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. Методології програмування»

«Успадкування та поліморфізм»

Варіант 21

Виконав студент ІП-13 Макарчук Лідія Олександрівна
(шифр, прізвище, ім'я, по батькові)

Перевірив Вєчерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота №5

Успадкування та поліморфізм

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

Варіант 21

Завдання:

21. Створити клас TVector, який представляє вектор і містить методи для визначення того, чи є інший вектор паралельним / перпендикулярним до нього та метод знаходження довжини вектора. На основі цього класу створити класи-нащадки, які представляють вектори з просторів R^2 та R^3. Створити 3 двовимірні та 4 тривимірні вектори. Знайти суму довжин векторів, паралельних до першого по порядку двовимірного вектора, та суму векторів, перпендикулярних до першого по порядку тривимірного вектора.

1. Виконання завдання мовою С++

Код:

//laba5cpp.cpp(main)

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <vector>
#include "TVector.h"
#include "Work_with_vectors.h"

using namespace std;

int main()
{
   srand(time(NULL));
   int n = 3; // R2
   int m = 4; // R3

   vector <TVector*> arr, arrPar, arrPerp;
   create_arr_of_vectors(arr, n, m);
   cout << "R2: \n";
   display_vector(arr, 0, n);
   cout << "R3: \n";
   display_vector(arr, n);

   double parallelSum = calc_sum_of_parallels(arr, n, m, arrPar);
   cout << "\nVectors which are parallel to the first R2 vector:\n";
   display_vector(arrPar);
   cout << "\nSum of R2 vectors, which are parallel = " << parallelSum << "\n";

   double perpendicular = calc_sum_of_perpendicular(arr, n, m, arrPerp);
   cout << "\nVectors which are perpendicular to the first R3 vector:\n";
   display_vector(arrPerp);
   cout << "\nSum of R3 vectors, which are perpendicular = " << perpendicular << "\n";
}
```

//TVector.h

```
#pragma once

class TVector
{
protected:
	double x;
	static int generate_number(int rangeMin, int rangeMax);
public:
	//constructors
	TVector(double x);
	TVector(int rangeMin = -3, int rangeMax = 3);
	//methods
	virtual void display();

	virtual bool is_parallel(TVector* ptr) = 0;
	virtual bool is_perpendicular(TVector* ptr) = 0;
	virtual double calc_length() = 0;
	friend class R2;
	friend class R3;
};

class R2 : public TVector
{
	double y;
public:
	R2(double x, double y);
	R2(int rangeMin = -5, int rangeMax = 5);

	void display() override;
	bool is_parallel(TVector* ptr) override;
	bool is_perpendicular(TVector* ptr) override;
	double calc_length() override;
};

class R3 : public TVector
{
	double y;
	double z;
public:
	R3(double x, double y, double z);
	R3(int rangeMin = -5, int rangeMax = 5);

	void display() override;
	bool is_parallel(TVector* ptr) override;
	bool is_perpendicular(TVector* ptr) override;
	double calc_length() override;
};
```

//TVector.cpp

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include <stdlib.h>
#include "TVector.h"

using namespace std;

TVector::TVector(double x)
{
	this->x = x;
}
TVector::TVector(int rangeMin, int rangeMax)
{
	x = generate_number(rangeMin, rangeMax);
}
int TVector::generate_number(int rangeMin, int rangeMax)
```

```
{
	int number = rand() % (rangeMax - rangeMin + 1) + rangeMin;
	return number;
}
void TVector::display()
{
	cout << setw(4) << x << " ; ";
}


R2::R2(double x, double y) : TVector(x)
{
	this->y = y;
}
R2::R2(int rangeMin, int rangeMax) : TVector(rangeMin, rangeMax)
{
	y = generate_number(rangeMin, rangeMax);
}
void R2::display()
{
	TVector::display();
	cout << setw(4) << y << "; ";
}
bool R2::is_parallel(TVector* ptr)
{
	double crossProduct = x * ((R2*)ptr)->y - y * ptr->x;
	return !crossProduct;
}
bool R2::is_perpendicular(TVector* ptr)
{
	double dotProduct = x * ptr->x + y * ((R2*)ptr)->y;
	return !dotProduct;
}
double R2::calc_length()
{
	return sqrt(pow(x, 2) + pow(y, 2));
}


R3::R3(double x, double y, double z) : TVector(x)
{
	this->y = y;
	this->z = z;
}
R3::R3(int rangeMin, int rangeMax) : TVector(rangeMin, rangeMax)
{
	y = generate_number(rangeMin, rangeMax);
	z = generate_number(rangeMin, rangeMax);
}

void R3::display()
{
	TVector::display();
	cout << setw(4) << y << "; " << setw(4) << z << "; ";
}
bool R3::is_parallel(TVector* ptr)
{
	double crossProduct = (y * ((R3*)ptr)->z - ((R3*)ptr)->y * z) - (x * ((R3*)ptr)->z - ptr->x * z)
+ (x * ((R3*)ptr)->y - ptr->x * y);
	return !crossProduct;
}
bool R3::is_perpendicular(TVector* ptr)
{
	double dotProduct = x * ptr->x + y * ((R3*)ptr)->y + z * ((R3*)ptr)->z;
	return !dotProduct;
}
double R3::calc_length()
{
```

```
	return sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2));
}
```

//Work_with_vectors.h

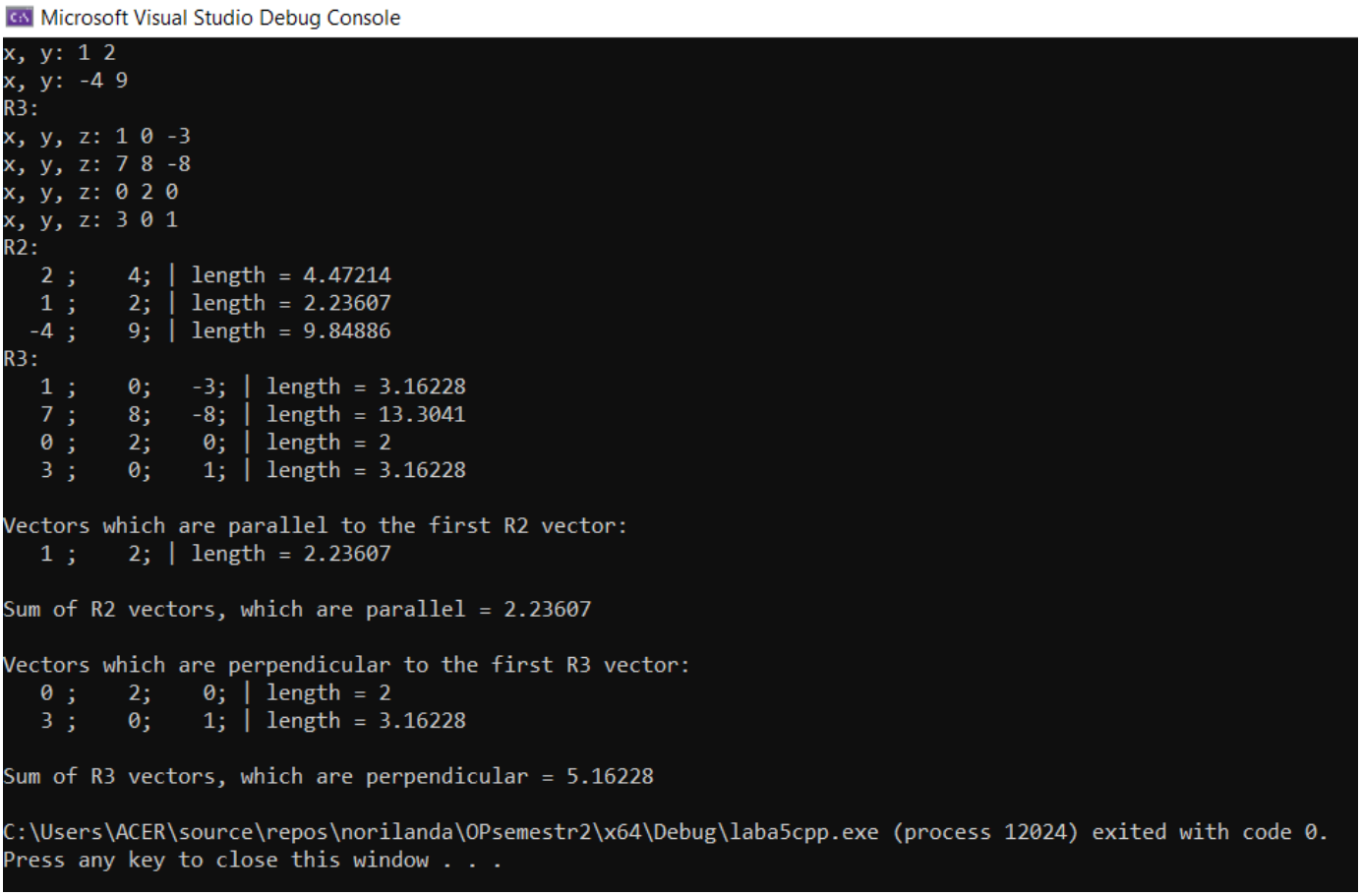
```
#pragma once
#include <vector>
#include "TVector.h"

using std::vector;

void create_arr_of_vectors(vector <TVector*>& arr, int n, int m);
void display_vector(vector <TVector*> arr, int start = 0, int end = 0);

double calc_sum_of_parallels(vector <TVector*> arr, int n, int m, vector <TVector*>& arrPar);
double calc_sum_of_perpendicular(vector <TVector*> arr, int n, int m, vector <TVector*>&
arrPerp);
```

//Work_with_vectors.cpp

```
#include <iostream>
#include <vector>
#include "TVector.h"
#include "Work_with_vectors.h"

using namespace std;

void create_arr_of_vectors(vector <TVector*>& arr, int n, int m)
{
	cout << "Random or manually? Random - enter 'r', manually - enter 'm'\n";
	char mode;
	cin >> mode;
	while (mode != 'r' && mode != 'm')
	{
		cout << "Try again! ";
		cin >> mode;
	}
	if (mode == 'r')
	{
		int rangeMin, rangeMax;
		cout << "Enter range (first min than max value): ";
		cin >> rangeMin;
		cin >> rangeMax;
		for (int i = 0; i < n; i++)
			arr.push_back(new R2(rangeMin, rangeMax));
		for (int i = 0; i < m; i++)
			arr.push_back(new R3(rangeMin, rangeMax));
	}
	else
	{
		double x, y, z;
		cout << "R2:\n";
		for (int i = 0; i < n; i++)
		{
			cout << "x, y: ";
			cin >> x;
			cin >> y;
			arr.push_back(new R2(x, y));
		}
		cout << "R3:\n";
		for (int i = 0; i < m; i++)
		{
			cout << "x, y, z: ";
			cin >> x;
			cin >> y;
```

```
                        cin >> z;
                        arr.push_back(new R3(x, y, z));
                }
        }
}

void display_vector(vector <TVector*> arr, int start , int end)
{
        if (end == 0)
                end = arr.size();
        if (end == 0)
                cout << " - ";
        for (int i = start; i < end; i++)
        {
                arr[i]->display();
                cout << "| length = " << arr[i]->calc_length() << "\n";
        }
}

double calc_sum_of_parallels(vector <TVector*> arr, int n, int m, vector <TVector*>& arrPar)
{
        double sum = 0;
        for (int i = 1; i < n; i++)
        {
                if (arr[0]->is_parallel(arr[i]))
                {
                        sum += arr[i]->calc_length();
                        arrPar.push_back(arr[i]);
                }
        }
        return sum;
}
double calc_sum_of_perpendicular(vector <TVector*> arr, int n, int m, vector <TVector*>& arrPerp)
{
        double sum = 0;
        for (int i = n + 1; i < n + m; i++)
        {
                if (arr[n]->is_perpendicular(arr[i]))
                {
                        sum += arr[i]->calc_length();
                        arrPerp.push_back(arr[i]);
                }
        }
        return sum;
}
```

Тестування програми:



2. Виконання завдання мовою Python

Код:

#laba5py.py(main)

```
from Work_with_vectors import *

n = 3
m = 4
arr = create_arr_of_vectors(n, m)
print("There are R2 and R3 vectors: ")
display_vector(arr)
print()

sum_par, arr_par = calc_sum_of_parallels(arr, n)
print("Vectors which are parallel to the first R2 vector:")
display_vector(arr_par)
print(f"Sum of R2 vectors, which are parallel = {sum_par}\n")

sum_perp, arr_perp = calc_sum_of_perpendicular(arr, n, m)
print("Vectors which are perpendicular to the first R3 vector:")
display_vector(arr_perp)
print(f"Sum of R3 vectors, which are perpendicular = {sum_perp}")
```

#TVector.py

```
from abc import ABC, abstractmethod
from math import sqrt

class TVector(ABC):
```

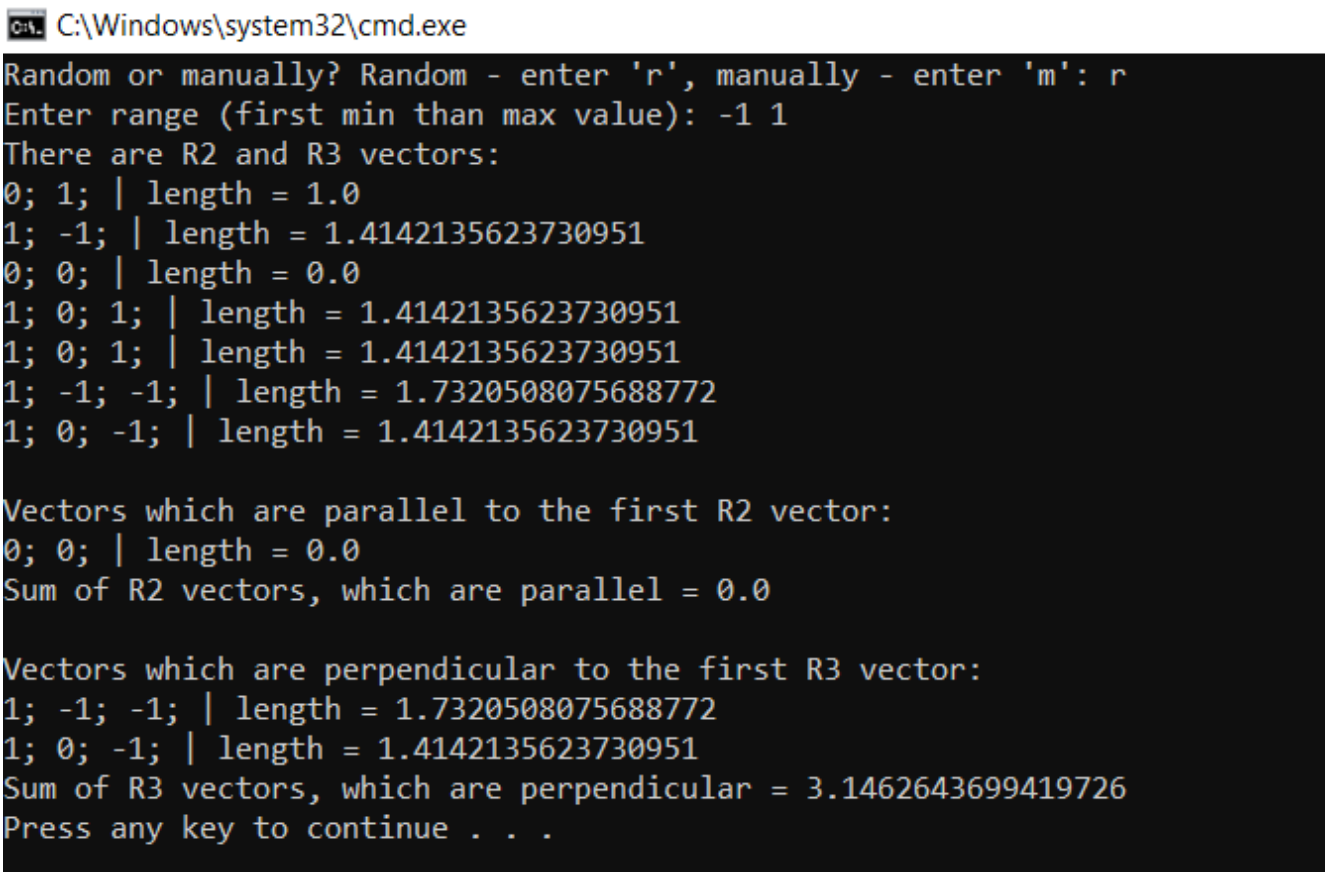
```
    def __init__(self, x):
        self.x = x

    def display(self):
        print(f"{self.x}; ", end = '')

    @abstractmethod
    def is_parallel(self, vector):
        pass

    @abstractmethod
    def is_perpendicular(self, vector):
        pass

    @abstractmethod
    def calc_length(self):
        pass


class R2(TVector):
    def __init__(self, x, y):
        super().__init__(x);
        self.__y = y

    def display(self):
        super().display()
        print(f"{self.__y}; ", end = '')

    def is_parallel(self, vector):
        cross_product = self.x * vector.__y - self.__y * vector.x
        if cross_product == 0:
            return True
        return False

    def is_perpendicular(self, vector):
        dot_product = self.x * vector.x + self.__y * vector.__y
        if dot_product == 0:
            return True
        return False

    def calc_length(self):
        return sqrt(self.x ** 2 + self.__y ** 2)


class R3(TVector):
    def __init__(self, x, y, z):
        super().__init__(x);
        self.__y = y
        self.__z = z

    def display(self):
        super().display()
        print(f"{self.__y}; {self.__z}; ", end = '')

    def is_parallel(self, vector):
        cross_product = (self.__y * vector.__z - vector.__y * self.__z) - (self.x * vector.__z - vector.x *
self.__z) + (self.x * vector.__y - vector.x * self.__y)
        if cross_product == 0:
            return True
        return False

    def is_perpendicular(self, vector):
        dot_product = self.x * vector.x + self.__y * vector.__y + self.__z * vector.__z
        if dot_product == 0:
            return True
        return False

    def calc_length(self):
```

```
        return sqrt(self.x ** 2 + self.__y ** 2 + self.__z ** 2)
```

#Work_with_vectors.py

```
from TVector import  R2, R3
import random

def create_arr_of_vectors(n, m):
    arr = []
    mode = input("Random or manually? Random - enter 'r', manually - enter 'm': ")
    while mode != 'r' and mode !='m':
        mode = input("Try again! ")
    if mode == 'r':
        limits = input("Enter range (first min than max value): ").split()
        rangeMin = int(limits[0])
        rangeMax = int(limits[1])
        for i in range(n):
            x = random.randint(rangeMin, rangeMax)
            y = random.randint(rangeMin, rangeMax)
            arr.append(R2(x, y))
        for i in range(m):
            x = random.randint(rangeMin, rangeMax)
            y = random.randint(rangeMin, rangeMax)
            z = random.randint(rangeMin, rangeMax)
            arr.append(R3(x, y, z))
    else:
        print("R2:")
        for i in range(n):
            values = input("x, y: ").split()
            x = int(values[0])
            y = int(values[1])
            arr.append(R2(x, y))
        for i in range(m):
            values = input("x, y, z: ").split()
            x = int(values[0])
            y = int(values[1])
            z = int(values[2])
            arr.append(R3(x, y, z))
    return arr


def display_vector(arr):
    for i in arr:
        i.display()
        print(f"| length = {i.calc_length()}")

def calc_sum_of_parallels(arr, n):
    arr_par = []
    sum_length = 0
    for i in range(1, n):
        if arr[0].is_parallel(arr[i]):
            arr_par.append(arr[i])
            sum_length += arr[i].calc_length()
    return sum_length, arr_par

def calc_sum_of_perpendicular(arr, n, m):
    arr_perp = []
    sum_length = 0
    for i in range(n + 1, n + m):
        if arr[n].is_perpendicular(arr[i]):
            arr_perp.append(arr[i])
            sum_length += arr[i].calc_length()
    return sum_length, arr_perp
```

Тестування програми:



Висновок: Під час виконання лабораторної роботи я вивчила механізми створення і використання класів та об’єктів. У результаті на основі створеного класу TVector, я створила два похідних класи R2 та R3.