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. Модульне програмування»
«Успадкування та поліморфізм»
Варіант 7

Виконав студент ІП-15, Гуменюк Олександр Володимирович
(шифр, прізвище, ім'я, по батькові)
Перевірила Вєчерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота 5

Успадкування та поліморфізм

1.1.1 Вимоги до програми

1) Реалізація коду відповідно до модульного підходу.
2) Виведення усіх вхідних, проміжних і вихідних даних.
3) У кожному варіанті завдання при описі класів самостійно визначити необхідні поля та методи вводу/виводу. Деякі методи класу-предка можуть бути віртуальними і абстрактними. У програмі-клієнті для збереження сукупності об'єктів використати масив.

Індивідуальне завдання

Варіант 7

7. Створити клас TVector, який представляє вектор і місить методи для обчислення довжини вектора та скалярного добутку векторів. На основі цього класу створити класи-нащадки, які представляють вектори з просторів R^2 та R^3. За допомогою цих класів обчислити значення виразу

$$S = \langle a,b \rangle + \langle c,d \rangle + |a|,$$

де $a,b \in R^3$, а $c,d \in R^2$.

Код C++

OP_Lab5.cpp

```
#include "functions.h"

int main()
{
    TVector *vectors[4];
    vectors[0] = inputTVector3D("Enter information for 3D Vector A");
    vectors[1] = inputTVector3D("\nEnter information for 3D Vector B");
    vectors[2] = inputTVector2D("\nEnter information for 2D Vector C");
    vectors[3] = inputTVector2D("\nEnter information for 2D Vector D");

    vectors[0]->print("\nVector A -> ");
    vectors[1]->print("Vector B -> ");
    vectors[2]->print("Vector C -> ");
```

```
    vectors[3]->print("Vector D -> ");

    double S = vectors[0]->getScalar(*vectors[1]) + vectors[2]-
>getScalar(*vectors[3]) + vectors[0]->getLength();
    cout << "\nResult of calculations S: " << S << endl;
}
```

functions.h

```
#pragma once
#include <string>
#include <exception>
#include <iostream>
#include "tvector.h"
#include "tvector2d.h"
#include "tvector3d.h"

using namespace std;

double inputDouble(string header);
TVector2D* inputTVector2D(string header);
TVector3D* inputTVector3D(string header);
```

functions.cpp

```
#include "functions.h"


TVector2D* inputTVector2D(string header) {
    cout << header << endl;

    double x, y;

    x = inputDouble("Input x coordinate: ");
    y = inputDouble("Input y coordinate: ");

    TVector2D* p = new TVector2D(x, y);
    return p;
}

TVector3D* inputTVector3D(string header) {
    cout << header << endl;

    double x, y, z;

    x = inputDouble("Input x coordinate: ");
    y = inputDouble("Input y coordinate: ");
    z = inputDouble("Input z coordinate: ");

    TVector3D* p = new TVector3D(x, y, z);
    return p;
}

double inputDouble(string header)
{
    string num;
    double parsedNumber = 0.0;
    bool isValid = false;
```

```
    while (!isValid) {
        try
        {
            cout << header;
            cin >> num;
            parsedNumber = std::stod(num);
            isValid = true;
        }
        catch (const exception e)
        {
            cout << "You can only enter a number!" << endl;
            isValid = false;
        }
    }

    return parsedNumber;
}
```

tvector.h

```
#pragma once
#include <string>

using namespace std;

class TVector {
public:
	virtual double getScalar(TVector &) = 0;
	virtual double getLength() = 0;
	virtual double getCoordinate(char) = 0;
	virtual void print(string) = 0;
};
```

tvector.cpp

```
#include "tvector.h"
```

tvector2d.h

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


using namespace std;

class TVector2D : public TVector {
	double x, y;
public:
	TVector2D(double x, double y) : x(x), y(y) {};

	double getScalar(TVector &);
	double getLength();
	double getCoordinate(char);
```

```
	void print(string);
};
```

tvector2d.cpp

```
#include "tvector2d.h"

double TVector2D::getCoordinate(char c)
{
	switch (c)
	{
	case 'x':
		return x;
		break;
	case 'y':
		return y;
		break;
	}
}

double TVector2D::getScalar(TVector& vector) {
	return x * vector.getCoordinate('x') + y * vector.getCoordinate('y');
}

double TVector2D::getLength() {
	return sqrt(x * x + y * y);
}

void TVector2D::print(string s) {
	cout << s << "TVector2D: " << " x = " << setw(4) << x << " y = " << setw(4)
<< y << endl;
}
```

tvector3d.h

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

using namespace std;

class TVector3D : public TVector {
	double x, y, z;
public:
	TVector3D(double x, double y, double z) : x(x), y(y), z(z) {};

	double getScalar(TVector&);
	double getLength();
	double getCoordinate(char);
	void print(string);
};
```

tvector3d.cpp

```
#include "tvector3d.h"
```

```
double TVector3D::getCoordinate(char c)
{
	switch (c)
	{
	case 'x':
		return x;
		break;
	case 'y':
		return y;
		break;
	case 'z':
		return z;
		break;
	}
}

double TVector3D::getScalar(TVector& vector) {
	return x * vector.getCoordinate('x') + y * vector.getCoordinate('y') + z *
vector.getCoordinate('z');
}

double TVector3D::getLength() {
	return sqrt(x * x + y * y + z * z);
}

void TVector3D::print(string s) {
	cout << s<< "TVector3D: " << " x = " << setw(4) << x << " y = " << setw(4) <<
y << " z = " << setw(4) << z << endl;
}
```

Код Python

main.py

```
from functions import inputTVector2D, inputTVector3D

if __name__ == "__main__":
    lst = [inputTVector3D("Enter information for 3D Vector A"),
           inputTVector3D("\nEnter information for 3D Vector B"),
           inputTVector2D("\nEnter information for 2D Vector C"),
           inputTVector2D("\nEnter information for 2D Vector D")]

    lst[0].print("\nVector A ->")
    lst[1].print("Vector B ->")
    lst[2].print("Vector C ->")
    lst[3].print("Vector D ->")

    S = lst[0].getScalar(lst[1]) + lst[2].getScalar(lst[3]) +
lst[0].getLength()
    print(f"\nResult of calculations S: {S:.2f}")
```

functions.py

```
from tvector2d import TVector2D
from tvector3d import TVector3D


def inputFloat(header):
    isValid = False
    while not isValid:
        try:
            num = float(input(header))
        except ValueError:
            print("You can only enter a number!")
        else:
            isValid = True
    return num


def inputTVector2D(header):
    print(header)
    x = inputFloat("Input x coordinate: ")
    y = inputFloat("Input y coordinate: ")

    return TVector2D(x, y)


def inputTVector3D(header):
    print(header)
    x = inputFloat("Input x coordinate: ")
    y = inputFloat("Input y coordinate: ")
    z = inputFloat("Input z coordinate: ")

    return TVector3D(x, y, z)
```

tvector.py

```
from abc import ABC, abstractmethod


class TVector(ABC):
    @abstractmethod
    def getScalar(self, other):
        pass

    @abstractmethod
    def getLength(self):
        pass

    @abstractmethod
    def getCoordinate(self, c):
        pass

    @abstractmethod
```

```
    def print(self, header):
        pass
```

tvector2d.py

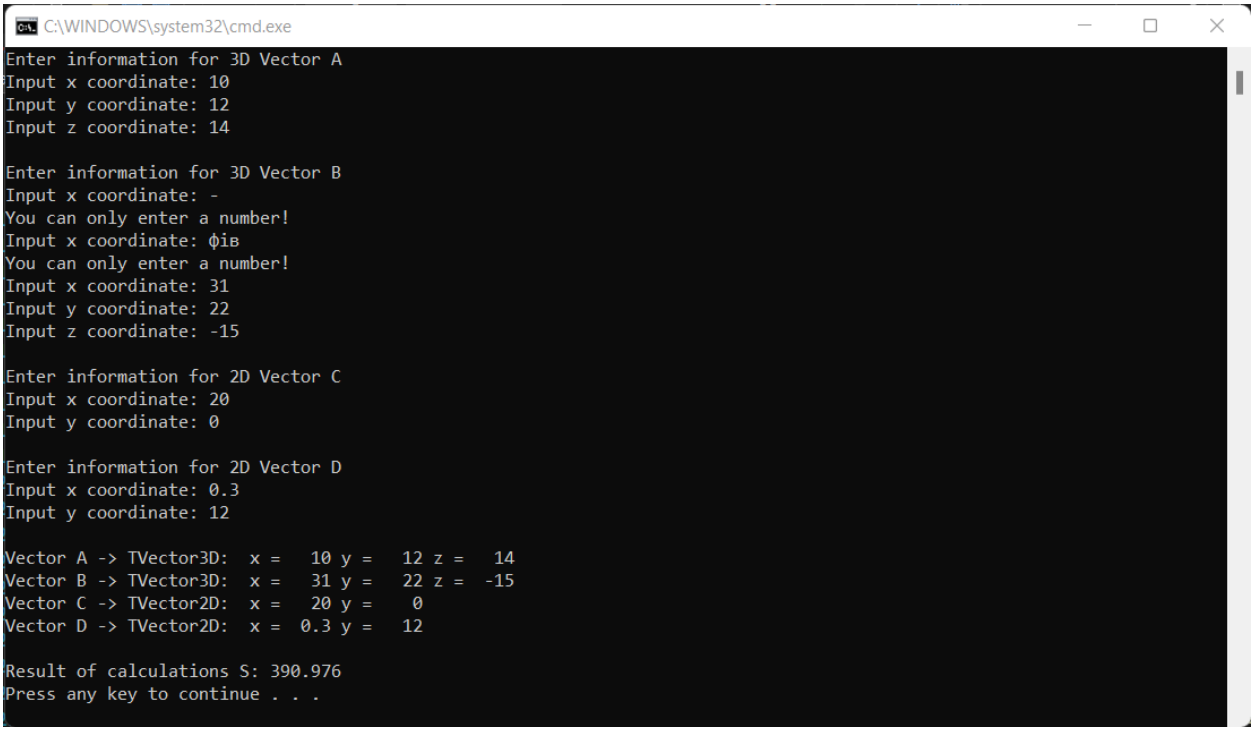
```
from tvector import TVector
from math import sqrt


class TVector2D(TVector):
    def __init__(self, x, y):
        self.__x = x
        self.__y = y

    def getScalar(self, other):
        return self.__x * other.getCoordinate("x") + self.__y *
other.getCoordinate("y")

    def getLength(self):
        return sqrt(self.__x * self.__x + self.__y * self.__y)

    def getCoordinate(self, c):
        if c == "x":
            return self.__x
        elif c == "y":
            return self.__y
        else:
            return 0

    def print(self, header):
        print(f"{header} TVector2D: x = {self.__x:5} y = {self.__y:5}")
```

tvector3d.py

```
from tvector import TVector
from math import sqrt


class TVector3D(TVector):
    def __init__(self, x, y, z):
        self.__x = x
        self.__y = y
        self.__z = z

    def getScalar(self, other):
        return self.__x * other.getCoordinate("x") + self.__y *
other.getCoordinate("y") + self.__z * other.getCoordinate("z")

    def getLength(self):
        return sqrt(self.__x * self.__x + self.__y * self.__y + self.__z *
self.__z)

    def getCoordinate(self, c):
        if c == "x":
            return self.__x
        elif c == "y":
            return self.__y
```

```
        elif c == "z":
            return self.__z
        else:
            return 0

    def print(self, header):
        print(f"{header} TVector3D: x = {self.__x:5} y = {self.__y:5} z =
{self.__z:5}")
```

Тестування C++



Тестування Python

```
"C:\Users\Productive Sasha\AppData\Local\Programs\Python\Python39\python.exe" "C:/Users/Productive Sasha/PycharmProjects/OP_Lab5/main.py"
Enter information for 3D Vector A
Input x coordinate: 10
Input y coordinate: 12
Input z coordinate: 14

Enter information for 3D Vector B
Input x coordinate: -
You can only enter a number!
Input x coordinate: фів
You can only enter a number!
Input x coordinate: 31
Input y coordinate: 22
Input z coordinate: -15

Enter information for 2D Vector C
Input x coordinate: 20
Input y coordinate: 0

Enter information for 2D Vector D
```