Министерство Образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования «Нижегородский Государственный Университет им. Н.И. Лобачевского» «Институт Информационных Технологий Математики и Механики»

**Отчет**

**По лабораторной работе №5**

**«Разработка системы для арифметических действий над многочленами от нескольких переменных»**

Выполнил:

Студент группы 0826-1

Смертин Д. С.

Проверил:

Доцент кафедры МОСТ

Сысоев А. В.

г. Нижний Новгород

2017 г.

Оглавление

[Введение 3](#_Toc483574460)

[Постановка задачи 4](#_Toc483574461)

[Руководство пользователя 5](#_Toc483574462)

[Руководство программиста 6](#_Toc483574463)

[Заключение 8](#_Toc483574464)

[Литература 9](#_Toc483574465)

[Приложение 10](#_Toc483574466)

# Введение

Полиномы как формальный объект хорошо изучены в математике. Математическая модель – алгебра полиномов.

Под многочленом понимается выражение из нескольких термов, соединенных знаками сложения или вычитания.

Терм включает коэффициент и моном, содержащий одну или несколько переменных, каждая из которых может иметь степень.

В число возможных вычислительных процедур над полиномами входят действия по вычислению значений полинома при заданных значений переменных, а также большинство известных арифметических операций.

# Постановка задачи

Цель данной лабораторной работы – разработать программу, в которой будет реализован класс полином. Результатом выполнения программы является сама реализация класса, которая включает в себя реализации методов вычисления, сложения и произведения на число полиномов.

К программе (классу) нужно написать тесты, которые будут подтверждать корректность ее работы.

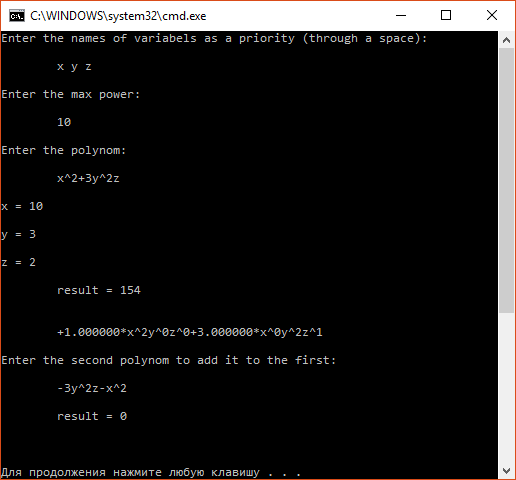
# Руководство пользователя

Чтобы запустить программу, откройте *“Polynominal.exe”*. Запустится консольная программа, в которой последовательно вводятся:

1. Переменные, используемые в выражениях полиномов (через пробел);
2. «Максимальная степень», кратная 10. Если реальная максимальная степень в вашем полиноме меньше 10, то вводится 10, если она больше 9, но меньше 100, то вводится 100, и так далее;
3. Полином;
4. Значения переменных (вещественные числа);
5. Второй полином, чтобы сложить его с первым.

Программа выводит:

1. Вычисленное значение первого полинома;
2. Первый полином в виде строки (для проверки). Если вычисленное значение равно нулю, то строка не выводится.
3. Вычисленное значение полинома, являющегося результатом сложения двух введенных полиномов.
4. Сумма полиномов в виде строки (если его значение не равно нулю).

а

Чтобы программа работала корректно, используйте те же конструкции ввода, что и в примере.

# Руководство программиста

Структура программы

Код программы содержится в трёх проектах: “*Polynominal*”, “*Polynominal\_test”* и *“gtest”.*

“*Polynominal*” содержит заголовочный файл “*Polynominal.h*” и файлы исходного кода: “*Polynominal.cpp*” и “*Source.cpp*”. “*Polynominal\_test”* содержит исходные файлы тестов, которые работают на основе библиотеки “*gtest.lib”*, которая образуется в результате сборки проекта *“gtest”*.

В “*Polynominal.h*” объявлены структуры *“TMonom”* и *“TLink”*, и класс *“TPolynominal”*. В них объявлены (где-то реализованы) их методы и поля.

В “*Polynominal.cpp*” реализованы методы класса *“TPolynominal”* и объявлены статистические поля.

В “*Source.cpp*” написан код, демонстрирующий работу с *“TPolynominal”.*

Алгоритмы

Алгоритм создания класс-типа «полином»

Пусть пользователь ввел строки с объявленными переменными «x y» и полином в виде строки «-3x^2+x^3y^2». Конструктор класса «TPolynominal(“полином”, ”переменные”)» переводит строку с объявленными переменными в массив строк, который хранится в статическом поле класса «arrVar», создается «голова» циклического списка «pHead». Далее он делит полином на мономы и вставляет их в циклический список «TLink» Звено циклического списка хранит значение с типом «TMonom», который, в свою очередь определяется значением общей степени монома и значением его коэффициента, и указатель на следующее звено с мономом или, если все мономы были записаны, на «голову» списка «pHead». Звено «pHead» состоит из монома с значением общей степени «-1» и коэффициента, равного «0». Благодаря этим значениям данное звено легко определить, а значит легко определить начало и конец созданного полинома.

Для вычисления общей степени «degree» монома пользователь вводит «максимальную степень» «pow», которая является статическим полем класса «TPolynominal». Пусть дан моном «x^3z», при объявленных переменных «x y z». Тогда общая степень данного монома равна

,

где n – количество переменных.

Алгоритм сложения двух полиномов

Как было сказано, структура «полином» - это циклический список, состоящий из звеньев-мономов и звена «pHead». Два полинома – это два списка. Для сложения двух полиномов мы проходимся одновременно по звеньям этих списков от «pHead» до «pHead», сравнивая общую степень мономов, хранящихся в этих звеньях. «Прохождение» осуществляется следующим образом: если степени мономов первого и второго полинома равны, то складываются их коэффициенты и получившийся моном с этими степенью и коэффициентом вставляется в новое звено результирующего полинома (при этом переходим на следующее звено обоих полиномов); если степени не равны, то в следующее звено результирующего полинома вставляется моном с большей степенью (при этом осуществляем переход только в полиноме, из которого был выделен моном).

# Заключение

В процессе выполнения лабораторной работы изучены такие структуры данных, как циклический список и полином, выполнена их реализация в виде класса.

В классе «полином» реализованы методы вычисления полиномов, их сложения и произведения на число.

К программе были написаны тесты, подтверждающие корректность её работы.

# Литература

1. Герберт Шилдт. Полный справочник по C++ = C++: The Complete Reference. — 4-е изд. — М.: Вильямс, 2011. — С. 800. — ISBN 978-5-8459-0489-8.
2. Бьёрн Страуструп. Программирование: принципы и практика использования C++, исправленное издание = Programming: Principles and Practice Using C++. — М.: Вильямс, 2011. — С. 1248. — ISBN 978-5-8459-1705-8.
3. Майкл Мейн, Уолтер Савитч. Структуры данных и другие объекты в C++ = Data Structures and Other Objects Using C++. — 2-е изд. — М.: Вильямс, 2002. — 832 с. — ISBN 0-201-70297-5.
4. Курош А. Г. Курс высшей алгебры, 9 изд. — М., 1968.

# Приложение

“*Polynominal.h*”

#pragma once

#include <iostream>

#include <string>

using namespace std;

struct TMonom

{

double coef;

int degree;

};

struct TLink

{

TMonom monom;

TLink \*pNext;

TLink()

{

monom.coef = 0;

monom.degree = -1;

}

};

class TPolynominal

{

TLink \*pHead;

void CreateHead()

{

pHead = new TLink;

pHead->monom.coef = 0;

pHead->monom.degree = -1;

}

public:

static double\* arrArg;

static unsigned int Power;

static unsigned int numVar; // количество переменных

static string\* arrVar; // массив объявленных переменных

TPolynominal(string strPol, string strVar);

TPolynominal();

double\* GetArrArg() { return arrArg; }

void ToArrVar(string strVar); // перевод в массив объявленных переменных

void StreamSetVar();

void SetVar(double\* arr);

TPolynominal operator + (const TPolynominal &polynom);

TPolynominal operator \* (double const c);

string ToString();

double Calculate();

};

“*Polynominal.cpp*”

#include "TPolynominal.h"

#include <cmath>

unsigned int TPolynominal::Power;

unsigned int TPolynominal::numVar;

double\* TPolynominal::arrArg;

string\* TPolynominal::arrVar;

// перевод в массив объявленных переменных

void TPolynominal::ToArrVar(string strVar)

{

string sV = strVar;

numVar = 0;

char space = ' ';

while (sV[0] == space)

{

sV.erase(0, 1);

}

while (sV[sV.size()-1] == space)

{

sV.erase(sV.size()-1, 1);

}

while (sV.find(" ") != string::npos)

{

sV.erase(sV.find(" "), 1);

}

sV += " ";

for (size\_t i = 0; i < sV.size(); i++)

if (sV[i] == ' ')

numVar++;

arrVar = new string[numVar]; // создание массива переменных

for (size\_t i = 0; i < numVar; i++)

{

arrVar[i] = "";

}

for (size\_t i = 0; i < numVar; i++)

{

size\_t pos = sV.find(" ");

string strTemp = sV;

arrVar[i] += strTemp.erase(pos, strTemp.size() - pos);

sV.erase(0, pos + 1);

}

}

void TPolynominal::StreamSetVar()

{

arrArg = new double[numVar];

for (size\_t i = 0; i < numVar; i++)

{

cout << arrVar[i] << " = ";

cin >> arrArg[i];

cout << endl;

}

}

void TPolynominal::SetVar(double \* arr)

{

arrArg = new double[numVar];

for (size\_t i = 0; i < numVar; i++)

{

arrArg[i] = arr[i];

}

}

TPolynominal::TPolynominal()

{

CreateHead();

}

TPolynominal TPolynominal::operator+(const TPolynominal & polynom)

{

TPolynominal result;

TPolynominal polF = \*this;

TPolynominal polS(polynom);

TLink \*pF = polF.pHead;

TLink \*pS = polS.pHead;

TLink \*pR = result.pHead;

pF = pF->pNext;

pS = pS->pNext;

while (!(pF == polF.pHead && pS == polS.pHead))

{

if (pF->monom.degree > pS->monom.degree)

{

TLink \*tmp = new TLink;

tmp = pF;

pR->pNext = tmp;

pR = pR->pNext;

pF = pF->pNext;

continue;

}

if (pF->monom.degree < pS->monom.degree)

{

TLink \*tmp = new TLink;

tmp = pS;

pR->pNext = tmp;

pR = pR->pNext;

pS = pS->pNext;

continue;

}

if (pF->monom.degree == pS->monom.degree)

{

TLink \*tmp = new TLink;

tmp->monom.coef = pF->monom.coef + pS->monom.coef;

tmp->monom.degree = pS->monom.degree;

pS = pS->pNext;

pF = pF->pNext;

pR->pNext = tmp;

pR = pR->pNext;

continue;

}

}

pR->pNext = result.pHead;

return result;

}

//CPolynom CPolynom::operator+(const CPolynom & polynom)

//{

// CPolynom result;

// CPolynom polF = \*this;

// CPolynom polS = polynom;

// TLink \*pF = polF.pHead->pNext;

// TLink \*pS = polS.pHead->pNext;

// TLink \*pR = result.pHead;

// while (pF != pHead &&)

//}

TPolynominal TPolynominal::operator\*(double const c)

{

TLink \*pR = new TLink;

pR = pHead->pNext;

while (pR != pHead)

{

pR->monom.coef \*= c;

pR = pR->pNext;

}

return \*this;

}

TPolynominal::TPolynominal(string strPol, string strVar)

{

string\* arrStrMon; // массив объявленных мономов

size\_t numMonom; // количество мономов

TMonom\* arrMonom; // массив мономов

CreateHead();

// перевод строки с переменными в массив переменных

ToArrVar(strVar);

// перевод строки полинома в массив строк мономов

{

string val = "-+^abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.";

for (size\_t i = 0; i < strPol.size(); i++)

{

if (val.find(strPol[i]) == string::npos)

{

throw "An invalid character was found";

}

}

string str = " " + strPol + " ";

string arop = "-+";

string var = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

if (strPol[0] == '^')

throw "First character is operations";

if (strPol[strPol.length() - 1] == '-' || strPol[strPol.length() - 1] == '+' || strPol[strPol.length() - 1] == '^')

throw "Last character is operations";

}

string sP = strPol;

numMonom = 0;

if (sP[0] != '+' && sP[0] != '-')

{

sP = "+" + sP;

}

for (size\_t i = 0; i < sP.size(); i++)

{

if (sP[i] == '+' || sP[i] == '-')

{

numMonom++;

}

}

arrStrMon = new string[numMonom];

sP += " ";

for (unsigned int i = 0; i < numMonom; i++)

{

size\_t pos;

for (size\_t j = 1; j < sP.size(); j++)

{

if (sP[j] == '+' || sP[j] == '-' || sP[j] == ' ')

{

pos = j;

break;

}

}

string strTemp = sP;

strTemp = strTemp.erase(pos, strTemp.size() - pos - 1);

arrStrMon[i] = strTemp;

sP = sP.erase(0, pos);

}

// добавление ^0 & ^1 к переменным

for (size\_t i = 0; i < numMonom; i++)

for (size\_t j = 0; j < numVar; j++)

{

if (arrStrMon[i].find(arrVar[j]) == string::npos)

{

if (j == 0)

{

size\_t k = 0;

string digits = "0123456789.+-";

while (digits.find(arrStrMon[i][k]) != string::npos)

{

k++;

}

string var = arrVar[j] + "^0";

arrStrMon[i].insert(k, var);

}

else

{

size\_t k = arrStrMon[i].find(arrVar[j - 1]) + arrVar[j - 1].length() + 1;

string digits = "0123456789.+-";

while (digits.find(arrStrMon[i][k]) != string::npos)

{

k++;

}

string var = arrVar[j] + "^0";

arrStrMon[i].insert(k, var);

}

}

if (arrStrMon[i][arrStrMon[i].find(arrVar[j]) + 1] != '^')

{

size\_t pos = arrStrMon[i].find(arrVar[j]) + 1;

arrStrMon[i].insert(pos, "^1");

}

}

// заполнение коэффициентов

arrMonom = new TMonom[numMonom];

for (size\_t i = 0; i < numMonom; i++)

{

string strTemp = arrStrMon[i];

size\_t pos = strTemp.find(arrVar[0]);

strTemp = strTemp.erase(pos, arrStrMon[i].size() - pos);

if (strTemp == "+")

arrMonom[i].coef = 1;

else

if (strTemp == "-")

arrMonom[i].coef = -1;

else

arrMonom[i].coef = atof(strTemp.c\_str());

}

// заполнение степеней

for (size\_t i = 0; i < numMonom; i++)

{

string monomWithOutVar = arrStrMon[i];

monomWithOutVar = arrStrMon[i].erase(0, arrStrMon[i].find(arrVar[0]));

for (size\_t j = 0; j < numVar; j++)

{

size\_t posVar = monomWithOutVar.find(arrVar[j]);

size\_t lengthVarStr = arrVar[j].length();

monomWithOutVar = monomWithOutVar.erase(posVar, lengthVarStr);

}

unsigned int \*arrPow = new unsigned int[numVar];

arrMonom[i].degree = 0;

for (size\_t j = 0; j < numVar; j++)

{

string tmp = monomWithOutVar.erase(0, 1);

size\_t pos;

if (monomWithOutVar.find("^") == string::npos)

{

arrMonom[i].degree += atoi(tmp.c\_str());

}

else

{

pos = monomWithOutVar.find("^");

tmp = tmp.erase(pos, tmp.size() + 1 - pos);

arrPow[j] = atoi(tmp.c\_str());

monomWithOutVar.erase(0, pos);

arrMonom[i].degree += arrPow[j] \* pow(Power, numVar - 1 - j);

}

}

}

// bublesort

for (size\_t i = 0; i < numMonom; i++)

for (size\_t j = numMonom - 1; j > i; j--)

if (arrMonom[j - 1].degree < arrMonom[j].degree)

{

swap(arrMonom[j - 1], arrMonom[j]);

}

TLink \*p = new TLink;

pHead->pNext = p;

p->monom = arrMonom[0];

for (size\_t i = 1; i < numMonom; i++)

{

TLink \*tmp = new TLink;

tmp->monom = arrMonom[i];

p->pNext = tmp;

p = p->pNext;

}

p->pNext = pHead;

}

string TPolynominal::ToString()

{

string result = "";

TLink \*p = new TLink;

p = pHead->pNext;

while (p != pHead)

{

string strCoef = to\_string(p->monom.coef);

if (p->monom.coef > 0)

strCoef = "+" + strCoef;

if (p->monom.coef == 0)

{

p = p->pNext;

continue;

}

result += strCoef + "\*";

unsigned int deg = p->monom.degree;

for (size\_t j = 0; j < numVar; j++)

{

result += arrVar[j] + "^" + to\_string(deg / int(pow(Power, numVar - 1 - j)));

deg = deg % (int)pow(Power, numVar - 1 - j);

}

p = p->pNext;

}

return result;

}

double TPolynominal::Calculate()

{

unsigned int power = Power;

double result = 0;

TLink \*p = new TLink;

p = pHead->pNext;

while (p != pHead)

{

int tmp = p->monom.degree;

double tempMonom = 1;

for (size\_t j = 0; j < numVar; j++)

{

int tmpDelPow = pow(power, numVar - 1 - j);

int tmpPower = (int)( tmp / tmpDelPow );

tempMonom \*= pow(arrArg[j], tmpPower);

tmp = tmp % (int)(pow(power, numVar - 1 - j));

}

result += p->monom.coef\*tempMonom;

p = p->pNext;

}

return result;

}

“*Source.cpp*”

#include "Polynominal.h"

void main()

{

string strPol, \_strPol;

string strVar;

double res;

cout << "Enter the names of variabels as a priority (through a space):\n\n\t";

getline (cin,strVar);

cout << "\n";

cout << "Enter the max power:\n\n\t";

cin >> TPolynominal::Power;

cout << "\n";

cout << "Enter the polynom:\n\n\t";

cin >> strPol;

cout << "\n";

TPolynominal polinom(strPol, strVar);

polinom.StreamSetVar();

res = polinom.Calculate();

cout << "\tresult = " << res << endl;

cout << "\n\n\t" << polinom.ToString() << "\n" << endl;

cout << "Enter the second polynom to add it to the first:\n\n\t";

cin >> \_strPol;

cout << "\n";

TPolynominal \_polinom(\_strPol, strVar);

TPolynominal pol = polinom + \_polinom;

res = pol.Calculate();

cout << "\tresult = " << res << endl;

cout << "\n\n\t" << pol.ToString() << endl;

system("PAUSE");

}