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 СТАТИСТИЧНИХ БІБЛІОТЕК МОДУЛІВ ЛІНІЙНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

ЗАВДАННЯ ВИДАВ
доцент кафедри кібербезпеки
та програмного забезпечення
Доренський О. П.
https://github.com/odorenskyi/

ВИКОНАВ
студент академічної групи КН-24
Мельник Дмитро

ПЕРЕВІРИЛА
викладачка кафедри кібербезпеки
та програмного забезпечення
Анастасія Коваленко

Кропивницький – 2025

ТЕМА: РЕАЛІЗАЦІЯ СТАТИСТИЧНИХ БІБЛІОТЕК МОДУЛІВ ЛІНІЙНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

МЕТА: набуття 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 8

— *ЗАДАЧА 8.1* —

За значеннями x, y, z обчислюється S:

$$S = \ln|\sin z| + \frac{\frac{1}{2}x^2 - \sqrt{|(y+z)^2 - x^5|}}{10\pi \cdot z^4}.$$

— *ЗАДАЧА 8.2* —

За послідовними запитами вводяться числа x, y, z та символи a і b.
Вивести (включити у потік STL — cout)*:

8.2.1. Прізвище та ім'я розробника програми зі знаком охорони авторського права «©» (від англ. copyright);

8.2.2. Результат логічного виразу в числовому вигляді (`1`/`0`):

$$a + 1 \le b \ ?$$

8.2.3. Значення x, y, z в десятковій і шістнадцятковій системах числення; S, що обчислюється функцією `s_calculation()` заголовкового файлу Modules*Прізвище*.**h**.

* Підзадачі 8.2.1–8.2.3 варто реалізувати у вигляді функцій, результат виконання яких включається у вихідний потік `cout` за допомогою оператора вставки `<<` (наприклад, “ `cout << YourFunc(a,b);` ”).

Задача 8.1

За значеннями x, y, z обчислюється S:

$$S = \ln|\sin z| + \frac{\frac{1}{2}x^2 - \sqrt{|(y+z)^2 - x^5|}}{10\pi \cdot z^4}.$$

Лістинг програми ModulesMelnyk

```
#include <iostream>
#include <cmath>

// Функція для обчислення значення математичного виразу за заданими параметрами
double s_calculation (double x, double y, double z){

const double Pi = 3.141592653589793238; // Константа Пі з високою точністю

    if(z == 0){ // Перевірка на ділення на нуль
        return NAN;
    };

    if(sin(z) == 0 || fabs(sin(z)) < 1e-10){ // Перевірка на логарифм від нуля або
дуже малого числа
        return NAN;
    }

    double sin_value = sin(z);
    double first = log(fabs(sin_value)); // Обчислення логарифма від модуля синуса

    double y_plus_z_squared = pow(y + z, 2); // Обчислення квадрата суми y та z
    double x_fifth = pow(x, 5); // Обчислення x у п'ятому степені
    double diff = y_plus_z_squared - x_fifth; // Різниця для підкореневого виразу


    if(diff < 0 && fabs(diff) > x_fifth) { // Перевірка на коректність значення під
коренем
        return NAN;
    }

    double top = 0.5 * pow(x, 2) - sqrt(fabs(diff)); // Обчислення чисельника дробу
    double bottom = 10 * Pi * pow(z, 4); // Обчислення знаменника дробу

    double result = first + top / bottom; // Обчислення кінцевого результату

    return result;
}
```

Функція s_calculation обчислює математичний вираз, перевіряючи можливі помилки (ділення на нуль, логарифм від нуля, некоректний корінь). Вона використовує логарифм, піднесення до степеня та квадратний корінь, а потім обчислює вираз: $S = \ln|\sin z| + \frac{\frac{1}{2}x^2 - \sqrt{|(y+z)^2 - x^5|}}{10\pi \cdot z^4}.$

Файл ModulesMelnyk.h

```
#ifndef MODULESMELNYK_H_INCLUDED
#define MODULESMELNYK_H_INCLUDED

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

#endif // MODULESMELNYK_H_INCLUDED
```

Лістинг файлу TestDriver

```
#include <iostream>
#include <windows.h>
#include "ModulesMelnyk.h"

using namespace std;

int main()
{
    SetConsoleOutputCP(65001);
    SetConsoleCP(65001);

    double x, y, z;

    cout << "Введіть x" << endl;
    cin >> x;
    cout << "Введіть y" << endl;
    cin >> y;
    cout << "Введіть z" << endl;
    cin >> z;

    double Result = s_calculation(x, y ,z);

    cout << "Відповідь: " << Result;
}
```

1. **Налаштовує консоль для роботи з кодуванням UTF-8, щоб коректно відображати український текст.**
2. **Приймає три числа (x, y, z) від користувача.**
3. **Викликає функцію s_calculation(x, y, z) (ймовірно, з бібліотеки ModulesMelnyk.h), яка виконує обчислення.**
4. **Виводить результат обчислення на екран.**

Лістинг програми Melnyk_task

```
#include <iostream>
#include <windows.h>
#include <iomanip>
#include <cstdint>
#include "ModulesMelnyk.h"

using namespace std;

void developer_information() {
    cout << "Мельник Дмитро ©, Усі права захищені." << endl;
}

double logical(int a, int b){
    cout << "Результат логічного виразу: ";

    if (a + 1 <= b){
        cout << "1";
        return 1;
    }else{
        cout << "0";
        return 0;
    }
}

void numbers_in_systems(double x, double y, double z){
    union {
        int d;
        int64_t i;
    } x_conv = {x}, y_conv = {y}, z_conv = {z};

    cout << "\nЗначення змінних в десятковій та шістнадцятковій системах числення:"
<< endl;
    cout << "x = " << fixed << setprecision(2) << x << " (десяткова) = 0x"
         << hex << uppercase << x_conv.i << " (шістнадцяткова)" << dec << nouppercase
<< endl;
    cout << "y = " << fixed << setprecision(2) << y << " (десяткова) = 0x"
         << hex << uppercase << y_conv.i << " (шістнадцяткова)" << dec << nouppercase
<< endl;
    cout << "z = " << fixed << setprecision(2) << z << " (десяткова) = 0x"
         << hex << uppercase << z_conv.i << " (шістнадцяткова)" << dec << nouppercase
<< endl;

    double s = s_calculation(x, y, z);
    cout << "\n\nРезультат обчислення функції s_calculation: " << fixed <<
setprecision(6) << s << endl;
}

int main()
{
    SetConsoleOutputCP(65001);
```

```
    SetConsoleCP(65001);

    double x, y, z;
    int a, b;

    cout << "Введіть x: ";
    cin >> x;
    cout << "Введіть y: ";
    cin >> y;
    cout << "Введіть z: ";
    cin >> z;
    cout << "Введіть a: ";
    cin >> a;
    cout << "Введіть b: ";
    cin >> b;

    developer_information();
    cout << "\n";
    logical(a, b);
    cout << "\n";
    numbers_in_systems(x, y, z);

    return 0;
}
```

Принцип роботи програми:

1. **Налаштування кодування**:
   - Програма встановлює UTF-8 для коректного відображення українського тексту в консолі.
2. **Ввід даних**:
   - Користувач вводить три числа (x, y, z) та два цілих числа (a, b).
3. **Вивід інформації про розробника**:
   - Викликається функція developer_information(), яка виводить інформацію про автора програми.
4. **Перевірка логічного виразу**:
   - Функція logical(a, b) перевіряє умову (a + 1 <= b), виводить 1, якщо вона виконується, і 0 — якщо ні.
5. **Перетворення чисел у різні системи числення**:
   - Функція numbers_in_systems(x, y, z) виводить введені числа у десятковій та шістнадцятковій системах.
   - Викликається функція s_calculation(x, y, z), яка обчислює певне значення та виводить його результат.
6. **Завершення програми**.

50 Аргементів

1. **Статичні бібліотеки забезпечують компактність коду при компіляції.**
2. **Модульна структура дозволяє розділити код на логічні компоненти.**
3. **Функціональна декомпозиція задач підвищує читабельність коду.**
4. **Code::Blocks надає зручне середовище для роботи з проектами модульного типу.**
5. **Використання GitHub для зберігання коду сприяє контролю версій.**
6. **Методологія модульного програмування покращує структуру коду.**
7. **Реалізація власних бібліотек розширює практичні навички програмування.**
8. **Статичні бібліотеки пришвидшують процес компіляції великих проектів.**
9. **Модульне тестування спрощує пошук помилок у програмному коді.**
10. **Крос-платформність коду збільшує його практичну цінність.**
11. **Розробка консольних застосунків формує розуміння базових принципів введення-виведення.**
12. **Робота з Git репозиторієм розвиває навички командної розробки.**
13. **Статичні бібліотеки зменшують залежності між різними частинами коду.**
14. **Модульне програмування сприяє повторному використанню коду.**
15. **Використання шаблону тестового набору формує навички автоматизованого тестування.**
16. **C++ як мова реалізації дозволяє використовувати об'єктно-орієнтований підхід.**
17. **Розв'язання лінійних обчислювальних задач формує алгоритмічне мислення.**
18. **Потокове введення-виведення в C++ спрощує роботу з даними.**
19. **Розробка власної бібліотеки підвищує рівень розуміння структури програмних проектів.**
20. **GNU GCC Compiler забезпечує високу якість оптимізації коду.**
21. **Іменування бібліотеки за прізвищем студента забезпечує унікальність реалізації.**
22. **Створення файлів заголовків сприяє кращому структуруванню коду.**
23. **Модульна архітектура полегшує подальше розширення функціоналу програми.**
24. **Статичні бібліотеки забезпечують незалежність проекту від зовнішніх залежностей.**
25. **Використання арифметичних і логічних операцій розвиває навички алгоритмізації.**

26. **Реалізація скалярних типів даних поглиблює розуміння системи типів C++.**
27. **Документування коду вдосконалює навички технічної комунікації.**
28. **Робота з бінарними файлами розширює знання про файлову систему.**
29. **Створення власних функцій для вирішення конкретних задач посилює практичні навички.**
30. **Тестування модулів допомагає виявити логічні помилки на ранніх етапах.**
31. **Розділення інтерфейсу та імплементації підвищує якість коду.**
32. **Використання заголовкових файлів полегшує інкапсуляцію даних.**
33. **Обробка конкретних задач формує навички прикладного програмування.**
34. **Тестовий набір Artifact_TEST_SUITE_lab.doc забезпечує стандартизацію перевірки.**
35. **Функціональна декомпозиція сприяє кращому розумінню складних алгоритмів.**
36. **Реалізація математичних функцій поглиблює знання чисельних методів.**
37. **Обробка скалярних типів даних формує розуміння представлення даних у пам'яті.**
38. **Консольний застосунок спрощує демонстрацію результатів роботи модулів.**
39. **Робота з бібліотеками формує навички використання сторонніх компонентів.**
40. **Розділення проекту на модулі сприяє паралельній розробці.**
41. **Статична лінковка спрощує розгортання програмного забезпечення.**
42. **Використання власного облікового запису на GitHub формує цифрове портфоліо студента.**
43. **Написання функцій для конкретних задач розвиває аналітичне мислення.**
44. **Створення бібліотек з конкретним призначенням підвищує спеціалізацію коду.**
45. **Блочне тестування дозволяє локалізувати помилки в конкретних модулях.**
46. **Кросплатформове середовище Code::Blocks забезпечує гнучкість розробки.**
47. **Реалізація потокового введення-виведення формує навички роботи з I/O.**
48. **Командна робота через Git формує навички співпраці в розробці.**
49. **Статичні бібліотеки забезпечують швидкодію програмного забезпечення.**

50. **Практичне застосування теоретичних положень закріплює навчальний матеріал.**

Відповіді на контрольні запитання:

1. **У відповідності до міжнародного стандарту ISO/IEC 12207 (або ISO/IEC/IEEE 12207:2016) мета і задачі процесів проектування програмного забезпечення полягають у:**
    - Розробці детального проекту програмного забезпечення, що реалізує визначені вимоги
    - Створенні чіткої архітектури програмного продукту з визначеними компонентами
    - Визначенні інтерфейсів між компонентами програми та із зовнішніми системами
    - Забезпеченні технічної узгодженості між вимогами та проектними рішеннями
    - Встановленні основи для верифікації та валідації програмного забезпечення
    - Створенні основи для подальшої реалізації, тестування та підтримки програмного продукту
    - Оптимізації проектних рішень з урахуванням технічних та економічних обмежень
2. **Функція мови програмування C/C++ відрізняється від модуля тим, що:**
    - Функція є базовою одиницею програмного коду, яка виконує певну операцію або набір операцій і може повертати значення, тоді як модуль є більшою структурною одиницею, що об'єднує логічно пов'язані функції, класи, дані та інші елементи програми
    - Функція має чітко визначений інтерфейс з параметрами та типом повернення, а модуль забезпечує інкапсуляцію та контроль доступу до групи взаємопов'язаних функцій та даних
    - Функція зазвичай виконує одну конкретну задачу, тоді як модуль реалізує більш високорівневу функціональність через набір взаємодіючих функцій
    - У C/C++ функція має локальну область видимості для своїх змінних, а модуль може створювати власний простір імен для уникнення конфліктів імен
    - Модуль сприяє повторному використанню коду на більш високому рівні абстракції, ніж окрема функці