Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Вычисление арифметических выражений»**

**Выполнил**:

студент группы 382003-1

Измайлов И.Р.

Нижний Новгород

2022г.

**Содержание**

[Постановка задачи 3](#_Toc103543801)

[Руководство пользователя 4](#_Toc103543802)

[Описание программной реализации 7](#_Toc103543803)

[Приложение 9](#_Toc103543804)

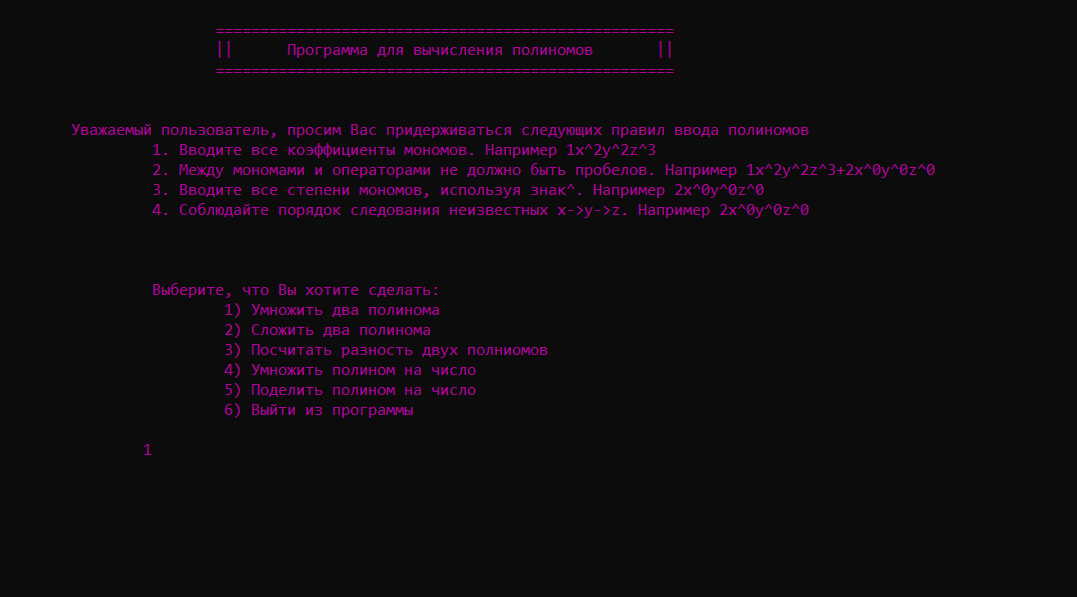
# Постановка задачи

Передо мной была поставлена задача – повторить весь материал по языку программирования С++, который мы успели пройти на момент окончания 4 семестра. На основе этих знаний написать программу, которая бы выполняла арифметические арифметические операции с полиномами трех переменных(x,y,z). Программа должна уметь: складывать, вычитать, умножать полиномы, умножать константу на полином.

В нашей реализации считается, что полином составлен из мономов от трех переменных с ограничением на степень каждой переменной от 0 до 9. Коэффициенты полинома – вещественные числа.

# Руководство пользователя

Пользователь на стартовой странице выбирает, что необходимо сделать.

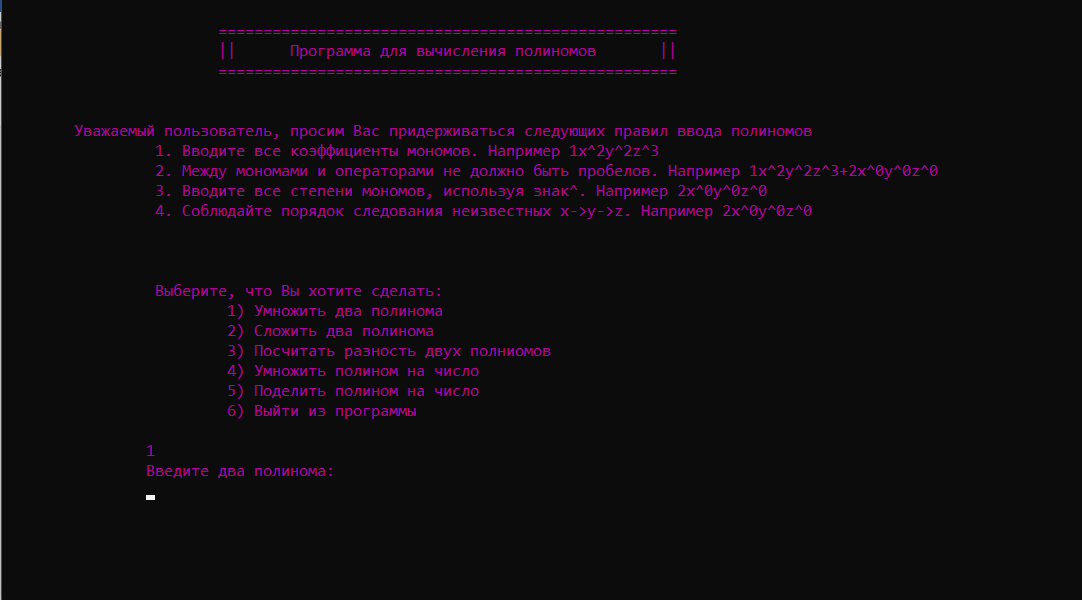


На экране пользователь получает правила для корректной работы с программой.

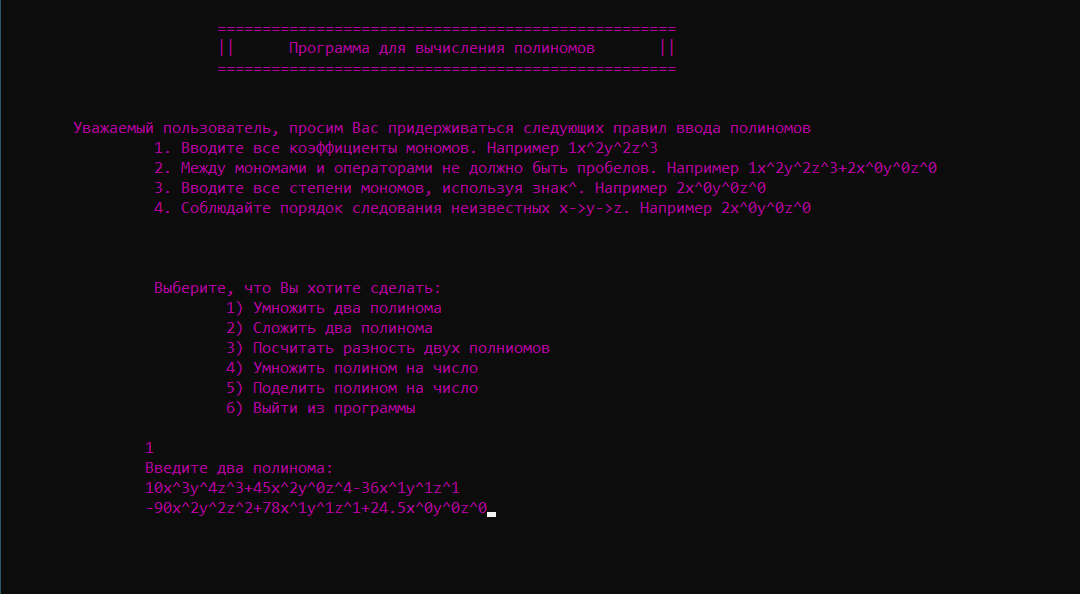
После ознакомления с инструкцией, пользователь может перейти к непосредственной работе. Перед ним появляется меню с действиями. Для продолжения работы необходимо ввести соответствующее число:

* 1. Умножить два полинома
  2. Сложить два полинома
  3. Посчитать разность двух полиномов
  4. Умножить полином на число
  5. Поделить полином на число
  6. Выйти из программы

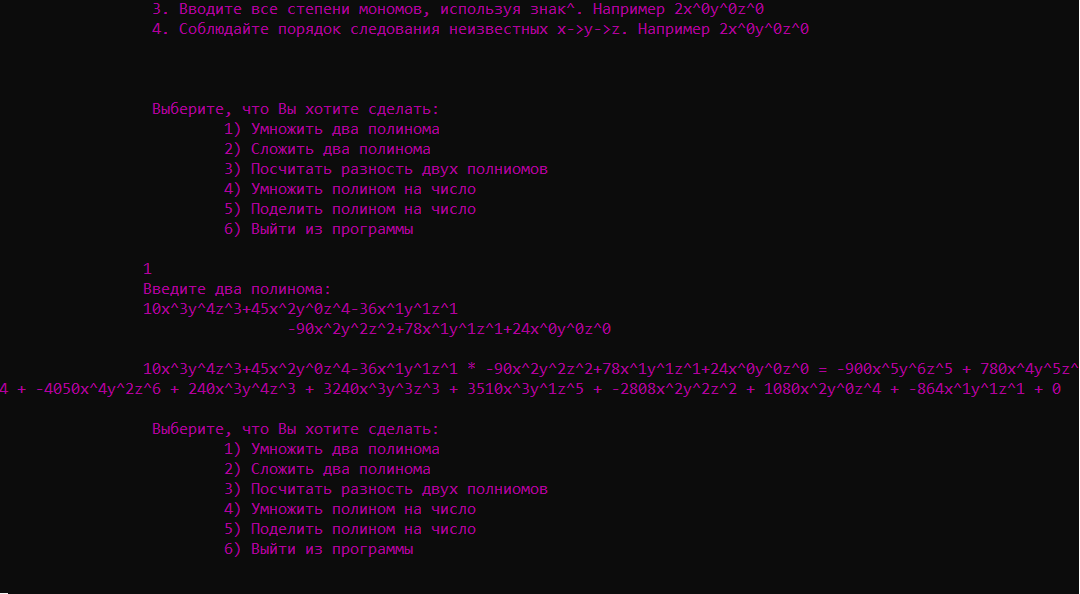
Для наглядности мы умножим два полинома. Введем число 1.



Программа просит ввести два полинома, которые будут умножаться. Введем каждый полином, соблюдая инструкцию.



После ввода полиномов нажмем Enter.



Программа выдала результат вычисления.

После вычислений перед пользователем высвечивается меню. Для дальнейшей работы программы необходимо так же ввести число. Мы завершим работу программы, введя число 6.

# Описание программной реализации

В качестве структуры хранения полинома используем односвязный список (List) мономов(monom). Степень монома хранится в «свернутом» виде, т.е. степень должна быть представлена как трехзначное число (в переменной row), где число сотен – это степень при переменной «х», число десятков – степень при переменной «у», число единиц – степень при переменной «z». Сложение полиномов осуществляется алгоритмом слияния упорядоченных массивов.

Если при умножении полиномов полученные степени переменных больше 9, выводится сообщение об ошибке.

**Подтверждение корректности**

Проверка корректности алгоритма осуществляется с помощью gtests.

# Приложение

polinom operator+(const polinom& m\_first, const polinom& m\_second) {

List<Monom>::iterator it\_1 = m\_first.begin();

List<Monom>::iterator it\_2 = m\_second.begin();

polinom result;

while (it\_1 != m\_first.end() && it\_2 != m\_second.end()) {

if (\*it\_1 > \*it\_2) {

result.push\_back(\*it\_1);

it\_1++;

}

else if (\*it\_1 < \*it\_2) {

result.push\_back(\*it\_2);

it\_2++;

}

else {

result.push\_back(\*it\_1 + \*it\_2);

it\_1++;

it\_2++;

}

}

if (it\_1 == m\_first.end() && it\_2 != m\_second.end()) {

result.push\_back(\*it\_2);

}

else if (it\_2 == m\_second.end() && it\_1 != m\_first.end()) {

result.push\_back(\*it\_1);

}

result.cleaning();

return result;

}

polinom operator-(const polinom& m\_first, const polinom& m\_second) {

return (m\_first + (-1) \* m\_second);

}

polinom operator\*(const polinom& m\_first, const polinom& m\_second) {

polinom result;

for (List<Monom>::iterator itf = m\_first.begin(); itf != m\_first.end(); itf++) {

for (List<Monom>::iterator its = m\_second.begin(); its != m\_second.end(); its++) {

result.insert((\*itf) \* (\*its));

}

}

return result;

}

polinom operator\*(const int& first, const polinom& m\_second) {

return static\_cast<double>(first)\*m\_second;

}

polinom operator\*(const double& first, const polinom& m\_second) {

polinom result;

List<Monom>::iterator it = m\_second.begin();

for (it; it != m\_second.end(); it++) {

result.push\_back(first \* (\*it));

}

return result;

}

polinom operator\*(const polinom& m\_second, const int& second) {

return (second \* m\_second);

}

polinom operator\*(const polinom& m\_second, const double& second) {

return (second \* m\_second);

}

polinom operator/(const polinom& m\_first, const int& second) {

return (m\_first / static\_cast<double>(second));

}

polinom operator/(const polinom& m\_first, const double& second) {

polinom result;

for (List<Monom>::iterator it = m\_first.begin(); it != m\_first.end(); it++) {

result.push\_back((\*it) / second);

}

return result;

}