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 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

студент академічної групи КБ-24

Заріцкий В. А.

ПЕРЕВІРИВ

викладач кафедри кібербезпеки   
та програмного забезпечення

Коваленко А. С.

Кропивницький – 2025

Мета роботи 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. **Варіант №9**

**Завдання до лабораторної роботи**

1. Реалізувати статичну бібліотеку модулів libModulesПрізвище C/C++, яка містить функцію розв’язування задачі 8.1.
2. Реалізувати програмне забезпечення розв’язування задачі 8.2 — консольний застосунок

# Аналіз і постановка задачі 9.1

Вхід: Кількість спожитої електроенергії (кВт\*год).

Вихід: Сума до сплати у гривнях за такими тарифами:

* До 150 кВтгод (включно): 130,843 коп./кВтгод.
* Від 150 до 800 кВтгод (включно): 241,945 коп./кВтгод.
* Понад 800 кВтгод: 534,047 коп./кВтгод.

План реалізації: написати функцію, яка приймає кількість кВт⋅год і за допомогою розгалужувальних операторів (if-else або тернарного оператора) визначає відповідний тариф та обчислює суму.

Лістинг задачі 9.1:

double calculateElectricityBill(double kWh) {

double rate;

if (kWh <= 150)

rate = 0.130843;

else if (kWh <= 800)

rate = 0.241945;

else

rate = 0.534047;

return kWh \* rate;

}

# Аналіз і постановка задачі 9.2

**Вхід:** Масив із 12 значень середньомісячної температури (°C).

**Вихід:** Середня температура за шкалою Цельсія та Фаренгейта.

План реалізації:

1. Обчислити середнє значення за Цельсієм.
2. Перевести його у Фаренгейти за формулою:

Лістинг задачі 9.2:

struct Temperature {

double celsius;

double fahrenheit;

};

Temperature calculateAverageTemperature(const double temps[12]) {

double sum = 0.0;

for (int i = 0; i < 12; i++) {

sum += temps[i];

}

double avgC = sum / 12.0;

Temperature result;

result.celsius = avgC;

result.fahrenheit = 32 + (9.0 / 5.0) \* avgC;

return result;

}

# Аналіз і постановка задачі 9.3

**Вхід:** Натуральне число N (від 0 до 5,740,500).

**Вихід:**

* Якщо 10-й біт () числа N рівний 0, – вивести кількість двійкових нулів.
* Інакше – кількість двійкових одиниць.

План реалізації:

1. За допомогою побітових операцій перевірте 10-й біт.
2. Для підрахунку нулів або одиниць використовуйте цикл та тернарний оператор «? : ».
3. Якщо N = 0, повертати 1.

Лістинг задачі 9.3:

int countBits(int N) {

if (N == 0) {

return 1;

}

bool bit10 = (N & (1 << 10)) != 0;

int count = 0;

while (N > 0) {

int bit = N & 1;

// Якщо bit10 == false – підраховуємо нулі, інакше одиниці.

count += (bit10 ? (bit == 1 ? 1 : 0) : (bit == 0 ? 1 : 0));

N >>= 1;

}

return count;

}

Тестові набори:

# ****Набір 1: «Нормальні» дані (Typical Cases)****

**Задача 9.1:**

**Вхід:** 100 кВт⋅год

**Очікуваний результат:**

* Тариф для ≤150 кВт⋅год: 130,843 коп./кВт⋅год
* Розрахунок: 100 ⋅ 0.130843 = **13.0843 грн** (якщо рахувати в гривнях, де 100 коп. = 1 грн)

**Задача 9.2:**

**Вхід:** Масив з 12 температур, наприклад:

double temps[12] = {0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0};

**Очікуваний результат:**

* Середня температура (°C): (0+1+2+…+11) / 12 = 5.5 °C
* Перетворення: 32 + (9.0/5) \* 5.5 ≈ **41.9 °F**

**Задача 9.3:**

**Вхід:** N = 100

* У двійковій формі: 100 → «1100100»
* Перевірка 10-го біта: для числа 100 10-й біт (індекс 10, якщо рахувати від 0) має значення 0.

**Очікуваний результат:**

* Оскільки 10-й біт = 0, потрібно підрахувати кількість нулів.
* «1100100» містить 4 нулі → **4**

# ****Набір 2: Граничні значення (Boundary Cases)****

**Задача 9.1:**

**Вхід:** 150 кВт⋅год

**Очікуваний результат:**

* 150 кВт⋅год потрапляє до першої категорії (до 150 включно)
* Розрахунок: 150 ⋅ 0.130843 = **19.62645 грн**

**Задача 9.2:**

**Вхід:** Масив з 12 температур з негативними та позитивними значеннями, наприклад:

double temps[12] = {-5.0, -3.0, 0.0, 5.0, 10.0, 15.0, 20.0, 18.0, 12.0, 7.0, 0.0, -2.0};

**Очікуваний результат:**

* Сума ≈ 77.0 → середнє ≈ 6.4167 °C
* Перетворення: 32 + (9.0/5) \* 6.4167 ≈ **43.55 °F**

**Задача 9.3:**

**Вхід:** N = 1024

* У двійковій формі: 1024 → «10000000000»
* 10-й біт (за індексацією від 0) встановлено (значення 1).

**Очікуваний результат:**

* Оскільки 10-й біт ≠ 0, підраховуємо кількість одиниць.
* У «10000000000» лише 1 одиниця → **1**

# ****Набір 3: Крайові/Екстремальні значення (Edge Cases)****

**Задача 9.1:**

**Вхід:** 1000 кВтгод

**Очікуваний результат:**

* Оскільки 1000 > 800, використовується тариф 0.534047
* Розрахунок: 1000 ⋅ 0.534047 = **534.047 грн**

**Задача 9.2:**

**Вхід:** Масив, де всі значення однакові, наприклад:

double temps[12] = {25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0};

**Очікуваний результат:**

* Середня температура = 25 °C
* Перетворення: 32 + (9.0/5) \* 25 = 32 + 45 = **77 °F**

**Задача 9.3:**

**Вхід:** N = 0

* Особливий випадок: 0 у двійковій системі зазвичай представляється як "0".
* За умовою, якщо число 0, очевидно, що результат – кількість нулів;
* **Примітка:** у функцію була додана обробка N = 0 (повертається 1, оскільки 0 має один розряд, який дорівнює 0).

**Очікуваний результат:** **1**

# Результат Unit-тестування задач 9.1-9.3:

Тестування задачі 9.1

========

Аргумент(-и): 100 kWh

Очікується: 13.0843

Отримано: 13.0843 - PASSED

========

Аргумент(-и): 150 kWh

Очікується: 19.6264

Отримано: 19.6264 - PASSED

========

Аргумент(-и): 1000 kWh

Очікується: 534.047

Отримано: 534.047 - PASSED

========

Тестування задачі 9.2

========

Аргумент(-и): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

Очікується: 5.5 °C, 41.9 °F;

Отримано: 5.5 °C, 41.9 °F - PASSED

========

Аргумент(-и): -5, -3, 0, 5, 10, 15, 20, 18, 12, 7, 0, -2

Очікується: 6.4167 °C, 43.55 °F;

Отримано: 6.41667 °C, 43.55 °F - PASSED

========

Аргумент(-и): 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25

Очікується: 25 °C, 77 °F;

Отримано: 25 °C, 77 °F - PASSED

========

Тестування задачі 9.3

========

Аргумент(-и): 100

Очікується: 4

Отримано: 4 - PASSED

========

Аргумент(-и): 1024

Очікується: 1

Отримано: 1 - PASSED

========

Аргумент(-и): 0

Очікується: 1

Отримано: 1 - PASSED

========

# Вихідний код проєкту ModulesZaritskyi

Лістинг main.cpp:

#include <cmath>

#include "ModulesZaritskyi.h"

using namespace std;

double s\_calculation(double x, double y, double z) {

// Перевірка області визначення: x має бути більше 4, щоб логарифм був додатнім.

if (x <= 4.0) {

return NAN; // або можна сигналізувати про помилку іншим способом

}

double logValue = log10(x - 3.0);

// Якщо логарифм не додатній, обчислення неможливе

if (logValue <= 0.0) {

return NAN;

}

// Обчислення виразу під квадратним коренем

double underSqrt = (12.0 \* x \* x) / logValue;

// Перевірка, що підкореневий вираз не від'ємний

if (underSqrt < 0.0) {

return NAN;

}

// Обчислення S згідно з формулою

double result = pow(3.0 \* sin(sqrt(underSqrt)), 2.0) + 0.5 \* z;

return result;

}

double calculateElectricityBill(double kWh) {

double rate;

if (kWh <= 150)

rate = 0.130843;

else if (kWh <= 800)

rate = 0.241945;

else

rate = 0.534047;

return kWh \* rate;

}

Temperature calculateAverageTemperature(const double temps[12]) {

double sum = 0.0;

for (int i = 0; i < 12; i++) {

sum += temps[i];

}

double avgC = sum / 12.0;

Temperature result;

result.celsius = avgC;

result.fahrenheit = 32 + (9.0 / 5.0) \* avgC;

return result;

}

int countBits(int N) {

if (N == 0) {

return 1;

}

bool bit10 = (N & (1 << 10)) != 0;

int count = 0;

while (N > 0) {

int bit = N & 1;

// Якщо bit10 == false – підраховуємо нулі, інакше одиниці.

count += (bit10 ? (bit == 1 ? 1 : 0) : (bit == 0 ? 1 : 0));

N >>= 1;

}

return count;

}

Лістинг ModulesZaritskyi.h:

#ifndef MODULESZARITSKYI\_H\_INCLUDED

#define MODULESZARITSKYI\_H\_INCLUDED

struct Temperature {

double celsius;

double fahrenheit;

};

double s\_calculation(double x, double y, double z);

double calculateElectricityBill(double kWh);

Temperature calculateAverageTemperature(const double temps[12]);

int countBits(int N);

#endif // MODULESZARITSKYI\_H\_INCLUDED

# Вихідний код TestDriver

#include <iostream>

#include "ModulesZaritskyi.h"

using namespace std;

void run\_tests() {

// Тестування задачі 9.1

int task1\_input[3] = {100, 150, 1000};

double task1\_expected[3] = {13.0843, 19.62645, 534.047};

cout << "Тестування задачі 9.1" << endl;

cout << "========" << endl;

for (int i = 0; i < 3; i++) {

double result = calculateElectricityBill(task1\_input[i]);

cout << "Аргумент(-и): " << task1\_input[i] << " kWh" << endl

<< "Очікується: " << task1\_expected[i] << endl

<< "Отримано: " << result

<< (abs(result - task1\_expected[i]) < 0.001 ? " - PASSED" : " - FAILED") << endl

<< "========" << endl;

}

// Тестування задачі 9.2

double task2\_input[3][12] = {

{0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0},

{-5.0, -3.0, 0.0, 5.0, 10.0, 15.0, 20.0, 18.0, 12.0, 7.0, 0.0, -2.0},

{25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0},

};

Temperature task2\_expected[3] = {

{5.5, 41.9},

{6.4167, 43.55},

{25, 77}

};

cout << "Тестування задачі 9.2" << endl;

cout << "========" << endl;

for (int i = 0; i < 3; i++) {

Temperature result = calculateAverageTemperature(task2\_input[i]);

cout << "Аргумент(-и): ";

for (int j = 0; j < 12; j++) {

cout << task2\_input[i][j];

if (j < 11) {

cout << ", ";

}

}

cout << endl;

cout << "Очікується: " << task2\_expected[i].celsius << " °C, " << task2\_expected[i].fahrenheit << " °F;" << endl

<< "Отримано: " << result.celsius << " °C, " << result.fahrenheit << " °F"

<< (abs(result.celsius - task2\_expected[i].celsius) < 0.01 &&

abs(result.fahrenheit - task2\_expected[i].fahrenheit) < 0.01 ? " - PASSED" : " - FAILED") << endl

<< "========" << endl;

}

// Тестування задачі 9.3

int task3\_input[3] = {100, 1024, 0};

int task3\_expected[3] = {4, 1, 1};

cout << "Тестування задачі 9.3" << endl;

cout << "========" << endl;

for (int i = 0; i < 3; i++) {

int result = countBits(task3\_input[i]);

cout << "Аргумент(-и): " << task3\_input[i] << endl

<< "Очікується: " << task3\_expected[i] << endl

<< "Отримано: " << result

<< (result == task3\_expected[i] ? " - PASSED" : " - FAILED") << endl

<< "========" << endl;

}

}

int main() {

run\_tests();

return 0;

}

# Аналіз і постановка задачі 9.4

Програма реагує на введення символів:

* 'n' → виклик s\_calculation() (функція з Lab8).
* 'v' → виклик функції задачі 9.1 (calculateElectricityBill).
* 'c' → виклик функції задачі 9.2 (calculateAverageTemperature).
* 'x' → виклик функції задачі 9.3 (countBits).
* Інші символи – генерувати звуковий сигнал (можна використати системну функцію або std::cout << "\a").

Після виконання – запит на введення символу:

* Якщо користувач вводить 'z' або 'Z' – вихід із програми.
* Інакше – повторити виконання.

Алгоритм задачі 9.4 у вигляді псевдокоду:

ПОЧАТОК

ПОКИ (введений\_символ НЕ дорівнює 'z' І НЕ дорівнює 'Z') ВИКОНУВАТИ:

ВИВЕСТИ "Введіть символ (n, v, c, x) або 'z' для виходу: "

ЗЧИТАТИ введений\_символ

ВИБІР введений\_символ:

ВИПАДОК 'n':

ВИКЛИКАТИ s\_calculation()

ВИХІД

ВИПАДОК 'v':

ВИВЕСТИ "Введіть кількість кВт\*год: "

ЗЧИТАТИ kWh

ВИВЕСТИ "Сума до сплати: ", calculateElectricityBill(kWh), " грн."

ВИХІД

ВИПАДОК 'c':

ВИВЕСТИ "Введіть 12 значень температури (°C): "

ДЛЯ i від 0 ДО 11:

ЗЧИТАТИ temperatures[i]

avg\_temperature ← calculateAverageTemperature(temperatures)

ВИВЕСТИ "Середня температура: ", avg\_temperature.celsius, "°C, ", avg\_temperature.fahrenheit, "°F"

ВИХІД

ВИПАДОК 'x':

ВИВЕСТИ "Введіть число N: "

ЗЧИТАТИ N

ЯКЩО (N & (1 << 10)) ТОДІ

тип\_аналізу ← "одиниць"

ІНАКШЕ

тип\_аналізу ← "нулів"

ВИВЕСТИ "Результат аналізу (кількість ", тип\_аналізу, "): ", countBits(N)

ВИХІД

ЗАМИНОК:

ВИВЕСТИ "Некоректне введення!"

ВИВЕСТИ "Бажаєте продовжити? (введіть інший символ, крім z/Z для повторення): "

ЗЧИТАТИ введений\_символ

КІНЕЦЬ ПОКИ

КІНЕЦЬ

# Вихідний код Zaritskyi\_task

#include <iostream>

#include "ModulesZaritskyi.h"

using namespace std;

int main() {

char ch;

do {

cout << "Введіть символ (n, v, c, x) або 'z' для виходу: ";

cin >> ch;

switch(ch) {

case 'n':

double x, y, z;

cout << "\nEnter x: ";

cin >> x;

cout << "Enter y: ";

cin >> y;

cout << "Enter z: ";

cin >> z;

cout << s\_calculation(x, y, z) << endl;

break;

case 'v': {

double kWh;

cout << "Введіть кількість кВт\*год: ";

cin >> kWh;

cout << "Сума до сплати: " << calculateElectricityBill(kWh) << " грн." << endl;

break;

}

case 'c': {

double temps[12];

cout << "Введіть 12 значень температури (°C): ";

for (int i = 0; i < 12; i++)

cin >> temps[i];

Temperature avg = calculateAverageTemperature(temps);

cout << "Середня температура: " << avg.celsius << "°C, "

<< avg.fahrenheit << "°F" << endl;

break;

}

case 'x': {

int N;

cout << "Введіть число N: ";

cin >> N;

cout << "Результат аналізу (кількість "

<< ((N & (1 << 10)) ? "одиниць" : "нулів")

<< "): " << countBits(N) << endl;

break;

}

default:

cout << "\aНекоректне введення!" << endl;

break;

}

cout << "Бажаєте продовжити? (введіть інший символ, крім z/Z для повторення): ";

cin >> ch;

} while(ch != 'z' && ch != 'Z');

return 0;

}



# Висновки до лабораторної роботи

У ході виконання лабораторної роботи було набуто навички реалізації технології модульного програмування, застосування операторів C++ арифметичних, логічних, побітових операцій, умови, циклів та вибору під час розроблення статичних бібліотек, заголовкових файлів та програмних засобів у середовищі Code::Blocks.

Було успішно реалізовано три функції для розв'язування задач 9.1, 9.2 та 9.3:

1. Функція calculateElectricityBill для розрахунку суми оплати за електроенергію за різними тарифами.
2. Функція calculateAverageTemperature для обчислення середньорічної температури за шкалами Цельсія та Фаренгейта.
3. Функція countBinaryDigits для підрахунку нулів або одиниць у двійковому представленні числа залежно від значення 10-го біта.

Ці функції були інтегровані в статичну бібліотеку libModulesZaritskyi.a та відповідний заголовковий файл ModulesZaritskyi.h. Для перевірки правильності роботи функцій було розроблено тестовий драйвер, який виконав модульне тестування з використанням наборів контрольних прикладів.

Також було створено програму для розв'язування задачі 9.4, яка дозволяє користувачеві вибирати функцію для виклику та вводити відповідні параметри. Програма використовує функції з розробленої статичної бібліотеки.

Під час виконання лабораторної роботи були успішно застосовані такі концепції програмування:

* Розгалужені обчислювальні процеси з використанням операторів if-else та switch-case
* Ітераційні обчислювальні процеси з використанням циклів for та while
* Побітові операції для роботи з бітами числа
* Тернарний оператор для скорочення коду
* Модульне програмування для розбиття задачі на окремі компоненти

Особливо цікавим було розв'язання задачі 9.3, яка вимагала роботи з бітами числа та використання тернарного оператора. Ця задача дозволила глибше зрозуміти принципи побітових операцій та їх застосування на практиці.

Для подальшого вдосконалення розробленого програмного забезпечення можна запропонувати:

1. Додати більш детальну валідацію вхідних даних для запобігання помилок при некоректному введенні
2. Розширити функціональність програми, додавши можливість зберігати результати обчислень у файл
3. Покращити інтерфейс користувача, використовуючи більш сучасні бібліотеки для роботи з консоллю

Загалом, мета лабораторної роботи була досягнута: набуто ґрунтовних вмінь і практичних навичок реалізації технології модульного програмування та застосування різноманітних операторів C++ при розробці статичних бібліотек та програмних засобів.

# Відповіді на контрольні запитання

1. **Яким оператором С/С++ можливо повноцінно замінити тернарний оператор?**  
   Умовним оператором if-else. Можна експериментально показати, що ви отримуєте той самий результат, використовуючи обидва підходи.
2. **Що в програмуванні розуміють під пріоритетом виконання операцій та асоціативністю?**  
   Пріоритет визначає порядок обчислення операцій, а асоціативність – напрямок виконання операцій з однаковим пріоритетом (зліва направо або справа наліво).
3. **Яку область видимості мають змінні, оголошені в тілі циклу або умови?**  
   Такі змінні мають блочну область видимості (локальні для цього блоку).
4. **Якою є асоціативність операцій (арифметичних, логічних, побітових, інкремент/декремент, тернарного, порівняння) в С/С++?**

Наприклад, операції множення/ділення мають вищий пріоритет, а тернарний оператор має праву асоціативність.

1. **Перелічіть випадки, за яких доцільно використовувати тернарний оператор С/С++**  
   Коли потрібно компактно записати умовний вираз, наприклад:

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

1. **Яке значення міститиме змінна cnt після виконання наступної інструкції: cnt--;**  
   cnt буде зменшено на 1.
2. **Чим константна змінна відрізняється від звичайної? Коли її використовувати?**  
   Константна змінна (з кваліфікатором const) не може бути змінена після ініціалізації, що покращує безпеку коду. Її використовують, коли значення не повинно змінюватися.
3. **Яких типів можуть бути операнди логічних операторів?**  
   У С/С++ операнди логічних операторів повинні бути приведені до типу bool.
4. **Яке значення міститиме змінна cnt при: bool cnt = !!0;**  
   cnt буде false (тобто 0), оскільки 0 перетворюється в false.
5. **Сформулюйте правило запису виразу ініціалізації у циклах з параметром (for)**  
   Вираз ініціалізації в циклі for виконується один раз перед початком циклу і визначає початкове значення лічильника або інших змінних.