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

**Таблицы**

Выполнил:

 студент гр. 381703-1

Громов Н.В.

Проверил:

ассистент кафедры

МОСТ ИИТММ

 Волокитин В.Д.

г. Нижний Новгород

2018г

Оглавление

[Введение 3](#_Toc530768225)

[Постановка учебно-практической задачи 4](#_Toc530768226)

[Руководство пользователя 5](#_Toc530768227)

[Руководство программиста 7](#_Toc530768228)

[Описание структур данных 8](#_Toc530768229)

[Заключение 11](#_Toc530768230)

[Список литературы 12](#_Toc530768231)

[Приложения 13](#_Toc530768232)

# Введение

Таблицы являются удобной формой для отображения информации. Но таблицы выполняют лишь тогда свою цель, когда между строчками и столбцами имеется смысловая связь, то есть информацию в них можно рассортировать неким образом, например, по дате или алфавиту.

С другой стороны, таблицы сложнее обычного текста. Так что применять их имеет смысл лишь там, где они действительно улучшают восприятие материала.

# Постановка учебно-практической задачи

**Цель работы:**

Разработать программу, которая предоставляет возможность хранить данные в таблицах 3-х типов:

1)Неупорядоченная таблицы

2)Упорядоченная таблица

3)Хэш-таблица

**Последовательность выполнения работы:**

1. Разработка классов для таблиц.
2. Обеспечение логирования количества произведенных операций.
3. Проверка работы методов классов при помощи Google Test-ов.
4. Создание консольного интерфейса пользователя.

**Исходные данные:**

N – количество полиномов в таблице.

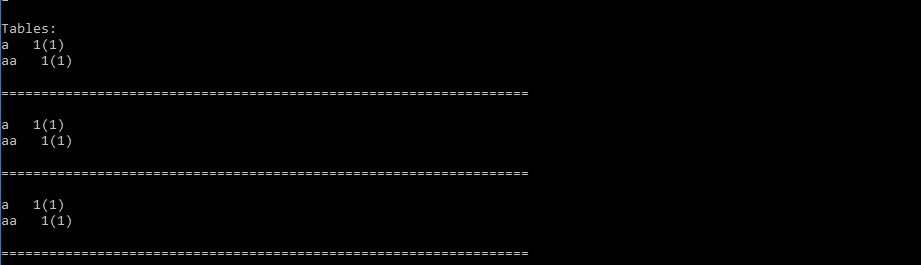
P – имя полинома.

С – сам полином.

**Требуемый результат:**

Три таблицы.

**Пример:**



# Руководство пользователя

Для начала работы с программой необходимо открыть файл main.exe.После запуска приложения на экране появится окно. Появится просьба ввода данных для таблиц (см. рис. 1)

.

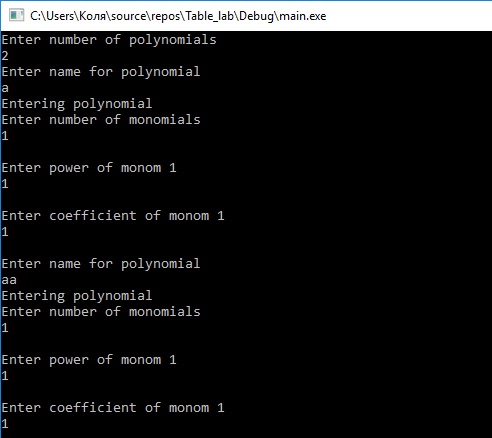


рис.1.Ввод данных

После ввода данных показываются таблицы и доступные с ними операции (см. рис. 2).

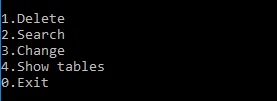


рис. 2. Доступные операции.

После произведения операции на экран выводятся редактированные таблицы (см. рис. 3).

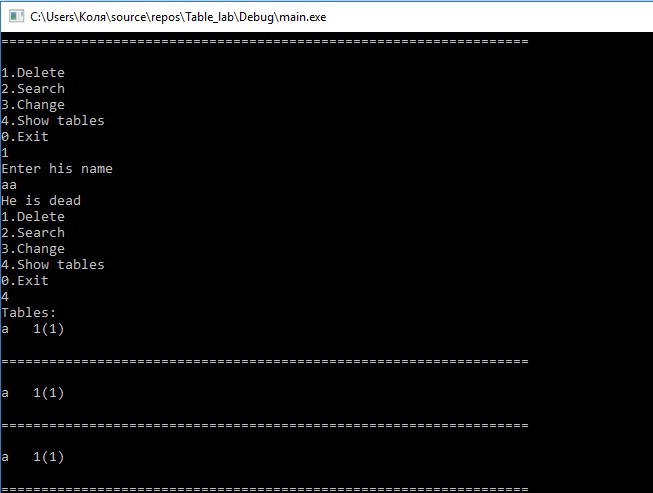


рис.3.Удаление.

Также можно просматривать результаты логирования в файле Log.txt (см. рис.4)

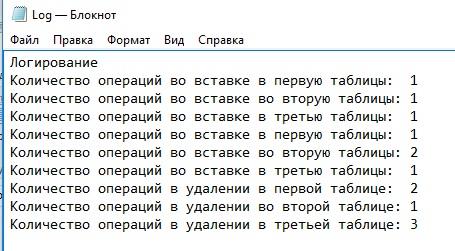


рис.4. Логирование.

# Руководство программиста

Решение поставленной задачи осуществляется с помощью следующих функций, структур и классов.

**Структура полином-имя(struct poly):**

**Конструктор по умолчанию(tablei( )).**

**Конструктор (poly(pol b, string s)).**

**Перегрузка оператора ==(bool operator==( poly b)).**

**Класс неупорядоченной таблицы(class table1).**

**Класс упорядоченной таблицы(class table2)(наследуется от table1).**

**Класс хеш-таблицы(class table3).**

**У них есть общие методы:**

**Конструктор по умолчанию(tablei( )).**

**Функция вставки(void insert(pol b, string s)).**

**Функция удаления(void delete( string b)).**

**Функция поиска (void search( string b)).**

**Функция печати(void Print( )).**

# 

# Описание структур данных

**Новые структуры:**

**stuct poly**(содержит имя и полином).

**class** **table1** (Класс неупорядоченной таблицы).

**class** **table2** (Класс упорядоченной таблицы).

**class** **table1** (Класс хеш-таблицы).

**Переменные:**

const int maxsize=256; - максимальный размер таблиц.

int r, j, k, p; - счетчики

double c; - коэффициент монома в полиномах

List a; - список для полинома

string s; - имя для полинома

table1 t1; - неупорядоченная таблица

table2 t2;- упорядоченная таблица

table3 t3;- хеш-таблица

ofstream out; - переменная для вывода результатов логирования в файл

**Общее описание структуры программного комплекса.**

Решение поставленной задачи осуществляется с помощью многофайловой программы.

**Заголовочный файл pol.h – ресурс для полиномов, реализованный в предыдущей лабораторной работе.**

**Заголовочный файл table.h**

В данном файле представлены имена классов, полей и методов.

**stuct poly**

**Поля:**

pol p - полином

string name – имя полинома

**class table1**

**Поля:**

poly a[maxsize] – массив для таблицы

int counter – счетчик текущих записей

int counterI – счетчик операций для вставки

int counterD – счетчик операций для удаления

int counterS – счетчик операций для поиска

**Методы:**

**Функция вставки(void insert(pol b, string s)).**

pol b – встявляемый полином

string s – его имя

**Функция удаления(void delete( string b)).**

string b – его имя

**Функция поиска (void search( string b)).**

string b – его имя

**Файл main.cpp**

Реализован диалог с пользователем. Переменные описаны в структуре данных.

# Заключение

В данной лабораторной работе был изучен ряд известных алгоритмов, и был создан программный комплекс, реализующий такие функциональные алгоритмы, как: алгоритм хранения полиномов в виде таблиц. Так же реализован набор гугл тестов, проверяющих правильность программы.

# Список литературы

Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. Quicksort // Алгоритмы: построение и анализ / Под ред. И. В. Красикова. 3-е изд., Пер. с англ. —М.: Издательский дом «Вильямс», 2013.

# Приложения

## Приложение 1. Файл table.h

#ifndef \_TABLE\_H\_

#define \_TABLE\_H\_

#include "pol.h"

#include <list>

#include <algorithm>

#include <string>

#include <iterator>

using namespace std;

const int maxsize = 256;

struct poly

{

pol p;

string name;

public:

poly() {};

poly(pol b, string s);

bool operator==(poly b);

};

class table1

{

protected:

poly a[maxsize];

int counter;

public:

int counterS;

int counterI;

int counterD;

table1() { counter = 0; counterS = 0; counterD = 0;

counterI = 0;

};

void insert(pol b, string s);

pol search(string b);

void Delete(string b);

void Print();

poly\* get\_vec() { return a; }

};

class table2:public table1

{

public:

void insert(pol b, string s);

pol search(string b);

int get\_i\_search(string s);

void Delete(string b);

};

class table3

{

list<poly> a[maxsize];

int n;

public:

int counterS;

int counterD;

int counterI;

table3();

void insert(pol b, string s);

pol search(string b);

void Delete(string b);

void Print();

};

#endif

## Приложение 2. Файл table.cpp

#include "table.h"

using namespace std;

poly::poly(pol b, string s)

{

p = b;

name = s;

}

bool poly::operator==(poly b)

{

if ((p == b.p) && (name == b.name))

return 1;

return 0;

}

void table1::insert(pol b, string s)

{

poly c(b, s);

counterI = 0;

counterI++;

if (counter + 1 > maxsize)

throw "Not enough space";

a[counter++]=c;

}

void table1::Delete(string s)

{

counterD = 0;

poly c(search(s), s);

for (int i = 0; i < counter; i++,counterD++)

if (a[i] == c)

{

a[i] = a[counter--];

}

}

pol table1::search(string s)

{

counterS = 0;

for (poly n : a)

{

counterS++;

if (n.name == s)

return n.p;

}

throw "Not found";

}

void table1::Print()

{

if (!counter)

throw "table is empty";

for (int i=0;i<counter;i++)

{

cout << a[i].name << " ";

a[i].p.Print();

cout << endl;

}

}

void table2::insert(pol b, string s)

{

int min,i,k;

k = 0;

counterI = 0;

counterI++;

if (counter + 1 > maxsize)

throw "Not enough space";

poly c(b, s);

for (int j = 0; j<counter; j++)

{

i = 0;

min = (s.length() > a[j].name.length()) ? a[j].name.length() : s.length();

while ((s[i] == a[j].name[i]) && (i < min))

i++;

if (i == min)

{

counterI++;

if (s.length() <= a[j].name.length())

{

for (int k = counter - 1; k >= j; k--)

a[k + 1] = a[k];

a[j]= c;

counter++;

counterI++;

return;

}

}

else

{

counterI++;

if (s[i] < a[j].name[i])

{

for (int k = counter - 1; k >= j; k--)

a[k + 1] = a[k];

a[j] = c;

counter++;

return;

}

}

}

a[counter++] = c;

}

pol table2::search(string s)

{

int i,l = 0;

int r = counter;

int mid,min;

counterS = 0;

while (l <= r) {

counterS++;

mid = (l + r) / 2;

if (a[mid].name == s)

return(a[mid].p);

counterS += 2;

i = 0;

min = (s.length() > a[mid].name.length()) ? a[mid].name.length() : s.length();

while ((s[i] == a[mid].name[i]) && (i < min))

i++;

if (i == min)

{

if (s.length() < a[mid].name.length())

r = mid - 1;

else l = mid + 1;

}

else

{

if (s[i] < a[mid].name[i])

r = mid - 1;

else l = mid + 1;

}

}

throw "is out";

}

int table2::get\_i\_search(string s)

{

int i, l = 0;

int r = counter;

int mid, min;

counterD = 0;

while (l <= r) {

mid = (l + r) / 2;

counterD++;

if (a[mid].name == s)

return mid;

counterD += 2;

i = 0;

min = (s.length() > a[mid].name.length()) ? a[mid].name.length() : s.length();

while ((s[i] == a[mid].name[i]) && (i < min))

i++;

if (i == min)

{

if (s.length() < a[mid].name.length())

r = mid - 1;

else l = mid + 1;

}

else

{

if (s[i] < a[mid].name[i])

r = mid - 1;

else l = mid + 1;

}

}

}

void table2::Delete(string s)

{

int k = get\_i\_search(s);

for (int i = k; i < counter-1; i++)

a[i] = a[i+1];

counter--;

}

table3::table3()

{

n = 0;

counterS = 0;

counterD = 0;

counterI = 0;

}

void table3::insert(pol b, string s)

{

int key=0;

poly c(b, s);

counterI = 0;

counterI++;

if (n + 1 > maxsize)

throw "Not enough space";

for (int i = 0; i < s.length(); i++)

key += int(s[i]);

key %= 256;

a[key].push\_front(c);

n++;

}

pol table3::search(string s)

{

int key = 0;

counterS = 0;

for (int i = 0; i < s.length(); i++)

key += int(s[i]);

key = key % 256;

counterS++;

if (!a[key].empty())

for (poly n : a[key])

{

counterS++;

if (n.name == s)

return n.p;

}

throw "Not found";

}

void table3::Delete(string s)

{

int key = 0;

counterD = 1;

if (n - 1 < 0)

throw "Table is empty";

for (int i = 0; i < s.length(); i++)

key += int(s[i]);

key = key % 256;

counterD++;

if (!a[key].empty())

for (poly n : a[key])

{

counterD++;

if (n.name == s)

{

poly c(n.p, s);

a[key].remove(c);

break;

}

}

n--;

}

void table3::Print()

{

for(int i=0;i<maxsize;i++)

if(!a[i].empty())

for (poly n : a[i])

{

cout << n.name << " ";

n.p.Print();

cout << endl;

}

}

## Приложение 3. Файл main.cpp

#include "table.h"

#include <iostream>

#include <fstream>

using namespace std;

void main()

{

int r, j, k, p;

double c;

List a;

string s;

table1 t1;

table2 t2;

table3 t3;

ofstream out;

out.open("\Log.txt");

out << "Логирование" << endl;

cout << "Enter number of polynomials" << endl;

cin >> r;

for (int i = 0; i < r; i++)

{

cout << "Enter name for polynomial" << endl;

cin >> s;

cout << "Entering polynomial" << endl;

cout << "Enter number of monomials" << endl;

cin >> k; cout << endl;

for (int j = 0; j < k; j++)

{

cout << "Enter power of monom" << " " << j + 1 << endl;

cin >> p; cout << endl;

while (p > 999)

{

cout << "Incorrect power, try again" << endl;

cin >> p; cout << endl;

}

cout << "Enter coefficient of monom" << " " << j + 1 << endl;

cin >> c; cout << endl;

a.Insert\_end(p, c);

}

pol b(a);

a.clear();

t1.insert(b, s);

t2.insert(b, s);

t3.insert(b, s);

out << "Количество операций во вставке в первую таблицы: " <<t1.counterI<< endl;

out << "Количество операций во вставке во вторую таблицы: " <<t2.counterI<< endl;

out << "Количество операций во вставке в третью таблицы: " <<t3.counterI<< endl;

}

cout << "Tables:" << endl;

t1.Print();

cout << endl;

cout << "==================================================================" << endl;

cout << endl;

t2.Print();

cout << endl;

cout << "==================================================================" << endl;

cout << endl;

t3.Print();

cout << endl;

cout << "==================================================================" << endl;

cout << endl;

while (k)

{

cout << "1.Delete" << endl;

cout << "2.Search" << endl;

cout << "3.Change" << endl;

cout << "4.Show tables" << endl;

cout << "0.Exit" << endl;

cin >> k;

switch (k)

{

case 1:

{

cout << "Enter his name" << endl;

cin >> s;

t1.Delete(s);

t2.Delete(s);

t3.Delete(s);

out << "Количество операций в удалении в первой таблице: " << t1.counterD << endl;

out << "Количество операций в удалении во второй таблице: " << t2.counterD << endl;

out << "Количество операций в удалении в третьей таблице: " << t3.counterD << endl;

cout << "He is dead" << endl;

break;

}

case 2:

{

cout << "Enter his name" << endl;

cin >> s;

pol b = t1.search(s);

t2.search(s);

t3.search(s);

out << "Количество операций в поиске в первой таблице: " << t1.counterS << endl;

out << "Количество операций в поиске во второй таблице: " << t2.counterS << endl;

out << "Количество операций в поиске в третьей таблице: " << t3.counterS << endl;

cout << endl;

b.Print();

cout << endl;

break;

}

case 3:

{

cout << "Enter his name" << endl;

cin >> s;

pol b1 = t1.search(s);

b1.Print();

pol b2;

cout << endl;

cout << "If you want to perform the operations witn new polynomial enter 1, with table polynomial enter 0" << endl;

cin >> k;

if (k) {

cout << "Entering 2-nd polynomial" << endl;

cout << "Enter number of monomials" << endl;

cin >> k; cout << endl;

for (int i = 0; i < k; i++)

{

cout << "Enter power of monom" << " " << i + 1 << endl;

cin >> p; cout << endl;

while (p > 999)

{

cout << "Incorrect power, try again" << endl;

cin >> p; cout << endl;

}

cout << "Enter coefficient of monom" << " " << i + 1 << endl;

cin >> c; cout << endl;

a.Insert\_end(p, c);

}

cout << "Entering of polynomials is complited";

pol b(a);

b2 = b;

}

else

{

string s1;

cout << "Enter his name" << endl;

cin >> s1;

pol b = t1.search(s1);

b2 = b;

}

cout << endl;

cout << "Choose operation" << endl;

cout << "1.Add polynomials" << endl;

cout << "2.Subtract polynomials" << endl;

cout << "3.Multiply polynomials" << endl;

cout << "4.Multiply 1-st polynomial on const" << endl;

pol b3;

cin >> k; cout << endl;

switch (k)

{

case 1:

{

b3 = b1 + b2;

t1.Delete(s);

t2.Delete(s);

t3.Delete(s);

t1.insert(b3, s);

t2.insert(b3, s);

t3.insert(b3, s);

break;

}

case 2:

{

b3 = b1 - b2;

t1.Delete(s);

t2.Delete(s);

t3.Delete(s);

t1.insert(b3, s);

t2.insert(b3, s);

t3.insert(b3, s);

break;

}

case 3:

{

b3 = b1 \* b2;

t1.Delete(s);

t2.Delete(s);

t3.Delete(s);

t1.insert(b3, s);

t2.insert(b3, s);

t3.insert(b3, s);

break;

}

case 4:

{

cout << "Enter constant" << endl;

cin >> c; cout << endl;

b3 = b1.Mull(c);

t1.Delete(s);

t2.Delete(s);

t3.Delete(s);

t1.insert(b3, s);

t2.insert(b3, s);

t3.insert(b3, s);

break;

}

default:

break;

}

break;

}

case 4:

{

cout << "Tables:" << endl;

t1.Print();

cout << endl;

cout << "==================================================================" << endl;

cout << endl;

t2.Print();

cout << endl;

cout << "==================================================================" << endl;

cout << endl;

t3.Print();

cout << endl;

cout << "==================================================================" << endl;

cout << endl;

break;

}

default:

break;

}

}

}