МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни
«Програмування»

для здобувачів вищої освіти
першого (бакалаврського) рівня
спеціальності F7 Комп'ютерна інженерія
усіх форм навчання.

Основи програмування мовою C/C++

2025

Методичні вказівки до лабораторних робіт з дисципліни «Програмування» для здобувачів вищої освіти першого (бакалаврського) рівня спеціальності F7 Комп'ютерна інженерія усіх форм навчання. Основи програмування мовою C/C++ / Укл.: О. В. Польська, Р. К. Кудерметов — Запоріжжя: НУ «Запорізька політехніка», 2025 — 55 с.

Укладачі: О. В. Польська, ст. викладач
Р. К. Кудерметов, доцент, к.т.н.

Рецензент: М. Ю. Тягунова, доцент, к.т.н.

Відповідальний
за випуск: О. В. Польська, ст. викладач

Затверджено:
на засіданні кафедри
«Комп'ютерні системи та мережі»
Протокол № 1 від 05 серпня 2025 р.

Рекомендовано до видання:
НМК факультету КНТ
Протокол № 1 від 11 серпня 2025 р.

ЗМІСТ

Вступ **5**

1 Базові типи змінних. Введення-виведення даних **6**
1.1 Основні типи даних 6
1.2 Прості обчислювальні операції 7
1.3 Форматоване введення-виведення даних 10
1.4 Потокове введення-виведення даних 11
1.5 Діапазони значень типів даних 12
1.6 Форматоване виведення рядків 14
1.7 Контрольні питання та завдання до розділу №1 15

2 Оператори вибору **17**
2.1 Умовний оператор `if...else` 17
2.2 Вкладені розгалуження з операторами `if...else` 18
2.3 Розв'язання практичних задач з `if...else` 20
2.4 Оператор-перемикач `switch()` 21
2.5 Контрольні питання та завдання до розділу №2 23

3 Оператори циклу **25**
3.1 Оператор циклу з передумовою `while()` 25
3.2 Оператор циклу `for()` 26
3.3 Вкладений оператор циклу `for()` 27
3.4 Використання `break` та `continue` у циклі з `while()` 29
3.5 Використання оператору циклу `do...while()` 31
3.6 Контрольні питання та завдання до розділу №3 32

4 Масиви і вказівники **34**
4.1 Одновимірні масиви, ініціалізація значень елементів 34
4.2 Одновимірні масиви, генерація випадкових значень 35
4.3 Вказівники. Операція розіменування вказівника 37
4.4 Сортування (впорядкування) елементів масиву 39
4.5 Введення-виведення рядків 40
4.6 Контрольні питання та завдання до розділу №4 42

5 Двовимірні і динамічні масиви. Масиви вказівників **44**
5.1 Обчислення у двовимірному масиві 44
5.2 Способи доступу до елементів двовимірного масиву 46
5.3 Масив вказівників 47
5.4 Масиви динамічної пам'яті (`new/delete`; `malloc()/free()`) . 48
5.5 Двовимірні динамічні масиви 50
5.6 Контрольні питання та завдання до розділу №5 53

Перелік джерел посилання **54**

Додаток А Список лістингів **55**

ВСТУП

Метою даних методичних вказівок є навчити розуміти основи програмування, методи обробки різних даних, принципи побудови алгоритмів та самостійно створювати програми мовами C/C++.

Освоєння матеріалів даних методичних вказівок передбачає значну самостійну роботу студентів, оскільки лише практичне програмування, вирішення конкретних проблем у кожній програмі може дати міцний фундамент знань та навичок у програмуванні.

Значна частина відомостей, викладена в даній роботі, спирається на кращі сучасні та класичні літературні джерела [1—4].

Автори вважають, що матеріали, практичні приклади та завдання, викладені в даних методичних вказівках, під силу студентам, які бажають пов'язати свою діяльність з комп'ютерною інженерією та комп'ютерними науками та сприятимуть набуттю професійних знань та навичок у програмуванні.

Послідовність викладу та рекомендованого вивчення даного матеріалу побудована за принципом «почати з нуля».

У процесі підготовки даних методичних вказівок у авторів, як і у всіх, хто програмує, виникали питання з програмування, проблеми з реалізацією прикладів з різних причин, зокрема, через недосконалість деяких інструментальних засобів та програмних бібліотек. Деякі труднощі можуть виникнути і у вас [5—7].

Для виконання завдань ви можете використовувати інтегровані середовища розробки (IDE, Integrated Development Environment), такі як Dev C++, Visual Studio, Qt Creator, Visual Studio Code, Xcode тощо.

Програми, що представлені у лістингах як навчальні приклади, розроблювались і тестувались у вільному IDE – Dev C++ (адреса ресурсу https://www.embarcadero.com/free-tools/dev-cpp). При роботі у інших IDE може знадобитись незначне коригування коду.

ЛАБОРАТОРНА РОБОТА 1
Базові типи змінних. Введення-виведення даних

Мета роботи – ознайомитися з основними типами даних та операторами мови програмування C/C++, навчитися розроблювати програми, що використовують лінійні алгоритми, опанувати методи введення-виведення даних.

1.1 Основні типи даних

У програмі з головною функцією `main()` (див. лістинг 1.1):

– визначено змінні різних типів даних, таких як `char`, `int`, `float`, які ініціалізовані деякими значеннями (рядки 5, 8, 10);

– реалізовано за допомогою функції `printf()` виведення значень змінних та результатів обчислення їх розмірів у байтах з застосуванням оператору `sizeof` (рядки 6, 7, 9, 11);

– у рядках 6, 7 реалізовано виведення значення символьної змінної `ch`, тобто символу `A`, у символьному вигляді (специфікатор перетворення формату `%c`) і у різних системах числення: `%d` – десяткова, `%o` – вісімкова, `%x` – шістнадцяткова, при цьому визначені значення відповідають кодовій таблиці ASCII (*American Standard Code for Information Interchange*);

– у рядку 11 виведення значення змінної дійсного типу реалізовано у звичайному (`%f`) та в експоненціальному (`%e`) вигляді.

Лістинг 1.1 – Основні типи даних (файл `lab1-1.cpp`)

```
1  #include <stdio.h>
2
3  int main() {
4     puts("My first program");
5     char ch = 'A';
6     printf("Symbol %c = %d (decimal) = %#o (octal) ", ch, ch,
          ch);
7     printf("= %#x (hexadecimal), char size is %d byte\n", ch,
          sizeof(ch));
8     int a = 78;
9     printf("a = %d, int size is %d bytes\n", a, sizeof(a));
10    float b = 526.2345;
11    printf("b = %f = %e, float size is %d bytes\n", b, b,
          sizeof(b));
12    return 0;
13 }
```

Результат роботи програми з лістингу 1.1:

```
My first program
Symbol A = 65 (decimal) = 0101 (octal) = 0x41 (hexadecimal),
    char size is 1 byte
a = 78, int size is 4 bytes
b = 526.234497 = 5.262345e+002, float size is 4 bytes
```

Примітка. Розмір типу даних залежить від компілятора і архітектури комп'ютера, тому результат обчислення може бути іншим.

Завдання до програми (загальне)

Реалізуйте та вивчить програму з лістингу 1.1. Додайте до програми фрагменти коду, що реалізують наступні завдання, та продемонструйте її працездатність із внесеними змінами:

– виведіть на екран своє прізвище;

– символьну змінну `ch` ініціалізуйте першою літерою вашого імені і виведіть її значення у різних системах числення;

– визначте нові змінні цілочисельних типів `short`, `long`, `long long`, ініціалізуйте їх будь-якими значеннями, виведіть на екран їх значення та розмір у байтах;

– визначте нові змінні дійсного типу `double`, `long double`, ініціалізуйте їх будь-якими значеннями, виведіть на екран їх значення у звичайному та експоненціальному вигляді, та їх розмір у байтах.

1.2 Прості обчислювальні операції

Програма (див. лістинг 1.2) виконує прості обчислення з застосуванням унарних, бінарних та тернарного операторів.

Рядки 4—11 демонструють операції з цілочисельними даними.

У рядку 5 – операція привласнення: лівий операнд `a2` отримує значення результату його складання з правим операндом `a1`.

У рядку 7 – операція привласнення суми результатів двох операцій. При цьому результат операції ділення двох цілих чисел буде цілочисельним `a2 / 4 = 11 / 4 = 2`. Результатом бінарної операції `%` буде остача від ділення двох цілих чисел, тобто `a2 % 4 = 11 % 4 = 3`.

У рядку 8 обчислюється результат логічної кон'юнкції `&&` двох операцій відношень: `b1 > a1` та `b1 > a2`. При цьому, як значення цих операцій, так і значення змінної `b2` (рядок 8) може бути або `0`, якщо логічний вираз є хибним (*false*), або `1`, якщо вираз є істиною (*true*).

У рядку 10 – префіксна операція декременту. Спочатку правий операнд `b2` змінює своє значення `--b2 = 0-1 = -1`, а потім лівий операнд `b3` отримує значення правого `b3 = b2 = -1`.

Рядки 12—15 демонструють операції з даними дійсного типу (дробовими числами). У рядку 13 – операція привласнення результату ділення двох операндів, результат – дійсне число.

У рядку 14 – операція привласнення лівому операнду `r2` результату обчислень умовного тернарного оператора, який повертає результат обчислень `f1 + 2`, якщо умова `(f1 < f2)` є істиною, і обчислює `--f2`, якщо результат умови є хибним.

Лістинг 1.2 – Прості обчислення (файл `lab1-2.cpp`)

```
1  #include <stdio.h>
2
3  int main() {
4     int a1 = 9, a2 = 2, b1, b2, b3;
5     a2 += a1; // a2=2+9=11
6     printf("a1=%d a2=%d\n", a1, a2);
7     b1 = a2 / 4 + a2 % 4; // b1=11/4+11%4=2+3=5
8     b2 = b1 > a1 && b1 > a2; // (5>9) and (5>11) ? false =>
          b2=0
9     printf("b1=%d b2=%d\n", b1, b2);
10    b3 = --b2; // b2-1=0-1=-1 => b2=-1, b3=b2=-1
11    printf("b2=%d b3=%d\n", b2, b3);
12    float f1 = 3.6, f2 = 0.5, r1, r2;
13    r1 = f1 / f2; // r1=3.6/0.5=7.2
14    r2 = (f1 < f2) ? f1 + 2 : --f2; // (3.6<0.5) ? false =>
          r2=(0.5-1)=-0.5
15    printf("r1=%5.2f r2=%5.2f\n", r1, r2);
16    return 0;
17 }
```

Результат роботи програми з лістингу 1.2:

```
a1=9 a2=11
b1=5 b2=0
b2=-1 b3=-1
r1= 7.20 r2=-0.50
```

Завдання до програми

Реалізуйте та вивчить програму з лістингу 1.2. Розробіть програму, що виконує задані за варіантом обчислення. Структуру програми бажано не змінювати. Спочатку спробуйте визначити та пояснити результати обчислень за допомогою коментарів, а вже потім запустіть програму і перевірте правильність ваших суджень.

Варіант 1.

```
a1 = 3; a2 = 8;
a2 -= a1;
b1 = a1 + 2 * a2 * a2;
b2 = !a1;
b3 = b2++;
f1 = 1.5; f2 = 1.5;
r1 = f1 * f1 + 2 * f2;
r2 = (f1 != f2) ? f1 - 0.3 : f2 + 0.4;
```

Варіант 2.

```
a1 = 2; a2 = 3;
a2 *= 10;
b1 = a1 * a1 * a1 + a2;
b2 = a1 > a2;
b3 = ++b2;
f1 = 1.5; f2 = 1.5;
r1 = 2 * f1 - f2;
r2 = (f1 == f2) ? f1 * 3 : f2 * 5;
```

Варіант 3.

```
a1 = 6; a2 = 20;
a2 /= 5;
b1 = 5 * a1 + a2 * a2;
b2 = a1 > a2;
b3 = --b2;
f1 = 1.5; f2 = 3.5;
r1 = f1 + 4 * f2;
r2 = (f1 >= f2) ? f1 - 2.5 : f2 + 5.2;
```

Варіант 4.

```
a1 = 3; a2 = 2;
a2 += a1 + 5;
b1 = a1 * a1 * a1 + 3 * a2;
b2 = a1 < a2;
b3 = b2--;
f1 = 1.5; f2 = 2.5;
r1 = f1 / f2 * 2;
r2 = (f1 <= f2) ? f1 - 0.2 : f2 + 0.7;
```

Варіант 5.

```
a1 = 5; a2 = 13;
a2 -= a1 + 5;
b1 = a1 + a2 * a2;
b2 = a1 > a2;
b3 = b2++;
f1 = 4.5; f2 = 1.8;
r1 = f1 + f2 / 2;
r2 = (f1 > f2) ? f1 + 1.2 : f2 - 1.4;
```

1.3 Форматоване введення-виведення даних

Програма (див. лістинг 1.3) обчислює об'єм циліндра за введеними з клавіатури (функція `scanf()`) значеннями радіусу та висоти. Введенні та обчислені значення виводяться на екран (функція `printf()`).

Лістинг 1.3 – Функції `scanf()` та `printf()` (файл `lab1-3.cpp`)

```
#include <stdio.h>

int main() {
  float h, r, V;
  puts("Enter values r and h:");
  scanf("%f %f", &r, &h);
  V = 3.141592f * r * r * h;
  printf("r =%6.2f cm, h =%6.2f cm\n", r, h);
  printf("V =%8.3f cm^3\n", V);
  return 0;
}
```

Результат роботи програми з лістингу 1.3:

```
Enter values r and h:
7.5
8.3
r =  7.50 cm, h =  8.30 cm
V =1466.731 cm^3
```

Примітка. Перше число у специфікаторі формату для виведення дійсних чисел – загальна кількість позицій для виведення числа (з урахуванням крапки), друге – кількість цифр дробової частини числа.

Наприклад: при параметрах формату `%7.2f` на екран виведеться результат з чотирма цифрами цілої частини і двома – дробової.

Завдання до програми

Реалізуйте та вивчить програму з лістингу 1.3. Розробіть програму, що виконує задані обчислення та виводить результат у заданому форматі (функція `printf()`). Визначте необхідні змінні. Значення змінних для обчислень введіть з клавіатури (функція `scanf()`). Для операції піднесення до степеня можна використовувати функцію `pow()` (файл `math.h`).

Варіант 1. Обчисліть вираз: $res = (4.2 \cdot x - z)^2/3.5$. Тип даних – `float`. Формат виведення: три цифри цілої частини, дві – дробової.

Варіант 2. Обчисліть вираз: $res = a + 2.8 \cdot b - c^2$. Тип даних – `double`. Формат виведення: дві цифри цілої частини, три – дробової.

Варіант 3. Обчисліть вираз: $res = k^2 + (g - k)/2.3$. Тип даних – `double`. Формат виведення: чотири цифри цілої частини, три – дробової.

Варіант 4. Обчисліть вираз: $res = s^3/v - f \cdot v$. Тип даних – `float`. Формат виведення: дві цифри цілої частини, чотири – дробової.

Варіант 5. Обчисліть вираз: $res = x^3 + z/h^2$. Тип даних – `double`. Формат виведення: три цифри цілої частини, три – дробової.

1.4 Потокове введення-виведення даних

Програма (див. лістинг 1.4) обчислює об'єм кулі за введеним із клавіатури (`cin>>`) значенням радіусу. Введенні та обчислені значення виводяться на екран (`cout<<`). Для об'єктів `cin` та `cout` з бібліотеки `iostream` (рядок 1) вказано використовуваний простір імен `std` на початку програми (рядок 4). Для керування форматом введення/виведення підключено бібліотеку `iomanip` (рядок 3).

Лістинг 1.4 – Введення/виведення з `cin` та `cout` (файл `lab1-4.cpp`)

```
1  #include <iostream>
2  #include <math.h>
3  #include <iomanip>
4  using namespace std;
5
6  int main() {
7     double r, V;
8     cout << "Enter radius:\n";
9     cin >> r;
10    V = 3.0 / 4.0 * M_PI * r * r * r;
11    cout << fixed << setprecision(4);
12    cout << "r = " << r << " cm\n";
13    cout << "V = " << setw(10) << V << " cm^3\n";
14    return 0;
15 }
```

Результат роботи програми з лістингу 1.4:

```
Enter radius:
5.5
r = 5.5000 cm
V =   392.0119 cm^3
```

Примітка. Маніпулятор `setw(10)` (рядок 13) задає загальну ширину поля для виведення числа, у даному випадку – 10. У рядку 11 використовується `fixed` разом з `setprecision(4)` для визначення точної кількості цифр дробової частини дійсного числа, у даному випадку – 4.

Завдання до програми

Реалізуйте та вивчить програму з лістингу 1.4. Розробіть програму, що виконує задані обчислення та виводить результат у заданому форматі (`cout<<` та маніпулятори). Визначте необхідні змінні. Значення змінних для обчислень введіть з клавіатури (`cin>>`).

Варіант 1. Обчисліть площу ромбу: $S = 1/2 \cdot d_1 \cdot d_2$. Формат виведення: три цифри цілої частини, дві – дробової.

Варіант 2. Обчисліть площу куба: $S = 6 \cdot a^2$. Формат виведення: дві цифри цілої частини, три – дробової.

Варіант 3. Обчисліть довжину кола: $L = 2 \cdot \pi \cdot r$. Формат виведення: чотири цифри цілої частини, три – дробової.

Варіант 4. Обчисліть площу прямокутника: $S = a \cdot b$. Формат виведення: дві цифри цілої частини, чотири – дробової.

Варіант 5. Обчисліть площу кола: $S = \pi \cdot r^2$. Формат виведення: три цифри цілої частини, три – дробової.

1.5 Діапазони значень типів даних

Програма (див. лістинг 1.5) демонструє, що при виконанні арифметичних дій над даними може відбуватись *переповнення* (*overflow*) і втрата бітів інформації. Отже для запобігання отримання хибних результатів обчислень потрібно враховувати діапазон значень використовуваного типу даних.

Лістинг 1.5 – Діапазон значень типів даних (файл `lab1-5.cpp`)

```
#include <stdio.h>

int main() {
    short a1 = 32766, a2, a3;
    char b1 = 126, b2, b3;
    int c1 = 2147483646, c2, c3;
    a2 = a1 + 1;
    a3 = a1 + 2;
    printf("short value: %d %d %d\n", a1, a2, a3);
    b2 = b1 + 1;
    b3 = b1 + 2;
    printf("char value: %d %d %d\n", b1, b2, b3);
    c2 = c1 + 1;
    c3 = c1 + 2;
    printf("int value: %d %d %d\n\n", c1, c2, c3);
    return 0;
}
```

Результат роботи програми з лістингу 1.5:

```
short value: 32766 32767 -32768
char value: 126 127 -128
int value: 2147483646 2147483647 -2147483648
```

Примітка. Діапазон значень типу даних залежить від кількості байтів, яку даний тип займає у пам'яті комп'ютера.

Завдання до програми

Реалізуйте та вивчить програму з лістингу 1.5. Розробіть програму, що демонструє переповнення у випадку використання числа, більше максимального або менше мінімального з діапазону допустимих значень зазначених типів даних. Для цього визначте та ініціалізуйте змінні заданими значеннями, виконайте необхідні обчислення та виведіть всі значення на екран.

Варіант 1. Додайте до значень змінних такі числа, щоб нові значення стали максимальними (`x2, y2, z2, v2`) та більшими за максимальні значення кожного з типів на одиницю (`x3, y3, z3, v3`):

```
short x1 = 32763;
unsigned short y1 = 65531;
char z1 = 122;
unsigned char v1 = 2;
```

Варіант 2. Відніміть від значень змінних такі числа, щоб нові значення стали мінімальними (`x2, y2, z2, v2`) та меншими за мінімальні значення кожного з типів на одиницю (`x3, y3, z3, v3`):

```
short x1 = -32763;
unsigned short y1 = 31;
char z1 = -120;
unsigned char v1 = 20;
```

Варіант 3. Додайте до значень змінних такі числа, щоб нові значення стали максимальними (`x2, y2, z2, v2`) та більшими за максимальні значення кожного з типів на одиницю (`x3, y3, z3, v3`):

```
short x1 = 30;
unsigned short y1 = 35000;
char z1 = -25;
unsigned char v1 = 20;
```

Варіант 4. Відніміть від значень змінних такі числа, щоб нові значення стали мінімальними (`x2, y2, z2, v2`) та меншими за мінімальні значення кожного з типів на одиницю (`x3, y3, z3, v3`):

```
short x1 = -32000;
unsigned short y1 = 65530;
char z1 = -108;
unsigned char v1 = 240;
```

Варіант 5. Додайте до значень змінних такі числа, щоб нові значення стали максимальними (`x2, y2, z2, v2`) та більшими за максимальні значення кожного з типів на одиницю (`x3, y3, z3, v3`):

```
int x1 = 2147483645;
unsigned int y1 = 4294967290;
char z1 = 125;
unsigned char v1 = 252;
```

1.6 Форматоване виведення рядків

Програма (див. лістинг 1.6) виводить на екран рядки (функція `printf()`), що визначені за допомогою директиви `#define`.

Лістинг 1.6 – Виведення рядків з `printf()` (файл `lab1-6.cpp`)

```
 1  #include <stdio.h>
 2  #define STR1 "Today we are "
 3  #define STR2 "studying the course "
 4  #define STR3 "\"Programming in C\""
 5
 6  int main() {
 7      printf("|%s|\n", STR1);
 8      printf("|%s|\n", STR2);
 9      printf("|%s|\n", STR3);
10      printf("|%75.5s|\n", STR1);
11      printf("|%-75.8s|\n", STR2);
12      printf("|%75.12s|\n", STR3);
13      printf("|%-75.8s|\n", STR3);
14      return 0;
15  }
```

Результат роботи програми з лістингу 1.6:

```
|Today we are |
|studying the course |
|"Programming in C"|
|                                                                    Today|
|studying                                                               |
|                                                          "Programming"|
|"Program                                                               |
```

Примітка. Для наочності маніпулювання положенням на екрані, специфікатор для виведення рядку розміщено в обмежувальних позначках |`%s`|. Перше число у специфікаторі формату для виведення рядку – загальна кількість позицій для виведення літер або символів, друге – кількість літер, що виводиться, яка рахується з початку рядкової константи. Знак мінус забезпечує зсув рядку до лівої межі поля.

Завдання до програми

Реалізуйте та вивчить програму з лістингу 1.6. Додайте до програми рядок `STR4` з заданими словами за допомогою директиви `#define`. Забезпечте виведення заданого рядку (з межами на початку та кінці екрану) повністю, потім на новому рядку тільки перше слово на початку рядка екрана і на наступному рядку перше слово у кінці рядка екрана.

Варіант 1. Ім'я та по батькові студента.

Варіант 2. Прізвище та ім'я студента.

Варіант 3. Назва навчальної дисципліни та прізвище викладача.

Варіант 4. Прізвище автора та назва книги.

Варіант 5. Назва міста та вулиці.

1.7 Контрольні питання та завдання до розділу №1

1. Які типи даних використовуються в мові C/C++ та скільки байт займає кожен з них? Наведіть код програми з визначення розмірів типів даних та результати її роботи у вашому середовищі розробки.

2. Перерахуйте та поясніть на прикладах знаки операцій (унарні, бінарні, тернарний), які були використані для обчислень у завданні до програми з лістингу 1.2 за вашим варіантом.

3. Який загальний склад аргументів функції `scanf()`? Наведіть приклад використання функції з завдання до програми з лістингу 1.3 за вашим варіантом.

4. Який загальний склад аргументів функції `printf()`? Наведіть приклад використання функції з завдання до програми з лістингу 1.3 за вашим варіантом.

5. Наведіть приклад використання операцій потокового введення-виведення (`cin>>` та `cout<<`) на основі коду з завдання до програми з лістингу 1.1.

Для цього замініть відповідні функції форматованого виведення, а значення змінних введіть з клавіатури замість їхньої ініціалізації. Результат роботи програми повинен залишитись незмінним.

6. Запишіть діапазони значень типів даних, отриманих при виконанні у вашому середовищі розробки завдання до програми з лістингу 1.5 за вашим варіантом.

7. Знайдіть помилки у рядках коду з лістингу 1.7, запишіть код, коректний з точки зору синтаксису C/C++, і поясніть виправлення. Наведіть результати роботи виправленої програми. У разі покращення коду, окремо поясніть таку необхідність.

Лістинг 1.7 – Валідація коду

```
1  #include stdio.h
2
3  void main; {
4     puts("%235 #5\n");
5     char ch = 'AXA';
6     printf("Symbol %c = %d", ch, ch, ch);
7     int i
8     i:=52
9     print('There are i weeks in the year')
10    int a;
11    scanf("%ld", a);
12    printf("a = %d, i = %d, int size is %d bytes\n", a,
          sizeof(int));
13    float b = 0.0005;
14    printf("b = %f, float size is %f bytes\n", b, sizeof(b));
15    return 0;
16 }
```

ЛАБОРАТОРНА РОБОТА 2
Оператори вибору

Мета роботи – ознайомитися з операторами умовного та множинного вибору мови програмування C/C++, навчитися розроблювати та тестувати найпростіші програми, що використовують алгоритми з розгалуженням.

2.1 Умовний оператор `if...else`

Програма (див. лістинг 2.1) перевіряє, чи є введене з клавіатури ціле число парним, і виводить відповідні повідомлення. Якщо результат виразу, тобто *умова*, `(n % 2 == 0)` у рядку 7 є істиною (`true`), то виводиться повідомлення, яке розміщене у рядку 8. У іншому випадку, результат виразу в умові є хибним (`false`) і виводиться повідомлення, яке розміщене у рядку 10.

Для тестування програм, що використовують алгоритми з розгалуженням, потрібно підібрати такі вхідні параметри, щоб перевірити коректність роботи усіх гілок розгалуження. Необхідну кількість тестових випадків потрібно визначити відповідно до очікувань розробника та вимог замовника програми.

Лістинг 2.1 – Умовний оператор `if...else` (файл `lab2-1.cpp`)

```
1  #include <stdio.h>
2
3  int main() {
4      int n;
5      puts("Enter an integer:");
6      scanf("%d", &n);
7      if (n % 2 == 0)
8          printf("The number %d is even\n", n);
9      else
10         printf("The number %d is odd\n", n);
11     return 0;
12 }
```

Результат роботи програми з лістингу 2.1 при тестуванні випадку, коли умова у рядку 7 є істиною, може бути наступним:

```
Enter an integer:
12578
The number 12578 is even
```

Примітка. Умову у рядку 7 реалізовано за допомогою операції `%` (остача від ділення), результат якої порівнюється (операція `==`) з нулем. Очевидно, що `4 % 2 = 0`, а `5 % 2 = 1`. Для даної задачі умова може бути такою: `(n % 2 != 1)`.

Варто зауважити, що результат обчислення виразу в умові може мати будь-яке числове значення. Умова вважається хибною тільки у випадку, коли результат обчислення виразу дорівнює нулю.

Завдання до програми

Реалізуйте та вивчить програму з лістингу 2.1. Розробіть програму, що перевіряє введене з клавіатури ціле число на відповідність заданій умові та виводить на екран відповідні повідомлення. Проведіть тестування гілок розгалуження.

Варіант 1. Перевірте, чи число більше за `100`.

Варіант 2. Перевірте, чи число менше ніж `20`.

Варіант 3. Перевірте, чи закінчується число на цифру `3`.

Варіант 4. Перевірте, чи закінчується число на цифру `0`.

Варіант 5. Перевірте, чи ділиться число на `5` без остачі.

2.2 Вкладені розгалуження з операторами `if...else`

Програма (див. лістинг 2.2) перевіряє декілька окремих умов: чи є введене з клавіатури число додатним та більшим чи меншим за задані числа.

Задачу можна сформулювати наступним чином: якщо число додатне, перевірити, чи число більше за `100`, у іншому випадку (число не додатне) перевірити, чи число менше ніж мінус `100`. Кожного разу виводиться одне загальне повідомлення щодо виконання або невиконання відповідних умов.

Для тестування програми з лістингу 2.2 доцільно перевірити коректність виведення повідомлень з усіх гілок розгалуження (рядки 10, 12, 14, 16).

Для цього потрібно запустити програму на виконання якнайменше чотири рази. Тестові випадки краще підбирати таким чином, щоб послідовно перевірити виведення усіх повідомлень. Наприклад, для перевірки виведення повідомлення з рядку 10 потрібно ввести будь-яке додатне число, що більше за 100, а для повідомлення з рядку 12 будь-яке додатне число, що не більше 100, і т.д.

Лістинг 2.2 – Вкладені оператори `if...else` (файл `lab2-2.cpp`)

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      float x;
6      cout << "Enter a number:" << endl;
7      cin >> x;
8      if (x > 0.0)
9          if (x > 100.0)
10             cout << x << " is positive and greater than
                   100.0" << endl;
11         else
12             cout << x << " is positive and not more than
                   100.0" << endl;
13     else if (x < -100.0)
14         cout << x << " is not positive and less than -100.0"
               << endl;
15     else
16         cout << x << " is not positive and not less -100.0"
               << endl;
17     return 0;
18 }
```

Результат роботи програми з лістингу 2.2 може бути наступним:

```
Enter a number:
-105.99
-105.99 is not positive and less than -100.0
```

Примітка. Оператор `else` (рядок 11) відповідає найближчому попередньому оператору `if` (рядок 9). Тоді як оператор `else` у рядку 13 відповідає оператору `if` у рядку 8.

Завдання до програми

Реалізуйте та вивчить програму з лістингу 2.2. Розробіть програму, що перевіряє введені з клавіатури дані на відповідність заданим умовам та виводить на екран чотири відповідні повідомлення. У варіантах з двома числами, для запобігання дублювання повідомлень, рекомендується розрізняти числа за порядком введення (перше число, друге число). Для реалізації алгоритму використовуйте вкладений умовний оператор `if...else`. Проведіть тестування гілок розгалуження.

Варіант 1. Якщо ціле число є парним, то чи є воно додатним; якщо непарне, то чи більше за `25`.

Варіант 2. Для більшого з двох дійсних чисел перевірити, чи знаходиться його значення у діапазоні від `5.0` до `30.0`.

Варіант 3. Якщо ціле число більше за 50, то чи кратне воно числу 4; якщо не більше, то чи кратне числу 3.

Варіант 4. Для меншого з двох цілих чисел перевірити, чи є воно непарним.

Варіант 5. Якщо ціле число є додатним, то чи кратне воно числу 5; якщо не додатне, то чи більше за мінус 75.

2.3 Розв'язання практичних задач з `if...else`

Програма (див. лістинг 2.3) визначає найбільше серед трьох введених з клавіатури чисел. У даному прикладі показано реалізацію одного з можливих алгоритмів.

Для тестування коректності роботи даного прикладу потрібно запустити програму на виконання якнайменше три рази. Наприклад, при введенні будь-яких трьох значень, при першому запуску першим ввести найбільше число серед введених, при другому запуску, щоб воно було другим, при третьому – третім.

Лістинг 2.3 – Пошук максимального числа (файл `lab2-3.cpp`)

```
#include <iostream>
using namespace std;

int main() {
    int a, b, c, max;
    cout << "Enter values of a, b, c:\n";
    cin >> a >> b >> c;
    if (a > b && a > c)
        max = a;
    else if (b > c)
        max = b;
    else
        max = c;
    cout << "Max value is " << max;
    return 0;
}
```

Результат роботи програми з лістингу 2.3 може бути наступним:

```
Enter values of a, b, c:
123
-4567
879
Max value is 879
```

Завдання до програми

Реалізуйте та вивчить програму з лістингу 2.3. Розробіть програму, яка виконує задані обчислення та виводить на екран відповідні повідомлення. Значення змінних повинні вводитись з клавіатури. Проведіть тестування гілок розгалуження.

Варіант 1. Визначте найменше серед чотирьох дійсних чисел і перевірте додатне воно чи ні.

Варіант 2. Визначте найбільше серед двох цілих чисел і перевірте у якому з наведених діапазонів знаходиться це число: `менше 0`, `від 0 до 50`, `від 51 до 100`, `більше 100`.

Варіант 3. Визначте найменше серед двох цілих чисел і перевірте у якому з наведених діапазонів знаходиться це число: `менше -100`, `від -99 до 0`, `від 1 до 100`, `більше 100`.

Варіант 4. Визначте найбільше серед чотирьох дійсних чисел і перевірте чи більше це число за `500`, чи ні.

Варіант 5. Визначте найменше серед трьох цілих чисел і перевірте парне воно чи ні.

2.4 Оператор-перемикач `switch()`

Замість написання великої кількості операторів `if...else` для перевірки однієї змінної, можна використовувати оператор-перемикач `switch`, який вибирає один з багатьох блоків коду для виконання. Вираз `switch` обчислюється один раз і порівнюється зі значеннями міток `case`. Вираз, що передається у `switch` повинен мати цілочисельний тип.

Програма (див. лістинг 2.4) обчислює і виводить на друк результат виконання арифметичної операції над двома дійсними числами, що вводяться з клавіатури, у залежності від введеного знаку операції (віднімання, складання, множення, ділення).

Знак арифметичної операції як символ передається у `switch()`, у тілі якого цей символ послідовно порівнюється з константами міток `case`.

Якщо знайдеться відповідне значення константи, то виконаються оператори даної мітки.

Якщо значення не буде знайдено, то виконаються оператори мітки `default`.

Оператор `break` забезпечує вихід з оператора `switch()` і ігнорує виконання решти коду. У разі відсутності `break` будуть виконані оператори усіх наступних міток.

Лістинг 2.4 – Оператор-перемикач `switch()` (файл `lab2-4.cpp`)

```
1  #include <stdio.h>
2
3  int main() {
4      float a, b;
5      char s;
6      puts("Enter values of a, b:");
7      scanf("%f %f", &a, &b);
8      fflush(stdin);
9      puts("Enter the operation sign (-, +, *, /):");
10     s = getchar();
11     switch (s) {
12     case '-':
13         printf("%.2f - %.2f = %.2f\n", a, b, a - b);
14         break;
15     case '+':
16         printf("%.2f + %.2f = %.2f\n", a, b, a + b);
17         break;
18     case '*':
19         printf("%.2f * %.2f = %.2f\n", a, b, a * b);
20         break;
21     case '/':
22         printf("%.2f / %.2f = %.2f\n", a, b, a / b);
23         break;
24     default:
25         printf("Error! Only sings: -, +, *, /.\n");
26     }
27     return 0;
28 }
```

Результат роботи програми з лістингу 2.4 може бути наступним:

```
Enter values of a, b:
5.12
7.35
Enter the operation sign (-, +, *, /):
+
5.12 + 7.35 = 12.47
```

Примітка. У даній програмі використано `fflush(stdin)` (рядок 8) після `scanf()` для очищення потоку введення перед роботою `getchar()`.

Завдання до програми

Реалізуйте та вивчить програму з лістингу 2.4. Розробіть програму, що виводить на екран повідомлення в залежності від введеного з клавіатури значення змінної. Для реалізації завдання використовуйте оператор-перемикач `switch()`, мітки `case` та `default`. Проведіть тестування можливих результатів виконання коду.

Розробіть програму, що виводить на екран повну назву деякого об'єкта за введеною з клавіатури першою літерою (великою або малою), або повідомлення про помилку, якщо введений символ не відповідає заданим.

Варіант 1. При введені номеру місяця року виводиться одне з повідомлень: «Весна», «Літо», «Осінь», «Зима».

Варіант 2. При введені першої літери слова (великої або малої) виводиться назва однієї з семи будь-яких столиць Європи.

Варіант 3. При введені номеру дня тижня виводиться одне з повідомлень: «Робочий день», «Субота», «Неділя».

Варіант 4. При введені першої літери слова (великої або малої) виводиться назва однієї з п'яти будь-яких країн Азії.

Варіант 5. При введені номеру місяця року виводиться одне з повідомлень: «Осінній семестр», «Весняний семестр», «Канікули».

2.5 Контрольні питання та завдання до розділу №2

1. Які оператори вибору використовуються у мові C/C++? Яке їх призначення.

2. Які значення може прийняти вираз умови в операторі `if(умова)`? Яке значення є істиною, а яке хибністю?

3. Наведіть блок-схему алгоритму із завдання до програми з лістингу 2.2 за вашим варіантом.

4. З'ясуйте, які повідомлення виведуться в результаті роботи фрагменту коду, та поясніть отримані результати.

```
1     int x = 5, y = 6, z = 10;
2     if (x == 3)
3        puts("AA1");
4     else
5        puts("BB2");
6     if (x = 10, x < y)
7        puts("CC3");
8     else
9        puts("DD4");
10    if (z < y, z = -4)
11       puts("EE5");
12    else
13       puts("FF6");
```

5. Визначте результати роботи логічної конструкції з нижченаведеного фрагменту коду для змінних `x` і `y` з усіма можливими комбінаціями значень `0` та `1`:
(a) `x = 0, y = 0`; (b) `x = 0, y = 1`; (c) `x = 1, y = 0`; (d) `x = 1, y = 1`.

```
1     int x, y;
2     cin >> x >> y;
3     if (x)
4        if (y)
5           cout << "mother";
6        else
7           cout << "key";
8     cout << "board";
```

З'ясуйте, що зміниться, і наведіть результати роботи такого фрагменту з усіма можливими комбінаціями значень для випадку, коли у фігурні дужки узяті 4 та 5 рядки коду наступним чином:

```
  {if (y)
           cout << "mother";}
```

Наведіть блок-схеми алгоритмів для обох варіантів.

6. Які значення може приймати `вираз` у операторі `switch(вираз)`? Наведіть приклади.

7. Яке призначення мітки `default` у операторі множинного вибору `switch()`? Чи завжди вона потрібна? Що станеться при її вилученні?

8. Яке призначення оператора `break` у операторі множинного вибору `switch()`? Що станеться при його вилученні? У якому випадку він непотрібний?

9. Наведіть код із завдання до програми з лістингу 2.3 за вашим варіантом. Запишіть варіанти вхідних значень змінних для тестування усіх гілок розгалуження коду і результати роботи цієї програми.

ЛАБОРАТОРНА РОБОТА 3
Оператори циклу

Мета роботи – ознайомитися з засобами організації циклічного обчислювального процесу у мові програмування C/C++, навчитися розроблювати програми, що використовують циклічні алгоритми, з застосуванням операторів `while()`, `do...while()`, `for()` та здійснювати контроль вхідних даних при наявності обмежень на їхні значення.

3.1 Оператор циклу з передумовою `while()`

Програма (див. лістинг 3.1) обчислює суму ряду цілих парних чисел, доки сума не стане більше 20.

Лістинг 3.1 – Оператор циклу `while()` (файл `lab3-1.cpp`)

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int a = 0, sum = 0, n = 0;
6      while (sum < 20) {
7          n++;
8          a += 2;
9          sum += a;
10         cout << "n = " << n << "\ta = " << a << "\tsum = " <<
               sum << '\n';
11     }
12     cout << "Number of loops is " << n;
13     return 0;
14 }
```

Результат роботи програми з лістингу 3.1:

```
n = 1    a = 2    sum = 2
n = 2    a = 4    sum = 6
n = 3    a = 6    sum = 12
n = 4    a = 8    sum = 20
Number of loops is 4
```

Завдання до програми

Реалізуйте та вивчить програму з лістингу 3.1. Розробіть програму, що виконує задані обчислення, виводить на екран відповідні повідомлення та результати обчислень.

Варіант 1. Обчисліть суму ряду цілих чисел, що кратні 3, доки останнє число не стане більше за 21.

Варіант 2. Обчисліть суму цілих чисел, що вводяться з клавіатури, доки сума не стане більше за 100.

Варіант 3. Обчисліть суму ряду цілих чисел, що кратні 4, доки останнє число не стане більше за 36.

Варіант 4. Обчисліть суму цілих чисел, що вводяться з клавіатури, доки сума не стане більше за 50.

Варіант 5. Обчисліть суму ряду цілих чисел, що кратні 5. Кількість циклів вводиться з клавіатури.

3.2 Оператор циклу `for()`

Програма (див. лістинг 3.2) обчислює квадрати ряду чисел зазначеної кількості, що кратні 5, або доки число не стане більшим за 100.

Лістинг 3.2 – Оператор циклу `for()` (файл `lab3-2.cpp`)

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int i, number, a = 0, square;
6      cout << "Enter number of loops\n";
7      cin >> number;
8      for (i = 0; i < number; i++) {
9          a += 5;
10         if (a > 100) {
11             cout << "a > 100" << endl;
12             break;
13         }
14         square = a * a;
15         cout << "a = " << a << "\tsquare = " << square <<
               '\n';
16     }
17     cout << "Number of loops is " << i;
18     return 0;
19 }
```

Результат роботи програми з лістингу 3.2 може бути наступним:

```
Enter number of loops
3
a = 5    square = 25
a = 10   square = 100
a = 15   square = 225
Number of loops is 3
```

Завдання до програми

Реалізуйте та вивчить програму з лістингу 3.2. Розробіть програму, що виконує задані обчислення, виводить на екран відповідні повідомлення і результати обчислень. Для організації циклу використати оператор `for()`. Загальну кількість чисел та числа вводити з клавіатури. У разі, якщо буде введено зазначене критичне значення, цикл повинен завершитись (оператор `break`) з відповідним повідомленням.

Варіант 1. Обчисліть суму та кількість тільки цілих додатних непарних чисел. При введенні числа `-11` цикл завершується.

Варіант 2. Обчисліть суму та кількість тільки цілих додатних парних чисел. При введенні числа, що більше за `101` цикл завершується.

Варіант 3. Обчисліть суму та кількість тільки додатних дійсних чисел, що більші `5`. При введенні числа `55` цикл завершується.

Варіант 4. Обчисліть суму та кількість тільки від'ємних дійсних чисел, що менші `-5`. При введенні числа меншого ніж `-99` цикл завершується.

Варіант 5. Обчисліть суму та кількість тільки цілих від'ємних непарних чисел. При введенні числа `13` цикл завершується.

3.3 Вкладений оператор циклу `for()`

Програма (див. лістинг 3.3) обчислює та виводить на екран таблицю множення, використовуючи вкладений оператор циклу.

З однією ітерацією зовнішнього циклу (рядки 7–13) виконується десять ітерацій внутрішнього циклу (рядок 10).

Лістинг 3.3 – Вкладений оператор циклу `for()` (файл `lab3-3.cpp`)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      int i, j;
6      /*outer loop*/
7      for (i = 3; i <= 5; i++) {
8          /*inner loop*/
9          for (j = 1; j <= 10; j++)
10             printf("%2d * %2d = %3d\n", i, j, i * j);
11         printf("\n");
12         system("pause");
13     }
14     return 0;
15 }
```

Результат роботи програми з лістингу 3.3 (фрагмент):

```
3 *  1 =   3
3 *  2 =   6
3 *  3 =   9
3 *  4 =  12
3 *  5 =  15
3 *  6 =  18
3 *  7 =  21
3 *  8 =  24
3 *  9 =  27
3 * 10 =  30

To continue, press any key . . .
4 *  1 =   4
4 *  2 =   8
4 *  3 =  12
4 *  4 =  16
4 *  5 =  20
4 *  6 =  24
4 *  7 =  28
4 *  8 =  32
4 *  9 =  36
4 * 10 =  40
```

Завдання до програми

Реалізуйте та вивчить програму з лістингу 3.3. Розробіть програму, що виконує обчислення значень функції, параметри якої змінюються у заданих діапазонах з заданим кроком. Для організації циклу використовуйте вкладений оператор циклу `for()`.

Результати виведіть на екран у вигляді рядків таблиць виразів обчислень (наприклад: `2 * 2 - 4 * 5 = -16`).

Варіант 1. Обчисліть $f(a,b) = a^2 + b^2$, де a змінюється у діапазоні $[4;20]$ з кроком 2, b – у діапазоні $[3;21]$ з кроком 3.

Варіант 2. Обчисліть $f(a,b) = a^2 - 4 \cdot b$, де a змінюється у діапазоні $[2;10]$ з кроком 1, b – у діапазоні $[5;20]$ з кроком 5.

Варіант 3. Обчисліть $f(a,b) = a^2 - b^2$, де a змінюється у діапазоні $[0;40]$ з кроком 4, b – у діапазоні $[6;20]$ з кроком 2.

Варіант 4. Обчисліть $f(a,b) = 2 \cdot a^2 + b$, де a змінюється у діапазоні $[3;30]$ з кроком 3, b – у діапазоні $[8;20]$ з кроком 4.

Варіант 5. Обчисліть $f(a,b) = 5 \cdot a + b^2$, де a змінюється у діапазоні $[5;50]$ з кроком 5, b – у діапазоні $[4;10]$ з кроком 2.

3.4 Використання `break` та `continue` у циклі з `while()`

Програма (див. лістинг 3.4) обчислює суму та добуток п'яти непарних значень введених з клавіатури чисел. Підраховується кількість чисел: загальна (рядок 12), непарних (рядок 17) та парних (рядок 14) чисел. Якщо введено число 0, то цикл завершується.

Лістинг 3.4 – Використання `break` та `continue` (файл `lab3-4.cpp`)

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int n_all = 0, n_odd = 0, n_even = 0, a, sum = 0, pr = 1;
6      const int amount_odd = 5;
7      cout << "Enter integers:\n";
8      while (n_odd < amount_odd) {
9          cin >> a;
10         if (a == 0)
11          break;
12         n_all++;
13         if (!(a % 2)) {
14             n_even++;
15             continue;
16         }
17         n_odd++;
18         sum += a;
19         pr *= a;
20         cout << "Number of odd values = " << n_odd << '\n';
21         cout << "Sum of odd values = " << sum << '\n';
22         cout << "Product of odd values = " << pr << '\n';
23     }
24     cout << "Number of entered values = " << n_all << '\n';
25     cout << "Number of even values = " << n_even;
26     return 0;
27 }
```

Результат роботи програми з лістингу 3.4 може бути наступним:

```
Enter integers:
2
13
Number of odd values = 1
Sum of odd values =13
Product of odd values =13
-2
-3
Number of odd values = 2
Sum of odd values =10
```

```
Product of odd values =-39
1
Number of odd values = 3
Sum of odd values =11
Product of odd values =-39
10
3
Number of odd values = 4
Sum of odd values =14
Product of odd values =-117
5
Number of odd values = 5
Sum of odd values =19
Product of odd values =-585
Number of entered values = 8
Number of even values = 3
```

Завдання до програми

Реалізуйте та вивчить програму з лістингу 3.4. Розробіть програму, що виконує задані обчислення з числами, що вводяться з клавіатури, виводить на екран відповідні повідомлення та результати обчислень.

Для організації циклу використовуйте оператор циклу `while()`. Кількість введених чисел задайте константою (`const`). Обчисліть суму та кількість чисел, які задовольняють заданій умові, та кількість інших чисел (оператор `continue`). При введенні деякого критичного значення організуйте завершення циклу (оператор `break`).

Варіант 1. Знайдіть найбільше з десяти введених цілих чисел. Обчисліть суму чисел, що більші за `5`. Завершення циклу при введенні числа `11`.

Варіант 2. Знайдіть найменше з десяти введених дійсних чисел. Обчисліть суму чисел, що більші за `2.5`. Завершення циклу при введенні числа менше ніж мінус `1`.

Варіант 3. Знайдіть найбільше з дванадцяти введених цілих чисел. Обчисліть суму чисел, що знаходяться у діапазоні від `48` до `122`, та виведіть їх у десятковому та символьному вигляді. Завершення циклу при введенні числа `0`.

Варіант 4. Знайдіть найменше з п'ятнадцяти введених дійсних чисел. Обчисліть суму від'ємних чисел. Завершення циклу при введенні числа менше ніж мінус `100`.

Варіант 5. Знайдіть найбільше за абсолютним значенням з шести введених довгих цілих чисел. Обчисліть суму чисел, які за абсолютним значенням менші ніж `3000`. Завершення циклу при введенні числа `999`.

3.5 Використання оператору циклу `do...while()`

Програма (див. лістинг 3.5) обчислює $\arccos arg$ і $\arcsin arg$. Значення arg при введенні контролюється на відповідність заданому діапазону (цикл з `do ... while()`).

Лістинг 3.5 – Оператор циклу `do...while()` (файл `lab3-5.cpp`)

```
#include <iostream>
#include <math.h>
using namespace std;

int main() {
    double arg;
    int counter = 0;
    do {
        cout << "Enter value: -1<= arg <=1\n";
        cin >> arg;
        counter++;
    } while (arg < -1.0 || arg > 1.0);
    cout << "arcsin (" << arg << ") = " << asin(arg) << '\n';
    cout << "arccos (" << arg << ") = " << acos(arg) << '\n';
    cout << "Total values entered: " << counter;
    return 0;
}
```

Результат роботи програми з лістингу 3.5 може бути наступним:

```
Enter value: -1<= arg <=1
-1.275
Enter value: -1<= arg <=1
1.36
Enter value: -1<= arg <=1
0.5
arcsin (0.5) = 0.523599
arccos (0.5) = 1.0472
Total values entered: 3
```

Примітка. Введення значень змінної `arg` відбувається у циклі, поки умова зазначена в операторі `while()` є істиною, тобто поки не знайдеться одне значення `arg`, для якого буде виконане обчислення після виходу із циклу. Підраховується загальна кількість введених значень.

Завдання до програми

Реалізуйте та вивчить програму з лістингу 3.5. Розробіть програму, що виконує задані обчислення, виводить на екран результати, кількість введених та обчислених чисел.

Значення `arg` повинні вводитись з клавіатури у циклі з `for()` та знаходитись у заданому діапазоні (оператор `do...while()`). Загальна кількість введених чисел не повинна перевищувати 12, інакше програма завершується (оператор `return -1;`).

Математичні функції для виконання даного завдання входять до бібліотеки `math` (або `cmath`): `sin()`, `cos()`, `tan()`, `sinh()`, `cosh()`, `atan()`. Для математичної константи π використовуйте константу `M_PI` з даної бібліотеки.

Варіант 1. Обчисліть синус та косинус тих `п'яти` дійсних чисел, що знаходяться в діапазоні $[0; 2\pi]$.

Варіант 2. Обчисліть секанс та косеканс тих `шести` дійсних чисел, що знаходяться в діапазоні $[\pi/20; \pi/2]$ ($\sec arg = 1/\cos arg$, $\mathrm{cosec}\, arg = 1/\sin arg$).

Варіант 3. Обчисліть гіперболічні синус та косинус тих `п'яти` дійсних чисел, що знаходяться в діапазоні $[-3\pi; 3\pi]$.

Варіант 4. Обчисліть тангенс та котангенс тих `шести` дійсних чисел, що знаходяться в діапазоні $[0; \pi]$ ($\mathrm{ctg}\, arg = 1/\mathrm{tg}\, arg$).

Варіант 5. Обчисліть арктангенс та арккотангенс тих `п'яти` дійсних чисел, що знаходяться в діапазоні $[\pi; 20\pi]$ ($\mathrm{arcctg}\, arg = \pi/2 - \mathrm{arctg}\, arg$).

3.6 Контрольні питання та завдання до розділу №3

1. Які оператори циклу використовуються у мові C/C++?
2. Наведіть код реалізації і результати виконання завдання до програми з лістингу 3.1 за вашим варіантом з застосуванням оператору циклу `for()` замість `while()`.
3. Наведіть код реалізації і результати виконання завдання до програми з лістингу 3.2 за вашим варіантом з застосуванням оператору циклу `while()` замість `for()`.
4. Наведіть код реалізації і результати виконання завдання до програми з лістингу 3.2 за вашим варіантом з застосуванням оператору циклу `do...while()` замість `for()`.
5. Як виконується оператор `for()` із роздільником кома (,)? Визначте та поясніть результати роботи нижченаведеного фрагменту коду.

```
1     int a, b, c;
2     for (a = 0, b = 0, c = 5; c > 10, a < 3; a++, b += c) {
3        c += 3;
4        printf("a = %d b = %d c = %d\n", a, b, c);
5     }
6     printf("Final result: a = %d b = %d c = %d\n", a, b, c);
```

6. Визначте та поясніть результати роботи наступного фрагменту коду. Поясніть результат роботи коду при заміні у рядку 1 значення змінної `i = 3`.

```
1     int i = 0, a = 2, b;
2     do {
3        b = i * i + a * a;
4        i++;
5        a += 2;
6        cout << "i = " << i << "\ta = " << a << "\tb = " <<  b
             << endl;
7     } while (i < 3);
```

7. Поясніть призначення операторів `continue`, `break`, `return` та наведіть приклади їх використання у програмах із завдання за вашим варіантом.

8. Дослідіть і поясніть результат роботи наступного фрагменту коду при заданих варіантах виразів умови. Для кожного випадку підрахуйте кількість циклів.

```
1     short n = 0;
2     while (a_condition)
3        cout << n << " ";
4     cout << " Last value of the number is " << n;
```

Варіанти заміни умови `(a_condition)` у рядку 2:

(a) `(n++, n < 5)`
(b) `(n < 5, n++)`
(c) `(n < 5, ++n)`

ЛАБОРАТОРНА РОБОТА 4
Масиви і вказівники

Мета роботи – ознайомитися з засобами організації роботи з масивами даних у мові C/C++, навчитися використовувати *одновимірні масиви* для зберігання, сортування та пошуку необхідних даних, використовувати *вказівники* при роботі з масивами, розроблювати програми, що використовують прості алгоритми сортування та пошуку.

4.1 Одновимірні масиви, ініціалізація значень елементів

Програма (див. лістинг 4.1) визначає найменше значення серед елементів масиву даних (див. рядки 16—18).

У рядках 6—8 визначено та ініціалізовано одновимірний масив цілих чисел із дев'яти елементів (`arr[9]`). Значення та адреси елементів масиву виводяться на екран у циклі з оператором `for`. Звернення до елементів масиву відбувається за допомогою імені масиву і індексу елементу (`arr[i]`), що змінюється від 0 до 8 (рядки 9—12).

У рядку 13 визначається *довжина* масиву (розмір у байтах).

Лістинг 4.1 – Пошук найменшого значення (файл `lab4-1.cpp`)

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4
5  int main() {
6     int arr[9] = {
7        4, 2, -3, 8, 12, -2, 9, 0, 15
8     };
9     for (int i = 0; i < 9; i++) {
10       cout << "arr[" << i << "] = " << setw(2) << arr[i];
11       cout << ", &arr[" << i << "] = " << &arr[i] << '\n';
12    }
13    cout << "The array size is: " << sizeof arr << " bytes\n";
14    int min;
15    min = arr[0];
16    for (int i = 0; i < 9; i++)
17       if (arr[i] < min)
18          min = arr[i];
19    cout << "The minimum element of the array is " << min;
20    return 0;
21 }
```

Результат роботи програми з лістингу 4.1 може бути наступним:

```
arr[0] =  4, &arr[0] = 0x6ffde0
arr[1] =  2, &arr[1] = 0x6ffde4
arr[2] = -3, &arr[2] = 0x6ffde8
arr[3] =  8, &arr[3] = 0x6ffdec
arr[4] = 12, &arr[4] = 0x6ffdf0
arr[5] = -2, &arr[5] = 0x6ffdf4
arr[6] =  9, &arr[6] = 0x6ffdf8
arr[7] =  0, &arr[7] = 0x6ffdfc
arr[8] = 15, &arr[8] = 0x6ffe00
The array size is: 36 bytes
The minimum element of the array is -3
```

Завдання до програми

Реалізуйте та вивчить програму з лістингу 4.1. Розробіть програму, в якій з клавіатури вводяться значення елементів масиву цілих чисел і виконуються задані обчислення. На екран виводяться значення елементів масиву, довжина масиву у байтах і результати обчислень.

Варіант 1. Обчисліть середнє арифметичне елементів з ненульовими значеннями. Розмір масиву `6`.

Варіант 2. Обчисліть середнє арифметичне елементів без обліку тих, які дорівнюють мінімальному значенню. Розмір масиву `8`.

Варіант 3. Обчисліть середнє арифметичне тільки додатних елементів. Розмір масиву `5`.

Варіант 4. Обчисліть суму і кількість тільки парних елементів. Розмір масиву `6`.

Варіант 5. Обчисліть суму квадратів і кількість тільки від'ємних елементів. Розмір масиву `7`.

4.2 Одновимірні масиви, генерація випадкових значень

Програма (див. лістинг 4.2) шукає додатні елементи одновимірного масиву з найбільшим та найменшим значеннями (див. рядки 15—18) і підраховує кількість додатних чисел між ними (див. рядки 25—29).

Елементам масиву привласнюються випадкові значення, які генерує функція `rand()` у заданому діапазоні $[-50, 100]$ (рядок 11). Функція `srand()` (рядок 9) встановлює початкове значення для генерації псевдовипадкових чисел. Для того, щоб початкове значення не було фіксованим і генерувалась унікальна послідовність чисел, використовують функцію `time()`.

Лістинг 4.2 – Вибір елементів за умовою (файл `lab4-2.cpp`)

```
#include <iostream>
#include <time.h>
#include <stdlib.h>
using namespace std;

int main() {
    const int n = 15;
    int a[n];
    srand(time(NULL));
    for (int i = 0; i < n; i++) {
        a[i] = rand() % 151 - 50;
        cout << " a[" << i << "]=" << a[i] << '\n';
    }
    int i_max = 0, i_min = 0;
    for (int i = 0; i < n; i++) {
        if (a[i] > a[i_max]) i_max = i;
        if (a[i] < a[i_min]) i_min = i;
    }
    cout << "The maximum element a[" << i_max << "] is " <<
        a[i_max] << '\n';
    cout << "The minimum element a[" << i_min << "] is " <<
        a[i_min] << '\n';
    int i_beg = i_max < i_min ? i_max : i_min;
    int i_end = i_max < i_min ? i_min : i_max;
    cout << "Between a[" << i_beg << "] and a[" << i_end <<
        "]:" << '\n';
    int counter = 0;
   for (int i = i_beg + 1; i < i_end; i++)
        if (a[i] > 0) {
            cout << " a[" << i << "]=" << a[i] << '\n';
            counter++;
        }
    cout << counter << " element(s) have a positive value";
    return 0;
}
```

Результат роботи програми з лістингу 4.2 може бути наступним:

```
a[0]=98
a[1]=74
a[2]=58
a[3]=26
a[4]=-39
a[5]=99
a[6]=66
a[7]=-16
a[8]=61
a[9]=-43
a[10]=49
```

```
a[11]=42
a[12]=41
a[13]=-38
a[14]=72
The maximum element a[5] is 99
The minimum element a[9] is -43
Between a[5] and a[9]:
a[6]=66
a[8]=61
2 element(s) have a positive value
```

Завдання до програми

Реалізуйте та вивчить програму з лістингу 4.2. Розробіть програму, що заповнює масив цілих чисел випадковими значеннями, виконує задані обчислення, виводить на екран елементи масиву та результати обчислень.

Варіант 1. Розмір масиву `15`. Діапазон значень: $[-40; 40]$. Визначте кількість елементів з від'ємними значеннями.

Варіант 2. Розмір масиву `12`. Діапазон значень: $[0; 60]$. Визначте кількість елементів, значення яких кратні `3`.

Варіант 3. Розмір масиву `16`. Діапазон значень: $[-20; 20]$. Визначте кількість елементів, значення яких парні.

Варіант 4. Розмір масиву `15`. Діапазон значень: $[0; 100]$. Визначте кількість елементів, значення яких кратні `5`.

Варіант 5. Розмір масиву `14`. Діапазон значень: $[-30; 30]$. Визначте кількість елементів, значення яких непарні.

4.3 Вказівники. Операція розіменування вказівника

Програма (див. лістинг 4.3) визначає найбільше значення серед елементів масиву, використовуючи метод доступу до елементів через розіменування вказівника (рядки 16, 17).

У рядку 11 визначено вказівник `ptrarr` типу `int`, який ініціалізовано ім'ям масиву `arr` (ім'я масиву є вказівником). При цьому вказівник приймає значення адреси нульового елементу масиву, тобто адреси елементу `arr[0]`. У рядку 13 змінній `max` привласнюється значення нульового елементу масиву за допомогою операції розіменування вказівника, у даному випадку `max => *ptrarr => arr[0] => 5`.

При додаванні до вказівника цілого числа `(ptrarr + i)` отримаємо адресу відповідного елементу масиву, а при застосуванні операції розіменування `*(ptrarr + i)` отримаємо значення елементу.

Лістинг 4.3 – Вказівники на масив (файл `lab4-3.cpp`)

```
#include <iostream>
using namespace std;

int main() {
    int arr[10] = {
        5, 3, 2, -4, 6, 7, 110, -17, 0, 13
    };
    cout << "Array elements:\n";
    for (int i = 0; i < 10; i++)
        cout << arr[i] << " ";
    int *ptrarr = arr;
    int max;
    max = *ptrarr;
    int i = 0;
    while (i < 10) {
        if (*(ptrarr + i) > max)
         max = *(ptrarr + i);
        i++;
    }
    cout << "\nThe maximum element is " << max;
    return 0;
}
```

Результат роботи програми з лістингу 4.3:

```
Array elements:
5 3 2 -4 6 7 110 -17 0 13
The maximum element is 110
```

Завдання до програми

Реалізуйте та вивчить програму з лістингу 4.3. Розробіть програму, що виконує задані обчислення, використовуючи метод доступу до елементів масиву через розіменування вказівника, та виводить на екран значення елементів масиву та результати обчислень. У програмі визначте масив цілих чисел заданого розміру і ініціалізуйте *значення першого елементу* (*індекс 0*), а значення решти елементів введіть з клавіатури.

Варіант 1. Визначте найменше значення серед елементів масиву, що кратні 5. Значення першого елементу масиву: 500. Розмір масиву: 8.

Варіант 2. Визначте індекс найменшого значення серед додатних елементів масиву. Значення першого елементу масиву: 1000. Розмір масиву: 6.

Варіант 3. Визначте індекс найбільшого значення серед від'ємних елементів масиву. Значення першого елементу масиву: -2000. Розмір масиву: 7.

Варіант 4. Визначте найбільше значення серед парних елементів масиву. Значення першого елементу масиву: -100. Розмір масиву: 8.

Варіант 5. Визначте найменше значення серед непарних елементів масиву. Значення першого елементу масиву: 201. Розмір масиву: 7.

4.4 Сортування (впорядкування) елементів масиву

Програма (див. лістинг 4.4) впорядковує елементи масиву за зростанням методом вибору найбільшого значення.

Лістинг 4.4 – Сортування за зростанням (файл `lab4-4.cpp`)

```
#include <iostream>
#include <iomanip>
#define N 10
using namespace std;

int main() {
   int dig[N] = {
        50, 3, 25, 4, 6, 7, 11, 17, 0, 13
    };
    cout << "Array to sort:\n";
    for (int i = 0; i < N; i++)
      cout << dig[i] << " ";
   int max, i_max, tmp;
    for (int i = N - 1; i >= 1; i--) {
        max = dig[0];
        i_max = 0;
        for (int j = 1; j <= i; j++)
            if (dig[j] > max) {
                max = dig[j];
                i_max = j;
            }
        tmp = dig[i];
        dig[i] = max;
        dig[i_max] = tmp;
    }
    cout << "\nArray after sorting:\n";
    for (int i = 0; i < N; i++)
        cout << dig[i] << " ";
    return 0;
}
```

Результат роботи програми з лістингу 4.4:

```
Array to sort:
50 3 25 4 6 7 11 17 0 13
Array after sorting:
0 3 4 6 7 11 13 17 25 50
```

Примітка. Сортування вибором працює циклічно: на кожному проході при сортуванні за зростанням знаходимо максимальне (мінімальне) значення серед елементів, що залишились, і ставимо його на кінець (початок) масиву шляхом обміну з порівнюваним елементом.

Завдання до програми

Реалізуйте та вивчить програму з лістингу 4.4. Розробіть програму, що впорядковує та виводить на екран одновимірний масив до і після сортування. Масив заповніть випадковими значеннями у заданому діапазоні.

Варіант 1. Значення елементів масиву (цілі, діапазон: $[-40; 40]$) впорядкувати за убуванням методом «бульбашки».

Варіант 2. Значення елементів масиву (цілі, діапазон: $[-60; 60]$) впорядкувати за зростанням методом прямого упорядкування.

Варіант 3. Значення елементів масиву (дійсні, діапазон: $[-50.5; 50.5]$) впорядкувати за убуванням методом вибору найменшого значення.

Варіант 4. Значення елементів масиву (цілі, діапазон: $[-100; 100]$) впорядкувати за зростанням методом «бульбашки».

Варіант 5. Значення елементів масиву (дійсні, діапазон: $[-60.5; 60.5]$) впорядкувати за убуванням методом прямого упорядкування.

4.5 Введення-виведення рядків

Програма (див. лістинг 4.5) демонструє введення з клавіатури рядків та виведення їх на екран різними способами (`scanf/printf`; `gets/puts`). Після введення кожного рядку з клавіатури натискається клавіша `Enter`.

Рядки `str1`, `str2`, `str3` визначені у програмі як одновимірні масиви символів довжиною `K` і можуть складатися з декількох слів розділених прогалинами (див. рядок 7).

Для роботи з рядками існує багато корисних функцій. Деякі з них можуть вам знадобитись для виконання завдання за варіантом. Для їх використання підключіть до програми заголовковий файл `<string.h>`.

Наприклад, щоб отримати довжину рядка `str1`, можна скористатися функцією `strlen(str1)`, результат роботи якої може відрізнятись від результату обчисленого `sizeof(str1)`.

Для об'єднання (конкатенації) двох рядків (`str1` та `str2`) можна використати функцію `strcat(str1,str2)`, яка приєднає вміст `str2` до вмісту `str1` і збереже результат у `str1`. Розмір `str1` при цьому має бути достатньо великим.

Лістинг 4.5 – Введення-виведення рядків (файл `lab4-5.cpp`)

```
#include <stdio.h>
#include <iostream>
#define K 80
using namespace std;

int main() {
   char str1[K], str2[K], str3[K];
    puts("Enter 1st string of 2 or 3 words:");
    scanf("%s", str1);
    fflush(stdin);
    printf("\n1st string: %s\n", str1);

    puts("\nEnter 2nd string of 2 or 3 words:");
    gets(str2);
    printf("\n2nd string: ");
    puts(str2);

    cout << "\nEnter 3rd string of 2 or 3 words:\n";
    cin >> str3;
    cout << "\n3rd string: ";
    cout << str3 << '\n';
    return 0;
}
```

Результат роботи програми з лістингу 4.5 може бути наступним:

```
Enter 1st string of 2 or 3 words:
Object-oriented programming

1st string: Object-oriented

Enter 2nd string of 2 or 3 words:
C++ programming language

2nd string: C++ programming language

Enter 3rd string of 2 or 3 words:
Programming environment

3rd string: Programming
```

Завдання до програми

Реалізуйте та вивчить програму з лістингу 4.5. Розробіть програму, що виконує задані дії з визначеними трьома рядками, виводить на екран усі рядки та кількість підрахованих символів, що містить кожен рядок.

У перший рядок задані слова (з прогалинами) ввести з клавіатури,

другий рядок ініціалізувати заданим повідомленням. Третій рядок повинен об'єднати дані першого та другого рядків.

Варіант 1. Перший рядок: назва навчальної дисципліни і прізвище та ім'я викладача. Другий: інформація про час проведення консультації.

Варіант 2. Перший рядок: прізвище та ім'я друга. Другий: побажання гарного настрою і успіхів у навчанні.

Варіант 3. Перший рядок: назва фільму і кінокомпанія виробник. Другий: інформація про час появи у прокаті.

Варіант 4. Перший рядок: поштова адреса. Другий: повідомлення про надходження листа на адресу.

Варіант 5. Перший рядок: назва книги, прізвище автора. Другий: повідомлення про надходження книги до бібліотеки.

4.6 Контрольні питання та завдання до розділу №4

1. Що таке масив? Як визначити та ініціалізувати масив у мові C/C++? Наведіть приклад масиву з ініціалізацією.

2. Наведіть приклади визначення масивів різних класів пам'яті (auto, static) локальної та глобальної області видимості. Як відбувається їх неявна ініціалізація.

3. Наведіть приклади різних способів визначення розміру масиву у мові C/C++?

4. Як обчислити загальний розмір масиву у байтах. Наведіть приклад та поясніть результат.

5. Що таке вказівник? Визначення та ініціалізація вказівника у мові C/C++. Що визначає його тип та який розмір вказівника у байтах? Що показує різниця вказівників?

6. Визначте та поясніть результати роботи наступного фрагменту коду:

```
1     short arr[5] = {
2        50, 30, 20, 60, 80
3     };
4     short *ptr = arr;
5     cout << "*ptr = " << *ptr << '\n';
6     cout << "*(ptr + 2) = " << *(ptr + 2) << '\n';
7     cout << "*ptr + 2 = " << *ptr + 2 << '\n';
8     cout << "(*++ptr - 1) = " << *++ptr - 1 << '\n';
9     cout << "sizeof arr = " << sizeof arr<< '\n';
10    cout << "sizeof arr[3] = " << sizeof arr[3] << '\n';;
11    cout << "sizeof ptr = " << sizeof ptr << '\n';;
12    cout << "sizeof *ptr = " << sizeof *ptr << '\n';
```

7. Якими способами можна звернутися до останнього елементу масиву `int A[8]`, використовуючи ім'я масиву/вказівника та індекс елементу? Запишіть приклади щонайменше восьми способів.

8. Які алгоритми впорядкування елементів масиву вам відомі? Наведіть код реалізації та поясніть один з алгоритмів на прикладі програми із завдання до лістингу 4.4 за вашим варіантом.

9. Як пов'язані між собою поняття ім'я масиву та вказівник? Поясніть різницю між ними. Що таке вказівник-константа? Поясніть результати роботи наступного фрагменту коду та визначте помилкові конструкції:

```
1     int A[10], X[10], *PA, *PB;
2     PA = &A[0];
3     PB = A;
4     X = PA;
```

10. Знайдіть помилки у рядках коду з лістингу 4.6, запишіть коректний код і поясніть виправлення. Наведіть результати роботи виправленої програми. У разі покращення коду, окремо поясніть таку необхідність.

Лістинг 4.6 – Валідація коду з масивами/вказівниками

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      const int N = 5;
6      int arr[6] = {
7          -20, -16, -20, -45, -85
8      };
9      int barr[N] = {
10         10, 15, -9, 4, 12, 0
11     };
12     int *ptr_arr, *ptr_barr;
13     ptr_arr = &arr[0];
14     ptr_barr = barr;
15     int max_arr = *ptr_arr;
16     for (int i = 0; i < 6; ++i)
17         if (ptr_arr[i] > max_arr)
18             max_arr = *(ptr_arr + i);
19     cout << "The max value is " << max_arr << '\n';
20     N = 6;
21     for (int i = 0; i <= N; ++i)
22         cout << barr[i] << "\n";
23     for (int j = 0; j < N; ++j)
24         ptr_barr[j] = 1;
25     return 0;
26 }
```

ЛАБОРАТОРНА РОБОТА 5
Двовимірні і динамічні масиви. Масиви вказівників

Мета роботи – ознайомитися з різними масивами у мові C/C++ (двовимірними, динамічними, вказівників), навчитися використовувати динамічну пам'ять при роботі з масивами, вдосконалити навички роботи з різними масивами та вказівниками для зберігання, пошуку та обробки необхідних даних, розроблювати та тестувати відповідні програми.

5.1 Обчислення у двовимірному масиві

Програма (див. лістинг 5.1) обчислює суму значень елементів матриці (*двовимірного масиву*) по кожному рядку. Результати обчислень зберігаються в одновимірному масиві. У рядках 15—19 реалізовано вивід елементів масиву на екран у вигляді матриці.

Лістинг 5.1 – Обчислення по рядках масиву (файл `lab5-1.cpp`)

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4  int arr[4][3] = {
5     { 2, 3, 4 },
6     { 7 },
7     { 1, 9 },
8     { 8, 4, 10 }
9  };
10 int s1[4];
11
12 int main() {
13     int i, j;
14     cout << "Array arr[4][3]\n";
15     for (i = 0; i < 4; i++) {
16         for (j = 0; j < 3; j++)
17             cout << setw(5) << arr[i][j];
18         cout << '\n';
19     }
20     cout << "Sum of elements in rows\n";
21     for (i = 0; i < 4; i++) {
22         for (j = 0; j < 3; j++)
23             s1[i] += arr[i][j];
24         cout << "s1[" << i << "]=" << s1[i] << '\n';
25     }
26     return 0;
27 }
```

Результат роботи програми з лістингу 5.1:

```
Array arr[4][3]
2    3    4
7    0    0
1    9    0
8    4   10
Sum of elements in rows
s1[0]=9
s1[1]=7
s1[2]=10
s1[3]=22
```

Примітка. Для обчислення суми значень елементів матриці по кожному стовпцю для масиву `s2[3]` рядки коду можна записати наступним чином:

```
cout << "Sum of elements in columns\n";
for (j = 0; j < 3; j++) {
   for (i = 0; i < 4; i++)
      s2[j] += arr[i][j];
   cout << "s2[" << j << "]=" << s2[j] << '\n';
}
```

Завдання до програми

Реалізуйте та вивчить програму з лістингу 5.1. Розробіть програму, яка виконує задані обчислення. Значення елементів двовимірного масиву цілих чисел введіть з клавіатури. Результати обчислень занесіть в одновимірний масив. Виведіть на екран масиви у вигляді матриць.

Для тестування програми потрібно підібрати дані, що демонструють коректне виконання обчислень. Наприклад, результат обчислення елементу `s1[1]` з лістингу 5.1 не показовий для перевірки підраховування суми значень.

Варіант 1. Обчисліть суму значень елементів у стовпцях масиву, що додатні і непарні. Розмір масиву `5` на `4`.

Варіант 2. Обчисліть кількість елементів у рядках масиву, що мають додатні і парні значення. Розмір масиву `3` на `6`.

Варіант 3. Обчисліть кількість елементів у стовпцях масиву, що мають від'ємні і непарні значення. Розмір масиву `5` на `3`.

Варіант 4. Обчисліть суму значень елементів у рядках масиву, що від'ємні і парні. Розмір масиву `3` на `5`.

Варіант 5. Обчисліть добуток значень елементів у стовпцях масиву, що знаходяться у діапазоні від 1 до 5. Розмір масиву `4` на `6`.

5.2 Способи доступу до елементів двовимірного масиву

Програма (див. лістинг 5.2) обчислює суму значень елементів двовимірного масиву, які знаходяться у діапазоні $(0; 30]$, і демонструє різні варіанти доступу до елементів, використовуючи ідентифікатори і знаки операцій.

Лістинг 5.2 – Способи доступу до елементів (файл `lab5-2.cpp`)

```
#include <stdlib.h>
#include <time.h>
#include <iostream>
#include <iomanip>
using namespace std;

const int N = 14,
    M = 13;
float A[N][M], sum;

int main() {
    srand(time(NULL));
    cout << "Matrix:\n";
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            *(*(A + i) + j) = rand() % 10000 * 0.01f - 50.f;
            cout << setw(8) << fixed << setprecision(2);
            cout << *(A[i] + j);
        }
        cout << '\n';
    }
    for (int i = 0; i < N; i++)
        for (int j = 0; j < M; j++)
            if (A[i][j] > 0.f && A[i][j] <= 30.f)
                sum += (*(A + i))[j];
    cout << "The sum of values of matrix elements in the
        range from 0 to 30 is " << sum;
    return 0;
}
```

Результат роботи програми з лістингу 5.2:

```
Matrix:
 29.00   11.31  -19.33
-14.08    9.44   12.83
 -9.37  -27.48   -9.88
 22.02   -8.61  -22.63
The sum of values of matrix elements in the range from 0 to 30
    is 84.60
```

Завдання до програми

Реалізуйте та вивчить програму з лістингу 5.2. Розробіть програму, у якій двовимірний масив заповнюється випадковими числами і виконуються задані обчислення. При роботі з масивом використовуйте різні можливі способи доступу до елементів. Масив виведіть на екран у вигляді матриці.

Варіант 1. Обчисліть кількість елементів, що більші за `15.5`. Діапазон значень: [`10.50;30.50`]. Розмір масиву `5` на `4`.

Варіант 2. Обчисліть середнє арифметичне додатних елементів. Діапазон значень: [`-25.0;65.0`]. Розмір масиву `6` на `3`.

Варіант 3. Обчисліть добуток елементів з ненульовими значеннями. Діапазон значень: [`-5;5`]. Розмір масиву `8` на `7`.

Варіант 4. Обчисліть кількість елементів, що менші за `20.5`. Розмір масиву `5` на `4`. Діапазон значень: [`-10.5;55.5`].

Варіант 5. Обчисліть кількість від'ємних елементів. Розмір масиву `5` на `6`. Діапазон значень: [`-50.000;30.000`].

5.3 Масив вказівників

Програма (див. лістинг 5.3) демонструє роботу з масивом рядків, що визначений як масив вказівників на рядки типу `char *` (рядок 6). Обчислюється розмір у байтах масиву вказівників, кожного вказівника окремо та кількість символів у кожному рядку (див. рядок 15).

Лістинг 5.3 – Масив вказівників на рядки (файл `lab5-3.cpp`)

```
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4
5  int main() {
6    const char *fi[3] = {"September","October","November"};
7    cout << fi[0] << " " << fi[1] << " " << fi[2] << "\n\n";
8
9    cout << "Size of array is " << sizeof(fi) << " bytes;\n";
10   for (int i = 0; i < 3; i++)
11       cout<<i+1<<" pointer is "<<sizeof(fi[i])<<" bytes;\n";
12
13   cout << '\n' << "Length of strings: " << '\n';
14   for (int i = 0; i < 3; i++)
15       cout<<fi[i]<<" is "<<strlen(fi[i])<<" symbols\n";
16   return 0;
17 }
```

Результат роботи програми з лістингу 5.3 може бути наступним:

```
September October November

Size of array is 24 bytes;
1 pointer is 8 bytes;
2 pointer is 8 bytes;
3 pointer is 8 bytes.

Length of string:
September is 9 symbols
October is 7 symbols
November is 8 symbols
```

Завдання до програми

Реалізуйте та вивчить програму з лістингу 5.3. Розробіть програму, у якій рядки масиву ініціалізуйте даними за варіантом, виконайте задані обчислення та виведіть результати на екран.

Варіант 1. Вміст рядків: прізвище, ім'я, по батькові. Обчисліть кількість символів `'o'` у третьому рядку.

Варіант 2. Вміст рядків: вулиця, місто, країна. Обчисліть кількість символів `'e'` у першому рядку.

Варіант 3. Вміст рядків: університет, факультет, спеціальність. Обчисліть кількість символів `'i'` у першому рядку.

Варіант 4. Вміст рядків: викладач, дисципліна, тема. Обчисліть кількість символів `'a'` у другому рядку.

Варіант 5. Вміст рядків: автор, книга, жанр. Обчисліть кількість символів `'y'` у другому рядку.

5.4 Масиви динамічної пам'яті (`new/delete`; `malloc()/free()`)

Програма (див. лістинг 5.4) демонструє роботу з динамічними масивами. В програмі створюється динамічний масив для змінних типу `int`. Розмір масиву та значення елементів масиву вводяться з клавіатури.

Запит пам'яті для масиву реалізовано за допомогою операції `new` (див. рядок 9), перевірка виділення пам'яті – у рядках 11—14. По закінченні роботи з масивом пам'ять звільняється (операція `delete`, рядок 22).

Для запиту у програмі з лістингу 5.4 динамічної пам'яті за допомогою функції `malloc()` рядок 9 потрібно замінити на рядок 10, а для звільнення пам'яті за допомогою функції `free()` – рядок 22 на рядок 23. Результат роботи програми буде аналогічним.

Лістинг 5.4 – Одновимірний динамічний масив (файл `lab5-4.cpp`)

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cout << "Enter size of a one-dimensional array:\n";
    cin >> n;
    int *ptr;
    ptr = new int[n];
    //ptr = (int*) malloc(n * sizeof(int));
    if (!ptr) {
        cout << "Error";
        return -1;
    }
    cout << "Enter elements of an array\n";
    for (int i = 0; i < n; i++)
        cin >> ptr[i];
    cout << "Contents of the array:\n";
    for (int i = 0; i < n; i++)
        cout << *(ptr + i) << " ";
    cout << '\n';
    delete[] ptr;
    //free(ptr);
    return 0;
}
```

Результат роботи програми з лістингу 5.4 може бути наступним:

```
Enter size of a one-dimensional array:
5
Enter elements of an array
-105
56
77
-19
23
Contents of the array:
-105 56 77 -19 23
```

Завдання до програми

Реалізуйте та вивчить програму з лістингу 5.4. Розробіть програму, у який створюється динамічний масив з використанням функції `malloc()`. Розмір масиву введіть з клавіатури. Перевірте, чи відбулось виділення пам'яті. Масив заповніть випадковими числами у заданому діапазоні та виведіть на екран. Після цього пам'ять звільніть.

Варіант 1. Тип даних: `long`. Діапазон значень: [`-5000;5000`].

Варіант 2. Тип даних: `float`. Діапазон значень: [`-50.5;50.5`].
Варіант 3. Тип даних: `double`. Діапазон значень: [`0.00;32.00`].
Варіант 4. Тип даних: `double`. Діапазон значень: [`-15.000;15.000`].
Варіант 5. Тип даних: `char`. Діапазон значень: [`65;122`]. Масив виведіть на екран у десятковому та символьному вигляді.

5.5 Двовимірні динамічні масиви

Програма (див. лістинг 5.5) демонструє роботу з двовимірними масивами динамічної пам'яті і обчислює суму двох квадратних матриць.

В програмі за допомогою операції `new` створюються два масиви. Розмір масивів та значення елементів першого масиву вводяться з клавіатури. Другий масив заповнюється випадковими числами у діапазоні [`-50;50`]. Результат обчислень заноситься до другого масиву. Масиви у вигляді матриць виводяться на екран. Звільняється пам'ять.

Лістинг 5.5 – Масиви динамічної пам'яті (файл `lab5-5.cpp`)

```
1  #include <stdlib.h>
2  #include <iostream>
3  #include <time.h>
4  #include <iomanip>
5  using namespace std;
6
7  int main() {
8      int N;
9      cout << "Enter the number of rows(columns) of a square
           matrix:\n";
10     cin >> N;
11     int **matr;
12     matr = new int*[N];
13     if (!matr) {
14         cout << "Error!\n";
15         return -1;
16     }
17     for (int i = 0; i < N; i++) {
18         matr[i] = new int[N];
19         if (!matr[i]) {
20             cout << "Error!\n";
21             return -2;
22         }
23     }
24     cout << "Enter elements of the 1st matrix:\n";
25     for (int i = 0; i < N; i++)
26         for (int j = 0; j < N; j++)
27             cin >> matr[i][j];
```

Кінець лістингу 5.5

```
28      cout << "Contents of the 1st matrix:\n";
29      for (int i = 0; i < N; i++) {
30         for (int j = 0; j < N; j++)
31            cout << setw(8) << matr[i][j];
32         cout << '\n';
33      }
34      int **mas;
35      mas = new int*[N];
36      if (!mas) {
37         cout << "Error!\n";
38         return -3;
39      }
40      for (int i = 0; i < N; i++) {
41         mas[i] = new int[N];
42         if (!mas[i]) {
43            cout << "Error!\n";
44            return -4;
45         }
46      }
47      srand(time(NULL));
48      cout << "Contents of the 2nd matrix:\n";
49      for (int i = 0; i < N; i++) {
50         for (int j = 0; j < N; j++) {
51            mas[i][j] = rand() % 101 - 50;
52            cout << setw(8) << mas[i][j];
53         }
54         cout << '\n';
55      }
56      cout << "Sum of matrices:\n";
57      for (int i = 0; i < N; i++) {
58         for (int j = 0; j < N; j++) {
59            mas[i][j] = matr[i][j] + mas[i][j];
60            cout << setw(8) << mas[i][j];
61         }
62         cout << '\n';
63      }
64
65      for (int i = 0; i < N; i++) {
66         delete matr[i];
67         delete mas[i];
68      }
69      delete[] matr;
70      delete[] mas;
71      return 0;
72   }
```

Примітка. У випадку з прямокутною матрицею програму з лістингу 5.5 потрібно доопрацювати.

Результат роботи програми з лістингу 5.5 може бути наступним:

```
Enter the number of rows(columns) of a square matrix:
2
Enter elements of the 1st matrix:
-15
26
135
-48
Contents of the 1st matrix:
         -15       26
         135      -48
Contents of the 2st matrix:
          33      -50
          30      -35
Sum of matrices:
          18      -24
         165      -83
```

Завдання до програми

Реалізуйте та вивчить програму з лістингу 5.5. Доповніть дану програму наступним: створіть третю матрицю – динамічний масив за допомогою операції `new`. Перевірте, чи відбулось виділення пам'яті. Виконайте задані відповідно варіанту дії. Результат обчислень занесіть в третю матрицю та виведіть її значення на екран. Потім звільніть пам'ять (`delete`).

Розмір масивів вводиться з клавіатури.

Варіант 1. Порівняйте значення елементів перших двох матриць, найбільше значення відповідних елементів занесіть в третю матрицю.

Варіант 2. Першу матрицю заповніть випадковими числами у діапазоні [`0;100`]. В третю матрицю занесіть суму елементів перших двох матриць, якщо її значення знаходиться у діапазоні [`65;122`], інакше занесіть число `46`. Результат виведіть на екран у символьному вигляді.

Варіант 3. Порівняйте значення елементів перших двох матриць. Якщо відповідні елементи від'ємні, то в третю матрицю занесіть `-1`, якщо додатні – `1`, інакше `0`.

Варіант 4. Обчисліть добуток першої та другої матриць і занесіть результат у третю матрицю.

Варіант 5. Першу матрицю заповніть випадковими числами у діапазоні [`-100;100`]. В третю матрицю занесіть суму абсолютних значень елементів перших двох матриць.

Примітка. Добуток матриць: $c_{ij} = \sum a_{ik} \cdot b_{kj}$.

5.6 Контрольні питання та завдання до розділу №5

1. Що таке багатовимірний масив? Як розташовуються в пам'яті елементи багатовимірного масиву?

2. Наведіть приклади повної і неповної явної ініціалізації двовимірного масиву розміром `3` на `4` елементів типу `int`.

3. Визначте та поясніть результати роботи наступного фрагменту коду:

```
short m[2][3] = {
    1, 5, 7,
    2, 6, 8
};
short *pm = m[0];
cout << " **(m + 1) = " << **(m + 1) << '\n';
cout << " *(*(m + 1) + 1) = " << *(*(m + 1) + 1) << '\n';
cout << " *pm + 2 = " << *pm + 2  << '\n';
cout << " *(pm + 2) = " << *(pm + 2) << '\n';
cout << " sizeof(m) = " << sizeof(m) << '\n';
cout << " sizeof(m[0]) = " << sizeof(m[0]) << '\n';
cout << " sizeof(m[0][0]) = " << sizeof(m[0][0]) << '\n';;
cout << " sizeof(pm) = " << sizeof(pm) << '\n';
```

4. Що таке масив вказівників? Наведіть приклад з явною повною ініціалізацією такого масиву.

5. Що таке динамічний масив? Чим відрізняється від звичайного?

6. Створіть динамічний масив `5` на `4` елементів типу `long` за допомогою функції `malloc()` та перевірте виділення пам'яті для масиву. Введіть значення елементів з клавіатури і виведіть масив на екран у вигляді матриці. Обчисліть суму остачі від ділення значень елементів на 10 по кожному рядку, занесіть результати обчислень у одновимірний динамічний масив і виведіть його на екран. Звільніть пам'ять.

7. Створіть динамічний масив `6` на `5` елементів типу `float` за допомогою оператора `new` та перевірте виділення пам'яті для масиву. Елементам масиву привласніть випадкові значення у діапазоні `[-50.0;50.0]`. Виведіть масив на екран у вигляді матриці. Обчисліть суму значень елементів по кожному стовпцю, занесіть результати обчислень у одновимірний динамічний масив і виведіть його на екран. Звільніть пам'ять.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Шпак З. Я. Програмування мовою С : навч. посіб., 2-ге вид., допов. Львів : Львівська політехніка, 2011. 436 с.

2. Gazi O. Modern C Programming: Including Standards C99, C11, C17, C23. Cham : Springer, 2023. 391 p. URL: https://doi.org/10.1007/978-3-031-45361-8.

3. Kernighan B. W., Ritchie D. M. The C programming language. Prentice Hall, 2016. 238 p.

4. Deitel P., Deitel H. C How to Program: with Case Studies in Applications and Systems Programming. Pearson, 2022. 832 p.

5. Уроки програмування на С++. URL: https://acode.com.ua/uroki-po-cpp/.

6. C Tutorial. URL: https://www.w3schools.com/c/index.php.

7. C++ Language. URL: https://cplusplus.com/doc/tutorial/.

ДОДАТОК А
Список лістингів

Лістинг 1.1 – Основні типи даних (файл `lab1-1.cpp`) 6
Лістинг 1.2 – Прості обчислення (файл `lab1-2.cpp`) 8
Лістинг 1.3 – Функції `scanf()` та `printf()` (файл `lab1-3.cpp`) 10
Лістинг 1.4 – Введення/виведення з `cin` та `cout` (файл `lab1-4.cpp`) 11
Лістинг 1.5 – Діапазон значень типів даних (файл `lab1-5.cpp`) . . . 12
Лістинг 1.6 – Виведення рядків з `printf()` (файл `lab1-6.cpp`) 14
Лістинг 1.7 – Валідація коду 16
Лістинг 2.1 – Умовний оператор `if...else` (файл `lab2-1.cpp`) . . . 17
Лістинг 2.2 – Вкладені оператори `if...else` (файл `lab2-2.cpp`) . . . 19
Лістинг 2.3 – Пошук максимального числа (файл `lab2-3.cpp`) 20
Лістинг 2.4 – Оператор-перемикач `switch()` (файл `lab2-4.cpp`) . . . 22
Лістинг 3.1 – Оператор циклу `while()` (файл `lab3-1.cpp`) 25
Лістинг 3.2 – Оператор циклу `for()` (файл `lab3-2.cpp`) 26
Лістинг 3.3 – Вкладений оператор циклу `for()` (файл `lab3-3.cpp`) . 27
Лістинг 3.4 – Використання `break` та `continue` (файл `lab3-4.cpp`) . 29
Лістинг 3.5 – Оператор циклу `do...while()` (файл `lab3-5.cpp`) . . . 31
Лістинг 4.1 – Пошук найменшого значення (файл `lab4-1.cpp`) . . . 34
Лістинг 4.2 – Вибір елементів за умовою (файл `lab4-2.cpp`) 36
Лістинг 4.3 – Вказівники на масив (файл `lab4-3.cpp`) 38
Лістинг 4.4 – Сортування за зростанням (файл `lab4-4.cpp`) 39
Лістинг 4.5 – Введення-виведення рядків (файл `lab4-5.cpp`) 41
Лістинг 4.6 – Валідація коду з масивами/вказівниками 43
Лістинг 5.1 – Обчислення по рядках масиву (файл `lab5-1.cpp`) . . . 44
Лістинг 5.2 – Способи доступу до елементів (файл `lab5-2.cpp`) . . . 46
Лістинг 5.3 – Масив вказівників на рядки (файл `lab5-3.cpp`) 47
Лістинг 5.4 – Одновимірний динамічний масив (файл `lab5-4.cpp`) . 49
Лістинг 5.5 – Масиви динамічної пам'яті (файл `lab5-5.cpp`) 50