**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 №4

«Полиномы»



|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-21Б |  | преподаватель каф. ИУ5 |
| Михалёв Я.М. |  | Козлов А.Д. |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |



Москва, 2022 г.

Постановка задачи

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

### Ввод

* Термы полинома могут вводиться в любом порядке.
* Коэффициент 1 или -1 может присутствовать или отсутствовать.
* Терм может вычитаться из другого терма (например, допустим ввод **3x^2 - x**)
* Пробелы могут появляться, где угодно.

### класс Term

* Целые члены-данные для коэффициента и показателя степени
* Три конструктора
  + Без параметров для представления 0x0
  + С одним параметром, например 3, для представления 3x1
  + С двумя параметрами, например 3 и 2, для представления 3x2
* Конструктор копирования и оператор присваивания
* Друзья класса: **operator \*, operator +**, каждый из которых получает 2 терма как параметры и возвращает терм-результат.
* Друг класса **ostream <<** для печати терма в виде:  
  **3x0** как **3**, **3x1** как **3x**, **1x3** как **x^3**, **-3x2**  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), и целое **degree**
* 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 хранения термов по возрастанию или убыванию степени
* Друзья класса: **operator \*, operator +**, каждый из которых получает 2 полинома как параметры и возвращает полином-результат.
* Друг класса **ostream <<** для печати полинома в виде:   
  **5x5 - 4x3 + 1x2 + 0x1 - 7x0** как **5x^5 -4x^3 + x^2 -7**

**Представьте** ***main()* для демонстрации работы класса *Polynomial*** и 5 файлов: **Term.h, Term.cpp, Polynomial.h, Polynomial.cpp, mymain.cpp.**

Разработка алгоритма

**Класс Term:**

Члены - данные:

* int x – коэффициент
* int y – степень
* friend class Polynomial

Конструкторы деструктор:

* Конструктор без параметров
* Конструктор с параметрами int x, int y = 1
* Конструктор копирования
* Деструктор

Перегруженные операторы

* operator=
* operator+
* operator-
* operator\*
* operator<<

Методы:

* GetX() – возвращает x
* GetY() – возвращает y
* SetX(int value) – меняет значение x на value
* SetY(int value) – меняет значение y на value
* IsZero() – возвращает x если терм == 0, иначе возвращает false

**Класс Polynomial:**

Члены - данные:

* vector<Term> poly – список одночленов
* degree – степень многочлена

Конструкторы деструктор:

* Конструктор без параметров
* Конструктор с параметрами int n
* Конструктор копирования
* Деструктор

Перегруженные операторы

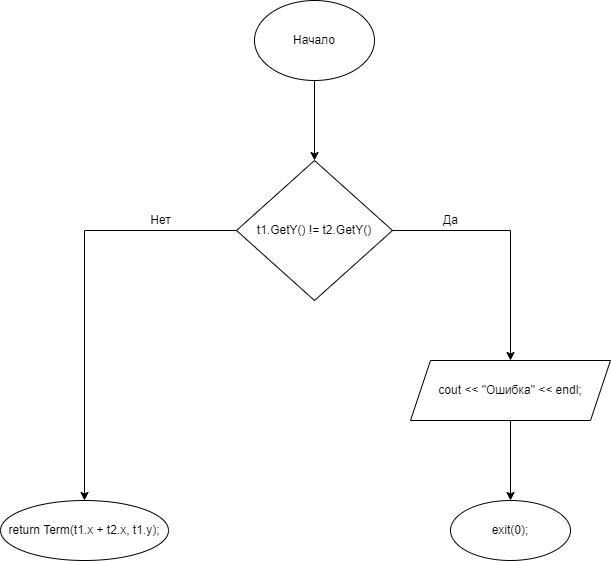
* operator=
* operator+=
* operator-=
* operator+
* operator-
* operator\*
* operator<<

Методы:

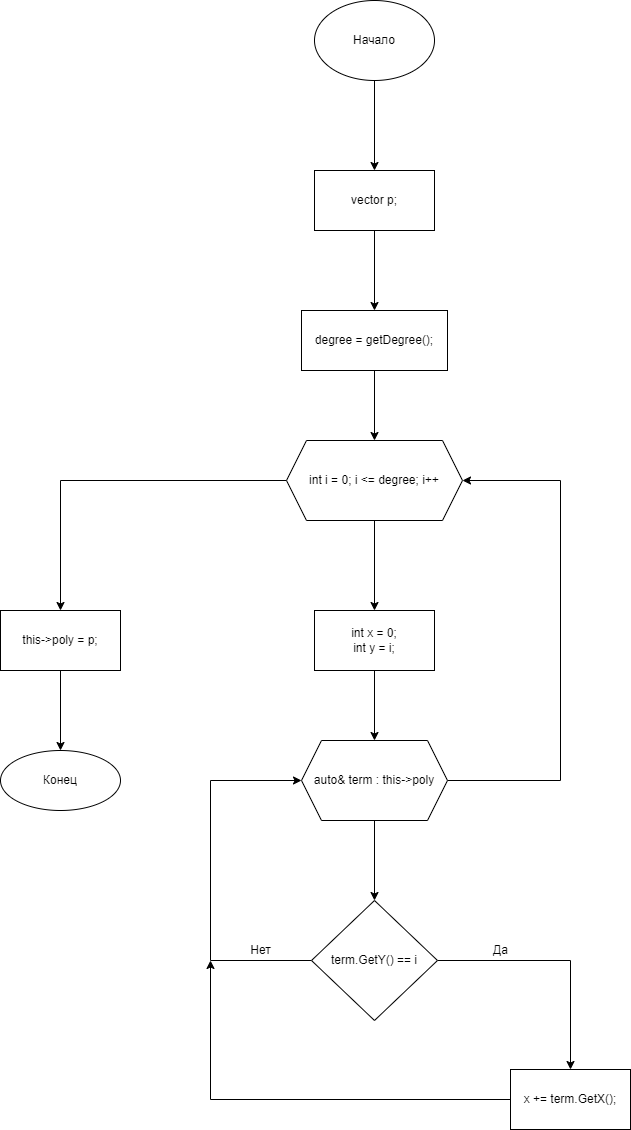
* void SetTerm(Term t, bool subtraction = false) – прибавляет одночлен t, если subtraction = true, то вычитает
* bool HaveTerm(Term t) – возвращает true, если полином содержит терм t, иначе возвращает false
* int getDegree() – возвращает степень полинома
* void Sort() – сортирует полином по возрастанию

**Схема алгоритма**

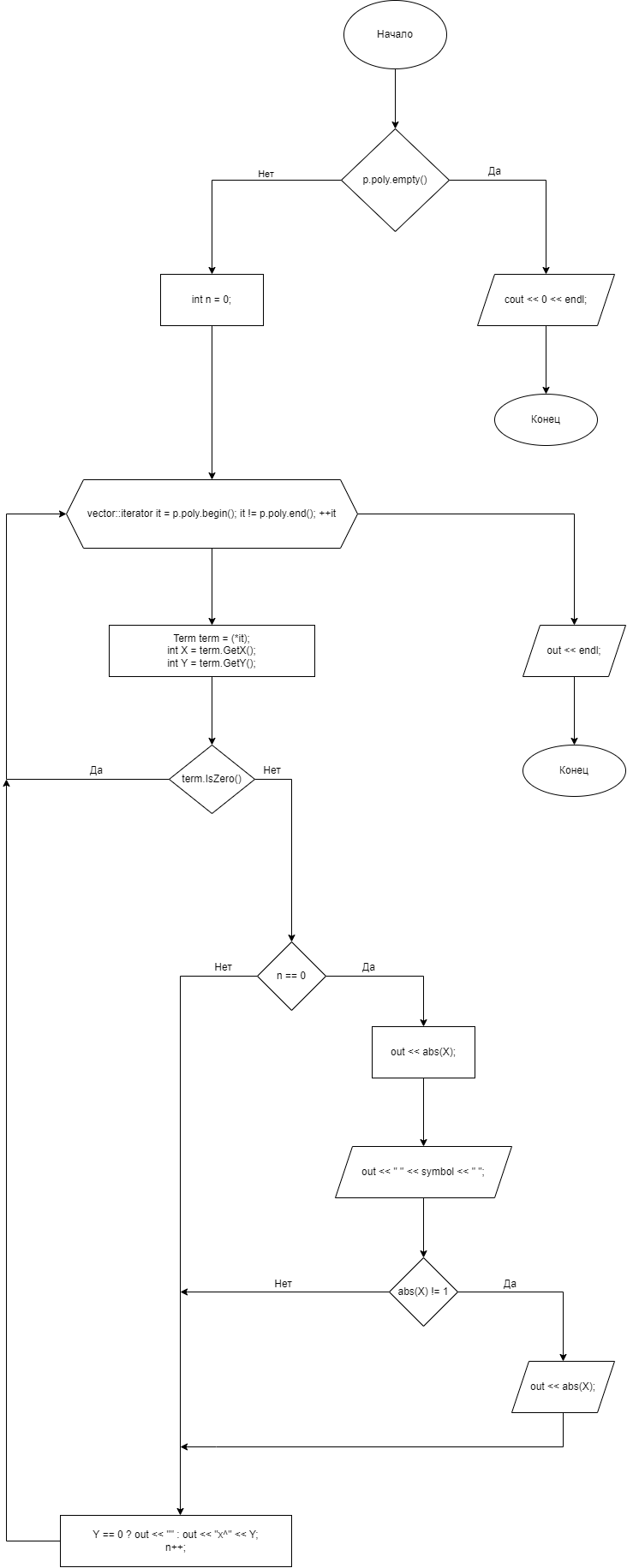
Term operator+

****

Polynomial Sort

****

Polynomial operator<<

****

Текст программы

Main.cpp

#include "Term.h"

#include "Polynomial.h"

**using** **namespace** std**;**

int main**()**

**{**

system**(**"chcp 1251 > nul"**);**

setlocale**(**LC\_ALL**,** "rus"**);**

Polynomial p1**,** p2**;**

p1 **+=** Term**(**1**,** 1**);**

p1 **+=** Term**(**2**,** 0**);**

cout **<<** p1**;**

p2 **+=** Term**(-**1**,** 1**);**

p2 **+=** Term**(**2**,** 0**);**

cout **<<** p2**;**

cout **<<** "\n\n\n\n"**;**

cout **<<** p1 **\*** p2**;**

**}**

Term.h

#include <iostream>

#include <vector>

**using** **namespace** std**;**

class Term

**{**

private**:**

int x**;**

int y**;**

friend class Polynomial**;**

public**:**

Term**()**

**{**

x **=** 0**;**

y **=** 0**;**

**}**

Term**(**int x**,** int y **=** 1**)**

**{**

**this->**x **=** x**;**

**this->**y **=** y**;**

**}**

// Конструктор копирования

Term**(**const Term**&** t**)**

**{**

**this->**x **=** t**.**x**;**

**this->**y **=** t**.**y**;**

**}**

// Перегруженный оператор присваивания

Term **operator=** **(**Term t**)**

**{**

**this->**x **=** t**.**x**;**

**this->**y **=** t**.**y**;**

**return** **\*this;**

**}**

// Друзья класса

friend Term **operator+** **(**Term t1**,** Term t2**)** **{**

**if** **(**t1**.**GetY**()** **!=** t2**.**GetY**())**

**{**

cout **<<** "Ошибка" **<<** endl**;**

exit**(**0**);**

**}**

**return** Term**(**t1**.**x **+** t2**.**x**,** t1**.**y**);**

**};**

friend Term **operator-** **(**Term t1**,** Term t2**)** **{**

**if** **(**t1**.**GetY**()** **!=** t2**.**GetY**())**

**{**

cout **<<** "Ошибка" **<<** endl**;**

exit**(**0**);**

**}**

**return** Term**(**t1**.**x **-** t2**.**x**,** t1**.**y**);**

**};**

friend Term **operator\*** **(**Term t1**,** Term t2**)** **{** **return** Term**(**t1**.**x **\*** t2**.**x**,** t1**.**y **+** t2**.**y**);** **};**

friend Term **operator\*** **(**double x**,** Term t**)** **{** **return** Term**(**x **\*** t**.**x**,** t**.**y**);** **};**

friend Term **operator\*** **(**Term t**,** double x**)** **{** **return** Term**(**x **\*** t**.**x**,** t**.**y**);** **};**

friend void **operator<<** **(**ostream**&** out**,** Term t**)** **{**

**if** **(**t**.**x **==** 0**)**

out **<<** 0 **<<** endl**;**

**else** **if** **(**t**.**y **==** 0**)**

out **<<** t**.**x **<<** endl**;**

**else** **if** **(**t**.**y **==** 1**)**

out **<<** t**.**x **<<** "x" **<<** endl**;**

**else**

out **<<** t**.**x **<<** "x^" **<<** t**.**y **<<** endl**;**

**};**

// Работа с приватными членами класса

int GetX**()** **{** **return** **this->**x**;** **}**

int GetY**()** **{** **return** **this->**y**;** **}**

void SetX**(**int value**)** **{** **this->**x **=** value**;** **}**

void SetY**(**int value**)** **{** **this->**y **=** value**;** **}**

bool IsZero**()** **{** **return** **this->**x **==** 0**;** **}**

**};**

Polynomial.h

#pragma once

class Polynomial

**{**

private**:**

vector**<**Term**>** poly**;**

int degree**;**

public**:**

Polynomial**()**

**{**

poly**.**push\_back**(**Term**(**0**,** 0**));**

degree **=** 0**;**

**}**

Polynomial**(**int n**)**

**{**

poly**.**push\_back**(**Term**(**n**,** 0**));**

degree **=** 0**;**

**}**

Polynomial**(**Term term**)**

**{**

poly**.**push\_back**(**term**);**

degree **=** max**(**degree**,** term**.**GetY**());**

**}**

// Конструктор копирования

Polynomial**(**const Polynomial**&** p**)**

**{**

poly **=** p**.**poly**;**

degree **=** p**.**degree**;**

**}**

void SetTerm**(**Term t**,** bool subtraction **=** **false)**

**{**

**for** **(**auto**&** term **:** poly**)**

**{**

**if** **(**term**.**GetY**()** **==** t**.**GetY**())**

**{**

term**.**SetX**(**subtraction **?** term**.**GetX**()** **-** t**.**GetX**()** **:** term**.**GetX**()** **+** t**.**GetX**());**

**}**

**}**

**}**

bool HaveTerm**(**Term t**)**

**{**

**for** **(**auto**&** term **:** **this->**poly**)**

**{**

**if** **(**term**.**GetY**()** **==** t**.**GetY**())**

**{**

**return** **true;**

**}**

**}**

**return** **false;**

**}**

int getDegree**()**

**{**

int deg **=** 0**;**

**for** **(**auto**&** term **:** **this->**poly**)**

**{**

deg **=** max**(**deg**,** term**.**GetY**());**

**}**

**return** deg**;**

**}**

void Sort**()**

**{**

vector**<**Term**>** p**;**

degree **=** getDegree**();**

**for** **(**int i **=** 0**;** i **<=** degree**;** i**++)**

**{**

int x **=** 0**;**

int y **=** i**;**

**for** **(**auto**&** term **:** **this->**poly**)**

**{**

**if** **(**term**.**GetY**()** **==** i**)**

**{**

x **+=** term**.**GetX**();**

**}**

**}**

p**.**push\_back**(**Term**(**x**,** y**));**

**}**

**this->**poly **=** p**;**

**}**

// Операторы присваивания

void **operator=** **(**Polynomial p**)** **{**

**this->**poly**.**clear**();**

**for** **(**auto**&** term **:** p**.**poly**)**

**{**

**this->**poly**.**push\_back**(**term**);**

**}**

Sort**();**

**};**

void **operator+=** **(**Polynomial p**)** **{**

**for** **(**auto**&** term **:** p**.**poly**)**

**{**

**this->**poly**.**push\_back**(**term**);**

**}**

Sort**();**

**};**

void **operator-=** **(**Polynomial p**)** **{**

**for** **(**auto**&** term **:** p**.**poly**)**

**{**

term**.**SetX**((-**1**)** **\*** term**.**GetX**());**

**this->**poly**.**push\_back**(**term**);**

**}**

Sort**();**

**};**

// Друзья класса

friend Polynomial **operator+** **(**Polynomial p1**,** Polynomial p2**)** **{**

Polynomial p **=** Polynomial**();**

**for** **(**auto**&** term **:** p1**.**poly**)**

**{**

p**.**poly**.**push\_back**(**term**);**

**}**

**for** **(**auto**&** term **:** p2**.**poly**)**

**{**

p**.**poly**.**push\_back**(**term**);**

**}**

p**.**Sort**();**

**return** p**;**

**};**

friend Polynomial **operator-** **(**Polynomial p1**,** Polynomial p2**)** **{**

Polynomial p **=** Polynomial**();**

**for** **(**auto**&** term **:** p1**.**poly**)**

**{**

p**.**poly**.**push\_back**(**term**);**

**}**

**for** **(**auto**&** term **:** p2**.**poly**)**

**{**

term**.**SetX**((-**1**)** **\*** term**.**GetX**());**

p**.**poly**.**push\_back**(**term**);**

**}**

p**.**Sort**();**

**return** p**;**

**};**

friend Polynomial **operator\*** **(**Polynomial p1**,** Polynomial p2**)** **{**

Polynomial p **=** Polynomial**();**

**for** **(**auto**&** term1 **:** p1**.**poly**)**

**{**

**for** **(**auto**&** term2 **:** p2**.**poly**)**

**{**

p**.**poly**.**push\_back**(**term1 **\*** term2**);**

**}**

**}**

p**.**Sort**();**

**return** p**;**

**};**

friend void **operator<<** **(**ostream**&** out**,** Polynomial p**)** **{**

**if** **(**p**.**poly**.**empty**())**

**{**

cout **<<** 0 **<<** endl**;**

**return;**

**}**

int n **=** 0**;**

**for** **(**vector**<**Term**>::**iterator it **=** p**.**poly**.**begin**();** it **!=** p**.**poly**.**end**();** **++**it**)**

**{**

Term term **=** **(\***it**);**

int X **=** term**.**GetX**();**

int Y **=** term**.**GetY**();**

**if** **(**term**.**IsZero**())** **continue;**

**if** **(**n **==** 0**)**

**{**

out **<<** abs**(**X**);**

**}**

**else**

**{**

char symbol **=** X **>** 0 **?** '+' **:** '-'**;**

out **<<** " " **<<** symbol **<<** " "**;**

**if** **(**abs**(**X**)** **!=** 1**)**

**{**

out **<<** abs**(**X**);**

**}**

**}**

Y **==** 0 **?** out **<<** "" **:** out **<<** "x^" **<<** Y**;**

n**++;**

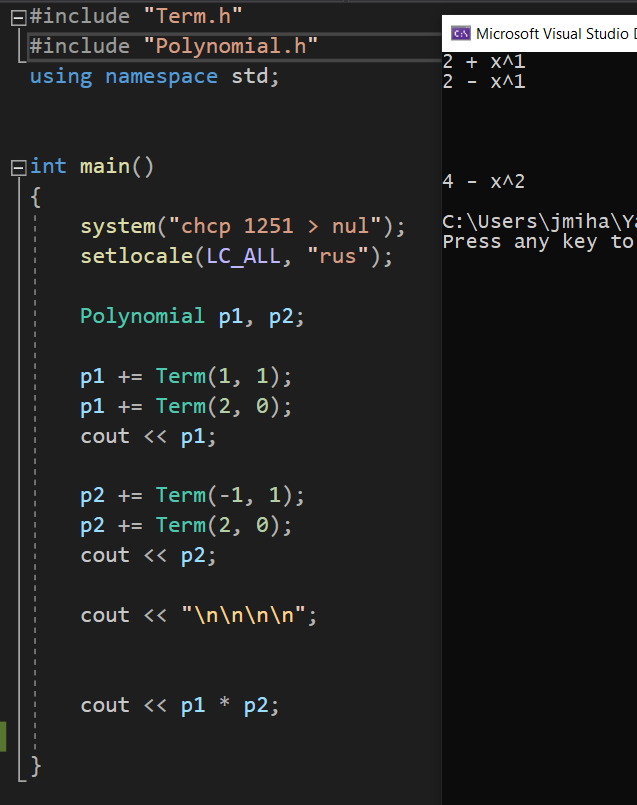
**}**

out **<<** endl**;**

**};**

**};**

Анализ результатов



Вывод

Я научился

* Работать с заголовками
* Перезагружать операторы
* Cортировать массивы пузырьком