Лабораторная работа №1

по программированию на языке С++

“Классы”

**Задание:**

Составить на языке C++ описания классов для указанных объектов. Написать методы класса, демонстрирующие использование объектов разработанного класса.

Многочлен от нескольких переменных на основе двунаправленного списка(сложение, вычитание, умножение, деление – две операции).

Класс, применяемый в программе:

struct elem

{

public:

float a; // коэффицент

int x; // степень x

int y; // степень y

int z; // степень z

elem \*next; // следующий

elem \*prev; // предыдущий

};

class Pol

{

private:

elem \*head; // начало списка

elem \*tail; // конец списка

public:

Pol():head(NULL),tail(NULL){}; // Конструктор

Pol(const Pol&a); // Конструктор копирования

~Pol(); // Деструктор

void AddN(float,int,int,int); // добавление в начало

void AddK(float,int,int,int); // добавление в конец

void DelN(); // удаление из начала

void DelK(); // удаление из конца

void Poisk(); // Поиск узла

void Vved2(); // Печать с начала

void Sort(); // Сортировка

void Sort2();

void Sum();

friend Pol operator+(const Pol& L,const Pol& E); // Сложение

friend Pol operator-(const Pol& L,const Pol& E); // Вычитание

friend Pol operator\*(const Pol& L,const Pol& E); // Произведение

friend Pol operator/(const Pol& L,const Pol& E); // Деление

friend Pol operator%(const Pol& L,const Pol& E); // Остаток от деления

};

**Описание работы программы:**

При старте программы пользователь попадает в главное меню, где должен выбрать над каким многочленом будут производиться действия(цифра 1 или 2). После чего путём ввода одной из цифр пользователь выбирает что ему сделать с этим многочленом(добавить элемент, удалить элемент, вывести многочлен, сложить многочлены и другие). Таким же способом можно вызвать один из подготовленных тестов. Каждый из двух многочленов представляется своим экземпляром класса.

Многочлен имеет вид: a\*xx1\*yy1\*zz1

Где a – это коэффицент типа float

x1,y1,z1 – степени при соответствующих переменных типа int

При вызове функции сложения многочленов два многочлена сначала соединяются в один новый многочлен с помощью функции-оператора Pol operator+(const Pol& L,const Pol& E). Потом все слагаемые в данном многочлене складываются с помощью функции void Pol::Sum(). После чего вызывается функция

void Pol::Sort() для сортировки слагаемых в получившемся многочлене и удалении лишних нулевых слагаемых.

Функция вычитания работает так же, только вызывается

Pol operator-(const Pol& L,const Pol& E) где коэффиценты элементов второго многочлена становятся отрицательным.

Функция произведения вызывает

Pol operator\*(const Pol& L,const Pol& E) где в новый многочлен(представляемый новым экземпляром класса) добавляются произведения элементов первого многочлена на второй(коэффиценты умножаются, степени складываюся). После вызываются те же функции void Pol::Sum() и void Pol::Sort() складывающие слагаемые внутри многочлена и сортирующие их.

Для функции деления сначала вызываем функции void Pol::Sum() и void Pol::Sort() для каждого из двух многочленов(экземпляров класса) а потом вызывается функция-оператор

Pol operator/(const Pol& L,const Pol& E) где первый многочлен становится делимым, а второй – делителем. Для деления многочленов используется способ деления в столбик. Используются также экземляры класса Rez и Ost куда записывается результат и остаток соответственно и промежуточный экземпляр Prov, который вычитается из экземпляра класса первого многочлена. Результат этой операции записывается вместо первого многочлена и алгоритм повторяется пока невозможно будет воспроизвести деление.

Вторая функция деления работает аналогично, за сключением того что выводит она содержимое экземляра Ost, а не Rez.

**Алгоритмы выполнения основных операций на псевдокоде:**

**Функция сложения**

{

Новый экземпляр класса S;

Объявляем указатель на “голову” однонаправленного списка связанного с 1 многочленом

Объявляем указатель на “голову” однонаправленного списка связанного с 2 многочленом

пока не конец первого многочлена

{

Добавляем слагаемое из первого многочлена в новый

}

пока не конец второго многочлена

{

Добавляем слагаемое из второго многочлена в новый

}

Вызов функции которая сложит все слагаемые из нового многочлена между собой

Вызов функции которая удалит лишние элементы и отсортирует их по убыванию степеней

Возвращение получившегося многочлена

}

**Функция вычитания**

{

Новый экземпляр класса N;

Объявляем указатель на “голову” однонаправленного списка связанного с 1 многочленом

Объявляем указатель на “голову” однонаправленного списка связанного с 2 многочленом

пока не конец первого многочлена

{

Добавляем слагаемое из первого многочлена в новый

}

пока не конец второго многочлена

{

Добавляем слагаемое из второго многочлена в новый с отрицательным коэффицентом

}

Вызов функции которая сложит все слагаемые из нового многочлена между собой

Вызов функции которая удалит лишние элементы и отсортирует их по убыванию степеней

Возвращение получившегося многочлена

}

**Функция произведения**

{

Новый экземпляр класса D;

Объявляем указатель на “голову” однонаправленного списка связанного с 1 многочленом

Объявляем указатель на “голову” однонаправленного списка связанного с 2 многочленом

пока не конец первого многочлена

{

Пока не конец второго многочлена

{

Добавляем произведение слагаемого многочлена на слагаемое из второго многочлена в новый многочлен

}

}

Вызов функции которая сложит все слагаемые из нового многочлена между собой

Вызов функции которая удалит лишние элементы и отсортирует их по убыванию степеней

Возвращение получившегося многочлена

}

**Функция деления:**

{

Объявление новых экземпляров класса Rez, Ost, Prov(Rez – целая часть от деления многочленов, Ost - остаток от деления многочленов)

Объявляем указатель на “голову” однонаправленного списка связанного с 1 многочленом

Объявляем указатель на “голову” однонаправленного списка связанного с 2 многочленом

Если( размер первого многочлена < размер второго многочлена)

{

Ost = первому многочлену

Rez = нулю

}

Если( размер первого многочлена >= размер второго многочлена)

{

Пока(не конец 1 многочлена)

{

Если(степени 1 многочлена < степеней 2 многочлена)

{

Вызываем функцию для перестановки элементов в 1 многочлене

Если(степени 1 многочлена < степеней 2 многочлена)

{

Ost = первому многочлену

Rez = нулю

}

}

Делим первое слагаемое 1 многочлена на 1 слагаемое 2 многочлена

Добавляем результат деления в Rez

Пока(не конец 2 многочлена)

{

Умножаем добавленное в Rez слагаемое на слагаемое из 2 многочлена

Результат умножения записываем в экземпляр класса Prov

}

Вычитаем из 1 многочлена Prov и записываем в новый экземпляр класса L2

Заменяем первый многочлен(экземпляр L1) экземпляром L2

// таким образом функция вернется в начало цикла и продолжит // работу до полного окончания деления

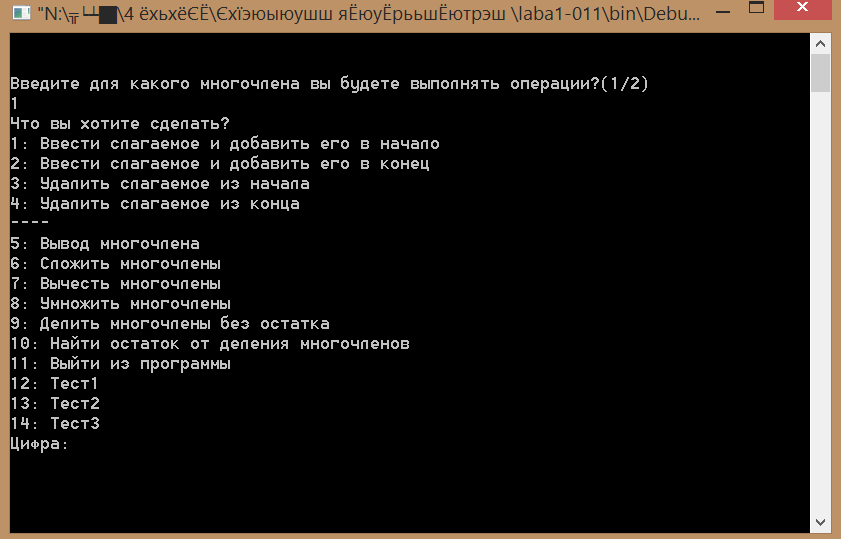
}

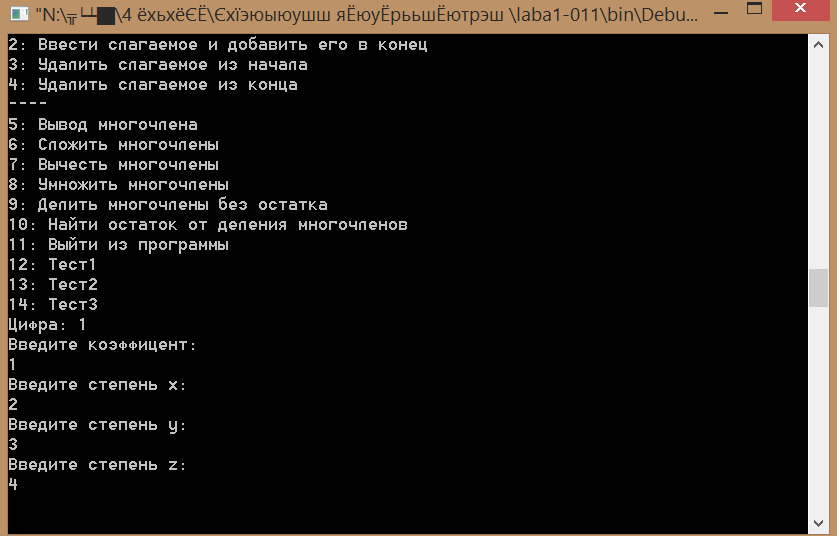
}

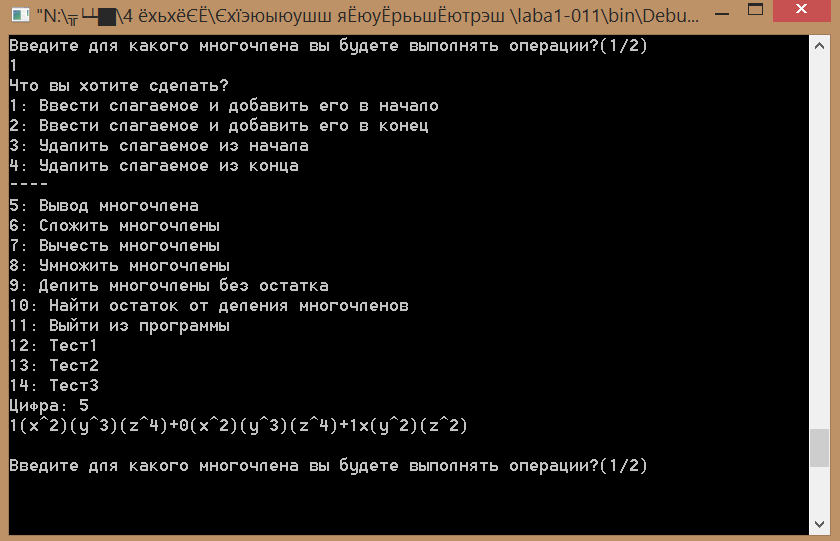
Выводим результат – Rez или Ost

}

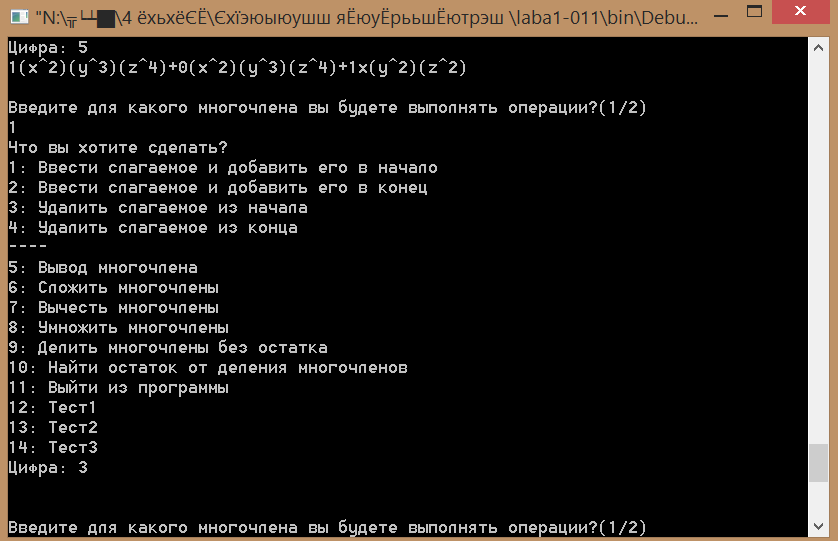
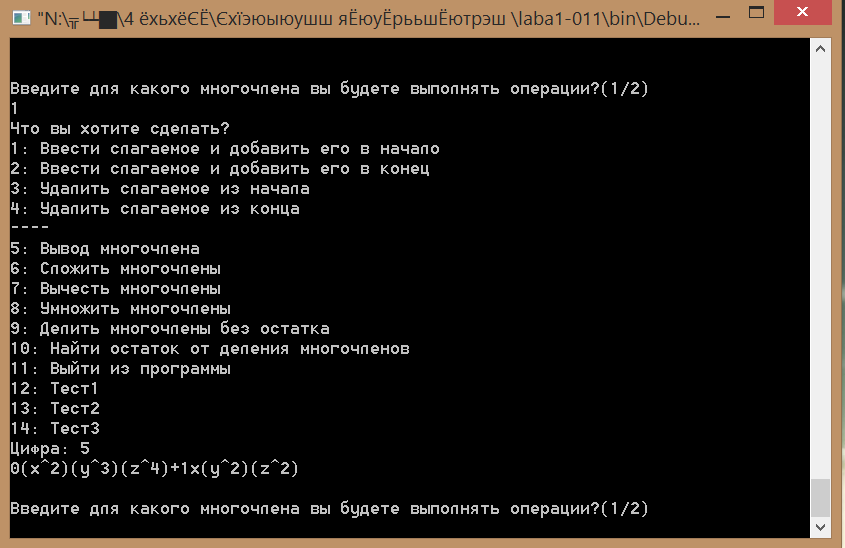
**Распечатки экранов при работе программы:**

Главное меню: 

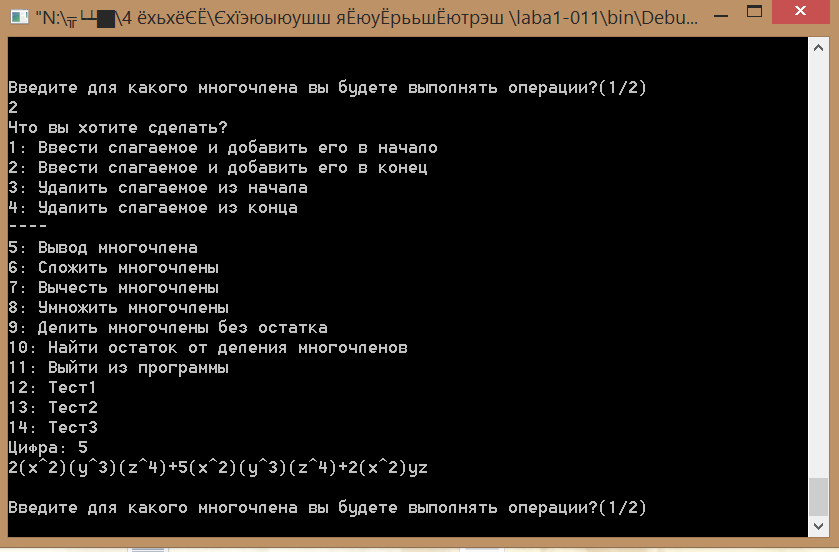
Ввод слагаемого в первое множество:

Вывод первого множества на экран(до этого были добавлены ещё два слагаемых):

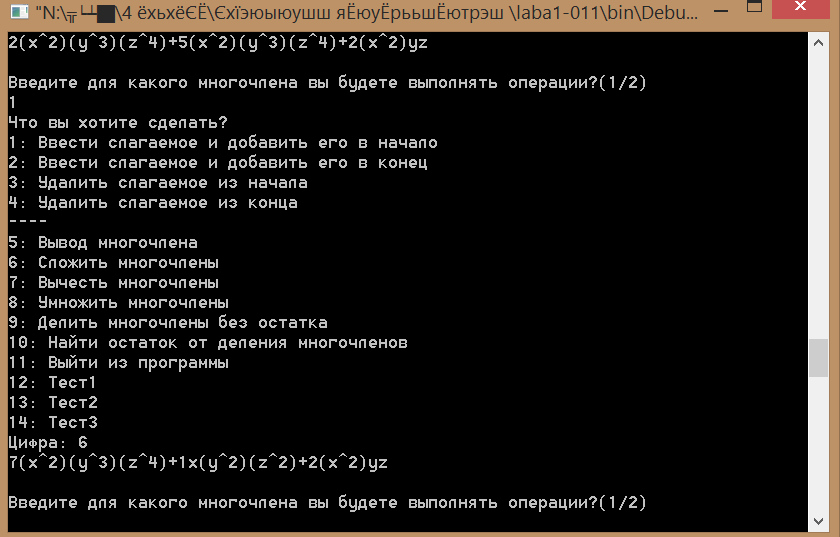
Удаление слагаемого из начала первого множества и вывод его на экран:



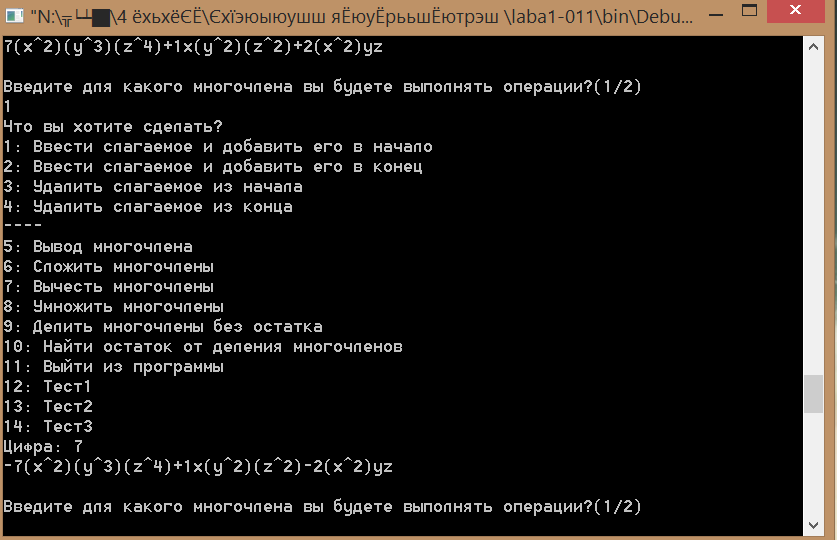
Вводим слагаемые для второго множества и выводим его на экран:



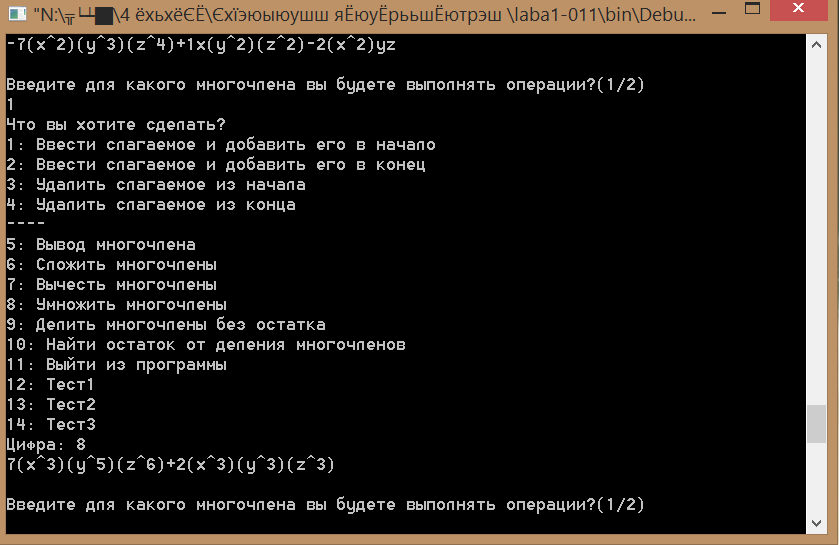
Сложение множеств:



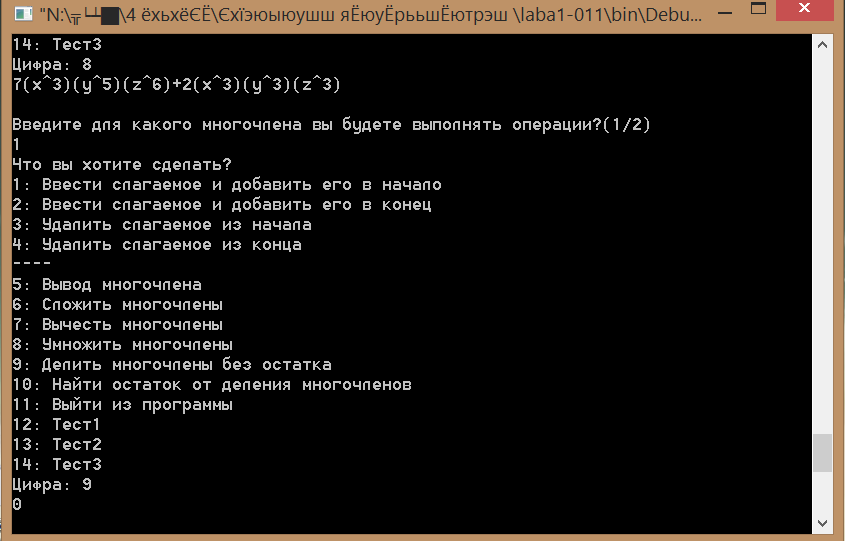
Вычитание множеств:



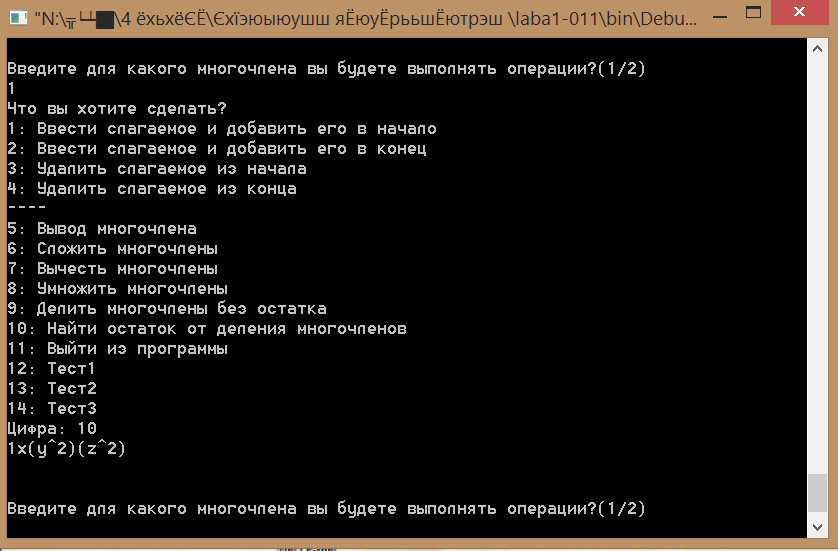
Произведение множеств:



Деление множеств без остатка:

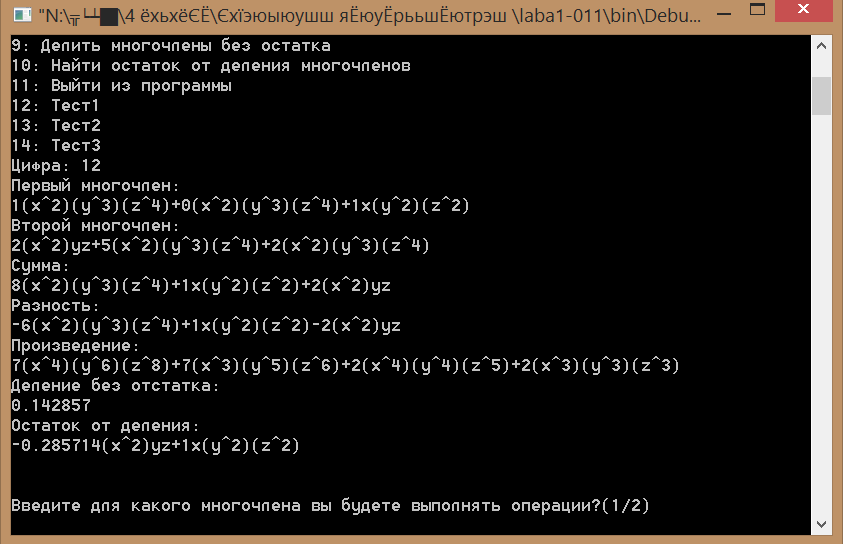


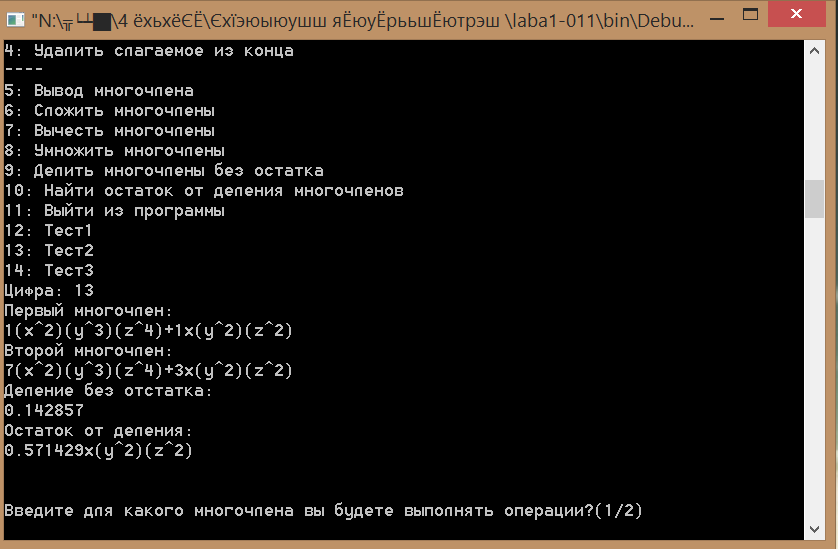
Остаток от деления множеств:



**Тесты:**

1 Тест:



2 Тест: 

3 Тест:

