Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 2

з навчальної дисципліни “Модульне 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-19

Москальов А.В.

ПЕРЕВІРИВ

старший викладач кафедри кібербезпеки та програмного забезпечення

Поліщук Л. І.

Кропивницький – 2020

Тема: Реалізація статичних 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/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:

1. Реалізувати статичну бібліотеку модулів libModulesMoskalev C/C++, яка містить функції розв’язування задач 2.1 та 2.2.

2. Реалізувати програмне забезпечення розв’язування задачі 2.3 — консольний додаток

Варіант №12:

**Задача 2.1:**

- За значеннями x,y,z обчислюється Q відповідно до виразу із задачі 1.2 (ЛР №1)

**Задача 2.2:**

- За значеннями x,y,z обчислюється S:

**Задача 2.3:**

- За послідовними запитами вводяться числа x,y,z та символи a і b.

- Вивести (включити у потік STL – cout):

2.3.1: Прізвище та ім’я розробника програми зі знаком охорони авторського права «©»;

2.3.2: Результат логічного виразу в числовому вигляді (1/0):

2.3.3: Значення x,y,z в десятковій і вісімковій системах числення;

Аналіз задачі 2.1:

Вхідними даними можуть бути лише числа. Цілі та дійсні. Число має бути в діапазоні від 1 до 142. Тип даних використовується дійсний. Для результату та маніпуляцій використовуються також дійсні числа. Вихідним буде 1 число дійсного типу, яке можна округлити за допомогою Round. В такому випадку число буде цілим.

Аналіз задачі 2.2:

Вхідними даними можуть бути лише числа. Цілі та дійсні. Число має бути в діапазоні від 1 до 100. Тип даних використовується дійсний. Для результату та маніпуляцій використовуються також дійсні числа. Вихідним буде 1 число дійсного типу, яке можна округлити за допомогою Round. В такому випадку число буде цілим.

**Тестові артефакти до задач 2.1 та 2.2:**

 

**Результати тестування з тестового драйверу до задачі 2.1:**

Test #0 is completed.

Input - 1, 1, 1. Output - 0

Test #1 is completed.

Input - 2, 3, 5. Output - 0

Test #2 is completed.

Input - 15, 1, 4. Output - -32762

Test #3 is completed.

Input - 3, 2, 17. Output - -64

Test #4 is completed.

Input - 9, 1, 5. Output - -1393

Test #5 is completed.

Input - 6, 2, 3. Output - -17

Test #6 is completed.

Input - 5, 6, 3. Output - 206

Test #7 is completed.

Input - 7, 2, 44. Output - -565035

Test #8 is completed.

Input - 1, 99, 1. Output - 2

Test #9 is completed.

Input - 5, 5, 1. Output – 97

**Результати тестування з тестового драйверу до задачі 2.2:**

Test #0 is completed.

Input - 1, 1, 1. Output - 4

Test #1 is completed.

Input - 2, 3, 5. Output - 730

Test #2 is completed.

Input - 15, 1, 4. Output - 4

Test #3 is completed.

Input - 3, 2, 17. Output - 96

Test #4 is completed.

Input - 9, 1, 5. Output - 4

Test #5 is completed.

Input - 6, 2, 3. Output - 96

Test #6 is completed.

Input - 5, 6, 3. Output - 23327

Test #7 is completed.

Input - 7, 2, 44. Output - 96

Test #8 is completed.

Input - 44, 1, 1. Output - 3

Test #9 is completed.

Input - 5, 5, 1. Output - 9374

**Код TestDriver.cpp:**

#include <iostream>

#include <cstdlib>

#include "ModulesMoskalev.h"

using namespace std;

int main()

{

// Task1

int Ans1[10] = {0,0,-32762,-64,-1393,-17,206,-565035,2,97};

int Qs1[10][3] = { {1,1,1}, {2,3,5}, {15,1,4}, {3,2,17}, {9,1,5}, {6,2,3}, {5,6,3}, {7,2,44}, {1,99,1}, {5,5,1} };

short i = 0;

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

{

if ((q\_calculation(Qs1[i][0],Qs1[i][1],Qs1[i][2]) == Ans1[i]))

{

cout << "Test #" << i << " is completed."<<endl;

cout << "Input - "<<Qs1[i][0]<<", "<<Qs1[i][1]<<", "<<Qs1[i][2]<<". Output - "<<q\_calculation(Qs1[i][0],Qs1[i][1],Qs1[i][2])<<endl;

}

if ((q\_calculation(Qs1[i][0],Qs1[i][1],Qs1[i][2]) != Ans1[i]))

{

cout << "Test #" << i << " is NOT completed."<<endl;

cout << "Input - "<<Qs1[i][0]<<", "<<Qs1[i][1]<<", "<<Qs1[i][2]<<". Output - "<<q\_calculation(Qs1[i][0],Qs1[i][1],Qs1[i][2])<<endl;

exit;

}

}

system("pause");

// Task2

int Ans2[10] = {4,730,4,96,4,96,23327,96,3,9374};

int Qs2[10][3] = { {1,1,1}, {2,3,5}, {15,1,4}, {3,2,17}, {9,1,5}, {6,2,3}, {5,6,3}, {7,2,44}, {44,1,1}, {5,5,1} };

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

{

if ((s\_calculation(Qs2[i][0],Qs2[i][1],Qs2[i][2]) == Ans2[i]))

{

cout << "Test #" << i << " is completed."<<endl;

cout << "Input - "<<Qs2[i][0]<<", "<<Qs2[i][1]<<", "<<Qs2[i][2]<<". Output - "<<s\_calculation(Qs2[i][0],Qs2[i][1],Qs2[i][2])<<endl;

}

if ((s\_calculation(Qs2[i][0],Qs2[i][1],Qs2[i][2]) != Ans2[i]))

{

cout << "Test #" << i << " is NOT completed."<<endl;

cout << "Input - "<<Qs2[i][0]<<", "<<Qs2[i][1]<<", "<<Qs2[i][2]<<". Output - "<<s\_calculation(Qs2[i][0],Qs2[i][1],Qs2[i][2])<<endl;

exit;

}

}

system("pause");

return(0);

}

**Код від ModulesMoskalev.cpp:**

#include "ModulesMoskalev.h"

#include <cstdlib>

#include <math.h>

using namespace std;

double q\_calculation(double x, double y, double z)

{

double a,b,c = 0;

a = x+y;

a = cos(a); // Первая часть в переменной А.

b = fabs(x);

c = pow(z,b);

b = sqrt(c); // Вторая часть в переменной Б.

c = pow(x,y);

c = 3.14\*c;

c = sqrt(c); // Третья часть в переменной С.

return round(a-b+c);

}

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

{

double a,b,c = 0;

a=sin(x);

b=y\*z;

c=x+y;

b=b/c;

b=fabs(b);

b=sqrt(b);

a=a/b; // Первая часть в переменной А.

b=pow(y,5);

c=3\*b; // Вторая часть в переменной С.

return round(a+c);

}

Аналіз задачі 2.3:

Задачу 2.3.1 простіше всього реалізувати за допомогою рядка (string). Знак копірайту запишемо як char(169).

Задача 2.3.2 використовую змінні a,b, задля створення валідатору їх також створюємо змінною string, після чого виконуємо перевірку на введений символ та кількість введених символів. Допустимі обмеження – 1 символ англійської клавіатури будь-якого регістру. Вивід булевого типу.

Задача 2.3.3 використовує обидві функції задач 2.1 та 2.2, тож беремо найменше обмеження, для задачі 2.2 це 1-100. Це обмеження вводимо для числових змінних. Числові змінні також зчитуємо рядком, щоб цілковито перевірити їх за допомогою валідатору.

Для користувача створюємо простий інтерфейс – меню вибору задачі, що слід вивести у консольний додаток та пропозиції повторити процес виконання.

Тестовий артефакт до задачі 2.3:



Висновок:

В процесі виконання лабораторної роботи ми на практиці знайомилися з мовою програмування С та С++, а також з кросплатформовим середовищем Code::Blocks (GNU GCC Compiler). Перейти на іншу мову програмування після паскаля було досить просто, адже синтаксис не дуже різниця, а місцями набагато зручніший. В ході лабораторної роботи ми знову повернулися до консольного додатку. Мною в процесі виконання роботи використовувалась мова С++, адже вона здалась мені зручнішою. Для виводу в консольний додаток рядку ми використовували оператор cout, а для вводу інформації – cin. Обидва ці додатки є частиною бібліотеки iostream, яку слід підключити на початку коду: #include <iostream>. Щодо самих завдань – самою складною частиною було переведення числа з десяткової системи числення до вісімкової. Також деякі труднощі викликало створення бібліотеки, проте досить швидко питання закінчилися. Розрахунок за формулою нічим не різниться від того, що ми виконували у паскалі, лише деякі оператори змінили назву (наприклад power у мові С – pow). Додавання символу копірайта у рядок мною здійснювалося методом char(169). Логічний вираз реалізується досить просто: створюється функція, що повертає булеве значення. Це значення включається у потік та виводиться у консольний додаток. Підключення модуля (задачі 2.1 та 2.2) не викликало ніяких труднощів, інтуїтивно все було зрозуміло. Також слід зауважити, що власні функції додаються також за допомогою оператору #include, але в лапках “” та з додаванням розширення (.h). Після цього ми просто використовуюємо описані функції, вони автоматично підтягуються у код.