МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

Направление подготовки «Прикладная математика и информатика»

**ОТЧЕТ**

по лабораторной работе

**Просматриваемые, упорядоченные и хэш таблицы.**

**Выполнила:** студентка группы 381603-1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.В.Жилина

**Проверила:** ст.преп., к.т.н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.Д.Кустикова

Нижний Новгород  
2017

Содержание

Введение 3

1 Постановка задачи 4

2 Руководство пользователя 5

3 Руководство программиста 6

3.1 Структура программы 6

3.2 Структуры данных 6

3.3 Программная реализация 6

3.3.1 Виртуальный класс Table 6

3.3.2 Класс ScanTable 6

3.3.3 SortedTable 7

3.3.4 Класс HashTable 7

Заключение 9

Литература 10

Приложение 11

Приложение А. Программная реализация виртуального класса таблиц 11

Приложение Б. Программная реализация просматриваемых таблиц 11

Приложение В. Программная реализация упорядоченных таблиц 11

Приложение Г. Программная реализация хэш таблиц 11

Приложение Д. Основная программа 11

# Введение

# Постановка задачи

Цель:

Контрольный пример:

Введенное пользователем выражение(рис.1) и полученный результат выражения введенного пользователем(рис.2):

|  |  |
| --- | --- |
|  |  |
| Рис.1. Введенное выражение | Рис.2. Полученный результат |

# Руководство пользователя

Для начала работы необходимо запустить файл [sample.exe](../build/lab1_polinom/Lab1_polinom.sln). На экране появится меню(рис.3), при выборе пункта 1-4, появится надпись с просьбой ввести выражение (рис 4.), если оно введено верно появится результат. При выборе пункта 5-выход из программы.

|  |  |
| --- | --- |
|  |  |
| Рис.3. Меню программы | Рис.3. Просьба о вводе полинома и константы |

# Руководство программиста

## Структура программы

Программа состоит из 4 проектов:

1. gtest

Содержит в себе [gtest.h](../gtest/gtest.h) и [gtest-all.cpp](../gtest/gtest-all.cc). Файлы содержат google тесты, которые необходимы для проверки корректности работы реализованных классов.

1. polinom\_lib

Содержит в себе:

[Node.h](file:///C:\Users\user\Desktop\mp2-practice-2\ZhilinaL\Lab1\include\Node.h)-заголовочный файл, содержащий объявление шаблонного класса Node и его реализацию.

[Monom.h](file:///C:\Users\user\Desktop\mp2-practice-2\ZhilinaL\Lab1\include\Monom.h)- заголовочный файл, содержащий объявление шаблонного класса monom.

[Polinom.h](file:///C:\Users\user\Desktop\mp2-practice-2\ZhilinaL\Lab1\include\Polinom.h)- заголовочный файл, содержащий объявление шаблонного класса Polinom

[List.h](file:///C:\Users\user\Desktop\mp2-practice-2\ZhilinaL\Lab1\include\List.h)-заголовочный файл, содержащий объявление шаблонного класса list и его реализацию.

[Monom.cpp](file:///C:\Users\user\Desktop\mp2-practice-2\ZhilinaL\Lab1\src\Monom.cpp)-содержит реализацию класса monom

[Polinom.cpp](file:///C:\Users\user\Desktop\mp2-practice-2\ZhilinaL\Lab1\src\Polinom.cpp)-содержит реализацию класса Polinom.

1. polinom\_test

Содержит в себе [Test\_main.cpp](../test/Test_main.cpp), [test\_polinoms.cpp](../test/test_polinoms.cpp), [test\_list.cpp](../test/test_list.cpp). Реализация тестов для проверки корректности работы реализованных классов.

1. sample

Содержит [Main.cpp](file:///C:\Users\user\Desktop\mp2-practice-2\ZhilinaL\Lab1\samples\Main.cpp)– реализация пользовательского интерфейса.

## Структуры данных

1.В качестве структуры хранения полинома используется циклический список с головой. Звенья списка – мономы, однозначно определяемые через коэффициент и обобщенную степень. Голова – фиктивное звено, не содержащее монома. Элементы списка хранятся в порядке возрастания обобщённых степеней.

Для его реализации был написан шаблонный класс Node-звено списка. Сам список представляет собой структуру с двумя полями: указатель на голову звена и указатель на текущее звено.

## Программная реализация

### Виртуальный класс Table

Поля:

T data – данные в звене

Node\* next - указатель на следующее звено

Методы:

Node() { next = nullptr; } -конструктор по умолчанию-создает пустое звено

Node(T Data) { data = Data; next = nullptr; }-конструктор с параметром-данные звена

bool operator< (const Node& z) const { return (data < z.data); }-перегрузка операции меньше, сравнивает данные звеньев

bool operator> (const Node& z) const { return (data > z.data); }-перегрузка операции больше, сравнивает данные звеньев

### Класс ScanTable

Поля:

double coeff-коэффициент монома

unsigned int abc - степень монома, x^a,y^b,z^c

Методы:

monom(double c = 0, unsigned int a = 0) - конструктор с параметрами

monom& operator=(const monom& m)- перегрузка оператора присваивания

bool operator< (const monom& m) const -перегрузка оператора меньше, сравнивает общую степень и коэффициент

bool operator> (const monom& m) const -перегрузка оператора больше, сравнивает общую степень и коэффициент

Перегрузка операторов сравнения:

bool operator==(const monom& m) const

bool operator!=(const monom& m) const

### SortedTable

Поля:

Node<T>\* head - указатель на голову списка-фиктивное звено

Node<T>\* cur - указатель на текущий элемент

Методы:

void Clean() - очистка списка

list() - конструктор по умолчанию

list(const list<T>& a) - конструктор копирования

~list() - деструктор

list<T>& operator=(const list<T> &a) - перегрузка оператора присваивания

void Insert(T elem) - вставка элемента в упорядоченный список

void InsertToTail(T elem) - вставка в конец списка

void InsertAfter(Node<T>\* N, T Data)- вставка после определенного звена

Перегрузка операторов сравнения:

bool operator==(const list<T>& sp) const

bool operator!=(const list<T>& sp) const

Методы навигации:

void Reset() - в начало списка

void Step() - шаг на одно звено вперед

Node<T>\* GetCur() const - получить текущий элемент

bool IsNotOver() const - проверка на окончание списка

### Класс HashTable

Поля:

list<monom> Plist - список из мономов

Методы:

list<monom> similar\_terms(list <monom> sp) - приведение подобных в полиноме

Polinom(const string pol = "") - разбор строки

Polinom(const Polinom& pol) - конструктор копирования

Polinom& operator=(const Polinom &pol)- перегрузка оператора присваивания

Polinom operator+(const Polinom& pol) const - перегрузка оператора сложения

Polinom operator\*(const double a) const - перегрузка оператора умножения с параметром-константа (умножение на константу слева)

friend Polinom operator\*(const double a, const Polinom& pol) - умножение на константу справа

Polinom operator-(const Polinom& pol) const - перегрузка оператора вычитания

Polinom operator\*(const Polinom& pol) const - перегрузка оператора умножения с параметром-полином (умножение полиномов)

Polinom operator-() const -унарный минус

friend ostream& operator<<(ostream &ostr, const Polinom &pol) - перегрузка оператора вывода

Polinom(list<monom> &list) : Plist(list) {} - конструктор по списку

Перегрузка операторов сравнения:

bool operator==(const Polinom& pol) const

bool operator!=(const Polinom& pol) const

# Заключение

# Литература

1. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ, Москва 2013
2. Гергель В.П. Рабочие материалы к учебному курсу «Методы программирования». 2002 г.

# Приложение

## Приложение А. Программная реализация виртуального класса таблиц

Table.h

## Приложение Б. Программная реализация просматриваемых таблиц

ScanTable.h

ScanTable.cpp

## Приложение В. Программная реализация упорядоченных таблиц

SortedTable.h

SortedTable.cpp

## Приложение Г. Программная реализация хэш таблиц

HashTable.h

HashTable.cpp

## Приложение Д. Основная программа