МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Алгоритмы и структуры данных»
Тема: БДП и хеш-таблицы

Студентка гр. 8381 _____ Ивлева О.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы.

Ознакомиться с основными характеристиками и особенностями такой структуры данных, как хеш-таблица, изучить особенности ее реализации на языке программирования C++.

Задание.

- По заданному файлу F (типа file of Elem), все элементы которого различны, построить структуру данных определённого типа – БДП или хеш-таблицу;
- Для построенной структуры данных проверить, входит ли в неё элемент e типа Elem, и если не входит, то добавить элемент e в структуру данных. Предусмотреть возможность повторного выполнения с другим элементом.

Основные теоретические положения.

Один из наиболее эффективных способов реализации словаря - хеш-таблица. Среднее время поиска элемента в ней есть O(1), время для наихудшего случая - O(n).

Хеширование полезно, когда широкий диапазон возможных значений должен быть сохранен в малом объеме памяти, и нужен способ быстрого, практически произвольного доступа. Хэш-таблицы часто применяются в базах данных, и, особенно, в языковых процессорах типа компиляторов и ассемблеров, где они изящно обслуживают таблицы идентификаторов. В таких приложениях, таблица - наилучшая структура данных.

Так как всякий доступ к таблице должен быть произведен через хэш-функцию, функция должна удовлетворять двум требованиям: Она должна быть быстрой, и она должна порождать хорошие ключи для распределения элементов по таблице. Последнее требование минимизирует коллизии (случаи, когда два разных элемента имеют одинаковое значение хеш-функции) и предотвращает

случай, когда элементы данных с близкими значениями попадают только в одну часть таблицы.

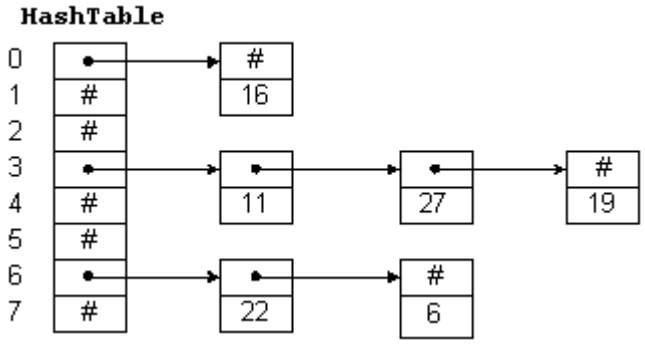

Рисунок 1 - Хеш-таблица

Выполнение работы.

Написание работы производилось на базе операционной системы Windows 10 в среде разработки QtCreator с использованием фреймворка Qt.

Для реализации программы был разработан графический интерфейс с помощью встроенного в QtCreator UI-редактора. Была добавлена кнопка, при нажатии на которую, происходит преобразование исходного выражения в хеш-таблицу.

Для реализации хеш-таблицы был создан динамический массив указателей на структуры Right, в которой хранится указатель на следующий элемент, если присутствует коллизия.

Также были реализованы функции поиска и добавления элемента в хеш-таблицу.

Функция do_hash_str(), которая принимает на вход заданную строку и указатель на массив структур Hash, в котором и будет храниться хеш-таблица.

Функция sum(), которая получает на вход элемент, считает сумму кодов знаков(хеш-функция), которая по данным значениям строит хеш-таблицу.

Функция find_el(), которая получает на вход дополнительные элементы и указатель на структуру, ищет данные элементы в хеш-таблице и если такого значения не оказалось, то добавляет его.

Функция print_hash() выводит хеш-таблицу на экран.

Оценка сложности алгоритма

Сложность алгоритма вставки в хеш-таблицу зависит от кол-ва коллизий, возникших при ее заполнении. Если известно кол-во коллизий, то сложность будет O(1) в среднем случае. В худшем случае, если все элементы создают коллизию, алгоритм каждый раз будет совершать кол-во итераций, равное числу элементов, то есть сложность будет равна O(n).

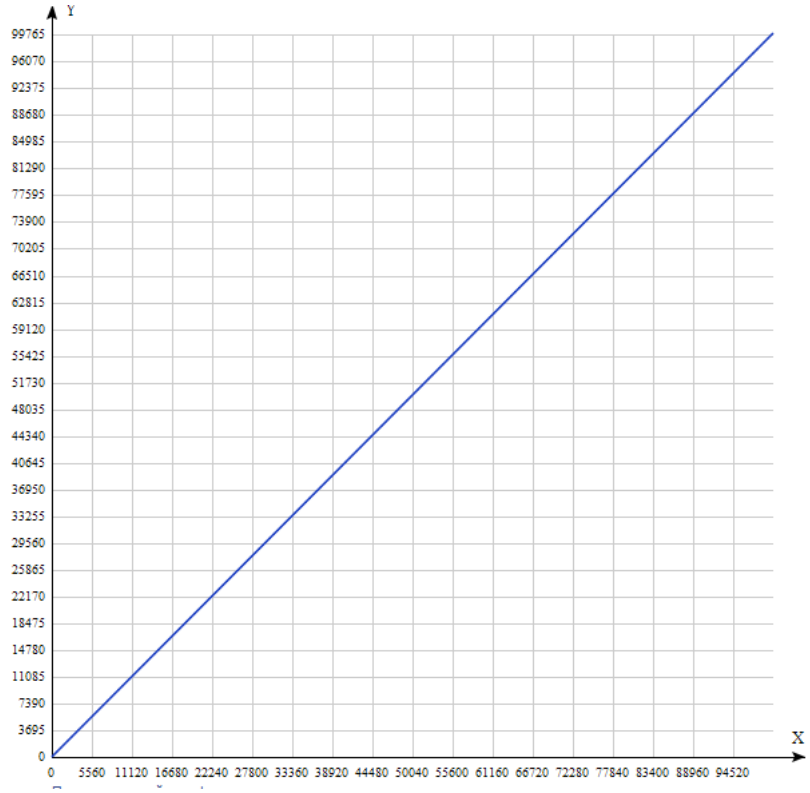

Рисунок 1 – Зависимость кол-ва операций от длины коллизии

Тестирование программы.

Вид программы после запуска представлен на рис. 2.

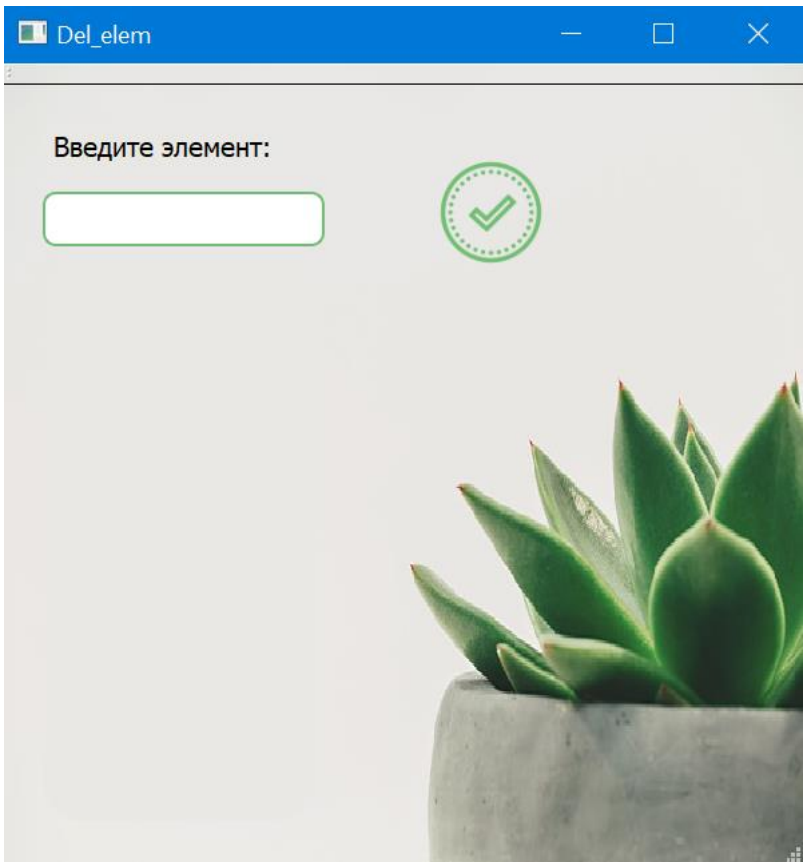

Рисунок 2 – Графический интерфейс программы

Работа программы представлена на рис. 3.

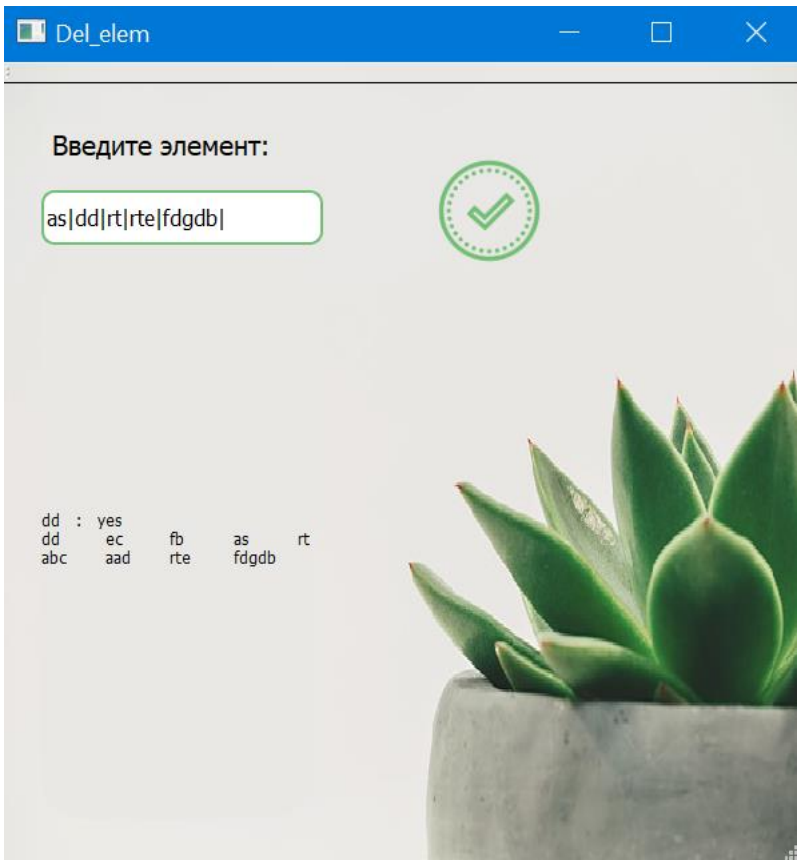

Рисунок 3 –Работа программы

Выводы.

В ходе выполнения лабораторной работы была написана программа, создающая хеш-таблицу согласно заданным элементам и выполняющая поиск и вставку новых элементов.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

Название файла: mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "functions.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_Hello_clicked();

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

Название файла: functions.h

```
#include <iostream>
#include <string>
#include <cctype>
```

```
#include <string>
#include <fstream>

using namespace std;

struct Right;
struct Hash;
void open(string &str);
void do_hash_str(Hash *ri, string &str);
void print_hash(Hash start[],string &output);
void resize(Hash *ri, int sum);
void find_el(string input, Hash* ri, string &output);

struct Right{
    Right *r = nullptr;
    string str = "";
};

struct Hash{
    Right *right = nullptr;
};
```

Название файла: functions.cpp

```
#include "functions.h"
int s = 10000;


void resize(Hash *ri, int sum){
    Hash *arr = new Hash[sum];
    for (int i=0; i < s; i++) {
        arr[i].right = ri[i].right;
    }
    //delete[] ri;
    ri = arr;
}

void open(string &str){
    ifstream fin; // создаем объект класса ifstream (считать)
    fin.open("D:\\prog\\cpp\\lab1\\text.txt");  //  открываем  файл  для
считывания
    fin >> str;
    if(!fin.is_open()) cout<<"ERROR";
    fin.close(); // закрываем файл
}

int sum(string strNew){
    int sum = 0;
    for(int i = 0; i < strNew.length(); i++){
        sum += int(strNew[i]);
```

```
    }
    return sum;
}

void do_hash_str(Hash *ri, string &str){
    string strNew = "";
    for(int i = 0; i < str.length(); i++){
        if (str[i] != '|'){
            strNew += str[i];
        }
        else {
            if(sum(strNew) > s) {
                resize(ri, sum(strNew));
                s = sum(strNew);
            }
            if (ri[sum(strNew)].right == nullptr) {
                Right *right1 = new Right;
                right1->str = strNew;
                ri[sum(strNew)].right = right1;
            }
            else{
                Right *now = ri[sum(strNew)].right;
                Right *ne = new Right;
                while (now->r){
                    now = now->r;
                }
                ne->str = strNew;
                now->r = ne;
            }
            strNew = "";
        }
    }
}

void print_hash(Hash* start, string &output){
    for (int i = 0; i < s; i++){
        if(start[i].right) {
            Right *now = start[i].right;
            while (now) {
                output+= now->str;
                output += "\t";
                now = now->r;
            }
        }
    }
}

void find_el(string input, Hash* ri, string &output){
    string strNew = "";
```

```
    for(int i = 0; i < input.length(); i++) {
        if (input[i] != '|') {
            strNew += input[i];
        }
        else {
            if (ri[sum(strNew)].right) {
                Right *now = ri[sum(strNew)].right;
                while (now->r) {
                    if (now->str == strNew) {
                        output += strNew;
                        output += "   :   yes";
                        output += "\n";
                        break;
                    }
                    now = now->r;
                    if (now->r == nullptr) {
                        Right *new1 = new Right;
                        new1->str = strNew;
                        new1->r = nullptr;
                        now->r = new1;
                        break;
                    }
                }
            } else {
                Right *right1 = new Right;
                right1->str = strNew;
                ri[sum(strNew)].right = right1;
            }
            strNew = "";
        }
    }
}
```

Название файла: mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->output->setWordWrap(true);
    QPixmap bkgnd("D:\\prog\\Styles\\plant-2004483_640.jpg");
    bkgnd = bkgnd.scaled(this->size(), Qt::IgnoreAspectRatio);
    QPalette palette;
```

```
        palette.setBrush(QPalette::Background, bkgnd);
        this->setPalette(palette);
    }

    MainWindow::~MainWindow()
    {
        delete ui;
    }

    void MainWindow::on_Hello_clicked()
    {
        string str;
        string input;
        //if (!ui->checkBox_2->isChecked()){
        open(str);
        input = qPrintable(ui->input->text());
        string output = "";
        int s = 10000;
        Hash *ri = new Hash[s];
        do_hash_str(ri, str);
        find_el(input, ri, output);
        print_hash(ri, output);
        ui->output->setText(QString::fromStdString(output));
    }
```

Название файла: mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>600</width>
    <height>600</height>
   </rect>
  </property>
  <property name="focusPolicy">
   <enum>Qt::NoFocus</enum>
  </property>
  <property name="windowTitle">
```

```
   <string>Del_elem</string>
  </property>
  <property name="autoFillBackground">
   <bool>false</bool>
  </property>
  <property name="styleSheet">
   <string notr="true"/>
  </property>
  <widget class="QWidget" name="centralWidget">
   <widget class="QPushButton" name="Hello">
    <property name="geometry">
     <rect>
      <x>320</x>
      <y>100</y>
      <width>91</width>
      <height>91</height>
     </rect>
    </property>
    <property name="cursor">
     <cursorShape>PointingHandCursor</cursorShape>
    </property>
    <property name="styleSheet">
     <string                                        notr="true">border-image:
url(:/C:/Users/Олеся/Downloads/icons8-ok-100_1.png);</string>
    </property>
    <property name="text">
     <string/>
    </property>
    <property name="autoRepeat">
     <bool>false</bool>
    </property>
   </widget>
   <widget class="QLabel" name="output">
    <property name="geometry">
     <rect>
      <x>30</x>
      <y>250</y>
      <width>201</width>
      <height>301</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 6pt &quot;MS Shell Dlg 2&quot;;
border-radius: 30%;
background-color: #e9e8e4;
</string>
    </property>
    <property name="text">
     <string/>
```

```
    </property>
   </widget>
   <widget class="QLineEdit" name="input">
    <property name="geometry">
     <rect>
      <x>30</x>
      <y>80</y>
      <width>211</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border-radius: 10%;
border: 2px solid #72be74;</string>
    </property>
   </widget>
   <widget class="QLineEdit" name="inputChar">
    <property name="geometry">
     <rect>
      <x>30</x>
      <y>190</y>
      <width>211</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border-radius: 10%;
border: 2px solid #72be74;</string>
    </property>
   </widget>
   <widget class="QCheckBox" name="checkBox">
    <property name="geometry">
     <rect>
      <x>320</x>
      <y>40</y>
      <width>191</width>
      <height>21</height>
     </rect>
    </property>
    <property name="cursor">
     <cursorShape>PointingHandCursor</cursorShape>
    </property>
    <property name="text">
     <string>Доп. Информация</string>
    </property>
   </widget>
   <widget class="QLineEdit" name="lineEdit">
    <property name="geometry">
     <rect>
```

```
      <x>30</x>
      <y>30</y>
      <width>181</width>
      <height>31</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">
font: 10pt &quot;MS Shell Dlg 2&quot;;
border-radius: 10%;
background-color: #e9e8e4;
</string>
    </property>
    <property name="text">
     <string> Введите строку:</string>
    </property>
   </widget>
   <widget class="QLineEdit" name="lineEdit_2">
    <property name="geometry">
     <rect>
      <x>30</x>
      <y>140</y>
      <width>181</width>
      <height>31</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">
font: 10pt &quot;MS Shell Dlg 2&quot;;
border-radius: 10%;
background-color: #e9e8e4;</string>
    </property>
    <property name="text">
     <string> Введите символ:</string>
    </property>
   </widget>
  </widget>
  <widget class="QMenuBar" name="menuBar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>600</width>
     <height>17</height>
    </rect>
   </property>
  </widget>
  <widget class="QToolBar" name="mainToolBar">
   <attribute name="toolBarArea">
```

```
    <enum>TopToolBarArea</enum>
   </attribute>
   <attribute name="toolBarBreak">
    <bool>false</bool>
   </attribute>
  </widget>
  <widget class="QStatusBar" name="statusBar"/>
 </widget>
 <layoutdefault spacing="6" margin="11"/>
 <resources/>
 <connections/>
</ui>
```