Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

**ХЕШ-ТАБЛИЦЫ**

**Отчет по лабораторной работе №5**

**По дисциплине**

**«Структуры и алгоритмы обработки данных в ЭВМ»**

Студент гр. 431-3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Гурулёв

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Проверил: профессор кафедры АСУ, д.т.н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Н. Горитов

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Томск 2023

# Задание на лабораторную работу

Написать программу, которая реализует метод закрытого хеширования с квадратичной последовательностью проб и хеш-функцией, основанной на методе деления с остатком. Хеш-таблица ориентирована на хранение символьной информации. Данные в хеш-таблицу заносятся из файла. Файл должен содержать не менее 15 слов. Вывести построенную хеш-таблицу на экран (вместе с количеством выполненных проб). Организовать поиск и удаление данных в хеш-таблице. Результаты поиска данных вывести на экран. Также вывести количество проб, которые были затрачены при поиске..

# Алгоритм решения задачи

1. Инициализируем хеш-таблицу;
2. Открываем файл;
3. Переносим данные из файла в таблицу;
4. Выводим таблицу;
5. Ищем позицию элемента “15”;
6. Удаляем элемент “8”
7. Ищем позицию элемента “15”;
8. Выводим таблицу;

# Листинг программы

**Для main.cpp:**

#include <iostream>

#include <conio.h>

#include <fstream>

#include <string>

#include "Hash-table.h"

using namespace std;

using namespace myLab;

int main()

{

MyHashTable table(50);

ifstream f("file.txt");

if (!f.is\_open())

{

cout << "error";

return 0;

}

string word;

while (f >> word)

{

if (!table.set\_new(word))

{

cout << "there is no place" << endl;

}

}

table.print\_table();

cout << endl;

//При записи 15, встречается колизия с 8, удалим её, чтобы посмотреть как функция найдет 15 в таблице, с таким пробелом

cout << table.find\_elem("fifteen") << endl;

table.delete\_elem("eight");

cout << table.find\_elem("fifteen") << endl;

cout << endl;

table.print\_table();

system("pause");

return 0;

}

**Для Hash-table.cpp:**

#include "Hash-table.h"

#include <iostream>

#include <iomanip>

#include <string>

#include <cmath>

using namespace std;

namespace myLab

{

MyHashTable::MyHashTable(int \_size)

{

size = \_size;

table = new str[size];

for (int i = 0; i < \_size; i++)

{

table[i].data = "";

table[i].status = 0;

}

}

MyHashTable::~MyHashTable()

{

delete[] table;

}

unsigned int MyHashTable::hash\_function(string \_data)

{

unsigned int hash = 0;

unsigned int key\_size = \_data.length();

for (int i = key\_size; i > 0; i--)

{

hash += \_data[i - 1] \* pow(2, i);

}

return hash;

}

bool MyHashTable::set\_new(string \_data)

{

unsigned int \_hashValue = (int)hash\_function(\_data);

bool flag = 1;

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

{

int position = (\_hashValue + (i\*i)) % size;

if (table[position].data.empty())

{

cout << "count of attemps: " << i + 1 << endl;

table[position].data = \_data;

table[position].status = 0;

flag = 0;

break;

}

}

if (flag)

{

return 0;

}

return 1;

}

int MyHashTable::find\_elem(string \_data)

{

unsigned int \_hashValue = (int)hash\_function(\_data);

int position = (\_hashValue) % size;

if (table[position].data == \_data)

{

cout << "count of attemps: " << 0 + 1 << endl;

return position;

}

for (int i = 1; (i < size) && ((!table[position].data.empty()) || (table[position].status)); i++)

{

position = (\_hashValue + (i\*i)) % size;

if (table[position].data == \_data)

{

cout << "count of attemps: " << i + 1<< endl;

return position;

}

}

cout << "Not found";

return -1;

}

bool MyHashTable::delete\_elem(string \_data)

{

int position = find\_elem(\_data);

if (position == -1)

{

return 0;

}

table[position].data = "";

table[position].status = 1;

cout << "was delete" << endl;

return 1;

}

bool MyHashTable::print\_table()

{

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

{

cout << setw(10) << table[i].data << " | " << table[i].status << endl;

}

return 0;

}

}

**Для Hash-table.h:**

#include <string>

using namespace std;

namespace myLab

{

class MyHashTable

{

private:

typedef struct

{

std::string data;

bool status;

}str;

str\* table;

int size;

public:

explicit MyHashTable(int \_size);

~MyHashTable();

unsigned int hash\_function(std::string \_data);

bool set\_new(std::string \_data);

int find\_elem(std::string \_data);

bool delete\_elem(std::string \_data);

bool print\_table();

};

}

# Пример решения

На рисунке 4.1 – 4.2 можно увидеть пример результата программы. Вывод состоит из таблицы и номеров попыток поиска некоторых данных.

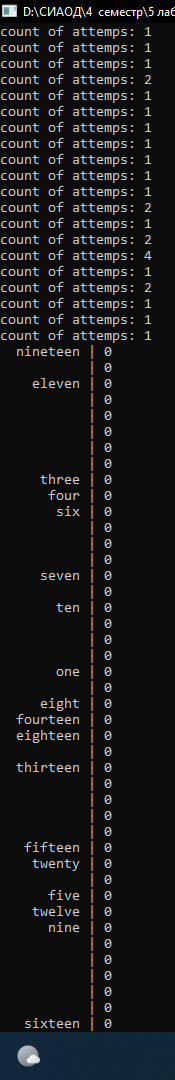


Рисунок 4.1 - Результат выполнения программы(1/2)

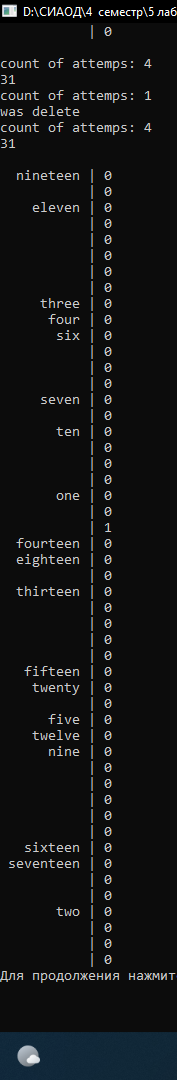


Рисунок 4.2 - Результат выполнения программы(2/2)

На рисунке 4.3 можно увидеть входные данные файла.

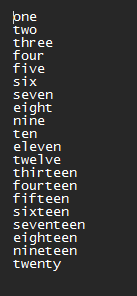


Рисунок 4.3 - Входные данные

# Вывод

 Я изучил как устроен АТД “Хеш-таблица”, его основные методы, а так же как с ними работать.