Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 4 з дисципліни
«Основи програмування 2. Модульне програмування»

«Перевантаження операторів»

Варіант №32

Виконав студент ІП-14 Шляхтун Денис Михайлович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вітковська Ірина Іванівна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота №4

Тема: перевантаження операторів.

Мета: вивчити механізми створення класів з використанням перевантажених операторів.

Хід роботи

Задача.

32. Визначити клас "Трикутник", членами якого є координати вершин трикутника в просторі. Реалізувати для нього декілька конструкторів, геттери, методи обчислення периметра трикутника. Перевантажити оператори: префіксний "++" / постфіксний "++" - для інкрементування усіх x-координат і усіх y-координат вершин трикутника відповідно, "+=" – для збільшення усіх координат вершин трикутника на вказану величину. Створити три трикутника (T1, T2, T3), використовуючи різні конструктори. Інкрементувати x-координати вершин трикутника T1 і y-координати вершин трикутника T2. Збільшити координати вершин трикутника T3 на вказану величину. Серед трикутників T1, T2, T3 визначити трикутник, що має найбільший периметр.

Виконання задачі.

Випробування коду на C++.

Код:

main.cpp

```cpp
#include "module.h"
using namespace std;

int main()
{
    //створення першого трикутника
    cout << "The first triangle: ";
    TPoint p[3];
    pointArrInput(p, 3);
    TTriangle T1(p[0], p[1], p[2]);
    T1.output();

    //створення другого трикутника
    cout << "\nThe second triangle: ";
    int len[3];
    lenArrInput(len, 3);
    TTriangle T2(len[0], len[1], len[2]);
    T2.output();

    //створення третього трикутника
    cout << "\nThe third triangle: ";
    TTriangle T3(T2);
    T3.output();

    //інкрементування x-координат
    ++T1;   T1.output();

    //інкрементування y-координат
    T2++;   T2.output();

    //збільшення координат на задану величину
    T3 += addInput();   T3.output();

    //визначення максимального периметру
    maxPerimeter(T1, T2, T3);

    system("pause");
}
```

TTriangle.h

```h
#pragma once
#include <iostream>
#include <iomanip>
#include "TPoint.h"

class TTriangle
{
    TPoint p1, p2, p3;  //вершини трикутника
    const int num;      //порядковий номер трикутника
    static int count;   //загальна кількість створених об'єктів класу
public:
    TTriangle(TPoint a, TPoint b, TPoint c);//конструктор зі списком ініціалізації
    TTriangle(int a, int b, int c);         //конструктор з параметрами
    TTriangle(TTriangle&);                  //конструктор копіювання
    TPoint getCertainVertex(int);   //геттер певної вершини трикутника
    void getAllVertices(TPoint&, TPoint&, TPoint&); //геттер для всіх вершин трикутника
    int getNum() { return num; };   //геттер дря порядкового номера
    static int getCount() { return TTriangle::count; }; //геттер для кількості створених об'єктів
    float perimeter();  //обчислення периметру трикутника
    TTriangle operator++();                 //перевантаження префіксного "++"
    TTriangle operator++(int notused);      //перевантаження постфіксного "++"
    const TTriangle operator+=(const float add);    //перевантаження "+="
    void output();  //виведення даних про трикутник
    ~TTriangle();   //деструктор
};
```

TTriangle.cpp

```cpp
#include "TTriangle.h"

//статична змінна кількості створених об'єктів класу
int TTriangle::count;

//конструктор зі списком ініціалізації
TTriangle::TTriangle(TPoint a, TPoint b, TPoint c)
    : p1(a), p2(b), p3(c), num(++count) {}

//конструктор з параметрами за замовчуванням шляхом відкладення відрізків на осях координат
TTriangle::TTriangle(int a = 1, int b = 1, int c = 1) : num(++count)
{
    p1.set(a, 0, 0);
    p2.set(0, b, 0);
    p3.set(0, 0, c);
}

//конструктор копіювання
TTriangle::TTriangle(TTriangle& T) : num(++count)
{
    T.getAllVertices(p1, p2, p3);
}
```

```cpp
//геттер певної вершини трикутника
TPoint TTriangle::getCertainVertex(int n = 1)
{
    switch (n)
    {
    case 1:
        return p1;
    case 2:
        return p2;
    case 3:
        return p3;
    default:
        cout << "\nWrong input detected. Value of first verticle is returned.";
        return p1;
    }
}

//геттер для всіх вершин трикутника
void TTriangle::getAllVertices(TPoint& a, TPoint& b, TPoint& c)
{
    a = p1; b = p2; c = p3;
}

//обчислення периметру трикутника
float TTriangle::perimeter()
{
    return sideLength(p1, p2) + sideLength(p2, p3) + sideLength(p1, p3);
}
```

```cpp
//перевантаження префіксного "++" для збільшення x-координат
TTriangle TTriangle::operator++()
{
    p1.x++; p2.x++; p3.x++;
    return *this;
}

//перевантаження постфіксного "++" для збільшення y-координат
TTriangle TTriangle::operator++(int notused)
{
    p1.y++; p2.y++; p3.y++;
    return *this;
}

//перевантаження "+=" для збільшення всіх координат на задану величину
const TTriangle TTriangle::operator+=(const float add)
{
    p1 += add; p2 += add; p3 += add;
    return *this;
}

//виведення даних про трикутник
void TTriangle::output()
{
    cout << "\nTriangle #" << num <<
        "\nP1:"; p1.output();
    cout << "\nP2:"; p2.output();
    cout << "\nP3:"; p3.output();
    cout << "\nGeneral number of triangles: " << count << endl;
}
```

```cpp
//деструктор
TTriangle::~TTriangle()
{
    count--;
}
```

TPoint.h

```h
#pragma once
#include <iostream>
#include <iomanip>
using namespace std;

class TPoint
{
    float x, y, z; //координати точки
public:
    void set(int a, int b, int c); //задання значення координат точки
    const TPoint operator+=(const float add); //перевантаження оператора "+="
    void input();   //введення координат точки
    void output();  //виведення координат точки
    friend float sideLength(TPoint a, TPoint b); //визначення довжини між двома точками
    friend class TTriangle; //дружній клас трикутник
};
```

TPoint.cpp

```cpp
#include "TPoint.h"

//задання значення координат точки
void TPoint::set(int a, int b, int c)
{
    x = a; y = b; z = c;
}

/*перевантаження оператора "+=" для
збільшення всіх координат на задану величину*/
const TPoint TPoint::operator+=(const float add)
{
    x += add; y += add; z += add; return *this;
}

//введення координат точки
void TPoint::input()
{
    cout << "\nEnter x: ";
    cin >> x;
    cout << "Enter y: ";
    cin >> y;
    cout << "Enter z: ";
    cin >> z;
}

//виведення координат точки
void TPoint::output()
{
    cout << setw(4) << x
        << setw(4) << y
        << setw(4) << z;
}
```

module.h

```cpp
#pragma once
#include <iostream>
#include <iomanip>
#include "TTriangle.h"

float sideLength(TPoint a, TPoint b); //визначення відстані між двома точками
void pointArrInput(TPoint* arr, int n); //введення масиву точок
void lenArrInput(int* arr, int n); //введення масиву відстаней
float addInput(); //введення значення для додавання до координат трикутника
int maxPerimeter(TTriangle&, TTriangle&, TTriangle&); //найбільший периметр
```

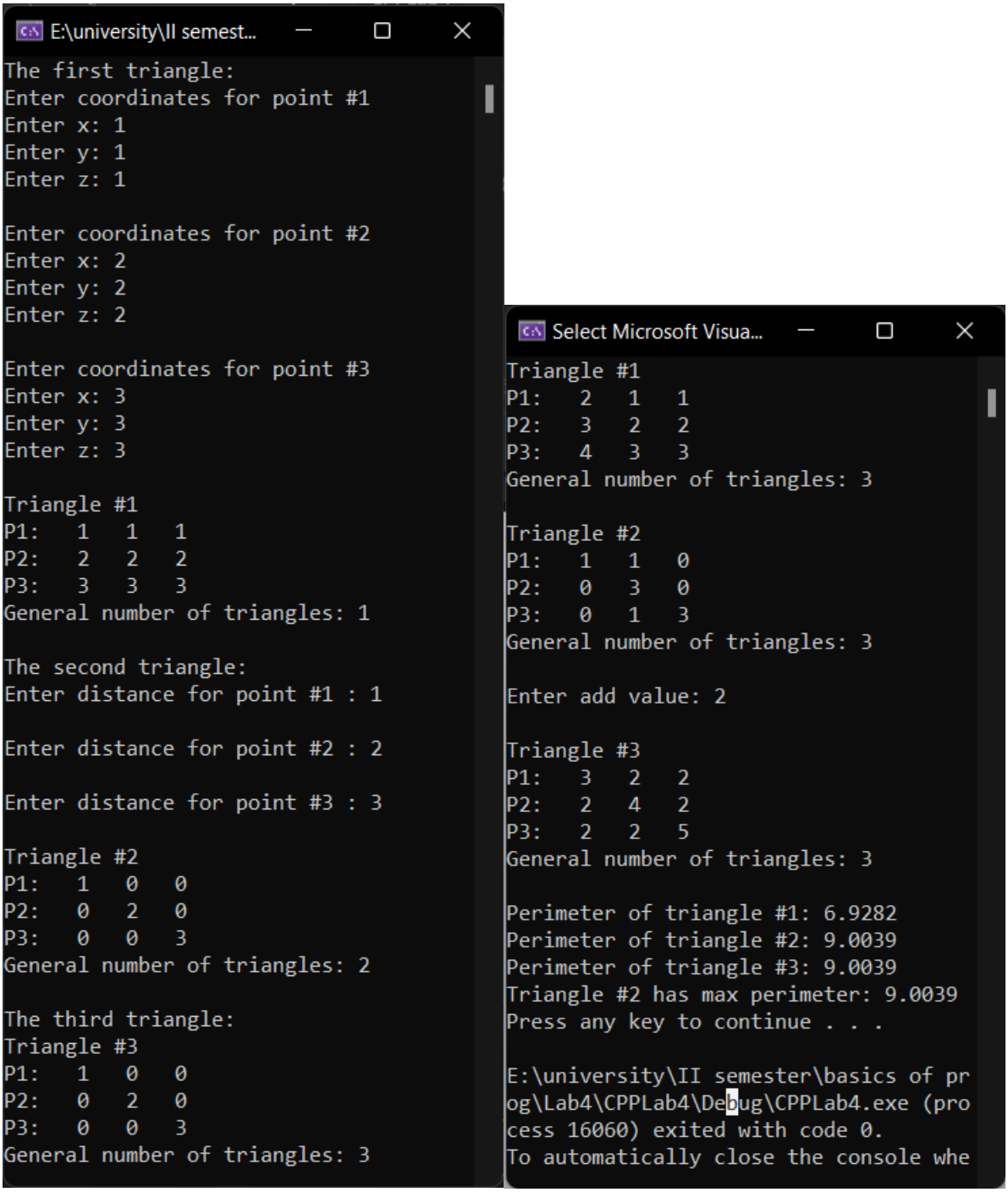
module.cpp

```cpp
#include "module.h"

//визначення відстані між двома точками
float sideLength(TPoint a, TPoint b)
{
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y) + (a.z - b.z) * (a.z - b.z));
}

//введення масиву точок
void pointArrInput(TPoint* arr, int n)
{
    for (int i = 0; i < n; i++)
    {
        cout << "\nEnter coordinates for point #" << i + 1;
        arr[i].input();
    }
}

//введення масиву відстаней
void lenArrInput(int* arr, int n)
{
    for (int i = 0; i < n; i++)
    {
        cout << "\nEnter distance for point #" << i + 1 << " : ";
        cin >> arr[i];
    }
}
```

```cpp
//введення значення для додавання до координат трикутника
float addInput()
{
    float add;
    cout << "\nEnter add value: ";
    cin >> add;
    return add;
}
```

```
38      //знаходження найбільшого периметру
39      int maxPerimeter(TTriangle& T1, TTriangle& T2, TTriangle& T3)
40      {
41          float per,
42              per1 = T1.perimeter(),
43              per2 = T2.perimeter(),
44              per3 = T3.perimeter();
45          int pos;
46          cout << "\nPerimeter of triangle #" << T1.getNum() << ": " << per1;
47          cout << "\nPerimeter of triangle #" << T2.getNum() << ": " << per2;
48          cout << "\nPerimeter of triangle #" << T3.getNum() << ": " << per3;
49
50          if (per1 >= per2)
51          {
52              per = per1;
53              pos = T1.getNum();
54          }
55          else
56          {
57              per = per2;
58              pos = T2.getNum();
59          }
60
61          if(per3>per)
62          {
63              per = per3;
64              pos = T3.getNum();
65          }
66
67          cout << "\nTriangle #" << pos << " has max perimeter: " << per << endl;
68
69          return pos;
70      }
```

Результат:



Висновок: При виконанні лабораторної роботи було вивчено механізми створення класів з використанням перевантажених операторів у мові програмування C++ та на основі набутих навичок було виконано задачу. Під час виконання лабораторної роботи було використано різні типи конструкторів (з параметрами, списком ініціалізацій та копіювання) та було здійснено перевантаження операторів: префіксного «++», суфіксного «++» та «+=».