

**Московский государственный технический  
университет им. Н.Э. Баумана.**

**Факультет «Информатика и системы управления»**

**Кафедра ИУ5. Курс «Программирование на основе классов и шаблонов»**

**Отчет по лабораторной работе № 5**

**«Полиномы»**

Выполнил:

студент группы ИУ5-23

Терентьев Владислав

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Козлов А. Д.

Подпись и дата:

Москва, 2019 г.

## 1. Постановка задачи

Разработать два класса **Term** (одночлен) и **Polynomial** (многочлен). Реализовать все основные арифметические операции для этих классов и перегрузить соответствующие операторы. Класс **Polynomial** должен использовать класс **Term**.

## 2. Разработка интерфейса класса

В программе описан класс **Term** с переменными-членами (спецификатор доступа `private`): meas типа `int` для хранения степени одночлена; coef типа `int` для хранения коэффициента одночлена. Класс содержит (спецификатор доступа `public`): default конструктор; getters; метод read типа `void` и параметрами типа `string&`, `int&` для чтения одночлена из консоли (параметры нужны для использования в классе **Polynomial**); перегруженные операторы `+`, `*`, `-`, `-` (унарный), `<<`.

Класс **Polynomial** содержит переменные-члены (спецификатор доступа `private`): poly указатель на массив типа **Term**, где хранятся термы многочлена; degree типа `int` для хранения степени многочлена. Так же содержит (спецификатор доступа `public`): конструкторы (default, копирующий, для одного терма); деструктор для удаления данных и освобождения памяти; getter для poly; перегруженные операторы `+`, `*`, `=`, `+=`, `*=`, `-`, `-` (унарный), `>>`, `<<`.

Схема алгоритма оператора вывода для класса Term:

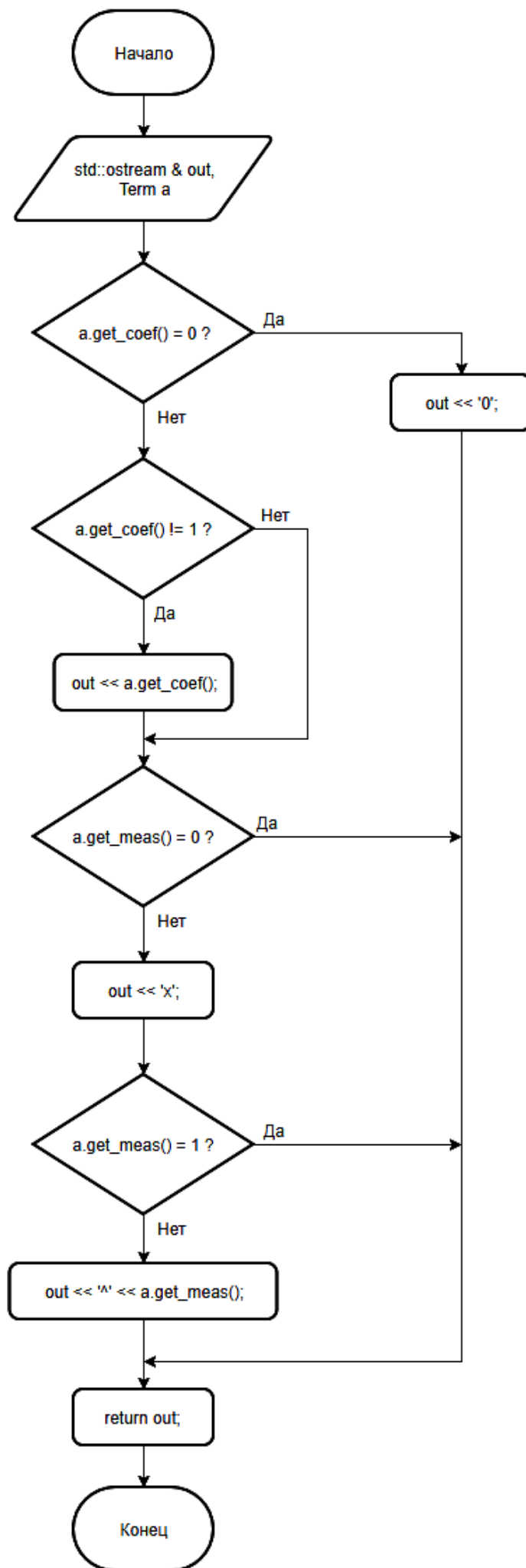
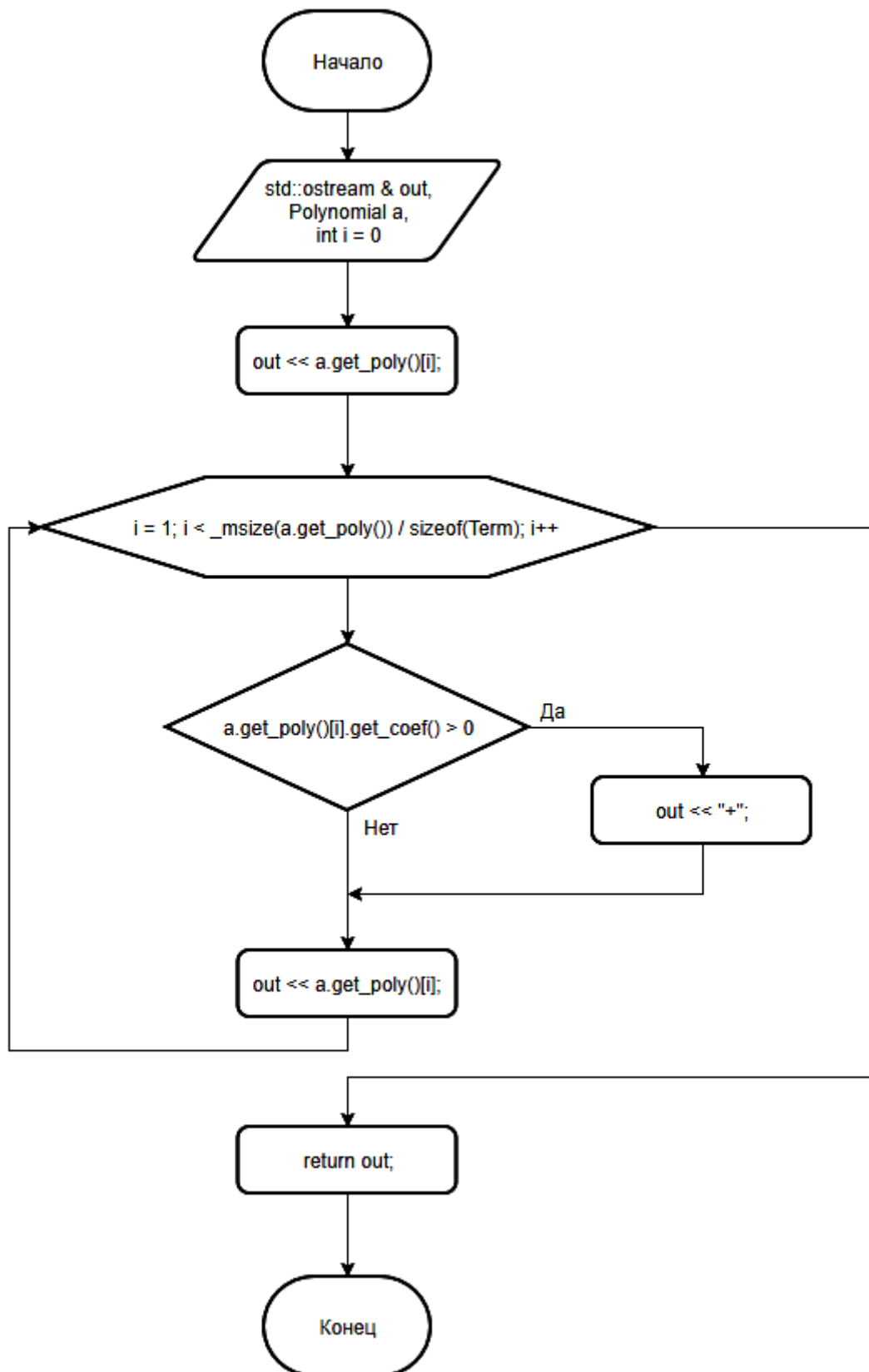


Схема алгоритма оператора вывода для класса Polynomial:



### 3. Текст программы

```
#include <iostream>
#include <string>

using namespace std;

class Term {
private:
    int meas;

    int coef;

public:
    Term(int a = 0, int b = 0) :meas(b), coef(a) {}

    void read(string& str, int& i) {
        char temp;
        int sign = 1;
        meas = 0;
        coef = 0;
        if (int(str[i]) == 43) {
            i++;
        }
        else {
            if (int(str[i]) == 45) {
                sign = -1 * sign;
                i++;
            }
        }
        while ((int(str[i]) >= 48) && (int(str[i]) <= 57)) {
            coef = coef * 10 + int(str[i]) - 48;
            i++;
        }
        coef = coef * sign;
        if (str[i] == 'x') {
            i++;
            if (coef == 0) {
                coef = sign;
            }
            if (str[i] == '^') {
                i++;
                while ((int(str[i]) >= 48) && (int(str[i]) <= 57)) {
                    meas = meas * 10 + int(str[i]) - 48;
                    i++;
                }
            }
            else {
                meas = 1;
            }
        }
        else {
            meas = 0;
        }
    }

    int get_meas() {
        return meas;
    }

    int get_coef() {
        return coef;
    }

    Term operator+ (Term t) {
        if (this->meas == t.meas) {
```

```

        return Term(this->coef + t.coef, t.meas);
    }
    else {
        cout << "Сложение термов разных степеней. Результат - первый терм без
изменений." << endl;
        return Term(this->coef, this->meas);
    }
}

Term operator* (Term& t) {
    return Term(this->coef * t.coef, this->meas + t.meas);
}

Term operator-() {
    return Term(-coef, meas);
}

Term operator- (Term & t) {
    return *this + (-t);
}

};

class Polynomial {
private:
    Term* poly;

    int degree;

public:
    Polynomial() :poly(new Term(0)), degree(0) {}

    Polynomial(const Polynomial& p) :degree(p.degree) {
        poly = new Term[_msize(p.poly) / sizeof(Term)];
        for (int i = 0; i < _msize(p.poly) / sizeof(Term); i++) {
            poly[i] = p.poly[i];
        }
    }

    Polynomial(const Term& t) {
        poly = new Term(t);
        degree = poly[0].get_meas();
    }

    ~Polynomial() {
        delete[] poly;
        poly = NULL;
    }

    Term* get_poly() {
        return poly;
    }

    friend std::istream& operator>>(std::istream& in, Polynomial& a);

    Polynomial& operator+ (Polynomial p) {
        int i, j, k, stp = 0, ij;
        Term* temp1 = new Term[_msize(this->poly) / sizeof(Term) + _msize(p.poly) /
sizeof(Term)];
        j = 0;
        for (i = 0; i < _msize(this->poly) / sizeof(Term); i++) {
            stp = 0;
            for (k = 0; k < j; k++) {
                if (temp1[k].get_meas() == this->poly[i].get_meas()) {
                    temp1[k] = temp1[k] + this->poly[i];
                    j--;
                }
            }
        }
    }

```

```

        stp = -1;
    }
}
if (stp != -1) {
    temp1[j] = this->poly[i];
}
j++;
}
for (i = 0; i < _msize(p.poly) / sizeof(Term); i++) {
    stp = 0;
    for (k = 0; k < j; k++) {
        if (temp1[k].get_meas() == p.poly[i].get_meas()) {
            temp1[k] = temp1[k] + p.poly[i];
            j--;
            stp = -1;
        }
    }
    if (stp != -1) {
        temp1[j] = p.poly[i];
    }
    j++;
}
Polynomial* rs = new Polynomial;
rs->poly = new Term[j];
for (i = 0; i < j - 1; i++) {
    ij = i;
    for (k = i + 1; k < j; k++) {
        if (temp1[k].get_meas() > temp1[ij].get_meas()) {
            ij = k;
        }
    }
    if (ij != i) {
        rs->poly[i] = temp1[ij];
        temp1[ij] = temp1[i];
        temp1[i] = rs->poly[i];
    }
    else {
        rs->poly[i] = temp1[ij];
    }
}
rs->poly[j - 1] = temp1[j - 1];
delete[] temp1;
temp1 = NULL;
rs->degree = rs->poly[0].get_meas();
return *rs;
}

Polynomial& operator* (Polynomial p) {
    int i, j, k, stp = 0, ij;
    Term* temp1 = new Term[_msize(this->poly) / sizeof(Term) * _msize(p.poly) /
sizeof(Term)];
    j = 0;
    for (i = 0; i < _msize(this->poly) / sizeof(Term); i++) {
        for (int q = 0; q < _msize(p.poly) / sizeof(Term); q++) {
            stp = 0;
            for (k = 0; k < j; k++) {
                if (temp1[k].get_meas() == this->poly[i].get_meas() +
p.poly[q].get_meas()) {
                    temp1[k] = temp1[k] + this->poly[i] * p.poly[q];
                    j--;
                    stp = -1;
                }
            }
            if (stp != -1) {
                temp1[j] = this->poly[i] * p.poly[q];
            }
            j++;
        }
    }
}

```

```

Polynomial* rs = new Polynomial;
rs->poly = new Term[j];
for (i = 0; i < j - 1; i++) {
    ij = i;
    for (k = i + 1; k < j; k++) {
        if (temp1[k].get_meas() > temp1[ij].get_meas()) {
            ij = k;
        }
    }
    if (ij != i) {
        rs->poly[i] = temp1[ij];
        temp1[ij] = temp1[i];
        temp1[i] = rs->poly[i];
    }
    else {
        rs->poly[i] = temp1[ij];
    }
}
rs->poly[j - 1] = temp1[j - 1];
delete[] temp1;
temp1 = NULL;
rs->degree = rs->poly[0].get_meas();
return *rs;
}

```

```

Polynomial& operator= (const Polynomial& p) {
    if (this == &p) {
        return *this;
    }
    degree = p.degree;
    if (poly != NULL) {
        delete[] poly;
        poly = NULL;
    }
    poly = new Term[_msize(p.poly) / sizeof(Term)];
    for (int i = 0; i < _msize(p.poly) / sizeof(Term); i++) {
        poly[i] = p.poly[i];
    }
    return *this;
}

```

```

Polynomial operator+= (Polynomial p) {
    *this = *this + p;
    return *this;
}

```

```

Polynomial operator*= (Polynomial p) {
    *this = *this * p;
    return *this;
}

```

```

Polynomial operator-() {
    Polynomial* temp = new Polynomial(*this);
    for (int i = 0; i < _msize(temp->poly) / sizeof(Term); i++) {
        temp->poly[i] = -temp->poly[i];
    }
    return *temp;
}

```

```

Polynomial& operator- (Polynomial& p) {
    return *this + (-p);
}

```

```

};

```

```

std::istream& operator>>(std::istream & in, Polynomial & a) {
    string str;
    Term temp2;
    Term* temp1;

```



```

int sz = 1, i, j = 0, k, ij, stp = 0;
getline(in, str);
for (i = 0; i < str.length(); i++) {
    if (str[i] == 32) {
        str.erase(i, 1);
        i--;
    }
    else {
        if ((str[i] == 43) || (str[i] == 45)) {
            sz = sz + 1;
        }
    }
}
temp1 = new Term[sz];
i = 0;
while (i < str.length()) {
    temp2.read(str, i);
    stp = 0;
    for (k = 0; k < j; k++) {
        if (temp1[k].get_meas() == temp2.get_meas()) {
            temp1[k] = temp1[k] + temp2;
            j--;
            stp = -1;
        }
    }
    if (stp != -1) {
        temp1[j] = temp2;
    }
    j++;
}
if (a.poly != NULL) {
    delete[] a.poly;
    a.poly = NULL;
}
a.poly = new Term[j];
for (i = 0; i < j - 1; i++) {
    ij = i;
    for (k = i + 1; k < j; k++) {
        if (temp1[k].get_meas() > temp1[ij].get_meas()) {
            ij = k;
        }
    }
    if (ij != i) {
        a.poly[i] = temp1[ij];
        temp1[ij] = temp1[i];
        temp1[i] = a.poly[i];
    }
    else {
        a.poly[i] = temp1[ij];
    }
}
a.poly[j - 1] = temp1[j - 1];
delete[] temp1;
temp1 = NULL;
a.degree = a.poly[0].get_meas();
return in;
}

std::ostream& operator<< (std::ostream & out, Term a) {
    if (a.get_coef() == 0) {
        out << '0';
        return out;
    }
    else {
        if (a.get_coef() != 1) {
            out << a.get_coef();
        }
        if (a.get_meas() == 0) {
            return out;

```

```

    }
    else {
        out << 'x';
        if (a.get_meas() == 1) {
            return out;
        }
        else {
            out << '^' << a.get_meas();
            return out;
        }
    }
}

}

std::ostream& operator<< (std::ostream & out, Polynomial a) {
    int i = 0;
    out << a.get_poly()[i];
    for (i = 1; i < _msize(a.get_poly()) / sizeof(Term); i++) {
        if (a.get_poly()[i].get_coef() > 0) {
            out << "+";
        }
        out << a.get_poly()[i];
    }
    return out;
}

int main()
{
    setlocale(LC_ALL, "Russian");

    Term a(-3), b(1, 2), c(0, 19), d(-7, 0), e(b), f(b * e), t, h(b - e);
    cout << a << endl << b << endl << c << endl << d << endl << e << endl << f << endl << t <<
endl << h << endl;

    cout << endl;


    Polynomial aa(a), tt(t), zz, bb(b), oo(bb + aa);
    cout << aa << endl << tt << endl << zz << endl << bb << endl << oo << endl;

    Polynomial aaa, bbb;
    cin >> aaa;
    cout << endl;
    cin >> bbb;
    cout << "a = " << aaa << endl << "b = " << bbb << endl << "a + b = " << aaa + bbb << endl <<
"a - b = " << aaa - bbb << endl << "a * b = " << aaa * bbb << endl;

    system("pause");
    return 0;
}

```

#### 4. Анализ результатов

 C:\Users\webma\source\repos\lab5\Debug\lab5.exe

```
-3
x^2
0
-7
x^2
x^4
0
0

-3
0
0
x^2
x^2-3
3x^2 + 5 x - 10

x^3 - 2x ^2 - 7x + 4
a = 3x^2+5x-10
b = x^3-2x^2-7x+4
a + b = x^3+x^2-2x-6
a - b = -1x^3+5x^2+12x-14
a * b = 3x^5-1x^4-41x^3-3x^2+90x-40
Press any key to continue . . . ■
```

Программа работает исправно.