Додаток 1

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»

«Успадкування та поліморфізм»

Варіант 15

Виконав студент ІП-13 Карамян Вартан Суренович
(шифр, прізвище, ім'я, по батькові)

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

Київ 2022

Лабораторна робота 4

Успадкування та поліморфізм

Варіант 15

Спроектувати клас TFunction, який представляє функцію і містить методи збільшення/зменшення всіх коефіцієнтів функції на вказану величину та обчислити значення функції в заданій точці. На основі цього класу створити класи-нащадки Лінійна функція (виду a1x+a0) та Квадратична функція (виду b2x^2+b1x+b0). Створити n лінійних функцій та m квадратичних функцій, згенерувавши дані для них випадковим чином. Коефіцієнти лінійних ф-цій збільшити на 3, а квадратичних - зменшити на 2. Визначити функцію, яка має найбільше значення у введеній користувачем точці.

Код програми на мові C++:

lab4 C++.cpp

```
#include "funcs.h"

int main()
{
    srand(time(NULL));
    vector<LinearFunction> list1 = createLinearList();
    vector<QuadraticFunction> list2 = createQuadraticList();

    changeFunctoin(list1, '+', 3);
    cout << "\nLinear functions increased by 3:\n";
    printList(list1);

    changeFunctoin(list2, '-', 2);
    cout << "\nQuadratic functions decreased by 2:\n";
    printList(list2);

    double x;
    cout << "\nEnter value of x0:"; cin >> x;

    LinearFunction maxLin = maxFunction(list1, x);
    QuadraticFunction maxQuad = maxFunction(list2, x);

    cout << "\nFunction with a max value in a dot " << x << ":\n";
    if (maxLin.calculate(x) > maxQuad.calculate(x))
    {
        maxLin.print();
        cout << "Value = " << maxLin.calculate(x) << endl;
    }
    else
```

```
    {
        maxQuad.print();
        cout << "Value = " << maxQuad.calculate(x) << endl;
    }

}
```

TFunction.h

```
#pragma once
#include <iostream>
#include <ctime>
#include <vector>

using namespace std;

class TFunction
{
protected:
    double a, b, c;
public:

    TFunction(double, double, double = 0);

    void increaseBy(double);

    void decreaseBy(double);

    virtual void print() = 0;

    virtual double calculate(double) = 0;

};
```

TFunction.cpp

```
#include "TFunction.h"

TFunction::TFunction(double a, double b, double c)
{
    this->a = a;
    this->b = b;
    this->c = c;
}

void TFunction::increaseBy(double num)
{
    a += num;
    b += num;
    c += num;
}

void TFunction::decreaseBy(double num)
{
    a -= num;
    b -= num;
    c -= num;
}
```

LinearFunction.h

```h
#pragma once
#include "TFunction.h"

class LinearFunction : public TFunction
{
public:
    LinearFunction(double a, double b) :
        TFunction(a, b)
    {
    }
    void print();
    double calculate(double x);
};
```

LinearFunction.cpp

```cpp
#include "LinearFunction.h"

void LinearFunction::print()
{
    if (a == 0)
        cout << "y = " << b;
    else if (a == 1)
        cout << "y = x";
    else if (a == -1)
        cout << "y = -x";
    else
        cout << "y = " << a << "x";

    if (b > 0)
        cout << " + " << b;
    else if (b < 0)
        cout << " - " << abs(b);
    cout << endl;
}

double LinearFunction::calculate(double x)
{
    return a * x + b;
}
```

QuadraticFunction.h

```h
#pragma once
#include "TFunction.h"

class QuadraticFunction : public TFunction
{
public:
    QuadraticFunction(double a, double b, double c) :
        TFunction(a, b, c)
    {}
    void print();
    double calculate(double);
};
```

QuadraticFunction.cpp

```cpp
#include "QuadraticFunction.h"
void QuadraticFunction::print()
{
    if (a == 0)
        cout << "y = ";
    else if (a == 1)
        cout << "y = x^2";
    else if (a == -1)
        cout << "y = -x^2";
    else
        cout << "y = " << a << "x^2";


    if (b == 1)
        cout << " + x";
    else if (b == -1)
        cout << " - x";
    else if (b != 0 )
    {
        if (a != 0)
        {
            if (b > 0)
                cout << " + " << b << "x";
            else
                cout << " - " << abs(b) << "x";
        }
        else
        {
            cout << " " << b << "x";
        }
    }

    if (a == 0 && b == 0)
        cout << c;
    else if (c > 0)
        cout << " + " << c;
    else if (c < 0)
        cout << " - " << abs(c);


    cout << endl;
}
double QuadraticFunction::calculate(double x)
{
    return a * pow(x, 2) + b * x + c;
}
```

Funcs.h

```cpp
#pragma once
#include "LinearFunction.h"
#include "QuadraticFunction.h"

vector<LinearFunction> createLinearList();

vector<QuadraticFunction> createQuadraticList();

template<class T>
void changeFunctoin(vector<T>& list, char mode, int num);

template<class T>
void printList(vector<T> list);

template<class T>
T maxFunction(vector<T> list, double x);
```

```
#include "templateFuncs.cpp"
```

Funcs.cpp

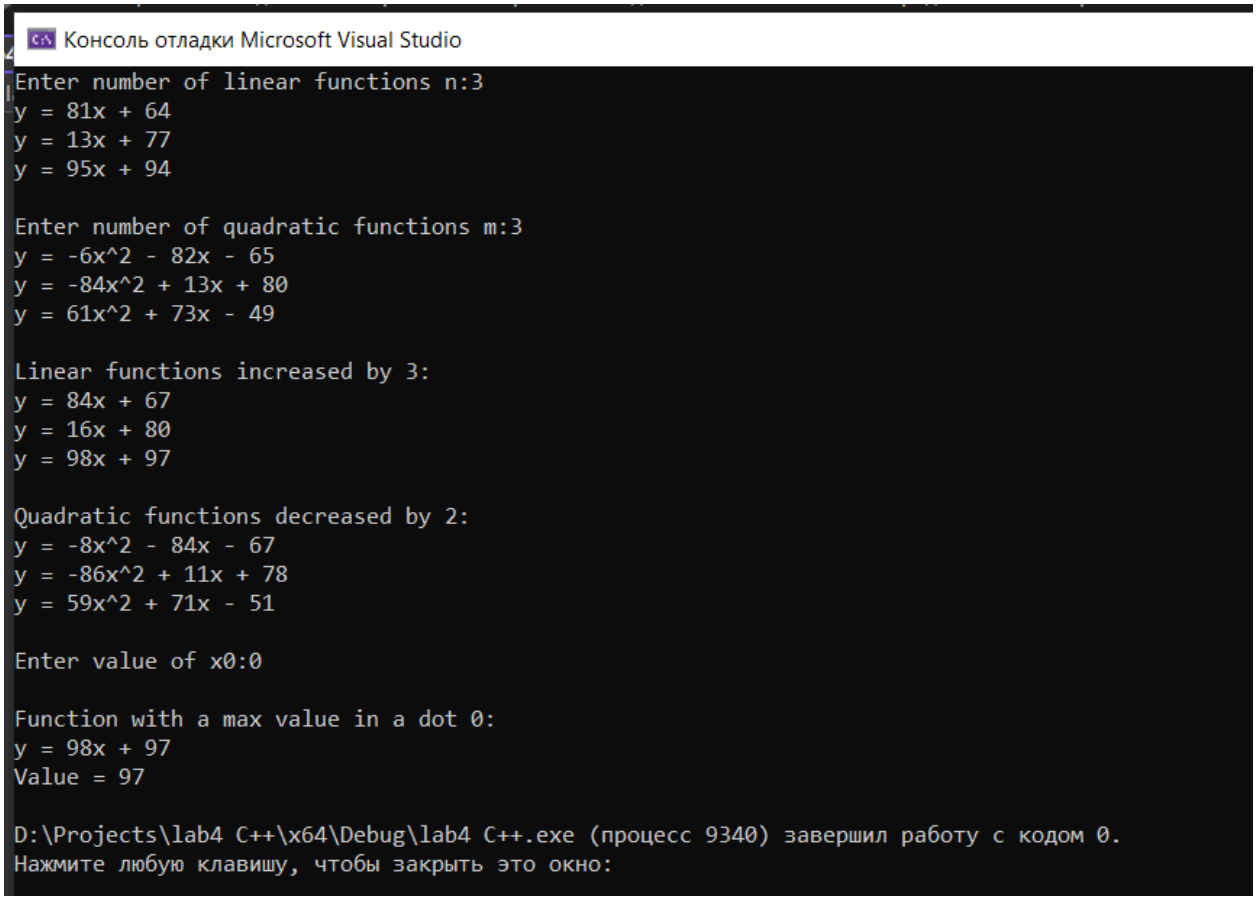
```
#include "funcs.h"

vector<LinearFunction> createLinearList()
{
   vector<LinearFunction> list;
   int size;
   double a, b;
   cout << "Enter number of linear functions n:";
   cin >> size;

   for (size_t i = 0; i < size; i++)
   {
      a = rand() % 101;
      b = rand() % 101;
      LinearFunction y(a, b);
      y.print();
      list.push_back(y);
   }
   return list;
}

vector<QuadraticFunction> createQuadraticList()
{
   vector<QuadraticFunction> list;
   int size;
   double a, b, c;
   cout << "\nEnter number of quadratic functions m:";
   cin.ignore();
   cin >> size;

   for (size_t i = 0; i < size; i++)
   {
      a = rand() % 201 - 100;
      b = rand() % 201 - 100;
      c = rand() % 201 - 100;
      QuadraticFunction y(a, b, c);
      y.print();
      list.push_back(y);
   }
   return list;
}
```

TemplateFuncs.cpp

```
#include "funcs.h"

template<class T>
void changeFunctoin(vector<T> &list, char mode, int num)
{
   for (size_t i = 0; i < list.size(); i++)
   {
      if (mode == '+')
         list[i].increaseBy(num);
      else if (mode == '-')
         list[i].decreaseBy(num);
   }

}

template<class T>
```

```
void printList(vector<T> list)
{
    for (size_t i = 0; i < list.size(); i++)
    {
        list[i].print();
    }
}

template<class T>
T maxFunction(vector<T> list, double x)
{
    double maxValue = list[0].calculate(x);
    T maxFunc = list[0];
    for (size_t i = 1; i < list.size(); i++)
    {
        if (list[i].calculate(x) > maxValue)
        {
            maxFunc = list[i];
            maxValue = list[i].calculate(x);
        }
    }
    return maxFunc;
}
```

Результати роботи:



Код програми на мові Python:

main.py

```
from module import *

list1 = create_linear_list()
list2 = create_quadratic_list()

print("Linear functions: ")
print_list(list1)

print("Quadratic functions: ")
print_list(list2)

change_funcs(list1, '+', 3)
print("\nIncreased by 3 linear functions: ")
print_list(list1)

change_funcs(list2, '-', 2)
print("Decreased by 2 quadratic functions: ")
print_list(list2)

x0 = int(input("Enter value of a dot: "))
max_func, max_value = find_max_function(list1, list2, x0)
print(f"Function with a max value in a dot {x0}")
max_func.print()
print(f"Value = {max_value}")
```

TFunction.py

```
class TFunction:
    def __init__(self, a, b, c=0):
        self._a = a
        self._b = b
        self._c = c

    def increase_by(self, num):
        self._a += num
        self._b += num
        self._c += num

    def decrease_by(self, num):
        self._a -= num
        self._b -= num
        self._c -= num

    def print(self):
        pass

    def calculate(self, x):
        pass
```

LinearFunction.py

```python
from TFunction import TFunction


class LinearFunction(TFunction):

    def calculate(self, x):
        return self._a * x + self._b

    def print(self):
        if self._a == 0:
            print(f"y = {self._b}", end="")
        elif self._a == 1:
            print("y = x", end="")
        elif self._a == -1:
            print("y = -x", end="")
        else:
            print(f"y = {self._a}x", end="")

        if self._b > 0:
            print(f" + {self._b}")
        elif self._b < 0:
            print(f" - {abs(self._b)}")
```

QuadraticFunction.py

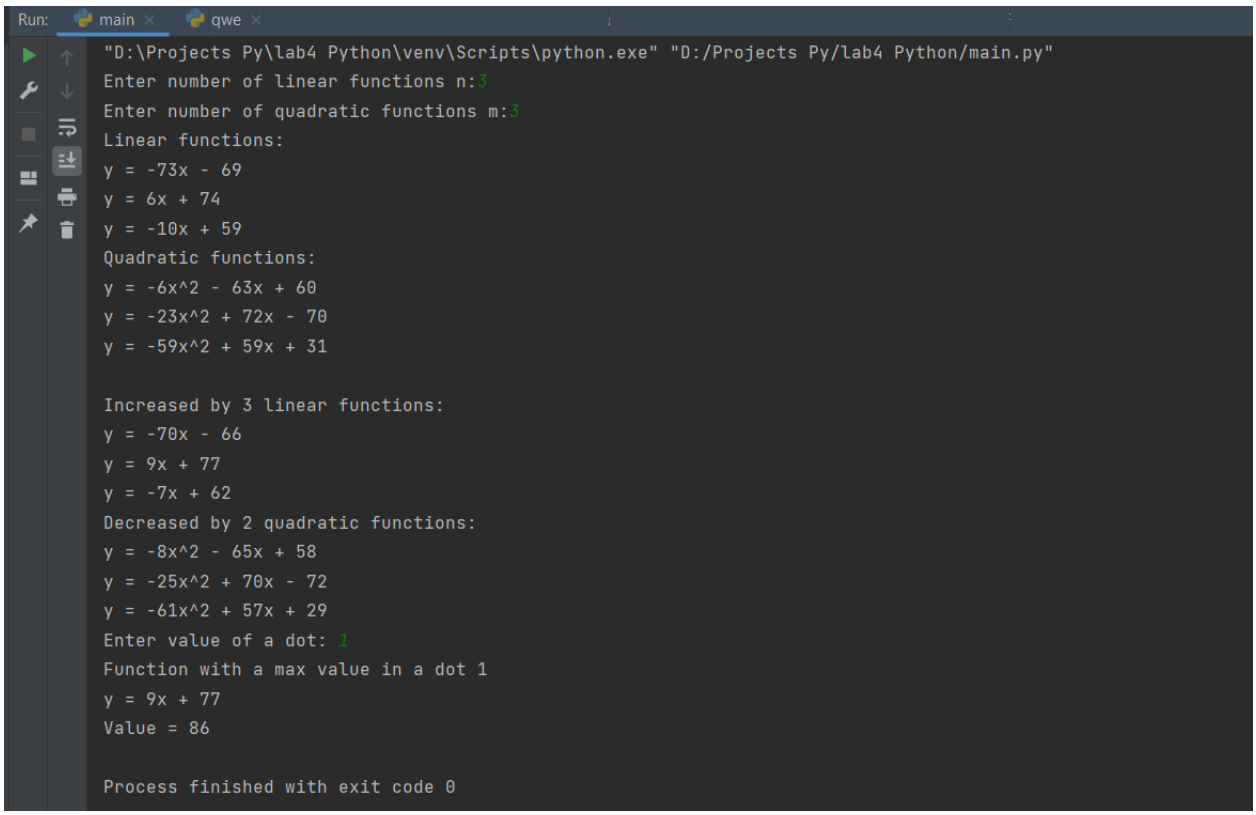
```python
from TFunction import TFunction


class QuadraticFunction(TFunction):

    def calculate(self, x):
        return self._a * x ** 2 + self._b * x + self._c

    def print(self):
        if self._a == 0:
            print("y = ", end="")
        elif self._a == 1:
            print("y = x^2", end="")
        elif self._a == -1:
            print("y = -x", end="")
        else:
            print(f"y = {self._a}x^2", end="")

        if self._b == 1:
            print(" + x", end="")
        elif self._b == -1:
            print(" - x", end="")
        elif self._b != 0:
            if self._a != 0:
                if self._b > 0:
                    print(f" + {self._b}x", end="")
                else:
                    print(f" - {abs(self._b)}x", end="")
            else:
                print(f" {self._b}x", end="")

        if self._a == 0 and self._b == 0:
            print(self._c)
        elif self._c > 0:
```

```
            print(f" + {self._c}")
        elif self._c < 0:
            print(f" - {abs(self._c)}")
```

Module.py

```
from LinearFunction import LinearFunction
from QuadraticFunction import QuadraticFunction
from random import randint


def create_linear_list():
    size = int(input("Enter number of linear functions n:"))
    list = []
    for i in range(size):
        y = LinearFunction(randint(-100, 100), randint(-100, 100))
        list.append(y)
    return list


def create_quadratic_list():
    size = int(input("Enter number of quadratic functions m:"))
    list = []
    for i in range(size):
        y = QuadraticFunction(randint(-100, 100), randint(-100, 100),
randint(-100, 100))
        list.append(y)
    return list


def print_list(list):
    for func in list:
        func.print()


def change_funcs(list, mode, num):
    for func in list:
        if mode == '+':
            func.increase_by(num)
        elif mode == '-':
            func.decrease_by(num)


def max_function(list, x):
    max_f = list[0]
    max_val = list[0].calculate(x)
    for i in range(1, len(list)):
        if list[i].calculate(x) > max_val:
            max_f = list[i]
            max_val = list[i].calculate(x)
    return max_f, max_val


def find_max_function(linear_list, quadratic_list, x):
    max_linear, max_lin_value = max_function(linear_list, x)
    max_quadratic, max_quad_value = max_function(quadratic_list, x)
    if max_lin_value > max_quad_value:
        return max_linear, max_lin_value
    else:
        return max_quadratic, max_quad_value
```

Результати роботи:



Висновок:

В результаті виконання лабораторної роботи ми вивчили та закріпили на практиці такий розділ ООП як наслідування та поліморфізм.