

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»

(ННГУ)

Институт информационных технологий, математики и механики

Направление подготовки «Прикладная математика и информатика»

ОТЧЕТ

по лабораторной работе

## **Арифметические выражения с полиномами**

Выполнил(а): студентка группы 381703-1

Кузнецова А.И.

\_\_\_\_\_ Подпись

Проверил: ассистент кафедры МОСТ ИИТММ

Волокитин В. Д.

\_\_\_\_\_ Подпись

Нижний Новгород

2019

## Оглавление

Введение.....	3
1. Постановка учебно-практической задачи.....	4
2. Руководство пользователя.....	6
3. Руководство программиста.....	7
3.1 Описание структур данных.....	7
3.2 Описание алгоритмов вычисления арифметического выражения.....	8
3.3. Описание структура программного комплекса.....	10
Заключение.....	11
Список литературы.....	12
Приложение.....	13

# Введение

Полиномом (многочленом) называется любая конечная сумма мономов. Мономом называют выражение вида  $sx_1^{i_1}x_2^{i_2}\dots x_n^{i_n}$ . Здесь  $s$  - константа,  $x_1, x_2, \dots, x_n$  - переменные,  $i_1, i_2, \dots, i_n$  - показатели степеней переменных.

Полином, или многочлен - одна из базовых алгебраических структур, которая встречается в школьной и высшей математике. Изучение полинома - важная 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.

В данной лабораторной работе мы будем проводить арифметические операции над полиномами: сложение, вычитание, умножение полиномов, умножение на константу.

В отчете приводится постановка задачи, описание алгоритмов, а также дается описание программы и правил ее использования, прилагается текст программы и результаты выполнения подсчетов.

# 1. Постановка учебно-практической задачи

## *Цель работы:*

Разработать программу, выполняющую арифметические операции с полиномами трех переменных (x, y и z): сложение, вычитание, умножение на константу, умножение двух полиномов. Считается, что полином составлен из мономов от трех переменных с ограничением на степень каждой переменной от 0 до 9 (Опционально можно расширить данное ограничение). Коэффициенты полинома - вещественные числа. Работоспособность программы необходимо проверить с помощью Google Test-ов. Кроме того, необходимо разработать пользовательское консольное приложение.

## *Исходные данные:*

Массив коэффициентов и массив степеней полинома.

## *Требуемый результат:*

1. Полином, полученный в результате сложения;
2. Полином, полученный в результате вычитания;
3. Полином, полученный в результате умножения на константу;
4. Полином, полученный в результате умножения полиномов.

## *Контрольный пример:*

Введем два полинома:  $5xyz + x^2yz^3 + x^3y^4z^2$  и  $x^2yz + 4xyz + xy^2z^3$  (Рис.1.)

```
Input polynom A
Input number of monoms: 3
Input a monom coefficient: 5
Input a degree of x: 1
Input a degree of y: 1
Input a degree of z: 1
Input a monom coefficient: 1
Input a degree of x: 2
Input a degree of y: 1
Input a degree of z: 3
Input a monom coefficient: 1
Input a degree of x: 3
Input a degree of y: 4
Input a degree of z: 2
Input polynom B
Input number of monoms: 3
Input a monom coefficient: 1
Input a degree of x: 2
Input a degree of y: 1
Input a degree of z: 1
Input a monom coefficient: 4
Input a degree of x: 1
Input a degree of y: 1
Input a degree of z: 1
Input a monom coefficient: 1
Input a degree of x: 1
Input a degree of y: 2
Input a degree of z: 3
```

Рис.1.

```

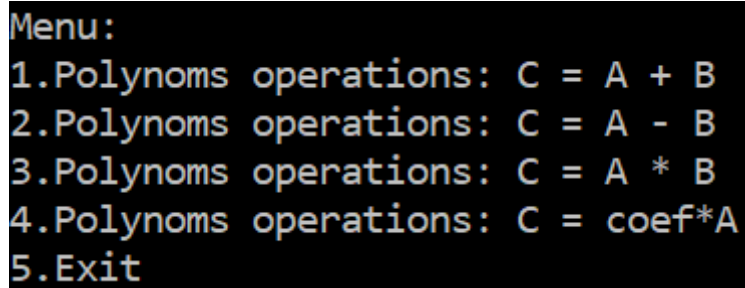
Menu:
1.Polynoms operations: C = A + B
2.Polynoms operations: C = A - B
3.Polynoms operations: C = A * B
4.Polynoms operations: C = coef*A
5.Exit
Input a menu number:
1
Polynom C = A + B
Polynom C = 1x^3*y^4*z^2 + 1x^2*y^1*z^3 + 1x^2*y^1*z^1 + 1x^1*y^2*z^3 + 9x^1*y^1*z^1
Input a menu number:
2
Polynom C = A - B
Polynom C = 1x^3*y^4*z^2 + 1x^2*y^1*z^3 + -1x^2*y^1*z^1 + -1x^1*y^2*z^3 + 1x^1*y^1*z^1
Input a menu number:
3
Polynom C = A * B
Polynom C = 1x^5*y^5*z^3 + 1x^4*y^2*z^4 + 5x^3*y^2*z^2
Input a menu number:
4
Polynom C = coef * A
Enter coef:
2
Polynom C = 2x^3*y^4*z^2 + 2x^2*y^1*z^3 + 10x^1*y^1*z^1
Input a menu number:

```

Рис.2. Результат контрольного примера

## 2. Руководство пользователя

Для запуска программы необходимо открыть файл Polynom.exe. На экране после запуска появится запрос ввода полиномов, а затем открывается главное меню (Рис.3).



```
Menu:
1.Polynoms operations: C = A + B
2.Polynoms operations: C = A - B
3.Polynoms operations: C = A * B
4.Polynoms operations: C = coef*A
5.Exit
```

Рис.3.

Требуется ввести номер пункта меню для проведения одной из арифметических операций (1-4).

Для выхода из программы необходимо выбрать 5 пункт меню.

## 3. Руководство программиста

### 3.1 Описание структур данных

Представим наши объекты в виде двусвязного списка, в котором звенья представлены структурами, содержащими степень в виде трехзначного числа от 0 до 999, где число сотен – это степень при переменной “x”, число десятков - степень при переменной “y”, число единиц - степень при переменной “z”, коэффициенты и указатели. Сам полином реализован с помощью класса Polynom и имеет одно поле — указатель на фиктивную голову.

```
struct Monom
{
    double coef; // вещественный коэффициент
    int deg; // степень
    Monom*prev; // указатель на предыдущий элемент в списке
    Monom*next; // указатель на следующий элемент в списке
};

class Polynom
{
    Monom*fhead; // указатель на фиктивную голову в списке
public:
    Polynom(); //конструктор по умолчанию
    Polynom(struct Monom*, int); //конструктор с параметрами
    Polynom(Polynom&); //конструктор копирования
    void Insert(double, int); //функция вставки
    void Delete(int); // функция удаления
    bool IsEmpty(); //функция проверки пустоты списка
    bool operator==(const Polynom &) const; //оператор сравнения
    Polynom& operator=(const Polynom &); // оператор присваивания
    Polynom operator+(Polynom &); //оператор сложения
    Polynom operator-(Polynom &); //оператор вычитания
    Polynom operator*(double); // оператор, осуществляющий умножение на
константу
    Polynom operator*(Polynom &); // оператор, осуществляющий умножение
полинома на полином
    friend ostream& operator<<(ostream&, Polynom&); //дружественная
функция вывода
    friend istream& operator>>(istream&, Polynom&); //дружественная
функция ввода
    ~Polynom(); //деструктор
}
```

## 3.2 Описание алгоритмов вычисления арифметического выражения

В программах комплекса реализованы алгоритмы и перегрузки арифметических операций, а также операторов сравнения и присваивания и тестирование функций с помощью Google test-ов.

Приведем общее описание методов их выполнения.

*Функция вставки*

`void Insert (double _coef, int _deg)` — функция принимает на вход коэффициент и степень в виде трехзначного числа, находит нужное место для монома и вставляет его. Учитываются все случаи: когда список пуст, существует такой моном(в этом случае коэффициенты складываются), коэффициент перед мономом равен нулю(в этом случае выход из функции).

*Функция удаления*

`void Delete (int _deg)` — функция последовательно ищет моном, удовлетворяющий запросу и удаляет его.

*Функция проверки на пустоту списка*

`bool IsEmpty()` — если список пуст, возвращает true, иначе false.

*Перегрузка ==*

`bool operator==(const Polynom &A) const` — возвращает true, если последовательное сравнение полиномов не вызвало несостыковок, иначе false.

*Перегрузка =*

`Polynom& operator=(const Polynom &A)` — сравнивает головы двух списков. Если они не равны, левый операнд удаляется, создается новый объект и в него вставляются мономы правого.

*Перегрузка +*

`Polynom operator+(Polynom &A)` — последовательное слияние двух списков в третий.

*Перегрузка \* на константу*

`Polynom operator*(double a)` — коэффициент каждого монома умножается на константу.

*Перегрузка -*

`Polynom operator-(Polynom &A)` — реализовано на основе умножения на константу -1 и сложения.

*Перегрузка \* полиномов*



*Polynom operator*\*(*Polynom* &A) — последовательно умножает каждый моном первого полинома на мономы второго: коэффициенты перемножаются, степени складываются(число, большее 999 быть не может).

### 3.3. Описание структура программного комплекса

Программный комплекс представлен программой, состоящей из четырех файлов:

1. В файле `arithmetic.h` реализованы методы класса `Polinom`, функции и перегрузки операторов.
2. В файле `main_arithmetic.cpp` находится основная программа.
3. Файл `test_arithmetic.cpp` содержит тесты для класса `Polinom` и функций.
4. Файл `test_main.cpp` запускает все google tests.

# Заключение

В лабораторной работе был разработан программный комплекс, включающий в себя класс `Polinom`, выполняющий арифметические операции над полиномами: сложение, вычитание, умножение на константу, умножение на другой полином. Программный комплекс позволяет в режиме диалога вводить полиномы. В качестве структуры хранения полинома использовался двусвязный список мономов с ненулевыми коэффициентами. Методы классов и функции тестировались с помощью Google test-ов. Приведенные эксперименты доказали работоспособность данного программного комплекса.

# Список литературы

1. Т. Кормен, Ч. Лейзерсон, Р. Ривест Алгоритмы: построение и анализ. М.:МЦНМО, 1999.- 960 с., 263 ил.
2. Шилдт, Г. С++ для начинающих: самоучитель / Шилдт, Г. - Изд-во: Эком, 2013г.

# Приложение

```
class Polynom
{
    Monom*fhead;
public:
    Polynom()
    {
        Monom*tmp = new Monom;
        tmp->coef = 0.0;
        tmp->deg = -1;
        tmp->next = nullptr;
        tmp->prev = nullptr;
        fhead = tmp;
    };

    Polynom(struct Monom*a, int N)
    {
        Monom*tmp = new Monom;
        tmp->coef = 0.0;
        tmp->deg = -1;
        tmp->next = nullptr;
        tmp->prev = nullptr;
        fhead = tmp;

        for (int i = 0; i < N; i++)
        {
            if ((a[i].deg < 0) || (a[i].deg > 999))
                throw "wrong degree";
            else Insert(a[i].coef, a[i].deg);
        }
    };

    Polynom(Polynom&A)
    {
        Monom*tmp1 = new Monom;
        tmp1->coef = 0.0;
        tmp1->deg = -1;
        tmp1->next = nullptr;
        tmp1->prev = nullptr;
        fhead = tmp1;

        Monom*tmp = A.fhead->next;
        while (tmp != nullptr)
        {
            Insert(tmp->coef, tmp->deg);
            tmp = tmp->next;
        }
    };

    void Insert(double _coef, int _deg)
    {
        if (_coef == 0.0)
            return;
    }
}
```

```

if ((_deg < 0) || (_deg > 999))
    throw "wrong degree value";
if (fhead->next == nullptr)
{
    Monom*tmp = new Monom;
    tmp->coef = _coef;
    tmp->deg = _deg;
    tmp->prev = fhead;
    tmp->next = nullptr;
    fhead->next = tmp;
}
else
{
    Monom*tmp = fhead;
    while ((tmp->next != nullptr) && (tmp->next->deg > _deg))
        tmp = tmp->next;
    if (tmp->next == nullptr)
    {
        Monom*temp = new Monom;
        temp->coef = _coef;
        temp->deg = _deg;
        temp->prev = tmp;
        temp->next = nullptr;
        tmp->next = temp;
    }
    else
    {
        if (tmp->next->deg != _deg)
        {
            Monom*temp = new Monom;
            temp->deg = _deg;
            temp->coef = _coef;
            temp->next = tmp->next;
            temp->next->prev = temp;
            tmp->next = temp;
            temp->prev = tmp;
        }
        else
        {
            tmp->next->coef += _coef;
            if (tmp->next->coef == 0.0)
                Delete(_deg);
        }
    }
}
};

```

```

void Delete(int _deg)
{
    if ((_deg < 0) || (_deg > 999))
        throw "wrong degree value";
    Monom*tmp = fhead->next;
    if (tmp == nullptr)
        return;
    else
    {
        while ((tmp->next != nullptr) && ((tmp->deg) != _deg))
            tmp = tmp->next;
        if ((tmp->deg) == _deg)
        {

```

```

        if (tmp->next != nullptr) {
            tmp->prev->next = tmp->next;
            tmp->next->prev = tmp->prev;
            delete tmp;
        }
        else
        {
            tmp->prev->next = nullptr;
            delete tmp;
        }
    }

};

```

```

bool IsEmpty()
{
    if (fhead->next == nullptr)
        return true;
    else return false;
};

```

```

bool operator==(const Polynom &A) const
{
    Monom*tmp = fhead->next;
    Monom*temp = A.fhead->next;
    while ((tmp != nullptr) && (temp != nullptr))
    {
        if (((tmp->coef) != (temp->coef)) || ((tmp->deg) != (temp->deg)))
            return false;
        tmp = tmp->next;
        temp = temp->next;
        if (((tmp != nullptr) && (temp == nullptr)) || ((tmp == nullptr) &&
(temp != nullptr)))
            return false;
    }
    return true;
};

```

```

Polynom& operator=(const Polynom &A)
{
    if (fhead != A.fhead)
    {
        while (fhead->next != nullptr)
            Delete(fhead->next->deg);
        Monom*tmp = A.fhead->next;
        while (tmp != nullptr)
        {
            Insert(tmp->coef, tmp->deg);
            tmp = tmp->next;
        }
    }
    return *this;
};

```

```

Polynom operator+(Polynom &A)
{
    Polynom C;
    Monom*tmp = fhead->next;
    Monom*temp = A.fhead->next;
    while ((tmp != nullptr) && (temp != nullptr))
    {
        if (tmp->deg < temp->deg)
        {
            C.Insert(tmp->coef, tmp->deg);
            tmp = tmp->next;
        }
        else
        {
            C.Insert(tmp->coef, tmp->deg);
            tmp = tmp->next;
        }
    }
    while (tmp != nullptr)
    {
        C.Insert(tmp->coef, tmp->deg);
        tmp = tmp->next;
    }
    while (temp != nullptr)
    {
        C.Insert(temp->coef, temp->deg);
        temp = temp->next;
    }
    return C;
};

```

```

Polynom operator-(Polynom &A)
{
    Polynom C;
    C = A * (-1);
    C = C + *this;
    return C;
};

```

```

Polynom operator*(double a)
{
    Polynom C(*this);
    Monom*tmp = C.fhead->next;
    while (tmp != nullptr)
    {
        tmp->coef = a * tmp->coef;
        tmp = tmp->next;
    }
    return C;
};

```

```

Polynom operator*(Polynom &A)

```



```

{
    Polynom C;
    int d, d1, d2, d3;
    Monom*tmp = fhead->next;
    Monom*temp = A.fhead->next;
    while (temp != nullptr)
    {
        d1 = ((int)temp->deg) / 100 + ((int)temp->deg) / 10 + ((int)temp->deg)
% 100;

        while (tmp != nullptr)
        {
            d2 = ((int)tmp->deg) / 100 + ((int)tmp->deg) / 10 + ((int)tmp->deg)
>deg) % 100;

            d = d1 + d2;
            d3 = ((int)(temp->deg + tmp->deg)) / 100 + ((int)(temp->deg +
tmp->deg)) / 10 + ((int)(temp->deg + tmp->deg)) % 100;
            if (d != d3)
                throw "degrees > 9";
            else
                C.Insert(temp->coef * tmp->coef, temp->deg + tmp->deg);
            tmp = tmp->next;
        }
        temp = temp->next;
    }
    return C;
};

friend ostream& operator<<(ostream& out, Polynom& A)
{
    int l = 0;
    Monom*temp = A.fhead->next;
    while (temp != nullptr) {
        if (temp->coef != 0)
            out << temp->coef;

        if ((int)temp->deg / 100 != 0)
            out << "x^" << temp->deg / 100;

        if ((int)temp->deg % 100 / 10 != 0)
            out << "y^" << temp->deg % 100 / 10;

        if (temp->deg % 10 != 0)
            out << "z^" << temp->deg % 10;

        temp = temp->next;
        if (temp!=nullptr)
            out << " + ";
        l = 1;
    }

    if (l == 0) out << "0";
    return out;
};

friend istream& operator>>(istream& in, Polynom& A)
{
    int k;
    cout << "Input number of monoms: ";
    in >> k;
    Monom* poly;

```

```

poly = new Monom[k];
for (int i = 0; i < k; i++) {
    int x = -1, y = -1, z = -1;
    cout << "Input a monom coefficient: ";
    in >> poly[i].coef;
    while ((x < 0) || (x > 9)) {
        cout << "Input a degree of x: ";
        in >> x;
    }

    while ((y < 0) || (y > 9)) {
        cout << "Input a degree of y: ";
        in >> y;
    }

    while ((z < 0) || (z > 9)) {
        cout << "Input a degree of z: ";
        in >> z;
    }

    poly[i].deg = x * 100 + y * 10 + z;
}

Polynom B(poly, k);
A = B;
return in;
};

~Polynom()
{
    while (fhead->next != nullptr)
    {
        Delete(fhead->next->deg);
    }
    delete fhead;
};

};

```