

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

## 1 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..

## 2 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;

## 3 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 коллизия 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();
    };
}
```



```
D:\СИАОД\4 семестр\5 лаб
| 0

count of attemps: 4
31
count of attemps: 1
was delete
count of attemps: 4
31

nineteen | 0
          | 0
eleven   | 0
          | 0
          | 0
          | 0
          | 0
          | 0
          | 0
three     | 0
four      | 0
six       | 0
          | 0
          | 0
seven     | 0
          | 0
ten       | 0
          | 0
          | 0
one       | 0
          | 0
          | 1
fourteen  | 0
eighteen  | 0
          | 0
thirteen | 0
          | 0
          | 0
          | 0
fifteen   | 0
twenty    | 0
          | 0
five      | 0
twelve    | 0
nine      | 0
          | 0
          | 0
          | 0
          | 0
sixteen   | 0
seventeen | 0
          | 0
          | 0
two       | 0
          | 0
          | 0
          | 0

Для продолжения нажмите
```

Рисунок 4.2 - Результат выполнения программы(2/2)

На рисунке 4.3 можно увидеть входные данные файла.

```
one
two
three
four
five
six
seven
eight
nine
ten
eleven
twelve
thirteen
fourteen
fifteen
sixteen
seventeen
eighteen
nineteen
twenty
```

Рисунок 4.3 - Входные данные

## **5 Вывод**

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