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. Методології програмування»
«Успадкування та поліморфізм»

Варіант 06

Виконав студент ІП-13 Вдовиченко Станіслав Юрійович
(шифр, прізвище, ім'я, по батькові)

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

Київ 2022

Лабораторна робота №4
Успадкування та поліморфізм

Мета роботи: вивчити механізми створення і використання класів та об'єктів.

Завдання: створити клас TArray, який представляє одновимірний масив і містить методи збільшення / зменшення всіх елементів на вказану величину та знаходження їх середнього арифметичного. Реалізувати класи-нащадки, що представляють одновимірні масиви з елементами цілого та дійсного типів. Випадковим чином створити по m масивів кожного виду. Елементи цілочисельних масивів збільшити на вказану величину, а дійсних масивів зменшити на цю ж величину. Знайти масив, середнє арифметичне елементів якого є найбільшим.

Код програми на C++.

TArray.h

```
class TArray {
protected:
    int m_length;
    TArray(int length) : m_length(length){};

public:
    virtual void fill_array() = 0;
    virtual void increase(int) = 0;
    virtual void decrease(int) = 0;
    virtual double get_average() = 0;
    virtual void show_array() = 0;
    virtual ~TArray()= default;
};
```

IntArray.h

```
#include "TArray.h"
#include "../header.h"
using namespace std;

class IntArray: public TArray {
private:
    int m_length;
    int* array;
public:
    IntArray(): TArray(0), array(nullptr) {}
     explicit IntArray(int length): TArray(length){
        if(length > 0){
            array = new int [m_length];
        }
        else{
            array = nullptr;
        }
    }

    void fill_array() override{
        for (int i = 0; i < m_length; ++i) {
            array[i] = rand()%100 - 1;
        }
    }
     void increase(int value) override{
        for (int i = 0; i < m_length; ++i) {
            array[i] = array[i] + value;
        }
    }
     double get_average() override{
        double sum = 0;
         for (int i = 0; i < m_length; ++i) {
             sum+= array[i];
         }
         return sum / m_length;
    }

     void decrease(int value) override {
        for (int i = 0; i < m_length; ++i) {
            array[i] = array[i] - value;
        }
    }
    void show_array() override{
        for (int i = 0; i < m_length; ++i) {
```

```
            cout << array[i] << " ";
        }
        cout << endl;
    }
    ~IntArray() override{delete[] array;}
};
```

DoubleArray.h

```
#include "TArray.h"
#include "../header.h"
using namespace std;

class DoubleArray: public TArray {
private:
    int m_length;
    double* array;
public:
    DoubleArray(): TArray(0),array(nullptr) {}
    explicit DoubleArray(int length): TArray(length){
        if(length > 0){
            array = new double [m_length];
        }
        else{
            array = nullptr;
        }
    }
    void fill_array() override{
        for (int i = 0; i < m_length; ++i) {
            array[i] = (rand()%1000 - 10) / 10.0;
        }
    }
    void increase (int value) override{
        for (int i = 0; i < m_length; ++i) {
            array[i] = array[i] + value;
        }
    }
    double get_average() override{
        double sum = 0;
        for (int i = 0; i < m_length; ++i) {
            sum+= array[i];
        }
        return sum / m_length;
    }
    void decrease (int value) override {
        for (int i = 0; i < m_length; ++i) {
            array[i] = array[i] - value;
        }
    }
    void show_array() override{
        for (int i = 0; i < m_length; ++i) {
            cout << array[i] << " ";
        }
        cout << endl;
    }
    ~DoubleArray() override{delete[] array;}
};
```

Header.h

```
#include "iostream"
#include "string"
```

```
#include "cstdlib"
#include "ctime"
#include <vector>
#include "classes/TArray.h"
#include "classes/IntArray.h"
#include "classes/DoubleArray.h"
using namespace std;

void init_arrays(int, vector<IntArray>&, vector<DoubleArray>&);
void show_arrays(vector<DoubleArray>, int);
void show_arrays(vector<IntArray>, int);
void increase_int_arrays(vector<IntArray>&, int, int);
void decrease_double_arrays(vector<DoubleArray>&, int, int);
double find_average(vector<IntArray>, vector<DoubleArray>, int, int&,
string&);
void show_average_array(vector<IntArray>, vector<DoubleArray>, int,
string);
```

Header.cpp

```
#include "header.h"
void init_arrays(int m, vector<IntArray>& int_arrays, vector<DoubleArray>&
double_arrays){
    const int SIZE = 5;
    for (int i = 0; i < m; ++i) {
        int_arrays.push_back(IntArray(SIZE));
        int_arrays[i].fill_array();
        double_arrays.push_back(DoubleArray(SIZE));
        double_arrays[i].fill_array();
    }
}
void show_arrays(vector<DoubleArray> double_arrays, int m){
    for (int i = 0; i < m; ++i) {
        double_arrays[i].show_array();
    }
    cout << endl;
}
void show_arrays(vector<IntArray> int_arrays, int m){
    for (int i = 0; i < m; ++i) {
        int_arrays[i].show_array();
    }
    cout << endl;
}
void increase_int_arrays(vector<IntArray>& int_arrays, int m, int value){
    for (int i = 0; i < m; ++i) {
        int_arrays[i].increase(value);
    }
}
void decrease_double_arrays(vector<DoubleArray>& double_arrays, int m, int
value){
    for (int i = 0; i < m; ++i) {
        double_arrays[i].decrease(value);
    }
}
double find_average(vector<IntArray> int_arrays, vector<DoubleArray>
double_arrays, int m, int &index, string &type){
    double temp;
    double average;
    type = "int";
    for (int i = 0; i < m; ++i) {
        temp = int_arrays[i].get_average();
        if (temp >= average) {
            average = temp;
```

```
            index = i;
        }
    }
    for (int i = 0; i < m; ++i) {
        temp = double_arrays[i].get_average();
        if(temp >= average) {
            average = temp;
            index = i;
            type = "double";
        }
    }
    return average;
}
void show_average_array(vector<IntArray> int_arrays, vector<DoubleArray>
double_arrays, int index, string type){
    cout << endl;
    cout << "Type: " << type << ". Index: " << index + 1 << endl;
    return type == "int" ? int_arrays[index].show_array() :
double_arrays[index].show_array();
}
```

Main.cpp

```
#include "header.h"
int main() {
    srand(time(nullptr));
    int m;
    double average;
    int value;
    int index;
    string type;
    do {
        cout << "Enter the number of arrays to be created: ";
        cin >> m;
    }while(m <= 0);
    vector<IntArray> int_arrays;
    vector<DoubleArray> double_arrays;
    init_arrays(m, int_arrays, double_arrays);
    cout << endl;
    show_arrays(int_arrays, m);
    show_arrays(double_arrays, m);
    cout << "Enter the value: "; cin >> value;
    increase_int_arrays(int_arrays, m, value);
    decrease_double_arrays(double_arrays, m, value);
    cout << endl << "Arrays after operations: " << endl;
    show_arrays(int_arrays, m);
    show_arrays(double_arrays, m);
    average = find_average(int_arrays, double_arrays, m, index, type);
    cout << endl << "Max average: " << average;
    show_average_array(int_arrays, double_arrays, index, type);
    return 0;
}
```

Код програми на Python.

T_array.py

```
class TArray:
    def __init__(self, length: int):
        self.size = length
        self.array = [0 for i in range (self.size)]
        self.fill_array()

    def __str__(self):
        pass

    def increase(self, value):
        for i in range(self.size):
            self.array[i] += value

    def decrease(self, value):
        for i in range(self.size):
            self.array[i] -= value

    def fill_array(self):
        pass
```

int_array.py

```
from random import *
from t_array import TArray


class IntArray(TArray):
    def get_average(self):
        sum = 0
        for i in self.array:
            sum += i
        return round(sum / self.size, 3)

    def __str__(self):
        return ' '.join([str(x).rjust(3, ' ') for x in self.array])

    def fill_array(self):
        for i in range(self.size):
            self.array[i] = randint(-100, 100)
```

float_array.py

```
from random import *
from t_array import TArray


class FloatArray(TArray):
    def get_average(self):
        sum = 0
        for i in self.array:
            sum += i
            return round(sum / self.size, 2)

    def __str__(self):
        return ' '.join(['{:.2f}'.format(x).rjust(6, ' ') for x in
self.array])
```

```
    def fill_array(self):
        for i in range(self.size):
            self.array[i] = round(random() + randint(-100, 100), 2)
```

main.py

```
from int_array import *
from float_array import *


def print_arrays(header: str, arrays: list[TArray]):
    print(header)
    for array in arrays:
        print(array)
    print("      ")

def get_max_average(int_arrays: list[IntArray], float_arrays:
list[FloatArray] ):
    max_average = int_arrays[0].get_average()
    type = "int"
    index = 0
    array = []
    for i in range(len(int_arrays)):
        if int_arrays[i].get_average() >= max_average:
            max_average = int_arrays[i].get_average()
            index = i
            array = int_arrays[i]
    for j in range(len(float_arrays)):
        if float_arrays[j].get_average() >= max_average:
            max_average = float_arrays[j].get_average()
            type = "float"
            index = j
            array = float_arrays[j]
    return array, type, index, max_average


def main():
    int_arrays = []
    float_arrays = []
    m = 0
    while m <= 0:
        m = int(input("Enter the number of arrays to be created: "))
    for i in range(m):
        int_arrays.append(IntArray(5))
        float_arrays.append(FloatArray(5))
    print_arrays('Int arrays: ', int_arrays)
    print_arrays('Float arrays: ', float_arrays)

    value = int(input("Enter the value: "))
    for i in range(m):
        int_arrays[i].increase(value)
        float_arrays[i].decrease(value)
    print_arrays('Int arrays after operations: ', int_arrays)
    print_arrays('Float arrays after operations: ', float_arrays)
    array, kind, index, max_average = get_max_average(int_arrays,
float_arrays)
    print(f'Max average: {max_average} \nType: {kind}. Index:
{index+1}.\nArray: {array}')


if __name__ == '__main__':
```

```
    main()
```

Робота програми

1.C++

```
C:\Users\Stas\CLionProjects\Lab5ver2\cmake-build-debug\Lab5ver2.exe
Enter the number of arrays to be created:5

55 33 65 4 58
-1 74 51 68 76
94 8 93 12 96
93 34 52 3 11
9 82 29 86 90

92.3 43.9 47.8 11.6 26.7
56.3 -0.2 73.8 26.4 82
75.7 43.1 92.9 6 85.3
68.2 71.5 54.5 3.8 14.3
97.2 6.4 59.8 55.9 7

Enter the value:14

Arrays after operations:
69 47 79 18 72
13 88 65 82 90
108 22 107 26 110
107 48 66 17 25
23 96 43 100 104

78.3 29.9 33.8 -2.4 12.7
42.3 -14.2 59.8 12.4 68
61.7 29.1 78.9 -8 71.3
54.2 57.5 40.5 -10.2 0.3
83.2 -7.6 45.8 41.9 -7


Max average: 74.6
Type: int. Index: 3
108 22 107 26 110

Process finished with exit code 0
```

2.Python

```
C:\Users\Stas\PycharmProjects\Lab5\venv\Scripts\python.exe C:/Users/Stas/PycharmProjects/Lab5/main.py
Enter the number of arrays to be created: 5
Int arrays:
-86  23 -20 -57 -41
 35  56  50  37 -36
 38  60 -37 -51  -1
-41 -28 -17  72   0
 -4  87 -42  77 -66

Float arrays:
 55.48   2.68 -46.00  99.75  24.52
-46.28  96.09 -51.63  58.35  95.86
 97.58 -11.43  97.72  -9.02  53.02
-43.09 -67.60 -80.16 -87.55 -26.42
-27.46 -58.29  84.04  71.66 -97.06

Enter the value: 10
Int arrays after operations:
-71  38  -5 -42 -26
 50  71  65  52 -21
 53  75 -22 -36  14
-26 -13  -2  87  15
 11 102 -27  92 -51

Float arrays after operations:
 40.48 -12.32 -61.00  84.75   9.52
-61.28  81.09 -66.63  43.35  80.86
 82.58 -26.43  82.72 -24.02  38.02
-58.09 -82.60 -95.16 -102.55 -41.42
-42.46 -73.29  69.04  56.66 -112.06

Max average: 43.4
Type: int. Index: 2.
Array:  50  71  65  52 -21

Process finished with exit code 0
```

Висновок: під час виконання даної лабораторної роботи я вивчив механізми створення та використання класів та об’єктів.