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 розгалужених та інтераційних 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

студент академічної групи КІ - 24

Федотов А. О.

ПЕРЕВІРИВ

ст. викладач кафедри кібербезпеки   
та програмного забезпечення

Коваленко А.С

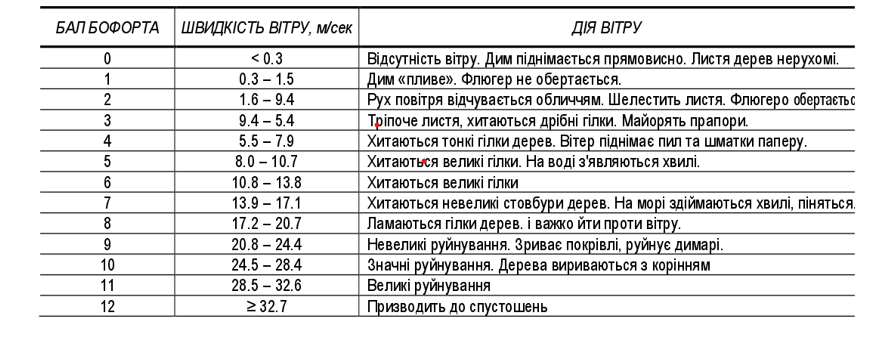
Кропивницький – 2025

**ТЕМА:** Реалізація програмних модулів розгалужених та інтераційних обчислювальних процесів

**МЕТА** Полягає у набутті ґрунтовних вмінь і практичних навичок реалізації технології модульного програмування, застосування операторів С / С++ арифметичних, логічних, побітових операцій, умови, циклів та вибору під час розроблення статичних бібліотек, заголовкових файлів та програмних засобів у кросплатформовому середовищі Code::Blocks

**ЗАДАЧА 9.1**

Вхід: бал Бафорта. Вихід: сила вітру, км/с.



Малюнок 1 – завдання 9.1

**ПОСТАНОВА ЗАДАЧІ**

**1.** Реалізувати функцію, яка приймає ціле число — бал за шкалою Бофорта (від 0 до 12 включно).

**2.** Для кожного значення визначити:

**2.1.** Мінімальну та максимальну швидкість вітру в метрах за секунду (м/с).

**2.2.** Перевести ці значення у кілометри за годину (км/год).

**2.3.** Описати вплив вітру на навколишнє середовище.

**3**. У випадку, якщо число виходить за межі допустимого діапазону (менше 0 або більше 12), повернути повідомлення про помилку.

**4**. Повернути рядок, що містить:

**4.1.** Швидкість вітру в м/с та км/год.

**4.1.** Відповідний опис погодних умов.

**5**. Використовувати тип stringstream для зручного форматування тексту.

**АНАЛІЗ ЗАДАЧІ**

**1.** Вхідним параметром функції є ціле число bofort, що відповідає балу за шкалою Бофорта.

**2.** Шкала Бофорта описує швидкість вітру і його вплив на навколишнє середовище. Кожен бал має свій діапазон швидкостей та опис.

**3.** Для значень від 0 до 12 встановлюються відповідні:

**3.1.** Мінімальні та максимальні значення швидкості вітру в м/с.

**3.2.** Опис вітру, наприклад: "Дим піднімається прямо", "Ламаються гілки дерев", тощо.

**4.** Швидкість вітру переводиться в км/год за допомогою множення на 3.6.

**5.** Якщо значення bofort не належить діапазону [0; 12], виводиться повідомлення про некоректне значення.

**6.** Для форматування вихідного тексту використовується stringstream, що дозволяє зручно формувати рядок з числовими значеннями та текстом.

**7.** Функція повертає сформований текстовий результат типу string.

**РЕАЛІЗАЦІЯ КОДУ В БІБЛІОТЕЦІ**

Вихідний файл ModuleFedotov.cpp

**string printWindSpeedInfo(int bofort) {**

**stringstream result;**

**double minSpeed = 0, maxSpeed = 0;**

**string description;**

**switch (bofort) {**

**case 0: minSpeed = 0; maxSpeed = 0.3; description = "Відсутність вітру. Дим піднімається прямо. Листя дерев нерухомі."; break;**

**case 1: minSpeed = 0.3; maxSpeed = 1.5; description = "Дим 'пливе'. Флюгер не обертається."; break;**

**case 2: minSpeed = 1.6; maxSpeed = 3.4; description = "Рух повітря відчувається обличчям. Шелестить листя."; break;**

**case 3: minSpeed = 3.5; maxSpeed = 5.4; description = "Тріпоче листя, хитаються дрібні гілки. Майорять прапори."; break;**

**case 4: minSpeed = 5.5; maxSpeed = 7.9; description = "Хитаються тонкі гілки дерев. Вітер піднімає пил та шматки паперу."; break;**

**case 5: minSpeed = 8.0; maxSpeed = 10.7; description = "Хитаються великі гілки. На воді з’являються хвилі."; break;**

**case 6: minSpeed = 10.8; maxSpeed = 13.8; description = "Хитаються великі гілки."; break;**

**case 7: minSpeed = 13.9; maxSpeed = 17.1; description = "Хитаються невеликі стовбури дерев. На морі здіймаються хвилі."; break;**

**case 8: minSpeed = 17.2; maxSpeed = 20.7; description = "Ламаються гілки дерев. Важко йти проти вітру."; break;**

**case 9: minSpeed = 20.8; maxSpeed = 24.4; description = "Невеликі руйнування. Зриває покрівлі, руйнує димарі."; break;**

**case 10: minSpeed = 24.5; maxSpeed = 28.4; description = "Значні руйнування. Дерева вириваються з корінням."; break;**

**case 11: minSpeed = 28.5; maxSpeed = 32.6; description = "Великі руйнування."; break;**

**case 12: minSpeed = 32.7; maxSpeed = 50.0; description = "Призводить до спустошень."; break;**

**default:**

**result << "Незадовільний бал Бофорда!" << endl;**

**return result.str();**

**}**

**result << "Швидкість вітру: " << minSpeed << " - " << maxSpeed << " м/с ("**

**<< minSpeed \* 3.6 << " - " << maxSpeed \* 3.6 << " км/ч)\n"**

**<< "Опис: " << description << endl;**

**return result.str();**

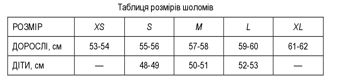
**}**

Заголовковий файл ModuleFedotov

string printWindSpeedInfo(int bofort);

**ЗАДАЧА 9.1**

Вхід: довжина обхвату голови у міліметрах.

****Вихід: розмір шолому.

Малюнок 2 – завдання 9.2

**ПОСТАНОВА ЗАДАЧІ**

1. Реалізувати функцію headCircumference, яка приймає одне ціле число — обхват голови в сантиметрах.
2. На основі обхвату голови визначити категорію користувача: "Діти" або "Дорослі".
3. Залежно від діапазону значення обхвату, визначити відповідний розмір шолома: XS, S, M, L, XL.
4. Якщо обхват голови не входить у жоден з передбачених діапазонів, вивести повідомлення "Розмір не знайдений!".
5. Повернути у вигляді рядка:

**5.1.** Категорію користувача.

* 1. Розмір шолома.

1. Використовувати stringstream для форматування виводу.

**АНАЛІЗ ЗАДАЧІ**

1. Вхідним параметром є ціле число headCircumference, що представляє обхват голови в сантиметрах.
2. Функція використовує послідовні умовні оператори if-else if, щоб визначити відповідний розмір шолома та категорію:
   1. Діти: 48–53 см.

**2.2** Дорослі: 53–62 см.

1. Розмір шолома залежить від обхвату голови:

**3.1.** Діти: S (48–49), M (50–51), L (52–53)

**3.2.** Дорослі: XS (53–54), S (55–56), M (57–58), L (59–60), XL (61–62)

1. Є перекриття для 53 см (входить і в "Діти", і в "Дорослі"). Через порядок перевірок, буде обрано категорію "Дорослі".
2. Якщо значення не підпадає під жоден з діапазонів, змінна size отримує значення "Розмір не знайдений!", а категорія залишається незаповненою.
3. За допомогою stringstream формується підсумковий рядок з інформацією про категорію та розмір.
4. Функція повертає цей рядок як результат.

**РЕАЛІЗАЦІЯ КОДУ В БІБЛІОТЕЦІ**

Вихідний файл ModuleFedotov.cpp

string headCircumference(int headCircumference) {

stringstream result;

string size, category;

if (headCircumference >= 53 && headCircumference <= 54) { size = "XS"; category = "Дорослі"; }

else if (headCircumference >= 55 && headCircumference <= 56) { size = "S"; category = "Дорослі"; }

else if (headCircumference >= 57 && headCircumference <= 58) { size = "M"; category = "Дорослі"; }

else if (headCircumference >= 59 && headCircumference <= 60) { size = "L"; category = "Дорослі"; }

else if (headCircumference >= 61 && headCircumference <= 62) { size = "XL"; category = "Дорослі"; }

else if (headCircumference >= 48 && headCircumference <= 49) { size = "S"; category = "Діти"; }

else if (headCircumference >= 50 && headCircumference <= 51) { size = "M"; category = "Діти"; }

else if (headCircumference >= 52 && headCircumference <= 53) { size = "L"; category = "Діти"; }

else { size = "Розмір не знайдений!"; }

result << "Категорія: " << category << "\nРозмір шолома: " << size << endl;

return result.str();

}

Заголовковий файл ModuleFedotov

string headCircumference(int headCircumference);

**ЗАДАЧА 9.3**

Вхід: натуральне число № від 0 до 8999.

Вихід: якщо біт D\_{10} числа № рівний 0, кількість двійкових нулів у ньому,

інакше кількість двійкових одиниць".

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

**ПОСТАНОВА ЗАДАЧІ**

1. Реалізувати функцію countBits, яка приймає одне беззнакове ціле число (unsigned int) в діапазоні від 0 до 8999.
2. Якщо число перевищує 8999 — функція повинна повернути 0.
3. Необхідно перевірити, чи встановлений **10-й біт (D10)** у числі (тобто біт під індексом 9).
4. Залежно від стану біта D10:

**4.1.** Якщо D10 встановлений (==1), підрахувати кількість **встановлених бітів** (1) у числі.

* 1. Якщо D10 не встановлений (==0), підрахувати кількість **вимкнених бітів** (0).

1. Повернути кількість знайдених бітів у вигляді unsigned int.

**АНАЛІЗ ЗАДАЧІ**

1. Функція приймає один параметр number типу unsigned int.
2. Перевірка if (number > 8999) обмежує обробку числа в межах від 0 до 8999. Якщо число більше — одразу повертається 0.
3. Змінна isD10Set визначає, чи встановлений 10-й біт (D10) у числі. Для цього використовується побітова операція number & (1 << 9).
4. Змінна count використовується для підрахунку кількості бітів згідно з умовою.
5. Цикл for перебирає всі 32 біти змінної number.

**5.1.** Для кожного біта виконується побітова перевірка: чи він дорівнює 1.

* 1. Якщо D10 == 1 — рахуємо лише біти, що рівні 1.

**5.3**. Якщо D10 == 0 — рахуємо лише біти, що рівні 0.

1. Повертається кількість відповідних бітів.
2. Таким чином, поведінка функції залежить від **одного конкретного біта**, який визначає, чи ми рахуємо 1 чи 0 в усьому числі.
3. Функція використовує лише базові побітові операції й не залежить від сторонніх бібліотек.

**РЕАЛІЗАЦІЯ КОДУ В БІБЛІОТЕЦІ**

Вихідний файл ModuleFedotov.cpp

unsigned int countBits(unsigned int number) {

if (number > 8999) {

return 0;

}

bool isD10Set = (number & (1 << 9)) != 0;

unsigned int count = 0;

for (int i = 0; i < 32; i++) {

bool bitIsSet = (number & (1 << i)) != 0;

count += (isD10Set ? bitIsSet : !bitIsSet) ? 1 : 0;

}

return count;

}

Заголовковий файл ModuleFedotov

unsigned int countBits(unsigned int number);

**ПОСТАНОВА ЗАДАЧІ**

1. Розробити консольну програму, яка виконує інтерактивне меню з можливістю виклику трьох задач (9.1, 9.2, 9.3) та функції s\_calculation.
2. Забезпечити коректну роботу з українською мовою в консолі за допомогою налаштувань кодування (1251).
3. Виводити початкову інформацію за допомогою функції Information() з зовнішнього модуля.
4. Реалізувати цикл, у якому користувач обирає одну з чотирьох дій:

**4.1.** Ввести \ для запуску функції s\_calculation(x, y, z) з введенням трьох чисел.

**4.2.** Ввести 5 для запуску задачі 9.1 (printWindSpeedInfo(bofort)).

**4.3.** Ввести 4 для запуску задачі 9.2 (headCircumference()).

**4.3.** Ввести 3 для запуску задачі 9.3 (countBits()).

1. У випадку некоректного введення — вивести повідомлення про помилку та звуковий сигнал.
2. Після виконання дії запитати користувача, чи бажає він завершити роботу програми. Якщо користувач вводить t, T або C, програма завершується.
3. Після виходу з циклу — викликати system("pause").

**АНАЛІЗ ЗАДАЧІ**

1. На початку програми встановлюються кодові сторінки для підтримки української мови: SetConsoleOutputCP(1251), SetConsoleCP(65001), SetConsoleCP(1251) — одна з них повторюється, що є надмірним.
2. Викликається функція Information(), імовірно, для виводу вступної інформації (з модуля OutputName.h).
3. Змінна b контролює завершення циклу. Програма працює в do-while циклі до введення користувачем символів t, T або C.
4. За введеним символом a користувач вибирає, яку задачу виконати:

**4.1** \: функція s\_calculation приймає три числа x, y, z і повертає обчислений результат.

* 1. '5': функція printWindSpeedInfo — очікує введення числа Бофорта (0–12). Некоректне значення просить ввести повторно.

**4.3** '4': функція headCircumference — очікує коректний обхват голови згідно з відомими діапазонами. Перевірка значень реалізована через багаторазове порівняння (можна оптимізувати).

**4.4 '3'**: функція countBits — перевіряє, що введене число в межах 0–8999. Якщо ні — просить ввести знову.

1. У разі введення будь-якого іншого символу виконується звуковий сигнал Beep і повідомлення про помилку.
2. Після кожної операції користувачу пропонується завершити роботу або повторити виконання циклу.
3. Програма завершується командою system("pause"), яка дозволяє побачити результати перед закриттям вікна (актуально для Windows).

**ПРОГРАМНА РЕАЛІЗАЦІЯ**

#include <iostream>

#include <windows.h>

#include <OutputName.h>

#include <ModulesFedotov.h>

using namespace std;

int main()

{

SetConsoleOutputCP(1251);

SetConsoleCP(65001);

SetConsoleCP(1251);

Information();

char b;

do {

char a;

cout << "Виберіть функцію\n" <<

"Введіть \\ щоб викликати функцію s\_calculation()\n" <<

"Введіть 5 щоб викликати функцію задачі 9.1\n" <<

"Введіть 4 щоб викликати функцію задачі 9.2\n" <<

"Введіть 3 щоб викликати функцію задачі 9.3\n";

cin >> a;

switch (a)

{

case '\\':

double x, y, z;

cout << "Введіть три числа для x, y, z: ";

cin >> x >> y >> z;

cout << "Результат роботи функції s\_calculation: " << s\_calculation(x, y, z) << endl;

break;

case '5':

int bofort;

while (true) {

cout << "Введіть бал Бафорта від 0-12: ";

cin >> bofort;

if (bofort > 12) {

cout << "Неправильне значення, спробуйте ще раз: ";

}

else {

cout << "Результат роботи задачі 9.1: " << printWindSpeedInfo(bofort) << endl;

break;

}

}

break;

case '4':

int headCircumferenc;

cout << "Введіть обхвад голови.\nДля дорослих від 53 або 54, 55 або 56, 57 або 58, 59 або 60, 61 або 62.\nДля дітей від 48 або 49, 50 або 51, 52 або 53 " << endl;

while (true) {

cin >> headCircumferenc;

if (headCircumferenc == 53 || headCircumferenc == 54 || headCircumferenc == 55 || headCircumferenc == 56 ||

headCircumferenc == 57 || headCircumferenc == 58 || headCircumferenc == 59 || headCircumferenc == 60 ||

headCircumferenc == 61 || headCircumferenc == 62 || headCircumferenc == 48 || headCircumferenc == 49 ||

headCircumferenc == 50 || headCircumferenc == 51 || headCircumferenc == 52 || headCircumferenc == 53) {

cout << "Результат роботи задачі 9.2: " << headCircumference(headCircumferenc) << endl;

break;

}

else {

cout << "Неправильне значення, спробуйте ще раз: ";

}

}

break;

case '3':

unsigned int number;

while (true) {

cout << "Введіть натуральне число від 0 до 8999: ";

cin >> number;

if (number > 8999) {

cout << "Неправильне значення, спробуйте ще раз: ";

}

else {

cout << "Результат роботи задачі 9.3: " << countBits(number) << endl;

break;

}

}

break;

default:

cout << "Помилкове введеня!" << endl;

Beep(1000, 500);

break;

}

cout << "Бажаєте завершити програму ? Якщо ні — введіть будь - який символ.Якщо так — введіть t, T або С :";

cin >> b;

} while (b != 't' && b != 'T' && b != 'C');

system("pause");

}

**РЕЗУЛЬТАТ ВИКОНАННЯ ПРОГРАМИ:**

----------------------------------------

| Andrey Fedotov, CUNTU, Kropyvnickyi |

| Андрій Федотов, ЦНТУ, Кропивницький |

-------- (c) All Rights Reserved --------

Виберіть функцію

Введіть \ щоб викликати функцію s\_calculation()

Введіть 5 щоб викликати функцію задачі 9.1

Введіть 4 щоб викликати функцію задачі 9.2

Введіть 3 щоб викликати функцію задачі 9.3

3

Введіть натуральне число від 0 до 8999: 5

Результат роботи задачі 9.3: 30

Бажаєте завершити програму ? Якщо ні — введіть будь - який символ.Якщо так — введіть t, T або С :C

Press any key to continue . . .

**ВИСНОВОКИ**

1. Засвоєно основи модульного програмування.
2. Навчився створювати заголовкові файли.
3. Навчився підключати бібліотеки в проєкт.
4. Використано оператори арифметики.
5. Застосовано логічні оператори.
6. Використано побітові операції.
7. Опрацьовано конструкції умов (if, else).
8. Застосовано конструкції вибору (switch).
9. Використано цикли (for, while).
10. Навчився будувати структуру проєкту.
11. Розділено код на модулі.
12. Поліпшено читабельність коду.
13. Підвищено зручність повторного використання коду.
14. Створено власні функції.
15. Використано передачу аргументів у функції.
16. Повернено значення з функцій.
17. Підключено зовнішні модулі.
18. Навчився налагоджувати помилки компіляції.
19. Освоєно компіляцію в Code::Blocks.
20. Використано консольний ввід/вивід.
21. Засвоєно синтаксис мови C++.
22. Отримано навички роботи з #include.
23. Вивчено роботу з заголовками .h.
24. Навчився створювати статичні бібліотеки.
25. Навчився правильно підключати .lib до проєкту.
26. Освоєно використання namespace std.
27. Навчився працювати з stringstream.
28. Відпрацьовано методи обробки введення користувача.
29. Засвоєно правила структурованого програмування.
30. Зрозуміло переваги модульності.
31. Полегшено тестування окремих частин програми.
32. Вивчено взаємодію між модулями.
33. Опановано роботу зі змінними.
34. Зрозуміло роль типів даних.
35. Закріплено роботу з умовними виразами.
36. Поглиблено знання побітових операцій.
37. Оцінено важливість коментарів у коді.
38. Навчився перевіряти правильність введення.
39. Реалізовано обробку некоректного вводу.
40. Навчився оформлювати головну функцію main().
41. Застосовано функції для виводу інформації.
42. Освоєно взаємодію з користувачем.
43. Опрацьовано структуру циклічної взаємодії.
44. Навчився реалізовувати багатофункціональне меню.
45. Отримано навички обробки умов завершення програми.
46. Використано оператор break для виходу з циклу.
47. Опрацьовано багатоваріантні гілки логіки.
48. Вивчено команду system("pause").
49. Ознайомлено з установкою кодування консолі.
50. Реалізовано обробку чисел і рядків.
51. Використано оператор << для виводу.
52. Отримано навички побітових зсувів.
53. Опановано побітову перевірку окремих бітів.
54. Реалізовано коректну роботу з unsigned int.
55. Навчився перевіряти окремі біти через &.
56. Освоєно умовний оператор ? :.
57. Поглиблено розуміння логічних виразів.
58. Підвищено рівень алгоритмічного мислення.
59. Отримано досвід розробки під Windows.
60. Закріплено практичні навички програмування в C++.

**ВІДПОВІДЬ НА КОНТРОЛЬНІ ЗАПИТАННЯ**

1. Тернарний оператор можна замінити конструкцією if-else. Оператор if-else виконує ту ж саму функцію, що й тернарний оператор, але займає більше місця. Вони функціонально еквівалентні, однак тернарний оператор є більш компактним і зручним для використання в коротких умовах.
2. Пріоритет виконання операцій визначає порядок, у якому виконуються оператори в виразі. Асоціативність визначає напрямок, у якому виконуються операції однакового пріоритету. Якщо оператори мають асоціативність зліва направо, операції виконуються зліва направо, і навпаки для правосторонньої асоціативності.
3. Змінні, оголошені в тілі циклу або умови, мають локальну область видимості. Вони доступні тільки в межах цього блоку коду і не можна використовувати їх за межами цього блоку.
4. Асоціативність операцій в С/С++ для арифметичних операцій зазвичай зліва направо, для логічних та порівняння — зліва направо, а для побітових операцій — зліва направо. Інкремент та декремент мають правосторонню асоціативність, а тернарний оператор має асоціативність зліва направо.
5. Тернарний оператор доцільно використовувати в умовах, коли потрібно обрати одне значення з двох можливих, залежно від умови, і записати це в одну стрічку.
6. Змінна cnt після виконання операції cnt-- міститиме значення на 1 менше за попереднє.
7. Константна змінна, оголошена через const, не може змінювати своє значення після ініціалізації. Вона відрізняється від звичайної змінної тим, що її значення фіксоване. Змінну варто оголошувати як константу, коли значення цієї змінної не повинно змінюватися впродовж виконання програми.
8. Операнди логічних операторів в С/С++ можуть бути типів bool, int, або будь-яким типом, який може бути приведений до логічного значення (0 для false і будь-яке значення, що не дорівнює нулю, для true).
9. Змінна cnt при bool cnt = !!0; міститиме значення false, оскільки подвійне заперечення (!!) від значення 0 дає результат false.
10. Правило запису виразу ініціалізації у циклах з параметром (for) в С++ полягає в тому, що ініціалізація змінних відбувається перед початком виконання циклу, і вона повинна бути лише один раз. Змінні для ітерацій мають бути ініціалізовані перед умовою циклу.