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 № 3 з дисципліни
«Основи програмування 2. Модульне програмування»

«Перевантаження операторів»

Варіант 9

Виконав студент ІП-15, Дзюбенко Даниїл Дмитрович
(шифр, прізвище, ім'я, по батькові)

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

Київ 2022

Лабораторна робота 3

Перевантаження операторів

Мета – вивчити механізми створення класів з використанням перевантажених операторів (операцій).

Варіант 9

Задача

9. Визначити клас "Багаточлен" ступеня 3, членами якого є коефіцієнти полінома. Реалізувати для нього декілька конструкторів, геттери, метод обчислення значення поліному в заданій точці. Перевантажити оператори додавання "+" і множення "*" поліномів. Створити три поліноми (P1, P2, P3), використовуючи різні конструктори. Визначити новий поліном P4 як суму поліномів P1 та P2 і новий поліном P5 як добуток поліномів P2 та P3. Обчислити значення поліномів P4 і P5 в заданій точці.

Код

```cpp
#include <iostream>
#include <vector>
#include "header.h"
using namespace std;

void main() {
    Polynomial P1, P2, P3, P4;
    vector<double> P5;
    double point;
    cout << "Polynomial 1 was created by default constructor!\n";
    cout << "\nPolynomial 2:\n";
    P2 = createPolynomial();
    cout << "\nPolynomial 3 was created by copy constructor!\n";
    P3 = Polynomial(P2);

    P4 = P1 + P2;
    P5 = P2 * P3;

    cout << "\nEnter a point: ";
    cin >> point;

    cout << "\nPolynomial 1:\n";
    printPolynomial(P1);

    cout << "\nPolynomial 2:\n";
    printPolynomial(P2);

    cout << "\nPolynomial 3:\n";
    printPolynomial(P3);

    cout << "\nPolynomial 4:\n";
    printPolynomial(P4);
    cout << "Its value at given point: " << P4.calcPolynomialValueAtGivenPoint(point) << endl;

    cout << "\nPolynomial 5:\n";
    printPolynomial(P5);
    cout << "Its value at given point: " << calcPolynomialValueAtGivenPoint(point, P5) << endl;

    system("pause");
}
```

```h
#pragma once
#include <iostream>
#include <vector>
using namespace std;

class Polynomial {
    private:
        double first_coefficient, second_coefficient, third_coefficient, fourth_coefficient;
    public:
        Polynomial(double first, double second, double third, double fourth);
        Polynomial();
        Polynomial operator+(const Polynomial p);
        vector<double> operator*(const Polynomial p);
        void setFirstCoefficient(double coefficient);
        void setSecondCoefficient(double coefficient);
        void setThirdCoefficient(double coefficient);
        void setFourthCoefficient(double coefficient);
        double getFirstCoefficient();
        double getSecondCoefficient();
        double getThirdCoefficient();
        double getFourthCoefficient();
        double calcPolynomialValueAtGivenPoint(double point);
};

Polynomial createPolynomial();
void printPolynomial(Polynomial p);
void printPolynomial(vector<double> p);
double calcPolynomialValueAtGivenPoint(double point, vector<double> p);
```

```cpp
#include "header.h"

Polynomial::Polynomial(double first, double second, double third, double fourth) {
    first_coefficient = first;
    second_coefficient = second;
    third_coefficient = third;
    fourth_coefficient = fourth;
}

Polynomial::Polynomial(){
    first_coefficient = 1;
    second_coefficient = 2;
    third_coefficient = 3;
    fourth_coefficient = 4;
}

Polynomial Polynomial::operator+(Polynomial p) {
    Polynomial temp;
    temp.setFirstCoefficient(first_coefficient + p.getFirstCoefficient());
    temp.setSecondCoefficient(second_coefficient + p.getSecondCoefficient());
    temp.setThirdCoefficient(third_coefficient + p.getThirdCoefficient());
    temp.setFourthCoefficient(fourth_coefficient + p.getFourthCoefficient());
    return temp;
}

vector<double> Polynomial::operator*(Polynomial p) {
    vector<double> temp;
    double first, second, third, fourth, fifth, sixth, seventh;
    first = first_coefficient * p.getFirstCoefficient();
    second = first_coefficient * p.getSecondCoefficient() + second_coefficient * p.getFirstCoefficient();
    third = first_coefficient * p.getThirdCoefficient() + second_coefficient * p.getSecondCoefficient() + third_coefficient * p.getFirstCoefficient();
    fourth = first_coefficient * p.getFourthCoefficient() + second_coefficient * p.getThirdCoefficient() + third_coefficient * p.getSecondCoefficient() +
      fourth_coefficient * p.getFirstCoefficient();
    fifth = second_coefficient * p.getFourthCoefficient() + third_coefficient * p.getThirdCoefficient() + fourth_coefficient * p.getSecondCoefficient();
    sixth = third_coefficient * p.getFourthCoefficient() + fourth_coefficient * p.getThirdCoefficient();
    seventh = fourth_coefficient * p.getFourthCoefficient();
    temp.insert(temp.end(), {first, second, third, fourth, fifth, sixth, seventh});
    return temp;
}
```

```cpp
void Polynomial::setFirstCoefficient(double coefficient) {
    first_coefficient = coefficient;
}

void Polynomial::setSecondCoefficient(double coefficient) {
    second_coefficient = coefficient;
}

void Polynomial::setThirdCoefficient(double coefficient) {
    third_coefficient = coefficient;
}

void Polynomial::setFourthCoefficient(double coefficient) {
    fourth_coefficient = coefficient;
}

double Polynomial::getFirstCoefficient(){
    return first_coefficient;
}

double Polynomial::getSecondCoefficient() {
    return second_coefficient;
}

double Polynomial::getThirdCoefficient() {
    return third_coefficient;
}

double Polynomial::getFourthCoefficient() {
    return fourth_coefficient;
}

double Polynomial::calcPolynomialValueAtGivenPoint(double point) {
    return first_coefficient * pow(point, 3) + second_coefficient * pow(point, 2) + third_coefficient * point + fourth_coefficient;
}

double calcPolynomialValueAtGivenPoint(double point, vector<double> p) {
    return p[0] * pow(point, 6) + p[1] * pow(point, 5) + p[2] * pow(point, 4) + p[3] * pow(point, 3) + p[4] * pow(point, 2) + p[5] * point + p[6];
}
```

```cpp
Polynomial createPolynomial() {
    double first, second, third, fourth;
    cout << "Enter first coefficient: ";
    cin >> first;
    cout << "Enter second coefficient: ";
    cin >> second;
    cout << "Enter third coefficient: ";
    cin >> third;
    cout << "Enter fourth coefficient: ";
    cin >> fourth;
    Polynomial p = Polynomial(first, second, third, fourth);
    return p;
}

void printPolynomial(Polynomial p) {
    cout << "f(x) = (" << p.getFirstCoefficient() << ")x^3 + (" << p.getSecondCoefficient() << ")x^2 + (" << p.getThirdCoefficient() << ")x + (" <<
      p.getFourthCoefficient() << ")\n";
}

void printPolynomial(vector<double> p) {
    cout << "f(x) = (" << p[0] << ")x^6 + (" << p[1] << ")x^5 + (" << p[2] << ")x^4 + (" << p[3] << ")x^3 + (" << p[4] << ")x^2 + (" << p[5] << ")x + (" << p[6] << ")
      \n";
}
```

```
Polynomial 1 was created by default constructor!

Polynomial 2:
Enter first coefficient: 12
Enter second coefficient: 3
Enter third coefficient: 4
Enter fourth coefficient: 5

Polynomial 3 was created by copy constructor!

Enter a point: 3

Polynomial 1:
f(x) = (1)x^3 + (2)x^2 + (3)x + (4)

Polynomial 2:
f(x) = (12)x^3 + (3)x^2 + (4)x + (5)

Polynomial 3:
f(x) = (12)x^3 + (3)x^2 + (4)x + (5)

Polynomial 4:
f(x) = (13)x^3 + (5)x^2 + (7)x + (9)
Its value at given point: 426

Polynomial 5:
f(x) = (144)x^6 + (72)x^5 + (105)x^4 + (144)x^3 + (46)x^2 + (40)x + (25)
Its value at given point: 135424
Для продолжения нажмите любую клавишу . . .
```