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 академічної групи КН-24

Марина Середа

ПЕРЕВІРИЛА

викладачка кафедри кібербезпеки

та програмного забезпечення

Анастасія КОВАЛЕНКО

Кропивницький – 2025

**Мета роботи:** 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).

**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.

**Варіант 7**

Задача 8.1

Вхідні дані:

double – x, y, z

Вихідні дані:

double s

Архітектура

1. Створюємо статичну бібліотеку ModulesSereda і заголовковий файл ModulesSereda.h , в якому міститься прототип s\_calculation, в файлі ModulesSereda.срр реалізація функції s\_calculation.
2. Текстовий проєкт TestDriver містить файл main.cpp, який викликає функцію s\_calculation і тестує різні вхідні данні.
3. Створюємо тестовий набір у TestSuite: файл text.txt, в якому містяться вхідні дані та очікуванні результати.

Алгоритм роботи

1. Користувач вводить значення змінних x, y і z.
2. Викликається функція s\_calculation з статичної бібліотеки ModulesSereda.h і обчислює значення s.
3. Всі данні передаються файлу TestDriver.
4. Результат s виводиться на екран.

|  |  |
| --- | --- |
| Назва тестового набору  Test Suite Description | TS\_8\_1 |
| Назва проекта / ПЗ  Name of Project / Software | TestDriver\_8.1 |
| Рівень тестування  Level of Testing | системний / System Testing |
| Автор тест-сьюта  Test Suite Author | Середа Марина |
| Виконавець  Implementer | Середа Марина |

|  |  |  |  |
| --- | --- | --- | --- |
| Ід-р тест-кейса / Test Case ID | Дії (кроки) /  Action  (Test Steps) | Очікуваний  результат /  Expected Result | Результат тестування (пройшов/не вдалося/ заблокований) /  Test Result (passed/failed/ blocked) |
| 1 | 1. Увести x = 1, y = 1, z = 1. | Введіть числа x, y, z: 1 1 1  S = 6.56698 | passed |
| 2 | 1. Увести x = 3, y = 6, z = 8. | Введіть числа x, y, z: 3 6 8  S = 82.6653 | passed |
| 3 | 1. Увести x = 0, y = 1, z = 1. | Введіть числа x, y, z: 0 1 1  S = 3.42539 | passed |
| 4 | 1. Увести x = 4, y = 5, z = 6. | Введіть числа x, y, z: 4 5 6  S = 41.6827 | passed |
| 5 | 1. Увести x = 5, y = 8, z = 9. | Введіть числа x, y, z: 5 8 9  S = 392.583 | passed |

Задача 8.2

Вхідні дані:

сhar – a, b

int – x, y, z, d, l, v

double – x, y, z

Вихідні дані:

bool – bool\_ab

double s

Алгоритм роботи

1. Користувач вводить a і b.
2. Програма перевіряє рівність виразу a + 5 >= b і виводить результат (true / false).
3. Користувач вводить значення x, y і z.
4. Програма переводить значення x, y і z в десяткове і шістнадцяткове значення.
5. Користувач вводить значення x, y і z.
6. Програма виводить значення s, яке обчислене за формулою з статичної бібліотеки.

|  |  |
| --- | --- |
| Назва тестового набору  Test Suite Description | TS\_8\_2 |
| Назва проекта / ПЗ  Name of Project / Software | Sereda\_task |
| Рівень тестування  Level of Testing | системний / System Testing |
| Автор тест-сьюта  Test Suite Author | Середа Марина |
| Виконавець  Implementer | Середа Марина |

|  |  |  |  |
| --- | --- | --- | --- |
| Ід-р тест-кейса / Test Case ID | Дії (кроки) /  Action  (Test Steps) | Очікуваний  результат /  Expected Result | Результат тестування (пройшов/не вдалося/ заблокований) /  Test Result (passed/failed/ blocked) |
| 1 | 1. Увести 1 1. 2. Увести 1 1 1. 3. Увести 1 1 1. | ---------------------------------------  |Розробник: © Середа М. О.|  |Назва ЗВО: ЦНТУ |  |Рік реалізації проєкту: 2024|  --------------------------------------  Введіть a і b: 1 1  a + 5 >= b -> true  Введіть x, y і z: 1 1 1  x 1  y 1  z 1  x 1  y 1  z 1  Введіть числа x, y, z: 1 1 1  S = 6.56698 | passed |
| 2 | 1. Увести 2 10. 2. Увести 1 1 1. 3. Увести 1 1 1. | ---------------------------------------  |Розробник: © Середа М. О.|  |Назва ЗВО: ЦНТУ |  |Рік реалізації проєкту: 2024|  --------------------------------------  Введіть a і b: 2 10  a + 5 >= b -> false  Введіть x, y і z: 1 1 1  x 1  y 1  z 1  x 1  y 1  z 1  Введіть числа x, y, z: 1 1 1  S = 6.56698 | passed |
| 3 | 1. Увести 1 1. 2. Увести 15 40 27. 3. Увести 1 1 1. | ---------------------------------------  |Розробник: © Середа М. О.|  |Назва ЗВО: ЦНТУ |  |Рік реалізації проєкту: 2024|  --------------------------------------  Введіть a і b: 1 1  a + 5 >= b -> true  Введіть x, y і z: 15 40 27  x 15  y 40  z 27  x f  y 28  z 1b  Введіть числа x, y, z: 1 1 1  S = 6.56698 | passed |
| 4 | 1. Увести 1 1. 2. Увести 1 1 1. 3. Увести 0 5 5. | ---------------------------------------  |Розробник: © Середа М. О.|  |Назва ЗВО: ЦНТУ |  |Рік реалізації проєкту: 2024|  --------------------------------------  Введіть a і b: 1 1  a + 5 >= b -> true  Введіть x, y і z: 1 1 1  x 1  y 1  z 1  x 1  y 1  z 1  Введіть числа x, y, z: 0 5 5  S = 37.5883 | passed |
| 5 | 1. Увести -1 -1. 2. Увести 0 1 0. 3. Увести 9 9 9. | ---------------------------------------  |Розробник: © Середа М. О.|  |Назва ЗВО: ЦНТУ |  |Рік реалізації проєкту: 2024|  --------------------------------------  Введіть a і b: -1 -1  a + 5 >= b -> true  Введіть x, y і z: 0 1 0  x 0  y 1  z 0  x 0  y 1  z 0  Введіть числа x, y, z: 9 9 9  S = 922.576 | passed |

**Висновок:** у ході роботи я набула практичних навичок модульного програмування, реалізовано метод функціональної декомпозиції задачі застосоване модульне тестування. Використовувала скалярні типи даних, арифметичні й логічні операції, організувала введення та виведення інформації. Програмні модулі розробила та протестувала в середовищі Code::Blocks (GNU GCC Compiler), що забезпечило кросплатформність рішень.

**Аргументи**

1. Практичні навички у цій темі зроблять мене кращим програмістом.
2. Підтримка декількох проєктів одночасно розвиває відповідальність.
3. Коли оволодію навичками працювати в Code::Blocks, процес написання та відлагоджування програм стане легшим.
4. Це була одна з найцікавіших лабораторних робіт із-за свої особливостей: нова інформація, дедлайн, здача звіту.
5. Дозволяє розбивати великі програми на логічні частини.
6. Виконання лабораторної роботи в подальшому допомагає в полегшоному розумінню коду.
7. Дозволяє повторно використовувати модулі в інших проєктах.
8. Сприяє глибшому розумінню архітектури програм.
9. Дозволяє застосовувати структурний та об'єктно-орієнтований підходи.
10. Допомагає реалізовувати принцип "розділяй і володарюй".

Дозволяє чітко розмежовувати рівні абстракції.

1. Дозволяє розширювати функціонал без значних змін у коді.
2. Зменшує складність налагодження.
3. Дозволяє знаходити помилки на ранніх етапах розробки.
4. Полегшує автоматизацію перевірки коду.
5. Підвищує якість програмного забезпечення.
6. Допомагає уникати регресій.
7. Зменшує витрати часу на пошук помилок.
8. Сприяє покращенню структури коду.
9. Дозволяє легко перевіряти функціональність окремих модулів.

Покращує відстеження помилок.

1. Дозволяє писати більш безпечний код.
2. Зменшує ризики при зміні програмного коду.
3. Полегшує інтеграцію нових функцій.
4. Робить процес розробки більш передбачуваним.
5. Дозволяє швидко знаходити джерело проблем у коді.
6. Використання типів даних, арифметичних і логічних операцій
7. Дозволяє працювати з різними форматами даних.
8. Сприяє гнучкості в обробці інформації.
9. Допомагає уникати втрати точності у числових обчисленнях.
10. Полегшує роботу з булевими значеннями.
11. Робить код більш зрозумілим.
12. Дозволяє використовувати побітові операції.
13. Підвищує продуктивність у математичних розрахунках.
14. Дозволяє створювати ефективні алгоритми сортування.
15. Полегшує побудову складної логіки програми.
16. Забезпечує правильне управління станами програми.
17. Вивчення алгоритмів допомагає в подальшій роботі з машинним навчанням.
18. Ефективна робота з масивами зменшує витрати оперативної пам’яті.
19. Програмісти з розумінням алгоритмів затребувані на ринку праці.
20. Робота з кодуваннями дозволяє створювати локалізовані програми.
21. **Покращення розуміння синтаксису та концепцій**. Практика допомагає краще запам’ятати синтаксис мов програмування та зрозуміти основні концепції, такі як змінні, цикли, умови, функції тощо.
22. **Розвиток алгоритмічного мислення**. Чим більше ви кодуєте, тим краще навчаєтесь розбивати складні проблеми на менші частини та знаходити ефективні алгоритми для їхнього вирішення.
23. **Збільшення швидкості та ефективності**. Практика дозволяє писати код швидше та більш оптимізовано, уникаючи помилок, які часто роблять новачки.
24. **Робота з реальними проєктам.** Практика дає можливість створювати власні проєкти або брати участь в open-source розробці, що формує портфоліо та підвищує конкурентоспроможність на ринку праці.
25. **Знайомство з інструментами та технологіями**. Під час роботи з кодом можна освоювати редактори, системи контролю версій (Git), середовища розробки (IDE) та інші важливі інструменти.
26. **Навчання на помилках**. В процесі написання коду програмісти стикаються з помилками, які допомагають краще розуміти, як працює система та як уникати подібних проблем у майбутньому.
27. **Підготовка до співбесід**. Компанії часто перевіряють практичні навички кандидата під час співбесіди, тому регулярна практика допомагає бути впевненим у своїх силах.
28. **Розвиток креативності**. Пошук рішень для різних завдань сприяє розвитку нестандартного мислення та вмінню знаходити ефективні способи вирішення проблем.
29. **Закріплення теоретичних знань**. Практика допомагає перетворити абстрактні концепції на реальні навички, що робить навчання ефективнішим.
30. **Формування алгоритмічного мислення**. Чим більше ви кодуєте, тим краще навчаєтеся аналізувати проблеми та розбивати їх на зрозумілі кроки.
31. **Розвиток навичок налагодження**. Написання коду включає роботу з помилками та їх виправлення, що формує навички дебагінгу.

Додатки

* Статична бібліотека ModulesSereda

#include <iostream>

#include <cmath>

#include <cstdio>

using namespace std;

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

double s;

if(z != 0 && y != 0){

s = sqrt(abs(pow(z, 2) \* (1.0/2) \* y)) + ((M\_PI \* x + pow(M\_E, abs(y))) / y);

}

return s;

}

* Заголовковий файл ModulesSereda

#ifndef MODULESSEREDA\_H\_INCLUDED

#define MODULESSEREDA\_H\_INCLUDED

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

#endif // MODULESSEREDA\_H\_INCLUDED

* TestDriver

#include <iomanip>

#include <locale>

#include <windows.h>

#include <iostream>

#include "C:\Users\Asus\Desktop\BMTP-LAB8-Sereda\prj\ModulesSereda\ModulesSereda.h"

using namespace std;

int main()

{

SetConsoleCP(CP\_UTF8);

SetConsoleOutputCP(CP\_UTF8);

double x, y, z, s;

cout << "Введіть числа x, y, z: ";

cin >> x >> y >> z;

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

cout << "S = "<< s << endl;

return 0;

}

* Sereda\_task

#include <iostream>

#include <clocale>

#include <cstdio>

#include <iomanip>

#include <cmath>

#include <windows.h>

#include "C:\Users\Asus\Desktop\BMTP-LAB8-Sereda\prj\ModulesSereda\ModulesSereda.h"

using namespace std;

void name(){

cout << " ----------------------------" << endl;

cout << "|Розробник: \u00A9 Середа М. О. |" << endl;

cout << "|Назва ЗВО: ЦНТУ |" << endl;

cout << "|Рік реалізації проєкту: 2024|" << endl;

cout << " ----------------------------" << endl;

}

int a\_b(char a, char b){

bool bool\_ab = a + 5 >= b;

cout << "a + 5 >= b -> " << boolalpha << bool\_ab << endl;

}

void oct\_hex(){

int x, y, z;

cout << "Введіть x, y і z: ";

cin >> x >> y >> z;

cout << "x " << dec << x << endl;

cout << "y " << dec << y << endl;

cout << "z " << dec << z << endl;

cout << "x " << hex << x << endl;

cout << "y " << hex << y << endl;

cout << "z " << hex << z << endl;

}

int main()

{

SetConsoleCP(CP\_UTF8);

SetConsoleOutputCP(CP\_UTF8);

name();

int d, l, v;

cout << "Введіть a і b: ";

cin >> l >> v;

d = a\_b(l, v);

oct\_hex();

double x, y, z, s;

cout << "Введіть числа x, y, z: ";

cin >> x >> y >> z;

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

cout << "S = "<< s << endl;

return 0;

}