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

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 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 №3**

**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 8.1**

**Вхідні параметри:**

* **x, y, z** (типу double).
* Зауважимо, що у формулі використовується лише **x** і **z**, а **y** може бути або не використовуватись (залежно від варіанту завдання).

Формула для обчислення S:

**Обмеження області визначення:**

* Логарифм визначений, якщо .
* При цьому, щоб не виникло ділення на нуль або негативного значення під квадратним коренем, потрібно забезпечити, що .
  + Оскільки , то коли .

Отже, основна умова для **x**:

Якщо ця умова не дотримується, функція може повертати NAN або сигналізувати про помилку.

Проектування архітектури модуля задачі 8.1

Інтерфейс функції:

* Прототип (заголовковий файл s\_calculation.h):

#ifndef S\_CALCULATION\_H

#define S\_CALCULATION\_H

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

#endif

Опис функції s\_calculation:

* Перевіряє, чи . Якщо ні, повертає NAN.
* Обчислює вираз:

1. Обчислюємо як log10(x-3).
2. Обчислюємо вираз під квадратним коренем:
3. Обчислюємо квадратний корінь, потім синус від нього.
4. Помножуємо синус на 3, підносимо результат до квадрату.
5. Додаємо .

Тест-сьют до задачі 8.1 наведено у файлі \Lab8\TestSuite\TS\_8\_1.doc.

Результати тестування s\_calculation зі статичної бібліотеки libModulesZaritskyi.а тестовим драйвером:

Test case #1: s\_calculation(5, 1, 2) = 1.20824 == 1.20824 --> passed

Test case #2: s\_calculation(4.1, 0, 0) = 3.68345 == 3.68345 --> passed

Test case #3: s\_calculation(10, 0, -4) = -1.99745 == -1.99745 --> passed

Test case #4: s\_calculation(6, 3, 10) = 13.46949 == 13.46949 --> passed

Test case #5: s\_calculation(100, 2, 20) = 13.90284 == 13.90284 --> passed

Test case #6: s\_calculation(4.0001, 5, 5) = 8.77684 == 8.77684 --> passed

Test case #7: s\_calculation(3.9, 1, 10) = nan == nan --> passed

Test case #8: s\_calculation(-5, 0, 0) = nan == nan --> passed

Test case #9: s\_calculation(7.5, 0, 0) = 4.00157 == 4.00157 --> passed

Test case #10: s\_calculation(50, 1, -10) = 2.44739 == 2.44739 --> passed

Лістинг файлу main.cpp проєкту ModulesZaritskyi:

#include <cmath>

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;

}

Лістинг файлу ModulesZaritskyi.h:

#ifndef MODULESZARITSKYI\_H\_INCLUDED

#define MODULESZARITSKYI\_H\_INCLUDED

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

#endif // MODULESZARITSKYI\_H\_INCLUDED

Лістинг файлу main.cpp проєкту TestDriver:

#include <iostream>

#include <sstream>

#include <iomanip>

#include <cmath>

#include "ModulesZaritskyi.h"

using namespace std;

string doubleToString(double value, int precision = 5) {

ostringstream oss;

oss << fixed << setprecision(precision) << value;

return oss.str();

}

void test\_s\_calculation() {

double x[10] = {5.0, 4.1, 10.0, 6.0, 100.0, 4.0001, 3.9, -5.0, 7.5, 50.0};

double y[10] = {1.0, 0.0, 0.0, 3.0, 2.0, 5.0, 1.0, 0.0, 0.0, 1.0};

double z[10] = {2.0, 0.0, -4.0, 10.0, 20.0, 5.0, 10.0, 0.0, 0.0, -10.0};

double expected\_results[10] = {1.20824, 3.68345, -1.99745, 13.46949, 13.90284, 8.77684, NAN, NAN, 4.00157, 2.44739};

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

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

string resultStr = doubleToString(result);

string expectedStr = doubleToString(expected\_results[i]);

bool passed = false;

// Для NaN використовується isNaN, оскільки NaN != NaN

if (std::isnan(result) && std::isnan(expected\_results[i])) {

passed = true;

} else if (resultStr == expectedStr) {

passed = true;

}

cout << "Test case #" << i+1 << ": s\_calculation("

<< x[i] << ", " << y[i] << ", " << z[i] << ") = "

<< resultStr << " == " << expectedStr

<< " --> " << (passed ? "passed" : "failed") << endl;

}

}

int main() {

test\_s\_calculation();

return 0;

}

**Аналіз і постановка задачі 8.2**

**Вхідні дані:**

* Числові значення: x, y, z.
* Значення для логічного виразу: наприклад, цілі числа a та b (якщо вводяться символи, їх можна трактувати за ASCII-кодом або сприймати як числові значення).

**Вихідні дані:**

За допомогою стандартного потоку виведення (cout) потрібно послідовно відобразити:

1. **(8.2.1)** Прізвище та ім'я розробника із знаком охорони авторського права, наприклад: Zaritskyi Viktor ©
2. **(8.2.2)** Результат логічного виразу у вигляді тексту «true» або «false». Вираз має вигляд:

Наприклад, якщо a=5 і b=3, то перевіряється, чи 5+1>∣3−2∣ (тобто, 6>1) і відповідно виводиться "true".

1. **(8.2.3)** Значення x, y, z у десятковому та шістнадцятковому форматах, а також значення S, яке обчислюється функцією s\_calculation() з заголовкового файлу ModulesZaritskyi.h.

## Проектування архітектури програмного забезпечення задачі 8.2

1. **Функція для 8.2.1 (Developer Info):**
   * Функція, яка повертає рядок (string) з ім'ям розробника і знаком авторського права.
2. **Функція для 8.2.2 (Логічний вираз):**
   * Функція, яка приймає в якості параметрів a і b (типу int або char) і повертає рядок "true" або "false" залежно від того, чи виконується вираз:
   * Наприклад:

string evaluateLogicalExpression(int a, int b) {

return ((a + 1) > std::abs(b - 2)) ? "true" : "false";

}

1. **Функція для 8.2.3 (Виведення значень і результату S):**
   * Функція, яка приймає x, y, z, форматує і виводить ці значення як у десятковій, так і в шістнадцятковій системах числення (для типу double можна використати маніпулятори dec/hexfloat) та обчислює значення S через виклик s\_calculation(x, y, z).
   * Наприклад:

void printValuesAndResult(double x, double y, double z) {

std::cout << "Values (decimal): x = " << x << ", y = " << y << ", z = " << z << std::endl;

std::cout << "Values (hexadecimal): x = " << std::hexfloat << x

<< ", y = " << y << ", z = " << z << std::endl;

std::cout << std::defaultfloat; // повертаємо формат до звичайного

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

std::cout << "S = " << S << std::endl;

}

**Аргументи досягнення мети лабораторної роботи:**

1. Завдання реалізовано за принципом модульного програмування, що спрощує налагодження та підтримку.
2. Створена статична бібліотека дозволяє повторно використовувати код у різних проектах.
3. Прототип функції, визначений у заголовковому файлі, сприяє чіткому розмежуванню інтерфейсу та реалізації.
4. Кожен модуль та функція супроводжуються коментарями, що полегшує розуміння логіки програми.
5. Код написаний із застосуванням стандартних бібліотек (cmath, iostream, iomanip), що забезпечує кросплатформеність.
6. Перевірки вхідних даних (наприклад, перевірка x > 4) дозволяють уникнути арифметичних помилок, що підвищує надійність програми.
7. Функція s\_calculation використовує математичні функції (sin, log10, sqrt), що демонструє практичне застосування бібліотеки cmath.
8. Розроблений тестовий драйвер перевіряє роботу функції на різних наборах вхідних даних, що забезпечує якість коду.
9. Перевірка числових результатів з використанням epsilon гарантує правильність роботи з числами з плаваючою крапкою.
10. Демонструється здатність форматувати вивід, що важливо для звітності.
11. Всі обчислення виконуються всередині окремої функції, що покращує структурованість коду.
12. Всі пункти методичних рекомендацій виконано, що підтверджує досягнення поставленої мети.
13. Використання Git-репозиторію забезпечує відстеження змін і історію розробки.
14. Розділення прототипів і реалізації покращує організацію коду та спрощує співпрацю.
15. Керування форматуванням виводу забезпечує чіткість і зрозумілість представленої інформації.
16. Проект реалізовано з урахуванням вимог міжнародних стандартів, що підвищує якість ПЗ.
17. Використання ефективних алгоритмів дозволяє скоротити час обчислень.
18. Розглянуто граничні значення вхідних даних, що свідчить про ретельний аналіз завдання.
19. Використання умовних операторів робить код компактним та зрозумілим.
20. Обробка випадків, коли результат обчислень є NAN, запобігає непередбачуваним помилкам.
21. Тест-сьют містить як позитивні, так і негативні кейси, що забезпечує повне тестування функціональності.
22. Логічні вирази правильно побудовані, що дозволяє отримувати очікувані результати.
23. Протоколювання тестових результатів сприяє аналізу і вдосконаленню розробки.
24. Окреме тестування кожної функції полегшує виявлення та локалізацію помилок.
25. Можливість автоматичного запуску тестів сприяє підвищенню ефективності перевірки коду.
26. Використання операцій множення, ділення та піднесення до степеня демонструє розуміння математичних принципів.
27. Логічний вираз у завданні перевіряється коректно, що підтверджує розуміння логіки програмування.
28. Всі завдання з методички виконано, що засвідчує досягнення мети роботи.
29. Реалізація проекту сприяє набуттю практичних навичок роботи з C++ та Code::Blocks.
30. Ретельна перевірка функцій підтверджує правильність їх роботи в різних умовах.
31. Забезпечує точність обчислень та використання математичних функцій.
32. Розроблена архітектура дозволяє легко додавати нові модулі або розширювати функціонал.
33. Чіткий і зрозумілий вивід числових даних сприяє кращій інтерпретації результатів.
34. Реалізація роботи з типами double та char демонструє універсальність підходу.
35. Коректне використання #include забезпечує правильну компіляцію проекту.
36. Розділення коду на окремі проекти (бібліотека, заголовковий файл, тестовий драйвер) сприяє організованості роботи.
37. Коректне збирання статичної бібліотеки та її підключення до тестового драйвера підтверджує інтеграцію компонентів.
38. Незважаючи на використання процедурного підходу, застосовано принципи інкапсуляції та абстракції.
39. Використання стандартних бібліотек гарантує портативність коду.
40. Програма коректно реагує на недопустимі вхідні значення, повертаючи NAN.
41. Умовні оператори дозволяють правильно обробляти різні логічні сценарії.
42. Розділення коду на модулі значно спрощує процес виявлення і виправлення помилок.
43. Тест-кейси охоплюють як нормальні, так і крайні ситуації, що підвищує довіру до результатів.
44. Демонструється вміння працювати з різними системами числення.
45. Результати роботи програми чітко структуровані та легко інтерпретуються.
46. Протокол тестування містить детальну інформацію про кожен тест-кейс.
47. Раннє тестування дозволило виявити і усунути потенційні проблеми у функціоналі.
48. Детальний аналіз і тестування забезпечують високий рівень якості кінцевого продукту.
49. Виконання всіх етапів (від аналізу до тестування) свідчить про комплексне розуміння методології розробки.
50. Завдання виконано відповідно до методичних рекомендацій, що підтверджує набуття практичних навичок і досягнення мети курсу «Базові методології та технології програмування».

**Відповіді на контрольні запитання:**

**1. Мета та задачі процесів проектування ПЗ за стандартами ISO/IEC 12207 / ISO/IEC/IEEE 15288:**

* Забезпечення єдиного підходу до організації процесів розробки, експлуатації, обслуговування та утилізації ПЗ (або систем в цілому).
* Створення системних, відтворюваних і контрольованих процесів для забезпечення високої якості кінцевого продукту.
* Чітке визначення, документування і контроль вимог до ПЗ протягом усього життєвого циклу.
* Передбачення циклічних процесів (аналіз, проектування, валідація, тестування) для постійного вдосконалення продукту.
* Забезпечення ефективної комунікації між розробниками, замовниками, користувачами та іншими учасниками процесу.
* Ідентифікація, оцінка та управління ризиками протягом усього циклу розробки.
* Систематичне оформлення документації для забезпечення відтворюваності та підтримки проекту.
* Забезпечення гнучкості процесів для реагування на зміни вимог і умов проекту.

**2. Відмінність функції від модуля у C/С++:**

1. **Функція:**

* Є окремою одиницею коду, яка виконує певну операцію, приймаючи аргументи і повертаючи результат.
* Має чітко визначений інтерфейс (прототип) та реалізацію, може бути викликана з будь-якої частини програми.
* Спрямована на вирішення конкретного завдання або обчислення.

1. **Модуль:**

* Є логічним блоком або групою пов’язаних функцій, класів і змінних, які об'єднані для реалізації певної функціональності системи.
* Забезпечує інкапсуляцію та розмежування інтерфейсу і реалізації, що сприяє організації коду та його повторному використанню (часто реалізується як статична або динамічна бібліотека).
* Має більший масштаб: модуль структуровано організовує множину функцій, забезпечуючи цілісність і ізоляцію окремих аспектів програми.



