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 № 5 з дисципліни

«Основи програмування – 2. Об’єктно-орієнтоване програмування»

«Успадкування та поліморфізм»

Варіант 21

Виконав студент ІП-11, Ляля Іван Олександрович

Перевірила Вітковська Ірина Іванівна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

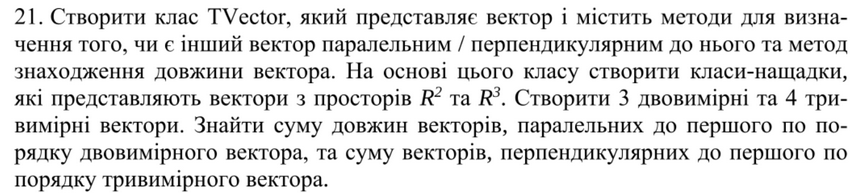
Київ 2022

**Лабораторна робота 5**

**Успадкування та поліморфізм**

**Мета** – вивчити механізми створення і використання класів і об’єктів.

**Задача:**



**Постановка задачі:**

В головній функції ми створюємо масив двовимірних і масив тривимірних векторів, ініціалізуємо значеннями, виводимо ці масиви на екран та виконуємо поставлене нам в останньому реченні завдання, розбивши його на дві частини – перша – функція Task1, друга – Task2.

Описуємо базовий клас TVector, в якому атрибутами є динамічний масив координат вектора та цілочисельна змінна – вимірність вектора n; є методи ініціалізації класу, конструктор копіювання та конструктор за замовчуванням, метод виведення show та віртуальні методи встановлення паралельності, перпендикулярності двох векторів та знаходження довжини вектора. В цьому класі всі методи описані для векторів довільної розмірності.

Клас R2\_Vector описує двовимірний масив, має атрибути-координати x,y. Методами класу є конструктор з параметрами, метод введення об’єкта, визначення довжини двовимірного вектора, встановлення паралельності та перпендикулярності двох двовимірних векторів. Останні три методи описані з метою зменшення витрати ресурсів на виконання, оскільки відповідні методи в базовому класі є значно більш часо- та ресурсомісткими.

Клас R3\_Vector має атрибути x,y,z. Методи є аналогічними як і в R2\_Vector.

**Текст і результат роботи програми на С++:**

**cpp lab 5.cpp:**

#include "Header.h"

int main()

{

const int n1 = 3, n2 = 4;

R2\_Vector Arr1[n1]; //масив двовимірних векторів

R3\_Vector Arr2[n2]; //масив тривимірних векторів

for (int i = 0; i < n1; i++) //заповнення масивів

Arr1[i].input("Arr1[" + to\_string(i + 1) + ']'); //

for (int i = 0; i < n2; i++) //

Arr2[i].input("Arr2[" + to\_string(i + 1) + ']'); //

cout << "Array 1:" << endl;

for (int i = 0; i < n1; i++)

Arr1[i].show();

cout << "Array 2:" << endl;

for (int i = 0; i < n2; i++)

Arr2[i].show();

Task1(Arr1, n1); //знаходження суми довжин векторів, паралельних до першого у масиві

Task2(Arr2, n2); //знаходження суми векторів, перпендикулярних до першого у масиві

return 0;

}

**Header.h:**

#pragma once

#include <iostream>

#include <string>

#include "TVector.h"

#include "R2\_Vector.h"

#include "R3\_Vector.h"

using namespace std;

void Task1(R2\_Vector arr[], const int size); //знаходження суми довжин векторів, паралельних до першого у масиві двовимірних векторів

void Task2(R3\_Vector arr[], const int size); //знаходження суми векторів, перпендикулярних до першого у масиві тривимірних векторів

**Source.cpp:**

#include "Header.h"

void Task1(R2\_Vector arr[], const int size)

{

float sum = 0;

for (int i = 1; i < size; i++)

if (arr[0].Parallel(arr[i])) sum += arr[i].Length();

cout << "Sum of lengths of vectors', parallel to the first two-dimensional vector = ";

cout << sum << endl;

}

void Task2(R3\_Vector arr[], const int size)

{

R3\_Vector sum;

for (int i = 1; i < size; i++)

if (arr[0].Perpendicular(arr[i])) sum += arr[i];

cout << "Sum of vectors, perpendicular to the first three-dimensional vector = ";

sum.show();

}

**TVector.h:**

#pragma once

#include <iostream>

using namespace std;

class TVector

{

protected:

int n; //вимірність простору

float\* coordinates; //покажчик на динамічний масив координат вектор

protected:

void initialize(int k, float\* arr); //ініціалізація об'єкта TVector

public:

TVector() : n(0), coordinates(nullptr) {}; //конструктор для об'єкта TVector за замовчуванням

TVector(TVector& mem); //конструктор копіювання

void show(); //метод виведення вектора будь-якої розмірності на екран

virtual bool Parallel(TVector vec); //метод встановлення паралельності двох векторів будь-якої розмірності

virtual bool Perpendicular(TVector vec); //метод встановлення перпендикудярності двох векторів будь-якої рзмірності

virtual float Length(); //метод встановлення довжини вектора будь-якї розмірності

~TVector() { delete[] coordinates; } //деструктор класу

};

**TVector.cpp:**

#include "TVector.h"

void TVector::initialize(int k, float\* arr)

{

n = k;

coordinates = new float[n];

for (int i = 0; i < n; i++)

coordinates[i] = arr[i];

}

TVector::TVector(TVector& mem)

{

n = mem.n;

coordinates = new float[n];

for (int i = 0; i < n; i++)

coordinates[i] = mem.coordinates[i];

}

void TVector::show()

{

cout << "(";

for (int i = 0; i < n - 1; i++)

cout << coordinates[i] << ':';

cout << coordinates[n - 1] << ')' << endl;

}

bool TVector::Parallel(TVector vec)

{

if ((\*this).n != vec.n) { cout << "Vectors of different size...\n"; return 0; }

else

{

bool flag = true;

for (int i = 0; i < n; i++)

if (coordinates[0] / vec.coordinates[0] != coordinates[i] / vec.coordinates[i]) flag = 0;

return flag;

}

}

bool TVector::Perpendicular(TVector vec)

{

if ((\*this).n != vec.n) { cout << "Vectors of different size...\n"; return 0; }

else

{

float scalar = 0;

for (int i = 0; i < n; i++)

scalar += coordinates[i] \* vec.coordinates[i];

if (scalar == 0) return 1;

else return 0;

}

}

float TVector::Length()

{

float len\_square = 0;

for (int i = 0; i < n; i++)

{

len\_square += pow(coordinates[i], 2);

}

return sqrt(len\_square);

}

**R2\_Vector.h:**

#pragma once

#include <iostream>

#include <string>

#include "TVector.h"

using namespace std;

class R2\_Vector :public TVector

{

float x, y;

public:

R2\_Vector(float a = 0, float b = 0) : x(a), y(b) //конструктор з параметрами для двовимірного вектора

{

float arr[2] = { a,b };

initialize(2, arr);

}

void input(string s); //метод введення двовимірного вектора

float Length() { return sqrt(pow(x, 2) + pow(y, 2)); } //визначення довжини двовимірного вектора

bool Parallel(R2\_Vector vec) //встановлення паралельності двох двовимірних векторів

{

if (x / vec.x == y / vec.y) return 1;

else return 0;

}

bool Perpendicular(R2\_Vector vec) //встановлення перпендикулярності двох двовимірних векторів

{

if (x \* vec.x + y \* vec.y == 0) return 1;

else return 0;

}

};

**R2\_Vector.cpp:**

#include "R2\_Vector.h"

void R2\_Vector::input(string s)

{

cout << "Input coordinates for vector " << s << " : " << endl;

cout << "x = "; float x1; cin >> x1;

cout << "y = "; float y1; cin >> y1;

x = x1; y = y1;

float arr[2] = { x1,y1 };

initialize(2, arr);

}

**R3\_Vector.h:**

#pragma once

#include <iostream>

#include <string>

#include "TVector.h"

using namespace std;

class R3\_Vector :public TVector

{

float x, y, z;

public:

R3\_Vector(float a = 0, float b = 0, float c = 0) : x(a), y(b), z(c) //конструктор з параметрами для тривимірного вектора

{

float arr[3] = { a,b,c };

initialize(3, arr);

}

void input(string s); //метод введення тривимірного вектора

float Length() { return sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2)); } //метод визначення довжини тривимірного вектора

bool Parallel(R3\_Vector vec) //встановлення паралельності двох тривимірних векторів

{

if (x / vec.x == y / vec.y == z / vec.z) return 1;

else return 0;

}

bool Perpendicular(R3\_Vector vec) //встановлення перпендикулярності двох двовимірних векторів

{

if (x \* vec.x + y \* vec.y + z \* vec.z == 0) return 1;

else return 0;

}

const R3\_Vector operator+=(const R3\_Vector vec); //перевизначення оператора скороченого додавання для двох тривимірних векторів

};

**R3\_Vector.cpp:**

#include "R3\_Vector.h"

void R3\_Vector::input(string s)

{

cout << "Input coordinates for vector " << s << " : " << endl;

cout << "x = "; float x1; cin >> x1;

cout << "y = "; float y1; cin >> y1;

cout << "z = "; float z1; cin >> z1;

x = x1; y = y1; z = z1;

float arr[3] = { x1,y1,z1 };

initialize(3, arr);

}

const R3\_Vector R3\_Vector::operator+=(const R3\_Vector vec)

{

x += vec.x;

y += vec.y;

z += vec.z;

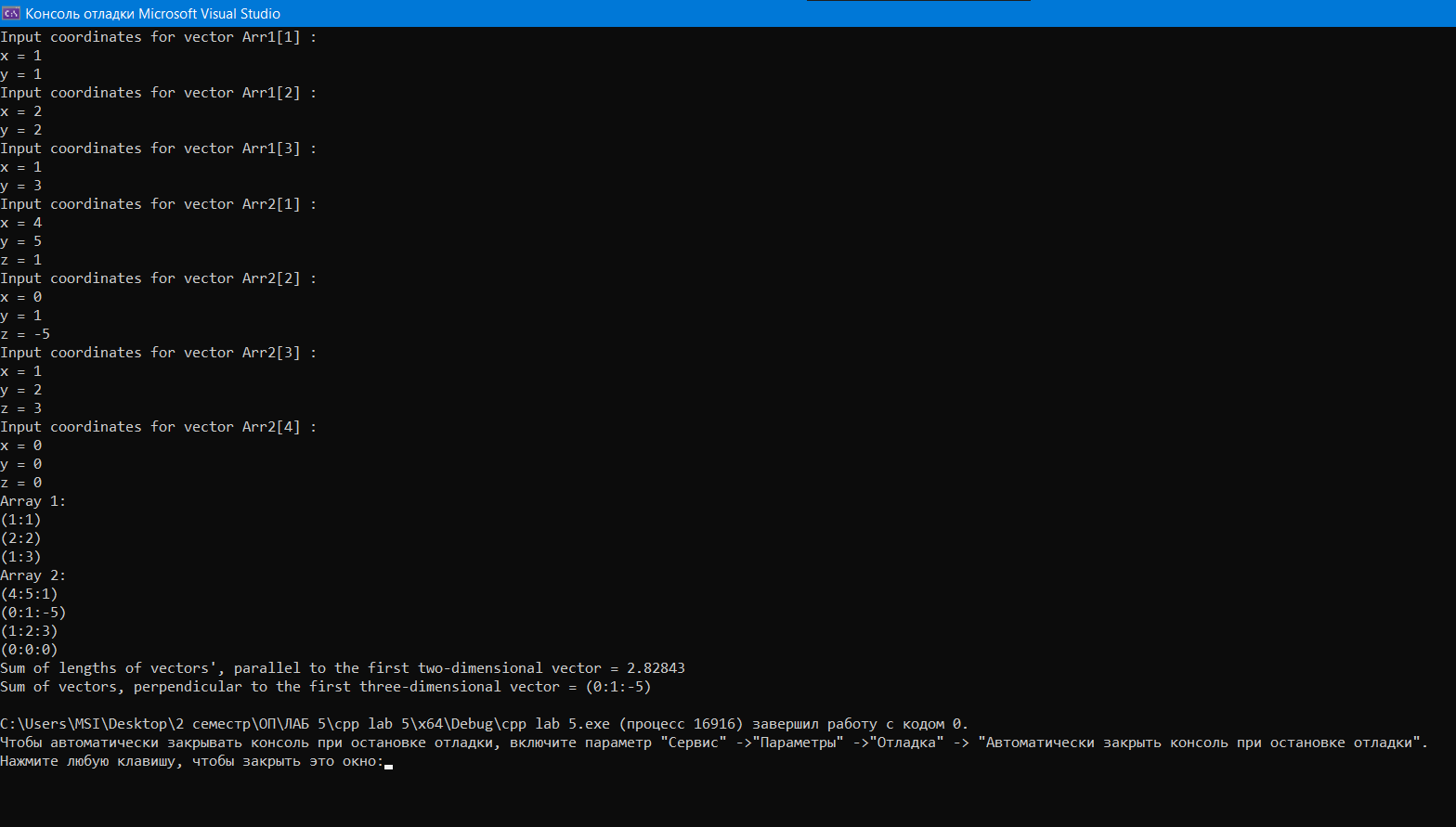
coordinates[0] += vec.x;

coordinates[1] += vec.y;

coordinates[2] += vec.z;

return \*this;

}



**Текст і результат роботи програми на Python:**

**main.py:**

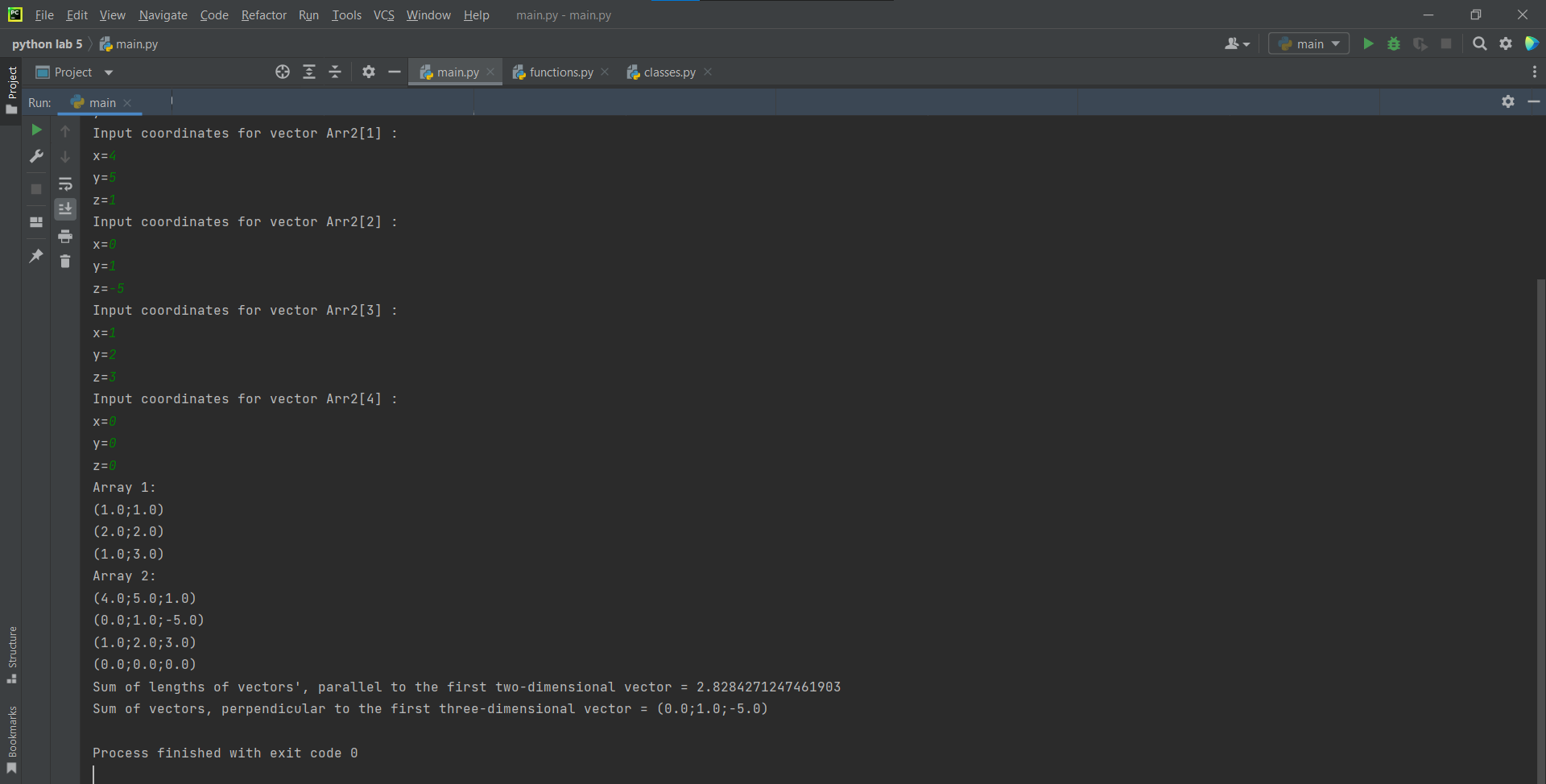
from functions import\*  
  
n1=3  
n2=4  
Arr1=[]  
Arr2=[]  
for i in range(n1):  
 vector=R2\_Vector()  
 vector.input(str("Arr1["+str(i+1)+"]"))  
 Arr1+=[vector]  
for i in range(n2):  
 vector=R3\_Vector()  
 vector.input(str("Arr2["+str(i+1)+']'))  
 Arr2+=[vector]  
print("Array 1:")  
for i in range(len(Arr1)):  
 Arr1[i].show()  
print("Array 2:")  
for i in range(len(Arr2)):  
 Arr2[i].show()  
Task1(Arr1)  
Task2(Arr2)

**functions.py:**

from classes import\*  
  
def Task1(arr):  
 sum=0  
 for i in range(1,len(arr)):  
 if arr[0].Parallel(arr[i]):  
 sum+=arr[i].Length()  
 print("Sum of lengths of vectors', parallel to the first two-dimensional vector = "+str(sum))  
  
def Task2(arr):  
 sum\_vec=R3\_Vector(0,0,0)  
 for i in range(1,len(arr)):  
 if arr[0].Perpendicular(arr[i]):  
 sum\_vec+=arr[i]  
 print("Sum of vectors, perpendicular to the first three-dimensional vector = ",end='')  
 sum\_vec.show()

**classes.py:**

class TVector:  
 def \_\_init\_\_(self,k,arr):  
 self.n=int(k)  
 self.coordinates=arr  
 def show(self):  
 print('('+str(self.coordinates[0]),end='')  
 for i in range(1,self.n):  
 print(';'+str(self.coordinates[i]),end='')  
 print(')')  
 def Parallel(self,vec):  
 if self.n!=vec.n:  
 print("Vectors of different size...\n")  
 return False  
 else:  
 flag=True  
 for i in range(1,len(self.coordinates)):  
 if self.coordinates[0]/vec.coordinates[0]!=self.coordinates[i]/vec.coordinates:  
 flag=False  
 return flag  
 def Perpendicular(self,vec):  
 if self.n!=vec.n:  
 print("Vectors of different size...\n")  
 return False  
 else:  
 scalar=0  
 for i in range(len(self.coordinates)):  
 scalar+=self.coordinates[i]\*vec.coordinates[i]  
 if scalar==0:  
 return True  
 else:  
 return False  
 def Length(self):  
 len\_square=0  
 for i in len(self.coordinates):  
 len\_square+=pow(self.coordinates[i],2)  
 return pow(len\_square,0.5)  
  
class R2\_Vector(TVector):  
 def \_\_init\_\_(self,a=0,b=0):  
 self.x=a  
 self.y=b  
 arr=[a,b]  
 super().\_\_init\_\_(2,arr)  
 def input(self,s):  
 print("Input coordinates for vector "+s+" :")  
 self.x=float(input("х="))  
 self.y=float(input("y="))  
 self.coordinates=[self.x,self.y]  
 def Length(self):  
 return pow(pow(self.x,2)+pow(self.y,2),0.5)  
 def Parallel(self,vec):  
 if vec.x==0:  
 X\_Relation=float('inf')  
 else:  
 X\_Relation=self.x/vec.x  
 if vec.y==0:  
 Y\_Relation=float('inf')  
 else:  
 Y\_Relation=self.y/vec.y  
 if X\_Relation==Y\_Relation:  
 return True  
 else:  
 return False  
 def Perpendicular(self, vec):  
 if self.x\*vec.x+self.y\*vec.y==0:  
 return True  
 else:  
 return False  
  
class R3\_Vector(TVector):  
 def \_\_init\_\_(self,a=0,b=0,c=0):  
 self.x=a  
 self.y=b  
 self.z=c  
 arr=[a,b,c]  
 super().\_\_init\_\_(3,arr)  
 def input(self,s):  
 print("Input coordinates for vector "+str(s)+" :")  
 self.x=float(input("х="))  
 self.y=float(input("y="))  
 self.z=float(input("z="))  
 self.coordinates=[self.x,self.y,self.z]  
 def Length(self):  
 return pow(pow(self.x,2)+pow(self.y,2)+pow(self.z,2),0.5)  
 def Parallel(self,vec):  
 if vec.x==0:  
 X\_Relation=float('inf')  
 else:  
 X\_Relation=self.x/vec.x  
 if vec.y==0:  
 Y\_Relation=float('inf')  
 else:  
 Y\_Relation=self.y/vec.y  
 if vec.z==0:  
 Z\_Relation=float('inf')  
 else:  
 Z\_Relation=self.z/vec.z  
 if X\_Relation==Y\_Relation==Z\_Relation:  
 return True  
 else:  
 return False  
 def Perpendicular(self, vec):  
 if self.x\*vec.x+self.y\*vec.y+self.z\*vec.z==0:  
 return True  
 else:  
 return False  
 def \_\_iadd\_\_(self, vec):  
 self.x+=vec.x  
 self.y+=vec.y  
 self.z+=vec.z  
 self.coordinates[0]+=vec.x  
 self.coordinates[1]+=vec.y  
 self.coordinates[2]+=vec.z  
 return self

****

**При однакових вхідних даних результати на обох мовах правильні та збігаються\***

**Висновок:**

Під час виконання лабораторної роботи я ознайомився з механізмами успадкування класів та поліморфізмом на С++ та Python. Я закріпив навички роботи з класами, їх конструкторами, деструкторами, перевантаженням операторів, об’єктами.