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.

**Проверила:**

Арисова А.Н.

Нижний Новгород

2022г.

**Содержание**

[Постановка задачи 3](#_Toc104135980)

[Руководство пользователя 4](#_Toc104135981)

[Описание программной реализации 7](#_Toc104135982)

[Приложение 12](#_Toc104135983)

[Заключение 14](#_Toc104135984)

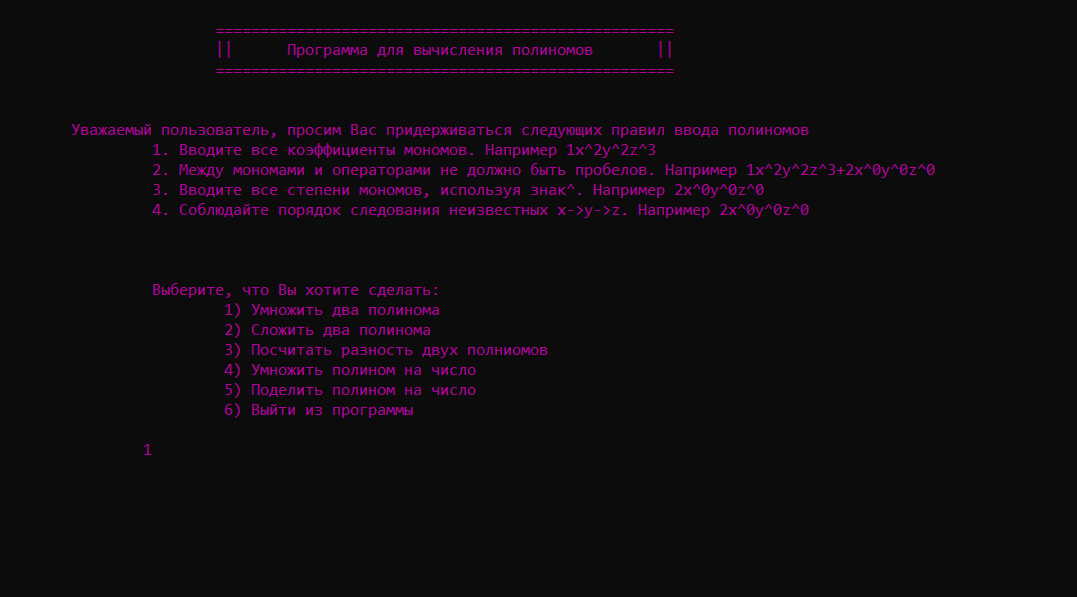
# Постановка задачи

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 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.

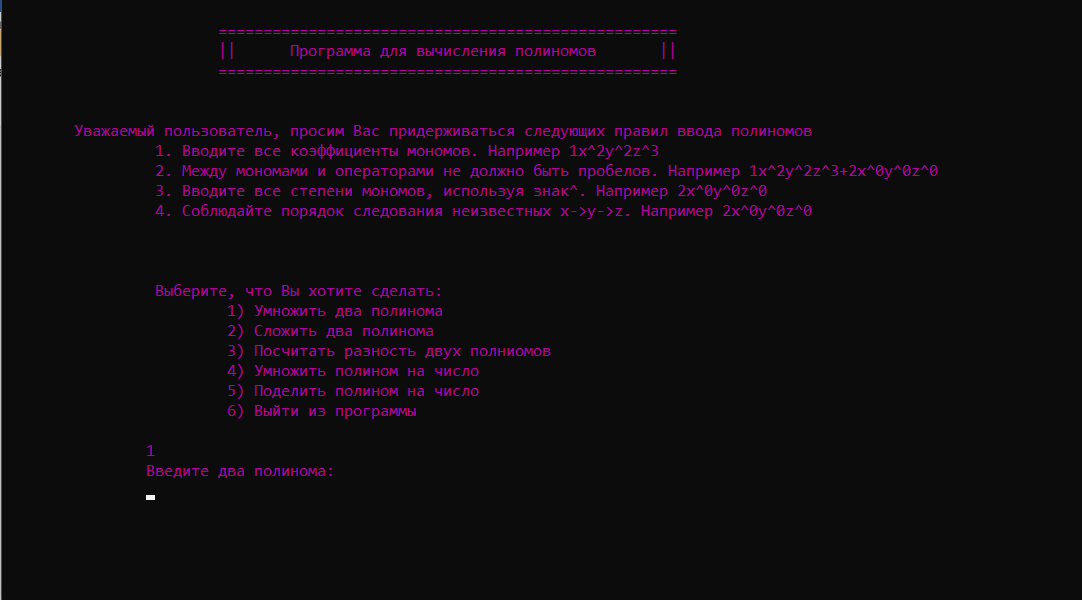


На экране пользователь получает правила для корректной работы с программой.

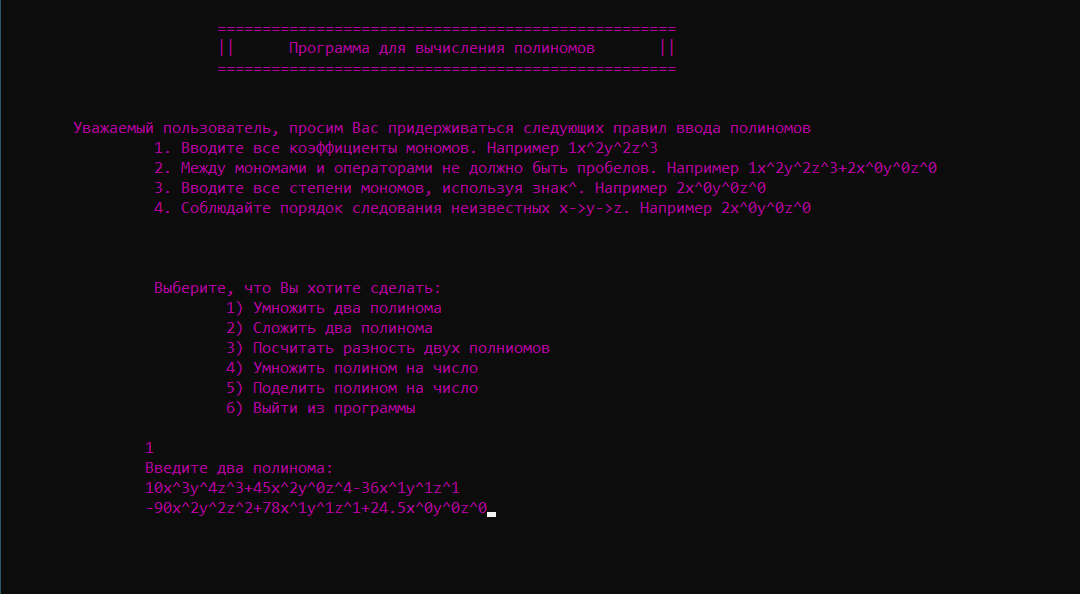
После ознакомления с инструкцией, пользователь может перейти к непосредственной работе. Перед ним появляется меню с действиями. Для продолжения работы необходимо ввести соответствующее число:

* 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

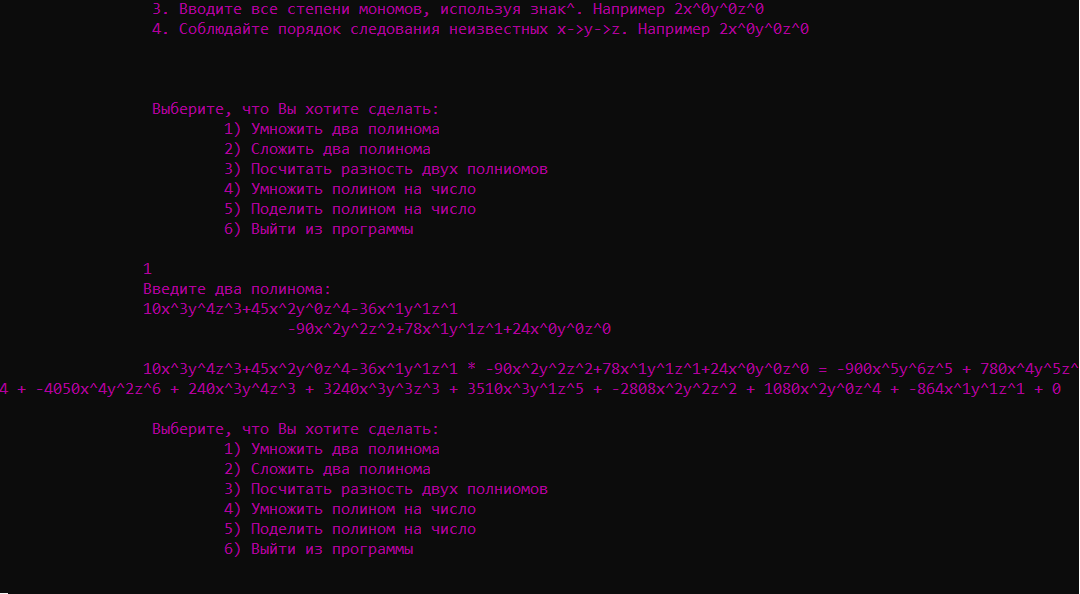
Для наглядности мы умножим два полинома. Введем число 1.



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

**Структуры данных**

Программа была реализована с помощью основных структур данных, усвоенных за этот учебный год. Для хранения полиномов использовался односвязный список (List). Узлом списка выступал класс 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.

Стоит отметить, что класс внутри класса список реализован вложенный класс итератор(iterator). Объекты этого класса позволяют проходить по списку и выводить данные, которые хранятся в узлах.

**Алгоритм выполнения**

Большинство операций над полиномами проводились с помощью слияния двух отсортированных списков. Создавался полином для сохранения результата, куда по очереди добавлялись мономы. При добавлении мономов, отсортированность полинома не нарушалась.

Алгоритмы сложения двух полиномов и вычета одного из другого были схожи. Перед вычетом первого полинома из второго, мы умножали второй – на минус единицу, а затем складывали.

Умножение и деление полиномов на число происходило линейно. Мы проходились по списку мономов в полиноме и по очереди умножали или делили коэффициент монома на это число.

**Руководство программиста**

Шаблонный класс списка был написан до того, как начал выполнять эту лабораторную работу. Старался придерживаться той же логики, что и std::list из стандартной библиотеки с++. List имеет следующие методы

* + List<T>& operator=(const List& List\_m);
  + List<T>& operator=(List&& List\_m);
  + T read\_first() { return first->data; }
  + int size() // возвращает размер(количество узлов) списка
  + bool empty() // метод проверки пустоты
  + void delElem(T data\_m); // удаление всех узлов, хранящие узлы data\_m
  + void clear(); // удаление всех узлов в списке
  + void push\_front(const T& m\_data); //вставка элемента в начало
  + void push\_back(const T& m\_data); // вставка элемента в конец списка
  + void push\_back(node<T>\* Node\_m); //вставка узла в конец списка
  + void pop\_front(); // удаление первого узла
  + void pop\_back(); // удаление последнего узла
  + void reverse(); // переворачивает список
  + void sort(); // сортировка списка «по убыванию»
  + void show(); // вывод элементов списка

Узлы списка реализованы в виде шаблонного класса node. В этом классе доступны следующие методы:

* bool operator==(const node<T>& m\_node);
* bool operator<(const node& m\_node);
  + - bool operator>(const node& m\_node);
    - bool operator>=(const node& m\_node);

Полином(polinom) – явная специализация класса List типом данных monom. Следующие функции, определяют операции над полиномами.

* polinom operator+(const polinom& m\_first, const polinom& m\_second);
* polinom operator-(const polinom& m\_first, const polinom& m\_second);
* polinom operator\*(const polinom& m\_first, const polinom& m\_second);
* polinom operator\*(const int& first, const polinom& m\_second);
* polinom operator\*(const double& first, const polinom& m\_second);
* polinom operator\*(const polinom& m\_second, const int& second);
* polinom operator\*(const polinom& m\_second, const double& second);
* polinom operator/(const polinom& m\_first, const int& second);
* polinom operator/(const polinom& m\_first, const double& second);

Класс monom содержит в себе методы сравнения.

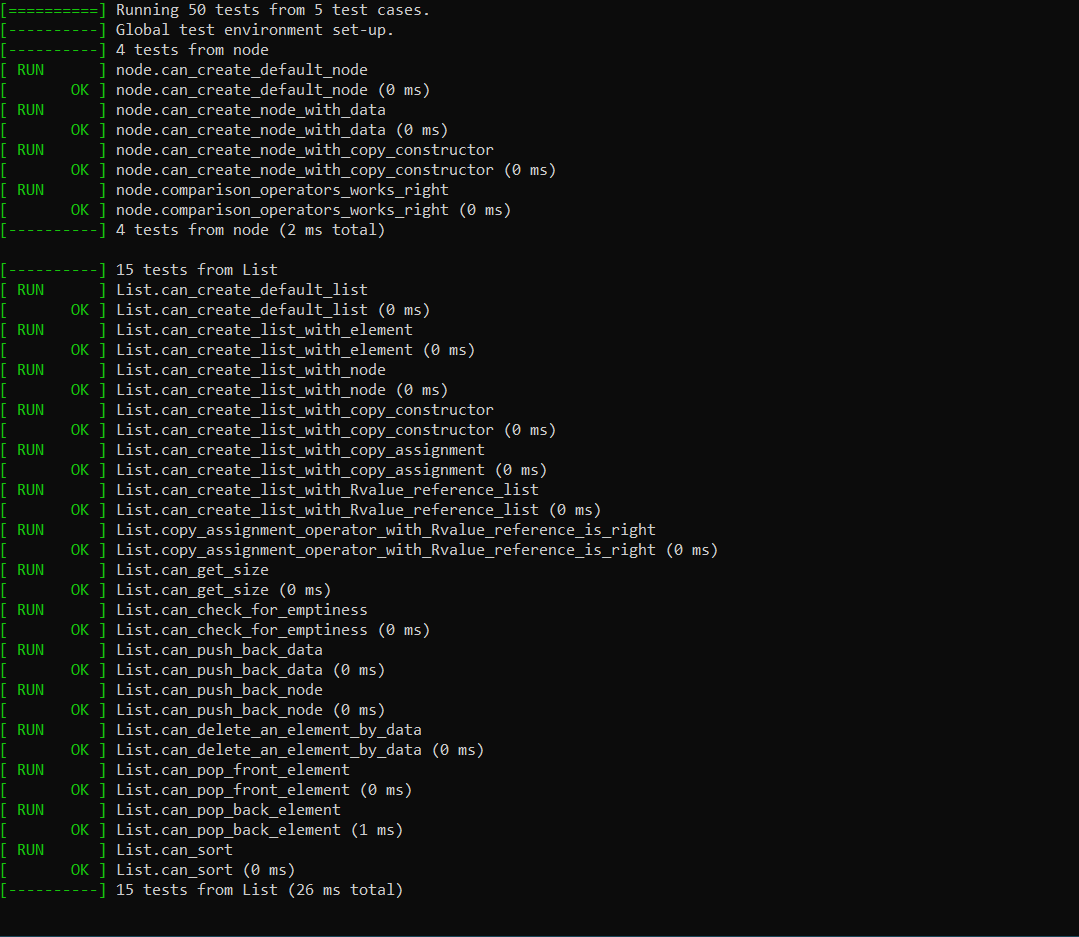
* bool operator==(const Monom& m\_monom);
* bool operator==(const Monom& m\_monom) const;
* bool operator<(const Monom& m\_monom);
* bool operator<(const Monom& m\_monom) const;
* bool operator>(const Monom& m\_monom);
* bool operator>(const Monom& m\_monom) const;
* bool operator>=(const Monom& m\_monom);
  + - bool operator>=(const Monom& m\_monom) const;

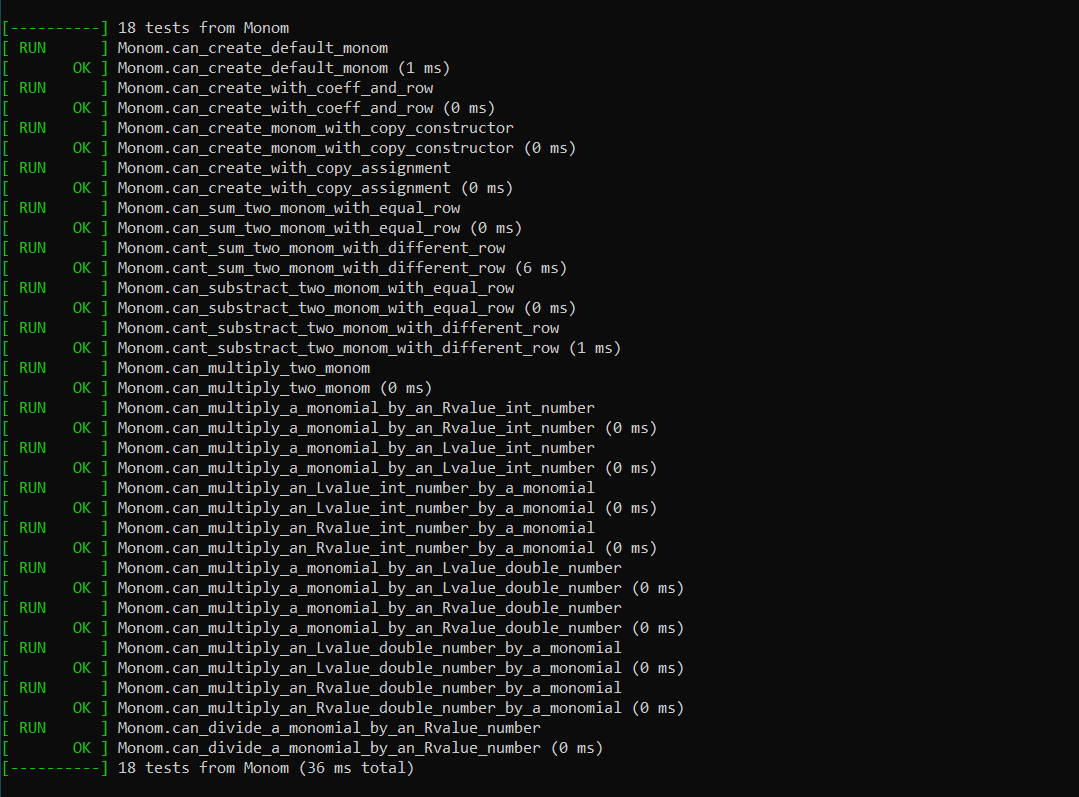
Следующие функции, определяют основные операции над мономами.

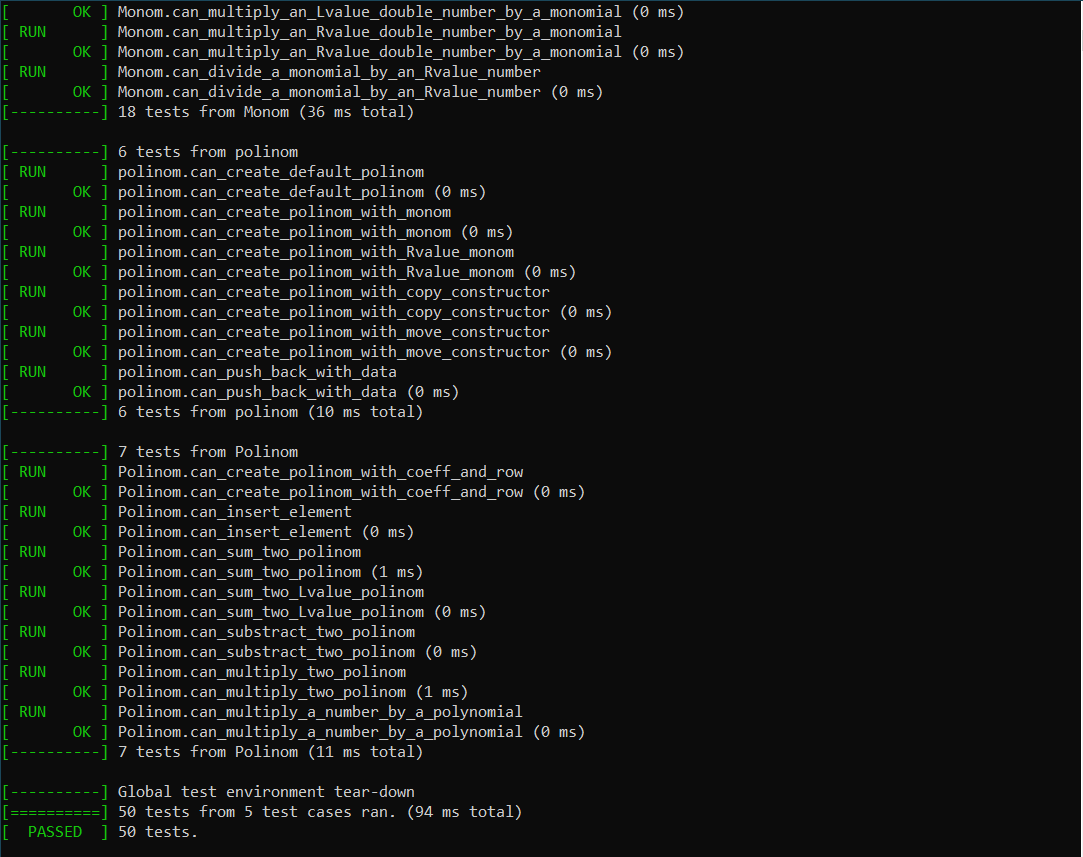
* Monom operator+(const Monom& m\_first, const Monom& m\_second);
* Monom operator-(const Monom& m\_first, const Monom& m\_second);
* Monom operator\*(const Monom& m\_first, const Monom& m\_second);
* Monom operator\*(const Monom& m\_first, const double& second);
* Monom operator\*(const Monom& m\_first, const int& second);
* Monom operator\*(const double& first, const Monom& m\_second);
* Monom operator\*(const int& first, const Monom& m\_second);
* Monom& operator/(const double& a);
* Monom& operator+=(const Monom& m\_monom);

**Подтверждение корректности**

Проверка корректности алгоритма осуществляется с помощью gtests.

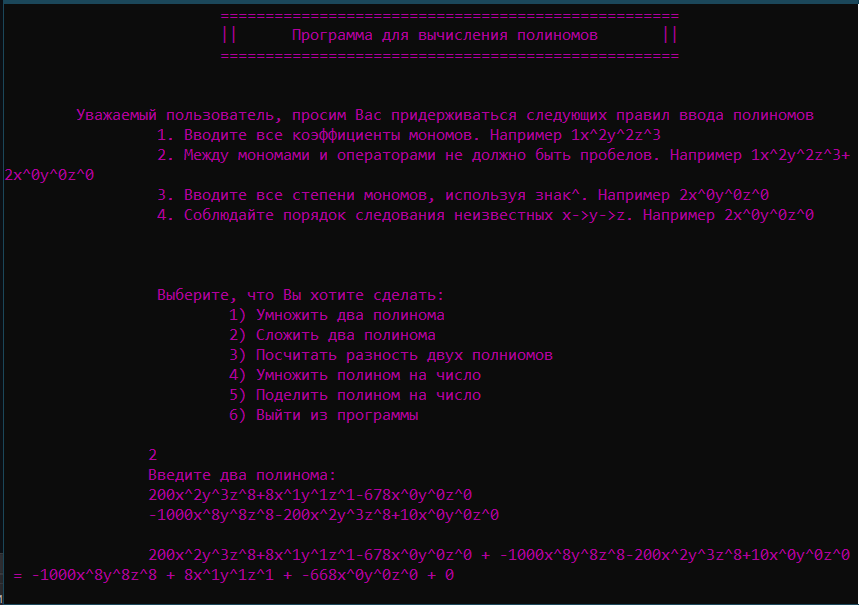






Для подтверждения корректной работы программы произведем операции над собственными полиномами. Убедимся, что полученный ответ – верный.

В качестве примера введем два полинома. Первый: 200x^2y^3z^8+8x^1y^1z^1-678x^0y^0z^0. Второй: -1000x^8y^8z^8-200x^2y^3z^8+10x^0y^0z^0. При сложении двух полиномов должен получится ответ -1000x^8y^8z^8+8x^1y^1z^1-668x^0y^0z^0



Как видно, программа выдала нам верный ответ.

# 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;

}

# Заключение

В процессе выполнения лабораторной работы, я повторил основные структуры данных и алгоритмы, пройденные за этот семестр, улучшил свои навыки проектирования программ.

Данная программа может быть полезна при расширении функциональных возможностей предыдущей лабораторной работы. Возможно встроить полином в калькулятор и выполнять более сложные арифметические операции.

Следующая лабораторная работа «таблицы» будет основываться на данную. Элементы, которые мы будем хранить в таблицах, будут полиномы.