МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №2**

**з дисципліни «Основи програмної інженерії»**

**на тему: *«*Модульне програмування. Метод висхідного програмування.»**

Виконав: студент гр. ПЗ2011

Бублик Микита Сергійович

Прийняла: Горбова О. В.

Дніпро, 2021

Тема. Модульне програмування. Метод висхідного програмування.

Мета роботи. Отримати практичні навички розробки програм за методом висхідного програмування.

# Завдання

Розробити програму-калькулятор для обчислення трьох математичних виразів за індивідуальним завданням. Розробку програми провести за висхідним методом через створення тестового драйверу для заданої в індивідуальному завданні пари функцій введення даних і обчислення виразу. Для обчислення функцій використати типи даних, вказані у індивідуальному завданні.

Вимоги до програми:

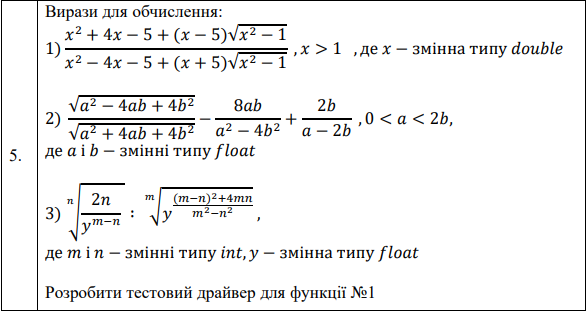
1) вибір виразу для обчислення здійснити за допомогою діалогу з користувачем;

2) введення даних і обчислення виразів реалізувати окремими функціями (всього 6 функцій);

3) для виклику функцій використати масиви вказівників на функції введення даних і обчислення виразів (функція обчислення виразу і відповідна їй функція введення даних повинні мати однаковий індекс в масивах, де вони зберігаються);

4) данні для обчислення виразів ввести з клавіатури, за потреби здійснити їх перевірку;

Варіант 5



**1. Стуктурна схема**

****

**2. Текст програми**

func.h

#pragma once

struct Exam

{

int m, n;

float y;

};

void\* input1();

void\* input2();

void\* input3();

double example1(void\* ptr);

double example2(void\* ptr);

double example3(void\* ptr);

Source.cpp

#include <iostream>

#include "func.h"

using namespace std;

//Тестовий драйвер для 3-го виразу

void\* inputTest()

{

static int x = 5;

return &x;

}

double testDriver(void\* ptr)

{

int\* x = static\_cast<int\*>(ptr);

return (\*x) \* (\*x);

}

int main()

{

setlocale(LC\_ALL, "ru");

//==============тестовий драйвер=====================

void\* par = inputTest();

cout << "x^2 = " << testDriver(par) << endl;

//===================================================

double(\*ptrExample[])(void\*) { example1, example2, example3 };

void\* (\*ptrInput[])() { input1, input2, input3 };

int index;

cout << "Введите необходимую цифру(1, 2, 3) для подсчёта выражения: "; cin >> index;

while (index > 3)

{

cout << "Выражения с таким номером не существует, введите номер еще раз." << endl;

cin >> index;

}

cout<< "Результат вычисления = "<<ptrExample[index-1](ptrInput[index-1]());

return 0;

}

func.cpp

#include "func.h"

#include<iostream>

#include<cmath>

#define a abc[0]

#define b abc[1]

#define c abc[2]

using namespace std;

/// <summary>

/// Ввод параметров 1-го выражения

/// </summary>

/// <returns>параметр для 1-го выражения</returns>

void\* input1()

{

static int y;

cout << "Данное выражение принимает один целочисленный параметр (y). y !=0 " << endl;

cout << "Введите значение y = "; cin >> y;

while (y==0)

{

cout << "'y' не может равнятся нулю, введите параметр еще раз" << endl;

cout << "Введите значение y = "; cin >> y;

}

return &y;

}

/// <summary>

/// Ввод параметров 2-го выражения

/// </summary>

/// <returns>параметр для 2-го выражения</returns>

void\* input2()

{

static Exam exam;

cout << "Данное выражение принимает два параметра: " << endl;

cout << "1) Вещественный x;" << endl;

cout << "2) Целый y; " << endl;

cout << "Введите значение x = "; cin >> exam.x;

cout << "Введите значение y = "; cin >> exam.y;

return &exam;

}

/// <summary>

/// Ввод параметров 2-го выражения

/// </summary>

/// <returns>параметр для 3-го выражения</returns>

void\* input3()

{

static double arr[3];

cout << "Данное выражение принимает 3 вещественных параметра (a, b, c) " << endl;

cout << "Введите значение a = "; cin >> arr[0];

cout << "Введите значение b = "; cin >> arr[1];

cout << "Введите значение c = "; cin >> arr[2];

while (arr[1] == 0)

{

cout << "Ошибка, на ноль делить нельзя. Введите параметр еще раз!" << endl;

cout << "Введите значение b = "; cin >> arr[1];

}

while (arr[2] == 0)

{

cout << "Ошибка, на ноль делить нельзя. Введите параметр еще раз!";

cout << "Введите значение c = "; cin >> arr[2];

}

return arr;

}

/// <summary>

/// Вычисление 1-го примера

/// </summary>

/// <param name="ptr"></param>

/// <returns>результат подсчёта</returns>

double example1(void\* ptr)

{

int\* y = static\_cast<int\*>(ptr);

double step1 = pow(2., 2. / 3.) + (27. \* pow(\*y, 3. / 5.));

double step2 = sqrt(2.) + 3. \* pow(\*y, 1. / 5.);

double step3 = step1 / step2;

double step4 = 3 \* pow(32. \* pow(\*y, 2.) - 2., 1. / 10.);

double step5 = step3 + step4;

double step6 = step5 \* pow(3, -2);

double step7 = pow(step6, 5);

return step7;

}

/// <summary>

/// Вычисление 2-го примера

/// </summary>

/// <param name="ptr"></param>

/// <returns>результат подсчёта</returns>

double example2(void\* ptr)

{

Exam\* ex = static\_cast<Exam\*>(ptr);

double step1 = (pow(ex->x, 2.) - pow(ex->y, 2)) \* (cbrt(ex->x) + cbrt(ex->y));

double step2 = cbrt(pow(ex->x, 5.)) + cbrt(pow(ex->x, 2.) \* pow(ex->y, 3));

double step3 = cbrt(pow(ex->x, 3.) \* pow(ex->y, 2))-cbrt(pow(ex->y,5));

double step4 = step2 - step3;

double step5 = step1 / step4;

double step6 = step5 - (cbrt(ex->x \* ex->y) + cbrt(pow(ex->y, 2)));

return step6;

}

/// <summary>

/// Вычисление 3-го примера

/// </summary>

/// <param name="ptr"></param>

/// <returns>результат подсчёта</returns>

double example3(void\* ptr)

{

double\* abc = static\_cast<double\*>(ptr);

double step1 = 1 /( b \* (a \* b \* c + a + c));

double step2 = 1 / (a + (1 / (b + (1 / c))));

double step3 = 1 / (a + (1 / b));

double step4 = step2 / step3;

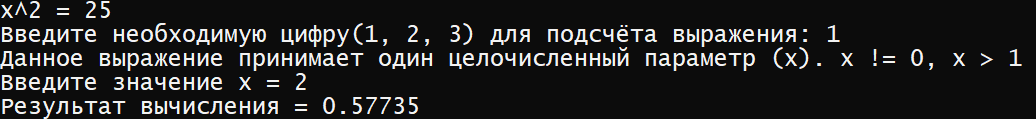
double step5 = step1 - step4;

return step5;

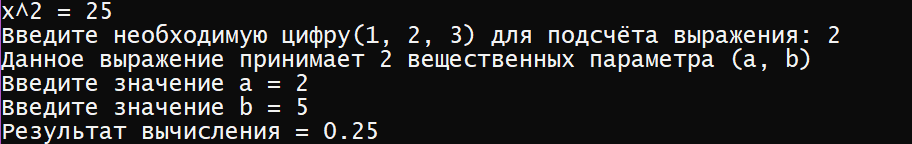
}

**3. Результати виконання**

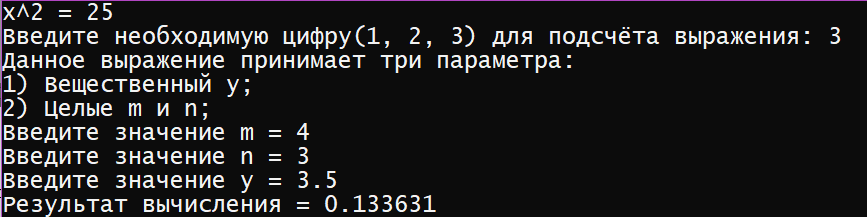
1-й вираз



2-й вираз



3-й вираз



**4. Висновок**

Під час виконання лабораторної роботи користувався методом висхідного програмування. Суть даного методу полягає у тому, що першими створюються модулі найнижчого рівня і поступопово розробка переходить до модулів більш високого рівня. Для тестування модулів створюється тестовий драйвер, щоб імітувати модуль вищого рівня.