**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Программирование на основе классов и шаблонов»

Отчет по лабораторной работе №6

«Полиномы»

| Выполнил: |  | Проверил: |
| --- | --- | --- |
| студент группы ИУ5-24б |  | преподаватель каф. ИУ5 |
| Чернев Н.А. |  |  |
| Подпись и дата: 29.05.2024 |  | Подпись и дата: |
|  |  |  |

Москва, 2024 г.

## Задание

## Задание

Написать программу ввода и оперирования полиномами, состоящими из термов. Для этого разработать классы Term and Polynomial, описание которых представлено ниже.

### Ввод полинома

* Термы полинома могут вводиться в любом порядке.
* Во вводимом терме может присутствовать отрицательные коэффициенты, например, -1.
* Терм (член полинома одного порядка) может складываться с другим термом. Например: 3x^2 – x^2, -3x^2 + x^2.
* Пробелы при вводе могут появляться где угодно.

К СВЕДЕНИЮ

Обратите внимание, что по заданию вы должны уметь работать со строками такого вида:

3x^2 - x^2 - 3x^2 + x ^2 + 5x^5 - 4x^3 +x^2 - 7

Пробелы расставлены в случайном порядке.

Результат разбора такой строки - полином. Этот разбор вы можете сделать в перегрузке ввода у полинома. В рамках данной перегрузки строку можно сначала разобрать на термы, потом термы сложить, и получить полином. При сложении вы можете вызывать перегрузку сложения полинома с термом.

### Класс Term

* Целые члены-данные для коэффициента и показателя степени.
* Три конструктора (можно обойтись одним):
  + Без параметров для представления
    - 0x0
  + С одним параметром, например 3, для представления
    - 3x0
  + С двумя параметрами, например 3 и 2, для представления
    - 3x2
* Перегруженный operator+, который получает 2 терма как параметры и возвращает терм-результат.
* Перегруженную операцию istream>> для поддержки ввода полинома в виде, определенном выше в разделе "Ввод полинома".
* Перегруженную операцию ostream<< для печати терма в виде:
  + 3x0 как 3,
  + 3x1 как 3x,
  + 1x3 как x^3,
  + −3x2 как -3x^2.
* Дружественный класс Polynomial.

К СВЕДЕНИЮ

Продемонстрируйте работу с классом Term в функции main() отдельно от Polynomial.

### Класс Polynomial

* Члены-данные poly (динамический массив 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 3x2
* Конструктор копирования и операторы присваивания =, +=, \*=.
* Скрытый член order\_ для указания способа хранения термов:
  + по возрастанию степени;
  + по убыванию степени.
* Друзья класса: operator\*, operator+ , каждый из которых получает 2 полинома как параметры и возвращает полином-результат.
* Перегруженную операцию istream>> для ввода полинома в виде, определенном выше в разделе "Ввод полинома".
* Друг класса ostream<< для печати полинома
* 5x5−4x3+1x2+0x1−7x0 в виде: 5x^5 - 4x^3 +x^2 - 7.

**Разработка алгоритма**

Библиотека math:

* Класс Term:
  + Поля:
    - int multiplier; - множитель
    - int power; - степень
  + Методы:
    - Term() - конструктор без параметров, устанавливает значения 0 и 0 для множителя и степени соответственно
    - Term(int \_multiplier) - конструктор с параметром int, устанавливает значения \_multiplier и 0 для множителя и степени соответственно
    - Term(int \_multiplier, int \_power) - конструктор с параметрами int и int, устанавливает значения \_multiplier и \_power для множителя и степени соответственно
    - Term operator + (const Term &t) const; - перегрузка оператора + для Term + Term
  + Друзья класса:
    - friend std::istream& operator >> (std::istream &in, Term &t); - перегрузка ввода Term
    - friend std::ostream& operator << (std::ostream &out, const Term &t); - перегрузка вывода Term
    - friend std::ostream& operator << (std::ostream &out, const Polynomial &p); - перегрузка вывода Polynomial
    - friend std::istream& operator >> (std::istream &in, Polynomial &p); - перегрузка ввода Polynomial
    - friend Polynomial operator \* (const Polynomial &p1, const Polynomial &p2); - перегрузка оператора \* для Polynomial \* Polynomial
    - friend Polynomial operator + (const Polynomial &p1, const Polynomial &p2); - перегрузка оператора + для Polynomial + Polynomial
    - friend class Polynomial; - дружественный класс Polynomial
* Класс Polynomial:
  + Поля:
    - Term \*poly; - массив одночленов
    - int size; - размер массива
    - int degree; - степень многочлена
    - bool order\_increase = false; - отвечает за вывод по возрастанию(true)/убыванию(false) степени
  + Методы:
    - Polynomial() - конструктор без параметров, заполняет поля нулевыми значениями
    - Polynomial(int multiplier); - конструктор с параметром int, создает массив одночленов и записывает туда multiplier\*x^0, заполняет остальные поля соответствующими значениями
    - Polynomial(const Term &t); - конструктор с параметром Term, создает массив одночленов и записывает туда переданный одночлен, заполняет остальные поля соответствующими значениями
    - Polynomial(const Polynomial &p); - конструктор копирования, создает многочлен со значениями из передаваемого многочлена
    - ~Polynomial() - деструктор, удаляет динамическую память(массив одночленов)
    - Polynomial& operator = (const Polynomial &p); - перегрузка оператора = для Polynomial = Polynomial
    - Polynomial& operator += (const Polynomial &p); - перегрузка оператора += для Polynomial += Polynomial
    - Polynomial& operator \*= (const Polynomial &p); - перегрузка оператора \*= для Polynomial \*= Polynomial
  + Друзья класса:
    - friend Polynomial operator \* (const Polynomial &p1, const Polynomial &p2); - перегрузка оператора \* для Polynomial \* Polynomial
    - friend Polynomial operator + (const Polynomial &p1, const Polynomial &p2); - перегрузка оператора + для Polynomial + Polynomial
    - friend std::istream& operator >> (std::istream &in, Polynomial &p); - перегрузка ввода многочлена
    - friend std::ostream& operator << (std::ostream &out, const Polynomial &p); - перегрузка вывода многочлена

**Текст программы**

**Библиотека math:**

**CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.23)

set(project "math")

project(${project})

set(CMAKE\_CXX\_STANDARD 17)

set(${project}\_SOURCES

Term.cpp

Polynomial.cpp)

set(${project}\_HEADERS

Term.h

Polynomial.h)

set(${project}\_SOURCE\_LIST

${${project}\_SOURCES}

${${project}\_HEADERS})

add\_library(${project}

STATIC

${${project}\_SOURCE\_LIST})

**Term.h**

#pragma once

#include <iostream>

class Polynomial;

class Term{

private:

int multiplier;

int power;

public:

Term() : multiplier(0), power(0) {}

Term(int \_multiplier) : multiplier(\_multiplier), power(0) {}

Term(int \_multiplier, int \_power) : multiplier(\_multiplier), power(\_power) {}

Term operator + (const Term &t) const;

friend std::istream& operator >> (std::istream &in, Term &t);

friend std::ostream& operator << (std::ostream &out, const Term &t);

friend std::ostream& operator << (std::ostream &out, const Polynomial &p);

friend std::istream& operator >> (std::istream &in, Polynomial &p);

friend Polynomial operator \* (const Polynomial &p1, const Polynomial &p2);

friend Polynomial operator + (const Polynomial &p1, const Polynomial &p2);

friend class Polynomial;

};

**Term.cpp**

#include "Term.h"

Term Term::operator + (const Term &t) const{

if (power != t.power) return Term();

return Term(multiplier + t.multiplier, power);

}

std::istream& operator >> (std::istream &in, Term &t) {

char str[1000], cur\_term[100];

bool is\_pos = true, is\_power = false, is\_0\_power = true;

int ind = 0, cur\_mul = 1, cur\_power = 0;

in.getline(str, 1000);

for (int i = 0; i < strlen(str); i++) {

if (str[i] == ' ' && (str[i + 1] == '+' || str[i + 1] == '-')) {

if (is\_0\_power) {

cur\_power = 0;

cur\_term[ind] = '\0';

cur\_mul = atoi(cur\_term);

}

else if (is\_power) {

cur\_term[ind] = '\0';

cur\_power = atoi(cur\_term);

}

else {

cur\_power = 1;

}

if (!is\_pos) cur\_mul \*= -1;

t.multiplier = cur\_mul;

t.power = cur\_power;

return in;

}

else {

if (str[i] == '-') is\_pos = false;

else if (str[i] == '+') is\_pos = true;

else if (str[i] == 'x') {

cur\_term[ind] = '\0';

if (ind == 0) cur\_mul = 1;

else cur\_mul = atoi(cur\_term);

is\_0\_power = false;

}

else if (str[i] == '^') {

ind = 0;

is\_power = true;

}

else if (str[i] >= '0' && str[i] <= '9') cur\_term[ind++] = str[i];

}

}

if (!is\_power) {

cur\_term[ind] = '\0';

if (ind == 0) cur\_mul = 1;

else cur\_mul = atoi(cur\_term);

if (!is\_0\_power) cur\_power = 1;

}

else {

if (is\_0\_power) cur\_power = 0;

else {

cur\_term[ind] = '\0';

if (ind == 0) cur\_power = 1;

else cur\_power = atoi(cur\_term);

}

}

if (!is\_pos) cur\_mul \*= -1;

t.multiplier = cur\_mul;

t.power = cur\_power;

return in;

}

std::ostream& operator << (std::ostream &out, const Term &t) {

if (t.multiplier == 0) return out;

if (t.power == 0) {

out << t.multiplier;

}

else {

if (t.multiplier == 1) out << 'x';

else if (t.multiplier == -1) out << "-x";

else out << t.multiplier << 'x';

if (t.power > 1) out << '^' << t.power;

}

return out;

}

**Polynomial.h**

#include "Term.h"

class Polynomial{

private:

Term \*poly;

int size;

int degree;

bool order\_increase = false;

public:

Polynomial() : poly(nullptr), degree(0), size(0) {}

Polynomial(int multiplier);

Polynomial(const Term &t);

Polynomial(const Polynomial &p);

~Polynomial() { delete[] poly; }

Polynomial& operator = (const Polynomial &p);

Polynomial& operator += (const Polynomial &p);

Polynomial& operator \*= (const Polynomial &p);

friend Polynomial operator \* (const Polynomial &p1, const Polynomial &p2);

friend Polynomial operator + (const Polynomial &p1, const Polynomial &p2);

friend std::istream& operator >> (std::istream &in, Polynomial &p);

friend std::ostream& operator << (std::ostream &out, const Polynomial &p);

};

**Polynomial.cpp**

#include "Polynomial.h"

Polynomial::Polynomial(int multiplier) {

poly = new Term[1];

poly[0].multiplier = multiplier;

poly[0].power = 0;

degree = 0;

size = 1;

}

Polynomial::Polynomial(const Term &t) {

poly = new Term[1];

poly[0] = t;

degree = t.power;

size = 1;

}

Polynomial::Polynomial(const Polynomial &p) {

poly = new Term[p.size];

for (int i = 0; i < p.size; i++) {

poly[i] = p.poly[i];

}

degree = p.degree;

size = p.size;

}

Polynomial& Polynomial::operator = (const Polynomial &p) {

if (size > 0) {

delete[] poly;

}

poly = new Term[p.size];

for (int i = 0; i < p.size; i++) {

poly[i] = p.poly[i];

}

degree = p.degree;

size = p.size;

return \*this;

}

Polynomial& Polynomial::operator += (const Polynomial &p) {

\*this = \*this + p;

return \*this;

}

Polynomial& Polynomial::operator \*= (const Polynomial &p) {

\*this = \*this \* p;

return \*this;

}

Polynomial operator + (const Polynomial &p1, const Polynomial &p2) {

Polynomial tmp;

int k = tmp.order\_increase ? -1 : 1;

tmp.poly = new Term[p1.size + p2.size];

int i1 = 0, i2 = 0, i = 0;

while (i1 < p1.size && i2 < p2.size) {

if (p1.poly[i1].power \* k < p2.poly[i2].power \* k) {

tmp.poly[i++] = p1.poly[i1++];

}

else if (p1.poly[i1].power == p2.poly[i2].power) {

tmp.poly[i++] = p1.poly[i1++] + p2.poly[i2++];

}

else {

tmp.poly[i++] = p2.poly[i2++];

}

}

while (i1 < p1.size) {

tmp.poly[i++] = p1.poly[i1++];

}

while (i2 < p2.size) {

tmp.poly[i++] = p2.poly[i2++];

}

tmp.degree = (p1.degree > p2.degree) ? p1.degree : p2.degree;

tmp.size = i;

return tmp;

}

Polynomial operator \* (const Polynomial &p1, const Polynomial &p2) {

Polynomial tmp;

for (int i = 0; i < p1.size; i++) {

for (int j = 0; j < p2.size; j++) {

tmp += Term(p1.poly[i].multiplier \* p2.poly[j].multiplier, p1.poly[i].power + p2.poly[j].power);

}

}

return tmp;

}

std::istream& operator >> (std::istream &in, Polynomial &p) {

if (p.size > 0) {

delete[] p.poly;

p.size = 0;

}

char str[1000], cur\_term[100];

bool is\_pos = true, is\_power = false, is\_0\_power = true;

int ind = 0, cur\_mul = 1, cur\_power = 0;

in.getline(str, 1000);

for (int i = 0; i < strlen(str); i++) {

if (str[i] == '+' || str[i] == '-') {

if (is\_0\_power) {

cur\_power = 0;

cur\_term[ind] = '\0';

cur\_mul = atoi(cur\_term);

}

else if (is\_power) {

cur\_term[ind] = '\0';

cur\_power = atoi(cur\_term);

}

else {

cur\_power = 1;

}

if (!is\_pos) cur\_mul \*= -1;

Term t(cur\_mul, cur\_power);

p += t;

ind = 0;

is\_pos = (str[i] == '+');

is\_0\_power = true;

is\_power = false;

cur\_mul = 1;

cur\_power = 0;

}

else {

if (str[i] == 'x') {

cur\_term[ind] = '\0';

if (ind == 0) cur\_mul = 1;

else cur\_mul = atoi(cur\_term);

is\_0\_power = false;

}

else if (str[i] == '^') {

ind = 0;

is\_power = true;

}

else if (str[i] >= '0' && str[i] <= '9') cur\_term[ind++] = str[i];

}

}

if (!is\_power) {

cur\_term[ind] = '\0';

if (ind == 0) cur\_mul = 1;

else cur\_mul = atoi(cur\_term);

if (!is\_0\_power) cur\_power = 1;

}

else {

if (is\_0\_power) cur\_power = 0;

else {

cur\_term[ind] = '\0';

if (ind == 0) cur\_power = 1;

else cur\_power = atoi(cur\_term);

}

}

if (!is\_pos) cur\_mul \*= -1;

Term t(cur\_mul, cur\_power);

p += t;

return in;

}

std::ostream& operator << (std::ostream &out, const Polynomial &p) {

int ind = 0;

while (ind < p.size - 1 && p.poly[ind].multiplier == 0) ind++;

out << p.poly[ind];

for (int i = ind + 1; i < p.size; i++) {

if (p.poly[i].multiplier < 0) out << p.poly[i];

else if (p.poly[i].multiplier > 0) out << '+' << p.poly[i];

}

return out;

}

**CMakeLists.txt проекта**

cmake\_minimum\_required(VERSION 3.23)

set(project "lab06")

project(${project})

set(CMAKE\_CXX\_STANDARD 17)

set(${project}\_SOURCES

main.cpp)

add\_subdirectory(math)

set(${project}\_SOURCE\_LIST

${${project}\_SOURCES})

add\_executable(${project}

${${project}\_SOURCE\_LIST})

target\_link\_libraries(${project} math)

**main.cpp**

#include "math/Polynomial.h"

int main() {

Term t1(1, 2), t2(1, 1);

Polynomial p1(t1), p2(t2);

p1 += p2;

std::cout << p1 << '\n';

p1 += Polynomial(Term(2, 2));

p1 += Polynomial(Term(-2, 5));

std::cout << p1 << '\n';

Polynomial p3(Term(1, 1));

p1 \*= p3;

std::cout << p1 << '\n';

Term t(-5, 2);

std::cout << t << '\n';

Polynomial p;

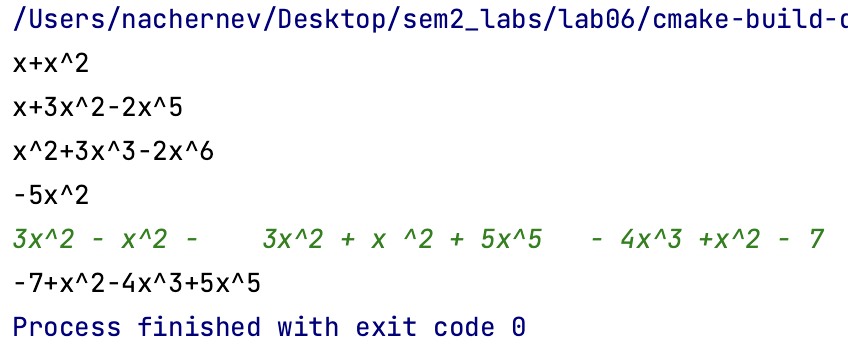
std::cin >> p;

std::cout << p;

return 0;

}

**Анализ результатов**

****