Министерство науки и высшего образования РФ

Федеральное государственное автономное образовательное   
учреждение высшего профессионального образования

«Нижегородский государственный университет им. Н.И. Лобачевского»

Институт информационных технологий математики и механики

**Отчет по учебной практике**

**Лабораторная работа № 5**

**Аналитические преобразования полиномов от нескольких переменных (списки)**

**Выполнил**:студент группы 381703-2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Полегайко В.А.

Подпись

**Научный руководитель**:

доцент каф. МОСТ, к.т.н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сысоев А.В.

Подпись

**Содержание**

[Введение 3](#__RefHeading__3039_1807997502)

[Постановка задачи 4](#__RefHeading__3041_1807997502)

[Руководство пользователя 5](#__RefHeading__3043_1807997502)

[Руководство программиста 6](#__RefHeading__3045_1807997502)

[Описание структуры программы 7](#__RefHeading__3047_1807997502)

[Описание алгоритмов 8](#__RefHeading__3049_1807997502)

[Заключение 9](#__RefHeading__3051_1807997502)

[Литература 10](#__RefHeading__3053_1807997502)

[Приложение 11](#__RefHeading__3055_1807997502)

# **Введение**

Системы аналитических вычислений (компьютерной алгебры) – это новейшее направление развития современной компьютерной математики. Основное их достоинство заключается в возможности выполнения вычислений в аналитическом виде и в возможности проведения арифметических и многих иных вычислений практически с любой желаемой точностью и без ограничений по максимальным (минимальным) значениям чисел. Наряду с привычным вычислительным применением компьютеры широко используются и для аналитической обработки данных. Пример организации обработки полиномов может рассматриваться как введение в проблематику аналитических вычислений на компьютере.

Полином как формальный объект описывается математической структурой — алгеброй полиномов. Под полиномом или многочленом понимается выражение из нескольких термов, соединённых знаками сложения и вычитания. Терм включает коэффициент и моном, содержащий одну или нескольких переменных, каждый из которых может иметь степень.

P(x0,x1,x2,...,xn) = Σ Coeff\* x0k0x1k1x2k2...xnkn

Например:

P=4\*x517x36+x53x67-25

В число возможных вычислительных процедур над полиномами входят действия по вычислению значения полинома при заданных значениях переменных, а также большинство известных математических операций: сложение, вычитание, умножение, вычисление частных производных, интегрирование и т.д.

# **Постановка задачи**

Лабораторная работа направлена на изучение методов компьютерной обработки полиномов. Упрощением проблемы в лабораторной работе применяется следующие ограничения:

* Переменных у полинома только три, которые имеют имя x,y и z соответственно.
* Максимальная степень переменных - <10.

Таким образом рассматриваются полиномы вида:

P(x,y,z) = Σ Coeff\* xaiybizci

Важная проблема, которая решена в работе — введение операции отношения мономов или терм. Вводится взаимнооднозначное соответствие степеней {a,b,c} и числом, которое называется сверткой или индексом и определяется по формуле:

Index=a\*MaxIndex2+b\*MaxIndex+c,

тогда моном с номером i больше монома с номером j, если индекс i-ого больше индекса j-ого.

При наличии трёх переменных в полиноме может потребоваться достаточно много памяти, поэтому предлагается не хранить мономы с нулевым коэффициентом. Для обеспечения правильности выполнения операции арифметики над полиномами необходимо ввести понятие 0-полинома.

Структурой хранения полинома могут быть:

* массив
* линейный список
* циклический список

Массив как структура хранения при отказе от хранения мономов с нулевыми коэффициентами при выполнении арифметических операций может вызвать перепаковку, что является нежелательным для реализации любого проекта. Реализации структуры хранения 0-полинома также определена неоднозначно.

Линейный список — усложняет все операции вставки, удаления и т. д., так как требует контроля наличия списка, контроля вставки в начало или в середину списка, сложно организовать структуру хранения 0-полинома.



Циклический список с введение специального звена в начало лишён недостатков как массива, так и линейного списка.



Ноль-полином при этом выглядит так:



# **Руководство пользователя**

Данная программа написана в среде разработки Microsoft Visual Studio 2017 на языке С++ и является консольным приложением. При запуске приложения, будет предложено инициализировать коэффициент и индекс первого монома , затем запишет его. Тоже самое и для второго .После этого мономы суммируются и получается полином, см. рис.1.

Рис. 1. Преобразования мономов в полином

Далее пользователь вводит другой полином, который преобразуется и выводится на консоль, см. рис.2.

Рис. 2. Преобразование полинома

Если вводится неверный индекс, выдается сообщение об ошибке, см. рис 3.



Рис.3. Ошибка

# **Руководство программиста**

Программа написана при помощи трех классов TMonom, TList, TPolinom.

Класс TMonom содержит следующие приватные поля:

1. int Coeff – Коэффициент перед мономом.
2. int Index – Индекс монома.
3. TMonom \*Link – Указатель на моном.

Класс TMonom содержит следующие методы:

1. TMonom(int \_Coeff=0, int Index = -1) – Конструктор класса.
2. ~TMonom() – Деструктор.
3. TMonom (const TMonom &tmp) – Конструктор копирования.
4. TMonom &operator=(TMonom tmp) – Оператор присваивания.
5. TMonom operator+(TMonom tmp) – Оператор сложения.
6. int GetCoeff() – Возвращает коэффициент монома.
7. int GetIndex() - Возвращает индекс монома.
8. TMonom\* GetLink() - Возвращает указатель монома.
9. Void SetCoeff (int C) – Устанавливает коэффициент монома.
10. Void SetIndex (int In) - Устанавливает индекс монома.
11. Void SetLink (TMonom\* TM) - Устанавливает указатель на моном.
12. String GetMon ( int MaxInd) – Возвращает моном, записанный в строку.

Класс TList содержит приватное поле:

1. TMonom \*PFirst – указатель на первый моном в списке.

Класс TList содержит следующие методы:

1. TList() – Конструктор класса.
2. TList (const TList &tmp) – Конструктор копирования.
3. TList &operator=(const TList &tmp) – Оператор присваивания.
4. ~TList() - Деструктор.
5. void AddMonom(int coeff, int index) – Добавляет моном в список.
6. string GetPolinom(int MaxInd) – возвращает полином из списка, записанный строкой.
7. TList operator+ (TList tmp) – Оператор сложения.

Класс TPolinom содержит следующие приватные поля:

1. int MaxDeg – Максимальная степень переменной.
2. TList CList – Переменная класса TList.
3. void StrToArr(string st, string \*w, int &k) – Записывает строки в массив.
4. int GetCoeff(string monom) – Возвращает коэффициент из строки монома.
5. Int GetIndex(string monom) – Возвращает индекс из строки монома.

Класс TPolinom содержит следующие методы:

1. TPolinom(string st = “”, int \_Max) – Конструктор класса.
2. ~TPolinom(){} – Деструктор.
3. string GetPolinom() – Возвращает строку полинома.
4. TPolinom (const TPolimon &tmp) – Конструктор копирования.
5. TPolinom &operator=(const TPolinom &tmp) – Оператор присваивания
6. TPolinom operator+ (const TPolinom &tmp) – Оператор сложения.

## **Описание структуры программы**

Программа содержит три основных проекта:

1. PolynomList3,
2. test,
3. gtest.

Проект PolynomList3 содержит в себе модули TMonom.h, TList.h, Polinom.h. Модули TMonom.h, TList.h, Polinom.h содержат в себе объявление классов TMonom, TList и TPolinom соответственно. Реализация методов классов содержится в отдельном файле source, который называет PolynomList3.cpp

Проект test содержит набор необходимых тестов Google Test, проверяющих правильность реализации основных классов. Тесты написаны только для всех классов.

Проект gtest содержит необходимую структуру для работы тестов Google Test.

В целом, программа построена на использовании интуитивно понятного пользователю меню.

## **Описание алгоритмов**

**Опишем алгоритмы класса TMonom**

1. TMonom(int \_Coeff = 0, int \_Index=-1) – Конструктор-инициализатор, с начальными значениями коэффициента 0, а индекса - -1. Присваивает значения коэффициента и индекса по умолчанию, указатель ссылается сам на себя.
2. void SetCoeff(int C) – Метод присваивает Coeff значение C .
3. void SetIndex(int In) – Метод присваивает Index значение In.
4. void SetLink (TMonom\* TM) – Метод присваивает Link значение TM.
5. string GetMon (int MaxInd) – Записывает в строковый поток Coeff, затем из индекса получает степени n,m,k перед x,y и z, после чего возвращает строку в виде ”Coeffx^ny^mz^k”.

**Опишем алгоритмы класса TList**

1. TList() - Конструктор класса, присваивает PFirst значение нового монома.
2. void AddMonom(int coeff, int index) – обходим список, если индексы мономов равны, суммируем их коэффициенты, если не равны, то вставляем моном в соответствии с индексом. Для нахождения индекса монома используем GetIndex().
3. string GetPolinom(int MaxInd) – проходим по списку и записываем мономы в полином, а затем возвращаем в строке.

**Опишем алгоритмы класса TPolinom**

1. TPolinom(string st =””, int \_MaxDeg=10) – Конструктор-инициализатор, с начальными значениями строки ”” и максимальной степенью переменной не больше 10. Присваивает значения максимальной степени по умолчанию, добавляет в пустой список мономы.
2. string GetPolinom() - составляет из списка полином, возвращая его строкой.

# **Заключение**

Данная лабораторная работа познакомила нас с алгоритмами создания, обхода и модификации циклического списка на примере операций над полиномами. Эта структура хранения очень эффективна относительно других, использованных нами ранее.

# **Литература**

1. Столлингс, В. Структурная организация и архитектура компьютерных систем, 5-е изд.: Пер. с англ. — М.: Издательский дом «Вильямс», 2002. — 896 с.: ил. — Парал. тит. англ.
2. Брайан Керниган, Деннис Ритчи «Язык программирования Си».
3. Герберт Шилдт - Полный справочник по C
4. Вирт Н. «Алгоритмы и структуры данных».
5. Кормен Т., Лейзерсон Ч., Ривест Р. «Алгоритмы: построение и анализ».

# **Приложение**

**Модуль TMonom.h**

#pragma once

#include <stdlib.h>

#include <sstream>

#include <iostream>

#include "string.h"

using namespace std;

class TMonom

{

int Coeff; // вещественные числа слишком длинные

int Index;

TMonom \*Link;

public:

TMonom(int \_Coeff=0, int \_Index=-1)

{

Coeff = \_Coeff;

Index = \_Index;

Link = this;

};

~TMonom()

{

Link = NULL;

};

TMonom(const TMonom &tmp)

{

Coeff = tmp.Coeff;

Index = tmp.Index;

Link = this;

};

TMonom &operator= (TMonom tmp)

{

Coeff = tmp.Coeff;

Index = tmp.Index;

return \*this;

};

TMonom operator+ (TMonom tmp)

{

TMonom res;

if (Index == tmp.Index)

{

if (Coeff + tmp.Coeff != 0)

{

res.Coeff = Coeff + tmp.Coeff;

res.Index = tmp.Index;

}

}

return res;

};

int GetCoeff()

{

return Coeff;

};

int GetIndex()

{

return Index;

};

TMonom\* GetLink()

{

return Link;

};

void SetCoeff(int C)

{

Coeff = C;

};

void SetIndex(int In)

{

Index = In;

};

void SetLink(TMonom\* TM)

{

Link = TM;

};

string GetMon (int MaxInd)

{

stringstream ss;

ss << Coeff;

int n;

int m;

int k;

k = Index % MaxInd;

m = (Index / MaxInd) % MaxInd;

n = Index / (MaxInd\*MaxInd);

if (n != 0)

{

ss << "x^";

ss << n;

}

if (m != 0)

{

ss << "y^";

ss << m;

;

}

if (k != 0)

{

ss << "z^";

ss << k;

}

return ss.str();

};

};

**Модуль TList.h**

#pragma once

#include "TMonom.h"

#include <sstream>

#include <iostream>

#include "string.h"

using namespace std;

class TList

{

private:

TMonom\* PFirst;

public:

TList()

{

PFirst = new TMonom();

};

TList(const TList &tmp)

{

PFirst = new TMonom();

TMonom\* PCur = PFirst;

TMonom\* tmpCur = tmp.PFirst->GetLink();

while (tmpCur != tmp.PFirst)

{

TMonom\* Copy = new TMonom(\*tmpCur);

PCur->SetLink(Copy);

Copy->SetLink(PFirst);

PCur = Copy;

tmpCur = tmpCur->GetLink();

};

}

TList &operator= (const TList &tmp)

{

TMonom\* PCur= PFirst->GetLink();

while (PCur != PFirst)

{

PFirst->SetLink(PCur->GetLink());

PCur->SetLink(0);

delete PCur;

PCur = PFirst->GetLink();

}

//PFirst = new TMonom();

TMonom\* tmpCur = tmp.PFirst->GetLink();

PCur = PFirst;

while (tmpCur != tmp.PFirst)

{

TMonom\* Copy = new TMonom(\*tmpCur);

PCur->SetLink(Copy);

Copy->SetLink(PFirst);

PCur = Copy;

tmpCur = tmpCur->GetLink();

};

return \*this;

}

~TList()

{

TMonom\* p = PFirst->GetLink();

while (p != PFirst)

{

PFirst->SetLink(p->GetLink());

p->SetLink(0);

delete p;

p = PFirst->GetLink();

}

delete PFirst;

};

void AddMonom(int coeff,int index )

{

TMonom\* tmp= new TMonom(coeff, index);

TMonom\* PPrev = PFirst;

TMonom\* PCur = PFirst->GetLink();

while (PCur->GetIndex() > tmp->GetIndex())

{

PPrev = PCur;

PCur= PCur->GetLink();

}

if (PCur->GetIndex() == tmp->GetIndex())

{

int sumcoeff = PCur->GetCoeff() + tmp->GetCoeff();

if (sumcoeff ==0)

{

PPrev->SetLink(PCur->GetLink());

PCur->SetLink(0);

delete PCur;

}

else

{

PCur->SetCoeff(sumcoeff);

}

}

else

{

tmp->SetLink(PCur);

PPrev->SetLink(tmp);

}

};

string GetPolinom(int MaxInd)

{

string st = "";

TMonom\* PCur = PFirst->GetLink();

while (PCur != PFirst)

{

string stM = PCur->GetMon(MaxInd);

if (stM[0] != '-')

{

st += "+" + stM;

}

else st += stM;

PCur = PCur->GetLink();

}

return st;

}

TList operator+ ( TList tmp)

{

string s1;

TList res(tmp);

TMonom\* PCur = PFirst->GetLink();

while (PCur != PFirst)

{

res.AddMonom(PCur->GetCoeff(),PCur->GetIndex());

PCur = PCur->GetLink();

}

return res;

}

}

**Модуль TPolinom.h**

#pragma once

#include "TList.h"

#include "TMonom.h"

#include <sstream>

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include "string.h"

class TPolimon

{

private:

int MaxDeg;

TList CList;

void StrToArr(string st, string \*w, int &k)

{

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

{

st = "+" + st;

}

k = -1;

for (int i = 0; i < st.length(); i++)

{

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

{

w[++k] = st[i];

}

else

{

w[k] += st[i];

}

}

k++;

}

int GetCoeff(string monom)

{

string coef = "";

for (int i = 0; (i < monom.length()) && ((monom[i] != 'x') && (monom[i] != 'y') && (monom[i] != 'z')); i++)

{

coef += monom[i];

}

int gc = atoi(coef.c\_str());

return gc;

}

int GetIndex(string monom)

{

int l;

char Var[3] = { 'x','y','z' };

int Deg[3] = { 0,0,0 };

for (int i = 0; i < 3; i++)

{

l = monom.find(Var[i]);

if ((l >= 0) && (l <= monom.length()))

{

if (monom[l + 1] == '^')

{

Deg[i] = monom[l + 2] - 48;

}

else

{

Deg[i] = 1;

}

}

}

if ((Deg[0] < MaxDeg) && (Deg[1] < MaxDeg) && (Deg[2] < MaxDeg))

{

return Deg[0] \* MaxDeg\*MaxDeg + Deg[1] \* MaxDeg + Deg[2];

}

else return -1;

}

public:

TPolinom(string st = "", int \_MaxDeg=10)

{

MaxDeg = \_MaxDeg;

string \*Monom;

Monom = new string[st.length() + 1];

int k;

StrToArr(st, Monom, k);

for (int i = 0; i < k; i++)

{

CList.AddMonom(GetCoeff(Monom[i]), GetIndex(Monom[i]));

}

};

~TPolinom(){};

string GetPolinom()

{

string s1 = CList.GetPolinom(MaxDeg);

if(s1=="")

{

s1 = "0";

}

else

{

if (s1[0] == '+')

{

//s1[0] = ' ';

s1.erase(0, 1);

}

}

return s1;

}

TPolinom(const TPolimon &tmp)

{

MaxDeg = tmp.MaxDeg;

CList = tmp.CList;

}

TPolinom &operator=(const TPolinom &tmp)

{

MaxDeg = tmp.MaxDeg;

CList = tmp.CList;

return \*this;

}

TPolinom operator+(const TPolinom &tmp)

{

TPolimon Sum(tmp);

Sum.CList = CList + tmp.CList;

return Sum;

}

};

**Модуль PolynomList3.cpp**

#include "pch.h"

#include <iostream>

#include "TMonom.h"

#include "TList.h"

#include "TPolinom.h"

using namespace std;

int main()

{

/\*

TMonom M1(2,105);

TMonom M2(M1);

TMonom M3;

M3 = M1;

TMonom M4(7, 155);

M3 = M1 + M4;

cout << "M3: \n " << "Coeff =" << M3.GetCoeff() << "Index =" << M3.GetIndex() << "Link =" << M3.GetLink() << endl;

M3 = TMonom(4, 206);

cout << "M3: \n " << "Coeff =" << M3.GetCoeff() << "Index =" << M3.GetIndex() << "Link =" << M3.GetLink() << endl;

M3.SetCoeff(-7);

M3.SetIndex(303);

cout << "M3: \n " << "Coeff =" << M3.GetCoeff() << "Index =" << M3.GetIndex() << "Link =" << M3.GetLink() << endl;

cout << M3.GetMon(10)<<endl;

cout << M1.GetMon(10) << endl;

cout << M4.GetMon(10) << endl;

TList P;

P.AddMonom(M1.GetCoeff(),M1.GetIndex());

P.AddMonom( M3.GetCoeff(), M3.GetIndex());

string s1 = P.GetPolinom(10);

cout <<"P= \n"<< s1<< endl;

TList R(P);

s1 = R.GetPolinom(10);

cout <<"copy R\n"<< s1 << endl;

TList Q=P;

s1 = Q.GetPolinom(10);

cout << "Q=P=\n"<<s1 << endl;

TMonom \*M5 = new TMonom(6, 543);

Q.AddMonom(M5->GetCoeff(),M5->GetIndex());

s1 = Q.GetPolinom(10);

cout << "Q=P+=\n" << s1 << endl;

Q.AddMonom(12, 303);

s1 = Q.GetPolinom(10);

cout <<"Q=\n"<< s1 << endl;

R = P + Q;

s1 = R.GetPolinom(10);

cout << "R=P+Q= " << s1 << endl;

R = Q + P;

s1 = R.GetPolinom(10);

cout << "R=Q+P=\n "<<s1 << endl;

Q.AddMonom(-12, 303);

s1 = Q.GetPolinom(10);

cout << s1 << endl;

TPolimon Pol("2xy-15z^2xy^3+4");

cout << Pol.GetPolinom() << endl;

string P1[5] = {"3x^5y^2z^5-5x^4y^3z^3+7x^3y^5z",

"3x^5y^2z^5-5x^4y^3z^3+7x^3y^5z",

"3x^5y^2z^5-5x^4y^3z^3+7x^3y^5z",

"3x^5y^2z^5-5x^4y^3z^3+7x^7y^5z",

"3x^5y^2z^5-5x^4y^3z^3+7x^7y^5z" };

string Q1[5] = { "4x^3y^2z^6-6x^2yz^8",

"4x^7y^2z^6-6x^6yz^8",

"4x^5y^2z^5+5x^4y^3z^3",

"4x^6y^2z^6-6x^2yz^8",

"-3x^5y^2z^5+5x^4y^3z^3-7x^7y^5z" };

for (int i = 0; i < 5; i++)

{

TPolimon Sum;

TPolimon Left(P1[i]);

TPolimon Right(Q1[i]);

Sum = Left+Right;

cout << endl;

cout << endl;

cout << Sum.GetPolinom() << endl;

}

\*/

TMonom M1;

int Coeff;

int Index;

cout << "Enter coeff(M1):";

cin >> Coeff;

M1.SetCoeff(Coeff);

cout << "Enter index(M1):";

cin >> Index;

if ((Index < 1000) || (Index <0))

{

M1.SetIndex(Index);

cout << "M1: \n " << "Coeff = " << M1.GetCoeff() << " Index =" << M1.GetIndex() << " Link =" << M1.GetLink() << endl;

cout << "M1=" << M1.GetMon(10) << endl;

}

else cout << "Wrong index" << endl;

TMonom M2;

cout << "Enter coeff(M2):";

cin >> Coeff;

M2.SetCoeff(Coeff);

cout << "Enter index(M2):";

cin >> Index;

if ((Index < 1000)|| (Index < 0))

{

M2.SetIndex(Index);

cout << "M2: \n " << "Coeff =" << M2.GetCoeff() << " Index = " << M2.GetIndex() << " Link = " << M2.GetLink() << endl;

cout << "M2=" << M2.GetMon(10) << endl;

}

else cout << "Wrong index" << endl;

TList L;

L.AddMonom(M1.GetCoeff(), M1.GetIndex());

L.AddMonom(M2.GetCoeff(), M2.GetIndex());

cout<<"M1 + M2 = "<< L.GetPolinom(10)<<endl;

string s1;

cout << "Enter polynom:" << endl;

cin >> s1;

TPolinom P(s1,10);

cout << "Transformed polynom:"<<P.GetPolinom() << endl;

system("pause");

}