Министр науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет информационных технологий и программирования

Лабораторная работа № 2

Классы. Перегрузка операторов.

Выполнила студентка группы № M3106
Шеин Максим Андреевич
Подпись:

Проверил:
Повышев Владислав Вячеславович

Санкт-Петербург

2020

Текст задания

Спроектировать и реализовать класс для описания сущности многочлен (полином), раздела математики - Алгебра.
Реализовать конструктор(ы), конструктор копирования, деструктор, а также следующие операторы:
1. =
2. ==, !=
3. +, - (унарный и бинарный), +=, -=
4. * , / (на число), *=, /=
5. <<, >>
6. [] (для получения коэффициента i-го члена)

Решение

```
#include<iostream>
#include<fstream>
#include<string.h>
using namespace std;
class Polynom
{
public:
   int N; //степень полинома
   double *kf; //указатель на коэффициент при i-й степени

   Polynom();//конструктор
   Polynom(int N_);

   ~Polynom();//деструктор

   Polynom(const Polynom &);//копирование

   Polynom operator=(const Polynom &);//оператор присваивания

   Polynom operator+=(const Polynom &);//оператор сложения

   Polynom operator- (const Polynom &);//оператор унарного минуса

   Polynom operator-=(const Polynom &);//оператор вычитания

   Polynom operator*=(const Polynom &);//оператор умножения

   Polynom operator/=(const Polynom &);//оператор деления

   int operator[] (const int i);//оператор индексирования

   bool operator== (Polynom&); //оператор сравнения

   friend ostream &operator<< (ostream &s, const Polynom &c); // перегруженный оператор вывода

   friend istream &operator>> (istream &s, Polynom &c); // перегруженный оператор ввода


};
```

```
Polynom::Polynom()                                  //конструктор
{
   kf = NULL;
}

Polynom::Polynom(int N_)
{
   int i;
   N = N_;

   kf = new double[N + 1];
   for (i = 0; i <= N; i++)
   {
      kf[i] = 0;
   }
}

Polynom::~Polynom()                                  //деструктор
{
   delete[]kf;
}

Polynom::Polynom(const Polynom &f)                          //копирование
{
   N = f.N;
   kf = new double[N + 1];
   for (int i = 0; i <= N; i++)
   {
      kf[i] = f.kf[i];
   }
}

Polynom Polynom::operator= (const Polynom &t)                  //оператор присваивания
{
   if(this!=&t)
   {
      delete[] kf;
      N = t.N;
      kf = new double[N + 1];
      for (int i = 0; i <= N; i++)
      {
         kf[i] = t.kf[i];
      }
   }
   return *this;
}

Polynom Polynom::operator+= (const Polynom &t)                  //оператор сложения
{
   int i;
   //A больше B
```

```
    if (N >= t.N)
    {
        Polynom Z=*this;
        for (i = 0; i <= N; i++)
        {
            Z.kf[i] = kf[i] + t.kf[i];
        }
        return Z;
    }
    //A меньше B
    else
    {
        Polynom Z=t;
        for (i = 0; i <= N; i++)
        {
            Z.kf[i] = t.kf[i] + kf[i];
        }
        return Z;
    }
}

Polynom Polynom::operator- (const Polynom &t)                    //оператор унарного минуса
{
    int i;
    Polynom Z = *this;
    for (i = 0; i <= N; i++)
    {
        Z.kf[i] = -Z.kf[i];
    }
    return Z;
}

int Polynom::operator[] (int i)                                  //оператор индексирования
{
    int a = -1;
    if(N >= 0 && i <= N)
    {
        return kf[i];
    }
    else
    {
        return a;
    }
}

Polynom Polynom::operator-=(const Polynom &t)                    //оператор вычитания
{
    int i;
    if (N >= t.N)//A больше B
    {
        Polynom Z = *this;
        for (i = 0; i <= t.N; i++)
```

```
        {
            Z.kf[i] = kf[i] - t.kf[i];
        }
        return Z;
    }
    else//B меньше A
    {
        Polynom Z(t.N);
        for (i = 0; i <= N; i++)
        {
            Z.kf[i] = -t.kf[i] + kf[i];
        }
        for (i = N + 1; i <= t.N; i++)
        {
            Z.kf[i] = -t.kf[i];
        }
        return Z;
    }
}

bool Polynom::operator== (Polynom&t)                    //оператор сравнения
{
    return(t.N==N);
}

Polynom Polynom::operator*=(const Polynom &t)           //оператор умножения
{
    int i, j, s = 0;
    Polynom Y(N + t.N);
    for (i = 0; i <= N; i++)
    {
        for (j = 0; j <= t.N; j++)
        {
            Y.kf[i + j] += kf[i] * t.kf[j];
        }
    }
    return Y;
}

Polynom Polynom::operator/= (const Polynom &o)
{

    bool a = true;

    Polynom R;
    Polynom o1;
    Polynom o2;
    Polynom o3;

    R.N = N - o.N;
    R.kf = new double[R.N + 1];
    memset(R.kf, 0, (R.N + 1)*sizeof(double));
```

```
o1.N = N;
o1.kf = new double[N + 1];
for (int i = N; i >= 0; i--)
{
   o1.kf[i] = kf[i];
}
o2.N = o.N;
o2.kf = new double[o.N + 1];
for (int i = o.N; i >= 0; i--)
{
   o2.kf[i] = o.kf[i];
}
o3.N = o1.N;
o3.kf = new double[o1.N + 1];

double mnoj;
int k = 0;
int i, j;
while (a)
{

   for (int i = o.N; i >= 0; i--)
   {
      o3.kf[i] = o.kf[i];
   }

   if (o2.N < o1.N)
   {
      for (i = o1.N,j = o2.N; i >= 0; i--, j--)
      if (j < 0)
      {
         o3.kf[i] = 0;
      }
      else
      {
         o3.kf[i] = o2.kf[j];
      }
   }
   mnoj = o1.kf[o1.N] / o3[o1.N];

   R.kf[R.N - k] = mnoj;
   k++;

   for (int i = 0; i <= o1.N; i++)
   {
      o3.kf[i] *= mnoj;
   }

   for (int i = 0; i <= o1.N; i++)
   {
      o1.kf[i] -= o3.kf[i];
```

```
        }

        o1.N--;
        if (o2.N > o1.N)
        {
            a = false;
        }

    }
    cout << "Остаток от деления: " << o1 << endl;
    cout << "Результат: ";
    return R;
}
istream &operator>>(istream &s, Polynom &c)// перегруженный оператор ввода
{
    int i;
    for (i = 0; i <= c.N; i++)
    {
        s >> c.kf[i];
    }
    return s;
}

ostream &operator<<(ostream &s, const Polynom &c)// перегруженный оператор вывода
{
    int i, n=0;
    for (i = 0; i <= c.N; i++)
    {
        if (c.kf[i] != 0)
        {
            n++;
        }
    }
    if (n != 0)
    {
        if (c.kf[0] != 0)
        {
            s << c.kf[0];
        }
        for (i = 1; i <= c.N; i++)
        {
            if (c.kf[i] < 0)
            {
                if(c.kf[i]!=-1)
                {
                    s << c.kf[i] << "X^" << i;
                }
                else
                {
                    s << " - " << "X^" << i;
                }
            }
```

```
            else
            {
                if (c.kf[i] != 0)
                {

                    if(c.kf[i] != 1)
                    {
                        s << " + " << c.kf[i] << "X^" << i;
                    }
                    else
                    {
                        s << " + " << "X^" << i;
                    }
                }
            }
        }
        s << '\n';
    }
    else
    {
        s << 0;
    }
    return s;
}
int main()
{
    setlocale(LC_ALL, "");
    int n, m, i;

    cout << "Введите степень полинома A:" << '\n';
    cin >> n;

    cout << "Введите степень полинома B:" << '\n';
    cin >> m;

    Polynom A(n), B(m), R;

    cout << "Введите коэфициенты полинома A:" << '\n';
    cin >> A;

    cout << "Введите коэфициенты полинома B:" << '\n';
    cin >> B;

    cout << "Введите индекс полинома A:" << '\n';
    cin >> i;

    cout << "Многочлен A: " << A;
    cout << '\n';

    cout << "Многочлен B: " << B;
    cout << '\n';
```

```
    cout << "Многочлен R = A+B: " << (A += B);
    cout << '\n';

    cout << "Многочлен R = A-B: " << (A -= B);
    cout << '\n';

    cout << "Многочлен R =  A*B: " << (A *= B);
    cout << '\n';

    cout << "Многочлен R = A/B: " << (A /= B);
    cout << '\n';

    cout << "if(размер A == размер B): " << (A == B) << '\n';
    cout << '\n';

    cout << "Многочлен R = -A: " << (A - B);
    cout << '\n';

    cout << "Многочлен R = A[i]: " << A[i];
    cout << '\n';

    A.~Polynom();

    system("pause");
}
```