Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 9

з навчальної дисципліни

“Базові методології та технології програмування”

РЕАЛІЗАЦІЯ ПРОГРАМНИХ МОДУЛІВ РОЗГАЛУЖЕННЯ ТА ІТЕРАЦІЙНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

ЗАВДАННЯ ВИДАВ

доцент кафедри кібербезпеки та програмного забезпечення

Доренський О. П.

[https://github.com/odorenskyi/](https://github.com/odorenskyi/Dmytro-Parkhomenko-KB18)

ВИКОНАВ

студентка академічної групи КБ-24

Селіхова К.С.

https://github.com/Levanori

ПЕРЕВІРИВ

ст. викладач кафедри кібербезпеки   
та програмного забезпечення

Коваленко А. С.

Кропивницький – 2025

**Мета роботи :** полягає у набутті ґрунтовних вмінь і практичнихнавичок реалізації технології модульного програмування,застосування операторів С/С++ арифметичних, логічних, побітовихоперацій, умови, циклів та вибору під час розроблення статичнихбібліотек, заголовкових файлів та програмних засобів у кросплатформовому середовищі Code::Blocks.

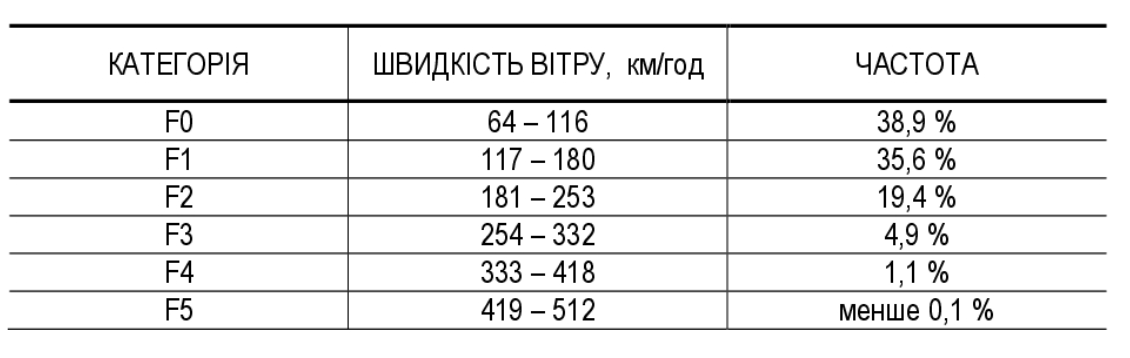
**BAPİAHT 1**

**Завдання :**

**ЗАДАЧА 9.1**

Вхід: швидкість вітру (км/год) під час торнадо.

Вихід: категорія торнадо за шкалою Фудзіти та частота їх виникнення.



**Загальна характеристика завдання**

Завдання 9.1 з Варіанту 1 орієнтоване на застосування базових конструкцій умовного розгалуження та опрацювання числових даних для аналізу природного явища — торнадо. Основною метою є автоматичне визначення категорії торнадо згідно з **шкалою Фудзіти** (шкалою оцінювання сили торнадо за швидкістю вітру) та виведення відповідної **частоти виникнення** такого явища.

Це завдання має прикладне значення: у реальних системах спостереження за погодою важливо оперативно інтерпретувати показники вимірювальних приладів, щоб зробити висновок про потенційну загрозу стихійного лиха. Таким чином, це не просто вправляння у програмуванні, а моделювання елемента системи прийняття рішень у сфері безпеки.

**Формулювання задачі в програмному вигляді**

Користувач вводить значення швидкості вітру. Програма має визначити, в який з діапазонів ця швидкість потрапляє, і видати відповідну категорію та частоту. Якщо введене значення менше за 64 або більше за 512, слід передбачити додаткову перевірку і повідомлення про те, що дана швидкість не належить жодній з категорій.

**Алгоритм дій**

1. Прочитати введене користувачем значення швидкості.
2. Виконати послідовну перевірку умов:
   * Якщо 64 ≤ швидкість ≤ 116 → F0, 38,9%
   * Якщо 117 ≤ швидкість ≤ 180 → F1, 35,6%
   * Якщо 181 ≤ швидкість ≤ 253 → F2, 19,4%
   * Якщо 254 ≤ швидкість ≤ 332 → F3, 4,9%
   * Якщо 333 ≤ швидкість ≤ 418 → F4, 1,1%
   * Якщо 419 ≤ швидкість ≤ 512 → F5, <0,1%
3. Вивести категорію і частоту.
4. Якщо значення не підходить до жодного з діапазонів, вивести повідомлення про помилку або надзвичайну ситуацію (наприклад, "дані поза межами шкали Фудзіти").

**ЗАДАЧА 9.2**

Вхід: температура повітря (у градусах за шкалою Цельсія), зафіксовану о 00:00, 04:00, 8:00, 12:00, 16:00, 20:00 год.

Вихід: середньодобова температура за шкалами Цельсія та Фаренгейта.

Знаючи температуру за шкалою Цельсія, температуру за шкалою Фаренгейта можна розрахувати наступним чином: 9 tp=32+-tc 5

де - температура за шкалою Фаренгейта, с температура за шкалою Цельсія.

**Суть завдання**

Мета завдання 9.2 — реалізувати програму, яка на основі температурних значень, зафіксованих протягом доби в певні моменти часу, обчислює **середньодобову температуру** в **градусах Цельсія** та **переводить її** в **градуси Фаренгейта**.

Завдання охоплює теми **введення масиву чисел**, **арифметичних операцій**, **розрахунку середнього значення**, а також **перетворення одиниць виміру**, що є актуальними в прикладній фізиці, метеорології та екологічних системах.

**Алгоритм розв’язання**

1. Зчитати 6 числових значень температури (можна реалізувати введення в циклі).
2. Обчислити **суму** усіх введених значень.
3. Розділити суму на 6 — отримати **середню температуру за Цельсієм**.
4. Застосувати формулу для обчислення температури за Фаренгейтом.
5. Вивести результати у зручному вигляді.

**ЗАДАЧА 9.3**

Вхід: натуральне число № від 0 до 65535.

Вихід: якщо біт До числа № рівний 0, кількість двійкових нулів у ньому,

інакше — кількість двійкових одиниць\*.

\*під час підрахунку кількості бінарних 0 або 1 рекомендовано використати тернарний оператор «?: ».

**Суть завдання**

Завдання полягає в тому, щоб проаналізувати **натуральне число в діапазоні від 0 до 65535**, тобто **16-бітне беззнакове ціле число**, і залежно від значення **молодшого біта** (тобто крайнього правого біта) обчислити кількість **нулячих** або **одиничних бітів** у його двійковому представленні.

Це завдання демонструє практичне використання **бітових операцій**, **двійкової системи числення** та **тернарного оператора** (в мовах типу C/C++).

**Алгоритм розв’язання**

1. **Зчитати число**.
2. **Визначити значення молодшого біта**:
   * Це можна зробити за допомогою операції **N % 2** або **N & 1**.
3. **Перетворити число в двійкову форму**.
4. **Порахувати кількість 1 або 0**, залежно від того, чому дорівнює молодший біт.
5. **Використати тернарний оператор ?:**, щоб скоротити логіку умовного вибору.

**ЗАДАЧА 9.4**

За введеним користувачем символом "z" викликається s calculation(), "г" - функція задачі 9.1, "s" - функція задачі 9.2, "t" - функція задачі 9.3; якщо користувач вводить інші символи, вони ігноруються, при чому видається звуковий сигнал про помилкове введення. Після цього, якщо користувач за запитом додатка вводить символ "q" або "о", відбувається вихід з програми, інакше виконання програми повторюється.

У випадку, якщо параметром і/або результатом функції є дані нестандартного типу (наприклад, складового), то такий тип варто реалізувати у заголовковому файлі.

**Суть завдання**

Це завдання — своєрідна **керуюча оболонка** (головне меню) до попередніх задач (9.1–9.3). Воно реалізує **обробку введення користувача**, де певні символи запускають відповідні функції. Такий підхід нагадує **меню керування в програмному інтерфейсі командного рядка**.

**Опис функціоналу**

* Користувач вводить **один символ**.
* Залежно від символу:
  + 'z' — викликає calculation(). (Уточнюється програмістом, можливо, додаткова обчислювальна функція.)
  + 'г' — викликається функція з **задачі 9.1** (визначення категорії торнадо).
  + 's' — викликається функція з **задачі 9.2** (середня температура в Цельсіях і Фаренгейтах).
  + 't' — викликається функція з **задачі 9.3** (аналіз бітів числа).
* Якщо введено інший символ — **ігнорується**, але також видається **звуковий сигнал** (наприклад, через \a у C++).
* Після виконання функції:
  + Якщо введено 'q' або 'о', програма **завершує роботу**.
  + Інакше програма **повторює цикл введення**.

**Алгоритм роботи**

1. Створити нескінченний цикл (while (true)).
2. Запросити в користувача символ.
3. Виконати switch або if-else за символом:
   * 'z' → виклик calculation()
   * 'г' → виклик task9\_1()
   * 's' → виклик task9\_2()
   * 't' → виклик task9\_3()
   * інше → beep + пропуск
4. Після виконання функції — запросити символ виходу:
   * 'q' або 'о' → вихід з програми.
   * інше → повернення до початку циклу.

**Пояснення структурованості програми**

Завдання змушує студента **інкапсулювати логіку кожної задачі в окрему функцію**, що дозволяє:

* краще **організувати код**,
* **тестувати окремі задачі**,
* **повторно використовувати код**.

Це відповідає принципам:

* **структурного програмування**,
* **розділення обов’язків**,
* **керованості через меню/введення**.

**Висновок до лабораторної роботи №9:**

У процесі виконання завдань 9.1–9.4 було закріплено та поглиблено знання з основ програмування на мові C++, зокрема щодо обробки умов, використання побітових операцій, виконання математичних розрахунків, а також організації взаємодії з користувачем у консольному режимі. Завдання були тісно пов’язані між собою і мали на меті сформувати практичні навички побудови модульної програми, яка здатна обробляти вхідні дані різного типу та приймати рішення на основі аналізу введеної інформації.

У задачі 9.1 було реалізовано класифікацію сили торнадо згідно зі шкалою Фудзіти, що базується на введеній користувачем швидкості вітру. Для цього було використано умовні конструкції для перевірки, у який з діапазонів потрапляє введене значення. Кожній категорії (від F0 до F5) відповідала не лише певна швидкість, але й статистична частота її виникнення. Таким чином, задача мала не тільки обчислювальний, але й аналітичний характер, оскільки дозволяла зрозуміти частоту природних явищ різної інтенсивності.

У другій задачі, 9.2, розглядалася обробка температурних даних, зафіксованих протягом доби у визначені години. Це завдання вимагало обчислити середньодобову температуру за шкалою Цельсія, а також перевести це значення у шкалу Фаренгейта за відомою формулою. Реалізація завдання включала арифметичні обчислення, роботу з масивами або окремими змінними, а також дотримання точності обчислень. Ця задача дозволила попрактикуватися у роботі з дійсними числами, математичними виразами та підготовці результатів для зручного виведення користувачу.

Задача 9.3 була однією з найтехнічніших — вона вимагала знання двійкової системи числення та побітових операцій. Користувач вводив натуральне число у межах від 0 до 65535. Відповідно до значення одного з його бітів, програма визначала, що потрібно підрахувати — кількість нулів чи одиниць у його двійковому поданні. Сам підрахунок здійснювався з використанням побітового зсуву та логічних операцій, а вибір між двома варіантами реалізовувався за допомогою тернарного оператора. Це завдання особливо важливе з точки зору розуміння внутрішнього представлення даних у пам’яті комп’ютера та оптимізації умовних обчислень.

Нарешті, у задачі 9.4 реалізовано програму-керівник, яка на основі введеного символу викликає відповідну функцію, пов’язану з однією з попередніх задач. У випадку некоректного введення користувач отримує звуковий сигнал та повідомлення про помилку. Такий підхід дозволяє створити простий, але ефективний інтерфейс для взаємодії з користувачем. Програма підтримує багаторазове виконання дій без потреби у повторному запуску — користувач може вибирати функції довільну кількість разів або завершити виконання за допомогою команди виходу. Особливістю цього завдання є також модульна структура програми, яка передбачає винесення складових типів та функцій у заголовкові файли для зручності та повторного використання.

Узагальнюючи, можна зазначити, що виконання цих завдань дозволило не лише закріпити теоретичні знання з програмування, а й отримати практичні навички створення структурованого, логічно зв’язаного коду з обробкою різних типів вхідних даних. Було відпрацьовано роботу з умовами, циклами, функціями, побітовими операціями, арифметикою, тернарними виразами, а також реалізовано взаємодію користувача з програмою у формі діалогу. Отримані результати можуть бути основою для створення більш складних систем обробки даних у майбутньому.

**Контрольні запитання:**

**1. Яким оператором С/С++ можливо повноцінно замінити тернарний оператор? Відповідь обґрунтуйте й доведіть експериментально.**

**Відповідь:**  
Тернарний оператор ?: можна повністю замінити оператором розгалуження if-else. Цей оператор виконує вибір між двома варіантами залежно від умови.

**Пояснення:**  
Тернарний оператор має формат:

condition ? expr1 : expr2;

Якщо condition істинна, виконується expr1, інакше – expr2.

Еквівалентний код з if-else:

if (condition)

результат = expr1;

else

результат = expr2;

**Експеримент:**

#include <iostream>

using namespace std;

int main() {

int a = 10, b = 20;

int max1, max2;

// За допомогою тернарного оператора

max1 = (a > b) ? a : b;

// За допомогою if-else

if (a > b)

max2 = a;

else

max2 = b;

cout << "max1 = " << max1 << "\nmax2 = " << max2 << endl;

return 0;

}

В обох випадках змінні max1 та max2 матимуть однакове значення.

**2. Що в програмуванні розуміють під пріоритетом виконання операцій та асоціативністю?**

**Відповідь:**

* **Пріоритет операцій** — порядок, у якому компілятор виконує операції у виразі. Наприклад, множення має вищий пріоритет, ніж додавання.
* **Асоціативність** — порядок виконання операцій одного рівня пріоритету, наприклад, зліва направо або справа наліво.

**Пояснення:**  
Вираз 3 + 4 \* 5 виконується як 3 + (4 \* 5) через пріоритет множення над додаванням.  
Вираз a = b = 5 виконується справа наліво, тому b = 5, а потім a = b.

**3. Яку область видимості мають змінні, оголошені в тілі циклу або умови (вибору)? Відповідь обґрунтуйте та доведіть експериментально.**

**Відповідь:**  
Змінні, оголошені в тілі блоку { ... } циклу або умови, мають **локальну область видимості** — вони доступні лише всередині цього блоку.

**Приклад:**

if (true) {

int x = 10;

cout << x << endl; // Доступна

}

// cout << x; // Помилка: x поза областю видимості

**Експеримент:**

#include <iostream>

using namespace std;

int main() {

if (true) {

int x = 100;

cout << "x всередині блоку: " << x << endl;

}

// cout << x << endl; // Тут буде помилка компіляції

return 0;

}

**4. Якою є асоціативність операцій арифметичних, логічних, логічних порозрядних, інкрементна, декремента, тернарної та порівняння в мові програмування С/С++?**

**Відповідь:**

| **Операції** | **Асоціативність** |
| --- | --- |
| Арифметичні (+, -, \*, /, %) | Зліва направо |
| Логічні (&&, ` |  |
| Порозрядні (&, ` | , ^`) |
| Інкремент ++, декремент -- (унарні) | Справа наліво |
| Тернарний оператор ?: | Справа наліво |
| Операції порівняння (==, !=, <, >, <=, >=) | Зліва направо |

**5. Перелічіть випадки, за яких доцільно використовувати тернарний оператор С/С++, й наведіть приклад його запису.**

**Відповідь:**  
Доцільно використовувати, коли:

* Потрібно виконати простий вибір між двома значеннями.
* Потрібен компактний запис без розгорнутого if-else.
* Коли код повинен бути більш читабельним.

**Приклад:**

int a = 5, b = 10;

int max = (a > b) ? a : b;

**6. Яке значення міститиме змінна cnt після виконання наступної інструкції: cnt--;?**

**Відповідь:**  
Значення cnt зменшиться на 1 від поточного значення.

**7. Чим константна змінна, оголошена за допомогою кваліфікатора const, відрізняється від змінної? Сформулюйте правило, коли змінну варто оголошувати саме константною.**

**Відповідь:**  
const змінна не може змінюватися після ініціалізації, тоді як звичайна змінна може змінюватися у будь-який момент.

**Правило:**  
Оголошуйте змінні як const, якщо їх значення не повинно змінюватися, щоб запобігти помилкам і зробити код більш безпечним.

**8. Яких типів можуть бути операнди логічних операторів С/С++?**

**Відповідь:**  
Операнди можуть бути будь-якими типами, які можна неявно привести до bool — найчастіше це цілі типи, покажчики, або булеві значення.

**9. Яке значення міститиме змінна cnt при: bool cnt = !!0;?**

**Відповідь:**  
!!0 — подвійне логічне заперечення:

* !0 = true (1)
* !true = false (0)

Отже, cnt = false (0).

**10. Сформулюйте правило запису виразу ініціалізації у циклах з параметром (for) С++.**

**Відповідь:**  
У циклі for ініціалізація — це вираз, який виконується один раз на початку циклу. Зазвичай це оголошення і присвоєння змінної лічильника.

**Приклад:**

for (int i = 0; i < 10; i++) {

// тіло циклу

}

**11. Наведіть приклад запису циклу for, який виконується до введення користувачем символа «!» без використання жодної змінної, оголошеної до цього циклу.**

**Відповідь:**

#include <iostream>

using namespace std;

int main() {

for (char c = getchar(); c != '!'; c = getchar()) {

cout << "Введено: " << c << endl;

}

return 0;

}

**12. Яким є синтаксис запису оператора повного й неповного розгалуження у С/С++? Перелічіть допустимі та недопустимі операції у логічних виразах оператора умови.**

**Відповідь:**

* **Повний оператор:**

if (умова) {

// тіло, якщо істина

} else {

// тіло, якщо хиба

}

* **Неповний оператор:**

if (умова) {

// тіло, якщо істина

}

**Допустимі логічні операції:** &&, ||, !, ==, !=, <, >, <=, >=

**Недопустимі:** операції зі зміною стану (наприклад, присвоєння у виразі умови, якщо не обґрунтовано).

**13. Які логічні операції реалізовано у С/С++? Перелічіть логічні оператори С/С++, типи їх операндів і результату виконання.**

**Відповідь:**

* ! — логічне НЕ (унарний оператор)
* && — логічне І
* || — логічне АБО

Операнди: будь-які типи, що приводяться до булевого значення.

Результат: тип bool.

**14. Яке значення міститиметься у змінній cnt після ++cnt -= 1;?**

**Відповідь:**  
Операція розбивається на:

1. ++cnt — збільшує cnt на 1, потім повертає посилання на cnt.
2. -= 1 — зменшує cnt на 1.

Отже, загальний ефект — значення не змінюється.

**15. Наведіть приклад опису константної змінної в модулі С/С++.**

**Приклад:**

const int DAYS\_IN\_WEEK = 7;

**16. Яке призначення має оператор break у С/С++ та в чому полягає особливість його виконання?**

**Відповідь:**  
break призначений для негайного виходу з циклу або оператора switch. Особливість — він припиняє поточний цикл без виконання подальших ітерацій.

**17. Які логічні порозрядні операції реалізовано у С/С++? Перелічіть унарні й бінарні оператори порозрядних операцій, типи їх операндів та результату виконання. Наведіть приклади.**

**Відповідь:**

* Унарний: ~ (побітове заперечення)
* Бінарні: & (AND), | (OR), ^ (XOR), << (зсув вліво), >> (з

сув вправо)

Операнди: цілі типи (int, unsigned int і т.п.)

Результат: той самий тип, що й операнди.

**Приклад:**

int a = 5; // 0101

int b = 3; // 0011

int c = a & b; // 0001 -> 1

int d = ~a; // побітове заперечення 5

**18. Наведіть приклади фрагментів алгоритмів із циклом із лічильником (for), циклом з передумовою (while) та циклом із післяумовою (do…while).**

**Відповідь:**

// for

for (int i = 0; i < 5; i++) {

cout << i << " ";

}

// while

int i = 0;

while (i < 5) {

cout << i << " ";

i++;

}

// do…while

int i = 0;

do {

cout << i << " ";

i++;

} while (i < 5);