Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра ИУ5.
Курс «Программирование на основе классов и шаблонов»

Отчет по лабораторной работе №5
«Полиномы»

Выполнил:		Проверил:
студент группы ИУ5-25Б		преподаватель каф. ИУ5
Петренко Сергей		
Подпись и дата:		Подпись и дата:

г. Москва, 2019 г.

Постановка задачи:

Написать программу ввода и оперирования полиномами, состоящими из термов. Использовать классы Term and Polynomial.

Ввод полинома

Термы полинома могут вводиться в любом порядке.

Во вводимом терме может присутствовать коэффициент -1.

Терм (член полинома одного порядка) может складываться с другим термом (например, допустим ввод **3x^2+(–x^2), -3x^2 + x^2**)

Пробелы при вводе могут появляться где угодно.

класс Term

Целые члены-данные для коэффициента и показателя степени

Три конструктора (можно обойтись одним):

Без параметров для представления $0x^0$

С одним параметром, например 3, для представления $3x^0$

С двумя параметрами, например 3 и 2, для представления $3x^2$

Перегруженный **operator +**, который получает 2 терма как параметры и возвращает терм-результат.

Перегруженную операцию istream>> для ввода терма в виде, определенном выше в разделе «Ввод полинома».

Перегруженную операцию **ostream <<** для печати терма в виде:

$3x^0$ как **3**, **$3x^1$** как **3x**, **$1x^3$** как **x^3**, **$-3x^2$** как **-3x^2**

Дружественный класс **Polynomial**

Опишите и протестируйте этот класс до создания класса **Polynomial**. Представьте ***main()*** для демонстрации работы этого класса независимо от **Polynomial**.

класс Polynomial

Члены-данные **poly** (массив из 6 термов или сортированный список), и целое **degree** (степень)

Три конструктора

Без параметров для представления полинома 0

С одним целым параметром, например 3, для представления полинома 3

С одним параметром-термом, например **Term(3,2)**, для представления полинома **$3x^2$**

Разработка интерфейса класса:

Polynom.h

```
#ifndef LAB5_POLYNOM_H
#define LAB5_POLYNOM_H
#include <vector>
#include <string.h>
#include <stdio.h>
#include <ios>
#include "Term.h"

class Polynom {
public:
  vector<Term> poly;
  bool order = true; //true-возрастание false-убывание или наоборот)

  Polynom() {
  }

  Polynom(Term term) {
     poly.push_back(term);
  }

  Polynom(int c) {
     poly.push_back(Term(c, 0));
  }

  void normalize() {
     for (int i = 0; i < poly.size(); i++) {
        while (!isLastTerm(poly[i])) {// есть ли несколько термов с одной степенью?
           for (int j = 0; j < poly.size(); j++) {
              if ((poly[i].getDegree() == poly[j].getDegree()) &&
                 i != j) { //если равны и не являются одним термом в полиноме
                 poly[i].changeCoefficient(poly[j].getCoefficient()); // сложение первого терма со вторым
                 poly.erase(poly.begin() + j); //удаление второго терма
              }
           }
           i = 0;
        }
     }
     //sort();
  }

  bool isLastTerm(Term t) {
     int res = 0;
     for (int i = 0; i < poly.size(); i++) {
        if (poly[i].getDegree() == t.getDegree()) {
           res++;
        }
     }
     if (res > 1) return false;
     return true;
  }

  void sort() {
     Term o;
     int h;
     for (int j = poly.size() - 1; j > 0; j--) {
        h = 0;
        for (int i = 0; i < j; i++) {
           if (order ? (!(poly[i] > poly[i + 1])) : (poly[i] > poly[i + 1])) {//добавить order
              o = poly[i];
              poly[i] = poly[i + 1];
              poly[i + 1] = o;
              h++;
```

```
            }
        }
        if (h == 0) {
            for (int i = 0; i < poly.size(); i++) {//удаление пустых термов
                if (poly[i].getCoefficient() == 0) {
                    poly.erase(poly.begin() + i);
                }
            }
            break;
        }
    }
}


friend ostream &operator<<(ostream &out, Polynom &p) {
    p.normalize();
    for (int i = 0; i < p.poly.size(); i++) {
        if (i != p.poly.size() - 1) {
            if (p.poly[i + 1].getCoefficient() > 0) {
                out << p.poly[i] << " + ";
            } else if (p.poly[i].getCoefficient() != 0 && p.poly[i].getDegree() != 0) {
                out << p.poly[i] << " ";
            }
        } else out << p.poly[i];
    }
    return out;
}

friend Polynom operator+(Polynom &p1, Polynom &p2) {
    Polynom buf;
    for (int i = 0; i < p1.poly.size(); i++) {
        buf.poly.push_back(p1.poly[i]);
    }
    for (int i = 0; i < p2.poly.size(); i++) {
        buf.poly.push_back(p2.poly[i]);
    }
    buf.normalize();
    return buf;
}

Polynom operator+=(Polynom &p2) {
    Polynom buf;
    buf = p2 + *this;
    buf.normalize();
    *this = buf;
    return *this;
}

Polynom operator*=(Polynom &p2) {
    Polynom buf;
    buf = p2 * *this;
    buf.normalize();
    *this = buf;
    return *this;
}

friend Polynom operator*(Polynom &p1, Polynom &p2) {
    Polynom buf;
    for (int i = 0; i < p1.poly.size(); i++) {
        for (int j = 0; j < p2.poly.size(); j++) {
            buf.poly.push_back(p1.poly[i] * p2.poly[j]);
        }
    }
    buf.normalize();
    return buf;
}
```

```
static void split(std::vector<std::string> &dest, const std::string &str,const char *delim)
{
    char *pTempStr = strdup(str.c_str());
    char *pWord = strtok(pTempStr, delim);
    vector<string> vec;
    char d = delim[0];
    while (pWord != NULL) {
        string str = pWord;
        if (delim[0] == '-')
            dest.push_back(str);
        else {
            str = delim+str;
        dest.push_back(str);

        }
        pWord = strtok(NULL, delim);
    }

    free(pTempStr);
}

friend istream &operator>>(istream &in, Polynom &polynom) {
    char res[200];
    in.getline(res, 200);
    string test = res;
    vector<string> vector1, resVector;
    vector<bool> boolVector;
    cout<<"Вы ввели: " << test << endl;

    vector<int> znaki;
    int res1 = 1;
    int s = 0; //счетчик скобок
    for (int i = 0; i <= test.size(); i++) {
        if(test[i]=='-'){
            res1*=-1;
        }
        else if(test[i]=='x'){
            znaki.push_back(res1);
            res1=1;
        }
        else if(test[i]=='(' || test[i]==')'){
            s=(test[i]=='(')?s+1:s-1;
        }
    }

    if(s==0){//удаляем знаки в скобках
        for(int i=0;i<test.size();i++){
            if(test[i]=='('){
                while(test[i]!=')'){
                    if(test[i]=='+'||test[i]=='-'){
                        test[i]=' ';
                    }
                    i++;
                }
            }
        }
    }
    int d=0,dx=0;
    //Проверяем есть ли коэффициент в скобках, если да, то при выводе
    for(int i=0;i<test.size();i++){
        if(test[i]=='x'&&d>0){
            boolVector.push_back(true);
            d=0;
            dx=0;
        }
```

```
        else if(test[i]=='x'&&d==0){
            boolVector.push_back(false);
        }
        if(test[i]=='('){
            for(int j = i; j<test.size();j++){
                if(isdigit(test[j])){
                    d++;
                    cout<<"d++"<<endl;
                }
                if(test[j]=='x'){
                    dx++;
                }
                if(test[j]==')'){
                    break;
                }

            }
        }

    }

    test.erase(std::remove(test.begin(), test.end(), ')'), test.end());
    test.erase(std::remove(test.begin(), test.end(), '('), test.end());
    test.erase(std::remove(test.begin(), test.end(), ' '), test.end());

    if(s!=0){
        cout<<"Неправильные скобки!"<<s<<endl;
        return in;
    }
    for(int i=0;i<test.size();i++){
        if(test[i]=='+'||test[i]=='-'||test[i]==')'||test[i]=='('){
            if(test[i]=='+'||test[i]=='-'){
                test[i]='|';
            }
            else
            test[i] =' ';
        }
    }
    split(vector1,test,"|");
    for(int i =0;i<vector1.size();i++){
        vector1[i][0]=(znaki[i]==-1)?'-':'+';
        Term a(vector1[i],boolVector[i]);
        polynom.poly.push_back(a);
    }

    return in;
  }




};


#endif //LAB5_POLYNOM_H
```

Term.h

```
#ifndef LAB5_TERM_H
#define LAB5_TERM_H

#include <vector>

using namespace std;

class Term {
public:
  int coefficient;
  int degree;
public:
  int getCoefficient(){ return this->coefficient;}
  int getDegree(){ return this->degree;}

  static void split(std::vector<std::string> &dest, const std::string &str,
                const char *delim)//минус не видит много букв из-за рекурсии
  {
     char *pTempStr = strdup(str.c_str());
     char *pWord = strtok(pTempStr, delim);
     vector<string> vec;
     char d = delim[0];
     while (pWord != NULL) {
        string str = pWord;
        dest.push_back(str);
        pWord = strtok(NULL, delim);
     }

     free(pTempStr);
  }

  void changeCoefficient(int c){//прибавляет c к коэффициенту
     this->coefficient= this->getCoefficient()+c;
  }
  Term(){
  }
  Term(int c,int d){
     this->coefficient=c;
     this->degree=d;
  }
  Term(string str,bool degr){
     vector<string> data;
     cout<<"STR"<<str<<endl;
     bool hasX;
     if(!str.find("x^")&&str.find("x")){
        hasX=false;
     }
     else hasX=true;
     split(data,str,"x^");
     cout<<"test str "<<data[0]<<" "<<data[1]<<" Size "<<data.size()<<endl;
     for(int i=0;i<data.size();i++){
        data[i].erase(std::remove(data[i].begin(), data[i].end(), ' '), data[i].end());
     }
     if(degr) {//Возводим коэффициент в степень тк он был в скобках
        if(data.size()==2) {
           cout<<"test find x^ "<<endl;
           this->degree = stoi(data[1]);
           this->coefficient = stoi(data[0]);
           int res = coefficient;
           for (int i = 0; i < degree - 1; i++)this->coefficient *= res;
        }
```

```
        else if(data.size()==1){
            cout<<"test find x "<<endl;
            this->coefficient = stoi(data[0]);

            if(hasX)
                this->degree = 1;
            else
                this->degree = 0;
        }
        else{
        }
    }
    else{//не возводим
        if(data.size()==2) {
            cout<<"test find x^ "<<endl;
            this->coefficient = stoi(data[0]);
            this->degree = stoi(data[1]);
        }
        else if(data.size()==1){
            cout<<"test find x "<<endl;
            this->coefficient = stoi(data[0]);
            if(hasX)
            this->degree = 1;
            else
                this->degree = 0;
        }
    }
}

Term( const Term& term){
    this->coefficient=term.coefficient;
    this->degree=term.degree;
}
Term& operator = (const Term& t){
    this->coefficient=t.coefficient;
    this->degree=t.degree;
    return *this;
}

 Term operator +(Term& t2){
     if(this->getDegree()==t2.getDegree()){
         Term buf(this->getCoefficient()+t2.getCoefficient(),t2.getDegree());
         return buf;
     }
     else {
         Term buf(0,0);
         return buf;
     }
}
Term operator *(Term& t2){
   Term buf(this->getCoefficient()*t2.getCoefficient(),this->getDegree()+t2.getDegree());
    return buf;
}

friend bool operator >(Term& t1, Term& t2){
    if(t1.getDegree()==t2.getDegree()){
        return bool(t1.getCoefficient()>t2.getCoefficient());
    }
    else{
        return t1.getDegree()>t2.getDegree();
    }
}
```

```
  friend ostream& operator<<(ostream& out, Term& term){
     switch(term.getCoefficient()){
        case 0:
           return out;
        case 1:
           break;
        default:
           if(term.getCoefficient()==-1)out<<"- ";
           else if(term.getCoefficient()<-1)out<<"- "<<term.getCoefficient()*-1;
           else out<<term.getCoefficient();
           break;}
     switch(term.getDegree()){
        case 0:
           out<<"1";
           break;
        case 1:
           out<<"x";
           break;
        default:
           out<<"x^"<<term.getDegree();
           break;
     }return out;
  }

  friend istream& operator>>(istream& in, Term& term) {
     string test;
     string c, d;
     int coef, degr, x_pos=0;
     in >> test;
     x_pos=test.find('x');
     for (int i = 0; i < test.size(); i++) {
        cout<<"\'"<<test[i]<<"\'"<<endl;
        if (test[i] == 'x') {
           x_pos = i;
           for (int j = 0; j < i; j++) {
              cout<<"\'"<<test[j]<<"\'"<<endl;
              c += test[j];
           }
           coef = atoi(c.data());
        }
     }
     if (test[x_pos + 1] == '^') {
        x_pos += 2;
        for (int i = x_pos; i < test.size(); i++) {
           d += test[i];
        }
        degr = atoi(d.data());
        term.coefficient=coef;
        term.degree=degr;

        return in;
     }
  }

};
```

Анализ данных:

```
/Users/sergei/CLionProjects/Lab5Petrenko/cmake-build-debug/Lab5Petrenko
 x^2 + 0  + (2)x^3 - x^4 -2x^4 + 100(x)^8
x^2 + 2x^3 - 3x^4 + 100x^8
```

```
/Users/sergei/CLionProjects/Lab5Petrenko/cmake-build-debug/Lab5Petrenko
0 + x^0 + x^1 + 3x^2-x^2 - 3x^(3) + -(-x^10)
1 + x + 2x^2 - 3x^3 + x^10
```

```
/Users/sergei/CLionProjects/Lab5Petrenko/cmake-build-debug/Lab5Petrenko
1+0 -1 +x^0
1
```

```
/Users/sergei/CLionProjects/Lab5Petrenko/cmake-build-debug/Lab5Petrenko
x^1 + 0
x
```