Міністерство освіти і науки України
Національний університет водного господарства та природокористування

Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра обчислювальної техніки

05/03-01М

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт
з навчальної дисципліни «**Програмування**» (частина 2)
для здобувачів вищої освіти першого (бакалаврського) рівня
за освітньо-професійною програмою «Комп'ютерна інженерія»
спеціальності F7 «Комп'ютерна інженерія»
та освітньо-професійною програмою «Інформаційна безпека»
спеціальності F5 «Кібербезпека та захист інформації»
денної та заочної форми навчання

Рекомендовано
науково-методичною радою
з якості ННІКІТІ
Протокол № 4 від 30.03.2026 р.

Рівне – 2026

Методичні вказівки до лабораторних робіт з навчальної дисципліни «Програмування» (частина 2) для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійною програмою «Комп'ютерна інженерія» спеціальності F7 «Комп'ютерна інженерія» та освітньо-професійною програмою «Інформаційна безпека» спеціальності F5 «Кібербезпека та захист інформації» денної та заочної форми навчання. [Електронне видання] / Сидор А. І., Бойчура М. В., Іванчук Н. В. – Рівне : НУВГП, 2026. – 61 с.

Укладачі: Сидор А. І., к.т.н., завідувач кафедри обчислювальної техніки;
Бойчура М. В., к.т.н., доцент кафедри обчислювальної техніки;
Іванчук Н. В., к.т.н., доцент кафедри комп'ютерних наук та прикладної математики.

Відповідальний за випуск: Сидор А. І., к.т.н., завідувач кафедри обчислювальної техніки.

Керівник групи забезпечення
спеціальності
F7 «Комп'ютерна інженерія» Сидор А. І.

Керівник групи забезпечення
спеціальності
F5 «Кібербезпека та захист інформації» Бабич С. В.

ЗМІСТ

Вступ

Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни «Програмування» (частина 2) створені для поглиблення студентами знань та практичних навичок у програмуванні на мові С++. У цій частині студенти мають змогу ознайомитись із більш складними механізмами мови, які є необхідними для розробки професійного програмного забезпечення.

Кожна лабораторна робота спрямована на розвиток навичок роботи з розширеними можливостями С++, такими як складні типи даних, обробка текстової інформації, керування пам’яттю, робота з масивами та структурами, а також основи модульного програмування. У ході виконання лабораторних робіт студенти навчаться реалізовувати алгоритми обробки даних, ефективно використовувати динамічну пам’ять, створювати власні типи даних та організовувати командну роботу над проєктом.

Кожна тема супроводжується теоретичними поясненнями, прикладами та завданнями для самостійного виконання, що сприяє кращому засвоєнню матеріалу.

Нижче наведено короткий огляд змісту лабораторних робіт.

Лабораторна робота №6 «Використання операторів та різних типів даних». Метою лабораторної роботи є закріплення навичок роботи з базовими операторами мови та різними типами даних. Студенти матимуть змогу детально вивчити пріоритети операцій, правила перетворення типів та специфіку виконання арифметичних і логічних виразів, що є фундаментом для написання коректного коду.

Лабораторна робота №7 «Робота з рядками». У лабораторній роботі розглядаються методи обробки текстової інформації. Студенти навчаться працювати як з масивами символів (C-string), так і з об’єктами класу string, виконувати операції пошуку, заміни, конкатенації та перетворення рядків, що є необхідним для взаємодії з користувачем та обробки текстових даних.

Лабораторна робота №8 «Використання циклів, розгалужень, перерахувань (enum) та структур». Лабораторна робота присвячена поглибленому вивченню керуючих конструкцій та користувацьких типів даних. Вона допоможе зрозуміти, як групувати різнотипні дані за допомогою структур та підвищувати читабельність коду, використовуючи перерахування. Студенти також вдосконалять навички побудови складних алгоритмів з розгалуженнями.

Лабораторна робота №9 «Робота з масивами». У лабораторній роботі студенти опанують роботу зі статичними одновимірними масивами. Це надасть розуміння принципів зберігання однотипних даних у пам’яті, методів доступу до них, а також алгоритмів пошуку та сортування, що є базовими для обробки великих обсягів інформації.

Лабораторна робота №10 «Робота з динамічними масивами». Лабораторна робота допоможе освоїти принципи роботи з динамічною пам’яттю. Студенти навчаться виділяти та звільняти пам’ять під час виконання програми, працювати з вказівниками та створювати масиви змінного розміру, що дозволяє писати більш гнучкі та ефективні програми.

Лабораторна робота №11 «Робота з масивами та функціями». Лабораторна робота спрямована на вивчення взаємодії між даними та підпрограмами. Студенти навчаться передавати масиви у функції, повертати

результати обчислень та структурувати код, розділяючи логіку обробки даних на окремі функціональні блоки.

Лабораторна робота №12 «Командна розробка програм з використанням функцій». Виконання цієї лабораторної роботи дозволить студентам закріпити навички модульного програмування та підготуватися до реальної командної розробки. Студенти навчаться розділяти програму на заголовкові файли та файли реалізації, інтегрувати код різних учасників та створювати спільні проєкти.

Лабораторна робота №6

Тема: Використання операторів та різних типів даних

1. Мета лабораторної роботи

Набути практичні навички роботи з основними операторами мов програмування, вивчити властивості різних типів даних та їх взаємодію при виконанні арифметичних, логічних і порівняльних операцій. Закріпити знання з перетворення типів, пріоритетів операторів та побудови виразів у програмному коді.

2. Навчальні результати

У результаті виконання лабораторної роботи студент буде:

- знати:
  - класифікацію типів даних (цілі, дійсні, логічні, символьні, рядкові);
  - призначення арифметичних, логічних, порівняльних і умовних операторів;
  - правила перетворення типів даних;
  - пріоритети та порядок виконання операторів;
- вміти:
  - використовувати оператори для реалізації обчислювальних алгоритмів;
  - застосовувати різні типи даних у програмах;
  - контролювати коректність типів і уникати помилок несумісності;
  - аналізувати результати виконання виразів і проводити відлагодження коду.

3. Теоретичні відомості

3.1. Типи даних

У більшості мов програмування (C, C++, Java, Python, C#) типи даних поділяються на:

- цілі (int, short, long) – використовуються для зберігання цілих чисел;
- дійсні (float, double) – для чисел з плаваючою комою;
- логічні (bool) – приймають значення true або false;
- символьні (char) – зберігають один символ;
- рядкові (string) – послідовність символів.

3.2. Оператори в C++

Оператори C++ – це символи, які виконують певні математичні або логічні обчислення над заданими значеннями. Вони є основою будь-якої мови програмування.

```
#include <iostream>
using namespace std;
int main() {
   int a = 10 + 20;
   cout << a;
   return 0;
}
```

Вихід: 30

Пояснення: Тут '+' – це оператор додавання, який виконує додавання операндів 10 та 20 і в результаті повертає значення 30.

Оператори C++ класифікуються на 6 типів залежно від операцій, які вони виконують.

3.2.1. Арифметичні оператори

Арифметичні оператори використовуються для виконання арифметичних або математичних операцій над операндами. Наприклад, '+' використовується для додавання.

1. Додавання + додає два операнди.
2. Віднімання - віднімає другий операнд від першого.

3. Множення * множить два операнди.
4. Ділення / ділить перший операнд на другий операнд.
5. Операція по модулю % повертає залишок від цілочисельного ділення.
6. Інкремент ++ збільшує значення операнда на 1.
7. Зменшення -- зменшує значення операнда на 1.

```
#include <iostream>

using namespace std;

int main() {

   int a = 8, b = 3;

   // Addition
   cout << "a + b = " << (a + b) << endl;
   // Subtraction
   cout << "a - b = " << (a - b) << endl;
   // Multiplication
   cout << "a * b = " << (a * b) << endl;
   // Division
   cout << "a / b = " << (a / b) << endl;
   // Modulo
   cout << "a % b = " << (a % b) << endl;
   // Increament
   cout << "++a = " << ++a << endl;
   // Decrement
   cout << "b-- = " << b--;

   return 0;
}
```

Вихід:

```
a + b = 11
a - b = 5
a * b = 24
a / b = 2
a % b = 2
++a = 9
--b = 2
```

Важливі моменти:

- Оператор модуля (%) слід використовувати лише з цілими числами. Інші оператори також можна використовувати зі значеннями з плаваючою комою.
- `++a` та `a++` є операторами інкременту, проте обидва дещо відрізняються. В `++a` значення змінної спочатку збільшується, а потім використовується в програмі. В `a++` значення змінної спочатку присвоюється, а потім збільшується. Аналогічно відбувається з оператором декременту.

Ви могли помітити, що деякі оператори працюють з двома операндами, а інші – з одним. На основі цього оператори також класифікуються як:

- унарний: працює з одним операндом;
- бінарний: працює з двома операндами;
- тернарний: працює з трьома операндами.

3.2.2. Реляційні оператори

Реляційні оператори використовуються для порівняння значень двох Операндів

- дорівнює `==` перевіряє, чи обидва операнди рівні;
- більше ніж `>` перевіряє, чи перший операнд більший за другий операнд;
- більше або дорівнює `>=` перевіряє, чи перший операнд більший або дорівнює другому операнду;
- менше ніж `<` перевіряє, чи перший операнд менший за другий операнд;
- менше або дорівнює `<=` перевіряє, чи перший операнд менший або дорівнює другому операнду;
- не дорівнює `!=` перевіряє, чи обидва операнди не рівні.

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

int main() {

   int a = 6, b = 4;
```

```
    // Equal operator
    cout << "a == b is " << (a == b) << endl;
    // Greater than operator
    cout << "a > b is " << (a > b) << endl;
    // Greater than Equal to operator
    cout << "a >= b is " << (a >= b) << endl;
    //  Lesser than operator
    cout << "a < b is " << (a < b) << endl;
    // Lesser than Equal to operator
    cout << "a <= b is " << (a <= b) << endl;
    // Not equal to operator
    cout << "a != b is " << (a != b);

    return 0;
}
```

Вихід:

```
a == b is 0
a > b is 1
a >= b is 1
a < b is 0
a <= b is 0
a != b is 1
```

Примітка: 0 означає хибність, а 1 – істину.

3.2.3. Логічні оператори

Логічні оператори використовуються для об'єднання двох або більше умов чи обмежень або для доповнення оцінки початкової умови, що розглядається. Результат повертає логічне значення, тобто «істина» або «хибність»:

- логічне І, &&, повертає значення true, лише якщо всі операнди істинні або не дорівнюють нулю;
- логічне АБО, ||, повертає значення «true», якщо будь-який з операндів є «true» або не дорівнює нулю;
- логічне НЕ, !, повертає true, якщо операнд дорівнює false або нулю.

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

int main() {
```

```
    int a = 6, b = 4;

    // Logical AND operator
    cout << "a && b is " << (a && b) << endl;
    // Logical OR operator
    cout << "a || b is " << (a || b) << endl;
    // Logical NOT operator
    cout << "!b is " << (!b);

    return 0;
}
```

Вихід:

```
a && b is 1
a || b is 1
!b is 0
```

3.2.4. Побітові оператори

Побітові оператори працюють на бітовому рівні. Отже, компілятор спочатку перетворює дані на бітовий рівень, а потім виконує обчислення над операндами:

- бінарне І, &, копіює біт у результат, якщо він існує в обох операндах;
- бінарне АБО, |, копіює біт у результат, якщо він існує в будь-якому з операндів;
- бінарне XOR, ^, копіює біт у результат, якщо він присутній в одному з операндів, але не в обох;
- лівий зсув, <<, зсуває значення ліворуч на кількість бітів, задану правим операндом;
- зсув праворуч, >>, зсуває значення праворуч на кількість бітів, задану правим операндом;
- доповнення до одиниці, ~, перетворює двійкові цифри 1 на 0 та 0 на 1.

Примітка: з побітовими операторами можна використовувати лише типи даних char та int.

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

```
int main() {

    int a = 6, b = 4;
    // Binary AND operator
    cout << "a & b is " << (a & b) << endl;
    // Binary OR operator
    cout << "a | b is " << (a | b) << endl;
    // Binary XOR operator
    cout << "a ^ b is " << (a ^ b) << endl;
    // Left Shift operator
    cout << "a << 1 is " << (a << 1) << endl;
    // Right Shift operator
    cout << "a >> 1 is " << (a >> 1) << endl;
    // One's Complement operator
    cout << "~(a) is " << ~(a);

    return 0;
}
```

Вихід:

```
a & b is 4
a | b is 6
a ^ b is 2
a<<1 is 12
a>>1 is 3
~(a) is -7
```

3.2.5. Оператори присвоєння

Оператори присвоєння використовуються для присвоєння значення змінній. Ми присвоюємо значення правого операнда лівому операнду відповідно до того, який оператор присвоєння ми використовуємо:

- присвоєння, =, присвоює значення праворуч змінній ліворуч;
- додавання та присвоєння, +=, додає правий операнд до лівого операнда та присвоює результат лівому операнду;
- віднімання та присвоєння, -=, віднімає правий операнд від лівого операнда та присвоює результат лівому операнду;

• множення та присвоєння, *=, множить лівий операнд на правий операнд і присвоює результат лівому операнду;

• ділення та присвоєння, /=, ділить лівий операнд на правий операнд та присвоює результат лівому операнду.

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

int main() {

   int a = 6, b = 4;
   // Assignment Operator.
   cout << "a = " << a << endl;
   //  Add and Assignment Operator.
   cout << "a += b is " << (a += b) << endl;
   // Subtract and Assignment Operator.
   cout << "a -= b is " << (a -= b) << endl;
   //  Multiply and Assignment Operator.
   cout << "a *= b is " << (a *= b) << endl;
   //  Divide and Assignment Operator.
   cout << "a /= b is " << (a /= b);

   return 0;
}
```

Вихід:

```
a = 6
a += b is 10
a -= b is 6
a *= b is 24
a /= b is 6
```

3.2.6. Тернарні або умовні оператори

Умовний оператор повертає значення на основі умови. Цей оператор приймає три операнди, тому він відомий як тернарний оператор.

Синтаксис:

Вираз 1 ? Вираз 2 : Вираз 3

У вищезазначеному твердженні:

• тернарний оператор ? визначає відповідь на основі обчислення Виразу1;

- якщо Вираз1 має значення true, тоді обчислюється Вираз2;
- якщо Вираз1 має значення false, то обчислюється Вираз3.

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

int main() {

   int a = 3, b = 4;
   // Conditional Operator
   int result = (a < b) ? b : a;
   cout << "The greatest number "
      "is " << result;

   return 0;
}
```

Вихід: The greatest number is 4

3.2.7. Інші оператори

Окрім наведених операторів, є декілька інших, які не підпадають під жодну з вищезазначених категорій.

- Оператор sizeof – це унарний оператор, який використовується для обчислення розміру свого операнда або змінної в байтах. Наприклад,

```
sizeof(char);
sizeof(var_name);
```

- Оператор коми (,) – це двійковий оператор, який використовується для різних цілей. Він використовується як роздільник або для обчислення першого операнда та відкидання результату; потім він обчислює другий операнд і повертає це значення (і тип):

```
int n = (m + 1, m - 2, m + 5);
int a, b, c;
```

- Оператора адресації (&) використовується для знаходження адреси пам’яті, в якій зберігається певна змінна. У C++ він також використовується для створення посилання:

```
&var_name;
```

- Оператор крапки (.) використовується для доступу до членів структурних змінних або об’єктів класу за допомогою їхніх імен об’єктів:

```
obj.member;
```

- Оператор стрілки (->) використовується для доступу до змінних класів або структур через його вказівник:

```
sptr->member;
```

Оператори кастингу (приведення типів) використовуються для перетворення значення одного типу даних на інший тип даних. Наприклад, для цілочисельного значення x:

```
(float)x;
static_cast<float>(x);
```

3.2.8. Пріоритетність та асоціативність

Пріоритетність та асоціативність операторів відіграють важливу роль. Коли в одному виразі є декілька операторів, пріоритет операторів та асоціативність визначають, у якому порядку та яка частина виразу обчислюється. Пріоритет вказує, яку частину виразу слід обчислювати першою. Це тому, що множення (*) повинне мати вищий пріоритет, ніж додавання (+).

Асоціативність вказує, в якому напрямку виконувати обчислення, коли у виразі присутні однакові оператори пріоритету. Асоціативність операторів означає, що якщо вираз має більше одного оператора з однаковим пріоритетом, то обчислення відбувається справа наліво або зліва направо.

4. Завдання

Завдання №1

Виведіть на екран всі парні числа, кратні п’яти, в інтервалі від 2 до 100 (включно).

Завдання №2

Сформуйте функцію int f(int h, int m, int s), яка приймає три цілих аргументи (години h, хвилини m і секунди s) і повертає кількість секунд, які пройшли з початку дня.

Завдання №3

Дано ціле число в діапазоні від 1 до 365. Визначте, який день тижня випадає на це число, якщо 1 січня – це понеділок.

Таблиця 1

Завдання згідно варіанту

№ варіанту	Завдання
1	Сформуйте функцію int f(int h, int m, int s), яка повертає кількість секунд, що залишились до кінця дня.
2	Виведіть усі парні числа від 50 до 150, сума цифр яких менша за 10.
3	Дано номер дня року. Визначте, який це буде день тижня через 100 днів.
4	Напишіть функцію int totalSeconds(int d, int h, int m, int s), яка повертає кількість секунд, що минула з початку місяця (вважаючи, що в кожному дні 24 години).
5	Виведіть усі числа, кратні 9, у межах від 1 до 500.
6	Напишіть функцію int secondsToMidnight(int h, int m, int s), яка обчислює кількість секунд до 00:00 наступного дня.
7	Виведіть усі числа, кратні 7, у межах від 10 до 200.
8	Дано день року. Визначте, чи належить він до першої, другої, третьої або четвертої пори року.
9	Напишіть функцію void showTime(int s), яка виводить час у форматі `год:хв:сек`, якщо задана кількість секунд від початку дня.
10	Дано номер дня (1–365). Визначте, чи це вихідний день (субота або неділя), якщо 1 січня – понеділок.
11	Напишіть функцію double hoursPassed(int h, int m), яка повертає кількість годин, що минула від початку дня (з дробовою частиною).
12	Виведіть усі числа, кратні 4 і непарні, з інтервалу 5–75.
13	Напишіть функцію bool isMorning(int h), яка повертає true, якщо час належить до ранку (з 6:00 до 12:00).
14	Дано день року. Визначте, скільки днів залишилося до

	найближчої неділі.
15	Знайдіть усі двозначні числа, які діляться на 3, але не діляться на 6.
16	Напишіть функцію int toMinutes(int h, int m, int s), яка повертає загальну кількість хвилин від початку дня.
17	Дано день року (1–366). Визначте день тижня, якщо 1 січня – неділя.
18	Виведіть усі непарні числа, кратні трьом, у діапазоні від 1 до 99.
19	Напишіть функцію int toHours(int m, int s), яка обчислює кількість повних годин за заданими хвилинами та секундами.
20	Виведіть усі числа, які одночасно діляться на 2 та 3, у межах від 1 до 60.
21	Напишіть функцію int halfOfDay(int h, int m, int s), яка визначає, чи поточний момент належить першій чи другій половині доби (повертає 1 або 2).
22	Виведіть усі числа від 100 до 300, які при діленні на 11 дають остачу 5.
23	Дано два номери днів року. Визначте, скільки між ними повних тижнів.
24	Напишіть функцію int secondsBetween(int h1, int m1, int s1, int h2, int m2, int s2), що обчислює кількість секунд між двома моментами часу.
25	Виведіть усі числа з проміжку 10–200, які мають остачу 2 при діленні на 7.
26	Дано номер тижня (1–52) та день тижня (1–7). Виведіть номер дня року.
27	Виведіть усі числа від 1 до 100, які не діляться на 5.
28	Дано номер дня (1–365). Визначте день тижня, якщо 1 січня – п’ятниця.
29	Напишіть функцію int f(int h, int m, int s), яка повертає кількість секунд, що пройшли з початку дня.
30	Напишіть програму, яка для заданого дня року виводить, чи це парний чи непарний тиждень.

5. Рекомендації щодо усунення типових помилок

- Помилка типів даних: використовуйте відповідні типи для обчислень. Наприклад, для операцій з часом

використовуйте int, для ділення – double, щоб уникнути втрати дробової частини.

- Відсутність приведення типів: якщо змішуєте int і float, виконуйте явне приведення:

```
double r = (double)a / b;
```

- Невірне використання операторів порівняння: не плутайте = (присвоєння) з == (порівняння).
- Помилки при роботі з діапазонами: завжди враховуйте включення або виключення меж інтервалу. Наприклад,

```
for (int i = 2; i <= 100; i++)
```

- Помилки у пріоритеті операторів: використовуйте дужки для контролю порядку обчислень. Наприклад:

```
if ((a % 2 == 0) && (a % 5 == 0))
printf("%d\n", a);
```

- Помилки у функціях: завжди вказуйте тип повернення (int, void тощо). Пам'ятайте, що повернене значення можна зберігати у змінну або одразу використовувати у виразі.
- Неправильне використання операторів логіки: не використовуйте & замість && і | замість ||, якщо потрібна логічна операція, а не побітова.
- Помилки у розрахунку секунд/хвилин: формула має бути узгодженою:

```
total_seconds = h * 3600 + m * 60 + s;
```

- Проблеми з індексацією днів тижня: використовуйте залишок від ділення (% 7) для переходу між днями тижня.

6. Додаткові матеріали

Для набуття практичних навичок використовувати матеріали за наступними посиланнями:

1. https://www.geeksforgeeks.org/cpp/operators-in-cpp/
2. https://www.geeksforgeeks.org/cpp/cpp-data-types/
3. https://courses.washington.edu/css342/zander/css332/datatypes.html

Лабораторна робота №7

Тема: Робота з рядками

1. Мета лабораторної роботи

Набути практичних навичок обробки рядків у програмі: створення, введення та виведення рядків, визначення їх довжини, копіювання, з'єднання, пошуку та заміни символів і підрядків. Ознайомитись із базовими функціями роботи з рядками, особливостями представлення рядків у пам'яті та способами їх перетворення.

2. Навчальні результати

Після виконання лабораторної роботи студент буде:

- знати:
  - поняття рядка, символьного масиву та нуль-термінатора ('\0');
  - основні функції стандартних бібліотек для роботи з рядками (наприклад, strlen, strcpy, strcmp, strcat, strchr, strstr);
  - принципи пошуку, заміни та порівняння рядків;
  - відмінності між рядками мов C, C++, Python, Java тощо;
- уміти:
  - створювати, зберігати та змінювати рядки;
  - визначати довжину рядка та виконувати базові операції над ним;
  - використовувати бібліотечні функції або власні алгоритми для обробки тексту;
  - знаходити символи або підрядки у рядку;
  - здійснювати перетворення символів і рядків (наприклад, у верхній/нижній регістр).

3. Теоретичні відомості

3.1. Поняття рядка

Рядок – це послідовність символів, що закінчується нуль-символом '\0'. У мовах програмування:

- C: рядок – це масив типу char

```
char s[20] = "Hello";
```

- C++: рядки можуть бути як char[], так і об'єктами класу std::string

```
std::string s = "Hello";
```

3.2. Основні операції над рядками

Таблиця 2

Основні операції над рядками

Операція	Приклад	Опис
Визначення довжини	strlen(s)	Повертає кількість символів без '\0'.
Копіювання	strcpy(dest, src)	Копіює рядок src у dest.
Конкатенація	strcat(s1, s2)	Додає s2 до кінця s1.
Порівняння	strcmp(s1, s2)	Повертає 0, якщо рядки рівні.
Пошук символу	strchr(s, ch)	Повертає вказівник на перше входження символу.
Пошук підрядка	strstr(s, sub)	Повертає адресу першого входження підрядка.

3.3. Перетворення символів

Для зміни регістру або перевірки типу символу використовують функції:

```
toupper(ch);  // у верхній регістр
tolower(ch);  // у нижній регістр
isalpha(ch);  // перевірка, чи літера
isdigit(ch);  // перевірка, чи цифра
isspace(ch);  // перевірка на пробіл
```

3.4. Приклади операцій над рядками

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

```
int main() {

  char s1[50] = "Hello";
  char s2[50] = "World";
  char result[100];

  // З'єднання
  strcpy(result, s1);
  strcat(result, " ");
  strcat(result, s2);
  printf("Результат: %s\n", result);

  // Підрахунок довжини
  printf("Довжина рядка: %lu\n", strlen(result));

  // Перетворення у верхній регістр
  for (int i = 0; i < strlen(result); i++)
     result[i] = toupper(result[i]);
  printf("Верхній регістр: %s\n", result);

  return 0;
}
```

4. Допоміжні конструкції

- Оголошення рядка:

```
char str[100];
std::string text;
```

- Введення/виведення:

```
gets(str);            // зчитування (небезпечне)
fgets(str, 100, stdin); // безпечне
puts(str);            // виведення
```

- Операції з рядками (C++)

```
string s1 = "Hello";
string s2 = "world";
string s3 = s1 + " " + s2;  // конкатенація
int len = s3.length();
if (s1 == "Hello") cout << "Так";
```

5. Завдання

Завдання №1

Напишіть програму, яка обчислює середнє арифметичне значення послідовності дробів, які вводяться з клавіатури. Після введення користувачем останнього числа програма повинна вивести мінімальне і максимальне числа з послідовності. Кількість чисел послідовності вводить користувач.

Завдання №2

1. Створити програму, що просить вести ім'я та прізвище через пробіл типу string та виведіть введений текст на екран.
2. Створіть програму, яка окремо вводить з клавіатури прізвище та ім'я, записує їх в різні змінні та об'єднує їх оператором +
3. Визначити довжину об'єднаного рядка.

Завдання №3

Таблиця 3

Завдання згідно варіанту

№ варіанту	Завдання
1	Написати програму, яка підраховує кількість голосних і приголосних у введеному рядку.
2	Створити програму, що перевіряє, чи є введене слово паліндромом.
3	Користувач вводить рядок. Вивести рядок без пробілів.
4	Написати програму, яка рахує кількість входжень певного символу у рядку. Символ задає користувач.
5	Ввести рядок і замінити всі пробіли на символ підкреслення.
6	Створити програму, що зчитує повне ім'я користувача і виводить ініціали (наприклад, «Іванов І.П.»).
7	Ввести рядок і вивести його у зворотному порядку.
8	Дано рядок. Вивести кількість слів у ньому.
9	Ввести речення. Замінити усі великі літери на малі, а малі – на великі.
10	Написати програму, яка визначає кількість цифр у рядку.

11	Ввести два рядки. Перевірити, чи один рядок є підрядком іншого.
12	Користувач вводить текст. Порахувати, скільки разів у ньому зустрічається слово «і».
13	Ввести речення і вивести перше та останнє слово окремо.
14	Написати програму, яка видаляє з рядка всі цифри.
15	Користувач вводить рядок. Перевірити, чи починається він зі слова «Hello».
16	Ввести рядок і підрахувати кількість слів, довших за 5 символів.
17	Дано речення. Вивести слова у зворотному порядку.
18	Написати програму, яка замінює всі голосні літери у рядку символом *.
19	Створити програму, що визначає, чи два введених рядки однакові без урахування регістру.
20	Написати програму, яка видаляє всі пробіли на початку та в кінці рядка.
21	Користувач вводить рядок. Знайти найдовше слово в цьому рядку.
22	Написати програму, яка виводить усі слова рядка, що починаються на літеру, введену користувачем.
23	Дано рядок, що містить дату у форматі «дд.мм.рррр». Вивести день, місяць і рік окремо.
24	Написати програму, яка з’єднує список слів у єдиний рядок, розділяючи їх комами.
25	Ввести текст і порахувати кількість речень (речення вважаються розділеними крапками).
26	Написати програму, яка видаляє всі повторювані пробіли у рядку.
27	Створити програму, що виводить позиції (індекси) усіх входжень заданої літери у рядку.
28	Ввести рядок і вивести тільки ті символи, що стоять на парних позиціях.
29	Дано речення. Перетворити першу літеру кожного слова на велику.
30	Ввести рядок. Замінити у ньому всі входження підрядка «abc» на «xyz».

6. Рекомендації щодо усунення типових помилок

- Некоректне введення рядка з пробілами:
  - помилка: використання cin >> str; замість getline(cin, str); призводить до зчитування лише першого слова;
  - виправлення: використовуйте:

```
string str;
getline(cin, str);
```

  щоб зчитати весь рядок, включно з пробілами.
- Зайвий залишок символів у буфері після cin:
  - помилка: після зчитування чисел через cin програма пропускає введення рядка;
  - виправлення: після cin >> потрібно додати:

```
cin.ignore();
```

  щоб очистити буфер вводу.
- Плутанина між символами (char) і рядками (string):
  - помилка: спроба порівнювати string із char напряму;
  - виправлення: якщо потрібно порівняти окремий символ рядка – використовуйте індексацію:

```
if (str[i] == 'a') ...
```

- Вихід за межі рядка:
  - помилка: доступ до елемента str[i], коли i >= str.length();
  - виправлення: завжди перевіряйте межі циклів і використовуйте метод str.size() для безпечного обходу.
- Неправильне порівняння рядків:
  - помилка: використання == для порівняння вмісту, якщо типи відрізняються (наприклад, char* і string);
  - виправлення: для об’єктів string порівняння через == допустиме, але для С-рядків потрібно використовувати strcmp().

- Ігнорування регістру при порівнянні: якщо потрібно порівняти слова без врахування регістру, приведіть обидва рядки до нижнього регістру:

```
transform(str.begin(), str.end(),
                                    str.begin(), ::tolower);
```

- Проблеми з юнікодними або кириличними символами. Якщо кириличні літери відображаються некоректно, додайте на початку програми:

```
setlocale(LC_ALL, "");
```

- Помилки при розрахунку кількості слів:
  - помилка: неправильне розділення слів, якщо між ними кілька пробілів;
  - виправлення: використовуйте istringstream:

```
istringstream ss(str);
while (ss >> word) count++;
```

- Неправильна робота з рядками у циклах:
  - помилка: зміна довжини рядка під час ітерації (наприклад, видалення символів у середині циклу);
  - виправлення: спочатку сформуйте новий рядок, не змінюючи поточний.
- Відсутність перевірки порожнього рядка:
  - помилка: якщо користувач нічого не ввів, виклик str[0] або пошук може викликати помилку;
  - виправлення: завжди перевіряйте:

```
if (str.empty()) cout << "Рядок порожній!";
```

- Використання кирилиці без налаштування кодування в IDE: якщо виведення українських букв відбувається некоректно, переконайтесь, що в проєкті вибрано UTF-8 або Windows-1251.
- Помилка при злитті рядків з числами:
  - помилка: при спробі додати число до рядка напряму виникає помилка типів;
  - виправлення: перетворіть число у рядок:

```
str = str + to_string(number);
```

- Використання індексації без перевірки символу: якщо шукаєте певний символ – використовуйте метод find():

```
int pos = str.find('a');
if (pos != string::npos) ...
```

- Забування про кінець рядка при виведенні: додавайте endl або '\n' для зручного форматування виводу:

```
cout << "Результат: " << result << endl;
```

- Неправильне використання методів substr, erase, replace. Пам'ятайте:
  - substr(pos, len) – починає з позиції pos і бере len символів.
  - erase(pos, len) – видаляє частину рядка, replace(pos, len, str2) – замінює фрагмент.

7. Додаткові ресурси

1. https://www.w3schools.com/cpp/cpp_strings.asp
2. https://www.programiz.com/cpp-programming/strings
3. https://www.tutorialspoint.com/cplusplus/cpp_strings.htm

Лабораторна робота №8

Тема: Використання циклів, розгалужень, перерахувань (enum) та структур

1. Мета лабораторної роботи

Ознайомитись із застосуванням операторів циклів та розгалужень, вивчити принципи створення та використання перерахувань (enum) і структур (struct) у мові програмування. Закріпити навички побудови програм із комбінуванням різних типів даних та керуючих конструкцій.

2. Навчальні результати

Після виконання лабораторної роботи студент буде:

- знати:
  - синтаксис та особливості роботи циклів for, while, do-while;
  - призначення операторів керування циклом break та continue;
  - відмінності між структурами (struct) та класами, а також принципи їх оголошення;
  - різницю між звичайними перерахуваннями (enum) та перерахуваннями з областю видимості (enum class);
  - правила вкладеності структур та використання структур у масивах;
- вміти:
  - обирати найбільш доцільний тип циклу для вирішення конкретної задачі;

- створювати користувацькі типи даних за допомогою struct для групування логічно пов’язаних змінних;
- використовувати enum для підвищення читабельності коду (замість магічних чисел);
- звертатися до полів структури та змінювати їх значення;
- реалізовувати меню програми за допомогою конструкції switch-case.

3. Теоретичні відомості

3.1. Оператори розгалуження

Розгалуження дозволяють змінювати послідовність виконання програми залежно від виконання певної умови. Основні конструкції:

- if, if...else, if...else if...else
- switch

Приклад:

```
if (x > 0)
   cout << "Додатне число";
else if (x < 0)
   cout << "Від'ємне число";
else
   cout << "Нуль";
```

3.2. Оператори циклів

Цикли дозволяють виконувати набір команд декілька разів:

- for – коли відома кількість повторень;
- while – коли умова відома заздалегідь;
- do...while – коли цикл має виконатись хоча б один раз.

Приклад:

```
for (int i = 1; i <= 5; i++)
   cout << i << " ";
```

3.3. Перерахування (enum)

Перерахування – це користувацький тип даних, який дозволяє задавати набір іменованих констант, що підвищують читабельність коду. Приклад:

```
enum Day { Mon, Tue, Wed, Thu, Fri, Sat, Sun };
Day today = Wed;
```

3.4. Структури (struct)

Структура – це складений тип даних, який об’єднує змінні різних типів у єдину логічну одиницю. Приклад:

```
struct Student {
   string name;
   int age;
   double grade;
};
```

Використання:

```
Student s1 = {"Іван", 19, 91.5};
cout << s1.name << " має середній бал " << s1.grade;
```

3.5. Приклад виконання

Приклад 1. Програма визначає середній бал студентів і класифікує їх за рівнем успішності:

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

enum GradeLevel { Low, Medium, High };

struct Student {
   string name;
   double average;
   GradeLevel level;
};

int main() {
   const int N = 3;
   Student group[N];

   for (int i = 0; i < N; i++) {
      cout << "Введіть ім'я студента: ";
         cin >> group[i].name;
```

```
        cout << "Введіть середній бал : ";
        cin >> group[i].average;

        if (group[i].average < 60)
            group[i].level = Low;
        else if (group[i].average < 85)
            group[i].level = Medium;
        else
            group[i].level = High;
    }

    cout << "\nРезультати:\n";
    for (int i = 0; i < N; i++) {
        cout << group[i].name << " – ";
        switch (group[i].level) {
        case Low: cout << "низький рівень\n";
            break;
        case Medium: cout << "середній рівень\n";
            break;
        case High: cout << "високий рівень\n";
            break;
        }
    }
    return 0;
}
```

4. Завдання

Завдання №1

1. Оголосити власне перерахування, визначити декілька змінних перелічуваного типу, вивести на екран позицію одного з перерахувань.

2. Написати функцію з використанням if-стейтментів для виводу енумераторів.

3. Створити перерахування з областю видимості.

4. Оголосити змінні типу int та double змінивши псевдоніми з використанням ключового слова typedef та type alias.

5. Оголосити та ініціалізувати структуру з 5-ти екземплярів та вивести на екран за допомогою функції.

Завдання №2

Напишіть програму, яка реалізує гру “Вгадай число”. Комп’ютер загадує число від 0 до 999 (використовуйте генерацію випадкових чисел), а гравець вгадує це число. На кожному кроці гравець робить припущення, а комп’ютер повідомляє, скільки цифр з числа вгадані і скільки з вгаданих цифр займають правильні позиції в числі. Наприклад, якщо загадано число 725 і висунуто припущення, що загадане число 523, то вгадані дві цифри (5 і 2), і одна з них займає вірну позицію.

Наприклад, комп’ютер загадав тризначне число. Ви повинні його відгадати. Після чергового числа Вам буде повідомлено, скільки цифр вгадано і скільки з них знаходиться на своїх місцях:

```
Ваш варіант: 123
Вгадано: 0. На своїх місцях: 0
Ваш варіант: 456
Вгадано: 1. На своїх місцях: 1
Ваш варіант: 654
Вгадано: 2. На своїх місцях: 2
Ваш варіант: 657
Вгадано: 2. На своїх місцях: 2
Ваш варіант: 658
Вгадано: 3. На своїх місцях: 3
***Ви вгадали число 658!***
```

Таблиця 4

Завдання згідно варіанту

№ варіанту	Завдання
1	1. Створити enum для днів тижня. Вивести день із найвищим індексом. 2. Додати лічильник спроб і повідомлення про кількість залишених спроб.
2	1. Створити enum для пір року. Написати функцію, що

	повертає назву пори року за номером. 2. Додати можливість повторного запуску гри після перемоги чи поразки.
3	1. Оголосити enum class для рівнів доступу (Admin, User, Guest). Вивести права доступу. 2. Дозволити гравцеві самому задавати межі числа.
4	1. Написати функцію, що приймає enum і через switch виводить опис елемента. 2. Додати рівні складності гри: легкий (2 цифри), середній (3), важкий (4).
5	1. Створити typedef для unsigned int як uint і використати його в структурі “Студент”. 2. Додати підказку – чи більше/менше число за введене.
6	1. Створити type alias для std::string як Text. Використати для імені користувача. 2. Додати обробку неправильного вводу (нечислове значення).
7	1. Створити структуру Book (назва, автор, рік, ціна). Вивести найдешевшу книгу. 2. Зберігати рекорд гравця (мінімальна кількість спроб).
8	1. Створити структуру Car (марка, рік, пробіг). Вивести найновіше авто. 2. Реалізувати вибір мови гри (українська / англійська).
9	1. Створити enum class для кольорів світлофора. Функція має повертати дію (“Йти”, “Чекати”). 2. Вивести історію попередніх спроб із результатами.
10	1. Створити enum для місяців року. Через switch вивести кількість днів у місяці. 2. Додати режим для кількох гравців (по черзі відгадують число).
11	1. Створити typedef для double як Decimal. Вивести значення ціни. 2. Реалізувати гру, де комп’ютер вгадує число користувача.
12	1. Створити enum class для оцінок (A, B, C, D, F). Функція повертає опис оцінки. 2. Додати кольоровий вивід підказок (через ANSI-коди).
13	1. Створити структуру Person (ім’я, вік, стать). Вивести дані про всіх осіб. 2. Порахувати кількість відгаданих цифр без урахування повторів.

14	1. Створити enum class для типів файлів (TXT, PDF, DOCX, MP3). Вивести розширення. 2. Обмежити час на введення відповіді (наприклад, 10 секунд).
15	1. Використати typedef для створення псевдоніма структури. Створити 5 екземплярів і вивести їх. 2. Зробити 3 рівні підказок: кількість цифр, позиції, “гаряче-холодно”.
16	1. Створити enum для рівнів сигналу Wi-Fi (Low, Medium, High). Вивести рівень за номером. 2. Виводити повідомлення після кожної спроби – “Тепліше” чи “Холодніше”.
17	1. Створити структуру City (назва, населення, площа). Визначити місто з найбільшою густотою. 2. Додати підказку про кількість правильних позицій у відсотках.
18	1. Створити enum class для музичних жанрів (Rock, Pop, Jazz, Classical). Вивести назву. 2. Дозволити користувачу вибрати кількість цифр у числі.
19	1. Створити структуру Student (ПІБ, курс, середній бал). Вивести студента з найвищим балом. 2. Зберігати усі спроби в масив і показати після завершення гри.
20	1. Створити enum class для типів пристроїв (Laptop, Tablet, Phone, PC). Вивести опис типу. 2. Реалізувати логіку “гаряче-холодно” на основі близькості чисел.
21	1. Створити typedef для long long як BigInt. Вивести приклад використання. 2. Додати ліміт у 10 спроб. Якщо не вгадано – показати правильне число.
22	1. Створити enum для розмірів файлу (Small, Medium, Large, Huge). Вивести розмір за індексом. 2. Виводити повідомлення про час, витрачений на гру.
23	1. Створити структуру Computer (процесор, RAM, SSD). Вивести характеристики. 2. Після перемоги запропонувати гравцю “зіграти знову або вийти”.
24	1. Створити enum class для станів системи (Loading, Running, Error, Finished).

	2. Додати можливість зберігати статистику в файл.
25	1. Створити структуру Animal (вид, вага, вік). Знайти найважчу тварину. 2. Додати підказку, якщо число має однакові цифри (наприклад, “усі різні” або “є повтори”).
26	1. Створити typedef для float як Real. Використати для обчислень площі. 2. Зробити підрахунок середнього часу на одну спробу.
27	1. Створити структуру Movie (назва, рік, рейтинг). Вивести фільм із найвищим рейтингом. 2. Додати вивід “ви близько” або “далеко” залежно від різниці чисел.
28	1. Створити enum class для типів операцій (Add, Sub, Mul, Div). Реалізувати функцію обчислення. 2. Додати режим автоматичного завершення після 5 невдалих спроб.
29	1. Створити enum для днів тижня з власною функцією друку вихідних. 2. Додати режим, коли гра підказує лише кількість вгаданих цифр без позицій.
30	1. Створити структуру Employee (ім’я, посада, зарплата). Вивести працівника з найвищою зарплатою. 2. Додати “інтелектуальний режим”, коли підказки стають точнішими з кожною спробою.

5. Рекомендації щодо усунення типових помилок

- Помилки при використанні циклів.
  1. Нескінченні цикли:
     - причина: умова виходу з циклу ніколи не стає хибною;
     - як уникнути: уважно перевіряйте змінні-лічильники та умови порівняння.

     ```
     for (int i = 0; i < 10; i++) // i++ обов’язково!
     ```
  2. Використання невірної межі діапазону:
     - типова помилка: for (int i = 0; i <= n; i++) замість i < n;
     - перевірте, чи включає цикл останній елемент.

3. Зміна лічильника всередині циклу: не змінюйте змінну циклу (i++) додатково всередині тіла без потреби.

4. Відсутність ініціалізації змінних перед циклом: завжди задавайте початкове значення змінним, що використовуються у виразах циклу.

- Помилки у розгалуженнях (if, else if, switch).

1. Плутанина з операторами = і ==
   - = – це присвоєння, == – порівняння;
   - if (a = 5) – завжди істинно;
   - if (a == 5)

2. Відсутність фігурних дужок {} – якщо у блоці кілька операторів, завжди використовуйте {}.

3. Відсутній break у switch
   - без break відбудеться “провалювання” в наступні гілки.

```
switch (x) {
   case 1: ...; break;
   case 2: ...; break;
}
```

4. Неврахування гілки default: для безпечності завжди додавайте default: навіть якщо він порожній.

- Помилки при використанні enum та enum class.

1. Змішування типів enum і int без явного приведення. enum class вимагає явного приведення типів:

```
enum class Day { Mon, Tue, Wed };
int x = static_cast<int>(Day::Tue);
```

2. Використання одного імені для різних елементів переліку: у звичайному enum імена мають бути унікальними в межах області видимості.

3. Відсутність описових назв у enum: використовуйте зрозумілі ідентифікатори (Winter, Spring, а не W, S).

- Помилки при роботі зі структурами.

1. Неініціалізовані поля структури: завжди задавайте початкові значення:

```
struct Point { int x = 0; int y = 0; };
```

2. Невірне передавання структур у функції: якщо структура велика, передавайте її за посиланням або константним посиланням:

```
void print(const Student & s);
```

3. Неправильне оголошення об'єктів структури: після оголошення структури необхідно вказувати тип:

```
Student a, b, c;  // а не просто a, b, c;
```

- Помилки при використанні typedef або using.

1. Плутанина між псевдонімом і новим типом: typedef не створює новий тип, лише альтернативну назву.

2. Неправильний синтаксис typedef. Правильно:

```
typedef unsigned int uint;
using real = double;
```

3. Використання псевдонімів у невідповідному контексті: не варто робити типові псевдоніми занадто короткими або неочевидними.

- Типові логічні помилки у грі "Вгадай число".

1. Невірна генерація випадкових чисел: завжди ініціалізуйте генератор випадкових чисел:

```
srand(time(0));
int n = rand() % 1000;
```

2. Порівняння чисел як рядків: якщо потрібно порівняти цифри – перетворіть число на рядок лише свідомо.

3. Неочищення буфера вводу: після cin >> перевіряйте стан потоку або очищуйте cin.ignore().

6. Додаткові ресурси

1. https://www.programiz.com/cpp-programming/examples/calculator-switch-case
2. https://www.w3schools.com/cpp/cpp_enum.asp
3. https://www.w3schools.com/cpp/cpp_structs.asp
4. https://www.programiz.com/cpp-programming/structure

Лабораторна робота №9

Тема: Робота з масивами

1. Мета лабораторної роботи

1. Закріпити навички оголошення та використання масивів.
2. Навчитися виконувати операції пошуку, сортування та обробки масивів.
3. Навчитися передавати масиви у функції та працювати з ними в процедурному стилі.

2. Навчальні результати

Після виконання лабораторної роботи студент буде:

- знати:
  - способи оголошення та ініціалізації статичних одновимірних масивів;
  - як розташовуються елементи масиву в пам’яті комп’ютера;
  - особливості індексації масивів у C++ (нумерація з нуля, вихід за межі);
  - базові алгоритми обробки масивів: лінійний пошук, знаходження мінімуму/максимуму;
  - принцип роботи простих алгоритмів сортування (наприклад, метод «бульбашки»);
- вміти:
  - оголошувати масиви заданого розміру та заповнювати їх даними (з клавіатури або випадковими числами);
  - виконувати обхід масиву за допомогою циклу for;
  - знаходити суму, середнє арифметичне та кількість елементів, що задовольняють умові;

- реалізовувати алгоритм перестановки елементів (swap) для сортування або інверсії масиву;
- відлагоджувати код для уникнення помилок виходу за межі масиву.

3. Теоретичні відомості

Масив – це структура даних, яка дозволяє зберігати фіксовану кількість елементів одного типу. Оголошення масиву:

```
int a[10]; // оголошення масиву з 10 елементів
int b[5] = { 1, 2, 3, 4, 5 }; // ініціалізований масив
```

Доступ до елементів:

```
a[0] = 25;    // запис
cout << b[2]; // читання
```

Введення масиву з клавіатури:

```
for (int i = 0; i < n; i++)
        cin >> a[i];
```

Передача масиву у функцію:

```
void print(int arr[], int size) {
        for (int i = 0; i < size; i++)
                cout << arr[i] << " ";
}
```

4. Допоміжні конструкції

Таблиця 5

Допоміжні конструкції

Операція	Приклад	Пояснення
Введення масиву	for (...) cin >> arr[i];	Вводимо елементи по одному
Виведення масиву	for (...) cout << arr[i];	Послідовний вивід
Пошук max	якщо arr[i] > max → max = arr[i]	Переглядаємо масив
Пошук min	аналогічно для min	
Середнє арифметичне	sum / n	де sum – сума елементів

Обмін елементів	temp = a;
	a = b;
	b = temp;
		Використовується в сортуванні

5. Завдання

Завдання №1

Напишіть програму, яка перевіряє, чи знаходиться введене з клавіатури число в масиві. Масив попередньо вводить користувач на початку виконання програми.

Завдання №2

Користувач вводить натуральне чотиризначне число. З'ясуйте, чи є воно паліндромом (читається однаково як зліва направо, так і справа наліво).

Приклад результату виконання програми:

```
Введіть число: 4884
4884 є паліндромом
```

Завдання №3

Напишіть програму, яка об'єднує два упорядкованих за зростанням масиви в один (теж упорядкований) масив.

Приклад результату виконання програми:

```
Введіть елементи першого масиву: 1 3 5 7 9
Введіть елементи другого масиву: 2 4 6 8 10
Масив-результат: 1 2 3 4 5 6 7 8 9 10
```

Завдання №4

Таблиця 6

Завдання згідно варіанту

Варіант	Завдання
1	(Завдання 3) Створити масив, який містить лише унікальні елементи першого масиву.
2	(Завдання 2) Визначити, чи є число простим.
3	(Завдання 1) Замінити всі від'ємні числа у масиві їх модулями.
4	(Завдання 3) Об'єднати масиви і знайти медіану отриманого масиву.
5	(Завдання 2) Перевірити, чи є число паліндромом (п'ятизначне).
6	(Завдання 1) Вивести індекс першого входження введеного числа в масив або повідомити, що його немає.
7	(Завдання 3) Об'єднати два масиви та впорядкувати отриманий масив за спаданням.

8	(Завдання 2) Обчислити суму цифр введеного числа.
9	(Завдання 1) Перевернути масив (вивести у зворотному порядку).
10	(Завдання 3) Виконати злиття двох масивів рекурсивно.
11	(Завдання 2) Перевірити, чи є число автоморфним (n^2 закінчується на його цифри).
12	(Завдання 1) Визначити, чи впорядкований масив за зростанням.
13	(Завдання 3) Об’єднати масиви так, щоб спочатку йшли парні, потім непарні елементи.
14	(Завдання 2) Обміняти місцями першу та останню цифри числа.
15	(Завдання 1) Порахувати кількість додатних, від’ємних і нульових елементів масиву.
16	(Завдання 3) Створити масив із елементів, що зустрічаються в обох масивах одночасно.
17	(Завдання 2) Визначити найбільшу цифру числа.
18	(Завдання 1) Знайти мінімальний і максимальний елементи масиву.
19	(Завдання 3) Об’єднати два масиви і видалити всі повторювані елементи.
20	(Завдання 2) Визначити, чи утворюють цифри числа арифметичну прогресію.
21	(Завдання 1) Видалити з масиву всі парні числа.
22	(Завдання 3) Заповнити масив випадковими числами і впорядкувати методом «бульбашки».
23	(Завдання 2) Визначити кількість парних і непарних цифр введеного числа.
24	(Завдання 1) Вивести всі елементи масиву, що більші за середнє значення.
25	(Завдання 3) Впорядкувати масив вибором (selection sort).
26	(Завдання 2) Визначити, чи є число степенем двійки.
27	(Завдання 1) Визначити другий за величиною елемент масиву.
28	(Завдання 3) Впорядкувати масив вставками (insertion sort).
29	(Завдання 2) Перевірити, чи є число «щасливим квитком» (сума перших 3 цифр = сумі останніх 3).
30	(Завдання 1) Написати програму, що визначає кількість входжень введеного числа в масив.

6. Рекомендації щодо усунення типових помилок

1. Індекс виходить за межі масиву. Насправді масив розміром n має індекси від 0 до n-1.

2. Неініціалізовані змінні min, max. Перед пошуком потрібно виконати інструкцію:

```
min = max = arr[0];
```

Неправильний розрахунок середнього. sum має бути типу double, інакше відбудеться цілочисельне ділення.

3. Переплутані умови в сортуванні. Для сортування по зростанню використовуйте:

```
if (arr[i] > arr[j]) swap(arr[i], arr[j]);
```

4. Не перевіряється правильність введення N. Насправді значення N має бути більшим, ніж 0.

7. Додаткові ресурси

1. https://www.w3schools.com/cpp/cpp_arrays.asp
2. https://cplusplus.com/doc/tutorial/arrays/
3. https://www.tutorialspoint.com/cplusplus/cpp_arrays.htm

Лабораторна робота №10

Тема: Робота з динамічними масивами

1. Мета лабораторної роботи

1. Ознайомитися з поняттям динамічних масивів.
2. Навчитися виділяти та звільняти пам’ять під масиви динамічно.
3. Розвинути навички обробки динамічних масивів (вводити, обчислювати, сортувати).
4. Навчитися передавати динамічні масиви у функції.

2. Навчальні результати

Після виконання лабораторної роботи студент буде:

- знати:
  - відмінність між статичною (Stack) та динамічною (Heap) пам’яттю;
  - синтаксис операторів new та delete для керування пам’яттю;
  - поняття вказівника та його роль у роботі з динамічними масивами;
  - життєвий цикл змінних у динамічній пам’яті та проблему витоку пам’яті (memory leaks);
  - переваги динамічних масивів порівняно зі статичними;
- вміти:
  - виділяти пам’ять під масив під час виконання програми залежно від вибору користувача;
  - коректно звільняти пам’ять після завершення роботи з масивом, щоб уникнути витоків;
  - працювати з динамічними масивами як із звичайними, використовуючи індексну арифметику;

- обнуляти вказівники (nullptr) після видалення масиву для безпеки;
- створювати та обробляти масиви, розмір яких невідомий на етапі компіляції.

3. Теоретичні відомості

Динамічний масив – масив, розмір якого визначається під час виконання програми. Пам’ять під масив виділяється у купі.

Оголошення та виділення пам’яті:

```
int n;
cin >> n;
int* arr = new int[n];  // виділення пам'яті
//Введення / виведення:
for (int i = 0; i < n; i++)
   cin >> arr[i];

for (int i = 0; i < n; i++)
   cout << arr[i] << " ";
//Звільнення пам'яті:
   delete[] arr;
arr = nullptr;  // безпечна практика
//Передача у функцію :
void printArray(int* arr, int n) {
   for (int i = 0; i < n; i++)
      cout << arr[i] << " ";
}
```

4. Допоміжні конструкції

Таблиця 7

Допоміжні конструкції

Операція	Приклад	Пояснення
Виділення пам’яті	int* arr = new int[n];	Виділяємо динамічний масив на n елементів
Звільнення пам’яті	delete[] arr;	Повертаємо пам’ять операційній системі
Доступ до елементів	arr[i]	Робота як зі звичайним масивом

Передача у функцію	void f(int* arr, int n)	Передаємо покажчик і розмір

Сортування	for (i = 0;i < n - 1;i++)
	for (j = i + 1;j < n;j++)
	if (arr[i] > arr[j])
	swap(arr[i], arr[j]);
		Простий алгоритм сортування

5. Завдання

Завдання №1

Цілочисельний масив заповнюється дев'ятьма випадковими елементами. Поміняйте місцями найбільший і найменший елементи масиву.

Завдання №2

У вказаному користувачем цілочисельному масиві видаліть елементи, які зустрічаються частіше, ніж 2 рази.

Завдання №3

Користувач вводить ціле число N. Створіть масив з N цілих чисел (числа генеруються випадковим чином). Визначте індекс найбільшого елемента масиву.

Завдання №4

Таблиця 8

Завдання згідно варіанту

Варіант	Завдання
1	(Завдання 1) Обміняти місцями перший і останній елементи динамічного масиву з 10 випадкових чисел.
2	(Завдання 2) Видалити з динамічного масиву всі числа, що повторюються більше одного разу.
3	(Завдання 3) Створити масив з N випадкових чисел та знайти індекс найменшого елемента.
4	(Завдання 1) Замінити максимальний елемент динамічного масиву на мінімальний.
5	(Завдання 2) Визначити і видалити всі парні елементи, що зустрічаються більше двох разів.
6	(Завдання 3) Створити масив із N елементів та визначити суму всіх елементів, що більші за середнє.
7	(Завдання 1) Поміняти місцями елементи з максимальним та мінімальним індексом у масиві.
8	(Завдання 2) Видалити з масиву всі нулі, які зустрічаються

	більше двох разів.
9	(Завдання 3) Знайти індекс другого за величиною елемента масиву з N елементів.
10	(Завдання 1) Обміняти місцями середній та крайній елементи масиву з 9 випадкових чисел.
11	(Завдання 2) Видалити всі повторювані елементи, що зустрічаються більше трьох разів.
12	(Завдання 3) Створити масив і знайти індекс першого елементу, що дорівнює максимальному.
13	(Завдання 1) Обміняти місцями найбільший та перший елементи масиву.
14	(Завдання 2) Видалити всі числа, що повторюються рівно два рази.
15	(Завдання 3) Створити масив з N елементів і знайти індекс мінімального елемента серед непарних чисел.
16	(Завдання 1) Поміняти місцями два крайніх елементи динамічного масиву.
17	(Завдання 2) Видалити всі повторювані елементи, що менші за середнє значення масиву.
18	(Завдання 3) Створити масив і знайти індекс першого додатного елемента.
19	(Завдання 1) Обміняти елементи з мінімальним та максимальним значенням.
20	(Завдання 2) Видалити з масиву всі непарні елементи, що зустрічаються більше двох разів.
21	(Завдання 3) Знайти індекс останнього максимального елемента динамічного масиву.
22	(Завдання 1) Поміняти місцями перший і останній парні елементи масиву.
23	(Завдання 2) Видалити всі числа, що повторюються більше двох разів і більші за 50.
24	(Завдання 3) Створити масив з N випадкових чисел та знайти індекс мінімального елемента.
25	(Завдання 1) Обміняти місцями елементи з максимальним та мінімальним індексом непарних елементів.
26	(Завдання 2) Видалити всі повторювані нулі та одиниці у масиві.
27	(Завдання 3) Створити масив та знайти індекс елемента, найближчого до середнього арифметичного.
28	(Завдання 1) Поміняти місцями крайні та середні елементи

	масиву з 9 випадкових чисел.
29	(Завдання 2) Видалити всі повторювані елементи, що кратні трьом.
30	(Завдання 3) Створити масив і визначити індекс елемента з найбільшим модулем.

6. Рекомендації щодо усунення типових помилок

1. Невиділення пам’яті або невірний розмір. Тому завжди перевіряйте значення `n > 0` перед виділенням пам’яті.
2. Не звільнена пам’ять (memory leak). Тому після використання масиву завжди робіть `delete[] arr;`
3. Вихід за межі масиву. Тому працюйте тільки з індексами `0..n-1`.
4. Неправильна передача динамічного масиву у функцію. Тому передавайте і покажчик, і розмір.
5. Змішування динамічних та статичних масивів. Тому для динамічних масивів не використовуйте `sizeof(arr) / sizeof(arr[0])`.

7. Додаткові ресурси

1. https://dev.to/gtanyware/dynamic-arrays-in-c-24oj
2. https://medium.com/@jcapona/dynamic-array-implementation-cpp-9deadaf1ba8e
3. https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3532.html

Лабораторна робота №11

Тема: Робота з масивами та функціями

1. Мета лабораторної роботи

- Закріпити навички роботи з одновимірними масивами.
- Навчитись передавати масиви у функції як параметри.
- Розвинути вміння побудови функцій, що виконують обчислення, пошук і трансформацію даних.
- Навчитись коректно повертати результати з функцій.
- Розуміти тонкощі передачі масивів за посиланням та роботи з індексами.

2. Навчальні результати

Після виконання лабораторної роботи студент буде:

- знати:
  - механізм передачі масивів у функції (передача вказівника на перший елемент);
  - чому при передачі масиву у функцію необхідно окремо передавати його розмір;
  - відмінність між передачею простих типів даних (за значенням) та масивів (фактично за посиланням);
  - способи повернення результатів обробки масиву (через return або модифікацію самого масиву);
  - принципи декомпозиції складної задачі на менші підзадачі-функції;
- вміти:
  - оголошувати прототипи функцій, що приймають одновимірні масиви як параметри;

- писати функції для ініціалізації, виведення, пошуку та сортування масивів;
- викликати функції з main() та обробляти отримані результати;
- уникати помилок зміни даних у масиві, коли це не передбачено логікою (використання const);
- структурувати код програми, виносячи логіку обробки даних в окремі процедури.

3. Теоретичні відомості

3.1. Масиви як параметри функцій

У мові C++ масив передається в функцію за адресою, тому копія масиву не створюється. Приклад:

```
void printArray(int arr[], int n);
```

Передаються два параметри:

- arr[] – вказівник на перший елемент масиву;
- n – розмір масиву.

3.2. Повернення значення з функції

Функції можуть повертати результати:

```
int getMax(int arr[], int n);
double average(double arr[], int n);
```

Функція не може повернути масив напряму, але може:

- повертати індекс;
- повертати окреме значення;
- змінювати масив через параметри.

3.3. Типові операції над масивами у функціях

- пошук максимального/мінімального елемента;
- обчислення середнього;
- розрахунок кількості елементів, що задовольняють певній умові;
- сортування;
- модифікація елементів (заміна, видалення, перестановка).

3.4. Робота з індексами

Індекси в усіх масивах починаються з 0:

0, 1, 2, ..., n-1

Спроба доступу за межі масиву призводить до непередбачуваних наслідків.

4. Допоміжні конструкції

4.1. Передача масиву в функцію

```
void fillRandom(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        arr[i] = rand() % 100;
    }
}
```

4.2. Функція, що повертає значення

```
int sum(int arr[], int n) {
    int s = 0;
    for (int i = 0; i < n; i++) s += arr[i];
    return s;
}
```

4.3. Функція, що повертає індекс

```
int indexOfMax(int arr[], int n) {
    int maxI = 0;
    for (int i = 1; i < n; i++)
        if (arr[i] > arr[maxI])
            maxI = i;
    return maxI;
}
```

4.4. Функція void для зміни масиву

```
void negateArray(int arr[], int n) {
    for (int i = 0; i < n; i++)
        arr[i] = -arr[i];
}
```

5. Завдання

Завдання №1

Напишіть програму, яка створює два масиви і заповнює їх випадковими числами. Потім вона повинна повернути третій масив, який містить тільки спільні для обох масивів значення (без дублів).

Завдання №2

Напишіть програму, яка виведе всі елементи, які мають значення більше, ніж 5, з наступного масиву:

```
a=[1,1,2,4,5,9,14,22,37,54,87,90,111,243,345]
```

Потім запитайте у користувача число. Виведіть числа з вищенаведеного масиву, які менше числа, вказаного користувачем.

Завдання №3

Гра “Бики та корови”. Правила:

- програма генерує випадковим чином 4-значне число;
- гравцеві пропонують вгадати згенероване програмою число;
- за кожну вгадану гравцем цифру, яка стоїть на правильній позиції, він отримує “корову”;
- за кожну вгадану гравцем цифру, яка стоїть на неправильній позиції, він отримує “бика”;
- після кожного припущення гравцеві повинна виводитися кількість “корів” та “биків”, які він заробив;
- гра закінчена тоді, коли гравець вгадав всі цифри.

Наприклад, комп’ютер загадав число 9978:

```
Ласкаво просимо в гру «Бики та корови»!
Введіть число:
9965
2 корови, 0 биків
9989
2 корови, 1 бик
...
```

6. Рекомендації щодо усунення типових помилок

- Помилка: вихід за межі масиву:
  - причина: неправильні індекси;
  - рішення: перевіряти діапазон $0 \le i < n$.
- Помилка: масив не передано правильно:
  - причина: студенти іноді намагаються повернути масив у вигляді `return arr;`

- рішення: масив НЕ повертається напряму – тільки його елементи або індекси.

- Помилка: ділення на нуль при обчисленні середнього. Рішення:

```
if (n == 0) return 0;
```

- Помилка: змішані типи (int/double). Рішення: приводити типи вручну:

```
double avg = (double)sum / n;
```

- Помилка: передавання не того розміру масиву. Рішення: завжди передавати точне n, яке ввів користувач.

7. Додаткові ресурси

1. https://www.w3schools.com/cpp/cpp_function_array.asp
2. https://www.geeksforgeeks.org/cpp/pass-array-to-functions-in-cpp/

Лабораторна робота №12

Тема: Командна розробка програм з використанням функцій

1. Мета лабораторної роботи

- Ознайомитись із принципами командної розробки програм.
- Навчитись розподіляти програму на окремі модулі та функції.
- Отримати навички обміну кодом, інтеграції компонентів і тестування спільного рішення.
- Сформувати навички структурного та функціонального проєктування.
- Навчитись встановлювати правила взаємодії в команді та стилі написання коду.

2. Навчальні результати

Після виконання лабораторної роботи студент буде:

- знати:
  - принципи модульного програмування та роздільної компіляції;
  - структуру C++ проєкту: заголовкові файли (.h/.hpp) та файли реалізації (.cpp);
  - призначення вартових включення (include guards: #ifndef, #define, #endif) або #pragma once;
  - правила роботи в команді: стиль кодування, розподіл відповідальності, узгодження інтерфейсів;
  - основи документування коду для спільного використання;
- вміти:

- розбивати монолітний код програми на логічні модулі;
- створювати власні заголовкові файли та правильно їх підключати;
- вирішувати конфлікти імен та дублювання коду при інтеграції частин проєкту;
- працювати з чужим кодом, використовуючи лише оголошення функцій (інтерфейс);
- проводити тестування окремих модулів перед їх об’єднанням у кінцеву програму.

3. Теоретичні відомості

3.1. Командна розробка програм: сутність

Командна розробка – це процес спільного створення програмного продукту кількома учасниками. Її ключові принципи:

- розподіл задач;
- узгодженість дій;
- спільні стандарти коду;
- прозорість логіки;
- інтеграція функцій у єдину систему.

3.2. Модульність та функціональний поділ

Командний проєкт зазвичай ділиться на:

- малі функції (атомарні операції);
- модулі (групи функцій);
- інтерфейси (.h);
- реалізації (.cpp).

Функціональний поділ дозволяє працювати незалежно, не заважаючи іншим учасникам.

3.3. Розподіл відповідальності в команді

Таблиця 9

Можливі ролі у команді

Роль	Обов’язки
Архітектор	Структура програми, поділ на модулі
Розробник	Реалізація функцій

Інтегратор	Об’єднання частин коду
Тестувальник	Перевірка роботи всіх функцій
Документатор	Коментарі, інструкції, специфікація

3.4. Використання заголовкових файлів

Правильний поділ:

MyFunctions.h

```
#ifndef MYFUNCTIONS_H
#define MYFUNCTIONS_H

int sum(int a, int b);
void printArray(int arr[], int n);

#endif
```

MyFunctions.cpp

```
#include "MyFunctions.h"
#include <iostream>

int sum(int a, int b) {
    return a + b;
}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++)
        std::cout << arr[i] << " ";
}
```

3.5. Правила інтеграції коду в команді

- Єдиний стиль оформлення.
- Узгоджена структура проєкту.
- Всі функції повинні бути протестовані окремо.
- Інтеграція – після перевірки.
- Всі зміни документуються.

3.6. Тестування під час командної розробки

- Тестування кожної функції окремо (unit-testing).
- Стрес-тести для межових значень.
- Перевірка взаємодії модулів.
- Перевірка продуктивності.

4. Допоміжні конструкції

4.1. Приклад функції як частини командного модуля

```
double computeAverage(double arr[], int n) {
    if (n == 0) return 0;
    double sum = 0;
    for (int i = 0; i < n; i++)
        sum += arr[i];
    return sum / n;
}
```

4.2. Коментарі для командної роботи (Doxygen-стиль)

```
/*
 * @brief Повертає кількість додатних елементів масиву
 * @param arr масив
 * @param n кількість елементів
 * @return кількість додатних чисел
 */
int countPositive(int arr[], int n);
```

4.3. Багатофайлова структура

```
Project/
├── main.cpp
├── mathModule.h
├── mathModule.cpp
├── utils.h
└── utils.cpp
```

5. Завдання

Завдання №1

Гра “Камінь, ножиці, папір”. Користувач повинен грати з комп’ютером, який випадковим чином генерує одне з наступних трьох значень:

- камінь, який ламає ножиці;
- ножиці, які ріжуть папір;
- папір, який покриває камінь.

В кінці гри користувачеві повинно виводитися повідомлення про результат гри і пропозиція зіграти ще раз.

Завдання №2

У поїзді 18 вагонів, в кожному з яких по 36 місць. Інформація про продані на поїзд квитки зберігається в двовимірному масиві, номери рядків якого відповідають номерам вагонів, а номери стовпців – номерам місць. Якщо квиток на те чи інше місце проданий, то відповідний елемент масиву має значення 1, в протилежному випадку – 0. Напишіть програму, яка визначить число вільних місць в будь-якому з вагонів поїзда.

Завдання №3

Напишіть генератор паролів. Зробіть три рівня складності генерації паролів (включаючи їх довжину) і запитайте у користувача, який рівень складності йому потрібен. Проявіть свою винахідливість: надійні паролі повинні складатися з комбінації малих літер, великих літер, цифр та символів. Паролі повинні генеруватися випадковим чином кожен раз, коли користувач запитує новий пароль.

6. Рекомендації щодо усунення типових помилок

- Конфлікти імен функцій:
  - проблема: декілька студентів можуть написати функції з однаковими назвами;
  - рішення:
    - узгодити назви;
    - використовувати префікси: `math_sum()`, `arr_sum()`, `team1_sum()`.
- Відсутність заголовкових файлів:
  - проблема: часто функції описані лише у .cpp, що ускладнює інтеграцію;
  - рішення: виконувати всі оголошення – у файлах *.h.
- Циклічні включення файлів:
  - проблема: призводить до помилки компіляції;

- рішення: використовувати #ifndef, #define, #endif.

- Різний стиль коду у різних учасників:
  - проблема: ускладнює читання та підтримку;
  - рішення: узгодити:
    - відступи;
    - фігурні дужки;
    - стиль назв функцій та змінних.
- Неправильна інтеграція чужих функцій:
  - проблема: студенти інколи змінюють параметри чи повернення функцій;
  - рішення: використовувати командну специфікацію функцій.
- Відсутність тестування:
  - проблема: злиття неперевіреного коду створює проблеми в спільному проєкті;
  - рішення: кожен учасник тестує свої функції перед інтеграцією.
- Використання глобальних змінних:
  - проблема: можливі конфлікти між модулями;
  - рішення: використовувати параметри функцій, а не глобальні дані.

7. Додаткові ресурси

1. https://www.learncpp.com/cpp-tutorial/header-files/
2. https://google.github.io/styleguide/cppguide.html
3. https://isocpp.org/wiki/faq/coding-standards

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gregoire M. Professional C++. 6th ed. Indianapolis : Wiley, 2024. 1376 p. URL: https://www.wiley.com/en-us/Professional+C%2B%2B%2C+6th+Edition-p-9781394193172

2. Horton I., Van Weert P. Beginning C++23: From Beginner to Pro. 7th ed. California : Apress, 2023. 918 p. URL: https://link.springer.com/book/10.1007/978-1-4842-9343-0

3. Dmitrović S. Modern C++ for Absolute Beginners: A Friendly Introduction to the C++ Programming Language and C++11 to C++23 Standards. 2nd ed. California : Apress, 2023. 440 p. URL: https://link.springer.com/book/10.1007/978-1-4842-9274-7

4. Паращук С.Д., Сурков К.Ю., Ізвалов О.В. Практикум із програмування мовою С++ : навч. посіб. Кропивницький : ЕТІ імені Роберта Ельворті, 2023. 260 с. URL: https://eti.edu.ua/images/Vasylieva/IT/pidruchniki/PosibnykC.pdf

5. Stroustrup B. A Tour of C++. 3rd ed. Boston : Addison-Wesley Professional, 2023. 320 p. (C++ In-Depth Series). URL: https://www.stroustrup.com/tour3.html

6. Iglberger K. C++ Software Design: Design Principles and Patterns for High-Quality Software. Sebastopol : O'Reilly Media, 2022. 438 p. URL: https://www.oreilly.com/library/view/c-software-design/9781098113155/

7. Deitel P., Deitel H. C++20 for Programmers: An Objects-Natural Approach. 3rd ed. Boston : Pearson, 2022. 960 p. (Deitel Developer Series). URL: https://www.pearson.com/en-us/subject-catalog/p/c20-for-

programmers-an-objects-natural-approach/P200000000211/9780136905691

8. Davidson G., Gregory K. Beautiful C++: 30 Core Guidelines for Writing Clean, Safe, and Fast Code. Boston : Addison-Wesley Professional, 2021. 352 p. URL: https://www.pearson.com/en-us/subject-catalog/p/beautiful-c-30-core-guidelines-for-writing-clean-safe-and-fast-code/P200000009446/9780137647866

9. Lakos J., Romeo V., Khlebnikov R., Meredith A. Embracing Modern C++ Safely. Boston : Addison-Wesley Professional, 2022. 1376 p. URL: https://www.informit.com/store/embracing-modern-c-plus-plus-safely-9780137380350

10. Roth S. Clean C++20: Sustainable Software Development Patterns and Best Practices. 2nd ed. California : Apress, 2021. 491 p. URL: https://link.springer.com/book/10.1007/978-1-4842-5949-8