Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

# ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 8

з навчальної дисципліни “Базові методології та технології програмування”

Реалізація статичних бібліотек модулів лінійних обчислювальних процесів

ВИКОНАВ

студент академічної групи

КІ-21-1 Лісевич Д.С.

ПЕРЕВІРИВ

викладач кафедри кібербезпеки

та програмного забезпечення

\_\_\_\_\_\_\_\_\_  П.С. Усік

Кропивницький – 2022

# ЛАБОРАТОРНА РОБОТА №8

**Тема:** Реалізація статичних бібліотек модулів лінійних обчислювальних процесів

**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 9**

Завдання:

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.

ХІД РОБОТИ

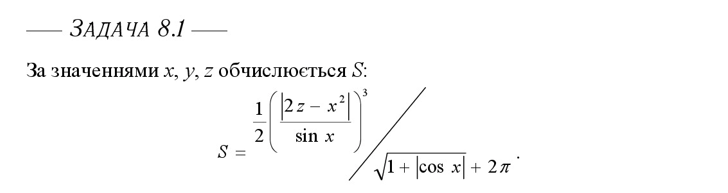
****

Рисунок 1.1 – Умова задачі 8.1

**Строга постанова задачі:**

* Вхідні дані: x, z – числа;
* Вихідні дані: S – результат;

**Проектування модуля:**

Модуль ModulesLisevych складається з двох частин: заголовковий файл(Header File) та вихідний код(s\_calculation).

Тестовий драйвер TestDriver створюється за для виведення вхідних значень, очікуваного результату, дійсного результату та статусу виконання тестового випадку (Passed / Failed).

Вихідний код TestDriver знаходиться у додатку В.

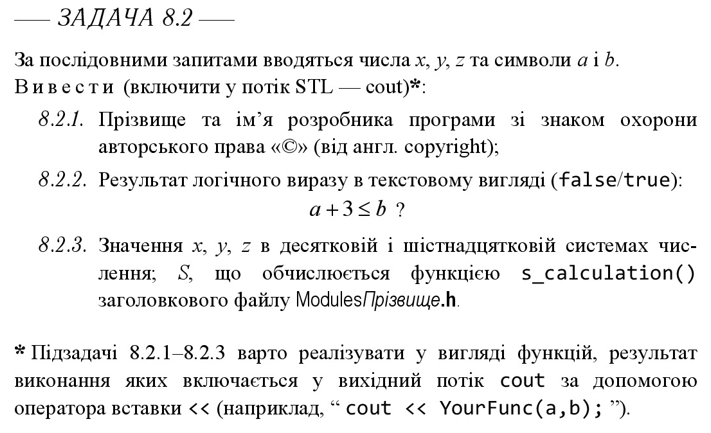


Рисунок 1.2 – Умова задачі 8.2

**Строга постанова задачі:**

* Вхідні дані: x, z – числа; a, b – символьні літерали;
* Вихідні дані: ім’я розробника із символом copyright; результат логічного виразу; значення x, z в десятковій та шістнадцятковій системі числення; результат функції s\_calculation.

**Процедурна алгоритмізація:**

Використовуємо модуль ModulesLisevych з попереднього проекту.

Функція copyrightHolder виводить ПІБ розробника зі знаком копірайту.

Функція inputNumbers приймає 4 (x, z, a, b) аргументів та використовує їх для запиту на введення користувачем.

Функція logicalExpression приймає 2 аргументи (a, b) виводить логічне числове (1 / 0) значення виразу a + 3 ≤ b.

Функція decimalHex\_numbersOutput приймає три аргументи (x, y, z) і виводить їх у десятковій та в шістнадцятковій системі числення.

Усі вищеописані функції викликаються у main. Також у потік включається функція s\_calculate, що приймає на вхід значення x, y, z та повертає результат обчислень.

**Обмеження й допущення:**

Оскільки неможливо належно порівняти точні типи даних із очікуваними результатами, введемо похибку до п’ятої цифри після коми.

**Вихідний код:**

#include <iostream>

#include <clocale>

#include <iomanip>

#include <cstring>

#include <conio.h>

#include <ModulesLisevych.h>

using namespace std;

int x = 0, z = 0;

char a = '0', b = '0';

string CopyrightHolder()

{

return "© Лісевич Дмитро Сергійович";

}

void inputNumbers(int X, int Z, int A, int B)

{

cout << "Введіть X: "; cin >> X;

x = X;

cout << "Введіть Z: "; cin >> Z;

z = Z;

cout << "Введіть символ(1): "; cin >> A;

a = A;

cout << "Введіть символ(2): "; cin >> B;

b = B;

cout << endl;

}

bool logicalExpression(int A, int B)

{

return A + 3 <= B;

}

void decimaHexNumberOuput(int X, int Z)

{

cout << "X в десятковій: " << dec << X << endl

<< "Z в десятковій: " << dec << Z << endl

<< endl;

cout << "X в шістнацятковій: " << hex << X << endl

<< "Z в шістнацятковій: " << hex << Z << endl

<< endl;

}

int main()

{

system("chcp 1251");

cout << CopyrightHolder() << endl << endl;

inputNumbers(x, z, a, b);

cout << "Результат: " << a << " + 3 <= " << b << "\n" << logicalExpression(a, b) << " (1 - true; 0 - false)"<< endl << endl;

decimaHexNumberOuput(x, z);

cout << "S = " << s\_calculation(x, z) << endl;

system("pause");

return 0;

}

**ВИСНОВОК**

Виконуючи лабораторну роботу набув 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).

Лабораторна робота виконувалась згідно «Порядок виконання лабораторної роботи та методичні вказівки». Лабораторна робота № 8 стала першою лабораторною роботою з використанням системи контроля версій – GitHub. Не зважаючи на це, особливих проблем під час виконання завдань не виникло, а навіть навпаки було легко та цікаво ним користуватися.

При виконанні завдань 8.1 та 8.2 був задіяний метод висхідного проектування програмного забезпечення, що полягав у початковій реалізації менш значимих (або допоміжних) функцій нижнього рівня. Такою функцією виявилась s\_calculation, з модуля ModulesLisevych, розробленим на етапі задачі 8.1. До даного застосовувались всі правила побудови файлів вихідного коду та заголовкового. Надалі, модуль ModulesLisevych використовувався в main функціях обох задач.

Файл формату .h називається заголовковим і складається з препроцессорних макропідстановок та оголошених прототипів функцій. #ifndef - макрос, що перевіряє визначеність файлу. Якщо файл не визначено - виконуються #define макропідстановка, що задає файлу визначеності. Внаслідок цього, виконується наступний набір операторів, а в кінці вихідного файлу, перевірка на визначеність файлу припиняється макросом #endif. Інакше кажучи - дані макроси дозволяють, або забороняють компіляцію виділеної ними частини коду.

Файл формату .cpp називається файлом вихідного коду та містить ініціалізацію функцій, раніше оголошених в .h файлі. Потребує підключення директиви .h файлу через макропідстановку #include для належної реалізації функцій.

При створенні функції s\_calculation використовувалась стандартна бібліотека C++ <cmath>, яка дозволяє реалізувати математичні функції. Були використані функції sqrt (корінь квадратний), pow (піднесення до степеня), sin (синус кута у радіанах), cos (косинус кута у радіанах).

Для модуля ModulesLisevych був створений тестовий драйвер із назвою TestDriver. Драйвер був створений з метою автоматизувати процес модульного тестування, що зобов'язує ізоляцію певної процедури всередині нього. Це гарантує більш якісне проведення процесу тестування, адже вірогідність виявити залежності та позбутися від них - збільшується.

Завдання 8.2 потребує використання відлагодженого модуля ModulesLisevych. Проект 8.2 користується як процедурною, так і модульною парадигмою програмування (дана потреба викликана умовою задачі). Завдання націлене на детальний огляд роботи потокового виведення за допомогою cout (з простору імен std), функціонування маніпуляторів форматованого виведення та запису виклику функції безпосередньо в потік виведення.

**ДОДАТОК А**

(Test Suite до завдання 8.1)

|  |  |  |
| --- | --- | --- |
| Artifact: Test Suite | Назва тестового набору /  **Test Suite Description** | Test-Suite8.1 |
| Date: 3/14/2022 | Назва проекта/ПЗ /  **Name of project** | Lab8.1-Lisevych.ехе |
|  | Рівень тестування /  **Level of testing** | Системний |
|  | Автор тест-сьюта /  **Test Suite Author** | Дмитро Лісевич |
|  | Виконавець /  **Implementer** | Дмитро Лісевич |

|  |  |  |  |
| --- | --- | --- | --- |
| Ід-р тест кейса /  **Test Case ID** | Дії (кроки)  /  **Action (Test Steps)** | Очікуваний результат  /  **Expected Result** | Результат тестування  /  **Test Result** |
| TS\_01 | 1. Увести 1  2. Увести 2  3. | S = 3.01129 | PASSED |
| TS\_02 | 1. Увести 3  2. Увести 5  3. | S = 23.12387 | PASSED |
| TS\_03 | 1. Увести 5  2. Увести 3  3. | S = -524.44301 | PASSED |
| TS\_04 | 1. Увести 10  2. Увести 10  3. | S = 9937.12941 | PASSED |
| TS\_05 | 1. Увести 1  2. Увести 1  3. | S = 0.111529 | PASSED |

**ДОДАТОК Б**

(Test Suite до завдання 8.2)

|  |  |  |
| --- | --- | --- |
| Artifact: Test Suite | Назва тестового набору /  **Test Suite Description** | Test-Suite8.2 |
| Date: 3/14/2022 | Назва проекта/ПЗ /  **Name of project** | Lab8.2-Lisevych.ехе |
|  | Рівень тестування /  **Level of testing** | Системний |
|  | Автор тест-сьюта /  **Test Suite Author** | Дмитро Лісевич |
|  | Виконавець /  **Implementer** | Дмитро Лісевич |

|  |  |  |  |
| --- | --- | --- | --- |
| Ід-р тест кейса /  **Test Case ID** | Дії (кроки)  /  **Action (Test Steps)** | Очікуваний результат  /  **Expected Result** | Результат тестування  /  **Test Result** |
| TS\_01 | 1. Запустити застосунок | © Лісевич Дмитро Сергійович | PASSED |
| TS\_02 | 1. Запустити застосунок  2. Натиснути Enter  3.Послідовно ввести 1, 1, a, b. | © Лісевич Дмитро Сергійович  Введіть X: 1  Введіть Z: 1  Введіть символ(1): a  Введіть символ(2): b  Результат: a + 3 <= b  1 (1 - true; 0 - false)  X в десятковій: 1  Z в десятковій: 1  X в шістнацятковій: 1  Z в шістнацятковій: 1  S = 0.111529 | PASSED |
| TS\_03 | 1. Запустити застосунок  2. Натиснути Enter  3.Послідовно ввести 1, 2, b, a. | © Лісевич Дмитро Сергійович  Введіть X: 1  Введіть Z: 2  Введіть символ(1): b  Введіть символ(2): a  Результат: b + 3 <= a  1 (1 - true; 0 - false)  X в десятковій: 1  Z в десятковій: 2  X в шістнацятковій: 1  Z в шістнацятковій: 2  S = 3.01129 | PASSEd |
| TS\_04 | 1. Запустити застосунок  2. Натиснути Enter  3.Послідовно ввести 3, 2, f, g. | © Лісевич Дмитро Сергійович  Введіть X: 3  Введіть Z: 2  Введіть символ(1): f  Введіть символ(2): g  Результат: f + 3 <= g  1 (1 - true; 0 - false)  X в десятковій: 1  Z в десятковій: 2  X в шістнацятковій: 1  Z в шістнацятковій: 2  S = 2890.48 | PASSED |
| TS\_05 | 1. Запустити застосунок  2. Натиснути Enter  3.Послідовно ввести 1, 1, g, f.  4. Натиснути Enter | Застосунок закривається. | PASSED |

**ДОДАТОК В**

(Результат тестування s\_calculation)

Test Case #1

X = 1

Z = 2

Expexted Result: 3.01129

Actual Result: 3.01129

Test Case #1: PASSED

Test Case #2

X = 3

Z = 5

Expexted Result: 23.1239

Actual Result: 23.1239

Test Case #2: PASSED

Test Case #3

X = 5

Z = 3

Expexted Result: -524.443

Actual Result: -524.443

Test Case #3: PASSED

Test Case #4

X = 7

Z = 7

Expexted Result: 9937.13

Actual Result: 9937.13

Test Case #4: PASSED

Test Case #5

X = 1

Z = 1

Expexted Result: 0.111529

Actual Result: 0.111529

Test Case #5: PASSED

Press any key to continue . . .

ДОДАТОК Г

**TestDriver:**

#include <iostream>

#include <conio.h>

#include <iomanip>

#include <clocale>

#include "ModulesLisevych.h"

using namespace std;

int main()

{

system("chcp 65001 & cls");

float x[5] = {1, 3, 5, 7, 1};

float z[5] = {2, 5, 3, 7, 1};

double Result[5] = {3.01129, 23.12387, -524.44301, 9937.12941, 0.111529};

double delta = 0.0005;

for(int i = 0; i < 5; i++){

cout << "Test Case #" << i + 1 << "\n"

"X = " << x[i] << "\n"

"Z = " << z[i] << "\n";

cout << "Expexted Result: " << Result[i] << endl;

cout << "Actual Result: " << s\_calculation(x[i], z[i]) - Result[i] << endl;

if(abs(s\_calculation(x[i], z[i]) - Result[i]) <= delta){

cout << "Test Case #" << i + 1 << ":" << " PASSED\n" << endl;

} else {

cout << "Test Case #" << i + 1 << ":" << " FAILED\n" << endl;

}

}

system("pause");

return 0;

}

**ModulesLisevych.h:**

#ifndef MODULESLISEVYCH\_H\_INCLUDED

#define MODULESLISEVYCH\_H\_INCLUDED

#include <cmath>

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

#endif // MODULESLISEVYCH\_H\_INCLUDED

**ModulesLisevych/main.cpp:**

#include <iostream>

#include <cmath>

#include "ModulesLisevych.h"

using namespace std;

double s\_calculation(float x, float z)

{

const double PI = 3.14159;

return (0.5\*pow((abs(2\*z - pow(x, 2))/sin(x)), 3))/(sqrt(1+abs(cos(x)))+2\*PI);

}