Додаток А

Міністерство Освіти і науки України
КПІ ім. Ігоря Сікорського
Кафедра ІПІ

ЗВІТ
з виконання лабораторної роботи № 4
з кредитного модуля
“Основи програмування-2. Методологія програмування”

Варіант № 9

Виконав:
студент 1-го курсу
гр. ІП-25 ФІОТ
Карпов Любомир Васильович

Київ 2023

Постановка задачі

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.

Текст програми

main.cpp

```cpp
#include <iostream>
#include "Polynomial.h"


int main() {
    double coefs[] = {2, 0, 0, -7};
    Polynomial p1(coefs); // 2x^3 - 7
    std::cout << "p1: " << p1.to_str() << '\n';

    Polynomial p2(1, 2, 3, 4); // x^3 + 2x^2 + 3x + 4
    std::cout << "p2: " << p2.to_str() << '\n';

    Polynomial p3(3); // x^3
    std::cout << "p3: " << p3.to_str() << '\n';


    Polynomial p4 = p1 + p2; // sum of p1 and p2
    std::cout << "p4: " << p4.to_str() << '\n';

    Polynomial p5 = p2 * p3; // product of p2 and p3
    std::cout << "p5: " << p5.to_str() << '\n';


}
```

Polynomial.h

```cpp
#ifndef LAB4_POLYNOMIAL_H
#define LAB4_POLYNOMIAL_H

#include <cmath>
#include <cstdarg>
#include <iostream>


class Polynomial {
    const int degree;
    double *a;

    Polynomial(int degree, double *a) : degree(degree), a(a) {}
```

```
public:

    Polynomial(double a, double b, double c, double d);

    Polynomial(const double coefs[4]);

    Polynomial(int degree);

    Polynomial() : degree(0), a(nullptr) {};

    Polynomial(const Polynomial &);

    ~Polynomial();

    Polynomial operator+(Polynomial other) const;

    Polynomial operator*(Polynomial other) const;


    double enumerate(double x) const;

    double get_coef(int num);

    void set_coef(int i, double coef);

    int get_degree() { return degree; }

    std::string to_str();


};


#endif //LAB4_POLYNOMIAL_H
```

Polynomial.cpp

```
#include "Polynomial.h"


double Polynomial::enumerate(double x) const {
    double res = 0;
    for (int i = 0; i < degree; ++i) {
        res += a[i] * pow(x, i);
    }
    return res;
}

Polynomial Polynomial::operator+(Polynomial other) const {
    int max_degree;
    if (degree >= other.degree)
        max_degree = degree;
    else
```

```
        max_degree = other.degree;

    auto *b = new double[max_degree + 1];

    for (int i = 0; i <= max_degree; ++i)
        b[i] = 0;

    for (int i = 0; i <= degree; ++i)
        b[i] += a[i];

    for (int i = 0; i <= other.degree; ++i)
        b[i] += other.a[i];


    return Polynomial(max_degree, b);
}


Polynomial Polynomial::operator*(Polynomial other) const {
    int degree = this->degree + other.degree;

    auto *b = new double[degree + 1];

    for (int i = 0; i <= degree; ++i)
        b[i] = 0;


    for (int i = 0; i <= this->degree; ++i) {
        for (int j = 0; j <= other.degree; ++j) {
            b[i + j] += this->a[i] * other.a[j];
        }
    }

    return {degree, b};
}

double Polynomial::get_coef(int num) {
    if (num >= degree)
        throw std::errc::result_out_of_range;

    return a[num];
}

std::string Polynomial::to_str() {
    std::string str;
    for (int i = degree; i > 0; --i) {
        if (a[i] != 0)
            str += std::to_string(a[i]) + ((i > 1) ? "x^" +
std::to_string(i) : "x") + " + ";
    }
    return str + ((a[0]) ? std::to_string(a[0]) : "\b\b ");
}
```

```cpp
Polynomial::Polynomial(double a, double b, double c, double d) : degree(3)
{

   this->a = new double[degree + 1];

   this->a[3] = a;
   this->a[2] = b;
   this->a[1] = c;
   this->a[0] = d;

}

Polynomial::~Polynomial() {
   delete[] a;
}

Polynomial::Polynomial(const Polynomial &other) : degree(other.degree) {

   a = new double[degree + 1];

   for (int i = 0; i <= degree; ++i) {
       a[i] = other.a[i];
   }
}

Polynomial::Polynomial(const double *coefs) : degree(3) {
   a = new double[4];

   for (int i = 0; i < 4; ++i) {
       a[i] = coefs[3 - i];
   }
}

void Polynomial::set_coef(int i, double coef) {
   if (coef >= degree)
       throw std::errc::result_out_of_range;

   a[i] = coef;
}

Polynomial::Polynomial(int degree) : degree(degree) {
   a = new double[degree + 1];

   for (int i = 0; i < degree; ++i) {
       a[i] = 0;
   }
   a[degree] = 1;
}
```

Результати тестування

```
p1: 2.000000x^3 + -7.000000
p2: 1.000000x^3 + 2.000000x^2 + 3.000000x + 4.000000
p3: 1.000000x^3
p4: 3.000000x^3 + 2.000000x^2 + 3.000000x + -3.000000
p5: 1.000000x^6 + 2.000000x^5 + 3.000000x^4 + 4.000000x^3
```