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

Кондратенко Д.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.

Завдання:

1. Реалізувати функції розв’язування задач 9.1–9.3 як складових статичної бібліотеки libModulesПрізвище.а (проект ModulesПрізвище, створений під час виконання лабораторної роботи №8).
2. Реалізувати програмне забезпечення розв’язування задачі 9.4 на основі функцій статичної бібліотеки libModulesПрізвище.а.

**Варіант №15:**

# **Аналіз і постановка задачі 9.1:**

**Вхідні дані:**

посадовий оклад (грн),

заробітна плата працівника (грн),

стаж роботи (років).

**Вихідні дані:**

сума до виплати з урахуванням надбавки (грн).

**Постановка задачі:**

1. Ввести три змінні: оклад, зарплата, стаж.
2. Визначити відсоток надбавки відповідно до стажу.
3. Обчислити надбавку та додати її до зарплати.
4. Вивести фінальну суму до виплати.

Лістинг задачі 9.1:

double salary\_calculation(double salary, double experience,

double office\_salary, double result\_salary){

cout << "Введіть розмір заробітної плати(грн): " << endl;

cin >> salary;

cout << "Введіть розмір посадового окладу(грн): " << endl;

cin >> office\_salary;

cout << "Введіть стаж роботи на підприємстві(роки): " << endl;

cin >> experience;

if(experience >= 3 && experience <= 5) {

office\_salary \* 10/100;

}

else if(experience >= 5 && experience <=7){

office\_salary \* 15/100;

}

else if(experience >= 7 && experience <= 15){

office\_salary \* 20/100;

}

else if(experience > 15) {

office\_salary \* 25/100;

}

result\_salary = salary + office\_salary;

cout << "Сума до виплати = " << result\_salary << endl;

}

# **Аналіз і постановка задачі 9.2:**

**Вхідні дані:**

6 значень температури в градусах Цельсія.

**Вихідні дані:**

Середня температура в Цельсіях і Фаренгейтах.

**Постановка задачі:**

1. Ввести 6 температур.
2. Знайти їх середнє арифметичне.
3. Перевести середню температуру з Цельсія у Фаренгейт.
4. Вивести обидва значення.

Лістинг задачі 9.2:

void temperature\_calculation(double avg\_celsius, int size, double avg\_fahrenheit, const double temps[]){

double sum = 0.0;

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

sum += temps[i];

}

avg\_celsius = sum / size;

avg\_fahrenheit = 32 + (9.0/5.0) \*avg\_celsius;

}

# **Аналіз і постановка задачі 9.3:**

**Вхідні дані:**

Натуральне число N

**Вихідні дані:**

Кількість нулів або одиниць у залежності від стану біта.

**Постановка задачі:**

1. Ввести число N
2. Перевірити 11-й біт числа.
3. Якщо він дорівнює 0 — порахувати нулі, інакше — одиниці.
4. Вивести результат.

Лістинг задачі 9.3:

double analyze\_bit(unsigned int N){

int bit11 = (N >> 11) & 1;

int count = 0;

bitset<32> bits(N);

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

if(bit11 == 0 && bits[i] == 0) count++;

if(bit11 == 1 && bits[i] == 1) count++;

}

return count;

}

Тестові набори до задачі 9.1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Зарплата(грн) | Оклад(грн) | Стаж(років) | Очікувана надбавка(%) | Очікувана сума до виплати(грн) |
| 1 | 10000 | 8000 | 4 | 10% -> 800грн | 10000+800=10800 |
| 2 | 12000 | 10000 | 6 | 15% -> 1500грн | 12000+150=13500 |
| 3 | 9000 | 7000 | 10 | 20% -> 1400грн | 9000+1400=13500 |

Тестові набори до задачі 9.2:

|  |  |  |
| --- | --- | --- |
| № | Очікувана середня(°C) | Очікувана (°F) |
| 1 | 14.3333 | 57.7999 |
| 2 | 5 | 41 |
| 3 | 20.5 | 68.9 |

Тестові набори до задачі 9.3:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Число N (у двійковому) | 11-й біт | Очікуваний результат |
| 1 | 00000100000000000000 (2048) | 1 | 1 |
| 2 | 0000010000101000 (2056) | 1 | 2 |
| 3 | 00000100000000000000 (2120) | 1 | 3 |

# **Результат Unit-тестування задач 9.1-9.3:**

# --- Тест задач? 9.1 ---

# ЗП: 10000, Стаж: 4, Оклад: 8000 -> Результат: 10800 | Оч?кується: 10800 Passed

# ЗП: 12000, Стаж: 6, Оклад: 10000 -> Результат: 13500 | Оч?кується: 13500 Passed

# ЗП: 9000, Стаж: 10, Оклад: 7000 -> Результат: 10400 | Оч?кується: 10400 Passed

# --- Тест задач? 9.2 ---

# Тест 1:

# Оч?кувана °C: 14.3333, Отримана: 14.3333 Passed

# Оч?кувана °F: 57.7999, Отримана: 57.8 Passed

# Тест 2:

# Оч?кувана °C: 5, Отримана: 5 Passed

# Оч?кувана °F: 41, Отримана: 41 Passed

# Тест 3:

# Оч?кувана °C: 20.5, Отримана: 20.5 Passed

# Оч?кувана °F: 68.9, Отримана: 68.9 Passed

# --- Тест задач? 9.3 ---

# Число: 2048 -> Результат: 1 | Оч?кується: 1 Passed

# Число: 2056 -> Результат: 2 | Оч?кується: 2 Passed

# Число: 2120 -> Результат: 3 | Оч?кується: 3 Passed

**Вихідний код проєкту ModulesKondratenko:**

#include <iostream>

#include <cmath>

#include <bitset>

using namespace std;

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

return fabs(abs(sin(y - pow(z, 2 ))) + sqrt(x) - sqrt(pow((y \* z), x) + y / (2 \* M\_PI)));

}

double salary\_calculation(double salary, int experience, double office\_salary){

double bonus = 0.0;

if(experience >= 3 && experience <= 5)

bonus = office\_salary \* 0.10;

else if(experience >= 5 && experience <=7)

bonus = office\_salary \* 0.15;

else if(experience >= 7 && experience <= 15)

bonus = office\_salary \* 0.20;

else if(experience > 15)

bonus = office\_salary \* 0.25;

return salary + bonus;

}

void temperature\_calculation(const double temps[], int size, double& avg\_celsius, double& avg\_fahrenheit){

double sum = 0.0;;

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

sum += temps[i];

avg\_celsius = sum / size;

avg\_fahrenheit = 32 + (9.0/5.0) \*avg\_celsius;

}

double analyze\_bit(unsigned int N){

int bit11 = (N >> 11) & 1;

int count = 0;

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

count += (((N >> i) & 1) == bit11) ? 1 : 0;

}

return count;

}

**Вихідний код проєкту ModulesKondratenko.h:**

#ifndef MODULESKONDRATENKO\_H\_INCLUDED

#define MODULESKONDRATENKO\_H\_INCLUDED

#include <cmath>

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

double salary\_calculation(double salary, int experience, double office\_salary);

void temperature\_calculation(const double temps[], int size, double& avg\_celsius, double& avg\_fahrenheit);

double analyze\_bit(unsigned int N);

#endif // MODULESKONDRATENKO\_H\_INCLUDED

**Вихідний код проєкту TestDriver:**

#include <iostream>

#include "ModulesKondratenko.h"\

#include <clocale>

using namespace std;

void test\_salary() {

cout << "--- Тест задачі 9.1 ---\n";

struct {

double salary;

int experience;

double office\_salary;

double expected;

} tests[] = {

{10000, 4, 8000, 10000 + 800},

{12000, 6, 10000, 12000 + 1500},

{9000, 10, 7000, 9000 + 1400}

};

for (auto& t : tests) {

double result = salary\_calculation(t.salary, t.experience, t.office\_salary);

cout << "ЗП: " << t.salary << ", Стаж: " << t.experience << ", Оклад: " << t.office\_salary

<< " -> Результат: " << result << " | Очікується: " << t.expected;

cout << (abs(result - t.expected) < 1e-2 ? " Passed\n" : " Failed\n");

}

}

void test\_temperature() {

cout << "\n--- Тест задачі 9.2 ---\n";

struct {

double temps[6];

double expected\_c;

double expected\_f;

} tests[] = {

{{10.0, 12.0, 15.0, 18.0, 17.0, 14.0}, 14.3333, 32 + (9.0/5.0) \* 14.3333},

{{5.0, 5.0, 5.0, 5.0, 5.0, 5.0}, 5.0, 32 + (9.0/5.0) \* 5.0},

{{20.0, 22.0, 19.0, 21.0, 23.0, 18.0}, 20.5, 32 + (9.0/5.0) \* 20.5}

};

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

double avg\_c = 0.0, avg\_f = 0.0;

temperature\_calculation(tests[i].temps, 6, avg\_c, avg\_f);

cout << "\nТест " << i+1 << ":\n";

cout << "Очікувана °C: " << tests[i].expected\_c << ", Отримана: " << avg\_c

<< (fabs(avg\_c - tests[i].expected\_c) < 1e-2 ? " Passed\n" : " Failed\n");

cout << "Очікувана °F: " << tests[i].expected\_f << ", Отримана: " << avg\_f

<< (fabs(avg\_f - tests[i].expected\_f) < 1e-2 ? " Passed\n" : " Failed\n");

}

}

void test\_bits() {

cout << "\n--- Тест задачі 9.3 ---\n";

struct {

unsigned int N;

int expected;

} tests[] = {

{0b0000100000000000, 1},

{0b0000100000001000, 2},

{0b0000100001001000, 3}

};

for (auto& t : tests) {

int result = analyze\_bit(t.N);

cout << "Число: " << t.N << " -> Результат: " << result << " | Очікується: " << t.expected;

cout << (result == t.expected ? " Passed\n" : " Failed\n");

}

}

int main() {

setlocale(LC\_ALL, "Ukrainian");

test\_salary();

test\_temperature();

test\_bits();

return 0;

}

**Аналіз та постановка задачі 9.4:**

Згідно з умовою задачі 9.4:

Користувач вводить символ:

"r" — викликається функція **задачі 9.1**

"s" — викликається функція **задачі 9.2**

"t" — викликається функція **задачі 9.3**

"z" — викликається s\_calculation()

Інші символи ігноруються, але викликається **звуковий сигнал про помилкове введення.**

Якщо користувач вводить "q" або "Q" — **програма завершується**.

В іншому випадку — виконання **повторюється**.

**Вимоги до ПЗ:**

Програма має:

Реагувати на введення певного символу (r, s, t, z, q, Q)

Ігнорувати інші символи та повідомляти про помилку (звуковий сигнал).

Кожна дія повинна викликати відповідну функцію задач:

r — надбавка до зарплати (Задача 9.1)

s — середньодобова температура (Задача 9.2)

t — підрахунок бітів (Задача 9.3)

z — спеціальний розрахунок s\_calculation()

Програма завершується при введенні "q" або "Q".

Програма має працювати в циклі до завершення.

Інші символи ігноруються, але викликається **звуковий сигнал про помилкове введення.**

**Проєктування архітектури:**

main()

└── цикл з очікуванням вводу символу

├── 'r' → call task\_9\_1()

├── 's' → call task\_9\_2()

├── 't' → call task\_9\_3()

├── 'z' → call s\_calculation()

├── 'q'/'Q' → вихід

└── інше → сигнал + повторення

**Вихідний код проєкту Kondratenko\_task:**

#include <iostream>

#include <cctype>

#include <limits>

#include <clocale>

#include "ModulesKondratenko.h"

using namespace std;

void run\_task() {

char command;

while (true) {

cout << "\nВведіть команду ('z', 'r', 's', 't') або 'q' для виходу: ";

cin >> command;

switch (tolower(command)) {

case 'z': {

double x, y, z;

cout << "Введіть x, y, z для обчислення S: ";

cin >> x >> y >> z;

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

cout << "Результат обчислення s\_calculation: " << result << endl;

break;

}

case 'r': {

double salary, office\_salary;

int experience;

cout << "Введіть заробітну плату, посадовий оклад і стаж: ";

cin >> salary >> office\_salary >> experience;

double result = salary\_calculation(salary, experience, office\_salary);

cout << "Сума до виплати: " << result << " грн\n";

break;

}

case 's': {

double temps[6];

cout << "Введіть температуру (00:00, 04:00, 08:00, 12:00, 16:00, 20:00): ";

for (int i = 0; i < 6; ++i) {

cin >> temps[i];

}

double avg\_c, avg\_f;

temperature\_calculation(temps, 6, avg\_c, avg\_f);

cout << "Середня температура: " << avg\_c << " °C, " << avg\_f << " °F\n";

break;

}

case 't': {

unsigned int N;

cout << "Введіть натуральне число N: ";

cin >> N;

int count = analyze\_bit(N);

cout << "Результат аналізу бітів: " << count << endl;

break;

}

case 'q':

cout << "Завершення програми.\n";

return;

default:

cout << "\aНевірна команда! Повторіть введення.\n";

}

}

}

int main() {

setlocale(LC\_ALL, "Ukrainian");

run\_task();

return 0;

};

**Висновки до лабораторної роботи:**

1. Отримано практичні навички реалізації технології модульного програмування.
2. Закріплено вміння використовувати арифметичні, логічні, побітові операції, умовні конструкції, цикли та оператори вибору в С++.
3. Реалізовано функції як частину статичної бібліотеки libModulesПрізвище.a.
4. Навченося розробляти заголовкові файли для доступу до функцій.
5. Освоєно методику створення та використання консольних застосунків у Code::Blocks.
6. Вивчено механізм підключення статичних бібліотек до проєктів.
7. Створено та виконано модульне тестування функцій (unit tests).
8. Навченося документувати результати тестування в консолі.
9. Освоєно методику створення системних тестів (test suite) для комплексної перевірки.
10. Здобуто досвід у використанні Git-репозиторію для збереження результатів роботи.
11. Визначено роль інтерфейсів у розділенні реалізації та використання функцій.
12. Продемонстровано важливість точного дотримання структури проєктів.
13. Одержано навички формулювання специфікацій до функцій.
14. Розвинуто здатність самостійно усувати помилки (debugging).
15. Навченося оформлювати звіт згідно з ДСТУ 3008:2015.
16. Удосконалено вміння коментування та структурування коду.
17. Здобуто досвід у проектуванні архітектури програмного забезпечення.
18. Розширено знання про область видимості змінних.
19. Удосконалено навички аналізу вимог до ПЗ.
20. Поглиблено розуміння критеріїв вибору типу циклу.
21. Удосконалено логічне мислення під час складання умов у коді.
22. Отримано практику застосування тернарного оператора.
23. Ознайомлено з методикою створення багатомодульного ПЗ.
24. Навченося писати драйвери для тестування.
25. Закріплено знання з налаштування компілятора у Code::Blocks.
26. Розглянуто особливості використання інкременту та декременту.
27. Освоєно поняття асоціативності та пріоритету операторів.
28. Поглиблено знання про логічні та побітові оператори.
29. Вивчено оператори break та continue.
30. Розуміння обмежень у виразах умов.
31. Визначено переваги використання const-змінних.
32. Розроблено власні приклади тестів.
33. Відпрацьовано навички командної роботи через GitHub.
34. Набуто досвіду роботи з документацією.
35. Навченося створювати цілісні проєкти від постановки задачі до її реалізації.
36. Удосконалено здатність аналізувати результати програмного виконання.
37. Навченося аналізувати доцільність вибору того чи іншого підходу в проектуванні.
38. Поглиблено розуміння модульного підходу в розробці ПЗ.
39. Здобуто навички автоматизованого тестування.
40. Отримано досвід інтеграції кількох модулів у одне застосування.
41. Зрозуміло, як розділити відповідальність між модулями.
42. Вивчено основи побудови бібліотек, зручних для повторного використання.
43. Навченося виявляти та виправляти логічні помилки у програмному коді.
44. Реалізовано повний цикл розробки програмного забезпечення.
45. Поглиблено знання мови програмування C++.
46. Удосконалено вміння реалізації алгоритмів.
47. Освоєно тестування edge-case ситуацій.
48. Засвоєно принципи модульного проєктування.
49. Удосконалено навички ведення звітної документації.
50. Сформовано вміння аналізувати та оцінювати результати лабораторної роботи.

Відповіді на контрольні запитання:

### **Яким оператором можна замінити тернарний** ?:**?**

Оператором if-else.

**Обґрунтування:** Тернарний оператор — це компактна форма умовного оператора if-else.

### **Пріоритет та асоціативність**

**Пріоритет** — порядок, у якому виконуються оператори (вищий виконується раніше).

**Асоціативність** — порядок виконання операторів з однаковим пріоритетом (зліва направо або справа наліво).

### **Область видимості змінних у циклах/умовах**

Змінні, оголошені **всередині тіла циклу або умов**, мають **локальну область видимості** — доступні лише в межах {}.

### **Асоціативність операторів:**

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

### **Коли доцільно використовувати тернарний оператор?**

Для **простого** вибору між двома значеннями.

Коли потрібно скоротити код до одного рядка.

### **Значення змінної після** cnt--;

cnt зменшиться на 1.

### **Чим відрізняється** const **змінна?**

**const** — це змінна, яку **не можна змінити** після ініціалізації.

**Правило:** оголошуйте const, якщо значення не повинне змінюватись у програмі.

### **Типи операндів логічних операторів:**

Можуть бути:

**Будь-які типи**, які **можна привести до** bool (наприклад, int, float, char).

### **Значення змінної** cnt **після:** bool cnt = !!0;

!!0 → !0 → true → !true → false  
→ cnt = false, тобто **0**.

### **Правило запису ініціалізації в** for **циклі:**

for (ініціалізація; умова; крок) {

// тіло циклу

}