Министерство образования и науки 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 учебной практике

**Операции с полиномами**

Выполнил:

студент гр. 381703-1

Поликсенов И.А.

Проверил:

Ассистент кафедры МОСТ ИИТММ

Волокитин В.Д.

г. Нижний Новгород

2018г

Оглавление

[Введение 3](#_Toc531969573)

[Постановка учебно-практической задачи 4](#_Toc531969574)

[Руководство пользователя 5](#_Toc531969575)

[Руководство программиста 6](#_Toc531969576)

[Описание структур данных 8](#_Toc531969577)

[Заключение 11](#_Toc531969578)

[Список литературы 12](#_Toc531969579)

[Приложения 13](#_Toc531969580)

[Приложение 1. Класс Stack 13](#_Toc531969581)

[Приложение 2. Класс TLexems 15](#_Toc531969582)

[Приложение 3. Файл arithmetic.cpp 15](#_Toc531969583)

# Введение

Вычисление арифметических выражений встречаются повсеместно на разном уровне.

Все отрасти современности так или иначе используют те или иные вычисления.

Полином — это сумма одночленов или, строго - конечная формальная сумма.

Алгебру активно используют в комбинаторике при расчётах наиболее оптимальных маршрутов, в кристаллографии при изучении структур кристаллов, математический анализ и комплексный анализ нужен в механике, компьютерной графике, теории обтекания крыльев самолёта. Это лишь конкретные примеры, реальных приложений - масса.

# 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 переменой от 0 до 9. Коэффициенты полинома - вещественные числа. Работоспособность программы необходимо проверить с помощью Google Test-ов. Кроме того, необходимо разработать пользовательское консольное приложение.

**Последовательность выполнения работы**

1. Разработка шаблонного класса List;
2. Реализация стандартных операций над полиномами;
3. Создание консольного интерфейса пользователя;

* Запрос полиномов.
* Выбор операций над ними.

1. Реализация проверки всех алгоритмов с помощью тестов.

**Исходные данные**

|  |  |
| --- | --- |
| A, B | Полиномы, реализованные в виде списка мономов, с которыми производятся действия |

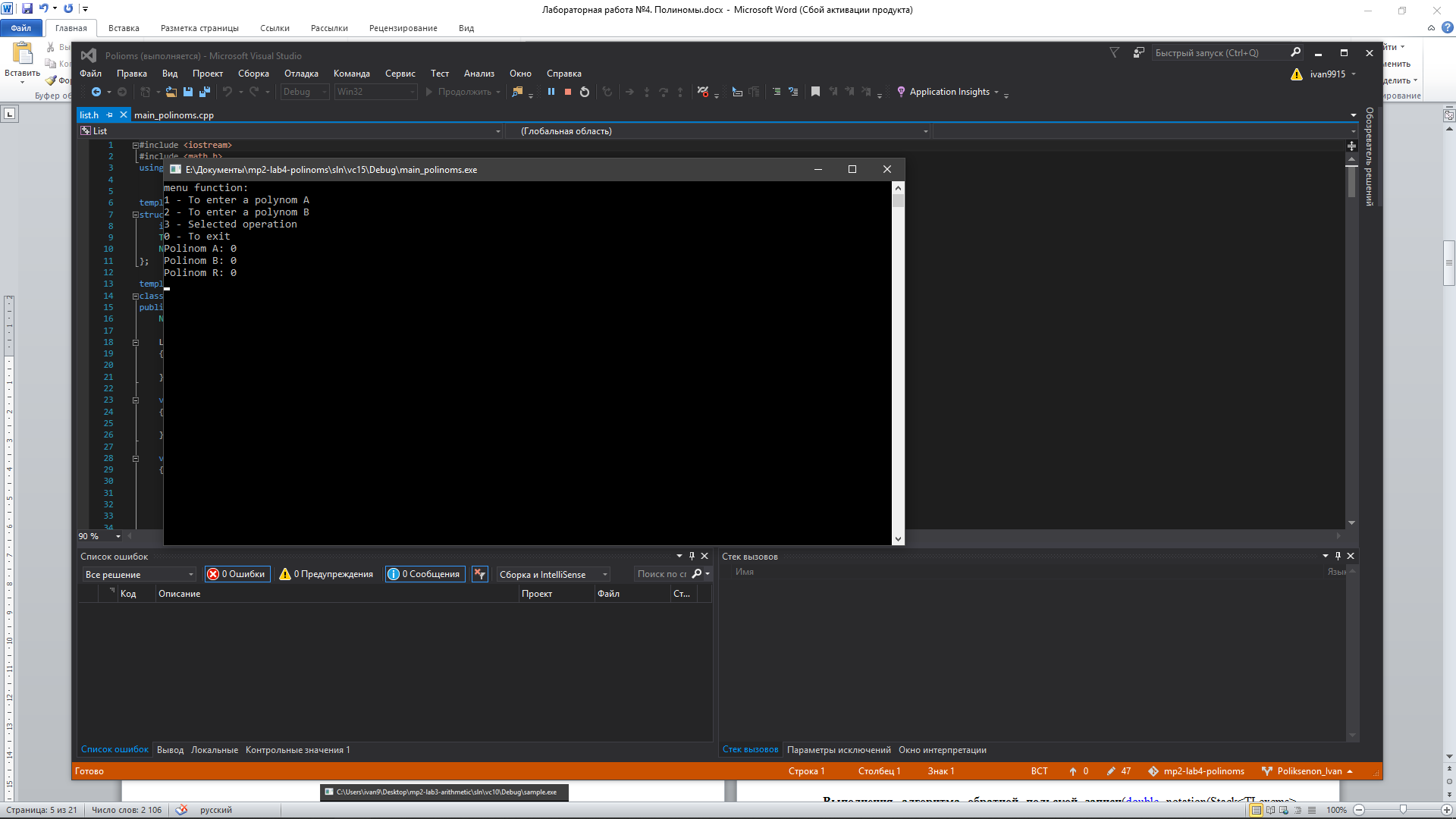
**Требуемый результат**

|  |  |
| --- | --- |
| R | Полином, реализованный в виде списка мономов, в который записывается результат выбранных действий. |

# Руководство пользователя

Для начала работы с программой необходимо открыть файл Polinoms.exe.

После запуска приложения на экране появится окно. Последует надпись, говорящая пользователю о необходимости выбора дальнейших действий. Пользователю необходимо ввести с клавиатуры число, соответствующее его выбору и нажать Enter. (см. рис. 1)



(рис.1)

Выбор очередного действия

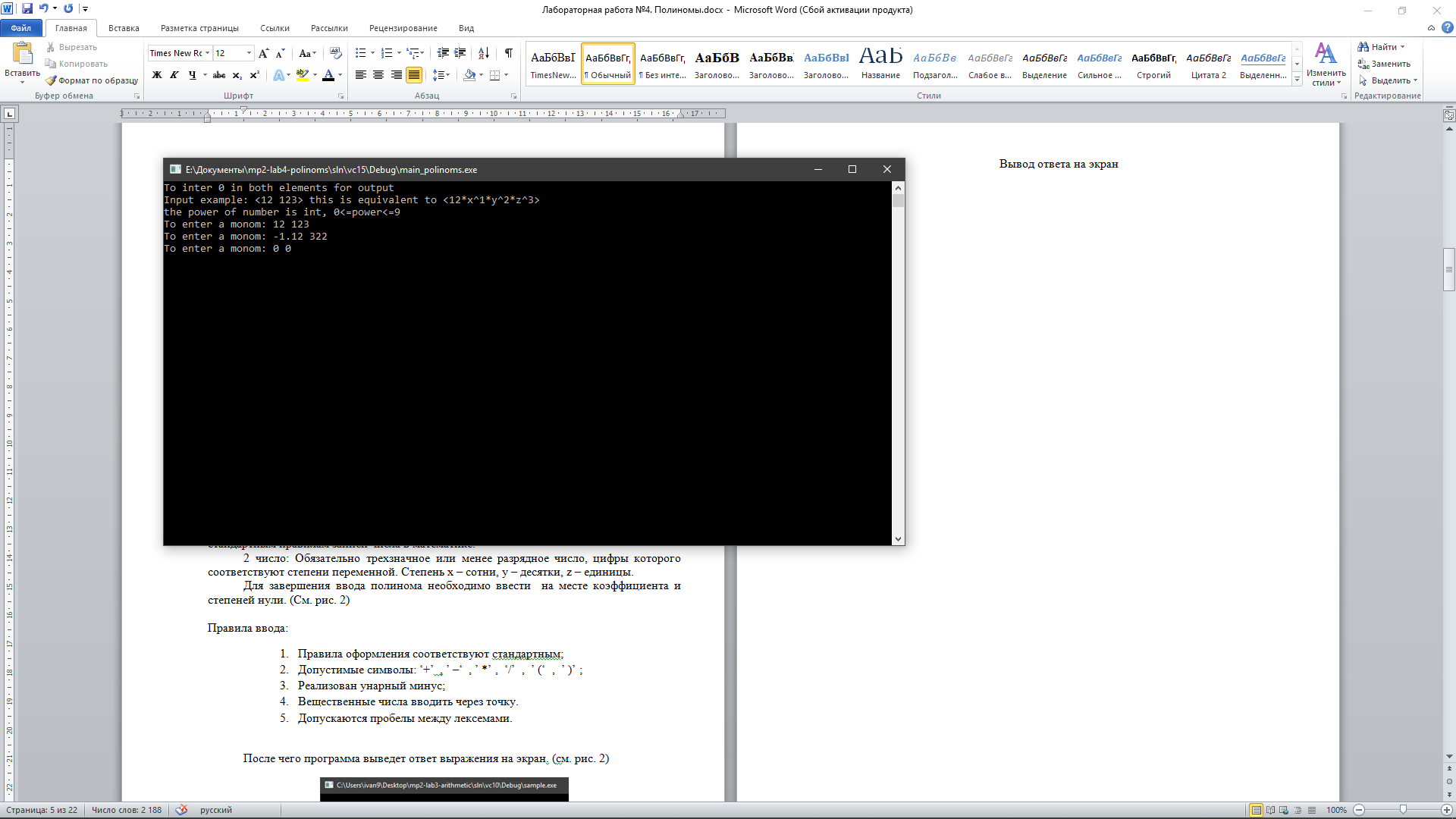
При выборе ввода полинома А или В:

После входа в меню ввода того или иного полинома пользователь должен заполнить свой многочлен заполняя каждый моном. Для заполнения мономы пользователю необходимо ввести 2 числа:

1 число: коэффициент перед мономом. Правила его ввода соответствуют стандартным правилам записи числа в математике.

2 число: Обязательно трехзначное или менее разрядное число, цифры которого соответствуют степени переменной. Степень х – сотни, y – десятки, z – единицы.

Для завершения ввода полинома необходимо ввести на месте коэффициента и степеней нули. (См. рис. 2)



(рис.2)

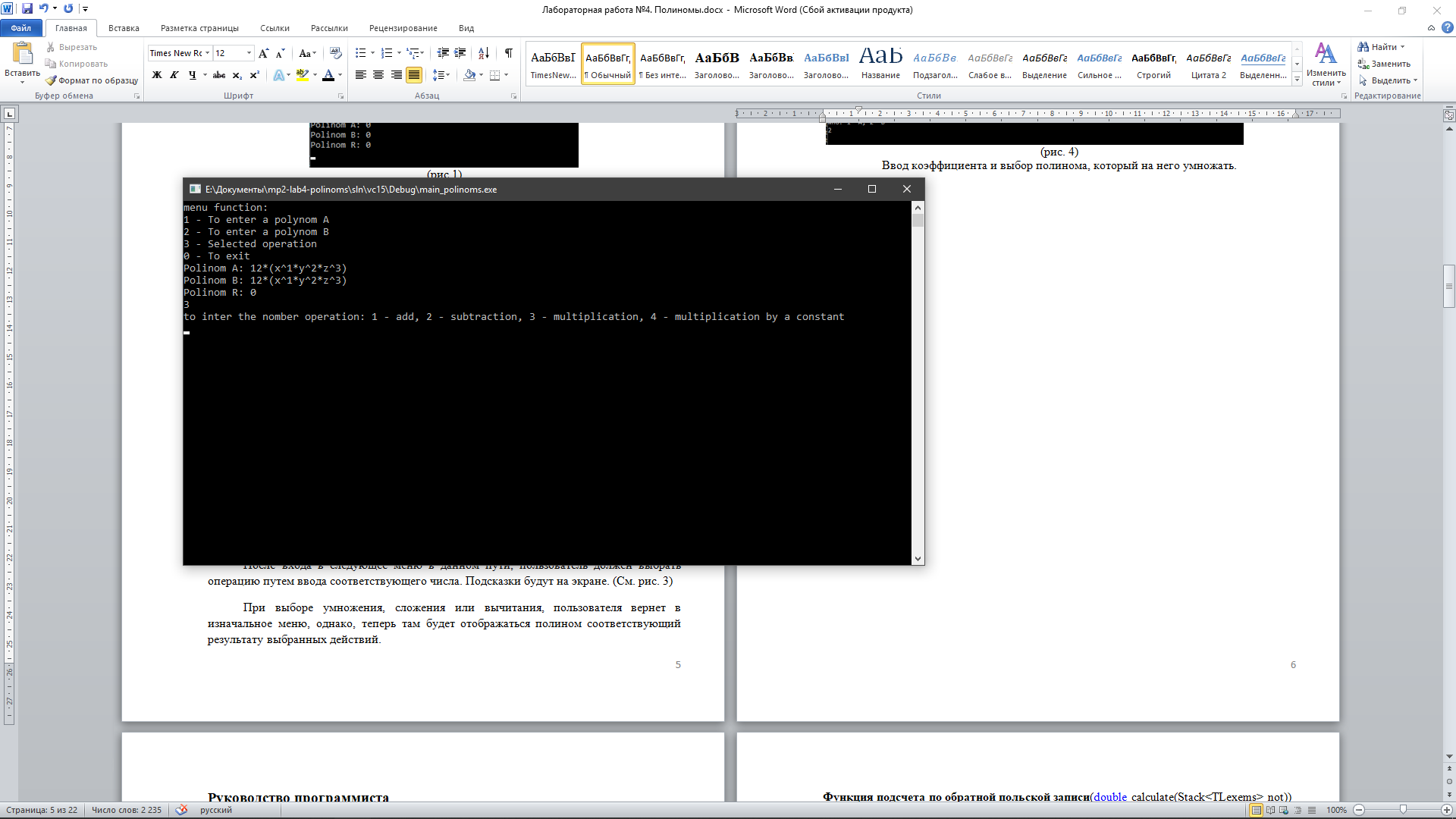
Ввод полинома путем заполнения мономов.

При входе в выбор операции над полиномами:

После входа в следующее меню в данном пути, пользователь должен выбрать операцию путем ввода соответствующего числа. Подсказки будут на экране. (См. рис. 3)

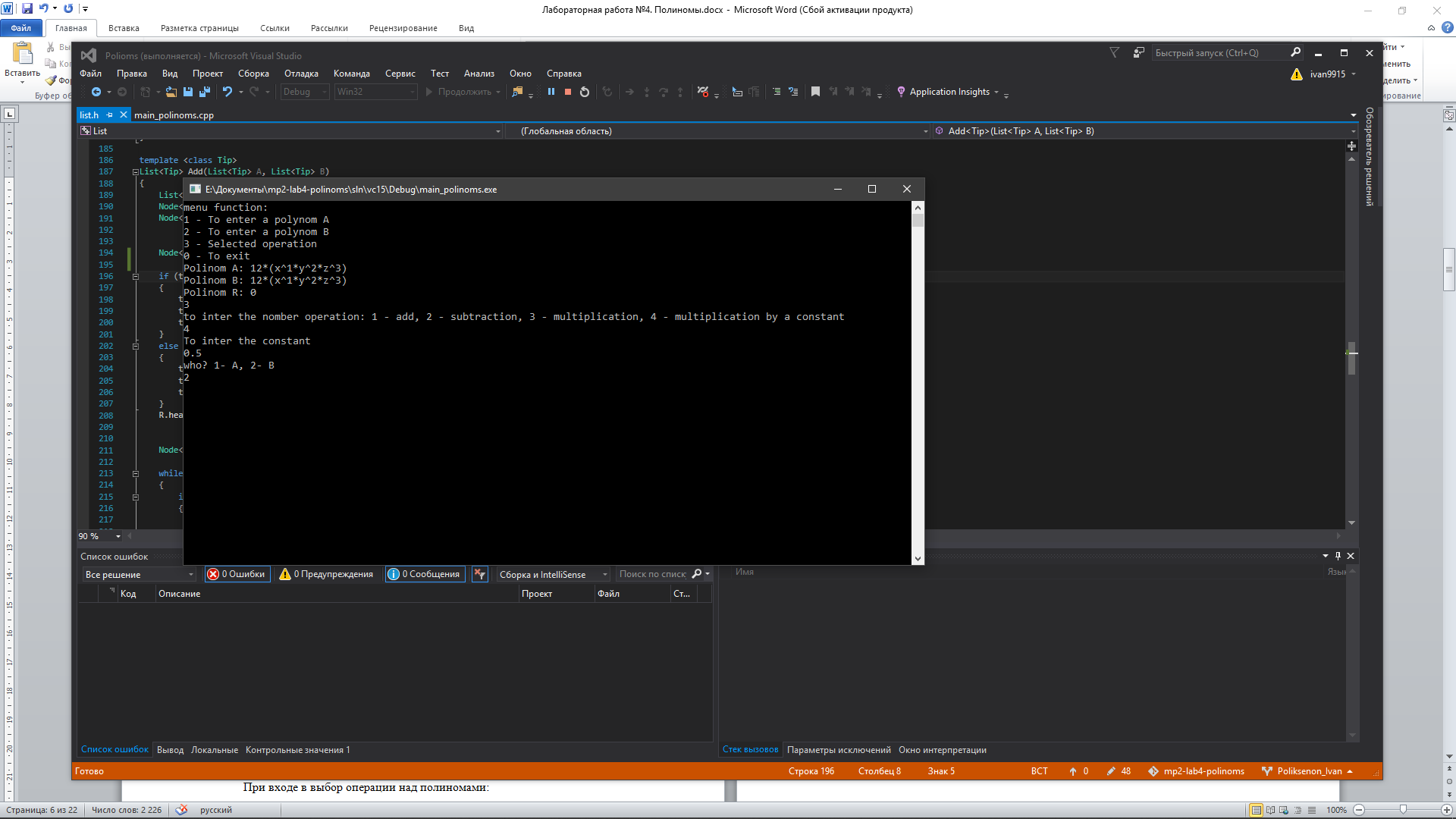
При выборе умножения, сложения или вычитания, пользователя вернет в изначальное меню, однако, теперь там будет отображаться полином соответствующий результату выбранных действий.

При выборе умножения на константу, пользовать должен ввести коэффициент. Правила его ввода соответствуют стандартным правилам записи числа в математике. А также выбрать полином, который необходимо умножить. (см. рис 4)



(рис. 3)

Выбор операции над полиномами.



(рис. 4)

Ввод коэффициента и выбор полинома, который на него умножать.

Для выхода из программы необходимо ввести 0.

# Руководство программиста

Решение поставленной задачи осуществляется с помощью следующих функций.

**Функция очистки мономов с нулевыми коэффициентами**

List<Tip>Clear(List<Tip>R)

Идя по списку, проверяет равенство коэффициента и нуля. Если таковые равенства обнаруживаются – удаляет их.

**Функция приведения подобных в отсортированном списке**

List<Tip> Turning(List<Tip> R)

Идет по списку и проверяет соседние элементы на соответствие степеней. Если таковые обнаруживаются – складывает коэффициенты в левый моном и удаляет правый.

После прохождения выполняет функцию удаления нулей, для случая, если таковые образовались.

**Функция сложения двух полиномов**

List<Tip> Add(List<Tip> A, List<Tip> B)

Сложения 2 полиномов реализовано с помощью алгоритма слияния. Проходит одновременно по двум спискам и записывает в результирующий список тот моном, у которого больше трехзначное число, характеризующее степень. При записывании такого монома двигает указатели в том, из которого взял, и в том, в который положил.

После того, как хотя бы в одном списке достигается край, просто приписывает в результат хвост от другого списка.

**Функция вычитания из первого полинома второго**

List<Tip> Mull(List<Tip> A, List<Tip> B)

Вычитание реализовано ка сложение первого со вторым, умноженным на -1.

**Функция умножения двух полиномов**

List<Tip> Mull(List<Tip> A, List<Tip> B)

Проходит по всем возможным парам мономов. Складывает их степени, умножает их коэффициенты и с помощью функции вставки размещает сформированный моном на нужное место в списке результата.

Также выполняется проверка на корректную степень.

**Функция эквивалентности двух мономов**

int IsEquel(List<Tip> A, List<Tip> B)

Сравнивает два списка по всем параметрам. Если нет различий – возвращает 1, иначе 0.

# Описание структур данных

**Новые типы:**

**struct Node**

**Переменные:**

int power – степень монома, записанная в трехзначном числе.

Tip coef – коэффициент перед мономом, указанного типа.

Node<Tip> \*next – указатель на структуру типа Tip.

**Class** **list** (реализует работу со списками).

**Переменные:**

Node<Tip> \*head – Указатель на начало данного списка.

**Общее описание структуры программного комплекса.**

Решение поставленной задачи осуществляется с помощью многофайловой программы.

1. **Заголовочный файл list.h**

В данном файле реализован следующий класс и методы:

template <class Tip>

class List {

**Поля:**

Node<Tip> \*head – Указатель на начало данного списка.

**Методы:**

1. ***Конструктор по умолчанию*** (List()).

Создает пустой список. Заполняет соответствующие параметры.

1. ***Очистка листа***(void ClrL()).

Передвигает указатель на последний элемент в начало. Также сбрасывает значение количества элементов.

1. ***Получение количества элементов списка*** (int get\_size()).
2. ***Печать данных списка в соответствии с данной задачей***(void print())
3. ***Удаление ячейки с указанной степенью***(void Delete(Tip pow))

Находит в списке ячейку, с теми же данными, что поступили на вход функции и удаляет его путем переопределения указателей.

1. ***Умножение всего списка на константу***(void Mull(double a))

Проходит по списку и в каждой ячейке изменяет коэффициент перед мономом в указанную величину.

1. ***Размещение элемента на место*** (void insert(int pow, Tip c)).

Создает ячейку (Node) изходя из заданных параметров и помещает ее на нужно место, путем прохождения списка и анализа существующих в нем элементов.

**4. Файл main\_arithmetic.cpp**

Основной файл программы. Реализует диалог с пользователем. Вызывает необходимые функции.

# Заключение

В данной лабораторной работе был изучен ряд известных алгоритмов, и был создан программный комплекс, реализующий следующие функциональные алгоритмы, касающиеся рассматриваемой темы:

1. Класс список.
2. Алгоритм сложения полиномов, записанных в виде списка мономов
3. Алгоритм умножения полиномов, записанных в виде списка мономов
4. Алгоритм вычитания полиномов, записанных в виде списка мономов
5. Ряд дополнительных функций, реализующих работу программы.
6. Реализованы тесты для проверки вышеуказанных алгоритмов

В ходе работы удалось достичь правильного подсчета выражений с определенным набором операций.

# Список литературы

* 1. Ознакомление с контейнером типа список и его основными алгоритмами [<https://codelessons.ru/cplusplus/spisok-list-v-s-polnyj-material.html>]

# Приложения

## Приложение 1. Класс list

template <class Tip>

struct Node {

int power;

Tip coef;

Node<Tip> \*next;

};

template <class Tip>

class List {

public:

Node<Tip> \*head;

List()

{

head = nullptr;

}

void ClrL()

{

head = nullptr;

}

void insert(int pow, Tip c)

{

if ((pow / 100>9) || ((pow / 10) % 10>9) || (pow % 10>9))

throw"The polynomial degree is higher than the permissible";

Node<Tip> \*tmp1 = new Node<Tip>;

tmp1->power = pow;

tmp1->coef = c;

Node<Tip> \*tmp2 = head;

if (head == nullptr)

{

tmp1->next = nullptr;

head = tmp1;

return;

}

if (tmp1->power == tmp2->power)

{

tmp2->coef += c;

return;

}

if (tmp1->power > tmp2->power)

{

tmp1->next = head;

head = tmp1;

return;

}

while (tmp2->next != nullptr)

{

if (tmp1->power == tmp2->next->power)

{

tmp2->next->coef += c;

return;

}

if ((tmp1->power < tmp2->power) && (tmp1->power > tmp2->next->power))

{

tmp1->next = tmp2->next;

tmp2->next = tmp1;

return;

}

tmp2 = tmp2->next;

}

tmp2->next = tmp1;

tmp1->next = nullptr;

}

void Delete(Tip pow)

{

Node<Tip> \*tmp = head;

if (head == nullptr)

return;

if (head->power == pow)

{

head = head->next;

return;

}

while (tmp->next != nullptr)

{

if (tmp->next->power == pow)

{

tmp->next = tmp->next->next;

break;

}

tmp = tmp->next;

}

}

void Mull(double a)

{

Node<Tip> \*tmp = head;

if (a == 0)

head = nullptr;

while (tmp != nullptr)

{

tmp->coef \*= a;

tmp = tmp->next;

}

}

void print()

{

if (head != nullptr)

{

Node<Tip> \*tmp = head;

cout << tmp->coef << "\*(x^" << tmp->power / 100 << "\*y^" << (tmp->power / 10) % 10 << "\*z^" << tmp->power % 10 << ")";

tmp = tmp->next;

while (tmp != nullptr)

{

cout << " + " << tmp->coef << "\*(x^" << tmp->power / 100 << "\*y^" << (tmp->power / 10) % 10 << "\*z^" << tmp->power % 10 << ")";

tmp = tmp->next;

}

}

else

cout << 0;

}

int get\_size()

{

Node<Tip> \*tmp = head;

int size = 0;

while (tmp != nullptr)

{

size++;

tmp = tmp->next;

}

return size;

}

};

## 

## Приложение 2. Файл list.cpp – Функции

template <class Tip>

List<Tip> Clear(List<Tip> R)//delete 0

{

Node<Tip> \*tmp3 = R.head;

if (tmp3->coef == 0)

R.Delete(tmp3->power);

while ((tmp3!=nullptr)&&(tmp3->next != nullptr))

{

if (tmp3->next->coef == 0)

R.Delete(tmp3->next->power);

tmp3 = tmp3->next;

}

return R;

}

template <class Tip>

List<Tip> Turning(List<Tip> R)

{

Node<Tip> \*tmp3 = R.head;

while (tmp3->next != nullptr)

if (tmp3->power == tmp3->next->power)

{

tmp3->coef += tmp3->next->coef;

if (tmp3->next->next != nullptr)

tmp3->next = tmp3->next->next;

else

tmp3->next = nullptr;

}

else

tmp3 = tmp3->next;

return Clear(R);

}

template <class Tip>

List<Tip> Add(List<Tip> A, List<Tip> B)

{

List<Tip> R;

Node<Tip> \*tmp1 = A.head;

Node<Tip> \*tmp2 = B.head;

Node<Tip> \*tmp = new Node<Tip>;

if (tmp1->power > tmp2->power)

{

tmp->coef = tmp1->coef;

tmp->power = tmp1->power;

tmp1 = tmp1->next;

}

else

{

tmp->coef = tmp2->coef;

tmp->power = tmp2->power;

tmp2 = tmp2->next;

}

R.head = tmp;

Node<Tip> \*tmp3 = R.head;

while ((tmp1 != nullptr) && (tmp2 != nullptr))

{

if (tmp1->power > tmp2->power)

{

tmp = new Node<Tip>;

tmp->coef = tmp1->coef;

tmp->power = tmp1->power;

tmp->next = nullptr;

tmp3->next = tmp;

tmp3 = tmp3->next;

tmp1 = tmp1->next;

}

else

{

tmp = new Node<Tip>;

tmp->coef = tmp2->coef;

tmp->power = tmp2->power;

tmp->next = nullptr;

tmp3->next = tmp;

tmp3 = tmp3->next;

tmp2 = tmp2->next;

}

}

while (tmp1 != nullptr)

{

tmp = new Node<Tip>;

tmp->coef = tmp1->coef;

tmp->power = tmp1->power;

tmp->next = nullptr;

tmp3->next = tmp;

tmp3 = tmp3->next;

tmp1 = tmp1->next;

}

while (tmp2 != nullptr)

{

Node<Tip> \*tmp = new Node<Tip>;

tmp->coef = tmp2->coef;

tmp->power = tmp2->power;

tmp->next = nullptr;

tmp3->next = tmp;

tmp3 = tmp3->next;

tmp2 = tmp2->next;

}

return Turning(R);

}

template <class Tip>

List<Tip> Sub(List<Tip> A, List<Tip> B)

{

List<Tip> R;

B.Mull(-1);

R = Add(A, B);

B.Mull(-1);

return R;

}

template <class Tip>

List<Tip> Mull(List<Tip> A, List<Tip> B)

{

List<Tip> R;

Node<Tip> \*tmp1 = A.head;

Node<Tip> \*tmp2 = B.head;

if ((A.head == nullptr) || (B.head == nullptr))

return R;

int pow = tmp1->power + tmp2->power;

if ((pow / 100>9) || ((pow / 10) % 10>9) || (pow % 10>9))

throw"The polynomial degree is higher than the permissible";

while (tmp1 != nullptr)

{

tmp2 = B.head;

while (tmp2 != nullptr)

{

R.insert(tmp1->power + tmp2->power, tmp1->coef\*tmp2->coef);

tmp2 = tmp2->next;

}

tmp1 = tmp1->next;

}

return R;

}

template <class Tip>

int IsEquel(List<Tip> A, List<Tip> B)

{

if (A.get\_size() != B.get\_size())

return 0;

Node<Tip> \*tmp1 = A.head;

Node<Tip> \*tmp2 = B.head;

while (tmp1 != nullptr)

{

if ((tmp1->coef != tmp2->coef) || (tmp1->power != tmp2->power))

return 0;

tmp1 = tmp1->next;

tmp2 = tmp2->next;

}

return 1;

}