажчиком аналогічна роботі з будь-яким покажчиком на масив:

```
para++;
```

пересуне його до наступного елементу масиву para

```
para->number;
```

дасть нам елемент number тієї структури, на яку в даний момент указує para і т.д.

Для обчислення розміру структури потрібно завжди використовувати sizeof, оскільки її розмір може не дорівнювати сумі розмірів її елементів із-за особливостей розміщення в пам’яті, а sizeof завжди поверне правильний розмір.

```
int i = sizeof(address);
address a;
int j = sizeof a.name+sizeof a.town+sizeof a.street+sizeof a.number;
// i = 14, a j = 10
```

Структурні змінні можна присвоювати, передавати як параметри функції і повертати з функції як результат. Решта операцій не визначена.

При цьому повинна забезпечуватися відповідність типів. Два структурні типи вважаються різними, навіть якщо вони мають однакові елементи.

Наприклад:

```
struct s1 { int a; };
struct s2 { int a; };
```

є два різні типи, тому

```
s1 x;
s2 y = x;               // помилка : невідповідність типів
```

Структурні типи також завжди відмінні від базових типів, тому

```
s1 x;
int i = x;              // помилка: невідповідність типів
```

12.4. Основи багатофайлових проектів

Мати всю програму в одному файлі звичайно неможливо, оскільки коди стандартних бібліотек і операційної системи знаходяться десь у іншому місці. Крім того, зберігати весь текст призначеної для користувача програми в одному файлі, як правило, непрактично і незручно. Спосіб поміщення програми у файли може допомогти читачу охопити всю структуру програми, а також може дати можливість компілятору реалізувати цю структуру. Оскільки компіляції підлягає весь файл, то у всіх випадках, коли до нього вноситься зміна (наскільки б малою вона не була), потрібно компілювати наново. Навіть для програми помірних розмірів час, що витрачається на перекомпіляцію, можна значно понизити за допомогою розбиття програми на файли відповідних розмірів.

Програма, що складається з декількох роздільно компільованих файлів, повинна бути узгодженою в сенсі використання імен і типів, так само, як і програма, що складається з одного вихідного файлу. У принципі це може забезпечити і компонувальник. Компонувальник – це програма, що з'єднує окремо скомпільовані частини разом.

Програміст може компенсувати недолік підтримки з боку компонувальника, надавши додаткову інформацію про типи (описи). Після цього узгодженість програми забезпечується перевіркою узгодженості описів, які знаходяться в окремо скомпільованих частинах.

Якщо не вказано інше, то ім'я, що не є локальним для функції або класу, в кожній частині програми, скомпільованої окремо, повинне належати до одного і того ж типу, значення, функції або об'єкта. Тобто в програмі може бути тільки один нелокальний тип, одне значення, одна функція або один об'єкт з цим ім'ям.

Об'єкт у програмі повинен визначатися тільки один раз. Описуватися він може багато раз, але типи повинні точно узгоджуватися.

Механізм включення за допомогою #include – це надзвичайно простий спосіб обробки тексту для складання шматків вихідної програми в одну одиницю (файл) для її компіляції. Директива

```
#include "to_be_included"
```

заміщає рядок, в якому зустрілося #include, вмістом файлу "to_be_included". Його вмістом повинен бути вихідний текст на C++, оскільки далі за нього читатиме компілятор.

Найростіше вирішити проблему розбиття програми на декілька файлів, помістивши функції і визначення даних у файли та описавши типи, необхідні для їх взаємодії, в одному заголовному файлі, який включається у всю решту файлів.

Програма на C++ звичайно складається з великої кількості вихідних файлів, кожний з яких містить описи типів, функцій, змінних і констант.

Щоб ім’я можна було використовувати в різних вихідних файлах для посилання на один і той самий об’єкт, воно повинно бути описано як зовнішнє. Наприклад:

```
extern double sqrt(double);
extern instream cin;
```

Найзвичайніший спосіб забезпечити узгодженість вихідних файлів – це помістити такі описи в окремі файли, звані заголовними (або хедер) файлами, а потім включити, тобто скопіювати ці заголовні файли у всі файли, де потрібні ці описи. Наприклад, якщо опис sqrt зберігається в заголовному файлі для стандартних математичних функцій math.h і необхідно добути квадратний корінь з 4, можна написати:

```
#include < math.h >
//...
x = sqrt(4);
```

Оскільки звичайні заголовні файли включаються в багато вихідних файлів, вони не містять описів, які не повинні повторюватися. Наприклад, тіла функцій даються тільки для inline-підставлюваних функцій, а ініціалізатори – тільки для констант. Як виняток заголовний файл є сховищем інформації про типи. Він забезпечує інтерфейс між окремо компільованими частинами програми.

У команді включення include ім’я файлу, узяте в кутові дужки, наприклад, належить до файлу з цим ім’ям у стандартному каталозі (часто це /usr/include/CC). На файли, що знаходяться в яких-небудь інших місцях, посилаються за допомогою імен, поміщених у подвійні лапки.

Наприклад:

```
#include "math1.h"
#include "/usr/bs/math2.h"
```

підставе math1.h з поточного призначеного для користувача каталогу, а math2.h – з каталогу /usr/bs.

Тут наводиться маленький закінчений приклад програми, в якому рядок визначається в одному файлі, а її друк проводиться в іншому. Файл header.h визначає необхідні типи:

```
// header.h
extern char* prog_name;
extern void f();
```

У файлі main.c знаходиться головна програма:

```
// main.c
#include "header.h"
char* prog_name = "приклад";
main()
{
   f();
}
```

а файл f.c друкує рядок:

```
// f.c
#include "header.h"
void f()
{
   cout << prog_name << "\n";
}
```

12.5. Правила використання заголовних файлів

Головний спосіб досягти узгодженості визначень у різних файлах – це включення заголовних файлів з інтерфейсною інформацією (типи даних, оголошення функцій і т.п.) у вихідні файли, що містять виконуваний код і визначення даних.

У заголовному файлі можуть міститися:

– визначення типів (класів, у т.ч. структур):
```
struct point {int x,y; };
```
– оголошення функцій:
```
int sort(int*, int);
```
– визначення функцій-підстановок:
```
inline int min(int a,int b) { return a<=b ? a : b; }
```
– оголошення даних:
```
extern int a;
```
– визначення констант:
```
const double pi = 3.14159;
```
– переліки:
```
enum bool { false, true };
```
– попередні оголошення імен:
```
struct list;
```
– оголошення типів typedef
```
typedef int I;
```
– директиви включення
```
#include<iostream.h>
```
– визначення макросів
```
#define bcase break; case
```
– коментарі.

У заголовні файли не можуть бути поміщені визначення:

– звичайних функцій
```
int min(int a,int b) { return a<=b ? a : b; }
```
– даних
```
int a;
```
– складних констант
```
const tbl[] = {/*...*/};
```

Усі ці правила не є вимогами мови, а просто дозволяють уникнути помилок.

Приклад 12.1. Описати структуру Student (прізвище, група, успішність (масив з 5 оцінок)).

Написати програму, що дозволяє працювати з інформацією про 10 студентів групи:

– змінювати дані про студента;

– виводити список відмінників (>75% оцінок 4 і 5);

– виводити список двієчників (>50% оцінок 2 і 3).

Схема програми прикладу наведена на рис. 12.1, 12.2.

Программа

```
//main.cpp
#include "header.h"
#include " declaration.h"
void main()
{
    // Створюємо змінну структури (в даному випадку масив)
    Student info[10]={
            {"Sokol","SM-07","45434"},
            {"Fedko","EK-07","55555"},
            {"Sobolenko","GK-07","45343"},
            {"Sergeenko","SE-07","25334"},
            {"Poltavets","SM-07","43544"},
            {"Reznichenko","TS-07","44443"},
            {"Dmitrenko","TS-07","53453"},
            {"Sokolenko","GK-07","55434"},
            {"Dubina","MK-07","55455"},
            {"Trezvon","EK-07","55333"},
        };
    int action;
    out(info);
    do
    {
        cout<<"\nChoose the action:"<<endl
        <<"press 1 to change the data of students"<<endl
        <<"press 2 to see the list of students with excellent score"<<endl
        <<"press 3 to see the list of students with bad score"<<endl
        <<"press 0 to exit"<<endl;
            cout<<"\nChoose the action:"<<endl;
        cin>>action;
        switch(action)
        {
            case 1:  data(info);   break;
            case 2:  excellent(info);break;
            case 3:  bad(info);break;
        }
```

```
    }while(action!=0);
}

//declaration.h
#include <iostream>
using namespace std;
#include <string.h>
struct Student
{
    char surname[20];
    char group[10];
    char score[6];
};

//header.h
void data(Student*info);
void excellent(Student*info);
void bad(Student*info);
void out(Student*info);

//data.cpp
#include " declaration.h"
void data(Student*sur)
{
    char proceed;
    do
    {
    cout<<"Which data do you want to change?"<<endl
        <<"Enter the number of student"<<endl;
    int i;
    cin>>i;
    i-=1;
    cout<<"Enter SURNAME\\GROUP\\SCORE of the student"<<endl;
    cin>>sur[i].surname
      >>sur[i].group
      >>sur[i].score;
    cout<<"Do you want to change data of other student? y/n"<<endl;
    cin>>proceed;
    cout<<endl;
    }
    while(proceed == 'y');
    cout<<"There is your new list\n"<<endl;
    out(sur);
}
```

```
void excellent(Student* info)
{
    int counter=0;
    cout<<"The list of students with excellent score:"<<endl;
    for(int i=0; i<10; i++)
    {
    for(int j=0; j<5; j++)
    {
        if(info[i].score[j]=='5' || info[i].score[j]=='4' )
        {
            counter +=1;
        }
    }
    if(counter >3) cout<<info[i].surname<<'\t'<<info[i].group<<endl;
    counter =0;
    }
}
void bad(Student*info)
{
    int counter =0;
    cout<<"The list of students with bad score:"<<endl;
    for(int i=0; i<10; i++)
    {
    for(int j=0; j<5; j++)
    {
        if(info[i].score[j]=='2' || info[i].score[j]=='3')
        {
            counter +=1;
        }
    }
    if(counter >=3) cout<<info[i].surname<<'\t'<<info[i].group<<endl;
    counter =0;
    }
}
void out(Student* info)
{
    cout<<"NUMBER\tSURNAME\t\tGROUP\t\tPROGRESS"<<endl;
    for(int i=0; i<10; i++)
    {
        cout<<i+1<<'\t'
            <<info[i].surname<<"\t\t"
            <<info[i].group<<"\t\t"
            <<info[i].score<<endl;
    }
}
```

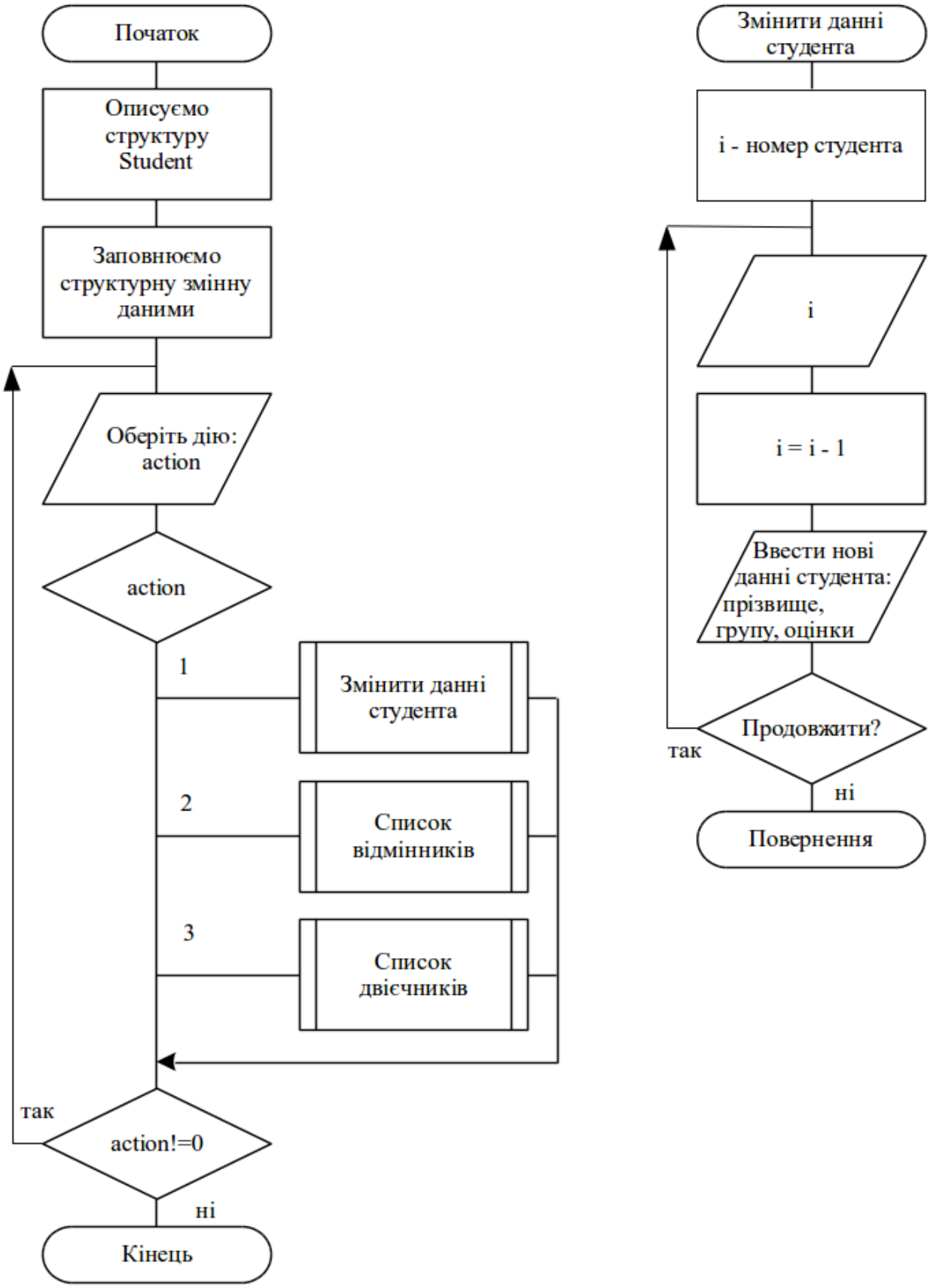


Рис. 12.1. Схема програми прикладу

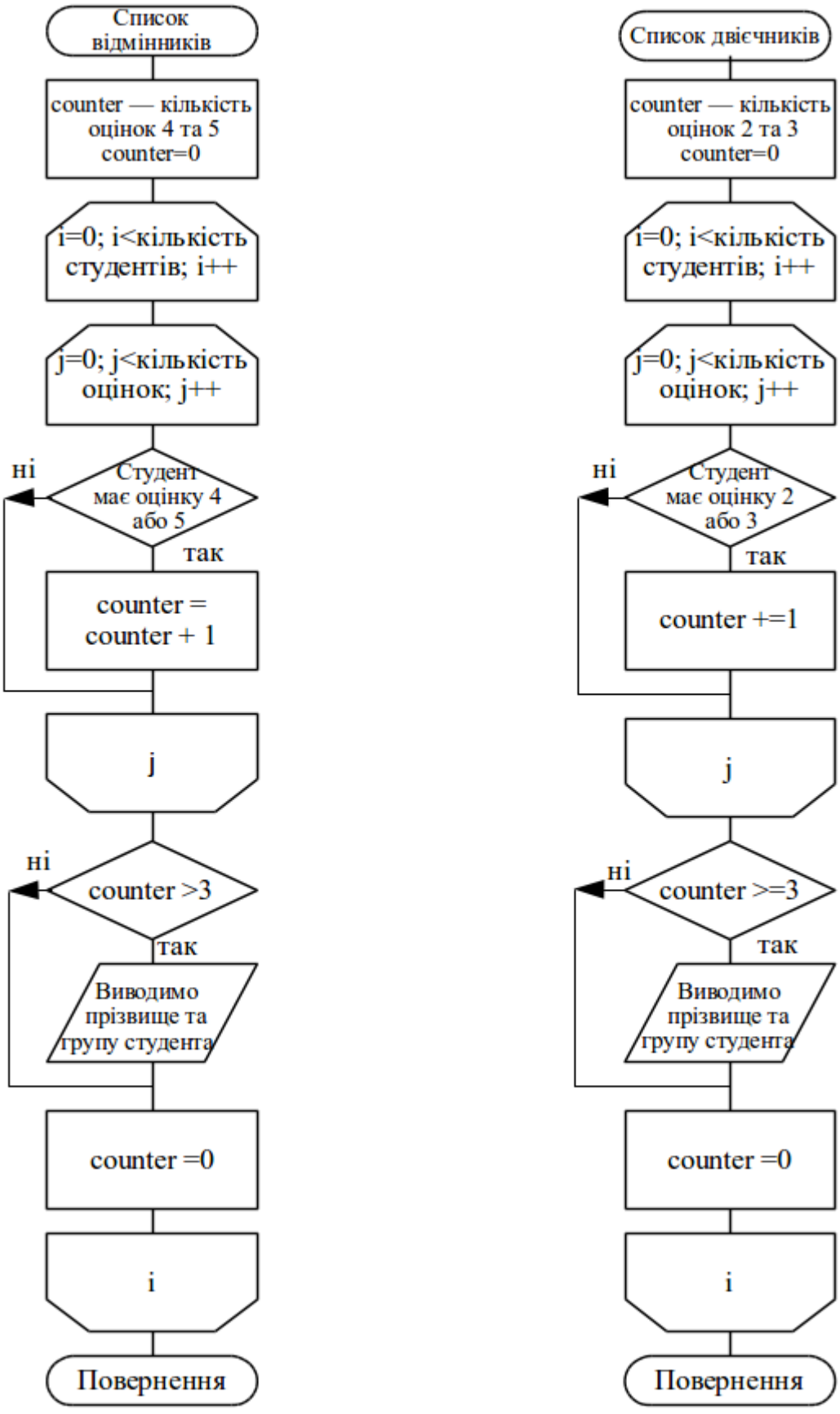


Рис. 12.2. Продовження схеми програми прикладу

Висновки

У даному розділі розглянуті наведені нижче основні питання:

– оголошення і визначення структур;

– масиви структур;

– багатофайлові проекти;

– використання заголовних файлів.

Контрольні питання

1. Що таке структура?
2. Як оголосити структуру?
3. Що таке елементи структури?
4. Як оголосити та ініціалізувати структурну змінну?
5. Яким чином отримати доступ до елементів структури?
6. Яким чином оголошується та ініціалізується масив структурних змінних?
7. Яким чином отримати доступ до масивів структури, використовуючи покажчик структурного типу?
8. Як створити багатофайловий проект?
9. Як додати заголовний файл до проекту?
10. Які дані можуть міститися в заголовному файлі?

13. ОРГАНІЗАЦІЯ СПИСКІВ

Навчальною метою розділу є ознайомлення студентів з організацією списків мови програмування С++.

У результаті вивчення даного розділу студенти повинні знати:

– визначення списку;
– різновиди списків;

уміти:

– застосовувати різні алгоритми роботи зі списками.

13.1. Список

Список – це сукупність однотипних даних, як правило розташованих у динамічній пам’яті. На відміну від масивів немає необхідності знати розмір списку, тобто кількість елементів, що входять до списку у момент його створення.

Список складається з елементів – вузлів, кількість яких необмежена (обмежується розміром динамічної пам’яті).

Як правило, елемент списку створюється в динамічній пам’яті та зв’язується з іншими елементами списку за допомогою своєї адреси. Наприклад, на рис. 13.1 показано розташування елементів масиву у пам’яті, а на рис. 13.2 – розташування елементів списку у пам’яті.

Кожен елемент списку складається з двох частин – даних та адреси наступного і/або попереднього елементу.

	mas[0]	mas[1]	mas[2]	mas[3]
mas	1	2	3	4
	0x0020	0x0024	0x0028	0x002c

Рис. 13.1. Зображення масиву у пам’яті

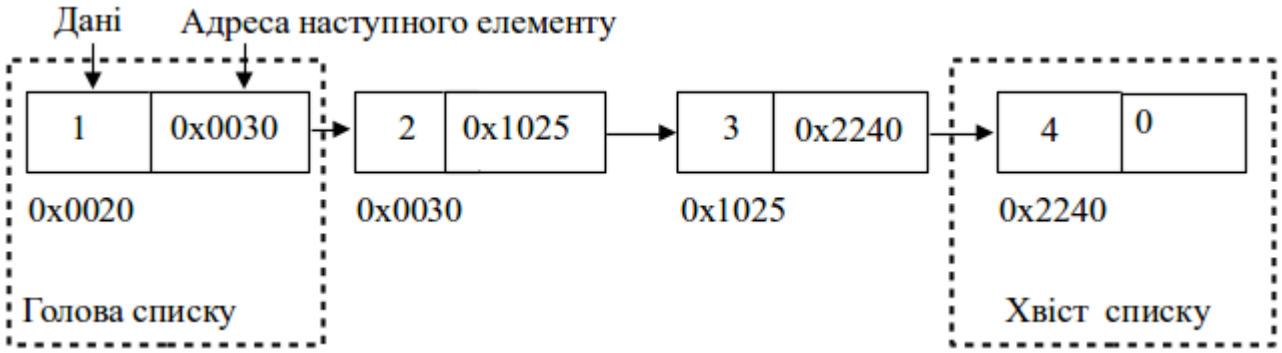


Рис. 13.2. Зображення списку у пам’яті

Як бачимо у масиві всі елементи розміщені в пам’яті один за одним, а у списку – довільно. Це тому, що для масиву пам’ять виділяється в момент його створення для всіх елементів одразу, а у списку – при створенні чергового елементу.

Тобто у список можна в будь-який момент додавати елементи, треба тільки зв’язати новий елемент з попередніми елементами у списку.

Список має голову – перший елемент (рис. 13.2) та хвіст – останній елемент. Якщо у списку лише один елемент, значення адреси наступного елементу дорівнює 0. В останнього елементу (хвоста) значення адреси також повинно дорівнювати 0.

13.2. Типи списків

Списки бувають односпрямовані, двоспрямовані, кільцеві.

Односпрямованим називається список, у якому кожен елемент має адресу наступного елементу (рис. 13.2).

Двоспрямований – це список, у якому кожен елемент має адресу наступного та попереднього елементу (рис. 13.3).

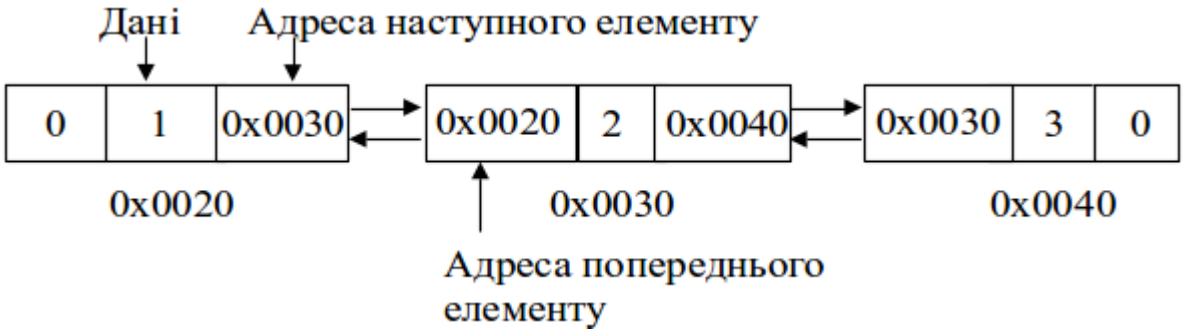


Рис. 13.3. Зображення двоспрямованого списку у пам’яті

Кільцевий – це список, у якому останній елемент має адресу першого (рис. 13.4).

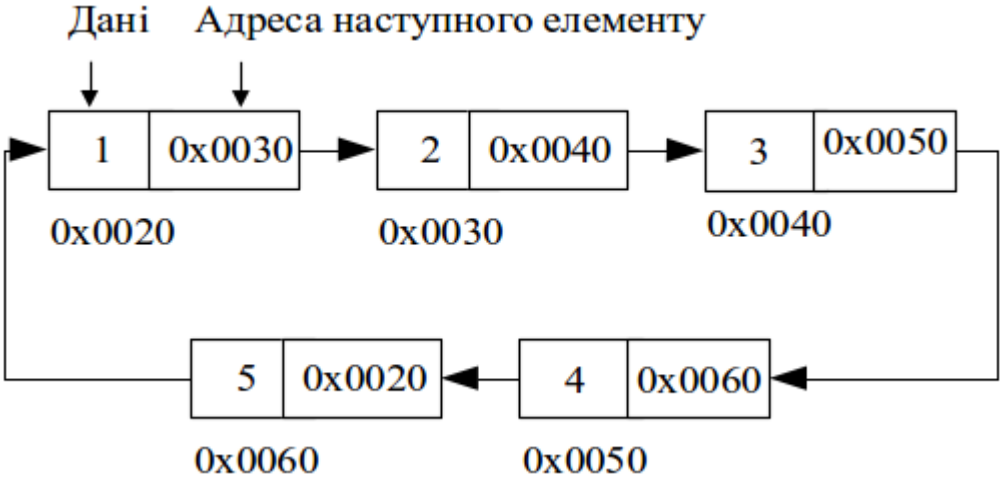


Рис. 13.4. Зображення кільцевого списку у пам’яті

13.3. Робота зі списками

Оскільки елемент списку складається з декількох різнотипних елементів (адреси і даних), а також, виходячи з того, що тип покажчика повинен збігатися з типом даних, на які він посилається, необхідно створити свій тип даних для елементу списку.

Для цього використаємо таку структуру:

```
struct node
{
  int  data;          // поле структури для зберігання даних вузла
  node *next;  // поле для зберігання адреси наступного елементу вузла
};
```

Після оголошення такої структури можна створювати вузли списку у динамічній пам'яті та зв'язувати їх у список.

```
node *head = new node;     // створюємо перший елемент списку
head->data = 1;            // записуємо дані в перший елемент списку
head->next = 0;            // записуємо адресу наступного елементу
node *temp = head;         // створюємо змінну для переміщення по списку
for (int i=1; i<10; i++)   // створюємо 9 вузлів списку
{
      temp->next = new node;  // створюємо вузол та зберігаємо його
                              // адресу у попередньому вузлі
      temp = temp->next;      // зберігаємо адресу нового вузла у змінній
                              // temp
      temp->data = i+1;       // записуємо дані в новий вузол
}
temp->next = 0;               // в останній елемент списку записуємо 0
                              // замість адреси наступного елементу
```

Таким чином, ми створили односпрямований список з 10 вузлів.

При роботі з односпрямованим списком необхідно зберігати у змінній (наприклад temp) адресу голови списку, інакше можна загубити список (не буде можливості звернутися до будь-якого вузла списку).

Якщо необхідно видалити елемент зі списку, треба не тільки видалити вузол із пам'яті, а і записати в поле адреси попереднього елементу відповідне значення.

Наприклад, видалимо 5-й вузол із створеного раніше списку (рис. 13.5). Спочатку необхідно від голови списку перейти до шуканого вузла, тобто 4-го.

```
temp = head;                        // прямуємо на початок списку
for (int i=1; i<4; i++)             // шукаємо 4-й вузол списку
      temp = temp->next;
node *del = temp->next;             // зберігаємо адресу вузла, що
                                    // необхідно видалити (5-го)
temp->next = temp->next->next;      // в поле адреси 4-го вузла
                                    // записуємо адресу 6-го
delete del;                         // видаляємо 5-й вузол
```

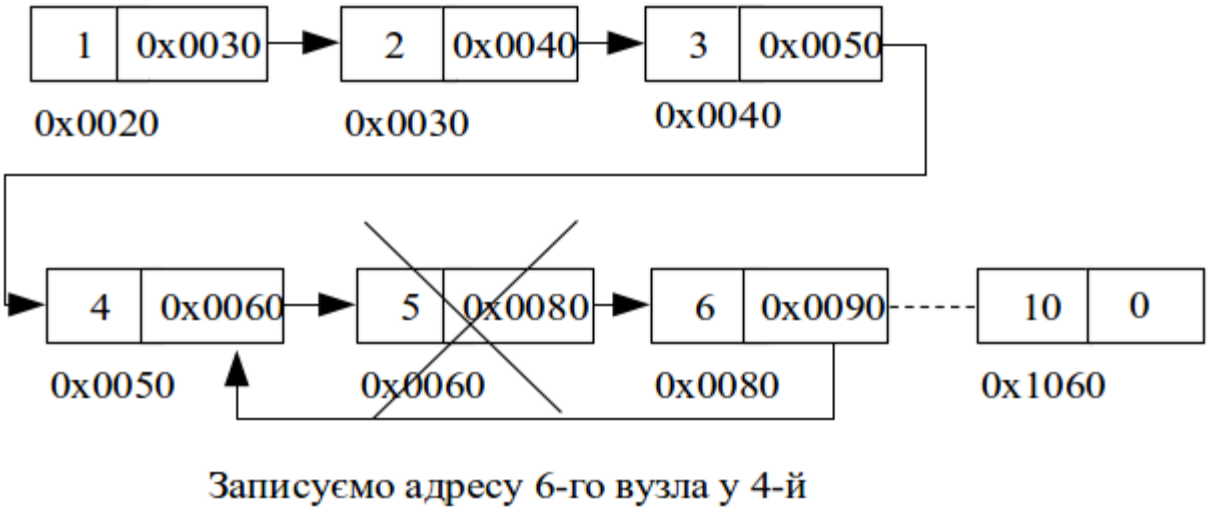


Рис. 13.5. Видалення вузла зі списку

Для того, щоб вивести на екран дані зі списку, необхідно вивести дані кожного вузла списку.

```
temp = head;              // прямуємо на початок списку
while (temp)              //в тому випадку, коли temp=0 — це хвост списку
{
        cout<<(temp->data)<<'\t';      // виводимо дані зі списку
        temp = temp->next;          // шукаємо вузол списку
}
```

Аналогічно виконуються й інші операції зі списками.

Висновки

У даному розділі розглянуті наведені нижче основні питання:

- визначення списку;
- типи списків;
- особливості роботи зі списками;
- коригування списків.

Контрольні питання

1. Що таке список?
2. З яких елементів складається список?
3. Як оголосити список?
4. Які типи списків бувають?
5. Що таке односпрямований список?
6. Що таке двоспрямований список?
7. Що таке кільцевий список?
8. Як створюється тип даних для елементу списку?
9. Що таке вузли списку?
10. Як видалити елемент списку?

14. ОРГАНІЗАЦІЯ ПОШУКУ І СОРТУВАННЯ

Навчальною метою розділу є ознайомлення студентів з організацією сортування і пошуку у мові програмування С++.

У результаті вивчення даного розділу студенти повинні знати:

- визначення лінійного пошуку;
- визначення двійкового пошуку;

уміти:

- застосовувати різні алгоритми сортування даних.

14.1. Пошук

Пошук, вставка і видалення – основні операції при роботі з даними. Розглянемо, як ці операції використовуються для найвідоміших об'єктів – масивів.

Існують різні способи пошуку елементів. Розглянемо лінійний і двійковий пошук на прикладі одновимірного масиву із семи елементів з числовими значеннями.

14.1.1. Лінійний пошук

Для того, щоб знайти в масиві потрібне число, можна використовувати *лінійний пошук*. Лінійний пошук полягає у такому: розглядаються по черзі всі елементи масиву, і визначається позиція потрібного нам елементу.

Тому кількість порівнянь буде дорівнювати кількості елементів у масиві, що не раціонально, оскільки для пошуку в масиві з великою кількістю елементів необхідно багато часу.

14.1.2. Двійковий пошук

Якщо відомо, що дані відсортовані, можна застосувати *двійковий пошук* (рис. 14.1).

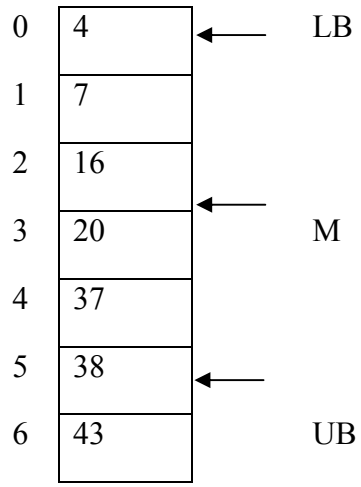


Рис. 14.1. Двійковий пошук

Змінні Lb і Ub містять відповідно верхню і нижню межі відрізка масиву, де знаходиться потрібний елемент. Починаємо завжди з дослідження середнього елементу відрізка.

Якщо шукане значення менше середнього елементу, переходимо до пошуку у верхній половині відрізка, де всі елементи менше тільки що перевіреного. Іншими словами, значенням Ub стає (M – 1) і на наступній ітерації працюємо з половиною масиву. Таким чином, у результаті кожної перевірки вдвоє звужуємо область пошуку.

Так, у нашому прикладі після першої ітерації область пошуку – всього лише три елементи, після другої залишається тільки один елемент. Таким чином, якщо довжина масиву дорівнює 6, нам достатньо трьох ітерацій, щоб знайти потрібне число.

Двійковий пошук – дуже могутній метод. Якщо, наприклад, довжина масиву дорівнює 1023, після першого порівняння область звужується до 511-ти елементів, а після другого – до 255. Легко порахувати, що для пошуку в масиві з 1023-х елементів достатньо 10 порівнянь.

14.2. Сортування

14.2.1. Сортування вставками

Один з простих способів відсортувати масив – *сортування вставками*.

Розглянемо сортування вставками на прикладі одновимірного масиву з 4 елементів. Перебір елементів починається з другого елементу (рис. 14.2, а). Другий елемент виймається (число 3), потім елементи, розташовані вище, зсуваємо вниз доти, поки не знайдемо місце, куди потрібно вставити 3. Тобто вниз зсуваються всі елементи, більші за елемент вставки (число 3). Потім переходимо до наступного (третього) елементу масиву (рис. 14.2, б) і шукаємо позицію для числа 1 за принципом, викладеним вище. Нарешті, (рис. 14.2, в) завершуємо сортування, помістивши четвертий елемент масиву (число 2) на потрібне місце.

Якщо довжина нашого масиву дорівнює *n*, потрібно пройтися по *n – 1* елементу. Кожного разу може виникнути потреба зсунути *n – 1* інших елементів. От чому цей метод вимагає дуже багато часу.

Сортування вставками належить до методів сортування *за місцем*. Іншими словами, йому не потрібна допоміжна пам’ять, отже, сортуємо елементи масиву, використовуючи тільки пам’ять, займану лише масивом. Крім того, цей метод є стійкім, і якщо серед сортовуваних ключів є однакові, після сортування вони залишаються в початковому положенні.

14.2.2. Сортування методом Шелла

Метод, запропонований Дональдом Л. Шеллом, є нестійким сортуванням за місцем. Ефективність методу Шелла пояснюється тим, що зсувані елементи швидко потрапляють на потрібні місця.

На рис. 14.3, а наведений приклад сортування вставками. Спочатку виймаємо 1, потім зсуваємо 3 і 5 на одну позицію вниз, після чого вставляємо 1. Таким чином, було потрібно два зсуви. Наступного разу використаємо знову

два зсуви, щоб вставити на потрібне місце 2. На весь процес витратимо 2+2+1=5 зсувів.

На рис. 14.3, б показано сортування методом Шелла. Починаємо сортування вставками з кроком 2. Спочатку розглядаємо числа 3 та 1: витягуємо 1, зсуваємо 3 на одну позицію з кроком 2, вставляємо 1.

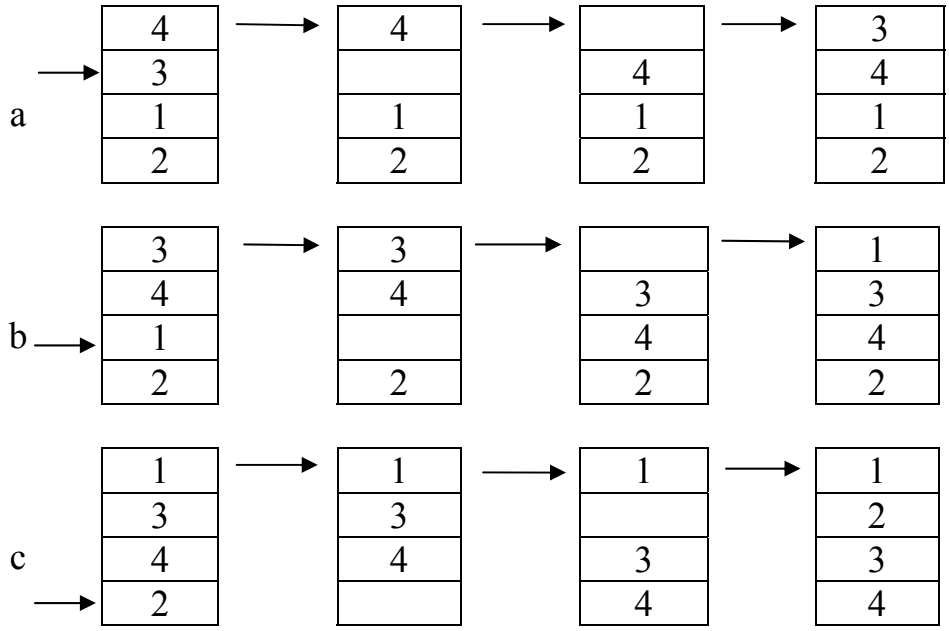


Рис. 14.2. Сортування вставками: a, b, c – перший, другий і третій кроки відповідно

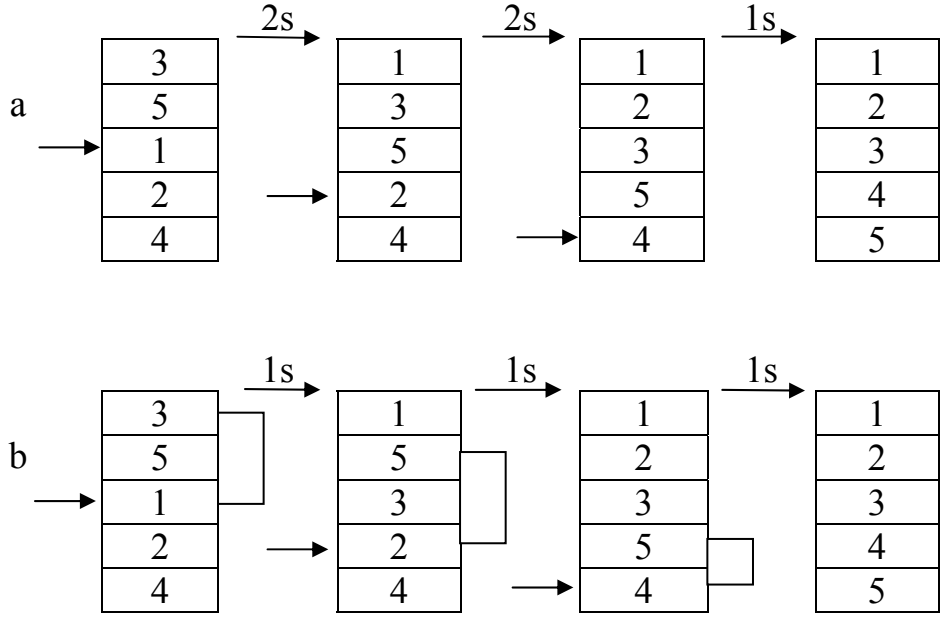


Рис. 14.3. Сортування методом Шела: a, b – перший і другий кроки відповідно

Потім повторюємо те саме для чисел 5 і 2: витягуємо 2, зсуваємо вниз 5, вставляємо 2 і т.д. Закінчивши сортування з кроком 2, проводимо його з кроком 1, тобто виконуємо звичайне сортування вставками.

Всього при цьому знадобиться 1 + 1 + 1 = 3 зсуви. Таким чином, використавши спочатку крок, більший за 1, добиваємося меншої кількості зсувів. Можна використовувати найрізноманітніші схеми вибору кроків.

Як правило, спочатку сортуємо масив з великим кроком, потім зменшуємо крок і повторюємо сортування. У самому кінці сортуємо з кроком 1. Хоча цей метод легко пояснити, його формальний аналіз досить важкий. Зокрема, теоретикам не вдалося знайти оптимальну схему вибору кроків.

Формула вибору кроків *h* для масиву довжини *N*:

у послідовності $h_1 = 1,\ h_{s+1} = 3h_s + 1$ узяти h_t, якщо $h_{t+2} \geq N$.

Ось декілька перших значень *h*:

$$h_1 = 1;$$
$$h_2 (3 \cdot 1) + 1 = 4;$$
$$h_3 (3 \cdot 4) + 1 = 13;$$
$$h_4 (3 \cdot 13) + 1 = 40;$$
$$h_5 (3 \cdot 40) + 1 = 121.$$

Щоб відсортувати масив завдовжки 100, перш за все знайдемо номер *s*, для якого $h_s \geq 100$. Згідно з наведеними формулами *s* = 5. Потрібне значення знаходиться двома рядками вище. Таким чином, послідовність кроків при сортуванні буде такою: 13-4-1. Звичайно, не потрібно зберігати цю послідовність: чергове значення *h* знаходиться з попереднього за формулою

$$h_{s-1} = (h_s - 1) / 3.$$

14.2.3. Швидке сортування

Цей алгоритм був розроблений Е. Хоаром. В алгоритмі швидкого сортування використовуються три ідеї:

– розділення сортованого масиву на 2 частини, ліву і праву;
– взаємне впорядкування двох частин (підмасивів) так, щоб всі елементи лівої частини не перевищували елементи правої;
– рекурсія, при якій підмасив упорядковується таким самим способом, як і весь масив.

Для розділення масиву на дві частини (підмасиви) потрібно вибрати деяке „бар'єрне" значення ключа. Це значення повинне задовольняти єдину умову: лежати в діапазоні значень для даного масиву (тобто між мінімальною і максимальною величиною).

За „бар'єр" можна вибрати значення ключа будь-якого елементу масиву, наприклад, першого або останнього, або того, що знаходиться усередині.

Далі потрібно зробити так, щоб у лівому підмасиві опинилися всі елементи з ключем, меншим за бар'єр, а в правом – більшим.

Потім, проглядаючи масив зліва направо, необхідно знайти позицію першого елементу з ключем, більшим за бар'єр, а переглядаючи справа наліво – знайти перший елемент з ключем, меншим за бар'єр. Слід поміняти ці

значення, потім продовжити зустрічний рух до наступної пари елементів, призначених для обміну.

Необхідно повторювати цю процедуру, поки між індексами лівого і правого переглядів не відбудеться збіг. Місце збігу стане межею між двома взаємно впорядкованими підмасивами.

Далі алгоритм рекурсивно застосовується до кожного з підмасивів (лівого і правого). Кінець кінцем приходимо до сукупності з *n* взаємно впорядкованих одноелементних масивів, які ділити далі неможливо. Ця сукупність утворює один повністю впорядкований масив. Сортування завершене.

Висновки

У даному розділі розглянуті наведені нижче основні питання:

- лінійний пошук;
- сортування вставками;
- сортування методом Шелла;
- швидке сортування.

Контрольні питання

1. Які основні операції застосовуються при роботі з даними?
2. У чому полягає лінійний пошук?
3. У чому полягає двійковий пошук?
4. Коли застосовують двійковий пошук?
5. Доведіть, що для двійкового пошуку в масиві з 1023-х елементів достатньо 10 порівнянь.
6. Які відомі способи сортування?
7. Який спосіб сортування є найпростішим?
8. У чому полягає спосіб швидкого сортування?
9. Розкажіть про спосіб сортування Шелла?
10. У чому полягає сортування вставками?
11. Який спосіб сортування розробив Е. Хоар?
12. Розкажіть про термін „бар’єр”, що вводиться при сортуванні?
13. При якому способі сортування вводиться термін „бар’єр”?
14. Який спосіб сортування належить до методів сортування за місцем?
15. Які методи сортування є стійкими за місцем?

ЧАСТИНА 2. ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ
15. ОСНОВНІ ПОНЯТТЯ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ

Навчальною метою розділу є ознайомлення студентів з основними поняттями об'єктно-орієнтованого програмування мови С++.

У результаті вивчення даного розділу студенти повинні знати:

– визначення об'єктно-орієнтованого програмування;
  визначення класів і об'єктів класу;

уміти:

– навести приклади конструкторів і деструкторів.

15.1. Класи

Класом у мові С++ називається тип даних, визначений користувачем.

Концепція класу призначена для того, щоб надати програмісту інструмент для створення нових типів, таких же зручних в обігу, як і вбудовані типи. В ідеалі клас повинен відрізнятися від вбудованого типу тільки способом створення, але не способом використання.

З класом в С++ зв'язуються чотири поняття:

1. Масив *елементів-даних*, які є представниками класу. У класі може бути будь-яке число (і нуль теж) елементів-даних довільного типу. Елементи-дані можуть називатися також даними-членами.

2. Масив *функцій-елементів*, що являють собою набір операцій, які можна застосовувати до об'єктів даного класу. У класі може бути будь-яке число (і нуль теж) функцій-елементів. Вони називаються інтерфейсом класу. Функції-елементи можуть називатися також функціями-членами або методами.

3. *Рівні доступу*. При визначенні нового типу основна ідея – відокремити неістотні подробиці реалізації від тих якостей, які істотні для його правильного використання. Елементи класу можна оголошувати як закриті (приватні, private), захищені (protected) і відкриті (загальні, public). Це дозволяє визначати в програмі рівні доступу до елементів. Звичайно зображення класу є закритим, а операції, які можна з ним виконувати, вважаються відкритими. Таке застосування оголошень закритих і відкритих елементів називається приховуванням даних. Говорять, що клас інкапсулює закрите внутрішнє зображення.

4. *Пов'язане з класом ім'я*, яке служить для визначеного користувачем класу специфікацією типу. Ім'я класу може з'явитися в будь-якому місці програми, де допустимо ім'я типу. Клас із закритим уявленням і відкритим набором операцій називається абстрактним типом даних.

Визначення класу містить у собі з дві частини: заголовок класу, який складається із службового слова class і наступного за ним імені класу, і тіло класу, поміщеного у фігурні дужки. За тілом класу може йти крапка з комою або список описів. Наприклад:

```
class Screen { //...
};
```

```
class Screen { //...
} s1, s2;
```

У тілі класу вказуються елементи-дані, функції-елементи і задається вид приховування даних.

15.2. Елементи-дані

Оголошення елементів-даних класу таке саме, як і оголошення змінних, за винятком того, що в класі не можна задавати явні ініціалізатори. Наприклад, клас Screen може бути записаний так:

```
class Screen {
  int height;       // висота
  int width;        // ширина
  char *cursor;         // змінна позиція курсору
  char *screen;         // масив екрана (висота * ширина)
};
```

Як і у випадку змінних, можна записати коротше:

```
class Screen {
  int height, width;
  char *cursor, *screen;
};
```

Тип елементів-даних може бути довільним. Все це аналогічно визначенню елементів структури.

15.3. Функції-елементи

Функції-елементи класу задають операції, допустимі для об’єктів такого класу.

Функція-елемент класу повинна оголошуватися в тілі класу. Її оголошення складається з прототипу функції:

```
class Screen {
public :
  void home();
  void move(int, int);
  char get();
  char get(int, int);
  void checkRange(int, int);
};
```

У тілі класу можна давати і визначення функції:

```
class Screen {
public :
  void home() { cursor = screen; }
  char get()  { return *cursor; }
};
```

Якщо визначення функції знаходиться в тілі класу, то така функція автоматично розглядається як функція-підстановка (inline).

Якщо функція-елемент більше одного-двох рядків, краще визначати її поза тілом класу. При цьому перед ім'ям у заголовку потрібно вказати ім'я класу:

```
void Screen::checkRange(int row, int col)
{
 if (row < 1 || row > height || col < 1 || col > width )
   {
    cerr << "координати за межами екрана \n";
    exit(-1);
   }
}
```

Якщо функція-елемент визначена поза тілом класу, то, щоб бути підстановкою, вона повинна вказати це явно за допомогою inline:

```
inline void Screen::move(int r, int c)
{
 checkRange(r,c);
 // поставити курсор згідно координат r,c
}
```

Функція-елемент, на відміну від звичайних функцій, володіє такими властивостями:

1. Вона має всі права доступу як до відкритих, так і до закритих елементів свого класу, тоді як у звичайній функції доступні тільки відкриті елементи класу. Функція-елемент одного класу взагалі немає доступу до елементів іншого класу.

2. Функція-елемент знаходиться в тій самій області видимості, що і її клас. Звичайні функції мають файлову область видимості. Це означає, що функція-елемент невидима поза областю видимості її класу.

3. Функція-елемент може використовуватися для перевантаження тільки іншої функції-елементу того ж класу (як get() у класі Screen), оскільки всі перевантажені функції повинні мати одну область видимості.

15.4. Приховування даних і рівні доступу

Приховування даних – це засіб мови, який обмежує доступ до внутрішній будови класу. Воно задається за допомогою розбиття тіла класу на відкриту, закриту і захищену частини. Кожна така частина починається із службового слова: public – для відкритої, private – для закритої і protected – для захищеної частини. Після службового слова повинна йти двокрапка. Елементи, описані у відповідній частині, стають відкритими, захищеними або закритими.

Кожному такому елементу відповідає певний рівень доступу:

1. Відкритий елемент доступний у будь-якій частині програми. У класі з приховуванням даних відкритими елементами є тільки функції, вони визначають інтерфейс класу.

2. Захищений елемент поводиться як відкритий елемент по відношенню до похідного класу і як закритий елемент по відношенню до всієї решти програми.

3. Закритий елемент доступний тільки у функції, яка є елементом або “другом” даного класу. Щоб забезпечити приховування даних, елементи-дані повинні оголошуватися в класі як закриті.

Наприклад:

```
class Screen {
public :
  void home();
  void move(int, int);
  char get();
  char get(int, int);
  //...
private :
  int height, width;
  char *cursor, *screen;
};
main() {        // або де-небудь, окрім елементів або “друзів” класу Screen
Screen s;
s.height = 200;   // height є закритим елементом і недоступний
}
```

У класі може бути декілька відкритих, захищених і закритих розділів. Розділ триває до наступного службового слова іншого розділу (public, private або protected) або до фігурної дужки, що закінчує визначення класу. Якщо службове слово не вказане, то за умовчанням розділ, що йде відразу після розкритої фігурної дужки, вважається закритим.

15.5. Визначення об’єктів класу

Визначення класу не викликає виділення якої-небудь пам’яті. Пам’ять виділяється у міру визначення об’єктів класу.

Так, після визначення

```
Screen my_screen;
```

буде відведена пам’ять під чотири елементи-дані класу Screen.

Об’єкти одного і того ж класу можна ініціалізувати і привласнювати один одному. Стандартне копіювання об’єкта зводиться до копіювання всіх його елементів:

```
Screen s2 = my_screen;
```

За умовчанням об’єкти класу передаються як параметри відповідно до значень і таким же чином повертаються як результат функції.

Покажчик на об’єкт класу можна ініціалізувати як у результаті взяття адреси, так у результаті застосування операції new. Присвоєння йому виконується аналогічно:

```
Screen *ptr = new Screen;
```

```
my_screen = *ptr;
ptr = &s2;
```

Поза тілом класу для доступу до елементу, що є функцією або даними, потрібна операція вибору елементу . або -> (. застосовується до об'єкта класу або посилання на нього, а -> – до покажчика на об'єкт класу):

```
ptr->move(10,10);
my_screen.home();
```

Усередині тіла класу та всередині тіла будь-якої функції-елементу (яке вважається таким, що теж належить тілу класу) елементи класу можна використовувати безпосередньо без операції вибору елементу:

```
void screen::home()
{
  cursor = screen;
}
```

15.5.1. Конструктори

У мові C++ звичайно об'єкти класу повинні ініціалізувалися таким чином: всякий раз, коли об'єкт класу визначається або розміщується в динамічній пам'яті за допомогою операції new, транслятор створює неявний виклик особливої функції-елементу, що називається конструктором.

Конструктор – це визначена користувачем функція ініціалізації, ім'я якої збігається з ім'ям класу.

Наприклад:

```
class Word {
public:
  Word (const char *, int =0);  // конструктор
private:
  int count;
  char *string;
};

inline Word::Word(const char *str, int cnt)
 // визначення конструктора
{
  string = new char[strlen(str)+1];
  strcpy(string, str);
  count = cnt;
}
```

Для конструктора не потрібно вказувати тип значення, що повертається і не потрібно повертати значення явно. В інших випадках його визначення таке саме, як і для звичайної функції-елементу. У нашому випадку конструктор має один обов'язковий параметр типу char * і один необов'язковий типу int.

Ось приклади визначення об’єкта класу Word з розрахунку на його конструктор:

```
 // явні виклики конструктора
   Word w1 = Word("abcd");            // виклик Word::Word("abcd",0);
   Word *pw = new Word("defg",1);  // виклик Word::Word("defg",1);
// неявні виклики конструктора
   Word w2("hklm");                   // виклик Word::Word("hklm",0);
   Word w2 = "strf";                  // виклик Word::Word("strf",0);
```

Якщо клас у C++ має один конструктор, то він викликатиметься неявно при ініціалізації об’єктів даного класу.

Звичайно клас має декілька конструкторів з різними параметрами, тобто конструктор можна перевантажувати. Всі правила ототожнення для перевантажених функцій справедливі і для конструкторів.

Наведемо приклад класу String, який є рядком символів з довжиною, що зберігається:

```
   class String {
   public :
     String(int);
     String(const char *);
   private :
     int len;
     char *str;
   };
```

У цьому класі визначено два конструктори: один для ініціалізації len, другий для str.

Ось приклад визначення конструктора для str:

```
    #include<string.h>
    String::String( const char *s )
    {
      len = strlen(s);
      str = new char[len+1];
      strcpy(str,s);
    }
```

Цей конструктор неявно викликатиметься кожного разу, коли в програмі визначається об’єкт класу String, що ініціалізувався рядком символів.

Ось приклад конструктора для ініціалізації len (під рядок відводиться пам’ять завдовжки len):

```
    String::String( int ln )
    {
      len = ln;
      str = new char[len+1];
      str[0] = '\0';
    }
```

Визначення об’єкта класу проходить повний контроль типів відносно виклику конструктора. Три такі визначення об’єкта класу String недопустимі:

```
int len = 1024;
String s1;              // помилка – немає параметра
String s2(&len);        // помилка – невідповідність типу int *
String s3("abcd",7);    // помилка – два параметри
```

Існують дві короткі (неявні) форми виклику конструктора:

```
String s("abcd");       // переважно
String s2 = "abcd";     // можливо для конструктора з одним параметром
```

15.5.2. Деструктори

Особлива, призначена для користувача функція-елемент, яка називається *деструкцією,* автоматично викликається кожного разу, коли:

- об’єкт класу виходить з поточної області видимості;
- операція delete застосовується до покажчика класу.

Якщо з поточної області видимості виходить посилання на об’єкт класу, то деструкція не викликається.

Деструкція – це функція-елемент з ім’ям класу, якому передує символ "~".

Наприклад:

```
class String {
public :
  ~String();     // деструкція
};
```

Деструкція не може мати параметрів і означає “не може бути перевантажена” (вона повинен бути у класі одна). Для неї не можна вказувати тип значення, що повертається або використовувати оператор return із значенням.

Ось приклад визначення деструкції:

```
String::~String() { delete str; }
```

Якщо конструктори в C++ можуть викликатися явно, а можуть і неявно, то деструкції практично завжди викликаються неявно.

Сама деструкція не звільняє пам’ять. Вона робить швидшою підготовку до її звільнення – це дії, що протилежні діям у конструкторі. Пам’ять звільняється після виклику деструкції.

У деструкції можна виконати будь-яку операцію, яка може бути необхідною після завершення роботи з об’єктом.

Висновки

У даному розділі розглянуті наведені нижче основні питання:

– визначення класів;
– визначення об’єктів класу;
– приховування даних і рівні доступу;
– конструктори і деструктори.

Контрольні питання

1. Дайте визначення класу в мові С++.
2. Які поняття в мові С++ зв’язуються з класом?
3. Скільки понять у мові С++ зв’язуються з класом?
4. Із скількох частин складається визначення класу в мові С++?
5. Як оголошуються елементи-дані класу?
6. Що задають функції-елементи класу?
7. Як оголошуються функції-елементи класу?
8. Якими властивостями володіє функція-елемент класу?
9. Для чого застосовується засіб приховування даних?
10. На які частини розбивають тіло класу?
11. Які рівні доступу назначають частинам тіла класу?
12. Як визначають об’єкти класу?
13. Що таке конструктор у мові С++?
14. Що означає в мові С++ деструктори і деструкція?
15. Наведіть приклад визначення деструкції.

16. ЗНАЙОМСТВО З MFC

Навчальною метою розділу є ознайомлення студентів із засобами автоматизованого створення прикладних програм компілятора Microsoft Visual C++.

У результаті вивчення даного розділу студенти повинні знати:

- засоби автоматизованого створення застосувань;
  поняття повідомлення;
- карти повідомлень;

уміти:

- створювати “діалогове вікно”.

16.1. Структура Windows-програм, створених на базі MFC

У компілятор Microsoft Visual C++ включені засоби автоматизованого створення прикладних програм. Вони називаються MFC AppWizard.

MFC – (Microsoft Foundation Class Library) базовий набір (бібліотека) класів, написаних мовою C++ і призначених для спрощення і прискорення процесу програмування для операційної системи Windows.

Усі Windows-програми починають виконання з виклику функції WinMain(). Функція WinMain() програми MFC захована всередині самої бібліотеки MFC.

Прикладні програми на базі MFC мають як мінімум два об’єкти – прикладна програма (похідний від класу CWinApp) і головне вікно (похідний від класу CWnd).

Об’єкт прикладна програма є глобальним і створюється ще до початку роботи функції WinMain().

Створення екземпляра головного вікна і його відображення здійснюється у функції CProjectNameApp :: InitInstance(). Головне вікно відображається за допомогою функції CDialog :: DoModal(), яка до того ж обробляє всі повідомлення, що направляються у вікно, доти, поки користувач не натисне кнопку OK або Cancel.

Робота Windows-програми зводиться до очікування й обробки подій. Подією вважається будь-яка зміна ресурсів, якими володіє вікно.

Так, до події можна віднести натиснення на кнопку миші, коли курсор знаходиться в області вікна, спрацьовування таймера вікна, натиснення на кнопку виходу із програми. Всі події обробляє об’єкт-вікно. Оскільки об’єкти є класами даних, то можна ще раз відзначити, що основний алгоритм програми під Windows не виконує ніяких значущих дій. Всі дії виконуються внутрішніми методами об’єкта.

Розглянемо, як об’єкт “діалогове вікно” обробляє події. Спочатку саме по собі діалогове вікно містить усього три повідомлення про події, доступні користувачу, – ініціалізацію, контури вікна і таймер вікна, пов’язаний із системним таймером.

Для цих подій вже існують стандартні функції-обробники, які за умовчанням виконують ряд дій, необхідних для коректної роботи програми.

При потребі користувач може перехопити повідомлення про події і перевизначити їх обробники. Але тоді необхідний обов'язковий додатковий виклик обробників, які не перевизначені.

Оскільки створене користувачем діалогове вікно є нащадком класу CDialog загального предка діалогових вікон у бібліотеці MFC, то завдяки спадкоємству виклик початкового обробника простий:

CDialog::Обробник();

Перевизначення вказаних обробників подій потрібне рідко, оскільки основна робота йде не із самим вікном, а з компонентами, які розміщені у вікні: кнопкою, перемикачем, рядком редагування, списком і т. д. Усі ці компоненти, будучи полями класу вікна, який визначений користувачем, у свою чергу самі є об'єкти класів, визначених у MFC або користувачем. Тому, якщо подія належить до компоненти вікна, наприклад, клацання кнопки миші, коли курсор знаходиться в діалоговому вікні на кнопці цього вікна, то воно передається компоненті (кнопці), яка викликає свій обробник (прорисовування натиснення).

З іншого боку, компонента, для якої була передана подія, можливо, після його попередньої обробки повертає повідомлення про подію своєму об'єкту-власнику – діалоговому вікну. При цьому повідомлення від компоненти для вікна також є подією, яку необхідно обробити, і, отже, користувач повинен написати функцію-обробник, що є методом класу діалогового вікна користувача.

Написання функцій-обробників – основна робота програміста на Visual C++ для Windows.

Варто сказати декілька слів про **карту повідомлень**. Вона містить зв'язаний список повідомлень про різні події, який також включає імена компонент, що посилають повідомлення, та імена обробників, визначуваних користувачем. Імена компонент і подій насправді є цілочисельними константами-ідентифікаторами, значення яких визначаються засобами Visual C++ автоматично.

З іншого боку, карта повідомлень є функцією, написаною за допомогою макросів карти повідомлень Visual C++, в основі якої лежить оператор switch(), що здійснює вибірку викликів функцій-обробників подій залежно від повідомлення. Карта повідомлень викликається під час роботи програми в циклі обробки повідомлень, описаному в WinMain().

16.2. Створення проекту програми "діалогове вікно"

Для створення програми "діалогове вікно" необхідно виконати такі дії (рис. 16.1):

1. Вибрати в основному меню пункт File/New/Project.
2. У вікні, що з'явилося, вибрати вікно Projects types, на якому вибрати пункт Visual C++/ MFC.
3. У вікні Templates вибрати пункт MFC Application.
4. У рядку Name вказати ім'я проекту (ім'я програми). При цьому в рядку Location вказати повний шлях до теки, де буде зберігатися проект (кожен проект повинен зберігатися в окремій теці).

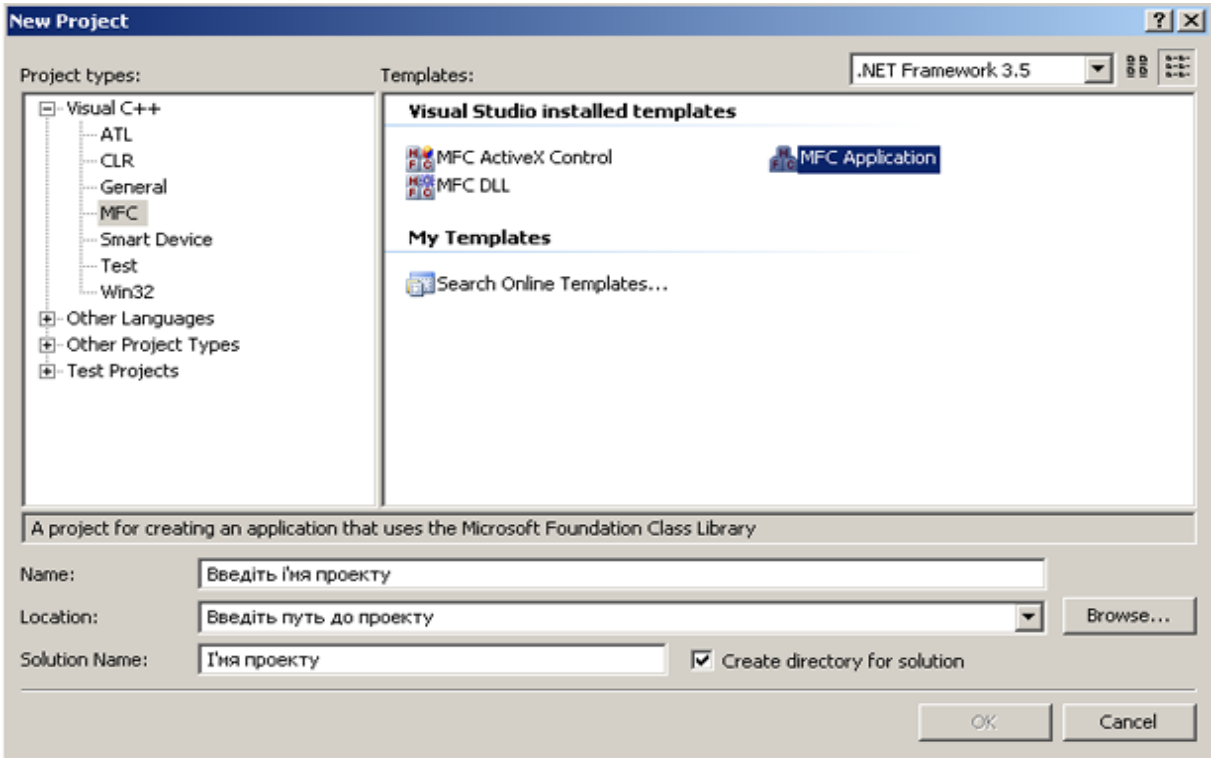


Рис. 16.1. Створення проекту

5. Натиснути OK.

6. У наступному вікні на вкладці Application Type встановити прапор Dialog based (побудова діалогової програми) і прибрати прапор Use Unicode libraries (рис. 16.2).

7. Натиснути Finish.

8. У діалоговому вікні, що з'явилося, видалити запрошення "TODO: Place dialog controls here", скомпілювати і запустити програму (рис. 16.3).

Розглянемо елементи розробки, що надаються середовищем. Для переміщення між файлами використовують вікно Solution Explorer. Ресурсами проекту є Resource View, класами – Class CView (рис. 16.4).

Class View містить посилання на всі змінні і класи, використовувані в проекті. Resource View містить посилання на всі ресурси проекту (в даному випадку інтерес становлять графічні ресурси). Solution Explorer – посилання на всі файли проекту.

При розробці програми «діалогове вікно» за допомогою майстра створюють три класи:

- CAboutDlg – клас діалогу для вікна About;
- CProjectNameApp – клас для прикладної програми в цілому, породжений CWinApp;
- CProjectNameDlg – клас діалогу для програми в цілому.

Діалогові вікна мають два асоційовні програмні об'єкти: ресурс і об'єкт. Ресурс зберігає властивості діалогового вікна, подібні до елементів керування, разом з їх екранними позиціями. Об'єкт є екземпляром класу, одержаного з CDialog. Об'єкт успадковує дані та методи від CDialog і його предків.

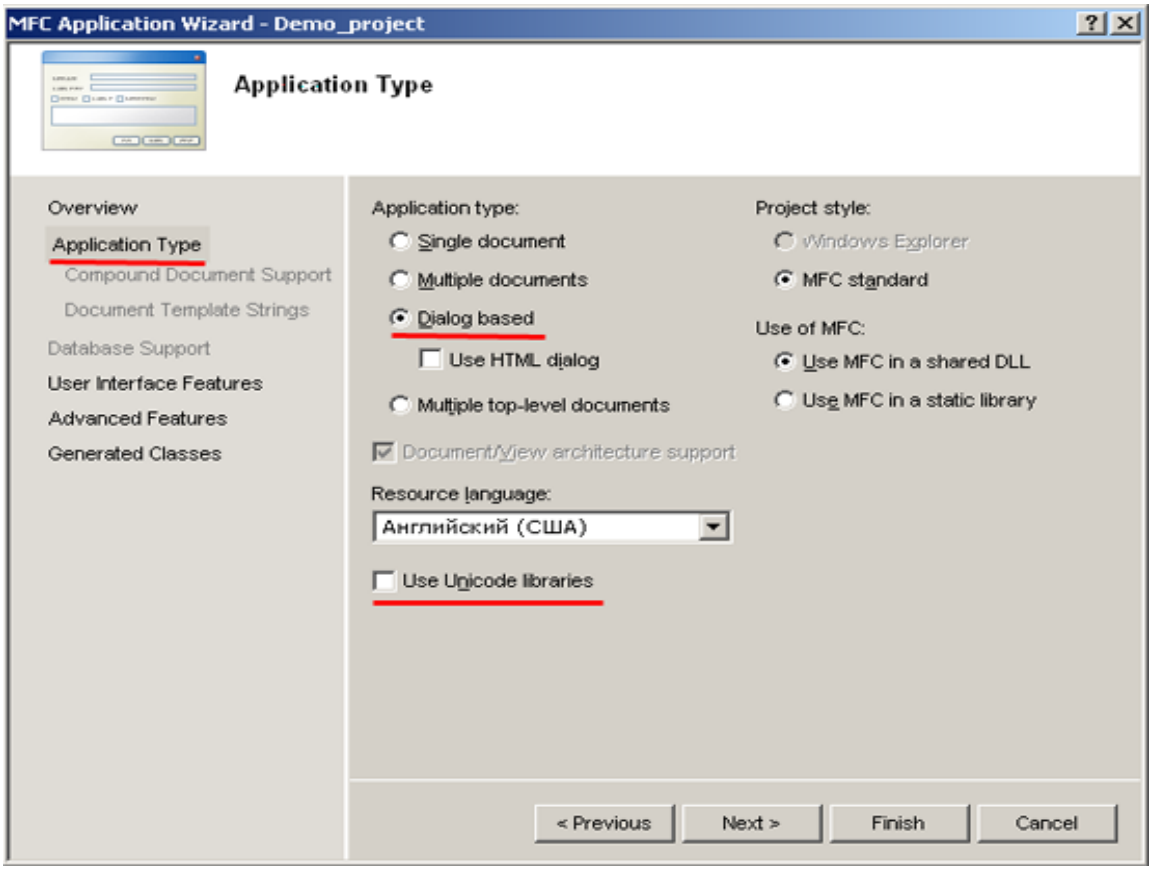


Рис. 16.2. Вибір типу проекту

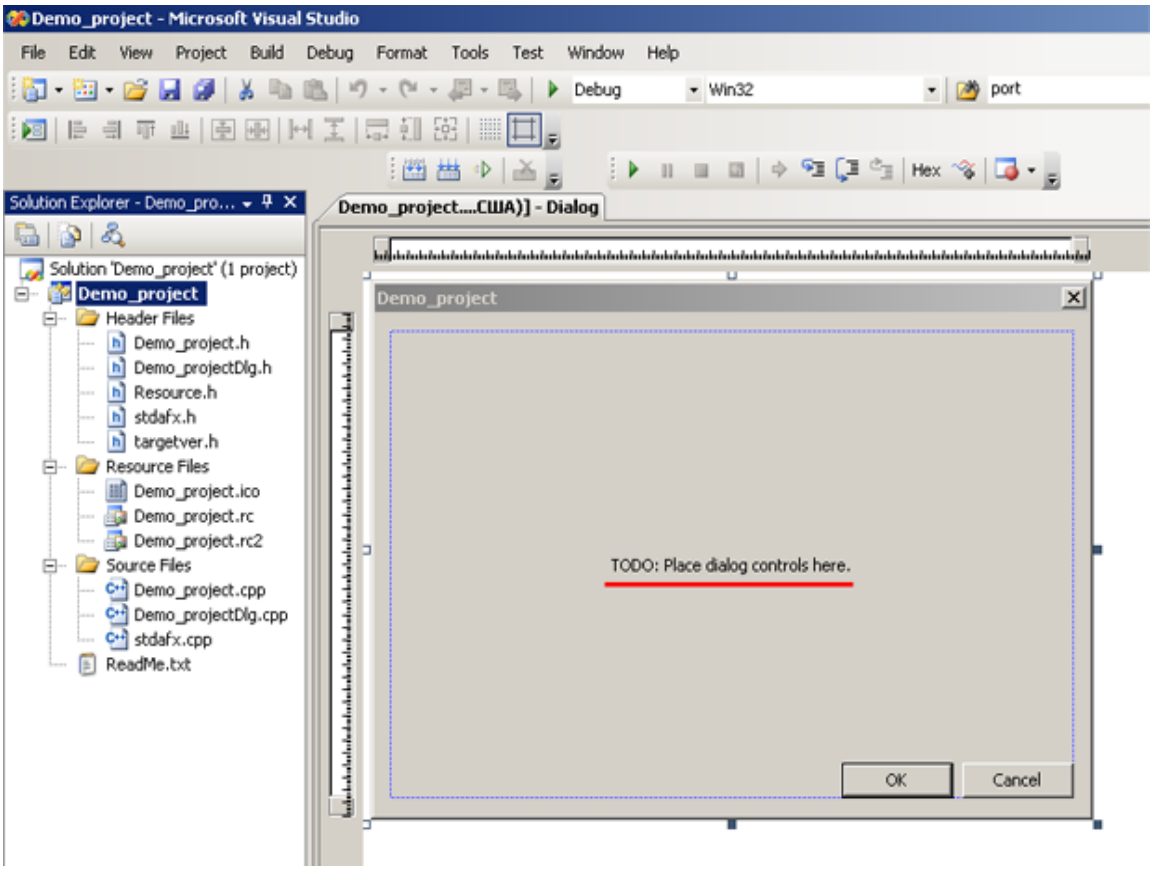


Рис. 16.3. Створений проект

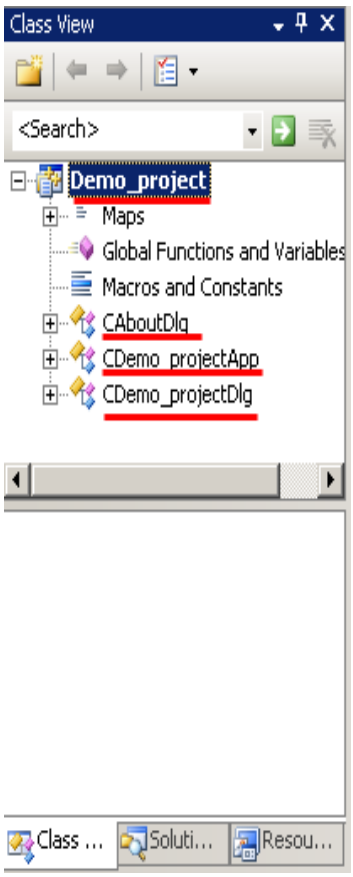


Рис. 16.4. Вікно Class View

16.3. Елементи керування і їх обробники

Елементи керування – спеціалізовані окремі віконні об'єкти, які формують частину призначеного для користувача інтерфейсу. Елементи керування є самостійними елементами, але вони не можуть існувати самі по собі, вони повинні міститися всередині іншого вікна (батьківського).

Відомі наведені далі стандартні елементи керування.

Статичний текст (Static text) – призначений для відображення інформації.

Цей елемент просто відображає текст для користувача.

Вікно редагування (Edit Box) – надає користувачу можливість відредагувати одну або декілька рядків текстових даних.

Кнопка (Button) – призначена для запиту інформації. Використовується для того, щоб дати програмі можливість реагувати на ініційовану користувачем дію.

Список (List Box) є модифікатор інформації, відображає список варіантів вибору.

Поле із списком (Combo Box) – дає можливість користувачу вибрати один з елементів списку.

Прапорець (Check Box) – дає можливість користувачу вибирати логічний стан («істина», «хибність», «не визначено»).

Кнопка-перемикач (Radio Button) – дає користувачу можливість взаємовиключного вибору з кінцевого числа варіантів, відомих на етапі проектування програми.

Значна частина кожної програми Windows складається з елементів, які не є програмним кодом: растрові зображення, курсори, піктограми, діалогові вікна і рядки. Ці елементи відомі як *ресурси*. Ресурси – це спеціальний вид даних, що ініціалізували, призначених тільки для читання.

Для додавання елементів керування і їх обробників необхідно виконати такі дії:

– у Resource View у теці Dialog знайти посилання на створене діалогове вікно (IDD_ PROJECT_NAME_1), натиснути на ньому лівою кнопкою миші та у вікні Properties вибрати мову (Language) – російська (рис. 16.5);

– активізувати створене діалогове вікно (лівою кнопкою миші натиснути два рази на ім'я діалогу);

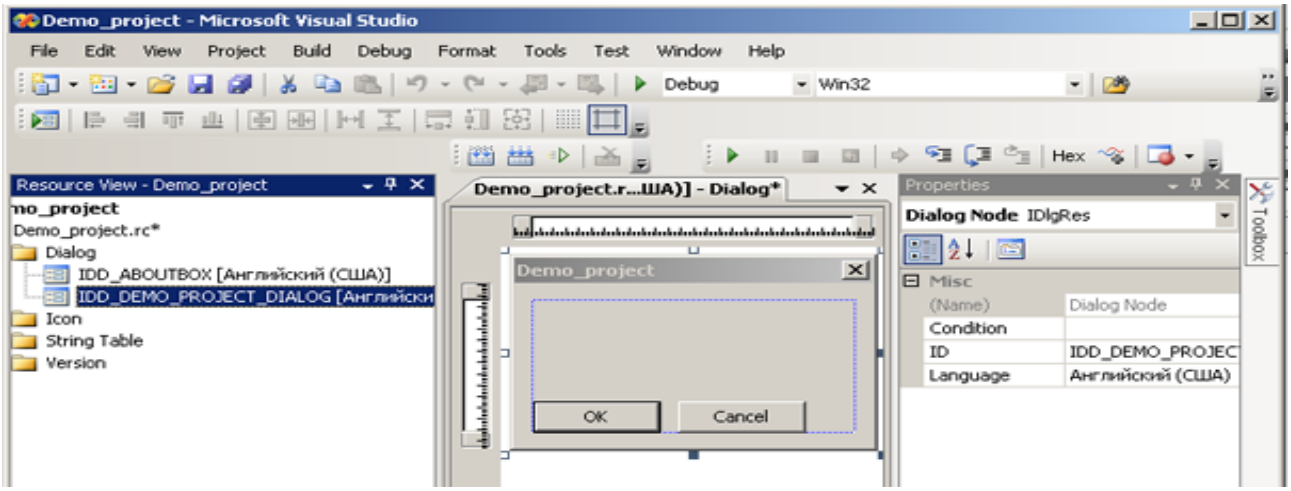


Рис. 16.5. Ресурси діалогового вікна

– активізувати меню “Toolbox” (праворуч). Після цих дій з’явиться панель з набором керівних компонент (рис. 16.6);

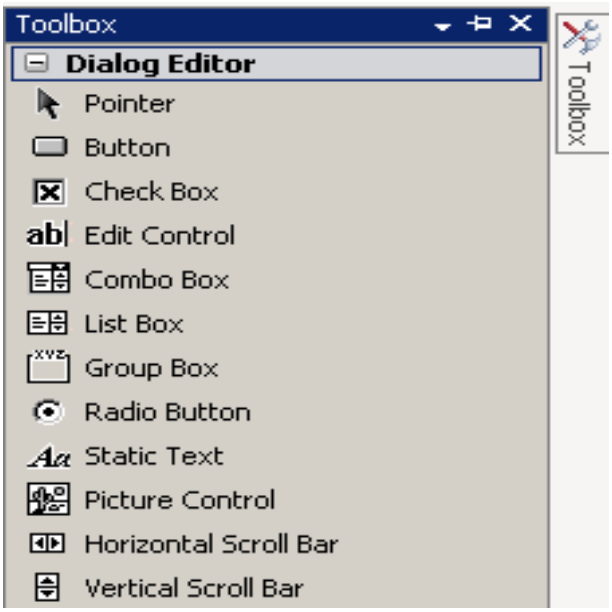


Рис. 16.6. Вікно Toolbox

– перетягнути елемент керування з панелі керівних компонент на створюване діалогове вікно;

– активізувати елемент керування, натиснувши ліву кнопку миші, у вікні Properties з’являться властивості елементу (рис. 16.5).

У вікні, що з’явилося, можна змінити напис на елементі керування і, що вкрай важливо, ідентифікатор ресурсу. Крім того, тут встановлюються властивості, що належать до зовнішнього вигляду і порядку використання елементу керування.

Оскільки кожне діалогове вікно в програмі є унікальним об’єктом, потрібно присвоювати вікнам і елементам керування, що входять до їх складу, ідентифікатори за власним вибором.

Ідентифікатори ресурсу – це певні константи препроцесора, створені при використанні директиви препроцесора C++ #define.

Рекомендується дотримувати угоду про префікси: ідентифікатори діалогових вікон мають префікс IDD_, а ідентифікатори елементів керування – IDC_.

Для того, щоб додати функцію-обробник до елементу керування, необхідно виконати такі дії:

– натиснути правою кнопкою миші на елемент та у випадному меню вибрати Add Event Handler (рис. 16.7);

- у списку “Message type” активізувати повідомлення (рис.16.7), для якого треба створити функцію-обробник;
- встановити у вікні Function handler name ім’я створюваного обробника дії від елементу керування;
- натиснути кнопку “Add and Edit”.

Спочатку функція-обробник дії від елементу керування порожня і містить тільки коментар-запрошення користувачу додати сюди свій код.

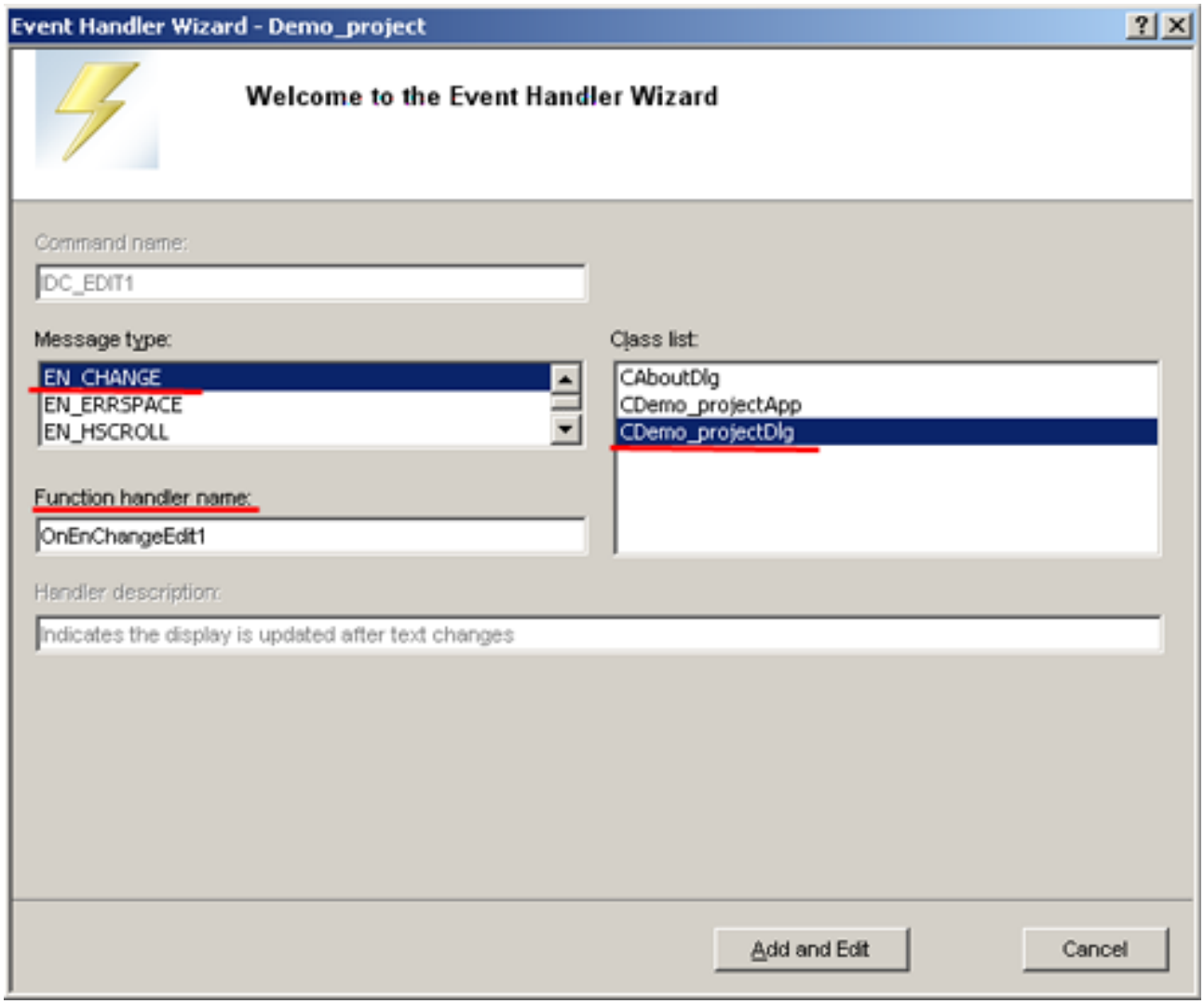


Рис. 16.7. Вікно Event Handler

16.4. Обмін діалоговими даними

Обмін діалоговими даними є ефективним засобом установлення взаємозв'язку між елементами керування та іншими типами даних, які не належать до елементів керування.

Функція CProjectNameApp :: DoDataExchange() підтримує обмін даними діалогу (DDX – dialog data exchange) і перевірку даних діалогу (DDV – dialog data verification). DDX дозволяє пов'язувати елементи даних з елементами керування так, щоб значення елементу даних завжди відображало вміст елементу керування. DDV дозволяє перевіряти дані, що вводяться користувачем, з метою відповідності набору правил.

Для взаємодії з елементом керування діалогового вікна необхідно встановити взаємозв'язок між елементом керування і компонентною змінною типу (Control), що керує, або інформаційною змінною-членом (Value) таким чином:

– активізувати елемент керування і, викликавши випадне меню та натиснувши по правій кнопці миші, вибрати пункт "Add Variable";

– у діалогове вікно, що з'явилося (рис.16.8), увести ім'я і тип змінної та її категорію (Control або Variable).

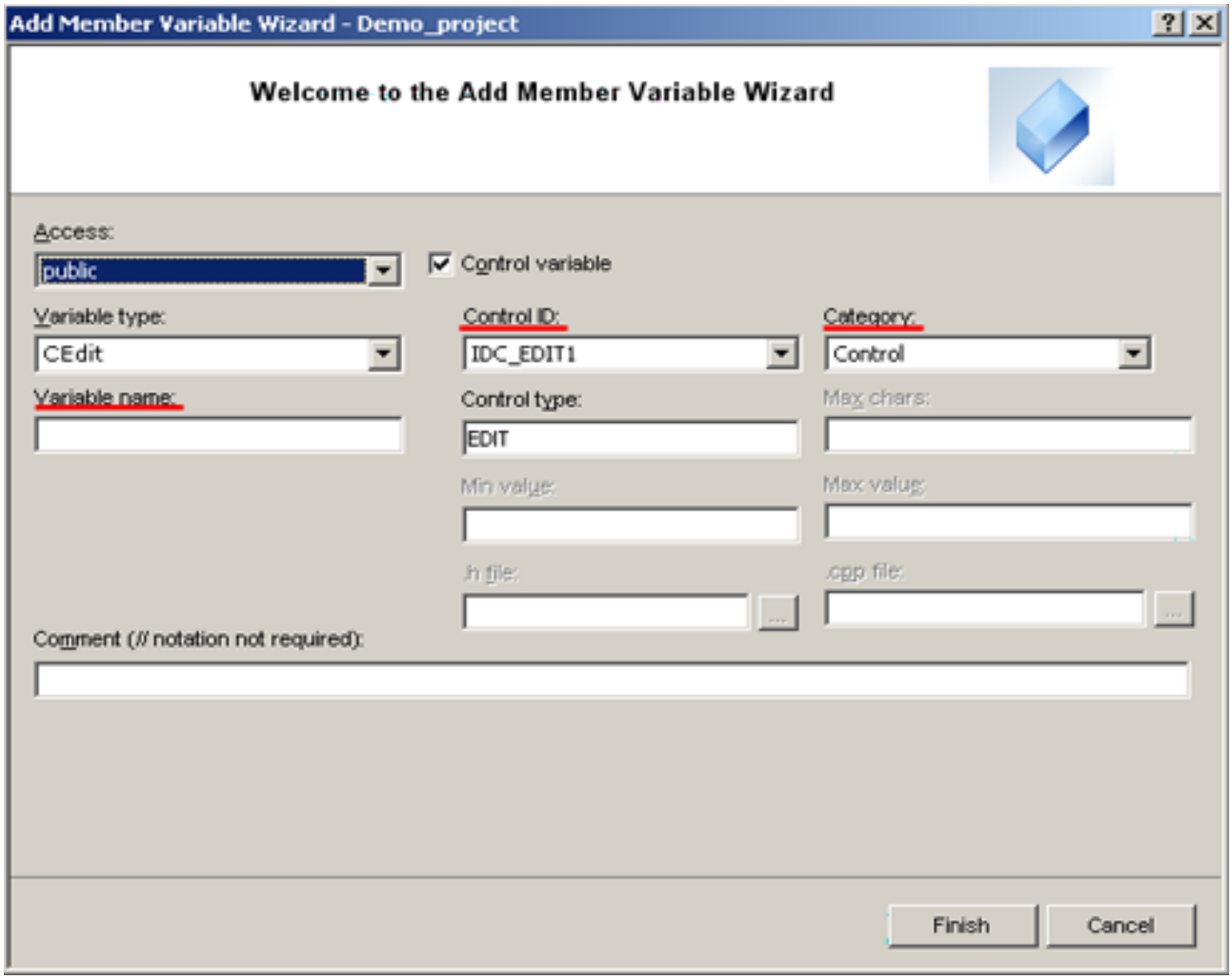


Рис. 16.8. Вікно Add Member Variable

Для того, щоб здійснити обмін даними між елементом керування та інформаційною змінною-членом, необхідно викликати функцію UpdateData() з потрібним параметром, який задає напрямок передачі інформації.

За допомогою компонентної змінної типу, що керує, можна маніпулювати елементом керування відповідно до конкретних потреб програми. Вживані при цьому методи обмежуються методами, доступними для того типу елементу управління, який був указаний при створенні змінної-члена.

Висновки

У даному розділі розглянуті наведені нижче основні питання:

- визначення MFC;
- структура прикладних програм, створених на базі MFC;
- створення проекту програми “діалогове вікно”;
- елементи керування та їх обробники;
- особливості обміну діалоговими даними.

Контрольні питання

1. У чому суть об’єктно-орієнтованого підходу при програмуванні для Windows?
2. З яких основних етапів складається робота програм для Windows і в чому їх суть?
3. Яка керованість подій Windows-програм і що таке поняття повідомлення?
4. Яка структура і призначення карти повідомлень?
5. Яке розташування і призначення циклу обробки повідомлень?
6. Який порядок створення проекту “діалогове вікно”?
7. Який порядок додавання в діалогове вікно кнопки та її обробника?
8. У якому місці проекту ініціалізується і запускається об’єкт “діалогове вікно”?

ЧАСТИНА 3. ЛАБОРАТОРНИЙ ПРАКТИКУМ

Лабораторний практикум містить у розділах 17 – 19 цикл взаємозв'язаних лабораторних робіт, у яких розглядаються і вивчаються питання подання інформації в комп'ютері, система числення, логічні операції, правила складання алгоритмів, ознайомлення із середовищем розробки програм Microsoft Visual Studio і структурою програм мовою С++, розробки програм для обчислювальних розгалужуваних і повторюваних процесів, використання одновимірних, багатовимірних і динамічних масивів і функцій.

Перед виконанням лабораторної роботи студенти повинні ознайомитися з розділами даного посібника, що відповідають темі лабораторної роботи, а потім скласти звіт, який включає теоретичні відомості, рисунки, таблиці, графіки тощо згідно з вимогами кожної лабораторної роботи.

Студентам також необхідно відповісти на контрольні питання, які наведені в методичних вказівках у кінці лабораторної роботи.

Під час проведення лабораторної роботи студенти повинні:

– обов'язково дотримуватися правил охорони праці;

– виконувати лабораторну роботу за відповідною методикою.

Після завершення лабораторної роботи вони показують виконане завдання викладачу, потім оформляють звіт і захищають його.

17. ОСНОВНІ ПОНЯТТЯ ПРО ПОДАННЯ ІНФОРМАЦІЇ ТА СХЕМИ ПРОГРАМ

17.1. Форма подання інформації у пам'яті комп'ютера і системи числення

Мета лабораторної роботи

Ознайомлення із системами числення, що найчастіше використовуються, та правилами виконання арифметичних операцій над двійковими числами.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і відповідні розділи цього посібника, такі питання:

– визначення системи числення;

– позиційні системи числення;

– форма подання інформації в комп'ютері;

– десяткова, двійкова, вісімкова і шістнадцяткова системи числення;

– правила переходу з однієї СЧс в іншу;

– правила виконання арифметичних операцій над двійковими числами.

Далі виконати такі дії:

– ознайомитися з табл. 17.1, у якій наведені числа від 0 до 15 в десятковій СЧс та їх еквіваленти у двійковій, вісімковій і шістнадцятковій СЧс;

– виконати переведення чисел *x, y, z* з однієї СЧс в інші відповідно до заданого викладачем варіанта (табл. 17.2);

– над цілими частинами двійкових зображень змінних *x* та *y* (табл. 17.2) виконати такі арифметичні дії:

а) додавання;

б) віднімання;

в) множення.

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

– тему і мету лабораторної роботи;

– опис завдання з початковими умовами і даними;

– виконане завдання відповідно до варіанта з детальним описом способу переведення чисел з однієї СЧс в іншу.

Таблиця 17.1

Подання чисел у різних СЧс

Системи числення			
Десяткова	Двійкова	Вісімкова	Шістнадцяткова
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Таблиця 17.2

Варіанти завдання

№ вар.	Завдання	№ вар.	Завдання
1.	$x=236_{10}$, $y=1101001110_2$, $z=216.38_{10}$. $x_{10}\rightarrow x_8\rightarrow x_2$ $y_2\rightarrow y_8\rightarrow y_{10}$ $z_{10}\rightarrow z_{16}$	10.	$z=1010111011_2$, $y=474_{10}$, $x=33.26_{10}$. $z_2\rightarrow z_{16}\rightarrow z_{10}$ $y_{10}\rightarrow y_8\rightarrow y_2$ $x_{10}\rightarrow x_2$
2.	$z=537_{10}$, $y=731_8$, $x=71.18_{10}$. $z_{10}\rightarrow z_{16}\rightarrow z_2$ $y_8\rightarrow y_2\rightarrow y_{16}$ $x_{10}\rightarrow x_2$	11.	$z=5F4_{16}$, $y=101100110_2$, $x=73.21_{10}$. $z_{16}\rightarrow z_{10}\rightarrow z_8$ $y_2\rightarrow y_8\rightarrow y_{10}$ $x_{10}\rightarrow x_2$
3.	$x=17A_{16}$, $y=11011011001_2$, $z=318.52_{10}$. $x_{16}\rightarrow x_{10}\rightarrow x_8$ $y_2\rightarrow y_8\rightarrow y_{10}$ $z_{10}\rightarrow z_{16}$	12.	$y=703_8$, $z=703_{10}$, $x=51.32_{10}$. $y_8\rightarrow y_2\rightarrow y_{16}$ $z_{10}\rightarrow z_{16}\rightarrow z_2$ $x_{10}\rightarrow x_2$
4.	$z=C3F_{16}$, $y=762_8$, $x=56.12_{10}$. $z_{16}\rightarrow z_2\rightarrow z_8$ $y_8\rightarrow y_{10}\rightarrow y_{16}$, $x_{10}\rightarrow x_2$	13.	$z=497_{10}$, $y=F4C_{16}$, $x=82.32_{10}$. $z_{10}\rightarrow z_{16}\rightarrow z_2$ $y_{16}\rightarrow y_{10}\rightarrow y_8$, $x_{10}\rightarrow x_8$
5.	$z=342_{10}$, $y=5FC_{16}$, $x=57.32_{10}$. $z_{10}\rightarrow z_{16}\rightarrow z_2$ $y_{16}\rightarrow y_{10}\rightarrow y_8$ $x_{10}\rightarrow x_2$	14.	$z=4F1_{16}$, $y=85_{10}$, $x=342.56_{10}$. $z_{16}\rightarrow z_{10}\rightarrow z_8$ $y_{10}\rightarrow y_2\rightarrow y_{16}$ $x_{10}\rightarrow x_8$
6.	$y=269_{10}$, $z=101101101101_2$, $x=194.47_{10}$. $y_{10}\rightarrow y_8\rightarrow y_2$ $z_2\rightarrow z_{16}\rightarrow z_{10}$ $x_{10}\rightarrow x_8$	15.	$x=1011011011_2$, $y=563_{10}$, $z=509.78_{10}$. $x_2\rightarrow x_8\rightarrow x_{10}$ $y_{10}\rightarrow y_8\rightarrow y_2$ $z_{10}\rightarrow z_{16}$
7.	$y=677_8$, $z=995_{10}$, $x=90.77_{10}$. $y_8\rightarrow y_2\rightarrow y_{16}$ $z_{10}\rightarrow z_{16}\rightarrow z_2$ $x_{10}\rightarrow x_2$	16.	$z=F4_{16}$, $y=1110110111_2$, $x=45.31_{10}$. $z_{16}\rightarrow z_2\rightarrow z_8$ $y_2\rightarrow y_8\rightarrow y_{10}$ $x10\rightarrow x2$
8.	$z=74_{10}$, $y=732_{10}$, $x=47.43_{10}$. $z_{10}\rightarrow z_2\rightarrow z_{16}$ $y_{10}\rightarrow y_8\rightarrow y_2$ $x_{10}\rightarrow x_2$	17.	$x=456_8$, $y=132_{10}$, $z=472.32_{10}$. $x_8\rightarrow x_2\rightarrow x_{10}$ $y_{10}\rightarrow y_8\rightarrow y_2$ $z_{10}\rightarrow z_{16}$
9.	$x=79_{10}$, $y=4AC_{16}$, $z=31.14_{10}$. $x_{10}\rightarrow x_2\rightarrow x_{16}$ $y_{16}\rightarrow y_8\rightarrow y_{10}$ $z_{10}\rightarrow z_8$	18.	$x=135_{10}$, $y=137.17_{10}$, $z=101101101111_2$. $x_{10}\rightarrow x_8\rightarrow x_2$ $z_2\rightarrow z_{16}\rightarrow z_{10}$ $y_{10}\rightarrow y_8$

Продовження табл. 17.2

№ вар.	Завдання	№ вар.	Завдання
19.	$x=347.78_{10}$, $y=143_8$ $z=767_{10}$. $z_{10}\rightarrow z_{16}\rightarrow z_2$ $y_8\rightarrow y_{10}\rightarrow y_2$ $x_{10}\rightarrow x_8$	25.	$x=110111100_2$, $y=6D_{16}$, $z=68.31_{10}$. $x_2\rightarrow x_{16}\rightarrow x_{10}$ $y_{16}\rightarrow y_2\rightarrow y_8$ $z_{10}\rightarrow z_8$
20.	$x=81.47_{10}$, $y=10111011100_2$ $z=74C_{16}$. $y_2\rightarrow y_8\rightarrow y_{10}$ $z_{16}\rightarrow z_2\rightarrow z_8$ $x_{10}\rightarrow x_2$	26.	$x=452_8$, $y=113_{10}$, $z=475.24_{10}$. $x_8\rightarrow x_2\rightarrow x_{10}$ $y_{10}\rightarrow y_8\rightarrow y_2$ $z_{10}\rightarrow z_{16}$
21.	$z=180_{10,}$ $y=5A_{16}$, $x=54.31_{10}$. $z_{10}\rightarrow z_{16}\rightarrow z_2$ $y_{16}\rightarrow y_{10}\rightarrow y_8$ $x_{10}\rightarrow x_2$	27.	$x=E3_{16}$, $y=132_{10}$, $z=472.32_{10}$. $x_{16}\rightarrow x_2\rightarrow x_8$ $y_{10}\rightarrow y_8\rightarrow y_2$ $z_{10}\rightarrow z_{16}$
22.	$z=537_{10}$, $y=731_8$, $x=71.18_{10}$. $z_{10}\rightarrow z_{16}\rightarrow z_2$ $y_8\rightarrow y_2\rightarrow y_{16}$ $x_{10}\rightarrow x_2$	28.	$y=703_8$, $z=703_{10}$, $x=51.32_{10}$. $y_8\rightarrow y_2\rightarrow y_{16}$ $z_{10}\rightarrow z_{16}\rightarrow z_2$ $x_{10}\rightarrow x_2$
23.	$x=17A_{16}$, $y=11011011001_2$, $z=318.52_{10}$. $x_{16}\rightarrow x_{10}\rightarrow x_8$ $y_2\rightarrow y_8\rightarrow y_{10}$ $z_{10}\rightarrow z_{16}$	29.	$z=497_{10}$, $y=F4C_{16}$, $x=82.32_{10}$. $z_{10}\rightarrow z_{16}\rightarrow z_2$ $y_{16}\rightarrow y_{10}\rightarrow y_8$ $x_{10}\rightarrow x_8$
24.	$z=C3F_{16}$, $y=762_8$, $x=56.12_{10}$. $z_{16}\rightarrow z_2\rightarrow z_8$ $y_8\rightarrow y_{10}\rightarrow y_{16}$ $x_{10}\rightarrow x_2$	30.	$z=4F1_{16}$, $y=85_{10}$, $x=342.56_{10}$. $z_{16}\rightarrow z_{10}\rightarrow z_8$ $y_{10}\rightarrow y_2\rightarrow y_{16}$ $x_{10}\rightarrow x_8$

Питання для підготовки до захисту лабораторної роботи

1. У якому вигляді подаються дані в комп'ютері?
2. Що таке система числення та які СЧс використовуються в ЕОМ?
3. Чим відрізняються позиційна та непозиційна системи числення?
4. Що таке основа СЧс?
5. За яким правилом виконується переведення цілої частини числа з десяткової СЧс у будь-яку іншу?
6. За яким правилом виконується переведення дробової частини числа з десяткової СЧс у будь-яку іншу?
7. Якими символами подаються числа в двійковій, вісімковій, десятковій і в шістнадцятковій системах числення?
8. Що таке тріади і тетради і яке їх призначення?

9. За якими правилами виконуються арифметичні операції над двійковими числами?

10. За яким правилом виконується переведення числа з будь-якої системи числення в десяткову?

17.2. Схеми програм

Мета лабораторної роботи

Вивчити правила складання схем програм.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і відповідні розділи цього посібника, такі питання:

– визначення алгоритму, схеми програми і їх властивості;

– базові конструкції керування.

Далі скласти схему програми для вирішення наведеного нижче завдання згідно з варіантом (табл. 17.3).

Таблиця 17.3

Варіанти завдань

№ вар.	Завдання
1.	Визначити в натуральному двозначному числі *n,* введеному з клавіатури, чи менше перша цифра за другу, і якщо так, то на скільки
2.	Дані дійсні числа x та y, що не дорівнюють один одному. Менше з цих двох чисел замінити половиною їх суми, а більше – їх подвоєним добутком
3.	Визначити, чи є сума цифр натурального чотиризначного числа однозначним числом
4.	Дане натуральне число *N*. Визначити, чи є воно кратним *K*, але не кратним *L*
5.	Визначити, чи є кожна наступна цифра даного натурального тризначного числа на одиницю більше за попередню
6.	Чи є сума цифр натурального чотиризначного числа, введеного з клавіатури, двозначним числом
7.	Визначити, чи є натуральне число *a* однозначним або тризначним
8.	Визначити, чи є всі цифри введеного з клавіатури натурального тризначного числа парними
9.	Визначити, чи є всі цифри введеного з клавіатури натурального тризначного числа однаковими
10.	Визначити, чи є в натуральному тризначному числі дві парні цифри
11.	Визначити, чи є в натуральному тризначному числі, введеному з клавіатури, тільки одна парна цифра
12.	Визначити, чи є сума першої і останньої цифр чотиризначного числа *n* непарним числом

Продовження табл. 17.3

№ вар.	Завдання
13.	Визначити, чи кратна 3 сума першої і останньої цифр тризначного числа *n*
14.	Визначити, чи є сума цифр тризначного числа *n* непарним числом
15.	Визначити, чи є добуток цифр тризначного числа *n* парним числом
16.	Визначити, чи є добуток цифр тризначного числа *n* числом кратним 5
17.	Визначити, чи є автобусний квиток щасливим, тобто чи дорівнює сума цифр трьох старших розрядів шестизначного числа сумі цифр трьох молодших розрядів
18.	Визначити, дільником яких чисел *A*, *B* або *C* є число *K*
19.	Визначити, чи є різниця другої і третьої цифр чотиризначного числа *n* парним числом
20.	Дане чотиризначне число *n*. Визначити, скільки парних цифр входить до цього числа
21.	Задані 3 числа. Вивести на екран більше число
22.	Дані дійсні числа *A*, *B*, *C*. Подвоїти ці числа, якщо $A > B > C$, і замінити їх абсолютними значеннями, якщо це не так
23.	Визначити, чи кратна 7 сума другої і третьої цифр чотиризначного числа *n*
24.	Задані 3 числа. Визначити, чи є сума яких-небудь двох з них позитивною

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- схему програми завдання згідно з варіантом.

Питання для підготовки до захисту лабораторної роботи

1. Які властивості повинен мати алгоритм і в чому його суть?
2. Дати визначення алгоритму.
3. Що використовується для зображення схем алгоритмів?
4. За допомогою якого символу позначають переходи керування за умовою?
5. Які дані записуються всередині символу переходів керування за умовою?
6. Який символ використовується при зверненнях до підпрограм?
7. Який символ застосовується для організації циклу?
8. Які існують базові конструкції алгоритмів?
9. Які варіанти реалізації конструкції «повторення» ви знаєте?

18. ОСНОВИ ПРОГРАМУВАННЯ МОВОЮ С++

18.1. Використання системи програмування для створення програм мовою С++

Мета лабораторної роботи

Ознайомитися із середовищем Microsoft Visual Studio, вивчити структуру програм мовою С++.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і відповідні розділи цього посібника, такі питання:

– основні компоненти Microsoft Visual Studio;
– етапи створення програми за допомогою системи програмування;
– структура програми мовою С++;
– написання програми мовою С++ та організація виводу даних у консоль.

Створити програму, що виводить на екран геометричну фігуру.

У табл. 18.1 для кожного варіанта завдання вказані вид геометричної фігури і символ її виводу.

Таблиця 18.1

Варіанти завдань

№ вар.	Геометрична фігура	Символ виводу	№ вар.	Геометрична фігура	Символ виводу
1.	Ромб	*	14.	Стрілка	z
2.	Квадрат	f	15.	Трапеція	+
3.	Трикутник	!	16.	Ромб	~
4.	Стрілка	a	17.	Квадрат	до
5.	Трапеція	s	18.	Трикутник	<
6.	Ромб	#	19.	Стрілка	h
7.	Квадрат	$	20.	Трапеція	d
8.	Трикутник	%	21.	Ромб	?
9.	Стрілка	^	22.	Квадрат	v
10.	Трапеція	0	23.	Трикутник	x
11.	Ромб	n	24.	Стрілка	@
12.	Квадрат	<	25.	Трапеція	u
13.	Трикутник	з	26.	Ромб	-

Підготувати звіт про виконання лабораторної роботи, який повинен включати:

– тему і мету лабораторної роботи;
– завдання з початковими умовами і даними;

– опис порядку створення та запуску консольної програми в середовищі Microsoft Visual Studio та основних його компонентів, що використовувалися при виконанні лабораторної роботи;

– текст програми мовою С++;

– детальний опис операторів та їх складових, що використовувались у програмі;

– результат запуску програми, тобто те, що програма виводить у консольне вікно.

Питання для підготовки до захисту лабораторної роботи

1. З яких компонентів складається середовище Visual мови С++?
2. Які зони входять до головного вікна Visual мови С++?
3. Що таке проект і з чого він складається?
4. З якої функції починається виконання будь-якої програми мовою С++?
5. Для чого використовується директива препроцесора *#include*?
6. Що входить до складу тіла функції main?
7. Для чого включається рядок *return 0* в кінці кожної функції main?
8. Яка операція здійснює вивід рядка на екран?
9. Як створити проект «консольна прикладна програма»?
10. Як запустити програму на виконання?

18.2. Змінні, базові типи і літерали. Арифметичні операції

Мета лабораторної роботи

Вивчити базові типи змінних, навчитися оголошувати та ініціалізувати змінні, вивчити арифметичні операції, використовувані в мові С++, їх особливості та пріоритети, а також правила обчислення арифметичних виразів.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і відповідні розділи цього посібника, такі питання:

– змінні і константи: визначення, види, використання;

– базові типи даних;

– вирази та оператори;

– арифметичні операції, які використовуються в мові С++.

Написати програму, що виконує дії, які описані нижче:

– вивести на екран у вигляді таблиці інформацію про базові типи даних – ім’я, розмір, діапазон значень. Для типу int визначити розмір даного типу (використовувати функцію sizeof()), обчислити кількість можливих значень і діапазон значень для даного типу

$$кількість\ значень = 2^{розмір},$$

де *розмір* – у бітах, визначається функцією sizeof() (у байтах);

– оголосити змінні типу unsigned short і short, присвоїти їм значення, що перевищують максимальне і мінімальне значення на 2, результат вивести на екран і пояснити;

– ввести з клавіатури значення змінних *a*, *b* (тип змінних див. табл. 18.2 згідно з варіантом) та *x*, обчислити і вивести на екран значення функції відповідно до варіанта (табл. 18.2). Приклади вбудованих математичних функцій, опис яких знаходиться в заголовному файлі <math.h>, подані в табл. 4.1;

– оголосити змінні *A*, *B*, *P*, *C*, присвоїти їм довільні значення і, використовуючи скорочену форму оператора присвоювання, обчислити:

$$A = \frac{A}{B + C * 5}, \text{ де } C=P \text{ і } B=B\%3.$$

Таблиця 18.2

Варіанти завдань

№ вар.	Функція	Тип даних введення	№ вар.	Функція	Тип даних введення
1.	$k = \frac{a \cos b^{\lvert a-b \rvert} + x \cos b}{e^{a-b}}$	Дійсний	13.	$y = \frac{x^a + b * \operatorname{arctg}(a)}{\ln \lvert b \rvert}$	Дійсний
2.	$t = \frac{(e^{b*x} + -a * x)}{2 * \sin(ax+b)}$	Цілий	14.	$z = \frac{\ln \lvert a^b \rvert + e^{a+b}}{\cos bx}$	Цілий
3.	$z = \frac{\lvert \ln a^b - \cos bx \rvert}{e^b}$	Дійсний	15.	$k = \frac{e^a}{\lvert \cos x + \sin x \rvert^{a-b}}$	Дійсний
4.	$r = \frac{x^{\lvert b-a \rvert} + b * \cos^3(a)}{\ln \lvert b \rvert}$	Цілий	16.	$d = \frac{e^{3*\lvert \sin bx \rvert} - a^x}{\cos x}$	Дійсний
5.	$k = \frac{\cos b^{\lvert a-b \rvert} + \cos b}{\lvert \ln x \rvert}$	Дійсний	17.	$t = \frac{e^{a*x} + e^{-a*x}}{2 * \sin(ax + b)}$	Цілий
6.	$k = \frac{e^x - \lvert \cos^2 x + \sin x \rvert}{e^{a+b}}$	Цілий	18.	$r = \frac{\lg \lvert ax - bx \rvert + \cos^2 x}{a+b}$	Дійсний
7.	$z = \frac{\operatorname{arctg}(e^{a*x} + \sqrt{b})}{x^a}$	Дійсний	19.	$\frac{j = e^a + \sin^2 a - b}{\sin(x - \sqrt{a})}$	Дійсний
8.	$y = \frac{(\sin(x-a) + b^x) * b}{\lvert \cos(x-b) \rvert}$	Дійсний	20.	$y = \frac{\sin(x-a) + b^x}{b \cos(x-b)} - a$	Цілий
9.	$f = \frac{\operatorname{arctg}(e^{a*x} + b / x)}{x^{\lvert b-a \rvert}}$	Цілий	21.	$z = \frac{e^{a-b} - e^{a-b}}{\lg \lvert x - a^b \rvert}$	Дійсний
10.	$n = \frac{\ln \lvert x^b \rvert + e^{a+b}}{\sin bx}$	Дійсний	22.	$k = \frac{\cos b^{\lvert a-b \rvert} + \sin a}{\sqrt{x} * \cos b}$	Цілий
11.	$q = \frac{\operatorname{arctg} \lvert a^b \rvert + e^a}{\cos(b * x^2)}$	Речовинний	23.	$m = \frac{\operatorname{arctg} x + e^a}{\sqrt{b}} + \lvert b \rvert$	Речовинний
12.	$r = \frac{\lg \lvert a^x \rvert - \sqrt{b}\, e^x}{\cos 2x}$	Дійсний	24.	$f = \frac{\ln \lvert x \rvert + \sin x^a}{\cos x + \sqrt{b}}$	Дійсний

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

– тему і мету лабораторної роботи;
– опис завдання з початковими умовами і даними;
– текст програми мовою С++;
– результат при запуску програми;
– пояснити отримані результати;
– помилки, що виникли при виконанні програми.

Питання для підготовки до захисту лабораторної роботи

1. Що таке змінна і які параметри вона має?
2. Що розуміють під типом даних?
3. Яке значення набуває неініціалізована змінна?
4. Яке значення набуває ціла знакова змінна при переповнюванні?
5. Що таке константа і які існують способи завдання констант?
6. Що таке вираз і чим визначається тип виразу?
7. Які арифметичні операції використовуються в мові С++?
8. Яка відмінність спостерігається між префіксними і постфіксними операціями?
9. Які відомі способи збільшення одиниці до цілої змінної x?
10. У чому різниця між операціями y++ і y+1?

18.3. Логічні вирази

Мета лабораторної роботи

Вивчити логічні та оператори відношення, використовувані в мові С++, а також одержати навики виконання побітових логічних операцій.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і відповідні розділи цього посібника, такі питання:

– логічні вирази;
– вирази порівняння;
– види і призначення бітових операцій;
– правила обчислення логічних виразів;
– скорочена схема обчислення;
– пріоритети операторів.

Послідовність виконання лабораторної роботи:

1. Над цілими частинами змінних *x, y, z,* узятих з лабораторної роботи підрозділу 17.1. відповідно до варіанта завдання, виконати такі порозрядні логічні операції:

1) обернення змінних *x, y, z*;

2) порозрядне логічне додавання за модулем 2 чисел *x* і *z*. Результат перевести у вісімкову СЧс;
3) порозрядне логічне додавання чисел *y* і *z*. Результат перевести в шістнадцяткову СЧс;
4) порозрядне логічне множення чисел *y* та *x*. Результат перевести в десяткову СЧс;
5) зсув управо змінної *y* на два розряди, результат перевести у вісімкову СЧс. Розмірність змінної *y* прийняти 2 байти;
6) зсув уліво змінної *x* на три розряди, результат перевести в шістнадцяткову СЧс. Розмірність змінної *x* прийняти 2 байти.

2. Написати програму, яка за допомогою форматованого виводу даних (див. підрозділ 5.5) виводить на екран у вигляді таблиці:

1) значення результатів, виконаних у попередньому пункті операцій, у заданому форматі;
2) приклади використання всіх операторів відношення;
3) приклади використання всіх логічних операторів для даних будь-якого типу;
4) знаходить значення виразу:

!(x < 5) && !(y >= 7) || (z!=y)

Пояснити принципи скороченої схеми обчислення.

5) обчислює результат виразів *x&y* і *x&& y* при $x=01_8$ і $y=02_8$.

Пояснити отримані результати.

Підготувати звіт про виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- виконане завдання 1;
- текст програми мовою С++ (завдання 2);
- пояснення до завдання 2 (пункти 4 та 5);
- результат при запуску програми;
- помилки, що виникли в ході виконання лабораторної роботи.

Питання для підготовки до захисту лабораторної роботи

1. У яких випадках використовуються логічні операції?
2. Які застосування побітових логічних операцій?
3. Якого типу формується результат логічної операції і чим він визначається?
4. Які логічні операції використовуються в мові С++?
5. Яким чином відбувається обчислення за скороченою схемою?
6. Справжнім або помилковим є значення -1?
7. Чи істинно таке твердження: операції відношення мають вищий пріоритет, ніж арифметичні?
8. Яка різниця між виразами x=3 і x = =3?

9. Чи рівнозначні в мові С++ вирази !!x і x?

10. Як створити логічні вирази, що відповідають таким умовам:

– *Weight* більше або дорівнює 115, але менше 125;

– *X* є парним, але не 26;

– *X* знаходиться в діапазоні 1000 – 2000?

18.4. Програмування обчислювальних процесів, що розгалужуються

Мета лабораторної роботи

Навчитися програмувати обчислювальні процеси мовою С++.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і відповідні розділи цього посібника, такі питання:

– конструкцію if;

– конструкцію switch;

– оператор break;

– оператор continue;

– використання налагоджувача.

Написати програми мовою С++:

– для вирішення завдання з лабораторної роботи підрозділа 17.2 відповідно до варіанта завдання за допомогою оператора *if*;

– для вирішення завдання з табл. 18.3 згідно з варіантом за допомогою оператора *switch*.

Таблиця 18.3

Варіанти завдань

№ вар.	Завдання
1.	За номером дня тижня вивести його назву
2.	За номером місяця визначити пору року
3.	За відомими періодами життя (дитсадок, школа, робота, пенсія) визначити орієнтовний вік людини
4.	При введенні одного із символів (*N* – називний, *R* – родовий, *D* – давальний, *Z* – знахідний, *O* – орудний, *P* – місцевий) вивести слово СТУДЕНТ у заданому відмінку однини
5.	За числом визначити декаду місяця
6.	Для будь-якого числа поточного місяця визначити – це вихідний або робочий день або видати повідомлення про те, що у цьому місяці такого числа немає
7.	Залежно від введених символів <, >, *, / виконати відповідну дію над двома введеними числами і вивести результат операції на екран
8.	Нехай змінна *N* набуває значення від 1 до 9. Надрукувати назву цифри, що зберігається в змінної (один, два, ...)

Продовження табл. 18.3

№ вар.	Завдання
9.	Для цілого числа *N* (від 20 до 30) надрукувати фразу "Мені *N* років", враховуючи при цьому, що при деяких значеннях *N* слово "років" треба замінити на слово "рік" або "роки"
10.	Для цілого числа *N* (від 1 до 8) надрукувати фразу "Ми знайшли *N* грибів", погодивши закінчення слова "гриб"
11.	За номером місяця визначити квартал
12.	За номером місяця вивести кількість днів цього місяця
13.	Залежно від введених символів +, - *, / виконати відповідну дію над двома введеними числами, вивести результат операції на екран. При виконанні операції ділення потрібно перевірити ділення на 0
14.	Вивести на екран круг, квадрат, ромб або трикутник залежно від вибору користувача
15.	Для цілого числа *N* (від 1 до 7) надрукувати фразу «*N*-поверхова будівля»
16.	Вивести число в заданому форматі. Формат задається введенням одного з таких символів: “*D*” – десятковий формат виводу, “*O*” – вісімковий формат, “*H*” – шістнадцятковий, а якщо інший символ – у всіх форматах
17.	У списку групи з перших десяти осіб залежно від введеного числа вивести прізвище
18.	Вивести на екран таблицю додавання, множення або піднесення до степеня числа 2 залежно від вибору користувача
19.	Вивести розклад на запитаний день тижня або повідомлення про те, що день вихідний
20.	Вивести розклад на понеділок і вівторок залежно від того чисельник або знаменник запитав користувач
21.	За введеним місяцем визначити період навчального процесу: осінній семестр, зимова сесія, весняний семестр, весняна сесія, літні канікули.
22.	Визначити, чи кратно введене число двом, трьом або п’яти. В іншому разі вивести негативну відповідь
23.	Залежно від запиту користувача порахувати площу круга $\left(S=\pi r^2\right)$, квадрата або трикутника $\left(S=\frac{abc}{4R}\right)$
24.	Залежно від запиту користувача порахувати площу трапеції $S=\frac{1}{2}\left(AD+BC\right)h$, де *AD* і *BC* підстави трапеції, *h* – висота), периметр прямокутника або довжину кола $\left(l=2\pi r\right)$

Підготувати звіт про виконання лабораторної роботи, який повинен включати:

– тему і мету лабораторної роботи;
– опис завдання з початковими умовами і даними;
– схему програми, виконану відповідно до стандартів;
– короткий опис програми;
– таблицю імен змінних, подану у вигляді табл. 18.4;
– текст програми мовою С++;
– результат при запуску програми;
– помилки, що виникли при виконанні програми.

Таблиця 18.4

Таблиця імен змінних

Ім’я змінної	Тип	Опис
…	…	…

Питання для підготовки до захисту лабораторної роботи

1. Який формат запису має конструкція *switch*?
2. Яке призначення конструкції *switch*?
3. Як відбувається виконання конструкції *switch*?
4. Який формат запису має конструкція *if*?
5. Яке призначення конструкції *if*?
6. Як відбувається виконання конструкції *if*?
7. Яке призначення має оператор *break*?
8. Як сформувати розгалуження конструкції *if*, що друкує слово „*Yes”* у випадку, якщо значення змінної *age* більше ніж 21?
9. Як сформувати розгалуження конструкції *if*, що друкує слово „*Yes”* у випадку, якщо значення змінної age більше ніж 21, і слово „*No”* у решті випадків?
10. Як сформувати розгалуження конструкції switch, що друкує слово „Yes” у випадку, якщо значення змінної *ch* дорівнює ‘*y*’, слово „No” у випадку, якщо „ch” дорівнює ‘*n*’, а також слово „Unknown” у решті випадків?

18.5. Циклічні обчислювальні процеси

Мета лабораторної роботи

Вивчити інструкції, використовувані в мові С++, для організації циклічних обчислювальних процесів: for, while, do-while.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і відповідні розділи цього посібника, такі питання:

– види конструкцій для організації циклів;
– конструкція циклу for;

– конструкція циклу while;
– конструкція циклу do-while;
– оператори break і continue.

Написати програму, що виконує завдання, подане в табл. 18.5, згідно з варіантом, враховуючи такі вимоги:

– програма повинна виконуватися доти, поки користувач не захоче припинити роботу з програмою;

– програма повинна мати зрозумілий і доступний будь-якому користувачу інтерфейс.

Таблиця 18.5

Варіанти завдання

№ вар.	Завдання
1.	Розрахувати факторіал додатного цілого числа, введеного з клавіатури $n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 1$; 0!=1; 1!=1.
2.	Визначити числа, кратні заданому користувачем числу. Кількість кратних чисел не повинна перевищувати 50. Знайти суму цих чисел
3.	Знайти найбільше з десяти послідовно введених цілих чисел
4.	Вивести на екран таблицю множення, введеного з клавіатури числа, на число в діапазоні від 1 до 20
5.	Розрахувати значення x в степені y. Для розрахунку використовувати тільки арифметичні операції
6.	Визначити найбільший загальний дільник двох чисел, введених з клавіатури. (Найбільший загальний дільник двох чисел – найбільше число, на яке діляться обидва числа.)
7.	Знайти добуток всіх двозначних чисел, кратних 5 і 2
8.	Вивести на екран порожній квадрат заданого розміру за допомогою визначеного символу. Наприклад: * * * * * * * * * * * * У програмі для виводу на екран квадрата можуть бути використані тільки такі оператори: cout<<'*'; cout<<' '; cout<<'\n';
9.	Перевести число, введене з клавіатури в двійковій системі, в десяткову систему числення і вивести результат на екран. Для виділення з числа розрядів справа наліво використовуйте таку формулу: $x_i=(x/10^i)\%10$, де x_i – і-й розряд числа. Нумерація розрядів починається з нуля

Продовження табл. 18.5

№ вар.	Завдання
10.	Перевести число, введене з клавіатури в десятковій системі, в двійкову систему числення і вивести результат на екран. Для переведення числа з десяткової системи числення в двійкову використовуйте операції знаходження остачі від ділення і цілочисельного ділення
11.	Визначити найменше загальне кратне двох чисел, введених з клавіатури. (Найменше загальне кратне двох чисел – найменше число, яке ділиться на обидва числа.)
12.	Підрахувати і вивести на екран середнє арифметичне послідовності чисел, введених користувачем. Вважайте, що остання читана величина є міткою 9999. Тобто потрібно знайти середнє значення всіх чисел, що передують мітці 9999
13.	Знайти найменше із введених з клавіатури цілих чисел. Кількість чисел, що вводяться, задає перше число
14.	Вивести на екран усі прості числа, що знаходяться в діапазоні між двома введеними
15.	Визначити *n*-й член ряду Фібоначчі. Кожне число ряду (після другого) є сумою тих двох, що стоять попереду чисел: 1, 1, 2, 3, 5, 8, 13, 21, 34 …
16.	Знайти суму всіх тризначних чисел, що діляться на 7
17.	Написати програму, яка пропонує введення пари чисел. Після вводу кожної пари чисел повідомляється середнє арифметичне введеної пари чисел. Програма завершується після вводу одного з чисел, що дорівнює нулю
18.	За допомогою зірочки (*) вивести на екран прямокутний трикутник, наприклад: * ** *** **** Усі зірочки повинні друкуватися одним оператором вигляду cout<<'*'; Розмір катетів (кількість зірочок) задає користувач
19.	Розрахувати значення π на основі нескінченного ряду $\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \ldots$ Визначити величину із заданою користувачем точністю. Вивести на екран таблицю, що відображає, як значення міняється залежно від кількості членів ряду
20.	Написати програму, яка пропонує введення чисел. Після вводу кожного числа повідомляється накопичувальна сума введених дотепер чисел. Програма завершується після вводу нуля

Закінчення табл. 18.5

№ вар.	Завдання
21.	Знайти суму ряду $\left(2+\frac{1}{2}\right)^2+\left(4+\frac{1}{4}\right)^2+\ldots+\left(2^n+\frac{1}{2^n}\right)^2$
22.	В арифметичній прогресії a3 і d задається користувачем. Визначити члени арифметичної прогресії від a1 до a30 і вивести отриманий результат на екран $a_n = a_1 + (n-1)d$
23.	У геометричній прогресії $b5$ і q задається користувачем. Визначити члени геометричної прогресії від $b20$ до $b1$ і вивести отриманий результат на екран $b_n = b_1 q^{n-1}$
24.	Розрахувати суму грошей, які ви одержите при вкладенні початкової суми з фіксованою процентною ставкою прибутку через визначену кількість років. Користувач повинен вводити з клавіатури початковий внесок, число років і процентну ставку

Підготувати звіт про виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- схему програми, виконану відповідно до стандартів;
- короткий опис програми;
- таблицю імен змінних, подану у вигляді табл. 18.4;
- текст програми мовою С++;
- результат при запуску програми;
- помилки, що виникли при виконанні програми.

Питання для підготовки до захисту лабораторної роботи

1. Назвіть і опишіть основне призначення кожного з трьох виразів, що входять до складу оператора циклу for.
2. Чи істинно таке твердження: інкрементальний вираз циклу може декрементувати лічильник циклу?
3. Який цикл буде виводити на екран числа від 100 до 110?
4. Скільки разів виконується тіло циклу do-while?
5. У чому відмінність циклу з передумовою від циклу з постумовою? До якого із вказаних видів можна віднести кожен цикл мовою С++?
6. Що виводить на друк такий фрагмент коду:

```
for( int j=0; j<11; j+=3)
cout<<j;
cout<<"\n"<<j<<"\n";
```

7. Що виводить на друк такий фрагмент коду:

```
int j=5;
while( ++j < 9 )
   cout<<j++<<”\n”;
```

8. Що виводить на друк такий фрагмент коду:

```
int j=8;
do
   cout<<” j = ”<<j<<”\n”;
while(j ++ < 5 )
```

9. Який цикл for буде виводити на друк значення 1, 2, 4, 8, 16, 32, 64 при збільшенні на 2 значення лічильника на кожному кроці циклу?

10. Яким чином можна включити в тіло циклу декілька операторів?

19. РОБОТА З МАСИВАМИ ТА ФУНКЦІЯМИ

19.1. Одновимірні масиви

Мета лабораторної роботи

Одержати основні навички роботи з одновимірними масивами.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і відповідні розділи цього посібника, такі питання:

– визначення масиву;
– використання масивів;
– проходження масиву.

Написати програму, яка виконувала б завдання, наведене в табл. 19.1, згідно з варіантом, враховуючи такі вимоги:

– при запуску програма повинна вимагати у користувача введення пароля (номер варіанта) доти, поки він не введе пароль правильно;

– програма повинна передбачати ініціалізацію масиву з клавіатури та явно (за вибором користувача);

– на екран монітора повинні виводитися початкові дані, а також результати всіх операцій;

– при знаходженні максимальних і (або) мінімальних елементів необхідно виводити на екран їх позиції і значення;

– програма повинна мати зрозумілий і доступний будь-якому користувачу інтерфейс;

– програма повинна виконуватися доти, поки користувач не захоче припинити роботу з програмою.

Таблиця 19.1

Варіанти завдань

№ вар.	Завдання
1.	У дійсному масиві *x*(20) підрахувати кількість додатних елементів і знайти мінімальний серед від'ємних
2.	Обчислити елементи вектора *z*(10) за формулою *zi*=*sqrt*((*xi*+*ai*)/2), якщо *xi* та *ai* – елементи цілочисельних масивів, що складаються з десяти елементів кожен. Поміняти місцями мінімальні елементи в масивах *x*(10) і *a*(10)
3.	Записати в масив *y*(20) куби парних чисел і квадрати непарних чисел, вказаних у цілочисельному масиві *n*(20). В одержаному масиві поміняти місцями максимальний і мінімальний елементи
4.	Вивести на екран окремо додатні елементи дійсного масиву *x*(16) і окремо від'ємні. Знайти максимальний елемент серед від'ємних і мінімальний серед додатних елементів

Продовження табл. 19.1

№ вар.	Завдання
5.	У дійсному масиві $x(21)$ знайти суму додатних елементів на парних позиціях і максимальний елемент серед від'ємних на непарних позиціях
6.	Обчислити елементи вектора $x(20)$ за формулою $x_i=4tg(1+0.1i)+e^{-0.5i}+2\pi$, $i=1,...,20$. Далі одержаний вектор перетворити за правилом: усі від'ємні елементи збільшити на 0,5, а всі додатні – замінити на 0,1. У перетвореному векторі знайти мінімальний елемент серед від'ємних
7.	Обчислити суму і різницю цілочисельних масивів $a(20)$ і $b(20)$. Результат вивести на екран у вигляді двох паралельних стовпців. Поміняти місцями мінімальні елементи в одержаних стовпцях
8.	Знайти середнє значення елементів речовинного масиву $x(12)$, що стоять на непарних позиціях і присвоїти набутого значення мінімальному елементу
9.	У дійсному масиві $y(16)$ знайти максимальний елемент на парних позиціях і вивести на екран номери елементів, що задовольняють умову $0<y_i<1$
10.	У дійсному масиві $x(10)$ знайти максимальний елемент. Змінній w присвоїти значення, що дорівнює сумі всіх елементів, які більше ніж ціле число b (b задано), передуючих максимальному елементу. Інакше $w=0$
11.	Дані два дійсних масиви $x(10)$ і $y(8)$. Сформувати масив $z(18)$ з додатних елементів масивів x та y. Якщо додатних елементів менше 18, то елементам масиву z, що залишилися, присвоїти значення +1, а останньому – значення максимального елементу в масиві x
12.	Обчислити елементи вектора $B(14)$ за формулою $B_i=i^2+\sin(0,1*i)$, $i=1,...,14$. Далі одержаний вектор перетворити за правилом: усі елементи на парних позиціях піднести до квадрата, а на непарних – замінити мінімальним елементом
13.	Записати -1 замість мінімального елементу в цілочисельному масиві $x(11)$ та $i+1$ – замість максимального елементу. Якщо між ними знаходиться хоч один нульовий елемент, то змінній p присвоїти *true*, інакше – *false*
14.	Знайти найбільший серед від'ємних елементів дійсного масиву $C(15)$. Визначити кількість чисел, що більше ніж їх середнє значення
15.	Даний дійсний масив $x(14)$. Знайти максимальний елемент, а всі елементи, що йдуть за ним і мають значення менше 2, замінити нулями
16.	Заданий масив $y(24)$. Визначити кількість елементів, які при діленні на 7 дають остачу 1, 2 або 5. Знайти серед них максимальний елемент

Закінчення табл. 19.1

№ вар.	Завдання
17.	Даний цілочисельний масив *q*(23). Одержати суму тих елементів масиву, які непарні і від'ємні. Присвоїти набутого значення максимальному елементу
18.	Даний цілочисельний масив *f*(13). Знайти в ньому максимальний елемент серед додатних, кратних 5. Поміняти місцями елементи, що межують з ним
19.	Даний дійсний масив *K*(15). Порахувати кількість додатних елементів на парних позиціях. Присвоїти набутого значення мінімальному елементу
20.	Даний цілочисельний масив *A*(20). Знайти мінімальний елемент на непарних позиціях. Усім додатним парним елементам, передуючим йому, присвоїти 0.
21.	Даний дійсний масив *B*(24). Від'ємні елементи на парних позиціях замінити на 0, а на непарних – на 1. Серед незмінених елементів знайти максимальний
22.	Даний цілочисельний масив *C*(19). Знайти добуток тих елементів масиву, які парні і більші за 0, виключаючи значення максимального елементу.
23.	Даний дійсний масив *A*(13). Знайти найменший серед додатних елементів масиву. Якщо сума елементів, передуючих даному елементу, більше половини суми всіх елементів масиву, то змінній *f* присвоїти *true*, інакше – *false*.
24.	Даний речовинний масив *A*(32). Одержати число від'ємних членів серед перших 20-ти елементів масиву і знайти мінімальний серед решти елементів

Підготувати звіт про виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- схему програми, виконану відповідно до стандартів;
- короткий опис програми;
- таблицю імен змінних, подану у вигляді табл. 19.2;
- текст програми мовою С++;
- результат при запуску програми;
- помилки, що виникли при виконанні програми.

Таблиця 19.2

Таблиця імен змінних

Ім'я змінної	Тип	Опис
. . .	. . .	. . .

Питання для підготовки до захисту лабораторної роботи

1. Що таке масив? Як визначити масив?
2. Які існують способи ініціалізації масиву?
3. Дайте визначення розмірності масиву.
4. Як отримати доступ до елементів масиву?
5. Як можна скопіювати один масив в інший? Наведіть приклад.
6. Який оператор визначає одновимірний дійсний масив, що містить 100 елементів?
7. Яким чином можна ініціалізувати масив нульовими значеннями?
8. Скільки елементів у даному масиві
   int Mas[]={2,3,6,4}; ?
9. Оголосіть масив цілих чисел розміром 10. Задайте 5-му елементу значення 3, інші обнуліть.
10. Скільки елементів у кожному масиві? Назвіть помилки в наведених оголошеннях, якщо вони є:

– int Mas[3]={2,3,6,4};
– int Mas[5]={2,3,6,4};

19.2. Двовимірні масиви

Мета лабораторної роботи

Одержати основні навички роботи з двовимірними масивами.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і відповідні розділи цього посібника, такі питання:

– визначення і використання двовимірного масиву;
– звернення до елементів двовимірного масиву;
– доступ до двовимірного масиву.

Написати програму, яка виконувала б завдання, наведене в табл. 19.3, згідно з варіантом, враховуючи такі вимоги:

– програма повинна передбачати ініціалізацію масиву з клавіатури та явно (за вибором користувача);
– на екран монітора повинні виводитися початкові дані, а також результати всіх операцій;
– при знаходженні максимальних і (або) мінімальних елементів необхідно виводити на екран їх позиції і значення;
– програма повинна мати зрозумілий і доступний будь-якому користувачу інтерфейс;
– програма повинна виконуватися доти, поки користувач не захоче припинити роботу з програмою.

Таблиця 19.3

Варіанти завдань

№ вар.	Завдання
1.	Дана матриця *M*(5,5). Серед елементів головної діагоналі знайти найбільший, а серед елементів побічної діагоналі – найменший, і поміняти їх місцями
2.	У матриці *A*(6,6) знайти добуток додатних елементів головної діагоналі і суму всіх елементів, що нижче за побічну діагональ
3.	У матриці *K*(6,6) знайти добуток від'ємних елементів, що нижчі за головну діагональ, і суму всіх ненульових додатних елементів матриці
4.	Дана матриця *M*(5,5). Серед елементів побічної діагоналі знайти найбільший елемент. Вивести його на друк та обнулити рядок і стовпець, в якому він розташований
5.	Дана матриця *B*(4,4). Серед елементів головної діагоналі знайти найменший елемент, вивести його, і якщо він менше за деяке число *K*, то до всіх елементів рядка, в якому він розташований, додати 1
6.	У матриці *Z*(6,6) знайти максимальний елемент, розташований на головній діагоналі, і добуток всіх від'ємних елементів матриці, що вищі за побічну діагональ
7.	Дана матриця *X*(6,6). Серед елементів, вищих за головну діагональ, знайти найбільший елемент, а серед елементів, нижчих за головну діагональ, – найменший, і поміняти їх місцями
8.	Серед елементів, розташованих нижче за головну діагональ матриці *A*(8,8), знайти ті елементи, які задовольняють умову *K2* <=*A*(*i,j*)<=*K1* (*K1, K2* – довільні числа), і сформувати з них одновимірний масив
9.	У матриці *T*(7,7) знайти максимальний елемент, розташований вище за побічну діагональ, вивести на екран його значення і координати, а стовпець, в якому він розташований, обнулити
10.	У матриці *B*(6,6) знайти максимальний елемент і серед елементів, розташованих вище за головну діагональ, знайти елементи, що задовольняють умову *B*(*i,j*)<*max* і *B*(*i,j*)>*i*+*j*, і сформувати з них одновимірний масив *X*
11.	У матриці *C*(6,6) знайти суму і добуток елементів, розташованих вище і нижче за головну діагональ відповідно
12.	У матриці *H*(5,5) знайти мінімальний елемент серед елементів, розташованих нижче за побічну діагональ, і кількість елементів матриці, значення яких не більше деякого числа *K*
13.	З масивів *X*(15) і *Y*(10) побудувати матрицю *A*(5,5) так, щоб елементи масиву *X* були розташовані на головній діагоналі і вище за неї
14.	У матриці *D*(6,6) знайти добуток ненульових елементів, розташованих на головній діагоналі і вище за неї. Підрахувати кількість нульових елементів у всій матриці

Продовження табл. 19.3

№ вар.	Завдання
15.	У матриці *C*(5,5) знайти суму елементів, розташованих на головній діагоналі і нижче за неї. Вивести набуте значення. Всі парні рядки матриці обнулити
16.	Для матриці *E*(6,6) обчислити суми елементів головної і побічної діагоналей, розташованих в одному стовпці, і сформувати одновимірний масив
17.	У матриці *Y*(5,5) знайти мінімальний елемент серед від'ємних елементів, розташованих нижче за побічну діагональ. Рядок, в якому знаходиться цей елемент, обнулити
18.	Дана матриця *H*(6,6). Знайти в кожному стовпці найбільший елемент і поміняти його місцями з елементом побічної діагоналі цього стовпця
19.	З масивів *X*(10) і *Y*(15) побудувати матрицю *A*(5,5) так, щоб елементи масиву X були розташовані вище за побічну діагональ
20.	Дана матриця *H*(6,6). Знайти в кожному рядку найменший елемент і поміняти його місцями з елементом головної діагоналі цього рядка
21.	Для матриці *C*(7,7) обчислити різницю елементів головної і побічної діагоналей, розташованих на одному рядку, і сформувати одновимірний масив
22.	Серед елементів, розташованих вище за побічну діагональ матриці *R*(6,6), знайти ті елементи, які задовольняють умову min<R(i,j)<max-min (min, max – мінімальні і максимальні елементи в матриці), і сформувати з них одновимірний масив
23.	Для матриці *P*(7,7) знайти мінімальні елементи серед елементів головної і побічної діагоналей, розташованих в одному рядку, і сформувати з них одновимірний масив
24.	Дана матриця *B*(5,5). Знайти найбільший елемент серед тих, що стоять на головній і побічній діагоналях, і поміняти його місцями з елементом, що стоїть на перетині цих діагоналей

Підготувати звіт про виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- схему програми, виконану відповідно до стандартів;
- короткий опис програми;
- таблицю імен змінних, подану у вигляді табл. 19.2;
- текст програми мовою C++;
- результат при запуску програми;
- помилки, що виникли при виконанні програми.

Питання для підготовки до захисту лабораторної роботи

1. Які існують способи ініціалізації багатовимірного масиву?
2. Як отримати доступ до елементів двовимірного масиву?
3. Як визначити двовимірний масив?
4. Який оператор визначає двовимірний дійсний масив, що містить 100 елементів?
5. Яким чином можна ініціалізувати двовимірний масив нульовими значеннями?
6. Скільки елементів у даному масиві
   int Mas[3][4]={2,3,6,4}; ?
7. Як оголосити двовимірний масив цілих чисел розміром 10, задати елементу нульового рядка і другого стовпця значення 3, а інші обнуліть?
8. Як сформулювати вираз для доступу до 4-ого елементу двовимірного масиву?
9. Як сформулювати послідовність операторів, яка друкує двовимірний масив у табульованому форматі?
10. Як присвоїти нульові значення першим 10-ти елементам двовимірного масиву?

19.3. Функції

Мета лабораторної роботи

Одержати навики створення функцій і вивчити деякі способи передачі параметрів у функцію.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і відповідні розділи цього посібника, такі питання:

- визначення функції;
- параметри функції;
- виклик функції;
- прототипи функції;
- способи передачі параметрів у функцію.

Потім виконати такі дії:

– написати функцію або декілька функцій, які призначені для вирішення завдання, наведеного в табл. 19.4, згідно з варіантом, враховуючи такі вимоги:

1) у функцію повинні передаватися параметри відповідно до значення;
2) один з параметрів повинен бути заданий за умовчанням;

– написати програму, яка ілюструє роботу з одержаними функціями.

Таблиця 19.4

Варіанти завдань

№ вар.	Завдання
1.	Знайти всі числа з інтервалу [*M*, *N*], що мають найбільшу кількість дільників
2.	Знайти цілі числа-паліндроми з інтервалу [*M*, *N*], які при піднесенні до квадрата також дають паліндроми ($22^2 = 484$). Паліндром – це поєднання символів, які читаються однаково в прямому і зворотному напрямах
3.	Два натуральні числа називаються дружніми, якщо кожне з них дорівнює сумі всіх дільників (окрім його самого) іншого числа. Знайти всі пари дружніх чисел, які не більше за дане число *N*. Наприклад числа 220 і 284 є дружніми, оскільки 220 → 1+2+5+4+10+11+20+22+ 44+55+110=284 284 → 1+2+4+71+142=220
4.	Два прості числа називаються “близнюками”, якщо вони відрізняються один від одного на 2 (наприклад, 41 і 43). Надрукувати всі пари “близнюків” з відрізка [n, 2n], де n – задане натуральне число, більше ніж 2
5.	Натуральне число, в записі якого *n* цифр, називається числом Армстронга, якщо сума його цифр, піднесена до ступеня *n*, дорівнює самому числу. Знайти всі числа Армстронга від 1 до *k*
6.	Знайти всі натуральні n-значні числа, цифри в яких утворюють строго зростаючу послідовність (наприклад, 1234, 5789)
7.	Знайти всі натуральні числа, які не перевищують заданого *n* і які діляться на кожну із своїх цифр
8.	Є частина котушки з автобусними квитками. Номер квитка шестизначний. Скласти програму, що визначає кількість щасливих квитків на котушці, якщо менший номер квитка – *N*, більший – *M* (квиток є щасливим, якщо сума перших трьох його цифр дорівнює сумі останніх трьох)
9.	Написати програму, що визначає суму *n*-значних чисел, що містять тільки непарні цифри. Визначити також, скільки парних цифр у знайденій сумі
10.	Із заданого числа відняти суму його цифр. З результату знов відняти суму його цифр і т.д. Скільки таких дій треба виконати, щоб вийшов нуль?
11.	Дане парне число $n > 2$. Перевірити для нього гіпотезу Гольдбаха: кожне парне n подається у вигляді суми двох простих чисел
12.	Дані натуральні числа a і b, що позначають відповідно чисельник і знаменник дробу. Скоротити дріб, тобто знайти такі натуральні числа *p* і *q,* що не мають загальних дільників, тобто $p/q = a/b$

Продовження табл. 19.4

№ вар.	Завдання
13.	Дане натуральне число *n*. Якщо це не паліндром, реверсуйте його цифри і складіть початкове число з числом, одержаним у результаті реверсування. Якщо сума не паліндром, то повторіть ті самі дії і виконуйте їх доти, поки не одержите паліндром. Нижче наведений приклад для початкового числа 78: 78 + 87=165; 165 + 561 = 726; 726 + 627=1353; 1353 + 3531 = 4884. Паліндром – це поєднання символів, які читаються однаково в прямому і зворотному напрямках
14.	Дані натуральне число *n* і цілі числа *a*1, *a*2, ..., *a*n. Знайти кількість чисел a_i (i = 1, 2, ..., n), що є степенями п’ятірки. (Визначіть функцію, що дозволяє розпізнавати степені п’ятірки.)
15.	Знайти цілі числа, які при піднесені до 3 або 4, або 5 степеня дають паліндроми, наприклад 11^3 = 1331. Паліндром – це поєднання символів, які читаються однаково в прямому і зворотному напрямках
16.	Для запису римськими цифрами використовуються символи *I*, *V*, *X*, *L*, *C*, *D*, *M*, що позначають відповідно числа 1, 5, 10, 50, 100, 500, 1000. Скласти програму, яка б запис будь-якого даного числа *n* (n < 3999) арабськими цифрами переводила в запис римськими
17.	Обчислити суму тих чисел із заданого відрізка [*a, b*] (*a*, *b* – натуральні числа), до запису яких входить цифра *k*
18.	Визначити кількість *M*-значних натуральних чисел, у яких сума цифр, що стоять у непарних розрядах, дорівнює *N* ($1 < N <= 30$, $0<M<5$)
19.	Скласти програму вилучення із запису десяткового числа *N* одиниць із збереженням порядку проходження цифр, що залишилися. Сформувати і надрукувати одержане число
20.	На відрізку [2, і] знайти всі натуральні числа, сума цифр яких при множенні числа на *A* не зміниться
21.	Дане натуральне число *n*. Серед чисел 1..., *n* знайти такі, запис яких збігається з останніми цифрами запису їх квадратів (наприклад, 6^2 =36, 25^2 = 625)
22.	Натуральне число *M* називається досконале, якщо воно дорівнює сумі всіх своїх дільників, включаючи 1, але виключаючи себе. Надрукувати всі досконалі числа, менші ніж задане число *N*
23.	Дане натуральне число *k*. Надрукувати *k*-ю цифру послідовності 2468101214161820222426283O..., в якій виписані підряд усі натуральні парні числа
24.	Дане натуральне число *N*. Одержати нове число *M*, яке утворюється з числа *N* шляхом заміни останньої цифри на значення найменшої цифри в записі числа *N*. Приклад: *N*=128452, *M*=129451

Підготувати звіт про виконання лабораторної роботи, який повинен включати:

– тему і мету лабораторної роботи;
– опис завдання з початковими умовами і даними;
– схему програми, виконану відповідно до стандартів;
– короткий опис програми;
– таблицю імен змінних, поданих у вигляді табл. 19.2;
– текст програми мовою С++;
– результат при запуску програми;
– помилки, що виникли при виконанні програми.

Питання для підготовки до захисту лабораторної роботи

1. Для чого використовуються функції?
2. Як викликати функцію?
3. Що таке функція?
4. Як визначити та оголосити функцію?
5. Що таке прототип функції?
6. Чим відрізняються формальні параметри функції від фактичних?
7. Які параметри називаються параметрами за умовчанням?
8. Як здійснюється передача параметрів у функцію відповідно до значення?
9. Що є процесом передачі параметрів у функцію?
10. Скільки можливих способів виклику функції, у якої два з трьох параметрів задані за умовчанням?

19.4. Покажчики. Робота з рядками

Мета лабораторної роботи

Одержати основні навики роботи з рядками. Використання покажчиків і передача масивів у функцію.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і відповідні розділи цього посібника, такі питання:

– визначення рядків;
– використання рядків;
– покажчики: визначення і використання;
– зв’язок масивів і покажчиків;
– передача параметрів за посиланням;
– передача параметрів за покажчиком.

Написати програму, яка виконувала б завдання, наведене в табл. 19.5, згідно з варіантом і відповідала таким вимогам:

– програма повинна містити функції для обробки масиву символів (рядки), необхідні для вирішення поставленого завдання;

– при виконанні завдання рядок символів зберігати в змінній типу char;

– для роботи з рядками не можна використовувати функції з бібліотеки string та змінні типу string;

– програма повинна передбачати ввід рядка за вибором користувача: з клавіатури або явно (за умовчанням);

– на екран повинні виводитися початкові дані, а також результати всіх операцій;

– програма повинна мати зрозумілий і доступний будь-якому користувачу інтерфейс;

– програма повинна виконуватися доти, поки користувач не захоче припинити роботу з програмою.

Таблиця 19.5

Варіанти завдань

№ вар.	Завдання
1.	Розбити рядок на слова. Функція одержує як параметри початковий рядок і символ, що розділяє слова в рядку
2.	Визначити кількість однакових букв у рядку і довжину рядка
3.	Визначити кількість слів у рядку. Всі символи в рядку перетворити в прописні
4.	Вставити один рядок в інший, починаючи з довільного індексу рядка
5.	Здійснити в рядку пошук заданого слова
6.	Дані два рядки: *A* і *B*. Необхідно створити третій рядок, в якому потрібно зібрати елементи рядків *A* і *B*, які не є загальними для них
7.	Замінити в рядку один заданий символ іншим. Прибрати в рядку всі пропуски
8.	Визначити кількість букв, цифр і решти символів, присутніх у рядку
9.	Виключити задане слово з рядка. Якщо такого слова немає, функція повинна повернути 0
10.	Створити функцію для виводу російського тексту у вікно консолі
11.	Вставити заданий символ у рядок на задану позицію. Визначити позицію першого заданого символу в рядку
12.	Даний рядок символів *S*. Надрукувати всі вхідні в цей рядок заголовні латинські букви від '*A*' до '*Z*' в алфавітному порядку
13.	Всі символи в рядку перетворити в рядкові. Замінити в ньому всі пропуски на табуляції
14.	З клавіатури вводиться ціле число в діапазоні від 0 до 100. Необхідно вивести його прописний строковий еквівалент
15.	Скопіювати задану кількість символів рядка. Замінити перші n символів рядка заданим символом
16.	Замінити всі символи рядка заданим символом за винятком пропусків і розділових знаків. Визначити кількість проведених замін
17.	Зібрати в рядок усі слова, що починаються з голосних з 2-х інших рядків

Продовження табл. 19.5

№ вар.	Завдання
18.	Порівняти два рядки. Якщо рядок1 > рядка2, функція повертає значення більше за 0, якщо рядок1 < рядка2, функція повертає значення меньше за 0, якщо рядок1 = рядку2, функція повертає значення, що дорівнює 0
19.	Знайти місце першої появи в рядку *S*1 якого-небудь символа з рядка *S*2, і якщо рядок *S*1 не містить символів рядка *S*2, то функція повертає значення –1
20.	Порівняти перші *n* символів двох рядків без урахування регістра символів. Якщо рядок1 > рядка2, функція повертає значення більше за 0, якщо рядок1 < рядка2, функція повертає значення меньше за 0, якщо рядок1 = рядку2, функція повертає значення, що дорівнює 0
21.	Відсортувати букви в рядку у порядку зростання з урахуванням регістра
22.	Перевірити коректність розстановки дужок у виразі
23.	Об’єднати два рядки. Перші *n* символів одержаного рядка інвертувати
24.	Перетворити рядок в ціле число

Підготувати звіт про виконання лабораторної роботи, який повинен включати:

- номер, тему, мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- схему програми, виконану відповідно до стандартів;
- короткий опис програми;
- таблицю імен змінних, подану у вигляді табл. 19.2;
- текст програми мовою С++;
- результат при запуску програми;
- помилки, що виникли при виконанні програми.

Питання для підготовки до захисту лабораторної роботи

1. Що таке рядок? Як оголосити рядок?
2. Що таке покажчик? Як оголосити та ініціалізувати покажчик?
3. Чим відрізняється передача параметрів у функцію за значенням і за посиланням?
4. Чим відрізняється передача параметрів у функцію за значенням і за покажчиком?
5. Чим відрізняється передача у функцію цілочисельного масиву від рядка символів?
6. Як визначається кінець рядка?
7. Який зв’язок між масивами і покажчиками?
8. Чи можна збільшити значення покажчика на ціле число?

19.5. Багатовимірні динамічні масиви

Мета лабораторної роботи

Одержати основні навики роботи з багатовимірними динамічними масивами.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і відповідні розділи цього посібника, такі питання:

– покажчики: визначення і використання;
– динамічний розподіл пам’яті;
– покажчики на покажчики;
– масиви покажчиків;
– багатовимірні динамічні масиви.

Написати програму, яка виконувала б завдання, наведене в табл. 19.6, згідно з варіантом і відповідала таким вимогам:

– програма повинна містити функції, необхідні для вирішення завдання;
– кількість елементів масиву задається користувачем або визначається в процесі виконання програми;
– програма повинна передбачати ініціалізацію масиву за вибором користувача: з клавіатури або випадково (за умовчанням);
– на екран повинні виводитися початкові дані, а також результати всіх операцій;
– програма повинна мати зрозумілий і доступний будь-якому користувачу інтерфейс;
– програма повинна виконуватися доти, поки користувач не захоче припинити роботу з програмою.

Таблиця 19.6

Варіанти завдань

№ вар.	Завдання
1.	Видалити рядок з двовимірного масиву за вказаним номером
2.	Додати рядок до двовимірного масиву у вказану позицію
3.	Додати рядок до двовимірного масиву в початок
4.	Додати рядок до двовимірного масиву в кінець
5.	Додати стовпець до двовимірного масиву у вказану позицію
6.	Видалити стовпець з двовимірного масиву за вказаним номером
7.	Відсортувати двовимірний динамічний масив чисел методом «бульбашки»
8.	Знайти у двовимірному динамічному масиві задане значення. Вивести номер рядка і стовпця, де зустрілося значення

Продовження табл. 19.6

№ вар.	Завдання
9.	Ввести прізвища студентів, а потім знайти серед них щонайдовше прізвище. Кількість прізвищ студентів указує користувач
10.	Ввести прізвища студентів, а потім відсортувати їх за збільшенням методом «бульбашки». Кількість прізвищ студентів указує користувач
11.	Здійснити циклічний зсув стовпців двовимірного масиву вліво на задане число позицій
12.	Здійснити циклічний зсув рядків двовимірного масиву вліво на задане число позицій
13.	Здійснити циклічний зсув стовпців двовимірного масиву вправо на задане число позицій
14.	Здійснити циклічний зсув рядків двовимірного масиву вправо на задане число позицій
15.	Здійснити зсув стовпців двовимірного масиву вліво на задане число позицій. Позиції, що звільнилися, заповнити нулями
16.	Ущільнити задану матрицю, видаляючи з неї рядки і стовпці, заповнені нулями
17.	Даний двовимірний масив. Переставити рядки, щоб вони розташовувалися так (приклад для масиву розміром 12): перша, дванадцята, друга, одинадцята, ..., п’ята, восьма, шоста, сьома
18.	Даний двовимірний масив. Перенести перші k стовпців у кінець масиву, дотримуючи порядку їх проходження
19.	Даний двовимірний масив. Перенести перші k рядків у середину масиву, дотримуючи порядку їх проходження
20.	Ввести прізвища студентів, а потім знайти серед них однофамільців. Кількість прізвищ студентів указує користувач
21.	Даний двовимірний масив цілих чисел. Вставити в нього рядок з нулів між усіма рядками, в яких кількість додатних елементів дорівнює кількості від’ємних
22.	Даний двовимірний масив. Видалити з нього всі рядки з n-го по m-й ($n < m$)
23.	Даний двовимірний масив. Переставити в зворотному порядку стовпці, розташовані між n-м і m-м стовпцями ($n < m$)
24.	Даний двовимірний масив. Поміняти місцями перший рядок і рядок, в якому знаходиться перший нульовий елемент

Підготувати звіт про виконання лабораторної роботи, який повинен включати:

- номер, тему, мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- схему програми, виконану відповідно до стандартів;
- короткий опис програми;

– таблицю імен змінних, подану у вигляді табл. 19.2;
– текст програми мовою С++;
– результат при запуску програми;
– помилки, що виникли при виконанні програми.

Питання для підготовки до захисту лабораторної роботи

1. Як виділити ділянку пам’яті в динамічній області?
2. Чим відрізняється статичне виділення пам’яті від динамічного?
3. Як звільнити пам’ять, виділену в динамічній області?
4. Що таке покажчик на покажчик?
5. У яких випадках використовуються покажчики на покажчики?
6. Як розіменувати покажчик на покажчик?
7. Як визначити масив покажчиків?
8. Як визначити двовимірний динамічний масив?
9. Як ініціалізувати двовимірний динамічний масив?
10. Як отримати доступ до елементів двовимірного динамічного масиву?

19.6. Визначений користувачем тип даних у вигляді структури

Мета лабораторної роботи

Навчитися визначати і використовувати структури як призначений для користувача тип даних.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і відповідні розділи цього посібника, такі питання:

– призначені для користувача типи даних;
– визначення структур;
– використання структур;
– масив структур;
– створення багатофайлового проекту.

Написати програму, яка виконувала б завдання, наведене в табл. 19.7, згідно з варіантом і відповідала таким вимогам:

– програма повинна містити функції, необхідні для вирішення поставленого завдання;

– програма повинна передбачати ініціалізацію структурної змінної за вибором користувача: з клавіатури або явно (за умовчанням);

– на екран повинні виводитися початкові дані, а також результати всіх операцій;

– програма повинна мати зрозумілий і доступний будь-якому користувачу інтерфейс;

– програма повинна виконуватися доти, поки користувач не захоче припинити роботу з програмою;

– проект програми повинен бути багатофайловий:
  1) функція main у файлі-джерелі, тобто з розширенням .cpp;
  2) решта функцій в іншому файлі-джерелі з розширенням .cpp;
  3) прототипи функцій у заголовному файлі з розширенням .h.

Таблиця 19.7

Варіанти завдань

№ вар.	Завдання
1.	Відомості про автомобіль складаються з його марки, номера і прізвища власника. Дані відомості про 13 автомобілів. Визначити кількість автомобілів даної марки
2.	Відомості про автомобіль складаються з його марки, номера і прізвища власника. Дані відомості про 10 автомобілів. Визначити марку і прізвище власника за номером автомобіля
3.	Задати структуру, що містить інформацію про багаж пасажира. Багаж пасажира характеризується прізвищем власника, кількістю речей і їх загальною вагою. Заповнити задану структуру інформацією про 10 пасажирів. З'ясувати, чи є такий пасажир, багаж якого складається з однієї речі вагою не менше 30 кг
4.	Відомості про автомобіль складаються з його марки, номера і прізвища власника. Дані відомості про 10 автомобілів. Знайти прізвища власників і номери автомобілів даної марки
5.	Дані відомості про 15 книг. Відомості про кожну з книг – це прізвище автора, назва книги і рік видання. Знайти назви книг даного автора, виданих з 1990 року
6.	Дані відомості про 16 книг. Відомості про кожну з книг – це прізвище автора, назва книги і рік видання. Визначити, чи є книги з назвою «Програмування». Якщо так, то вивести прізвище автора і рік видання
7.	Задати структуру, що містить інформацію про багаж пасажира. Багаж пасажира характеризується прізвищем власника, кількістю речей і їх загальною вагою. Заповнити задану структуру інформацією про 12 пасажирів. Знайти число пасажирів, які мають більше двох речей, і число пасажирів, кількість речей яких перевершує середнє число речей
8.	Дані відомості про 15 книг. Відомості про кожну з книг – це прізвище автора, назва книги і рік видання. Визначити кількість книг даного автора, виданих у період з 1990 по 2004 р.
9.	Дані відомості про 15 співробітників установи. Відомості про співробітників – це прізвище, ініціали і номер телефону. Знайти телефон співробітника за його прізвищем та ініціалами
10.	Дані відомості про 14 кубиків: довжина ребра в сантиметрах, його колір і матеріал (дерев'яний, металевий, пластмасовий). Знайти кількість дерев'яних кубиків з ребром 3 см і кількість металевих кубиків з ребром, більшим за 5 см

Продовження табл. 19.7

№ вар.	Завдання
11.	Дані відомості про 17 кубиків: довжина ребра в сантиметрах, його колір і матеріал (дерев'яний, металевий, пластмасовий). Знайти кількість кубиків даного кольору і їх сумарний об'єм
12.	Дані відомості про 13 співробітників установи. Відомості про співробітників – це прізвище, ініціали і номер телефону. Знайти прізвище та ініціали співробітника за його номером телефону
13.	Дані відомості про 14 співробітників установи. Відомості про співробітників – це прізвище, ініціали і номер телефону. Знайти номери телефонів однофамільців
14.	Дані відомості про 10 товарів, що експортуються: найменування товару, країна, куди експортується товар, і об'єм партії, що поставляється, в штуках. Знайти країни, в які експортується даний товар
15.	Дані відомості про 14 товарів, що експортуються: найменування товару, країна, куди експортується товар, і об'єм партії, що поставляється, в штуках. Знайти загальний об'єм експорту даної країни
16.	Дані відомості про 13 товарів, що експортуються: найменування товару, країна, куди експортується товар, і об'єм партії, що поставляється, в штуках. Визначити товари, які експортує дана країна
17.	Дані відомості про 13 речовин: указується назва речовини, його питома вага і провідність (провідник, напівпровідник, ізолятор). Знайти загальну питому вагу всіх напівпровідників
18.	Дані відомості про 15 речовин: указується назва речовини, його питома вага і провідність (провідник, напівпровідник, ізолятор). Визначити назви всіх ізоляторів
19.	Дані відомості про 13 речовин: указується назва речовини, його питома вага і провідність (провідник, напівпровідник, ізолятор). Визначити за назвою речовини його питому вагу і провідність
20.	Інформація про академічну групу включає назву групи, кількість осіб у групі і рік вступу у ВНЗ. Дана інформація про 10 груп. Визначити, в якому році був найбільший набір
21.	Інформація про академічну групу включає назву групи, кількість осіб у групі і рік вступу у ВНЗ. Дана інформація про 15 груп. Визначити, яку кількість студентів було набрано в дану групу в заданому році
22.	Інформація про академічну групу включає назву групи, кількість осіб у групі і середній бал групи. Дана інформація про 12 груп. Визначити групи, у яких найбільший і найменший середній бал
23.	Інформація про академічну групу включає назву групи, кількість осіб у групі і середній бал групи. Дана інформація про 16 груп. Скласти перелік груп, які мають середній бал, більший ніж 3,99

Закінчення табл. 19.7

№ вар.	Завдання
24.	Дані відомості про 10 співробітників установи. Відомості про співробітника – це прізвище, заробітна платня за місяць і стаж. Серед даних працівників знайти тих, чия зарплата за місяць нижче за середню по підприємству, а також роздрукувати список тих, хто пропрацював на підприємстві більше 10 років, з вказівкою їх прізвища, зарплати, стажу роботи
25.	Інформація про учня включає прізвище, стать, зріст і вагу. Дана інформація про 20 учнів класу. Визначити середню масу хлопчиків і середній зріст дівчаток. Хто з учнів класу найвищий?

Підготувати звіт про виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- схему програми, виконану відповідно до стандартів;
- короткий опис програми;
- таблицю імен змінних, подану у вигляді табл. 19.2;
- текст програми мовою С++;
- результат при запуску програми;
- помилки, що виникли при виконанні програми.

Питання для підготовки до захисту лабораторної роботи

1. Як оголосити структуру?
2. Що таке структура?
3. Що таке поля структури?
4. Як оголосити та ініціалізувати структурну змінну?
5. Яким чином отримати доступ до полів структури?
6. Яким чином оголошується та ініціалізується масив структурних змінних?
7. Яким чином отримати доступ до полів структури, використовуючи покажчик структурного типу?
8. Як створити багатофайловий проект?
9. Як додати заголовний файл до проекту?
10. Які дані можуть міститися в заголовному файлі?

ЧАСТИНА 4. КУРСОВЕ ПРОЕКТУВАННЯ
20. РОЗРОБКА ПРОГРАМИ МОВОЮ С++

20.1. Завдання на курсовий проект

Метою курсового проекту для студентів з дисципліни „Програмування” є вивчення методів програмної організації динамічних списків за допомогою мови програмування С++.

Необхідно написати програму, що демонструє роботу з динамічним списком.

У програмі слід застосувати функції, за допомогою яких можна створити список та відповідно до нього виконувати задані дії (згідно з варіантом):

– додавати елемент до списку (наприкінці списку);

– видаляти елемент зі списку (з кінця списку);

– видалити список;

– виводити весь список на екран.

Також необхідно виводити меню для вибору заданих операцій або для завершення програми.

Проект програми повинен бути багатофайловим, тобто прототипи функцій повинні бути у заголовному файлі, а визначення функцій – у файлі-джерелі.

При виконанні проекту студент одержує практичні навики з:

- використання і написання функцій;
- використання структур, покажчиків, масивів;
- проектування програмного забезпечення.

Для виконання проекту студенту необхідні знання з базових дисциплін (основи комп’ютерних мереж, вища математика, програмування).

Тема курсового проекту – використання динамічних списків.

Для проектування пропонуються: 25 варіантів типу списків, дані, що зберігатимуться у списку, дії, які необхідно виконувати із динамічним списком (табл. 20.1).

Таблиця 20.1

Варіанти завдання

№ вар.	Тип динамічного списку	Дані, що зберігатимуться у списку	Дії, які необхідно виконувати із динамічним списком
1	2	3	4
1.	Односпрямований	Прізвище працівника, його зарплата та стаж роботи	– додавати дані працівника до списку за алфавітом; – вилучати дані працівника із списку; – замінити дані працівника; – знайти працівників із заданою зарплатою;

Продовження табл. 20.1

1	2	3	4
			– підвищити зарплату працівникам, які мають стаж роботи більше 10 років на 10%;
2.	Двоспрямований	Дійсні числа	– знайти середнє арифметичне чисел списку; – замінити всі числа *x* на число *y*; – додавати та вилучати число; – перевірити, чи упорядкований список за зростанням; – інвертувати список;
3.	Двоспрямований	Температура повітря	– додавати дані про температуру повітря; – визначити день з найменшою або найбільшою температурою; – вивести температуру і день її вимірювання за спаданням; – знайти дні з однаковою температурою;
4.	Односпрямований	Звичайні дроби $\frac{P}{Q}$, (P – чисельник, Q – знаменник)	– додавати та вилучати дріб; – знайти суму, різницю, частку або добуток усіх дробів або заданих; – знайти дробі, що повторюються, та залишити тільки один з них; – скоротити всі дроби або задані;
5.	Двоспрямований	Слова	– додавати нові слова; – сортувати слова за алфавітом; – вилучати слова; – знайти найдовше слово; – знайти слова, що повторюються, та залишити тільки одне з них;
6.	Односпрямований	День тижня, час початку занять, назва предмету, прізвище викладача	– вивести розклад заданого викладача на тиждень або на заданий день; – додавати, вилучати або змінювати інформацію (не додавати інформацію до списку, якщо вона там вже є); – розрахувати кількість занять на тиждень для заданого викладача; – вивести список предметів, які читає викладач;

Продовження табл. 20.1

1	2	3	4
7.	Односпрямований	Назва предмета, номер групи, прізвище студента, його оцінка з предмета за підсумками поточної сесії	– додавати, вилучати або змінювати інформацію (не додавати інформацію до списку, якщо вона там вже є); – вивести інформацію про заданого студента; – розрахувати середній бал студента; – розрахувати кількість відмінників, хорошистів та двієчників у групі; – скласти перелік предметів, з яких студент отримав незадовільну оцінку;
8.	Двоспрямований	Ціле число	– додавати або вилучати число (не додавати число до списку, якщо воно там вже є); – додавати число на задану позицію; – знайти всі прості числа; – знайти всі паліндроми у списку; – перевірити, чи упорядкований список за спаданням;
9.	Двоспрямований	Ціле число	– створити список заданим розміром; – заповнити список випадковими числами; – упорядкувати список за зростанням; – знайти суму чисел, що знаходяться між мінімальним та максимальним числами; – додати або вилучити число;
10.	Односпрямований	Прізвище спортсмена-багатоборця, вид спорту і кількість набраних балів у цьому виді спорту	– додавати, вилучати або змінювати інформацію (не додавати інформацію до списку, якщо вона там вже є); – вивести інформацію про заданого спортсмена; – розрахувати кількість балів, що набрав спортсмен з усіх видів спорту; – вивести таблицю результатів змагання; – вивести на екран інформацію про спортсменів, які посіли 1, 2 та 3 місця;

Продовження табл. 20.1

1	2	3	4
11.	Двоспрямований	Ціле число	– додавати, вилучати або змінювати інформацію; – упорядкувати числа: спочатку від'ємні, потім додатні; – знайти числа, що повторюються; – вилучати всі входження заданого числа у список; – поміняти місцями максимальний та мінімальний елементи у списку; – видалити елементи, що кратні 5;
12.	Односпрямований	Найменування, кількість, ціна, термін зберігання товара	– додавати, вилучати або змінювати інформацію (не додавати інформацію до списку, якщо вона там вже є); – зменшити ціну товара, якщо товар зберігався на складі довше *n* місяців, у 2 рази; – вивести найменування товарів, кількість яких менше 3; – вивести інформацію про товар; – перевірити, чи є товар у наявності;
13.	Односпрямований	Прізвище працівника, його зарплата та стаж роботи	– додавати дані працівника до списку за алфавітом; – вилучати дані працівника із списку; – замінити дані працівника; – знайти працівників із заданою зарплатою; – підвищити зарплату працівникам, які мають стаж роботи більше 10 років на 10%;
14.	Двоспрямований	Ціле число	– додавати, вилучати або змінювати інформацію; – знайти кількість чисел, що не повторюються; – знайти добуток чисел, що знаходяться між елементами *n* та *m*; – вилучити елементи з *n*-го по *m*-й; – сортувати список за зростанням;
15.	Двоспрямований	Дійсні числа	– знайти середнє арифметичне чисел списку; – замінити всі числа *x* на число *y*;

Продовження табл. 20.1

1	2	3	4
			– додавати та вилучати число; – перевірити, чи упорядкований список за зростанням; – інвертувати список;
16.	Односпрямований	Слова – іменники	– додавати нові слова; – вилучати слова; – знайти найкоротше слово; – інвертувати список; – знайти слова, що не повторюються, та вивести їх на екран;
17.	Двоспрямований	Температура повітря	– додавати дані про температуру повітря; – визначити день з найменшою або найбільшою температурою; – вивести дані про температуру і день її вимірювання за спаданням; – знайти дні з однаковою температурою;
18.	Односпрямований	Звичайні дроби $\frac{P}{Q}$, (P – чисельник, Q – знаменник)	– додавати та вилучати дріб; – знайти суму, різницю, частку або добуток усіх дробів або заданих; – знайти дроби, що повторюються, та залишити тільки один з них; – скоротити всі дроби або задані;
19.	Односпрямований	День тижня, час початку занять, назва предмету, прізвище викладача	– вивести розклад заданого викладача на тиждень або на заданий день; – додавати, вилучати або змінювати інформацію (не додавати інформацію до списку, якщо вона там вже є); – розрахувати кількість занять на тиждень для заданого викладача; – вивести список предметів, які читає викладач;
20.	Двоспрямований	Ціле число	– додавати або вилучати число (не додавати число до списку, якщо воно там вже є); – додавати число на задану позицію; – знайти всі прості числа; – знайти всі паліндроми у списку;

Продовження табл. 20.1

1	2	3	4
			– перевірити, чи упорядкований список за спаданням;
21.	Двоспрямований	Ціле число	– створити список заданого розміру; – заповнити список випадковими числами; – упорядкувати список за зростанням; – знайти суму чисел, що знаходяться між мінімальним та максимальним числами; – додати або вилучити число;
22.	Односпрямований	Прізвище спортсмена-пловця, вид стилю, результат виступу у цьому виді	– додавати, вилучати або змінювати інформацію (не додавати інформацію до списку, якщо вона там вже є); – вивести інформацію про заданого спортсмена; – вивести таблицю результатів змагання; – вивести на екран інформацію про спортсменів, які посіли 1, 2 та 3 місця;
23.	Двоспрямований	Ціле число	– додавати, вилучати або змінювати інформацію; – упорядкувати числа: спочатку від'ємні, потім додатні; – знайти числа, що повторюються; – вилучати всі входження заданого числа у список; – поміняти місцями максимальний та мінімальний елементи у списку; – вилучати елементи, що кратні 5;
24.	Односпрямований	Найменування, кількість, ціна, термін зберігання товара	– додавати, вилучати або змінювати інформацію (не додавати інформацію до списку, якщо вона там вже є); – зменшити ціну товара, якщо товар зберігався на складі довше *n* місяців, у 2 рази; – вивести найменування товарів, кількість яких менше 3; – вивести інформацію про товар; – перевірити, чи є товар у наявності;

Закінчення табл. 20.1

1	2	3	4
	Односпря-мований	Найменування, кількість, ціна, термін зберігання товара	– додавати, вилучати або змінювати інформацію (не додавати інформацію до списку, якщо вона там вже є);зменшити ціну товара, якщо товар зберігався на складі довше *n* місяців, у 2 рази; – вивести найменування товарів, кількість яких менше 3; – вивести інформацію про товар; – перевірити, чи є товар у наявності;

20.2. Вимоги до складових курсового проекту

Курсовий проект включає текстову і графічну частини.

Текстова частина виконується у вигляді пояснювальної записки обсягом не менше ніж 20 сторінок рукописного, машинописного або машинного тексту без урахування додатків. Вона повинна у стислій і чіткій формі розкрити основні рішення, які були прийняті у курсовому проекті.

Пояснювальна записка містить:

– титульний аркуш (рис. 20.1);
– завдання на курсовий проект (рис. 20.2);
– реферат;
– зміст;
– вступ;
– опис визначення та використання динамічного списку;
– опис програмної організації динамічного списку;
– опис процесу розробки програми;
– опис програми;
– висновки;
– перелік посилань;
– додатки.

При оформленні курсового проекту перший аркуш завдання повинен мати вигляд, як показано на рис. 20.2, а далі наводяться дані, вибрані з табл. 20.1 відповідно до варіанта, призначеного викладачем.

Завдання та календарний план його виконання підписується студентом і керівником курсового проекту.

До додатків входять:

– відомість матеріалів курсового проекту (рис. 20.3);
– текст програм.

Пояснювальна записка є основним документом курсового проекту і її розділи повинні мати обсяги, наведені в табл. 20.2.

Графічна частина повинна містити прийняті у проекті рішення у вигляді схеми програми (креслення на одному аркуші формату А1).

Міністерство освіти і науки України
Державний ВНЗ
“НАЦІОНАЛЬНИЙ ГІРНИЧИЙ УНІВЕРСИТЕТ”

ІНСТИТУТ ЕЛЕКТРОЕНЕРГЕТИКИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра автоматизації та комп’ютерних систем

ПОЯСНЮВАЛЬНА ЗАПИСКА
курсового проекту

з дисципліни *”Програмування”*

Виконавець,
студент гр. КІіт-16-1 _____ *С.К. Семенов*
(підпис)

Керівник, проф. _____ *Л.І. Цвіркун*
(підпис)

Дніпропетровськ
2016

Рис. 20.1. Приклад оформлення титульного аркуша пояснювальної записки курсового проекту

ЗАВДАННЯ
на курсовий проект
з дисципліни *"Програмування"*
студента групи *КІіт-16-1 Семенова Семена Костянтиновича*

Розділ	Зміст завдання	Термін виконання
Визначення та використання динамічного списку	*Дати визначення та описати типи і використання динамічного списку*	*21.09.2016 р.*
Програмна організація динамічного списку	*Описати програмну організацію динамічного списку*	*15.10.2016 р.*
Розробка програми	*Розробити програму з використанням динамічного списку згідно із завданням. Описати процес розробки програми*	*15.11.2016 р.*
Опис програми	*Описати програму*	*01.12.2016 р.*
Графічна частина	*Скласти схеми програми (креслення на 1 аркуші формату А1)*	*15.12.2016 р.*

Завдання видав проф. _____ *Л.І. Цвіркун*
(підпис)

Завдання прийняв до виконання _____ *С.К. Семенов*
(підпис)

Дата видачі завдання *01.09.2016 р.*

Термін подання курсового проекту до захисту *15.12.2016 р.*

Рис. 20.2. Приклад оформлення завдання на курсовий проект

Додаток А
Відомість проекту

№ рядка	Формат	Позначення	Найменування	Кількість аркушів	Шифр документа	Примітка
1			**Документація**			
2	А4	КП 16.14.ПЗ	Пояснювальна записка	28	ПЗ	
3			**Графічні матеріали**			
4	А1	КП 16.14.01.СП	Схема програми	1	СП	
5						

Зм.	Лист	№ докум.	Підп.	Дата	**АКС.КП.16.14.ДА.ПЗ**			
Розробив		Семенов С.К.		15.12.16	Програмне забезпечення з використанням динамічного списку Відомість проекту	Літ.	Аркуш	Аркушів
Перевірив		Цвіркун Л.І.		15.12.16			1	1
						ФІТ КІіт-16-1		
Н. контр.		Цвіркун Л.І.		15.12.16				
Затвердив		Цвіркун Л.І.		15.12.16				

Рис. 20.3. Приклад оформлення відомості матеріалів курсового проекту

Таблиця 20.2

Обсяги розділів пояснювальної записки

Назва розділу	Обсяг, с.
Завдання	2
Реферат	1
Зміст	1
Визначення та використання динамічного списку	До 4–5
Програмна організація динамічного списку	До 4–5
Розробка програми	До 5–6
Опис програми	До 5–6
Висновки	1
Перелік посилань	1–2

20.3. Рекомендації до оформлення курсового проекту

20.3.1. Загальні положення

Матеріали курсового проекту повинні бути оформлені у вигляді альбому технічної документації. Курсовий проект з дисципліни „Програмування” включає в себе документи тільки одного виду: програмні.

Оформлення технічної документації цього виду має свої особливості, що регулюються державними стандартами. Так, правила і положення щодо порядку розробки, оформлення і застосування програмних документів визначають комплексом державних стандартів ЄСПД (клас 19). Виконання вимог державних стандартів при оформленні матеріалів курсового проекту – обов’язкове.

У відповідних місцях текстової частини пояснювальної записки неодмінно роблять посилання на креслення, що додаються.

Пояснювальна записка повинна бути викладена літературною мовою, технічно грамотно.

Якщо курсовий проект виконується на аркушах з рамкою відповідно до ГОСТ 2.104-68, то відстань від рамки до тексту така: на початку рядків – не менше 5, у кінці рядків – не менше 3, від верхнього і нижнього боків рамки – не менше 10 мм.

Відомість матеріалів курсового проекту оформляється згідно з рис. 20.3. Графа „шифр документа” заповнюється відповідно до ГОСТ 2.701-84.

20.3.2. Оформлення пояснювальної записки

Пояснювальну записку курсового проекту виконують на одному боці аркушів білого паперу формату А4 (210х297 мм) від руки (чорним кольором), машинописним або машинним (за допомогою комп’ютерної техніки) способом.

Пояснювальну записку виконують згідно з вимогами стандартів щодо оформлення звітів і підготовки документів з використанням друкувальних та графічних пристроїв виводу ЕОМ відповідно до ДСТУ 3008-98.

При використанні машинописного способу пояснювальну записку друкують через півтора інтервала, а машинного – з розрахунку не більше 40 рядків на сторінці за умови рівномірного її заповнення. Висота літер і цифр – не менше ніж 1,8 мм, береги: верхній, лівий і нижній – не менше 20, правий – не менше 10 мм.

Допускається включення до пояснювальної записки сторінок, зроблених методом репрографії, а також можна окремі частини виконувати різними способами (від руки, машинописним або машинним).

Необхідно дотримуватись рівномірної щільності, контрастності й чіткості зображення. Лінії, літери, цифри та інші знаки мають бути чіткі, не розпливчасті, однаково чорні на всіх сторінках записки.

Окремі слова, формули, знаки, які вписують у надрукований текст, мають бути чорного кольору, щільність вписаного тексту повинна максимально наближуватись до щільності основного зображення.

Помилки, описки та графічні неточності допускається виправляти підчищенням або зафарбовуванням білою фарбою і нанесенням машинописним способом або від руки чорним кольором на тому ж місці або між рядками виправленого зображення.

Прізвища, назви установ, організацій, фірм та інші власні назви у записці наводять мовою оригіналу. Допускається транслітерувати власні назви і наводити назви організацій у перекладі на мову звіту, додаючи (при першій згадці) назву оригіналу.

Скорочення слів і словосполучень у пояснювальній записці вживати відповідно до чинних стандартів з бібліотечної та видавничої справи.

Структурні елементи “ЗМІСТ”, “ВСТУП”, “ВИСНОВКИ”, “ПЕРЕЛІК ПОСИЛАНЬ” не нумерують, а їх назви служать заголовками структурних елементів.

Розділи і підрозділи повинні мати **заголовки**. Пункти і підпункти можуть мати заголовки.

Заголовки структурних елементів записки і заголовки розділів слід розташовувати посередині рядка і друкувати великими літерами без крапки в кінці, не підкреслюючи.

Заголовки підрозділів, пунктів і підпунктів звіту слід починати з абзацного відступу і друкувати маленькими літерами (перша літера велика), не підкреслюючи, без крапки у кінці.

Абзацний відступ повинен бути однаковим упродовж усього тексту звіту і дорівнювати п’яти знакам.

Якщо заголовок складається з двох і більше речень, їх розділяють крапкою. Перенесення слів у заголовку розділу не допускається.

Відстань між заголовком і подальшим чи попереднім текстом має бути:

- за машинописного способу – не менше ніж три інтервали;
- за машинного способу – не менше ніж два рядки.

Відстань між рядками заголовку, а також між двома заголовками приймають такою, як у тексті.

Не допускається починати назву розділу, підрозділу, а також пункту й підпункту в нижній частині сторінки, якщо після них поміщується тільки один рядок тексту.

Оформлення тексту, ілюстрацій і таблиць за машинного способу виконують відповідно до вимог стандарту з оформлення документації, звітів у галузі науки і техніки – з урахуванням можливостей комп'ютерного обладнання згідно з ДСТУ 3008-98.

Сторінки пояснювальної записки слід нумерувати арабськими цифрами, додержуючись наскрізної нумерації впродовж усього тексту записки. Номер сторінки проставляють у правому верхньому куті сторінки без крапки у кінці.

Додатки, які мають свій структурний елемент „ЗМІСТ", необхідно нумерувати ще й внизу, посередині сторінки.

Титульний аркуш включають до загальної нумерації сторінок записки. Номер сторінки на титульному аркуші не проставляють.

Ілюстрації і таблиці, розміщені на окремих сторінках, включають до загальної нумерації сторінок записки.

Розділи, підрозділи, пункти, підпункти звіту слід нумерувати арабськими цифрами.

Розділи пояснювальної записки повинні мати порядкову нумерацію і позначатися арабськими цифрами без крапки, наприклад, 1, 2, 3 і т. д.

Підрозділи повинні мати порядкову нумерацію в межах кожного розділу.

Номер підрозділу складається з номера розділу і порядкового номера підрозділу, відокремлених крапкою.

Після номера підрозділу крапку не ставлять, наприклад, 1.1, 1.2 і т. д.

Пункти повинні мати порядкову нумерацію в межах кожного розділу або підрозділу.

Номер пункту складається з номера розділу і порядкового номера пункту або з номера розділу, порядкового номера підрозділу та порядкового номера пункту, відокремлених крапкою. Після номера пункту крапку не ставлять, наприклад, 1.1, 1.2 або 1.1.1, 1.1.2 і т. д.

Якщо текст поділяють тільки на пункти, їх слід нумерувати порядковими номерами.

Номер підпункту складається з номера розділу, порядкового номера підрозділу, порядкового номера пункту і порядкового номера підпункту, відокремлених крапкою, наприклад, 1.1.1.1, 1.1.1.2, 1.1.1.3 і т. д.

Якщо розділ не має підрозділів і поділяється на пункти й підпункти, номер підпункту складається з номера розділу, порядкового номера пункту та порядкового номера підпункту, відокремлених крапкою, наприклад, 1.1.3, 1.2.1 і т. д. Після номера підпункту крапку не ставлять.

Якщо розділ або підрозділ складається з одного пункту або пункт складається з одного підпункту, його нумерують.

Ілюстрації (креслення, рисунки, графіки, схеми, діаграми, фотознімки) слід розміщувати у записці безпосередньо після тексту, де вони згадуються вперше, або на наступній сторінці. На всі ілюстрації мають бути посилання у записці.

Креслення, рисунки, графіки, схеми, діаграми мають відповідати вимогам стандартів ЄСКД та ЄСПД.

Фотознімки, розмір яких менше за формат А4, мають бути наклеєні на форматні аркуші (А4).

Ілюстрації повинні мати назву, яку розміщують безпосередньо під ілюстрацією.

Ілюстрація позначається словом “Рисунок”, яке разом з назвою ілюстрації розміщують після пояснювальних даних, наприклад, “Рисунок 3.1 – Схема розміщення”.

Ілюстрації слід нумерувати арабськими цифрами порядковою нумерацією у межах розділу, за винятком ілюстрацій, наведених у додатках.

Номер ілюстрації складається з номера розділу і порядкового номера ілюстрації, відокремлених крапкою, наприклад, рисунок 3.2 – другий рисунок третього розділу.

Якщо у пояснювальній записці вміщено тільки одну ілюстрацію, її нумерують.

Якщо ілюстрація не вміщується на одній сторінці, її можна переносити на інші, але назва ілюстрації повинна бути на першій сторінці. Пояснювальні дані зазначають на кожній сторінці: “Рисунок __, аркуш __”.

Ілюстрації за необхідності можуть бути перелічені у змісті із зазначенням їх номерів, назв і номерів сторінок.

Цифровий матеріал, як правило, оформлюють у вигляді **таблиць**.

Горизонтальні та вертикальні лінії, які розмежовують рядки таблиці, а також лінії зліва, справа і знизу, що обмежують таблицю, можна не проводити, якщо їх відсутність не утруднює користування таблицею.

Таблицю слід розташовувати безпосередньо після тексту, у якому вона згадується вперше, або на наступній сторінці. На всі таблиці мають бути посилання в тексті пояснювальної записки.

Таблиці слід нумерувати арабськими цифрами порядковою нумерацією в межах розділу, за винятком таблиць, що наводяться у додатках.

Номер таблиці складається з номера розділу і порядкового номера таблиці, відокремлених крапкою, наприклад, таблиця 2.1 – перша таблиця другого розділу.

Якщо у пояснювальній записці одна таблиця, її нумерують.

Таблиці повинні мати назву, яку друкують малими літерами (крім першої великої) і розміщують над таблицею. Назва має бути стислою і відбивати зміст таблиці.

Якщо рядки або графи таблиці виходять за межі формату сторінки, таблицю поділяють на частини, розміщуючи одну частину під одною або поруч, або переносять частину таблиці на наступну сторінку з повторенням у кожній частині таблиці її головки.

При поділі таблиці на частини допускається її головку або боковик заміняти відповідно номерами граф чи рядків, нумеруючи їх арабськими цифрами у першій частині таблиці.

Слово “Таблиця” вказують один раз зліва над першою частиною таблиці, над іншими частинами пишуть “Продовження таблиці ...”, наприклад: “Продовження таблиці 2.3” – третя таблиця другого розділу.

Заголовки граф таблиці починають з великої літери, а підзаголовки – з малої, якщо вони складають одне речення із заголовком без крапки у кінці.

Підзаголовки, що мають самостійне значення, пишуть з великої літери. Заголовки і підзаголовки граф указують в однині.

Таблиці за необхідності можуть бути перелічені у записці із зазначенням їх номерів, назв (якщо вони є) та номерів сторінок, на яких вони розміщені.

Переліки також можуть бути наведені всередині пунктів або підпунктів. Перед переліком ставлять двокрапку, а потім малу літеру української абетки з дужкою (крім літер ґ, є, з, і, ї, й, о, ч) або, не нумеруючи, дефіс (перший рівень деталізації). Для подальшої деталізації переліку використовують арабські цифри з дужкою (другий рівень деталізації). Переліки першого рівня деталізації друкують малими літерами з абзацного відступу, другого – з відступом відносно місця розташування переліків першого рівня.

Формули та рівняння розташовують безпосередньо після тексту, в якому вони згадуються, посередині сторінки. Вище і нижче кожної формули або рівняння повинно бути залишено не менше одного вільного рядка.

Формули і рівняння у записці (за винятком формул і рівнянь, наведених у додатках) слід нумерувати порядковою нумерацією в межах розділу.

Номер формули або рівняння складається з номера розділу і порядкового номера формули або рівняння, відокремлених крапкою, наприклад, формула (1.3) – третя формула першого розділу.

Номер формули або рівняння зазначають на рівні формули або рівняння в дужках у крайньому правому положенні на рядку.

Пояснення значень символів і числових коефіцієнтів, що входять до формули чи рівняння, слід наводити безпосередньо під формулою у такій послідовності, у якій вони наведені у формулі чи рівнянні.

Пояснення значення кожного символу та числового коефіцієнта слід давати з нового рядка. Перший рядок пояснення починають з абзацу словом “де” без двокрапки.

Переносити формули чи рівняння на наступний рядок допускається тільки на знаках „+” або „х” (знак множення), повторюючи знак операції на початку наступного рядка.

Якщо у пояснювальній записці тільки одна формула чи рівняння, їх нумерують.

Формули, що йдуть одна за одною й не розділені текстом, відокремлюють комою.

Посилання в тексті пояснювальної записки на джерела слід зазначати арабськими цифрами у квадратних дужках відповідно до переліку посилань, наприклад, “... у роботах [4–6]...”.

При посиланнях на розділи, підрозділи, пункти, підпункти, ілюстрації, таблиці, формули, рівняння, додатки зазначають їх номери.

При посиланнях слід писати: “... у розділі 4...”, “... дивись 2.1...”, “... за 3.3.4...”, “... відповідно до 2.3.4.1...”, “... на рисунку 1.3...”, “... у таблиці 3.2...”, “... (дивись 3.2)...”, “... за формулою (3.1)...”, “... у рівняннях (1.23) – (1.25)...”, “... у додатку Б...”.

20.3.3. Оформлення графічної частини

Графічний матеріал проекту виконується на папері стандартного формату А1 (594х841 мм) креслярським олівцем (в окремих випадках допускається тушшю) або за допомогою спеціалізованих пакетів прикладних програм та засобів оргтехніки (принтерів, плотерів та ін.). Аркуші графічної частини повинні мати рівномірне заповнення. Кожний аркуш графічної частини повинен супроводжуватися основним написом згідно з вимогами стандартів ЄСКД .

При виконанні схем програм використовують умовні графічні позначення, наведені в стандарті ЄСПД (ГОСТ 19.701-90).

ЧАСТИНА 5. НАВЧАЛЬНА ПРАКТИКА

21. ЗАСТОСУВАННЯ ЗНАНЬ МОВИ С++ ДЛЯ ПРАКТИЧНОГО ПРОГРАМУВАННЯ

Індивідуальні завдання для навчальної практики з дисципліни "Програмування" розраховані для студентів напряму підготовки 6.050102 Комп'ютерна інженерія.

Індивідуальні завдання можуть виконуватися в будь-яких версіях середовища програмування Visual C++ і операційної системи Windows.

Перелік індивідуальних завдань включає чотири найменування, до кожного з яких підготовлені методичні вказівки. Номер завдання, яке видається студенту, відповідає номеру, під яким він записаний у списку групи.

За результатами роботи студент оформляє звіт, захищає виконане індивідуальне завдання і відповідає на контрольні питання.

Зміст звіту повинен складатися з:

- титульного аркуша (рис. 21.1);
- описа завдання, включаючи номер і початкові дані вибраного варіанта;
- словесно-аналітичного опису алгоритму;
- таблиці імен;
- схеми програми;
- програми мовою С++;
- результатів обробки тестових даних при запуску програми;
- тестових даних і короткого опису програми;
- завантажувального модуля на диску.

Міністерство освіти і науки України
Державний ВНЗ
“НАЦІОНАЛЬНИЙ ГІРНИЧИЙ УНІВЕРСИТЕТ”

ІНСТИТУТ ЕЛЕКТРОЕНЕРГЕТИКИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра автоматизації та комп’ютерних систем

ЗВІТ
З НАВЧАЛЬНОЇ ПРАКТИКИ

дисципліна *”Програмування”*

Виконавець,
студент гр. КІіт-16-1 _____ *С.К. Семенов*
(підпис)

Керівник, проф. _____ *Л.І. Цвіркун*
(підпис)

Дніпропетровськ
2016

Рис. 21.1. Приклад оформлення титульного листа звіту з навчальної практики

21.1. Сортування і пошук

Мета індивідуального завдання

Вивчити деякі алгоритми сортування і пошуку, а також навчитися реалізовувати ці алгоритми за допомогою мови програмування С++.

Організація виконання індивідуального завдання

Для виконання індивідуального завдання необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій та методичні вказівки, методи сортування і пошуку.

Скласти схему програми і програму мовою С++, що виконує такі дії:

1) формування одновимірного масиву із заданих елементів двовимірного масиву розміром *nxn* (*n* – задається користувачем). Вибірку елементів проводити у вказаному напрямку (табл. 21.1) та згідно з варіантом. Елементи двовимірного масиву повинні задаватися користувачем або за умовчанням (за допомогою генератора випадкових чисел);

2) пошук заданого користувачем елементу в одновимірному масиві, лінійним методом і підрахунок кількості порівнянь;

3) сортування вектора, використовуючи методи сортування і пошуку, згідно з варіантом (табл. 21.1);

4) пошук заданого користувачем елементу в одновимірному масиві двійковим методом і підрахунок кількості порівнянь.

Кожне завдання повинне бути реалізоване у вигляді функції. Проект програми має бути багатофайловим.

Оцінити ефективність лінійного і двійкового пошуку, а також використовуваного методу сортування.

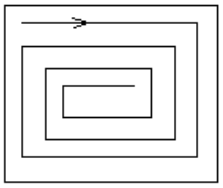
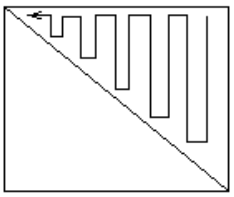
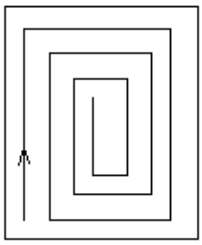
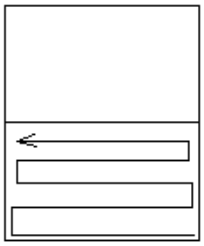
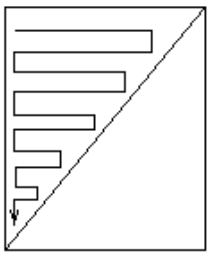
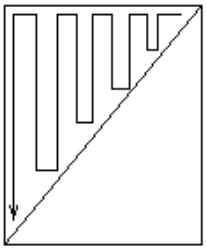
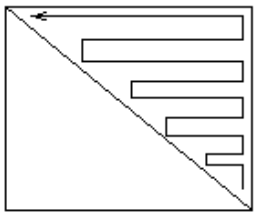
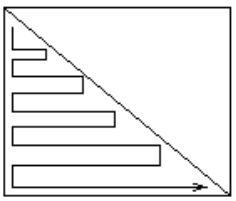
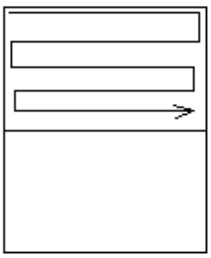
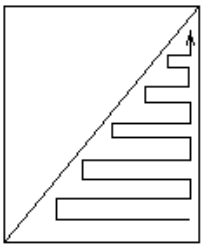
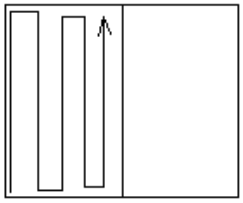
Оформити звіт про виконане завдання.

Питання для підготовки до захисту індивідуального завдання

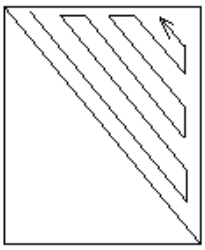
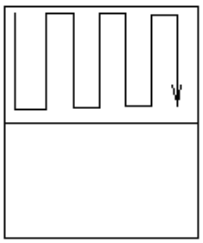
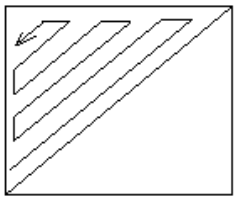
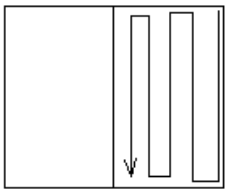
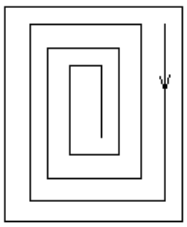
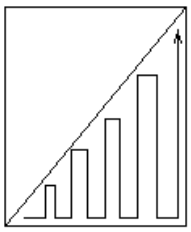
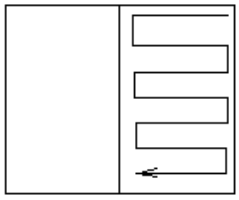
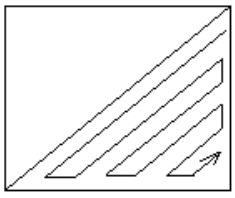
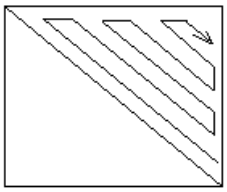
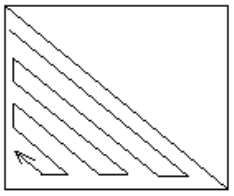
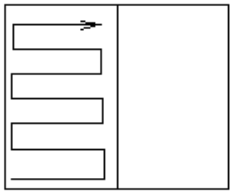
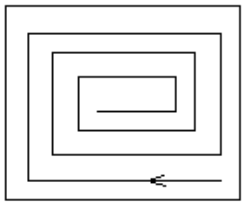
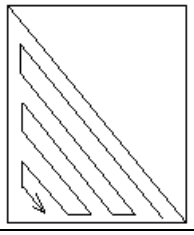
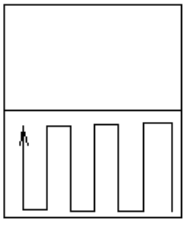
1. У чому полягає лінійний пошук?
2. У чому полягає двійковий пошук?
3. Коли застосовують двійковий пошук?
4. Доведіть, що для двійкового пошуку в масиві з 1023 елементів достатньо 10 порівнянь.
5. Які відомі способи сортування?
6. Який спосіб сортування є найпростішим?
7. У чому полягає спосіб швидкого сортування?
8. У чому полягає спосіб сортування Шела?
9. У чому полягає сортування вставками?
10. Який спосіб сортування розробив Е. Хоар?
11. При сортуванні вводиться термін „бар’єр”. Що він означає?
12. При якому способі сортування вводиться термін „бар’єр”?
13. Який спосіб сортування належить до методів сортування за місцем?
14. Які методи сортування є стійкими за місцем?

Таблиця 21.1

Варіанти завдань

№ вар.	Метод сортування масиву	Напрямок вибірки елементів для формування вектора	№ вар.	Метод сортування масиву	Напрямок вибірки елементів для формування вектора
1.	Швидке сортування		7.	Сортування Шелла	
2.	Сортування Шелла		8.	Сортування вставками	
3.	Сортування вставками		9.	Швидке сортування	
4.	Швидке сортування		10.	Сортування Шелла	
5.	Сортування Шелла		11.	Сортування вставками	
6.	Сортування вставками		12.	Швидке сортування	

Продовження табл. 21.1

№ вар.	Метод сортування масива	Напрямок вибірки елементів для формування вектора	№ вар.	Метод сортування масиву	Напрямок вибірки елементів для формування вектора
13.	Швидке сортування		20.	Сортування Шелла	
14.	Сортування Шелла		21.	Сортування вставками	
15.	Сортування вставками		22.	Швидке сортування	
16.	Швидке сортування		23.	Сортування Шелла	
17.	Сортування Шелла		24.	Сортування вставками	
18.	Сортування вставками		25.	Швидке сортування	
19.	Швидке сортування		26.	Сортування Шелла	

21.2. Створювання власних класів

Мета індивідуального завдання

Ознайомитися з основними принципами об'єктно-орієнтованого програмування, навчитися створювати власні класи мовою програмування С++ і працювати з екземплярами класів.

Організація виконання індивідуального завдання

Для виконання індивідуального завдання необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій та методичні вказівки, основні принципи об'єктно-орієнтованого програмування.

Скласти схему програми і програму мовою С++, що виконує такі дії:

а) оголошення класу згідно з варіантом (табл. 21.2);

б) ілюстрування всіх властивостей класу.

Створити багатофайловий проект, тобто оголошення класу, визначення функцій-членів класу, а також розміщення демонстраційної програми в окремих модулях (файлах).

Оформити звіт про виконане завдання.

Питання для підготовки до захисту індивідуального завдання

1. Які поняття в мові С++ зв'язуються з класом?
2. Дайте визначення класу в мові С++.
3. Скільки понять у мові С++ зв'язуються з класом?
4. Із скількох частин складається визначення класу в мові С++?
5. Як оголошуються елементи-дані класу?
6. Що задають функції-елементи класу?
7. Як оголошуються функції-елементи класу?
8. Якими властивостями володіє функція-елемент класу?
9. Для чого застосовується засіб утаювання даних?
10. На які частини розбивають тіло класу?
11. Які рівні доступу назначаються для частин тіла класу?
12. Як виконується визначення об'єктів класу?
13. Що таке конструктор у мові С++?
14. Що означає в мові С++ деструктори і деструкція?
15. Наведіть приклад визначення деструкції.

Таблиця 21.2

Варіанти завдань

№ вар.	Опис класу
1.	Дайте визначення класу, що зображує трикутник. Клас має атрибути – координати вершин трикутника. Функції-члени повинні виконувати такі дії: – створення об'єкта і його ініціалізація;

Продовження табл. 21.2

№ вар.	Опис класу
	– призначення початкових значень елементам даних; – відображення трикутника на екрані за допомогою заданого символу; – обчислення довжин сторін трикутника; – обчислення периметра і площі трикутника ($S = \frac{abc}{4R}$, де *a*, *b*, *c* – довжини сторін трикутника, *R* – радіус описаного кола).
2.	Дайте визначення класу, що зображує чотирикутник. Елементи даних повинні містити координати точок чотирикутника. Функції-члени повинні виконувати такі дії: – створення об'єкта і його ініціалізація; – призначення початкових значень елементам даних; – відображення чотирикутника на екрані за допомогою заданого символу; – обчислення довжини і ширини прямокутника; – обчислення периметра і площі чотирикутника.
3.	Дайте визначення класу, що зображує коло. Клас має атрибути: радіус кола, який за умовчанням дорівнює 1, і центр кола. Клас має функції-члени, які повинні виконувати такі дії: – створення об'єкта і його ініціалізація; – призначення початкових значень елементам даних; – обчислення довжини кола $\left(L = 2\pi r\right)$; – вивід кола на екран за допомогою заданого символу; – знаходження площі, що описується колом $\left(S = \pi r^2\right)$.
	Комплексне число складається з двох частин: дійсної та уявної. Одним із способів запису комплексного числа є такий: (3.0, 4.5i). Тут 3.0 – дійсна частина числа, а 4.5 – уявна. Припустімо, що *a*=(*A*, *Bi*) *c*=(*C*, *Di*). Ось деякі операції, виконувані над комплексними числами: – додавання: *a* + *c* = (*A* + *C*, (*B* + *D*) *i*); – віднімання: *a* - *c* = (*A* - *C*, (*B* - *D*) *i*); – множення: *a c* = (*A C* - *B D*, (*A D* + *B C*) *i*). Визначте клас, що оперує комплексними числами.
5.	Створіть клас "Дата" з такими можливостями: – створення об'єкта і його ініціалізація; – призначення початкових значень елементам даних; – вивід дат у декількох форматах: а) DD YYYY; б) MM/DD/YY; в) місяць, число, рік; – визначення дня тижня, якщо відомий один з днів тижня поточного місяця.

Продовження табл. 21.2

№ вар.	Опис класу
6.	Розробіть клас “Person”, що містить члени для зберігання імені, дати народження, статі, номера телефону. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних; – форматований вивід даних про людину; – визначення віку людини.
7.	Дайте визначення класу, що зберігає список товарів. Клас має атрибути: назва продукту, його ціна. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних; – обчислення ціни продукту з 20%-ною знижкою; – вивід на екран величини знижки на продукт; – вивід на екран даних про продукт.
8.	Дайте визначення класу, що зображує конус. Елементи даних повинні містити радіус основи конуса і його висоту. Функції-члени повинні виконувати такі дії, як: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних; – обчислення довжини кола основи $\left(L = 2\pi r\right)$, площі бічної поверхні $\left(S_{áîê} = \pi rL\right)$ та об’єму конуса $\left(V = \frac{1}{3}\pi r^2 h\right)$; – вивід усіх відомих параметрів фігури.
9.	Дайте визначення класу, що зображує циліндр. Елементи даних повинні містити радіус підстави циліндра і його висоту. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних; – обчислення довжини кола основи $\left(L = 2\pi r\right)$, площі бічної поверхні $\left(S_{áîê} = 2\pi rh\right)$ та об’єму циліндра $\left(V = \pi r^2 h\right)$; – вивід усіх відомих параметрів фігури.
10.	Дайте визначення класу, що зображує кулю. Елементи даних повинні містити радіус кулі. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних; – обчислення площі $\left(S = 4\pi r^2\right)$ та об’єму кулі $\left(V = \frac{4}{3}\pi r^3\right)$; – вивід усіх відомих параметрів фігури.

Продовження табл. 21.2

№ вар.	Опис класу
11.	Дайте визначення класу, що зберігає одновимірний масив цілих чисел. Елементи даних повинні містити кількість елементів масиву, масив цілих чисел. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних; – інвертування елементів масиву; – знаходження суми елементів у масиві; – форматований вивід масиву на екран.
12.	Дайте визначення класу, що зберігає одновимірний масив дійсних чисел. Елементи даних повинні містити кількість елементів масиву, масив дійсних чисел. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних; – пошук заданого елементу у масиві; – визначення кількості від’ємних елементів у масиві; – форматований вивід масиву на екран.
13.	Дайте визначення класу, що зберігає одновимірний масив дійсних чисел. Елементи даних повинні містити кількість елементів масиву, масив дійсних чисел. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних; – визначення кількості додатних елементів у масиві; – визначення добутку елементів у масиві; – форматований вивід масиву на екран.
14.	Дайте визначення класу, що зберігає одновимірний масив цілих чисел. Елементи даних повинні містити кількість елементів масиву, масив цілих чисел. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних; – сортування елементів масиву за збільшенням; – визначення кількості нульових елементів; – форматований вивід масиву на екран.
15.	Дайте визначення класу, що зберігає одновимірний масив цілих чисел. Елементи даних повинні містити кількість елементів масиву, масив цілих чисел. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних; – визначення кількості заданих символів у масиві; – сортування за спаданням; – форматований вивід масиву на екран.

Продовження табл. 21.2

№ вар.	Опис класу
16.	Дайте визначення класу, що зберігає одновимірний масив цілих чисел. Елементи даних повинні містити кількість елементів масиву, масив цілих чисел. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних; – визначення найбільшого спільного дільника елементів масиву; – визначення мінімального елементу масиву; – форматований вивід масиву на екран.
17.	Дайте визначення класу, що зберігає одновимірний масив цілих чисел. Елементи даних повинні містити кількість елементів масиву, масив цілих чисел. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних; – визначення найменшого спільного кратного елементів масиву; – визначення максимального елементу масиву; – форматований вивід масиву на екран.
18.	Дайте визначення класу, що виконує арифметичні операції над двома цілими числами. Елементи даних повинні містити два цілих числа. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних; – визначення суми і різниці чисел; – ділення чисел за модулем; – знаходження заданого відсотка від числа; – форматований вивід результату всіх допустимих операцій над двома цілими числами на екран.
19.	Дайте визначення класу, що виконує арифметичні операції над дробами. Елементи даних повинні містити два дробових числа. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних; – визначення найменшого спільного кратного; – визначення суми, різниці та добутку чисел; – форматований вивід результату всіх допустимих операцій над двома дробовими числами на екран.
20.	Дайте визначення класу, що зберігає рядок символів. Елементи даних повинні містити початковий рядок. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних;

Закінчення табл. 21.2

№ вар.	Опис класу
	– визначення довжини рядка; – заміну всіх символів рядка заданим символом; – вивід рядка на екран.
21.	Дайте визначення класу, що зберігає рядок символів. Елементи даних повинні містити початковий рядок. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних; – знаходження заданого символу в рядку; – вставка символа в кінець рядка, вивід рядка на екран.
22.	Дайте визначення класу, що виконує операції над двома цілими числами. Елементи даних повинні містити два цілих числа. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних; – визначення найбільшого спільного дільника; – визначення найменшого спільного кратного; – ділення чисел; – форматований вивід результату всіх допустимих операцій над двома цілими числами на екран.
23.	Розробіть клас “Студент”, що містить члени для зберігання імені, віку, статі, адреси, року вступу у ВНЗ. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних, зміна даних; – форматований вивід даних про студента; – визначення року народження студента і місця проживання (тобто чи є студент жителем даного міста).
24.	Дайте визначення класу, що являє собою магазин. Клас має атрибути: назва магазину, його адреса, перелік товарів і їх ціна. Функції-члени повинні виконувати такі дії: – створення об’єкта і його ініціалізація; – призначення початкових значень елементам даних, зміна даних; – вивід інформації про заданий товар; – вивід на екран даних про магазин.
25.	Дайте визначення класу, що являє собою каталог фільмів. Клас має атрибути: назва фільму, його жанр, рік випуску. Функції-члени повинні виконувати такі дії: – створення об’єкту і його ініціалізація; – призначення початкових значень елементам даних; – зміна даних; – вивід на екран даних про фільм.

21.3. Створення діалогової прикладної програми для операційної системи Windows

Мета індивідуального завдання

Одержати практичні навики в створенні простої діалогової програми для операційної системи Windows.

Організація виконання індивідуального завдання

Для виконання індивідуального завдання необхідно ознайомитися, використовуючи рекомендовану літературу, конспект лекцій та методичні вказівки, з основними принципами програмування Windows-програм.

Створити проект „діалогове вікно”, при цьому програма повинна мати вигляд, відображений на рис. 21.2.

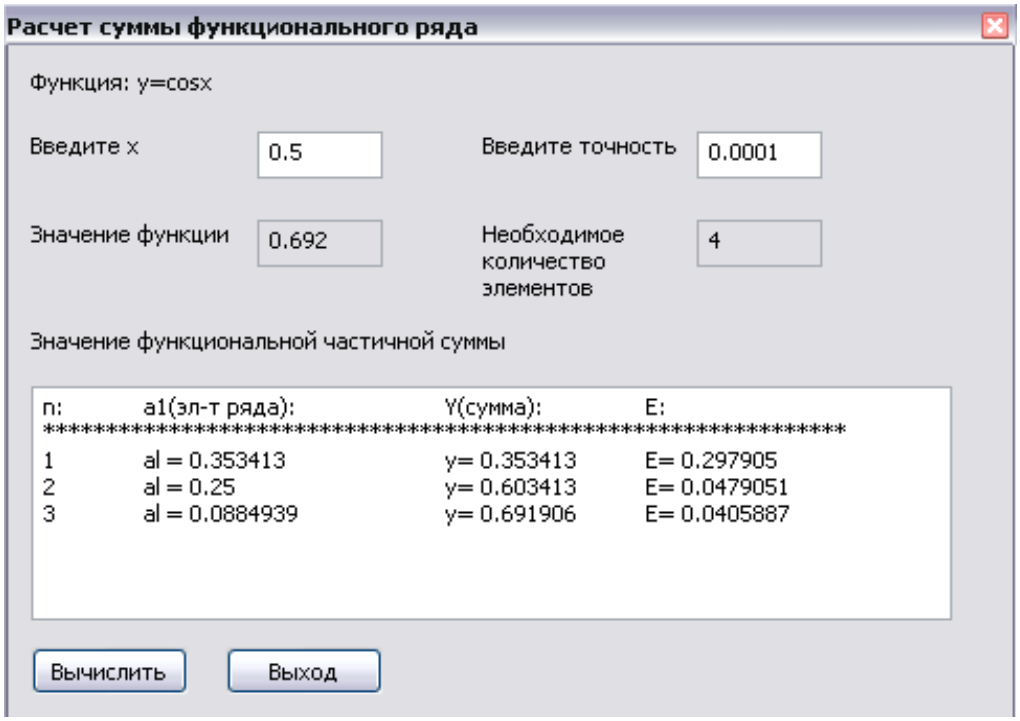


Рис. 21.2. Вигляд програми „діалогове вікно ”

Дослідити структуру проекту.

Програма повинна виконувати такі обчислення:

– значення функціональної часткової суми свого варіанта, виконаної для числа членів *i* від 1 до *n* додатків (табл. 21.3), аргумент при цьому змінюється в діапазоні, визначеному у табл. 21.3;

– значення функції за її формулою (табл. 21.3);

– значення суми ряду *fn(x),* функції *f(x),* що відрізняється від ідеального значення, з необхідною точністю (ε=0.0001);

– визначення кількості потрібних для цього елементів суми *n*, а також вивід результатів у вигляді таблиці, зазначивши кількість підсумовуванних елементів, значення суми і її відхилення від функції.

Оформити звіт про виконане завдання.

Таблиця 21.3

Функціональні ряди

№ ряду	Функціональні ряди	Діапазон зміни аргументу	Кількість підсумовуваних членів ряду (n)	Функція
1.	$\sum_{i=0}^{n} \frac{\ln^i 3}{i!} x^i$	[0,1;1]	10	$y=3^x$
2.	$\sum_{i=1}^{n} \frac{\cos ix}{i}$	[π/5; 9π/5]	16	$y=-\ln \lvert 2\sin(x/2) \rvert$
3.	$\sum_{i=0}^{n} (-1)^{i+1} \frac{\sin ix}{i}$	[π/5; 4π/5)	18	$y=x/2$
4.	$\sum_{i=0}^{n} \frac{x^i}{i!}$	[1;2]	15	$y=e^x$
5.	$\sum_{i=0}^{n} \frac{\cos i\frac{\pi}{4}}{i!} x^i$	[0,1;2]	13	$y=e^{x\cos(\pi/4)} \cos(x\sin(\pi/4))$
6.	$\sum_{i=0}^{n} (-1)^i \frac{x^{2i}}{(2i)!}$	[0,1;2]	10	$y=\cos x$
7.	$\sum_{i=0}^{n} x^i \sin i\frac{\pi}{4}$	[0,1;0,8)	14	$y=\frac{x\sin\frac{\pi}{4}}{1-2x\cos\frac{\pi}{4}+x^2}$
8.	$\sum_{i=0}^{n} \frac{x^{4i+1}}{(2i+1)}$	[-0,3;0,4)	17	$y=\frac{1}{4}\ln\frac{1+x}{1-x}+\frac{1}{2}arctgx$
9.	$\sum_{i=0}^{n} \frac{\cos ix}{i!}$	[-0,3;3]	20	$y=e^{\cos x}\cos(\sin x)$
10.	$\sum_{i=0}^{n} \frac{2i+1}{i!} x^{2i}$	[-0,3;1,3]	10	$y=(1+2x^2)e^{x^2}$
11.	$\sum_{i=0}^{n} \frac{1}{2i+1}\left(\frac{x-1}{x+1}\right)^{2i+}$	[0,2;1]	10	$y=\frac{1}{2}\ln x$
12.	$\sum_{i=0}^{n} (-1)^i \frac{\cos ix}{i^2}$	[π/5; π)	20	$y=\frac{1}{4}(x^2-\frac{\pi^2}{3})$

Продовження табл. 21.3

№ ряду	Функціональні ряди	Діапазон зміни аргументу	Кількість підсумовуваних членів ряду (n)	Функція
13.	$\sum_{i=1}^{n}(-1)^{i+1}\frac{x^{2i+1}}{4i^2-1}$	[0,1;1]	15	$y=\frac{1+x^2}{2}arctgx-\frac{x}{2}$
14.	$\sum_{i=0}^{n}\frac{x^{2i}}{(2i)!}$	[-0,3;1,3]	10	$y=\frac{e^x+e^{-x}}{2}$
15.	$\sum_{i=0}^{n}\frac{(2x)^i}{i!}$	[-0,3;1,2]	13	$y=e^{2x}$
16.	$\sum_{k=1}^{\infty}\frac{\cos kx}{k}$	[3;2π)	11	$y=-\ln(2\sin\frac{x}{2})$
17.	$\sum_{k=1}^{\infty}\frac{\cos kx}{k^2}$	(0;2π)	16	$y=\frac{3x^2-6\pi x+2\pi^2}{12}$
18.	$\sum_{k=1}^{\infty}\frac{\sin kx}{k}$	[2;3,5]	10	$y=\frac{\pi-x}{2}$
19.	$\sum_{k=1}^{\infty}(-1)^{k+1}\frac{\cos kx}{k}$	[0;1]	20	$y=\ln(2\cos\frac{x}{2})$
20.	$\sum_{k=1}^{\infty}(-1)^{k+1}\frac{\sin kx}{k}$	[0;2]	10	$y=\frac{x}{2}$
21.	$\sum_{k=1}^{\infty}(-1)^{k+1}\frac{\cos kx}{k^2}$	[0;2]	12	$y=\frac{\pi^2-3x^2}{12}$
22.	$\sum_{k=1}^{\infty}(-1)^{k+1}\frac{\sin kx}{k^3}$	[0;2]	14	$y=\frac{\pi^2x-x^3}{12}$
23.	$\sum_{k=0}^{\infty}\frac{\cos(2k+1)x}{2k+1}$	[0,5;3]	15	$y=-\frac{1}{2}\ln(tg\frac{x}{2})$
24.	$\sum_{k=0}^{\infty}\frac{\sin(2k+1)x}{2k+1}$	[0,6;3]	10	$y=\frac{\pi}{4}$
25.	$\sum_{k=0}^{\infty}\frac{\cos(2k+1)x}{(2k+1)^2}$	[0;π]	15	$y=\frac{\pi^2-2\pi x}{8}$

Питання для підготовки до захисту індивідуального завдання

1. З яких основних етапів складається розробка програм для Windows і в чому їх суть?
2. У чому суть об’єктно-орієнтованого підходу при програмуванні під Windows?
3. У чому суть керованості Windows-програм? Поясніть поняття повідомлення.
4. Структура і призначення карти повідомлень.
5. Розташування і призначення циклу обробки повідомлень.
6. Порядок створення проекту “діалогове вікно”.
7. Порядок додавання в “діалогове вікно” кнопки і її обробника.
8. У якому місці проекту ініціалізується і запускається об’єкт “діалогове вікно”?

21.4. Обробка текстових файлів

Мета індивідуального завдання

Ознайомитися з методами обробки текстових файлів.

Організація виконання індивідуального завдання

Використовуючи методичні вказівки, ознайомитися з основними методами роботи з файлами.

Скласти схему програми, для вирішення поставленого завдання згідно з варіантом (табл. 21.4).

Скласти програму мовою С++ для вирішення поставленого завдання:

– написати функцію;

– розробити програму, що демонструє роботу функції;

– створити багатофайловий проект. Основну програму, оголошення, а також визначення функцій розмістити в окремих модулях (файлах).

– оформити звіт про виконане завдання.

Питання для підготовки до захисту індивідуального завдання

1. Як зберігається текстова інформація в ЕОМ?
2. Що таке файл?
3. Які види файлів бувають?
4. Яка функція застосовується для відкриття файлу?
5. Яка функція застосовується для читання файлу?
6. Які функції застосовуються для запису, оновлення та закриття файлів?
7. Яка функція може перевіряти результат виконання операцій з файлами?
8. Як створити багатофайловий проект?
9. Як додати заголовний файл до проекту?
10. Які дані можуть міститися в заголовному файлі?

Таблиця 21.4

Варіанти завдань

№ вар.	Опис завдання
1.	Скласти програму «Тестування колективу». Нехай цілочисельна матриця розміром *mxn* містить інформацію про учнів деякого класу з *n* осіб. У першому стовпці проставлена маса (кг), у другому – ріст (см), у третьому – успішність (середній бал) і т.д. (використовуйте свої додаткові показники). Учень називається середньостатистичним за *k*-м параметром (унікальним за *k*-м параметром), якщо на ньому досягається мінімум (максимум) модуля різниці середнього арифметичного чисел з *k*-го стовпця і значення *k*-го параметра цього учня. Учень називається найунікальнішим (середнім), якщо він унікальний (середньостатистичний) з найбільшої кількості параметрів. За даною матрицею визначити найунікальниших і середніх учнів. Інформація про учнів прочитується з початкового файлу, результат роботи програми записується у вихідний файл.
2.	Визначити функцію, яка обчислює довжину гіпотенузи правильного трикутника, коли дві інші сторони відомі. Використовуйте цю функцію в програмі для визначення довжини гіпотенузи n трикутників, n задається з клавіатури. Початкові дані прочитуються з файлу, який виглядає так: 1 3.0 4.0 2 5.0 12.0 3 8.0 15.0 Вихідні дані записуються у файл у такому вигляді: 1 5 2 13 3 17 Програма формує початковий і вихідний файли.
3.	Скласти програму призначення стипендії студентам за результатами сесії, використовуючи такі правила: – якщо всі оцінки 5 – підвищена стипендія; – якщо всі оцінки 4 і 5 – звичайна стипендія; – якщо є оцінка 3, стипендія не призначається. У результаті роботи програми у файлі повинен бути збережений список групи з оцінками і середнім балом кожного студента і два списки прізвищ на підвищену і звичайну стипендії.
4.	Підрахувати кількість рядків заданої цілочисельної матриці *N*x*N* чисел, що є перестановкою 1, 2, ..., *N* (тобто, що містять кожне з чисел 1, 2, ..., *N* тільки один раз). Матриця прочитується з початкового файлу, результат роботи програми записується у вихідний файл. Програма формує початковий і вихідний файли.

Продовження табл. 21.4

<table>
<tr><th>№ вар.</th><th>Опис завдання</th></tr>
<tr><td>5.</td><td>Знайти максимальний елемент серед усіх елементів тих рядків заданої матриці, які впорядковані (або за збільшенням, або за спаданням).
Матриця прочитується з початкового файлу, результат роботи програми записується у вихідний файл.
Програма формує початковий і вихідний файли.</td></tr>
<tr><td>6.</td><td>Характеристикою стовпця цілочисельної матриці назвемо суму модулів його від’ємних непарних елементів. Переставляючи стовпці заданої матриці, розташувати їх відповідно до зростання характеристик.
Матриця прочитується з початкового файлу, результат роботи програми записується у вихідний файл.
Програма формує початковий і вихідний файли.</td></tr>
<tr><td>7.</td><td>Є база даних, що містить відомості про деяку групу людей (кожен запис містить 5 полів). Скласти програму, яка, використовуючи відомості з бази даних, дозволяє заповнювати запрошення, вписуючи ці відомості в потрібні місця у вказаному документі у відповідних відмінках, особах, годинах і т.д.</td></tr>
<tr><td>8.</td><td>Ви є власником складу металевих виробів і потребуєте інвентаризації, після якої ви будете знати, скільки всього різних виробів на складі та яка вартість кожного з них.
Напишіть програму, яка виконувала б такі дії:
– дозволяла вводити дані про кожний виріб;
– давала можливість друкувати список усіх виробів;
– отримувала інформацію про кожний виріб з файлу і могла її змінювати;
– виводила інформацію про вироби у файл;
– підраховувала загальну вартість виробів.

Використовуйте таку інформацію для початку роботи з вашою програмою:
<table>
<tr><th>Номер запису</th><th>Назва інструмента</th><th>Кількість, шт.</th><th>Вартість, грн</th></tr>
<tr><td>3</td><td>Шліфувальний верстат</td><td>7</td><td>57,98</td></tr>
<tr><td>17</td><td>Молоток</td><td>76</td><td>11,99</td></tr>
<tr><td>24</td><td>Механічний лобзик</td><td>21</td><td>11,00</td></tr>
<tr><td>39</td><td>Газонокосарка</td><td>3</td><td>79,50</td></tr>
<tr><td>56</td><td>Електропилка</td><td>18</td><td>99,99</td></tr>
<tr><td>68</td><td>Викрутка</td><td>106</td><td>6,99</td></tr>
<tr><td>77</td><td>Ковальський молот</td><td>11</td><td>21,50</td></tr>
<tr><td>83</td><td>Гайковий ключ</td><td>34</td><td>7,5</td></tr>
</table>
</td></tr>
</table>

Продовження табл. 21.4

№ вар.	Опис завдання
9.	Дані числовий файл. Підрахувати, скільки різних чисел у цьому файлі. Наприклад, у файлі записані числа 8, 7, 8, то різних чисел 2 (8 і 7). Видалити повтори з файлу. Програма формує початковий і вихідний файли.
10.	Дані два символьні файли, перший з яких складається з 10 компонент, другий – з 20. Сформуйте третій файл за такими правилами: спочатку в результуючий файл записується перша компонента з 1-го файлу, після цього – дві компоненти з другого файлу, потім друга компонента з 1-го файла і т.д. Програма формує початкові та вихідний файли.
11.	Дані два числових файли. Визначити кількість загальних елементів і видати їх у результуючий файл. Програма формує початкові і вихідний файли.
12.	У заданому файлі видалити частину тексту, яка поміщена в дужки (разом з дужками). Програма формує початковий і вихідний файли.
13.	Написати функцію, що міняє порядок проходження цілих чисел файлу на протилежний. Якщо файл не існує або містить менше двох елементів, то функція не виконує ніяких дій.
14.	Дане натуральне число. Написати функцію, яка видаляє з числа всі одиниці, залишивши порядок решти цифр незмінним. Натуральне число прочитується з початкового файлу, а результат записується у вихідний файл. Програма формує початковий і вихідний файли.
15.	Здійснити циклічний зсув елементів масиву $T(n)$ на m позицій вліво, тобто одержати масив: t_{m+1}, ., tn t_1, ., tm. При цьому необов'язково, щоб m<n. Матриця прочитується з початкового файлу, результат роботи програми записується у вихідний файл. Програма формує початковий і вихідний файли.
16.	Дана матриця A розміром M x N і числа i, j. Написати функцію, яка видаляє з матриці A рядок і стовпець, що містить елемент $A[i,j]$ (якщо $i > M$ або $j > N$, то матриця не змінюється). Матриця прочитується з початкового файлу, результат роботи програми записується у вихідний файл. Програма формує початковий і вихідний файл.
17.	Написати функцію, що знаходить наближене значення функції $exp(x)$: $exp(x) = 1 + x + x^2 / 2! + x^3 / 3! + ... + xn / n! + ...$. У сумі враховувати всі доданки та великі числа *eps*. За допомогою функції знайти наближене значення експоненти для даного x при n різних значеннях *eps*.

Закінчення табл. 21.4

№ вар.	Опис завдання
	Значення *x, eps* і *n* прочитуються з файлу, а результат роботи програми записується у файл.
18.	Розглянемо деяке натуральне число n. Якщо воно парне, то розділимо його на 2, якщо ні – помножимо на 3 і додамо 1. Повторюватимемо такі дії (кроки) доти, поки не вийде 1. Одержана послідовність називається послідовністю Хейеса, а найбільше з чисел цієї послідовності – її вершиною. Потрібно написати функцію, що обчислює для заданого *n* послідовність Хейеса, яка підраховує кількість кроків у ній і знаходить вершину. Число прочитується з початкового файлу, результат роботи програми записується у вихідний файл.

СПИСОК ЛІТЕРАТУРИ

1. Цвіркун Л.І. Розробка програмного забезпечення комп'ютерних систем. Програмування: навч. посіб. / Л.І. Цвіркун, А.А. Євстігнєєва, Я.В. Панферова; М-во освіти і науки України, Нац. гірн. ун-т. – 2-ге вид., випр. – Дніпропетровськ: НГУ, 2011. – 222 с.
2. Фомин С. В. Системы счисления / С.В. Фомин. – 5-е изд. – Москва: Наука, 1987. – 48 с.
3. Страуструп Б. Язык программирования С++: пер. с англ. / Б. Страуструп. – Санкт-Петербург: БИНОМ – Невский Диалект, 2002. – 1099 с.
4. Семакин И. Г. Основы программирования: учебник / И. Г. Семакин, А. П. Шестаков. – Москва: Мастерство, 2002. – 432 с.
5. Павловская Т. А. С/С++. Программирование на языке высокого уровня / Т. А. Павловская. – Санкт-Петербург: Питер, 2003. – 461 с.
6. Шилдт Г. Самоучитель С++: пер. с англ. / Г. Шилдт. – 3-е изд. – Санкт-Петербург: БХВ-Петербург, 2005. – 688 с.
7. Лафоре Р. Объектно-ориентированное программирование в С++. Классика Computer Science / Р. Лафоре. – 4-е изд. – Санкт-Петербург: Питер, 2005. – 924 с.
8. Культин Н. Б. С/С++ в заданиях и примерах / Н. Б. Культин. – Санкт-Петербург: БХВ-Петербург, 2005. – 288 с.
9. Шпак З.Я. Програмування мовою С / З.Я. Шпак. – Львів: Оріяна-Нова, 2006. – 432 с.

ПЕРЕЛІК СКОРОЧЕНЬ

АКС – автоматизація комп'ютерних систем
ЕОМ – електронно-обчислювальна машина
НГУ – Національний гірничий університет
ОЗП – оперативно-запам'ятовуючий пристрій
ПСЧс – позиційна система числення
СЧс – система числення
ASCII – American Standard Code for Information Interchange
MFC – Microsoft Foundation Class Library

Додаток А
Опис функцій бібліотеки С++

Математичні функції

abs, fabs
Синтаксис:
int *abs* (**int** x) ;
double *fabs* **(double** x) **;**
Повертає ціле (*abs*) або дробове (*fabs*) абсолютне значення аргументу, для якого можна використовувати вираз відповідного типу.
Заголовний файл: <math.h>

acos, asin, atan, acosl, asinl, atanl
Синтаксис:
double *acos* (**double** x) ;
double *asin* (**double** x) ;
double *atan* (**double** x) ;
long double *acosl* (**long double** x) ;
long double *asinl*(**long double** x);
long double *atanl* (**long double** x) ;
Повертає виражену в радіанах величину кута, косинус, синус або тангенс якого переданий відповідній функції як аргумент. Аргумент функції повинен знаходитися в діапазоні від -1 до 1.
Заголовний файл: <math.h>

cos, sin, tan
cosl, sinl, fanl
Синтаксис:
double cos (**double** x) ;
double sin (**double** x) ;
double tan (**double** x) ;
long double cosl (**long double** x) ;
long double sinl (**long double** x) ;
long double tanl (**long double** x) ;
Повертає синус, косинус або тангенс кута. Величина кута повинна бути задана в радіанах.
Заголовний файл: <math.h>

exp, expl
Синтаксис:
double exp(**double** *x);*
long double exp(**long double** (x));
Повертає значення, що дорівнює експоненті аргументу (тобто e^x, де e – основа натурального логарифма).
Заголовний файл: <math.h>

pow, powl
Синтаксис:
double pow (**double** *x,* **double** y);
long double powl(**long double (x)**, **long double** (y));
Повертає значення, що дорівнює x^y.
Заголовний файл: <math.h>

sqrt
Синтаксис:
double sqrt(**double** *x);*
Повертає значення, що дорівнює квадратному кореню з аргументу.
Заголовний файл: <math.h>

rand
Синтаксис:
int rand(**void**);
Повертає випадкове ціле число в діапазоні від 0 до RAND_MAX. Перед першим зверненням до функції rand необхідно ініціалізувати генератор випадкових чисел. Для цього треба викликати функцію srand.

srand
Синтаксис:
void srand (**unsigned** x) ;
Ініціалізує генератор випадкових чисел. Звичайно як параметр функції використовують змінну, значення якої передбачити наперед не можна, наприклад, це може бути поточний час.
Заголовний файл: <stdlib.h>

Функції переведення

Наведені нижче функції виконують переведення рядків у числове значення і чисел у рядкове зображення.

atof
Синтаксис:
double atof (**const char* s) ;**
Повертає дробове число, значення якого передане функції як аргумент. Функція обробляє рядок доти, поки символи рядка є допустимими. Рядок може бути значенням числа як в експоненціальному так і у форматі з плаваючою точкою.
Заголовний файл: <stdlib.h>

atoi, atol
Синтаксис:
int atoi (**const char*** s) ;
long atol (**const char*** *s) ;*
Повертає ціле число відповідного типу, зображення якого віддане функції як аргумент. Функція обробляє символи рядка доти, поки не зустріне символ, що не є десятковою цифрою.
Заголовний файл: <stdlib.h>

gcvt
Синтаксис:
char *gcvt(double *Значення,* int *Цифра* char* *Рядок) ;*
Переводить дробове число в рядок. При переведенні робиться спроба одержати вказану кількість значущих цифр якщо це зробити неможливо, то число зображається у формі плаваючої точки.
Заголовний файл: <stdlib.h>

itoa, Itoa, ultoa
Синтаксис:
char* itoa (**int** *Значення* **char*** *Рядок,* **int** *Основа) ;*
char* Itoa (**long** *Значення* **char*** *Рядок,* **int** *Основа);*
char* ultoa(**unsigned long** *Значення* **char*** *Рядок,* **int** *Основа);*
Відповідно переводять ціле, довге ціле і довге беззнакове ціле число в рядок. Число зображається у вказаному рядку при виклику функції в певній системі числення.
Рядок – покажчик на рядок, куди буде поміщене зображення числа.
Основа – задає основу системи числення (від 2 до 36).
Максимальна довжина рядка, формованого функцією itoa, 17 байт, а функціями Itoa і ultoa – 33 байти.
Заголовний файл: <stdlib.h>

sprintf
Синтаксис:
int sprintf (**char** **Рядок,* **const char*** *Формат, Список Змінних*
Виконує форматований вивід у рядок.
Спісок Змінних – розділені комами імена змінних, задає змінні, значення яких повинні бути виведені. Параметр *Формат* задає спосіб відображення значень змінних.
Дія функції sprintf аналогічна дії функції printf, але вивід виконується в рядок-буфер, а не на екран.
Заголовний файл: <stdio.h>

Функції вводу-виводу

prnff

Синтаксис:

int printf(const char *format [, argument]...);

Виконує форматований вивід на екран.

const char *format – рядок, який буде виведений на екран, argument – розділені комами імена змінних, значення яких підставляються в рядок.

Значення першої змінної виводиться відповідно до першого специфікатора формату, другий – до другого і т.д.

Рядок форматування, окрім тексту, може містити специфікатори формату і керуючі символи.

Специфікатор формату – набір символів, який визначає формат виводу значення змінної, вказаної як аргумент. Виглядає так:

%[flags] [width] [**.**precision] type

flags – символ, параметри вирівнювання, заповнення або формат виводу значення змінної. Він може набувати значення, які наведені в табл. А.1;

Таблиця А.1

Можливі значення символа flags

Символ flags	Значення	Значення за умовчанням
-	Значення вирівнюється зліва по ширині поля виводу	Вирівнювання справа
+	При виведенні числа додається префікс (+ або -), якщо число знакове	Знак виводиться тільки для від'ємних чисел
0	Незайнята область ширини поля виводу значення заповнюється нулями	–
#	Коли використовується разом з нулем **x** або **X,** значення змінної супроводжується відповідним префіксом **0, 0x** або **0X**	–
#	Коли використовується разом з **e**, **E**, **f**, **a** або **A,** значення змінної супроводжується десятковою точкою у будь-якому випадку	Десяткова точка виводиться тільки тоді, коли число дійсно містить дробову частину
#	Коли використовується разом з **g** або **G,** значення змінної супроводжується десятковою точкою у будь-якому випадку і нулями	–

width – десяткове число, задає ширину поля, що виділяється для виводу значення змінної;

precision – точність, з якою виводиться значення. Залежить від типу значення.

Точність може набувати значення, які наведені в табл. А.2.

Таблиця А.2

Можливі значення точності

Тип значення	Значення	Значення за умовчанням
a, A	Кількість знаків після точки	Точність за умовчанням – 6. Коли точність дорівнює 0, точка не виводиться, якщо не використаний знак #
c, C	Точність не враховується	–
d, i, u, o, x, X	Задає мінімальне число знаків. Якщо в числі кількість знаків менша ніж задана точність, до числа зліва додаються нулі	Точність за умовчанням – 1.
e, E, f, g, G	Кількість знаків після коми. Останній знак округляється	Точність за умовчанням – 6. Коли точність дорівнює 0, точка не виводиться, якщо не використаний знак #
s, S	Максимальна кількість символів, що виводяться	Виводиться весь рядок

Type – тип значення, що виводиться
Type може набувати значення, які вказані в табл. А.3.

Таблиця А.3

Можливі значення type

Символ	Тип	Формат виводу
c	int	Символ
C	int	Символ
d	int	Знакове ціле в десятковій системі числення
i	int	Знакове ціле в десятковій системі числення
o	int	Беззнакове ціле у вісімковій системі числення
u	int	Беззнакове ціле в десятковій системі числення
x	int	Беззнакове ціле у шістнадцятковій системі числення. Використовується символ «abcdef»
X	int	Беззнакове ціле у шістнадцятковій системі числення. Використовується символ «ABCDEF»
e	double	Знакове дійсне число у формі [–]d**.**dddd e [*sign*]dd[d]. d – одна десяткова цифра dddd – одна або декілька десяткових цифр dd[d] – 2 або 3 цифри, залежить від формату виводу, розміру експоненти і знаку + або -

Продовження табл. А.3

Символ	Тип	Формат виводу
E	double	Аналогічно e
f	double	Знакове дійсне число у формі [–]dddd.dddd. dddd – одна або декілька десяткових цифр
g	double	Знакове дійсне число. Для виводу використовується формат e або f залежно від довжини числа.
G	double	Аналогічно g
a	double	Знакове дійсне число у шістнадцятковій системі числення у формі [X]0xh.*hhhh* p±*dd*, де *h.hhhh* – шістнадцяткові цифри або символи «abcdef» – мантиса *dd* – 1 або 2 цифри експоненти
A	double	Знакове дійсне число у шістнадцятковій системі числення у формі [X]0Xh.*hhhh* p±*dd*, де *h.hhhh* – шістнадцяткові цифри або символи «ABCDEF» – мантиса *dd* – 1 або 2 цифри експоненти
n	Покажчик на ціле	Виводиться покажчик на ціле
p	Покажчик на void	Виводиться адреса у шістнадцятковому зображенні
s	String	Виводиться рядок символів
S	String	Виводиться рядок символів

Керуючі символи можуть набувати значення, які наведені в табл. А.4.

Таблиця А.4

Можливі значення керуючих символів

Символ	Дія
\n	Переводить курсор на початок наступного рядка
\t	Переводить курсор у чергову позицію табуляції
\\	Символ ‘\’
\'	Лапка

Заголовний файл: <stdio.h>

scanf
Синтаксис:
int scanf(const char *_format_ [, _argument_]...);
Вводить з клавіатури значення змінних відповідно до вказаних специфікатором формату. Перша змінна набуває значення за першим специфікатором формату, друга – за другим і т.д. Параметри:
const char *format – рядок, який містить специфікатори формату (аналогічно printf)
argument – розділені комами адреси змінних, в які будуть записані введені з клавіатури значення.

Спеціальні символи для введення даних можуть набувати значення, які вказані в табл. А.5.

Таблиця А.5

Спеціальні символи для введення даних

Специфікатор	Дія
%i %d	Десяткове число із знаком
%u	Беззнакове ціле десяткове число
%f %e	Дробове число
%s	Рядок символів
%c	Символ

Заголовний файл: <stdio.h>

puts
Синтаксис:
int puts(const char *_str_);
Виводить на екран рядок символів і переводить курсор на початок наступного рядка екрану. Як параметр функції можна використовувати рядкову константу або рядкову змінну.
Заголовний файл: <stdio.h>

gets
Синтаксис:
char *gets(char* s);
Вводить з клавіатури рядок символів. Уведений рядок може містити пропуски.
Заголовний файл: <stdio.h>

putch
Синтаксис:
int putch(int c);
Виводить на екран символ.
Заголовний файл: <conio.h>

getch
Синтаксис:
int getch (void);
Повертає код символу натиснутої клавіші. Якщо натиснута службова клавіша, то функція getch повертає 0. У цьому випадку, для того щоб визначити, яка службова клавіша натиснута, потрібно звернутися до функції getch ще раз.
Зауваження
Функція getch не виводить на екран символ, відповідний натиснутій клавіші.
Заголовний файл: <conio.h>

Функції роботи з файлами

fopen
Синтаксис:
FILE *fopen(const char **filename*, const char **mode*);
Відкриває файл із вказаним ім'ям (filename) для дії, яка задається параметром режиму (mode).
Параметр режиму може набувати значення, які наведені в табл. А.6.

Таблиця А.6

Режими відкриття файлу

Режим	Дія
R	Тільки запис. Файл відкривається тільки для читання
W	Читання. Файл відкривається для запису. Якщо файл із вказанм ім'ям як перший параметр функції fopen вже існує, то нові дані записуються поверх старих, тобто старий файл фактично знищується
A	Додавання. Файл відкривається для запису даних у кінці існуючого файлу. Якщо файл із вказаним ім'ям як перший параметр функції fopen не існує, то він буде створений

Якщо файл відкривається як текстовий, то після символьної константи, що визначає режим відкриття файлу, потрібно додати символ t. Наприклад, рядок rt означає, що для читання відкривається текстовий файл.
У разі успішного відкриття файлу функція fopen повертає покажчик на потік, з якого можна читати або в який можна записувати. Якщо з якої-небудь причини операція відкриття файлу не була виконана, fopen повертає NULL. У цьому випадку, щоб одержати інформацію про причину помилки, слід звернутися до функції ferror.
Заголовний файл: <stdio.h>

fprintf
Синтаксис:
int fprintf(FILE *_stream_, const char *_format_ [, _argument_]...);
Виконує форматований вивід (див. printf) у файл, пов'язаний з потоком, вказаним як перший параметр.
Файл, пов'язаний з потоком, повинен бути відкритий як текстовий у режимі, що допускає запис (див. fopen).
Заголовний файл: <stdio.h>

fscanf
Синтаксис:
int fscanf (FILE *_stream_, const char *_format_ [, _argument_]...);
Виконує форматоване (див. scanf) читання значень змінних з файлу, пов'язаного з потоком, вказаним як перший параметр.
Файл, пов'язаний з потоком, повинен бути відкритий як текстовий у режимі, що допускає читання (див. fopen).
Заголовний файл: <stdio.h>

fgets
Синтаксис:
char *fgets (char *_str_, int _n_, FILE *_stream_);
Параметри:
char *_str_ – рядок, в який записується результат
int _n_ – кількість прочитуваних символів
FILE *_stream_ – файл, з якого прочитується рядок
Читає із вказаного потоку символи і записує їх у рядок. Читання закінчується, якщо прочитаний символ з номером _n-1_ або якщо черговий символ є символом нового рядка.
Прочитаний з файлу символ нового рядка замінюється нульовим символом.
Файл, пов'язаний з потоком, повинен бути відкритий як текстовий у режимі, що допускає читання (див. fopen).
Заголовний файл: <stdio.h>

fgetc
Синтаксис:
int fgetc(FILE *_stream_);
Читає із вказаного потоку символ.
Файл, пов'язаний з потоком, повинен бути відкритий як текстовий у режимі, що допускає читання (див. fopen).
Заголовний файл: <stdio.h>

fputs
Синтаксис:
int fputs (const char *_str_, FILE *_stream_);
Записує у вказаний поток рядок символів. Символ кінця рядка, нуль-символ у поток не записується.
Параметри:
char *_str_ – рядок, який записується у файл
FILE *_stream_ – файл, у який записується рядок
Файл, пов'язаний з потоком, повинен бути відкритий як текстовий у режимі, що допускає запис (див. f open).
Заголовний файл: <stdio.h>

fputc
Синтаксис:
int fputc(int c, FILE *stream);
Записує у вказаний потік ***stream*** символ *c*.
Файл, пов'язаний з потоком, повинен бути відкритий як текстовий у режимі, що допускає запис (див. f open).
Заголовний файл: <stdio.h>

fseek
Синтаксис:
int fseek(FILE *_stream_, long _offset_, int _origin_);
Переміщає файловий покажчик на задану позицію.
Параметри:
FILE *_stream_ – початковий файл
long _offset_ – кількість байт, які необхідно відступити від початкової позиції
int _origin_ – початкова позиція файлового покажчика. Набуває одне з таких значень:
SEEK_CUR – поточна позиція файлового покажчика.
SEEK_END – кінець файлу.
SEEK_SET – початок файлу.
Заголовний файл: <stdio.h>

fsetpos
Синтаксис:
int fsetpos(FILE *_stream_, const fpos_t *_pos_);
Встановлює індикатор позиції у потоці на задане значення *pos*.
Заголовний файл: <stdio.h>

fgetpos
Синтаксис:
int fgetpos(FILE *_stream_, const fpos_t *_pos_);
Записує в змінну *pos* значення індикатора позиції у потоці.
Заголовний файл: <stdio.h>

ferror
Синтаксис:
int ferror (FILE **stream*);
Повертає ненульове значення, якщо остання операція із вказаним потоком завершилася помилкою.
Заголовний файл: <stdio.h>

feof
Синтаксис:
int feof (FILE **stream*);
Повертає ненульове значення, якщо в результаті виконання останньої операції читання з потоку досягнутий кінець файлу.
Заголовний файл: <stdio.h>

fclose
Синтаксис:
int fclose (FILE **stream*);
Закриває вказаний потік.
Заголовний файл: <stdio.h>

Функції роботи з рядками

strcat
Синтаксис:
char *strcat (char **strDestination*, const char **strSource*);
Об'єднує рядки *strDestination* і *strSource* і записує результат у рядок *strDestination.*
Заголовний файл: <string.h>

strncat
Синтаксис:
char *strncat(char **strDest*, const char **strSource*, size_t *count*);
Об'єднує задане число символів *count* рядка *strDestination* і *strSource* і записує результат у рядок *strDestination.*
Заголовний файл: <string.h>

strcpy
Синтаксис:
char *strcpy (char **strDestination*, const char **strSource*);
Копіює рядок *strSource* в рядок *strDestination.*
Заголовний файл: <string.h>

strncpy
Синтаксис:
char *strncpy(char **strDest*, const char **strSource*, size_t *count*);

Копіює задане число символів *count* рядка *strSource* в рядок *strDestination.*
Заголовний файл: <string.h>

strlen
Синтаксис:
int strlen (const char* string);
Повертає довжину рядка. Нульовий символ не враховується.
Заголовний файл: <string.h>

strcmp
Синтаксис:
int strcmp(const char *_string1_, const char * _string2_);
Порівнює рядки *string1* і *string2.* Повертає 0, якщо рядки рівні між собою, число менше нуля, якщо *string1< string2*, і число більше нуля, якщо *string1> string2.*
Заголовний файл: <string.h>

strncmp
Синтаксис:
int strncmp(const char *_string1_, const char *_string2_, size_t _count_);
Порівнює задане число символів *count* рядка *string1* і *string2.* Повертає 0, якщо рядки рівні між собою, число менше нуля, якщо *string1< string2*, і число більше нуля, якщо *string1> string2.*
Заголовний файл: <string.h>

strlwr
Синтаксис:
char* strlwr(char* *string);
Перетворює рядкові символи на великі (обробляє тільки букви латинського алфавіту).
Заголовний файл: <string.h>

strupr
Синтаксис:
char* strupr(char* *string);
Перетворює великі символи на рядкові (обробляє тільки букви латинського алфавіту).
Заголовний файл: <string.h>

strset
Синтаксис:
char * strset(char *_str_, int _sym_);
Заповнює рядок *str* вказаним при виклику функції символом *sym.*
Заголовний файл: <string.h>

strchr
Синтаксис:
char* strchr(const char* *string,* int *sym);*
Виконує пошук символу *sym* в рядку *string* і повертає покажчик на перший знайдений символ або, якщо символ не знайдений, то на null.
Заголовний файл: <string.h>

strstr
Синтаксис:
char *strstr(const char ****str*, const char ****strSearch*);
Параметри:
const char **str* – початковий рядок
const char **strSearch* – рядок, який необхідно знайти
Виконує пошук рядка в рядку і повертає покажчик на перший знайдений рядок або, якщо рядок не знайдений, то на null.
Заголовний файл: <string.h>

strtok
Синтаксис:
char *strtok(char ****strToken*, const char ****strDelimit*);
Ділить рядок на слова.
Параметри:
char **strToken* – початковий рядок
const char **strDelimit* – рядок, що містить символи, які можуть розділяти слова в рядку.
Функція знаходить один із розділювальних символів, замість нього в рядок записує нульовий символ і повертає адресу решти рядка. Якщо весь рядок оброблений, функція повертає NULL. Тобто, щоб розділити рядок на слова, необхідно викликати цю функцію кілька разів, поки повертаним значенням не буде NULL.
Заголовний файл: <string.h>

strrev
Синтаксис:
char *strrev(char ****str*);
Функція проводить інвертування рядка.
Заголовний файл: <string.h>

ПРЕДМЕТНИЙ ПОКАЖЧИК

Навчальне видання

Цвіркун Леонід Іванович
Євстігнєєва Анна Анатоліївна
Панферова Яна Володимирівна

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ. ПРОГРАМУВАННЯ

Навчальний посібник

Видання третє, виправлене

Під загальною редакцією професора Л.І. Цвіркуна

Редактор Ю.В. Рачковська

Підписано до друку 02.03.16. Формат 30х42/4.
Папір офсет. Ризографія. Ум. друк. арк. 12,4.
Обл.-вид. арк. 12,4. Тираж 350 пр. Зам. №

Підготовлено до друку та видрукувано
у Державному ВНЗ "Національний гірничий університет".
Свідоцтво про внесення до Державного реєстру ДК № 1842 від 11.06.2004.
49005, м. Дніпропетровськ, просп. К. Маркса, 19.