**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по практической работе №5**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: Хеш-таблицы

|  |  |  |
| --- | --- | --- |
| Студент гр. 8381 |  | Киреев К.А. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2019

## Цель работы.

Изучить основные характеристики и реализовать структуру данных хеш-таблица. В данной реализации для разрешения коллизий использовать метод цепочек. Исследовать сложность операции удаления элемента из таблицы.

## Задание.

Реализовать хеш-таблицу с цепочками.

Обязательная функциональность:

1. По заданному файлу F (типа file of Elem), все элементы которого различны, построить структуру данных определённого типа – БДП или хеш-таблицу;

2. Для построенной структуры данных проверить, входит ли в неё элемент е типа Elem, и если входит, то удалить элемент е из структуры данных. Предусмотреть возможность повторного выполнения с другим элементом.

## Основные теоретические положения.

**Хеш-таблица**  — это [структура данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), реализующая интерфейс [ассоциативного массива](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2), а именно, она позволяет хранить пары (ключ, значение) и выполнять три операции: операцию добавления новой пары, операцию поиска и операцию.

Существуют два основных варианта хеш-таблиц: **с цепочками** и **открытой адресацией**. Хеш-таблица содержит некоторый массив, элементы которого есть пары (хеш-таблица с открытой адресацией) или списки пар (хеш-таблица с цепочками).

Выполнение операции в хеш-таблице начинается с вычисления хеш-функции от ключа. Получающееся хеш-значение играет роль индекса в массиве. Затем выполняемая операция (добавление, удаление или поиск) перенаправляется объекту, который хранится в соответствующей ячейке массива.

Ситуация, когда для различных ключей получается одно и то же хеш-значение, называется **коллизией**. Поэтому механизм разрешения коллизий — важная составляющая любой хеш-таблицы.

В некоторых специальных случаях удаётся избежать коллизий вообще. Например, если все ключи элементов известны заранее (или очень редко меняются), то для них можно найти некоторую совершенную хеш-функцию, которая распределит их по ячейкам хеш-таблицы без коллизий. Хеш-таблицы, использующие подобные хеш-функции, не нуждаются в механизме разрешения коллизий, и называются хеш-таблицами с прямой адресацией.

Число хранимых элементов, делённое на размер массива (число возможных значений хеш-функции), называется **коэффициентом заполнения хеш-таблицы** (load factor) и является важным параметром, от которого зависит среднее время выполнения операций.

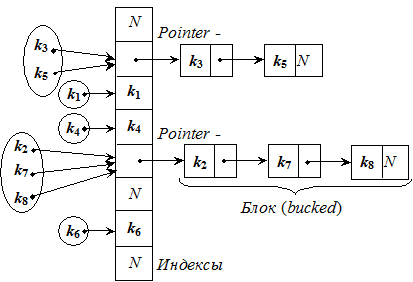


Рисунок 1 – Метод цепочек

## Выполнение работы.

Для решения задачи был разработан графический интерфейс с помощью QtCreator.

Таблица 1 – Слоты класса MainWindow и их назначение

|  |  |
| --- | --- |
| Метод | Назначение |
| **void on\_openFile\_clicked()** | Считывание из файла |
| **void on\_saveFile\_clicked()** | Запись из поля вывода в файл |
| **on\_averagecase\_clicked()** | Генерация массива |
| **on\_create\_clicked()** | Создание хеш-таблицы |
| **on\_deleteItem\_clicked()** | Удаление элемента из таблицы |

Для реализации хеш-таблицы был создан класс Hash.

Также были реализованы функции, создающие и изменяющие хеш-таблицу, некоторые из них приведены в табл. 2.

Таблица 2 – Основные функции работы с хеш-таблицой

|  |  |
| --- | --- |
| Функция | Назначение |
| void **createHashTable**(QTextEdit \*&output, QTextEdit \*&in, QTextEdit \*&info) | Создает хеш-таблицу по заданной строке |
| void **deleteItemFromHashTable**(QTextEdit \*&output, QLineEdit \*&input, QTextEdit \*&info) | Удаляет значение из хеш-таблицы |
| void **generateAverageCase**(QTextEdit \*&in, QLineEdit \*&sizeac) | Генерирует массив строк заданного количества |
| string **generateRandomString**(size\_t length) | Генерирует строку |

Программа имеет возможность графического отображения полученной хеш-таблице с помощью поля QTextEdit output. Также были созданы методы, необходимые для обработки хеш-таблицы.

## Оценка эффективности алгоритма.

Важное свойство хеш-таблиц состоит в том, что, при некоторых разумных допущениях, все три операции (поиск, вставка, удаление элементов) в среднем выполняются за время О(1). Но при этом не гарантируется, что время выполнения отдельной операции мало́. При предположении, что каждый элемент может попасть в любую позицию таблицы H с равной вероятностью и независимо от того, куда попал любой другой элемент, среднее время работы операции поиска элемента составляет Θ(1 + α), где α — коэффициент заполнения таблицы.

## 

Диаграмма 1 – Зависимость количества операций от позиции в цепочки

## Тестирование программы.

Таблица 3 – Результаты тестирования программы

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | Кол-во коллизий | Коэффициент заполнения |
| 10 | 0 | 0.000909091 |
| 100 | 4 | 0.00872727 |
| 1000 | 408 | 0.0538182 |
| 5000 | 4070 | 0.0845455 |
| 10000 | 9050 | 0.0870909 |

## Выводы.

В ходе выполнения лабораторной работы были изучены основные характеристики и реализована такая структура данных, как хеш-таблица с цепочками. Исследована сложность операции удаления элемента из таблицы.

# Приложение А Исходный код программы hash.cpp

#include "hash.h"

Hash::**Hash**(int n)

{

this->BUCKET = n; // constructor

table = new list<string>[BUCKET];

}

void Hash::**insertItem**(string value) // inserts a key into hash table

{

int index = hashFunction(value);

table[index].push\_back(value);

}

void Hash::**deleteItem**(string key) // deletes a key from hash table

{

// get the hash index of key

int index = hashFunction(key);

// find the key in (inex)th list

list <string> :: iterator i;

for (i = table[index].begin(); i != table[index].end(); i++)

{

if (\*i == key)

break;

}

// if key is found in hash table, remove it

if (i != table[index].end())

table[index].erase(i);

}

void Hash::**clearHashTable**()

{

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

table[i].clear();

}

int Hash::**hashFunction**(string x) // adaptive method

{

int s = 0;

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

s += x[i];

return s % SIZE;

}

int Hash::**hashFunction2**(int x) // multiplicative method

{

double r = x \* 0.13 ;

r = r - (int)r;

return r \* SIZE;

}

int Hash::**hashFunction3**(int x) // simple method

{

return x % SIZE;

}

void Hash::**displayInfo**(QTextEdit \*&info)

{

int counter\_coll = 0, counter = 0, counter\_data = 0;

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

{

if(table[i].size() > 1)

counter\_coll += (table[i].size() - 1);

counter += table[i].size();

if(table[i].size() > 0)

counter\_data += 1;

}

float lf = (float)counter\_data/BUCKET;

info->setPlainText("Amount of Collisions: " + QString::number(counter\_coll) +

"\nNumber of Stored Items: " + QString::number(counter) +

"\nNumber of Buckets: " + QString::number(BUCKET) +

"\nLoad Factor: " + QString::number(lf));

}

void Hash::**displayHash**(QTextEdit \*&output) //print hash table

{

QString outputstr;

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

{

outputstr += "[" + QString::number(i) + "] ";

for (auto x : table[i])

outputstr += QString::fromStdString(x) + " ----- ";

outputstr += "NULL\n";

}

outputstr += "\n";

output->setPlainText(outputstr);

}

# Приложение Б Исходный код программы hash.h

#ifndef HASH\_H

#define HASH\_H

#include <iostream>

#include <fstream>

#include <string>

#include <list>

#include <cstdlib>

#include <QMainWindow>

#include <QGraphicsItem>

#include <QGraphicsView>

#include <QGraphicsEffect>

#include <QFileDialog>

#include <QStandardPaths>

#include <QtGui>

#include <QLabel>

#include <QColorDialog>

#include <QInputDialog>

#include <QMainWindow>

#include <QPushButton>

#include <QMessageBox>

#include <QStringList>

#include <QTextEdit>

#include <stack>

#define SIZE 15 // 601, 211, 89, 34996 175939

using namespace std;

class **Hash**

{

int BUCKET; // No. of buckets

list <string> \*table; // Pointer to an array containing buckets

public:

**Hash**(int n);

void **insertItem**(string value);

void **deleteItem**(string key);

int **hashFunction**(string x);

int **hashFunction2**(int x);

int **hashFunction3**(int x);

void **displayHash**(QTextEdit \*&output);

void **displayInfo**(QTextEdit \*&info);

void **clearHashTable**();

};

#endif // HASH\_H

# Приложение b Исходный код программы mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <hash.h>

#include <processing.h>

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~***MainWindow***()

{

delete ui;

}

void MainWindow::**on\_create\_clicked**()

{

createHashTable(*ui->output*, *ui->inputht*, *ui->info*);

}

void MainWindow::**on\_deleteItem\_clicked**()

{

deleteItemFromHashTable(*ui->output*, *ui->input*, *ui->info*);

}

void MainWindow::**on\_averagecase\_clicked**()

{

generateAverageCase(*ui->inputht*, *ui->sizeac*);

}

void MainWindow::**on\_openfile\_clicked**()

{

QString fileName = QFileDialog::getOpenFileName(this, tr("load"), QDir::homePath(), tr("\*.txt"));

if (QString::compare(fileName, QString()) != 0)

{

ifstream f(qPrintable(fileName), ios::in);

string out;

getline(f, *out*);

f.close();

ui->inputht->setPlainText(QString::fromStdString(out));

}

}

void MainWindow::**on\_savefile\_clicked**()

{

QString filePath = QFileDialog::getSaveFileName(this, tr("save"), QDir::homePath(), tr("\*.txt"));

if (QString::compare(filePath, QString()) != 0)

{

ofstream ff(qPrintable(filePath));

ff << qPrintable(ui->output->toPlainText());

ff.close();

}

}

# Приложение г Исходный код программы mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

namespace **Ui** { class **MainWindow**; }

QT\_END\_NAMESPACE

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

**MainWindow**(QWidget \*parent = nullptr);

~***MainWindow***();

private slots:

void **on\_create\_clicked**();

void **on\_openfile\_clicked**();

void **on\_savefile\_clicked**();

void **on\_deleteItem\_clicked**();

void **on\_averagecase\_clicked**();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

# Приложение д Исходный код программы main.cpp

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(*argc*, argv);

MainWindow w;

w.show();

return a.exec();

}

# Приложение е Исходный код программы processing.cpp

#include <processing.h>

Hash h(SIZE);

void **createHashTable**(QTextEdit \*&output, QTextEdit \*&in, QTextEdit \*&info)

{

h.clearHashTable();

QStringList array = in->toPlainText().split(" ");

string\* a = new string[array.length()];

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

a[i] = array[i].toStdString();

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

h.insertItem(a[i]);

h.displayHash(*output*);

h.displayInfo(*info*);

}

void **deleteItemFromHashTable**(QTextEdit \*&output, QLineEdit \*&input, QTextEdit \*&info)

{

string x = input->text().toStdString();

h.deleteItem(x);

h.displayHash(*output*);

h.displayInfo(*info*);

}

void **generateAverageCase**(QTextEdit \*&in, QLineEdit \*&sizeac)

{

int size = sizeac->text().toInt();

srand(time(NULL));

string arr[size];

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

arr[i] = "";

size\_t newLen;

string newRandomValue;

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

{

newLen = rand() % 10 + 1;

newRandomValue = "";

newRandomValue = generateRandomString(newLen);

arr[i] = newRandomValue;

i++;

}

QString outputstr;

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

outputstr += QString::fromStdString(arr[i]) + " ";

outputstr.remove(outputstr.size()-1, 1);

in->setPlainText(outputstr);

}

string **generateRandomString**(size\_t length)

{

auto randchar = []() -> char

{

const char alphanum[] =

"0123456789"

"ABCDEFGHIJKLMNOPQRSTUVWXYZ"

"abcdefghijklmnopqrstuvwxyz";

return alphanum[ rand() % (sizeof(alphanum) - 1) ];

};

string str(length, 0);

generate\_n(str.begin(), length, randchar);

return str;

}

# Приложение Ж Исходный код программы processing.h

#ifndef PROCESSING\_H

#define PROCESSING\_H

#include <hash.h>

#include <ctime>

int **beginning**(QTextEdit \*&output, bool flga, QLineEdit \*&input, QTextEdit \*&in);

void **deleteItemFromHashTable**(QTextEdit \*&output, QLineEdit \*&input, QTextEdit \*&info);

void **createHashTable**(QTextEdit \*&output, QTextEdit \*&in, QTextEdit \*&info);

void **generateAverageCase**(QTextEdit \*&output, QLineEdit \*&sizeac);

string **generateRandomString**(size\_t len);

#endif // PROCESSING\_H