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

ВИКОНАВ

Студент(КБ-24)

Габур А.О

ПЕРЕВІРИВ

викладач 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.

Варіант №10

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

Потрібно реалізувати функцію s\_calculation(x, y, z), яка обчислює значення виразу згідно з формулою варіанту 10. Функція має бути розміщена в окремому модулі у складі статичної бібліотеки libModulesПрізвище. Тип даних — double.

Завдання демонструє принципи модульного програмування та функціональної декомпозиції. Математичні обчислення реалізуються через стандартні функції мови C++ з бібліотеки <cmath>. Результат використовується в основній програмі в іншій частині завдання (8.2).

S=In(x-y)+

Лістинг до задачі 8.1

#include <iostream>

#include <iomanip>

#include <cmath>

#include <windows.h>

using namespace std;

int main() {

    SetConsoleOutputCP(65001);

    SetConsoleCP(65001);

    double x, y, z, S;

    cout << "Введіть значення x, y, z: ";

    cin >> x >> y >> z;

    if (x <= y) {

        cout << "Помилка: вираз ln(x - y) не визначений при x ≤ y." << endl;

        return 1;

    }

    S = log(x - y) + pow(cos(x), 2) - fabs(z);

    cout << fixed << setprecision(4);

    cout << "Результат обчислення S = " << S << endl;

    system("pause");

    return 0;

}

Результат:  
Введіть значення x, y, z: 10

2

3

Результат обчислення S = -0.2165

Аналіз і постановка задачі 8.2

У задачі передбачено обробку вхідних даних: трьох цілих чисел x, y, z та двох символів a і b. Необхідно реалізувати консольний застосунок, який виконує послідовно три підзавдання:

8.2.1.

Вивести прізвище та ім’я розробника програми із вказівкою авторського права (символ ©), як підтвердження авторства.

8.2.2.

Розрахувати значення логічного виразу a + 10 ≥ b та вивести результат у числовому вигляді:

1, якщо твердження істинне;

0 — якщо хибне.

Символи a і b при цьому мають бути інтерпретовані як їх ASCII-коди.

8.2.3.

Вивести значення змінних x, y, z у:

десятковій системі числення,

шістнадцятковій системі числення.

Також потрібно обчислити значення виразу S за допомогою функції s\_calculation(), яка оголошена у заголовковому файлі Modules/Прізвище.h.

Результат кожної підзадачі має бути виведений у потік cout у вигляді окремих функцій.

Лістинг main.cpp:

#include <iostream>

#include <iomanip>

#include <windows.h>

#include "Modules/Gabur.h"

using namespace std;

int main() {

    SetConsoleOutputCP(65001);

    SetConsoleCP(65001);

    double x, y, z;

    char a, b;

    cout << "Введіть значення x, y, z: ";

    cin >> x >> y >> z;

    cout << "Введіть символи a та b: ";

    cin >> a >> b;

    if (x <= y) {

        cout << "Помилка: вираз ln(x - y) не визначений при x ≤ y." << endl;

        return 1;

    }

    // 8.2.1

    printAuthor();

    // 8.2.2

    printLogicalResult(a, b);

    // 8.2.3

    printValues(x, y, z);

    // Обчислення S

    double S = calculateS(x, y, z);

    cout << fixed << setprecision(4);

    cout << "Результат обчислення S = " << S << endl;

    system("pause");

    return 0;

}

Лістинг Gabur.cpp:

#include <cmath>

#include <iostream>

#include <iomanip>

#include "Gabur.h"

using namespace std;

// 8.1 - Обчислення за формулою

double calculateS(double x, double y, double z) {

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

}

// 8.2.1 - Автор

void printAuthor() {

    cout << "© Габур Артем" << endl;

}

// 8.2.2 - Логічний вираз

void printLogicalResult(char a, char b) {

    int a\_num = static\_cast<int>(a);

    int b\_num = static\_cast<int>(b);

    cout << "Результат логічного виразу a + 10 ≥ b: " << (a\_num + 10 >= b\_num) << endl;

}

// 8.2.3 - Значення x, y, z

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

    cout << "x = " << x << " (hex: " << hex << static\_cast<int>(x) << dec << ")\n";

    cout << "y = " << y << " (hex: " << hex << static\_cast<int>(y) << dec << ")\n";

    cout << "z = " << z << " (hex: " << hex << static\_cast<int>(z) << dec << ")\n";

}

Лістинг Gabur.h:

#ifndef GABUR\_H

#define GABUR\_H

double calculateS(double x, double y, double z);

void printAuthor();

void printLogicalResult(char a, char b);

void printValues(double x, double y, double z);

#endif

Результат:  
Введіть значення x, y, z: 10

2

3

Введіть символи a та b: a

b

© Габур Артем

Результат логічного виразу a + 10 ≥ b: 1

x = 10 (hex: a)

y = 2 (hex: 2)

z = 3 (hex: 3)

Результат обчислення S = 7.5822

Аргументи досягнення мети лабораторної роботи

1. Завдання реалізовано за принципом модульного програмування, щоспрощує налагодження та підтримку.

2. Створена статична бібліотека дозволяє повторно використовувати код урізних проектах.

3. Прототип функції, визначений у заголовковому файлі, сприяє чіткомурозмежуванню інтерфейсу та реалізації.

4. Кожен модуль та функція супроводжуються коментарями, що полегшуєрозуміння 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