МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №9**

**«Полиномы»**

**Выполнил:** студент группы 381706-2

Макарихин Семён Андреевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Руководитель:**

Ассистент кафедры МОСТ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2019

Содержание

[1.Введение 3](#_Toc1336360)

[2. Цели и задачи 4](#_Toc1336361)

[3. Руководство пользователя 5](#_Toc1336363)

[4. Руководство программиста 6](#_Toc1336364)

[4.1. Описание структуры программы 6](#_Toc1336365)

[4.2. Описание функций и процедур, их алгоритмов 6](#_Toc1336366)

[5. Эксперименты 11](#_Toc1336367)

[6. Заключение 12](#_Toc1336368)

[7. Литература 13](#_Toc1336369)

[8. Приложения 14](#_Toc1336370)

# 1.Введение

В математике, многочлены или полиномы от одной переменной - функции вида

8de2160610dbeafbceb3de7e48c4acdf где ci фиксированные коэффициенты, а x — переменная. Многочлены составляют один из важнейших классов элементарных функций.

Полиномиальные уравнения и их решение составляет едва ли не главный объект «классической алгебры». С изучением многочленов связан целый ряд преобразований в математике: введение в рассмотрение нуля, отрицательных, а затем и комплексных чисел, а также появление теории групп как раздела математики и выделение классов специальных функций в анализе.

Многочлены также играют ключевую роль в алгебраической геометрии, объектом которой являются множества, определённые как решения систем многочленов. Особые свойства преобразования коэффициентов при умножении многочленов используются в алгебраической геометрии, алгебре, теории узлов и других разделах математики для кодирования, или выражения многочленами свойств различных объектов.

Лабораторная работа направлена на изучение методов компьютерной обработки полиномов. Основной учебной целью работы является практическое освоение методов организации структур хранения данных с помощью линейных списков.

# 2. Цели и задачи

Для работы с полиномами предлагается реализовать следующие операции:

* конструкторы инициализации и копирования;
* метод присваивания;
* метод сложения полиномов.

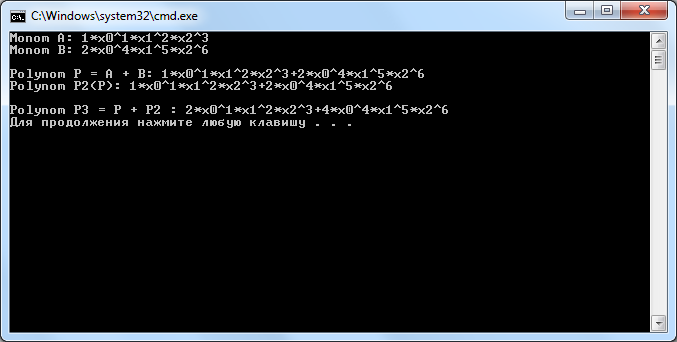
Дополнительные операции могут быть определены при разработке спецификации класса для полиномов.

В процессе выполнения лабораторной работы требуется использовать систему контроля версий [Git](https://git-scm.com/book/ru/v2). Выполнение работы предполагает решение следующих задач:

1. Реализация класса TMonomial.
2. Реализация класса TPolynomial, построенного с использованием класса TMonomial.
3. Реализация класса для обработки исключений– TException, которые могут возникнуть при выполнении различных операций, согласно заданному интерфейсу.
4. Обеспечение работоспособности тестов и примера использования.
5. Реализация заготовок тестов, покрывающих все методы созданных классов.
6. Модификация примера использования в тестовое приложение, позволяющее задавать полиномы и осуществлять основные операции над ними.

# 3. Руководство пользователя

Запускаем программу polynomial.cpp. Программа выведет пример корректной работы некоторых основных операций с мономами и полиномами.



# 

# 4. Руководство программиста

Разработка системы вычисления проводились в среде “Microsoft Visual Studio 2010”.

В данной работе будет использовано 3 класса:

* Класс «Моном» (TMonomial).
* Класс «Полином» (TPolynomial).
* Класс исключения (TExсeption).

## 4.1. Описание структуры программы

Модульная структура программы:

1. tmonomial.h– модуль с классом TMonomial, в котором определен интерфейс класса Моном и реализация его методов.
2. tpolynomial.h– модуль с классом TPolynomial, в котором определен интерфейс класса Полином и реализация его методов.
3. exсeption.h – модуль с классом исключения TExсeption.
4. sample\_tpolynomial.cpp, sample\_performance\_check.cpp– модуль программы тестирования, с которым работает пользователь, в котором проводятся эксперименты.
5. test\_main.cpp, test\_tmonomial\_and\_tpolynomial.cpp – модуль с функциями тестирования для созданных классов. Содержат 23 теста для класса TMonomial и 17 тестов для класса TPolynomial.

## 4.2. Описание функций и процедур, их алгоритмов

Рассмотрим реализацию методов класса TMonomial:

class TMonomial

int\* power - массив степеней переменных монома.

int n - количество переменных в мономе(размерность).

double coeff - коэффициент при мономе.

TMonomial \*next – указатель на следующий моном.

1. Monomial( int \_n = 10 ) - конструктор класса, принимающий количество переменных в мономе. По умолчанию создается моном размера 10.

2. TMonomial( int \_n, int \*\_power, double \_coeff ) - конструктор класса, принимающий количество переменных в мономе, массив степеней переменных монома и коэффициент.

3. TMonomial(const TMonomial &A ) - конструктор копирования.

Принимает ссылку на объект класса TMonomial.

4. ~TMonomial() - деструктор. Освобождает выделенную под моном память.

5. void SetPower( int \*\_power ) – метод установления массива степеней монома.

6. void SetN( int \_n ) – метод установления кол-ва переменных монома.

7. void SetCoeff( double \_coeff ) – метод установления коэффициента при мономе.

8. void SetNext( TMonomial\* \_next ) – метод установления адреса следующего монома.

9. int \*GetPower() - метод получения массива степеней монома.

10. int GetN() const - метод получения кол-ва переменных монома.

11. double GetCoeff() const - метод получения коэффициента при мономе.

12. TMonomial\* GetNext() - метод получения адреса следующего монома.

13. bool ComparePowers( const TMonomial &A) – метод сравнения массива степеней монома.

14. TMonomial &operator = ( const TMonomial &A ) - перегрузка оператора присваивания одного монома другому.

Присваивает полям первого объекта класса поля второго объекта класса.

15. TMonomial operator + ( TMonomial &A ) - перегрузка оператора сложения мономов.

16. TMonomial &operator += ( const TMonomial &A ) - перегрузка оператора “+=” для мономов.

17. TMonomial operator - ( TMonomial &A ) - перегрузка оператора вычитания одного монома из другого.

18. TMonomial &operator -= ( const TMonomial &A ) - перегрузка оператора “-=” для мономов.

19. TMonomial operator \* ( TMonomial &A ) - перегрузка оператора умножения мономов.

20. TMonomial &operator \*= ( TMonomial &A ) - перегрузка оператора “\*=” для мономов.

21. bool operator == ( TMonomial &A ) - перегрузка оператора сравнения.

Выполняется проверка мономов на равенство. Возвращает true, если равенство выполняется, false в противном случае.

22. bool operator < ( TMonomial &A ) - перегрузка оператора “<” для мономов.

23. bool operator > ( TMonomial &A ) - перегрузка оператора “>” для мономов.

24. friend istream& operator>>( istream &istr, TMonomial &m ) - дружественный метод. Перегрузка оператора ввода мономов.

25. friend ostream& operator<<( ostream &ostr, TMonomial &m ) - дружественный метод. Перегрузка оператора вывода мономов.

Рассмотрим реализацию методов класса TPolynomial:

class TPolynomial

TMonomial \*start - указатель на первый моном в полиноме.

int n – поле размерности монома в полиноме.

int k – количество мономов в полиноме.

1. TPolynomial() – конструктор класса без параметров.

2. TPolynomial(int \_n) - конструктор класса, принимающий количество переменных в мономе.

3. TPolynomial(const TPolynomial &A) - конструктор копирования.

Принимает ссылку на объект класса TPolynomial.

4. ~TPolynomial() - деструктор. Освобождает выделенную под полином память.

5. int GetN() - метод получения кол-ва переменных монома.

6. int GetSize() – метод получения кол-ва мономов в полиноме.

7. TMonomial\* GetStart() - метод получения указателя на первый моном в полиноме.

8. TPolynomial operator-(TPolynomial &A) - перегрузка оператора вычитания полиномов.

9. TPolynomial operator+(TPolynomial &A) - перегрузка оператора сложения полиномов.

10. TPolynomial& operator=(const TPolynomial &A) - перегрузка оператора присваивания одного полинома другому.

Присваивает полям первого объекта класса поля второго объекта класса.

11. bool operator==(const TPolynomial &A) - перегрузка оператора сравнения.

Выполняется проверка полиномов на равенство. Возвращает true, если равенство выполняется, false в противном случае.

12. bool operator!=(const TPolynomial &A) - перегрузка оператора сравнения.

Выполняется проверка полиномов на неравенство. Возвращает true, если неравенство выполняется, false в противном случае.

13. TPolynomial operator\*(TPolynomial &A) - перегрузка оператора умножение полиномов.

14. TPolynomial& operator+=( const TMonomial &m) - перегрузка оператора “+=” для сложения полинома с мономом.

15. TPolynomial& operator-=( const TMonomial &m) - перегрузка оператора “-=” для вычитания из полинома монома.

16. friend std::ostream& operator<<(std::ostream &ostr, TPolynomial& Tm) - дружественный метод. Перегрузка оператора вывода полиномов.

# 5. Эксперименты

В качестве примера рассмотрим перегрузку оператора “+=” для сложения полинома с мономом для класса полинома (TPolynomial).

Теоретическая сложность выполнения алгоритма O(1).

Мы провели измерение прибавляя к полиному мономы с разными кол-вами переменных: 10, 100, …, 1000000 переменных. Ниже вы можете увидеть график зависимости времени выполнения операции “+=” от количества переменных монома.

По приведенным данным можно сделать вывод, что практическая сложность выполнения алгоритма равна теоретической.

По горизонтали – количество переменных монома.

По вертикали - время выполнения программы.

**6. Заключение**

В результате лабораторной работы был разработан класс полиномов, а также освоены такие инструменты разработки программного обеспечения, как система контроля версий [Git](https://git-scm.com/book/ru/v2) и фрэймворк для разработки автоматических тестов [Google Test](https://github.com/google/googletest).

Созданный класс был протестированы с использованием Google Tests, а также были проведены эксперименты для сравнения теоретической и практической сложности выполнения операций на методе класса.

# 7. Литература

1. Лабораторный практикум: Учебно-методическое пособие / Мееров И.Б. [и др.] – Нижний Новгород: Нижегородский госуниверситет , 2017. – 105с.
2. Касперски К. Техника оптимизации программ. Эффективное использование памяти. — СПб.: БХВ-Петербург, 2003. — 464 с.: ил.
3. <https://ru.wikipedia.org/wiki/Многочлен>

# 8. Приложение