Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 8

з навчальної дисципліни

“Базові методології та технології програмування”

РЕАЛІЗАЦІЯ СТАТИЧНИХ БІБЛІОТЕК

МОДУЛІВ ЛІНІЙНИХ 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

Мирончук А.А

ПЕРЕВІРИВ

викладач кафедри кібербезпеки   
та програмного забезпечення

Коваленко А. С.

Кропивницький – 2025

Мета роботи полягає у набутті ґрунтовних вмінь і практичних навичок застосування теоретичних положень методології модульного програмування, реалізації метода функціональної декомпозиції задач, метода модульного (блочного) тестування, представлення мовою програмування С++ даних скалярних типів, арифметичних і логічних операцій, потокового введення й виведення інформації, розроблення програмних модулів та засобів у кросплатформовому середовищі Code::Blocks (GNU GCC Compiler).

**Варіант №5**

**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 вимог та проектування архітектури для задачі 8.1**

1. **Аналіз вимог:**

Обчислення виразу для змінних x,y,z.

Виведення результатів у потік cout.

Реалізація функції s\_calculation(), яка буде обчислювати значення S.

Робота з десятковими та шістнадцятковими представленнями чисел.

Проектування архітектури:

Виділити функцію s\_calculation() у заголовковому файлі Modules/Прізвище.h.

Окремі функції для:

Виведення інформації про автора.

Логічного порівняння символів.

Форматованого виведення значень у різних системах числення.

Детальне проектування:

Написати реалізацію кожної функції у відповідних .cpp файлах.

Забезпечити коректне введення та обробку даних.

Лістинг файлу main.cpp проєкту ModulesMyronchuk:

#include <iostream>

#include <cmath>

#include <stdexcept>

using namespace std;

// Функція обчислення S з проміжними виводами

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

// Обчислюємо підкореневий вираз знаменника

double cosValue = cos(z + y \* z) + x \* x;

cout << "cos(z + yz) + x^2 = " << cosValue << endl;

if (cosValue <= 0) {

throw runtime\_error("Знаменник під коренем є недопустимим (менше або дорівнює нулю)!");

}

double denominator = sqrt(cosValue);

cout << "Знаменник: sqrt(cosValue) = " << denominator << endl;

// Обчислюємо підкореневий вираз чисельника

double absValue = fabs(y - 0.5 \* z);

cout << "abs(y - 0.5 \* z) = " << absValue << endl;

double sqrtAbsValue = sqrt(absValue);

cout << "sqrt(absValue) = " << sqrtAbsValue << endl;

double numerator = pow(2 \* z + 1, 2) - sqrtAbsValue;

cout << "Чисельник: (2z + 1)^2 - sqrt(absValue) = " << numerator << endl;

// Перевірка на ділення на нуль

if (denominator == 0) {

throw runtime\_error("Ділення на нуль: некоректні вхідні значення.");

}

double fraction = numerator / denominator;

cout << "Частка: numerator / denominator = " << fraction << endl;

double result = z + M\_PI \* fraction;

cout << "Результат: S = " << result << endl;

return result;

}

Лістинг файлу ModulesMyronchuk.h:

#ifndef MODULES\_MYRONCHUK\_H

#define MODULES\_MYRONCHUK\_H

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

#endif

Лістинг файлу main.cpp проєкту TestDrive:

#include <iostream>

#include <cmath>

#include <iomanip>

#include <stdexcept>

#include <clocale> // Підключаємо бібліотеку для setlocale

#include "ModulesMyronchuk.h"

// Функція для виконання тесту

void run\_test(double x, double y, double z, double expected, double tolerance = 1e-6) {

std::cout << "Тест: x = " << x << ", y = " << y << ", z = " << z << std::endl;

try {

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

std::cout << "Отриманий результат: " << std::fixed << std::setprecision(6) << result << std::endl;

if (fabs(result - expected) < tolerance) {

std::cout << "Статус: PASSED ✅\n" << std::endl;

} else {

std::cout << "Статус: FAILED ❌ (Очікувано: " << expected << ")\n" << std::endl;

}

} catch (const std::exception& e) {

std::cout << "Помилка виконання: " << e.what() << std::endl;

std::cout << "Статус: FAILED ❌ (Очікувався коректний результат)\n" << std::endl;

}

}

int main() {

// Встановлюємо українську локаль

setlocale(LC\_ALL, "Ukrainian");

std::cout << "=== Запуск тестового драйвера ===\n" << std::endl;

// Набір тест-кейсів (x, y, z, очікуваний результат)

run\_test(1.0, 2.0, 3.0, 511.927768);

run\_test(0.0, 1.0, 2.0, NAN);

run\_test(1.5, -2.0, 0.5, 4.94105);

run\_test(-1.0, 0.0, 1.0, 21.99196);

run\_test(2.0, 3.0, 4.0, 148.0907);

run\_test(3.0, 4.0, 5.0, 124.044159);

run\_test(-2.0, -1.0, 0.0, 0.0);

run\_test(0.5, 0.25, 0.75, 20.939914);

run\_test(1.0, -1.0, 1.0, 18.272274);

run\_test(2.0, -3.0, -1.0, -1.964394);

std::cout << "=== Тестування завершено ===" << std::endl;

return 0;

}

**Аналіз і постановка задачі 8.2**

**1. Вимоги до програми**

Програма повинна:

-Отримувати вхідні дані: три числа **x, y, z** та два символи **a, b**.

-Виводити:

-Прізвище та ім’я розробника зі знаком охорони авторського права.

-Результат логічного виразу **a + 1 ≥ b** у вигляді true або false.

-Значення **x, y, z** у десятковій та шістнадцятковій системах числення.

-Значення **S**, розраховане за допомогою функції s\_calculation() із -заголовкового файлу ModulesПрізвище.h.

**2. Аналіз задачі**

-Потрібно розробити окремі функції для кожного підзавдання:

-printDeveloperInfo() — виводить ім’я розробника.

-evaluateExpression(char a, char b) — обчислює логічний вираз і повертає true або false.

-printNumbersInDifferentBases(double x, double y, double z) — виводить числа у різних системах числення.

-calculateAndPrintS(double x, double y, double z) — обчислює та виводить значення S.

**3. Проектування архітектури**

Програма матиме:

**-Головну функцію (main)**, яка координує виклик допоміжних функцій.

**-Допоміжні функції** для виконання окремих підзадач.

**-Функцію s\_calculation()**, яка міститься у бібліотеці ModulesПрізвище.h.

**Аргументація досягнення цілей лабораторної роботи:**

1. Покращення навичок програмування на мові C++.
2. Закріплення розуміння та практичного використання математичних функцій у програмуванні.
3. Ознайомлення з бібліотекою cmath для виконання математичних обчислень.
4. Використання операторів cout і cin для введення та виведення даних у програмі.
5. Форматування виводу за допомогою бібліотеки iomanip.
6. Робота з заголовковими файлами (.h) для організації коду.
7. Вивчення модульної структури програмування для підвищення зручності та читабельності коду.
8. Відпрацювання написання та виклику функцій у програмі.
9. Освоєння методів перетворення чисел між десятковою та шістнадцятковою системами числення.
10. Використання операцій округлення значень у числових обчисленнях.
11. Практичне застосування тригонометричних функцій із бібліотеки cmath.
12. Робота з функціями для обчислення абсолютних значень чисел.
13. Визначення змінних та їх використання у програмі.
14. Використання директиви #include для підключення заголовкових файлів.
15. Виконання перевірок умов та застосування логічних операторів.
16. Ознайомлення з основами математичної логіки та їх застосування у програмуванні.
17. Використання арифметичних операторів (+, -, \*, /, %).
18. Закріплення концепції інкапсуляції у функціях.
19. Робота зі стандартними бібліотеками мови C++.
20. Оптимізація коду за допомогою namespace std.
21. Використання різних типів змінних (int, double, char).
22. Форматування чисел за допомогою setprecision і fixed.
23. Відпрацювання концепції "чорного ящика" у функціях.
24. Використання структурного підходу до програмування та розбиття коду на модулі.
25. Аналіз результатів логічних виразів.
26. Використання операторів порівняння (==, !=, <, >).
27. Робота з оператором return для повернення значень з функцій.
28. Аналіз особливостей арифметичних операцій між різними типами даних.
29. Налагодження програми за допомогою cout.
30. Робота з константами у програмі.
31. Використання параметрів у функціях для передачі значень.
32. Створення заголовкових файлів для багаторазового використання коду.
33. Робота з дробовими числами (double).
34. Використання модульного підходу до програмування.
35. Опрацювання способів передачі змінних у функції.
36. Застосування бібліотеки iomanip для управління форматуванням виводу.
37. Використання коментарів (// та /\* ... \*/).
38. Написання зрозумілого та добре документованого коду.
39. Дотримання стилю кодування.
40. Закріплення принципів побудови математичних виразів.
41. Практика у використанні обчислень за формулами.
42. Використання функції fabs() для знаходження абсолютного значення.
43. Оголошення та виклик функцій у відповідності до стандартів.
44. Робота з логічними операторами (&&, ||, !).
45. Ознайомлення зі структурним програмуванням.
46. Аналіз функцій, що працюють з різними типами даних.
47. Відпрацювання операторів введення/виведення.
48. Використання оператора sizeof для оцінки пам’яті змінних.
49. Вибір оптимальних типів даних.
50. Написання продуктивного та ефективного коду.

**Відповіді до контрольних запитань:**

**Мета й задачі процесів проектування ПЗ відповідно до ISO/IEC 12207 або ISO/IEC/IEEE 15288:2016**  
Мета – стандартизувати процеси життєвого циклу ПЗ, забезпечити якість і ефективність розробки.  
Задачі:

-Визначення вимог

-Проектування архітектури

-Реалізація, тестування та інтеграція

-Супровід і модернізація

**Відмінність функції від модуля в С/С++**  
Функція – це блок коду, що виконує певне завдання й може повертати значення.  
Модуль – це набір функцій і змінних, об'єднаних у файлі (наприклад, .h + .cpp).

**Відмінність функції main від інших функцій**

-Є точкою входу в програму

-Має фіксований формат (int main())

-Використовується для керування виконанням програми

**Призначення маніпуляторів у C++**  
Використовуються для форматування виводу в cout, наприклад:

-setw(n) – задає ширину виводу

-setprecision(n) – задає точність дробових чисел

-ixed – фіксований формат виводу

**Призначення заголовкового файлу під час препроцесингу**

-Містить оголошення функцій та змінних

-Дозволяє розділяти код на модулі для повторного використання

-Підключається через #include "file.h" або #include <iostream>

**Що використано з <iostream> та для яких функцій**

-std::cout – для виводу даних

-std::cin – для введення даних

-std::endl – для переходу на новий рядок

**Висновок:**

Під час виконання лабораторної роботи було закріплено навички роботи з функціями, математичними операціями та заголовковими файлами. Опановано передавання параметрів у функції, форматування виводу, конвертацію чисел у різні системи числення. Важливим аспектом стало розуміння правильної структури коду та використання заголовкових файлів. Завдяки виконанню завдань лабораторної роботи вдалося покращити навички програмування та навчитися ефективніше писати код.