Федеральное агентство по образованию Российской Федерации

Государственное образовательное учреждение

высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Институт Информационных технологий, Математики и Механики

Отчёт по лабораторной работе

Аналитические преобразования полиномов от нескольких переменных (списки)

Выполнил:

студент ф-та ИТММ гр. 381808-2

Осипов Николай

Проверил:

ассистент каф. МО ЭВМ, ВМК

Панов А

Нижний Новгород

2019 г.

**Содержание**

Введение........................................................................................................................................3

Постановка задачи........................................................................................................................4

Руководство пользователя...........................................................................................................5

Руководство программиста.........................................................................................................6

Описание структур программы..................................................................................................6

Описание структур данных.........................................................................................................6

Описание алгоритмов……...........................................................................................................7

Заключение...................................................................................................................................9 Литература....................................................................................................................................10Приложения.................................................................................................................................11

Исходный код программы..........................................................................................................11

**Введение**

Полином - одна из базовых алгебраических структур, которая встречается в школьной и высшей математике. Изучение полинома - важнейшая тема в курсе алгебры, поскольку с одной стороны многочлены достаточно просты по сравнению с другими типами функций, с другой - широко применяются в решении задач математического анализа.

При решении различных прикладных задач нередко возникают такие объекты как полиномы. При достаточно большом количестве слагаемых выполнение «вручную» простых арифметических операций над ними становиться затруднительным. Поэтому возникает необходимость разработки программы, которая способна выполнять элементарные арифметические операции с полиномами и быстро вычислять результат той или иной операции.

В данной лабораторной работе будет рассматриваться один из возможных способов реализации полинома.

**Постановка задачи**

Требуется разработать программу, выполняющую арифметические операции с полиномами одной переменной (x): сложение, вычитание, умножение двух полиномов и нахождение полинома в заданной точке. Считается, что полином составлен из мономов от одной переменной. Коэффициенты полинома - вещественные числа. Степени полинома – целые числа.

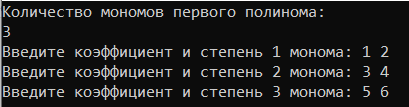
**Руководство пользователя**

Для начала работы программы необходимо запустить на выполнение файл Polynom.cpp

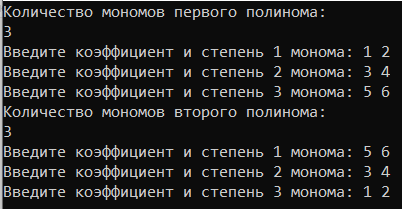
1. В появившемся окне необходимо ввести кол-во мономов первого полинома.



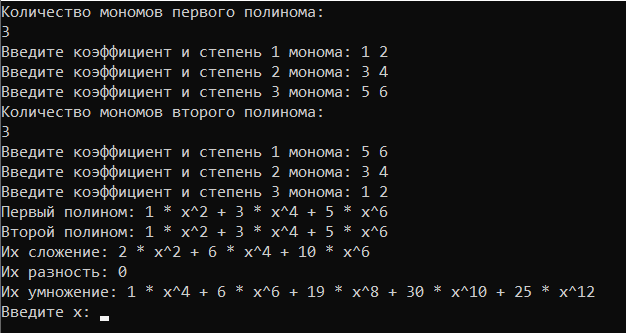
1. Далее для каждого монома вводим, через пробел, значение коэффициента и степени соответственно.



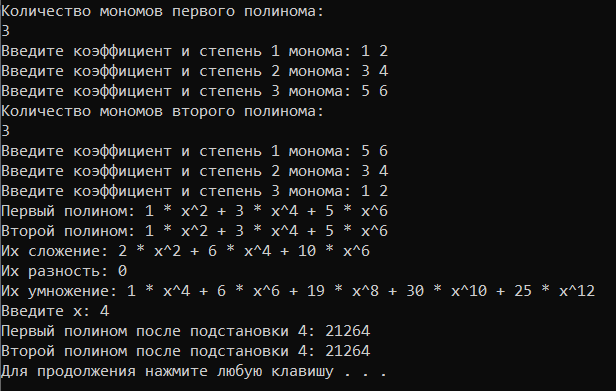
1. Повторяем шаги 1 и 2 для второго полинома.



1. Программа выводит: первый полином, второй полином, сумму полиномов, разность полиномов и результат умножения первого полинома на второй.



1. Программа просит ввести значение переменной Х, вводим его. В итоге программа выводит нам результат вычисления полинома 1 и полинома 2 в заданной точке x.



## **Руководство программиста**

## **Описание структуры программы**

Программа содержит в себе следующие файлы:

1. Polynom.h – заголовочный файл,
2. Polynom.cpp – модуль, реализующий и тестирующий полином;

## **Описание структур данных**

В классе Polynom содержится структура Monom, которая необходима для реализации монома, и сам класс Polynom, представляющий из себя полином. Данный класс позволяет вывести, введенный пользователем полином и выполнить с ним необходимые арифметические операции.

Класс включает в себя следующие поля:

Monom\* head; // заголовок

int size; // длинна списка

Класс позволяет работать со следующими методами и перегрузками:

Polynom();// конструктор

Polynom(const Polynom& polynom); // конструктор копирования

~Polynom();// деструктор

int getSize() { return size; }// к-во звеньев

void clear();// очистить список

void insert(Monom monom); // вставка в список

void insert(double coef, int pow);

void merge(const Polynom& polynom); // добавление полинома к другому

void pop\_front();// удаление элемента из списка

void push\_front(Monom monom); // вставка в начало

void push\_back(Monom monom); // вставка в конец

void del(Monom monom); // удаление элемента из списка

bool find(int pow); // проверка на наличие в списке

int Calculate(int x); // вычисление

Polynom operator+(const Polynom& right) const; // сложение

Polynom operator-(Polynom& right); // вычитание

Polynom operator\*(const Polynom& right); // умножение

Polynom operator\*(const double& val) const;

const Polynom& operator = (const Polynom& right); // присваивание

friend ostream& operator << (ostream& os, const Polynom& object); // вывод

**Описание алгоритмов**

Класс TPolynom:

1. Конструктор Polynom::Polynom()  
   Выполняется инициализация head и size нулями.
2. Конструктор Polynom::Polynom(const Polynom& polynom) : Polynom() копирования  
   Позволяет сделать копию полинома. Происходит поэлементная вставка указателей для коэффициэнтов и степеней и для следующих элементов. Так же копируется длина.
3. Деструктор Polynom::~Polynom()  
   Вызывает метод ClearList()
4. Очистка списка void Polynom::clear()  
   Происходит поэлементная очистка списка с помощью метода pop\_front()
5. Проверка на наличие в списке bool Polynom::find(int pow)  
   Выполняется поэлементная проверка на наличие заданого pow.
6. Вычисление значения int Polynom::Calculate(int x)  
   Позволяет вычислить значение полинома при заданом значении x.
7. Оператор сложения Polynom Polynom::operator + (const Polynom& right) const  
   Позволяет сложить один полином с другим. Происходит сложение мономов в полниномах.
8. Оператор вычитания Polynom Polynom::operator-(Polynom& right)  
   Позволяет вычесть один полином из другого.
9. Оператор умножения Polynom Polynom::operator \* (const Polynom& right)  
   Происходит умножение коэффициэнтов одного полинома на мономы второго полинома.
10. Оператор присваивания const Polynom& Polynom::operator = (const Polynom& right)

Присваивает один полином другому.

1. Вставка в список void Polynom::insert(Monom monom)  
   Вставляет переданый моном в список. Происходит добавление элементов. Длина списка увеличивается на 1.
2. Добавление другого полинома void Polynom::merge(const Polynom& pol)  
   Выполняет слияние двух полиномов. Если один элемент не совпадает, то происходит его добавление в другой полином.
3. Вставка в начало void Polynom::push\_front(Monom monom)  
   Позволяет вставить элемент в начало списка. Увеличивает длину списка на 1.
4. Удаление первого элемента void Polynom::pop\_front()  
   Удаляет первый элемент в списке. Передвигает указатель и очищает память. Уменьшает длину списка на 1.
5. Вставка в конец списка void Polynom::push\_back(Monom monom)  
   Позволяет вставить элемент в конец списка. Находит конец списка и добавляет указатель на новый элемент. Увеличивает длину списка на 1. При отсутствии заголовка - создает новый.
6. Удаление элемента из списка void Polynom::del(Monom monom)  
   Если модуль заголовка меньше малого значения, то происходит удаление первого элемента. Происходит поиск и удаление элемента из списка. Выполняется очистка памяти и уменьшение длины списка.
7. Оператор вывода ostream& operator << (ostream& os, const Polynom& object)  
   Осуществляет форматированный вывод.

**Заключение**

В данной лабораторной работе реализован программный комплекс, способный выполнять элементарные арифметические операции над полиномами одной переменной, причём степень каждой переменной является целым числом.

Достигнутые результаты:

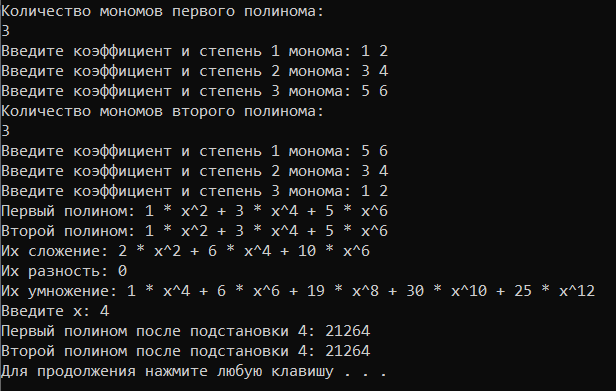
1. Реализация требуемых арифметических операций (сложение, вычитание, умножение полиномов, вычисление полинома в заданной точке);
2. Корректное выполнение арифметических операций;
3. Диалоговое пользовательское приложение.

**Литература**

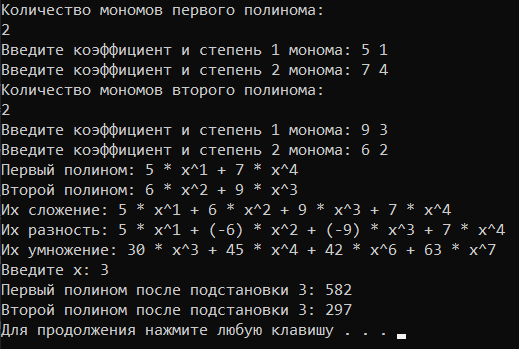
Лабораторный практикум. Составители: Барышева И.В., Мееров И.Б., Сысоев А.В., Шестакова Н.В. Под редакцией Гергеля В.П. Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2017.

**Приложения**

Приложение 1



Приложение 2



## **Исходный код**

Polynom.h

struct Monom

{

int Power;

double Coeff;

Monom\* Next;

Monom()

{

Coeff = 0;

Power = 0;

Next = nullptr;

}

Monom(double coeff, int power);

Monom(double coeff, int power, Monom\* next);

Monom(const Monom& tm);

Monom& operator=(const Monom& tm)

{

Coeff = tm.Coeff;

Power = tm.Power;

return \*this;

}

bool operator==(const Monom& tm);

int operator<(const Monom& tm)

{

return Power < tm.Power;

}

};

class Polynom

{

private:

Monom\* head; // заголовок

int size; // длинна списка

public:

Polynom();// конструктор

Polynom(const Polynom& polynom); // конструктор копирования

~Polynom();// деструктор

int getSize() { return size; }// к-во звеньев

void clear();// очистить список

void insert(Monom monom); // вставка в список

void insert(double coef, int pow);

void merge(const Polynom& polynom); // добавление полинома к другому

void pop\_front();// удаление элемента из списка

void push\_front(Monom monom); // вставка в начало

void push\_back(Monom monom); // вставка в конец

void del(Monom monom); // удаление элемента из списка

bool find(int pow); // проверка на наличие в списке

int Calculate(int x); // вычисление

Polynom operator+(const Polynom& right) const; // сложение

Polynom operator-(Polynom& right); // вычитание

Polynom operator\*(const Polynom& right); // умножение

Polynom operator\*(const double& val) const;

const Polynom& operator = (const Polynom& right); // присваивание

friend ostream& operator << (ostream& os, const Polynom& object); // вывод

};

Polynom.cpp

Monom::Monom(double \_coef, int \_pow)

{

if (\_pow > 999 || \_pow < 0) {

throw exception();

}

Coeff = \_coef;

Power = \_pow;

Next = 0;

}

Monom::Monom(double \_coef, int \_pow, Monom\* \_next)

{

Coeff = \_coef;

Power = \_pow;

Next = \_next;

}

Monom::Monom(const Monom& monom)

{

Coeff = monom.Coeff;

Power = monom.Power;

Next = monom.Next;

}

bool Monom::operator==(const Monom& right)

{

if (Power == right.Power && abs(Coeff - right.Coeff) < eps) return true;

else return false;

}

Polynom::Polynom()

{

size = 0;

head = 0;

}

Polynom::Polynom(const Polynom& polynom) : Polynom()

{

Monom\* tmp = polynom.head;

while (tmp != 0)

{

this->insert(tmp->Coeff, tmp->Power);

tmp = tmp->Next;

}

size = polynom.size;

}

Polynom::~Polynom()

{

clear();

}

inline void Polynom::clear()

{

while (size && head != 0) {

pop\_front();

}

}

void Polynom::merge(const Polynom& pol) // слияние

{

Polynom tmp;

Monom\* curr1 = (this)->head;

Monom\* curr2 = pol.head;

while (curr1 != 0 && curr2 != 0) {

if (curr1->Power <= curr2->Power) {

Monom newMonom(\*curr1);

newMonom.Next = 0;

tmp.push\_back(newMonom);

curr1 = curr1->Next;

}

else {

Monom newMonom(\*curr2);

newMonom.Next = 0;

tmp.push\_back(newMonom);

curr2 = curr2->Next;

}

}

while (curr1 != 0) {

Monom newMonom(\*curr1);

newMonom.Next = 0;

tmp.push\_back(newMonom);

curr1 = curr1->Next;

}

while (curr2 != 0) {

Monom newMonom(\*curr2);

newMonom.Next = 0;

tmp.push\_back(newMonom);

curr2 = curr2->Next;

}

\*this = tmp;

tmp.head = 0;

tmp.size = 0;

}

inline void Polynom::pop\_front()//удаление первого

{

Monom\* tmp = head;

head = head->Next;

delete tmp;

size--;

}

inline void Polynom::push\_front(Monom monom) //вставка в голову очереди

{

head = new Monom(monom.Coeff, monom.Power, head);

size++;

}

void Polynom::push\_back(Monom monom)//вставка в конец очереди

{

if (head == 0) {

head = new Monom(monom);

}

else {

Monom\* curr = head;

while (curr->Next != 0) {

curr = curr->Next;

}

curr->Next = new Monom(monom);

}

size++;

}

void Polynom::insert(Monom monom)

{

if (head == 0 || head->Power > monom.Power) {

push\_front(monom);

return;

}

Monom\* curr = head;

while (curr->Next != 0 && curr->Next->Power <= monom.Power && curr->Power <= monom.Power) {

curr = curr->Next;

}

if (curr->Power == monom.Power) {

curr->Coeff += monom.Coeff;

if (abs(curr->Coeff) < eps) {

del(\*curr);

}

return;

}

Monom\* newObject = new Monom(monom);

newObject->Next = curr->Next;

curr->Next = newObject;

size++;

}

void Polynom::insert(double coef, int pow) // вставить по коэф и степени

{

Monom monom(coef, pow);

if (head == 0 || head->Power > monom.Power) {

push\_front(monom);

return;

}

Monom\* curr = head;

while (curr->Next != 0 && curr->Next->Power <= monom.Power && curr->Power <= monom.Power)

{

curr = curr->Next;

}

if (curr->Power == monom.Power)

{

curr->Coeff += monom.Coeff;

if (abs(curr->Coeff) < eps)

{

del(\*curr);

}

return;

}

Monom\* newObject = new Monom(monom);

newObject->Next = curr->Next;

curr->Next = newObject;

size++;

}

void Polynom::del(Monom monom)

{

Monom\* curr = head;

if (curr->Power == monom.Power && abs(curr->Coeff - monom.Coeff) < eps)

{

pop\_front();

return;

}

while (curr->Next != 0 && curr->Next->Power != monom.Power)

{

curr = curr->Next;

}

Monom\* tmp = curr->Next->Next;

delete curr->Next;

curr->Next = tmp;

size--;

}

bool Polynom::find(int pow) // найти по степени

{

Monom\* curr = head;

while (curr != 0)

{

if (curr->Power == pow)

{

return true;

}

curr = curr->Next;

}

return false;

}

Polynom Polynom::operator\*(const double& val) const //умножить коэф-ы полинома на val

{

Polynom tmp;

Monom\* a = this->head;

while (a != 0)

{

tmp.insert(a->Coeff, a->Power);

a = a->Next;

}

Monom\* temp = tmp.head;

while (temp != 0) {

temp->Coeff = temp->Coeff \* val;

temp = temp->Next;

}

return tmp;

}

const Polynom& Polynom::operator = (const Polynom& right) // присваивание полинома

{

if (this->head != right.head) {

clear();

Monom\* tmp = right.head;

while (tmp != 0) {

insert(tmp->Coeff, tmp->Power);

tmp = tmp->Next;

}

}

return \*this;

}

Polynom Polynom::operator + (const Polynom& right) const //сложение полиномов

{

Polynom curr;

Monom\* thisHead = this->head;

while (thisHead != 0) {

curr.insert(thisHead->Coeff, thisHead->Power);

thisHead = thisHead->Next;

}

Monom\* rightHead = right.head;

while (rightHead != 0) {

curr.insert(rightHead->Coeff, rightHead->Power);

rightHead = rightHead->Next;

}

return curr;

}

Polynom Polynom::operator \* (const Polynom& right)// умножение полиномов

{

Polynom curr;

Monom\* f = head;

Monom\* s = right.head;

while (f != 0) {

s = right.head;

while (s != 0)

{

double coef = s->Coeff \* f->Coeff;

int pow = s->Power + f->Power;

curr.insert(coef, pow);

s = s->Next;

}

f = f->Next;

}

return curr;

}

Polynom Polynom::operator-(Polynom& right)//вычитание полиномов

{

Polynom c = right \* (-1.0);

return \*this + c;

}

int Polynom::Calculate(int x) //вычисление

{

Monom\* tmp = head;

int res = 0;

while (tmp != 0)

{

res += tmp->Coeff \* pow(x, tmp->Power);

tmp = tmp->Next;

}

return res;

}

ostream& operator << (ostream& os, const Polynom& object) //вывод полинома

{

if (object.head == 0)

{

os << 0;

return os;

}

Monom\* temp = object.head;

while (temp != 0)

{

if (temp->Coeff > 0)

{

os << temp->Coeff << " ";

}

else

{

os << "(" << temp->Coeff << ") ";

}

if (temp->Power != 0)

{

os << "\* x^" << temp->Power << " ";

}

if (temp->Next != 0)

{

os << "+ ";

}

temp = temp->Next;

}

return os;

}

void inputPolynom(Polynom& a, int \_number)

{

for (int i = 0; i < \_number; i++) {

cout << "Введите коэффициент и степень " << i + 1 << " монома: ";

double coef;

int ppow;

cin >> coef >> ppow;

a.insert(coef, ppow);

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

try

{

int kM1;

cout << "Количество мономов первого полинома: \n";

cin >> kM1;

Polynom pol1;

inputPolynom(pol1, kM1);

cout << "Количество мономов второго полинома: \n";

int kM2;

cin >> kM2;

Polynom pol2;

inputPolynom(pol2, kM2);

cout << "Первый полином: " << pol1 << "\n";

cout << "Второй полином: " << pol2 << "\n";

cout << "Их сложение: " << pol1 + pol2 << "\n";

cout << "Их разность: " << pol1 - pol2 << "\n";

cout << "Их умножение: " << pol1 \* pol2 << "\n";

cout << "Введите x: ";

int val;

cin >> val;

cout << "Первый полином после подстановки " << val << ": " << pol1.Calculate(val) << "\n";

cout << "Второй полином после подстановки " << val << ": " << pol2.Calculate(val) << "\n";

}

catch (exception str) {

throw exception(str);

}

system("pause");

}