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

Студент гр. 8381 _____ Облизанов А.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы.

Ознакомиться с основными характеристиками и особенностями такой структуры данных, как хэш-таблица. Исследовать сложность таких операций, как вставка, удаление из хеш-таблицы.

Задание.

Необходимо реализовать хеш-таблицу с открытой адресацией и следующей функциональностью:

- Построение хеш-таблицы по заданному файлу F (типа file of *Elem*), все элементы которого различны;
- Для построенной структуры данных проверить, входит ли в неё элемент *e* типа *Elem*, и если входит, то удалить элемент *e* из структуры данных. Предусмотреть возможность повторного выполнения с другим элементом.

Выполнение работы.

Написание работы производилось на базе операционной системы Windows 10 в среде разработки QtCreator с использованием фреймворка Qt. Сборка, отладка производились в QtCreator, запуск программы осуществлялся через командную строку. Исходные коды файлов программы представлены в приложениях А-Л.

Для реализации программы был разработан графический интерфейс с помощью встроенного в QtCreator UI-редактора. Он представляет из себя кнопку для выбора файла, содержащего элементы для хеш-таблицы, поле ввода ключа для удаления элемента, кнопку подтверждения и поле вывода. Слоты класса MainWindow, отвечающие за обработку элементов графического интерфейса, описаны в приложении В.

Для реализации хеш-таблицы были созданы шаблонные классы для пары – Pair, и динамического массива – CVector. Основные методы класса Pair представлены в табл. 1.

Таблица 1 – Основные методы класса Pair

Метод	Назначение
`S getFirst();`	Возвращает первый элемент пары типа S
`T getSecond();`	Возвращает второй элемент пары типа T
`int isActive();`	Возвращает true, если в пару заносились и не удалялись какие-либо значения
`void setFirst(S);`	Устанавливает первый элемент типа S
`void setSecond(T);`	Устанавливает второй элемент типа T
`void setDeleted();`	Устанавливает флаг, помечающий пару как ранее удаленную
`const Pair & operator= (const Pair &other);`	Копирует элементы пары (присваивание)

Динамический массив CVector для хэш-таблицы используется для хранения пар «ключ-значение», однако реализован также с помощью шаблонов. Основные методы класса приведены в табл. 2.

Таблица 2 – Основные методы класса CVector

Метод	Назначение
`unsigned int getCapacity();`	Возвращает текущий объем выделенной памяти для динамического массива в элементах
`int resize (unsigned int nsize);`	Изменяет объем выделенной памяти, точнее выделяет память с новым объемом, копируя туда элементы массива
`T & operator[] (unsigned int index);`	Позволяет обращаться к элементу массива с помощью скобок [] по индексу
`CVector<T> & operator= (const CVector<T> &);`	Копирует элементы массива (присваивание)

Для реализации хеш-таблицы был создан шаблонный класс HashTable, который хранит в себе динамический массив пар и обеспечивает выполнение операций вставки и удаления. Основные метода класса приведены в табл. 3.

Таблица 3 – Основные методы класса HashTable

Метод	Назначение
`void setSize(int size);`	Изменяет объем выделенной памяти для таблицы (динамического массива)
`int getSize();`	Возвращает объем выделенной памяти для таблицы (количество пар)
`Pair<S,T> & operator[] (unsigned int index);`	Позволяет обращаться к элементу таблицы с помощью скобок [] по индексу
`T & operator[] (unsigned int index);`	Позволяет обращаться к элементу массива с помощью скобок [] по индексу
`unsigned int set(S x, T y, unsigned int index);`	Осуществляет вставку в таблицу по правилам открытой адресации
`Pair<S,T> get(S key, unsigned int index);`	Возвращает элемент таблицы (пару) по заданному ключу и индексу (хеш-функции)
`unsigned int remove(S key, unsigned int index);`	Удаляет элемент таблицы (пару) по заданному ключу и индексу (хеш-функции), помечая этот элемент как удаленный для реализации открытой адресации
`string print();`	Возвращает строку, в которой содержится представление хеш-таблицы в формате: <индекс> (<ключ>,<значение>)
`string getString(S key, unsigned int index);`	Возвращает строку, в которой содержится информация о паре в хеш-таблице по заданному ключу и индексу (хеш-функции)

Для демонстрации работы алгоритмов хеширования, вставки и удаления из хеш-таблицы для ключей и значений был выбран формат строк. Для функций,

отвечающих за ввод и вывод, был создан класс StrStrWorker. Основные методы класса приведены в табл. 4.

Таблица 4 – Основные методы класса StrStrWorker

Метод	Назначение
`QString getFromFile (QString fileName);`	Возвращает содержимое файла по указанному пути в виде строки
`QString createHashTable (QString input);`	Создает хеш-таблицу, где ключи и значения представлены строками, и возвращает строку с информацией о числе итераций и элементами таблицы
`QString deleteElem (QString input);`	Удаляет элемент, если таковой имеется, из хеш-таблицы, возвращает строку с информацией о числе итераций и элементами таблицы
`void saveStrToFile(QString output, QString fileName);`	Записывает строку в указанный файл
`unsigned int hFunc (string key);`	Возвращает индекс элемента по ключу. Индекс формируется как целое среднее значение кодов символов в ключе

Оценка сложности алгоритма

Сложность алгоритма вставки в хеш-таблицу зависит от количества коллизий, возникших при ее заполнении. Если известно количество коллизий $C - const$, то число итераций не превысит $O(C)$, то есть в среднем случае сложность алгоритма $O(1)$. В худшем случае, если все элементы создают коллизию, алгоритм каждый раз будет совершать количество итераций, равное числу элементов в таблице. Таким образом, в худшем случае сложность алгоритма вставки элемента равна $O(N)$.

Сложность алгоритма удаления элемента из хеш-таблицы таким же образом зависит от количества итераций и в худшем случае составляет $O(N)$, а в среднем случае $O(1)$.

Тестирование программы.

Графический интерфейс программы после заполнения хеш-таблицы представлен на рис. 1.

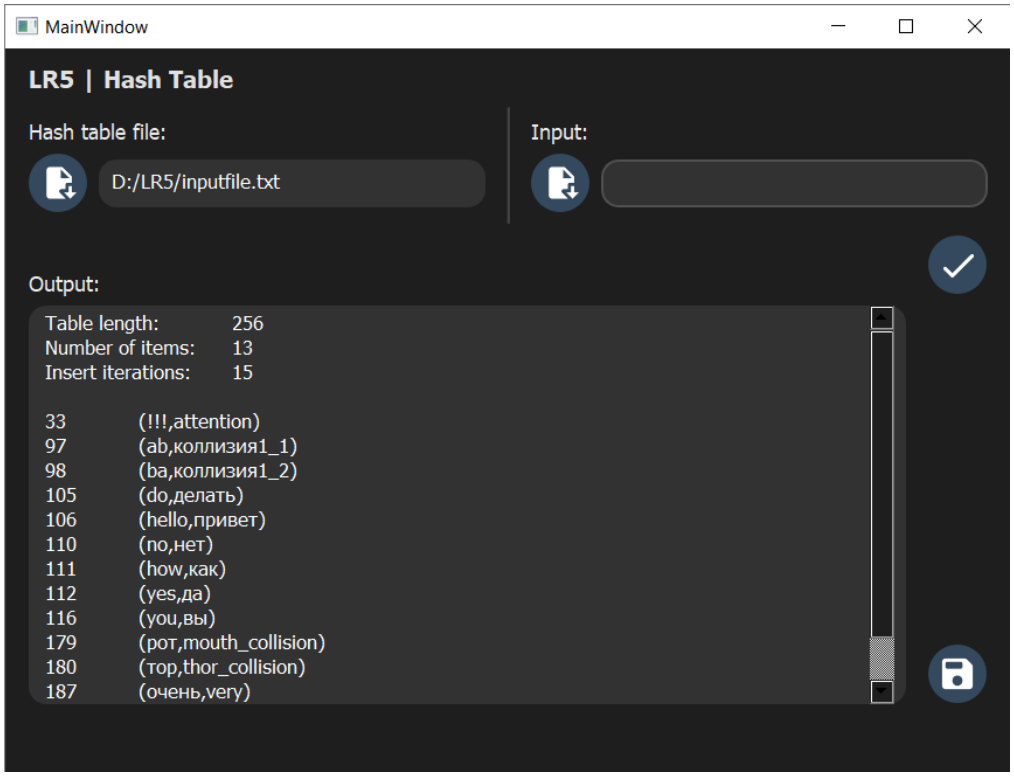

Рисунок 1 – Графический интерфейс программы

Для практического подтверждения оценки сложности был проведен ряд тестов, в которых исследовалась зависимость числа итераций во время заполнения таблицы от количества элементов. Так как для вставки одного элемента сложность алгоритма $O(1)$, то для вставки N элементов ожидается линейная зависимость числа итераций от числа элементов.

Для исследования среднего случая хеш-таблица была заполнена элементами, создающими небольшие коллизии. Пример такого заполнения представлен на рис. 2.

```
33      (!!!,attention)
97      (ab,коллизия1_1)
98      (ba,коллизия1_2)
105     (do,делать)
106     (hello,привет)
110     (no,нет)
111     (how,как)
112     (yes,да)
116     (you,вы)
179     (тор,thor_collision)
180     (орт,orth_collision)
181     (рот,mouth_collision)
187     (очень,very)
192     (наверное,maybe)
```

Рисунок 2 – Пример среднего случая заполнения хеш-таблицы

Результат тестирования для среднего случая приведен в табл. 5.

Таблица 5 – Тестирование среднего случая алгоритма вставки

№	Число элементов	Число коллизий	Число итераций
1	5	1	6
2	10	3	15
3	15	5	25
4	20	6	35
5	25	8	57

На основе полученных данных был построен график зависимости числа итераций от числа элементов хеш-таблицы $n(N)$, представленный на рис. 3.

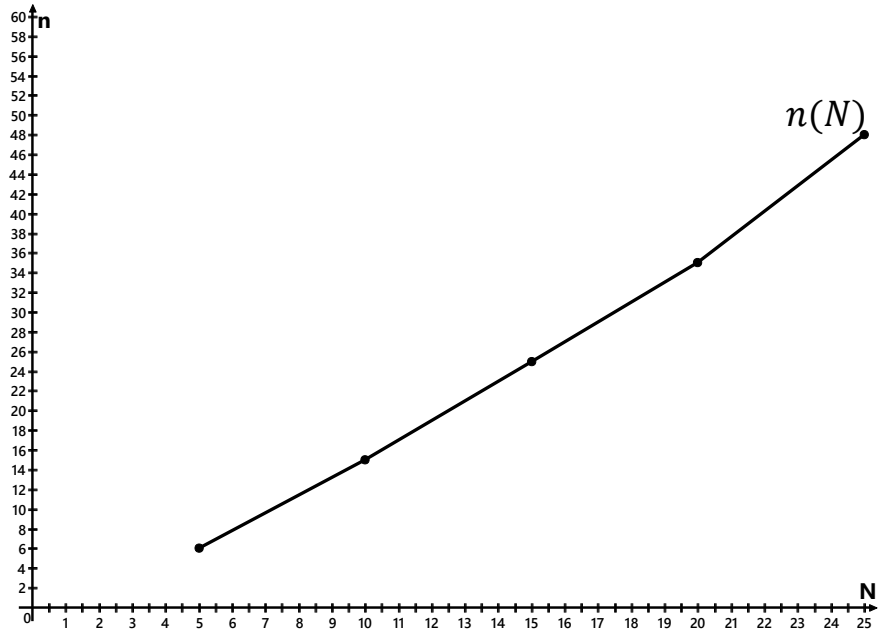

Рисунок 3 – График зависимости числа итераций от числа элементов

На графике видна линейная зависимость n от N, что подтверждает оценку алгоритма вставки одного элемента $O(1)$.

Для исследования худшего случая хеш-таблица была заполнена элементами, которые создают коллизии со всеми остальными элементами. Результат тестирования приведен в табл. 6.

Таблица 6 – Тестирование худшего случая алгоритма вставки

№	Число элементов	Число коллизий	Число итераций (одна вставка)	Число итераций (все вставки)
1	5	4	5	15
2	10	9	10	55
3	15	14	15	120
4	20	19	20	210
5	25	24	25	325

На основе полученных данных видно, что в худшем случае максимальное число итераций для вставки одного элемента совпадает с числом элементов, что соответствует сложности $O(N)$. Это связано с тем, что алгоритм вставки с открытой адресацией, добавляя последний элемент, проходит по всем предыдущим элементам.

Для исследование среднего случая работы алгоритма удаления хеш-таблица была заполнена элементами, создающими небольшие коллизии. Для элемента, который выбирался для удаления, считалось количество C других элементов, вступающих с ним в коллизию. Результат тестирования приведен в табл. 7.

Таблица 7 – Тестирование среднего случая алгоритма удаления

№	Число элементов коллизии C	Число итераций
1	5	3
2	5	5
3	10	10
4	10	6
5	15	3
6	15	13
7	20	6
8	20	17
9	20	10
10	20	20

Количество итераций для алгоритма удаления элемента зависит не только от числа элементов в коллизии, но и от позиции выбранного для удаления

элемента. Чем меньше разница между индексом элемента и хэш-функцией, тем быстрее будет найден и удален элемент. В целом, количество итераций для удаления элемента не превышает число C. Если оно известно, то сложность алгоритма можно оценить как $O(1)$ в среднем случае.

Выводы.

В ходе выполнения лабораторной работы была написана программа, реализующая хеш-таблицу для любого типа данных с открытой адресацией и возможностью удаления элементов, которая была использована в программе для работы с парами строк. Была исследована сложность алгоритмов вставки и удаления элементов из хеш-таблицы.

ПРИЛОЖЕНИЕ А

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 Б

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.H

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include "strstrworker.h"

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_pushOpenFile_clicked();

    void on_pushOpenText_clicked();

    void on_pushOk_clicked();

    void on_pushSave_clicked();

private:
    Ui::MainWindow *ui;
    StrStrWorker ssw;
};
#endif // MAINWINDOW_H
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.CPP

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

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->textOutput->setReadOnly(true);
    ui->fileInput->setReadOnly(true);
    ui->pushOk->setVisible(false);
}

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


void MainWindow::on_pushOpenFile_clicked()
{
    QString fileName = QFileDialog::getOpenFileName(this, tr("Choose input file
(TXT)"), QDir::homePath(), tr("*.txt"));
    if (QString::compare(fileName, QString()) != 0)
    {
        ui->fileInput->setPlainText(fileName);
        QString                              output                              =
ssw.createHashTable(ssw.getFromFile(ui->fileInput->toPlainText()));
        ui->textOutput->setPlainText(output);
        if (output.contains("Error"))
            ui->pushOk->setVisible(false);
        else
            ui->pushOk->setVisible(true);
    }
}

void MainWindow::on_pushOpenText_clicked()
{
    QString fileName = QFileDialog::getOpenFileName(this, tr("Choose input file
(TXT)"), QDir::homePath(), tr("*.txt"));
    if (QString::compare(fileName, QString()) != 0)
    {
        ui->textInput->setPlainText(ssw.getFromFile(fileName));
```

```
    }
}

void MainWindow::on_pushOk_clicked()
{
    if (QString::compare(ui->textInput->toPlainText(), QString()) != 0)
    {
        QString output = ssw.deleteElem(ui->textInput->toPlainText());
        ui->textOutput->setPlainText(output);
    }
}

void MainWindow::on_pushSave_clicked()
{
    QString filePath = QFileDialog::getSaveFileName(this, tr("Save to TXT file"),
QDir::homePath(), tr("*.txt"));
    if (QString::compare(filePath, QString()) != 0)
    {
        ssw.saveStrToFile(ui->textOutput->toPlainText(), filePath);
    }
}
void MainWindow::on_outputBtn_toggled(bool checked)
{
    if (checked)
    {
        ui->graphicsView->hide();
        ui->output->show();
    }
    else
    {
        ui->output->hide();
        ui->graphicsView->show();
    }
}
```

ПРИЛОЖЕНИЕ Г

ИСХОДНЫЙ КОД ПРОГРАММЫ. PAIR.H

```
#ifndef PAIR_H
#define PAIR_H
#include "libraries.h"

template <typename S, typename T>
class Pair
{
public:
    Pair();
    Pair(S,T);
    Pair(Pair &);
    ~Pair();
    const Pair & operator=(const Pair &other);

    string toString();

    S getFirst();
    T getSecond();
    void setFirst(S);
    void setSecond(T);
    void setDeleted();

    int isActive();
    int wasDeleted();
private:
    S *f;
    T *s;
    bool active;
    bool deleted;
};

template <typename S, typename T>
Pair<S,T>::Pair()
{
    f = NULL;
    s = NULL;
    active = false;
    deleted = false;
}

template <typename S, typename T>
Pair<S,T>::Pair(S x, T y)
{
    f = new S;  *f = x;
```

```
    s = new T;  *s = y;
    active = true;
    deleted = false;
}

template <typename S, typename T>
S Pair<S,T>::getFirst()
{
    if (f != NULL)
        return *f;
    else
        return NULL;
}

template <typename S, typename T>
T Pair<S,T>::getSecond()
{
    if (s != NULL)
        return *s;
    else
        return NULL;
}

template <typename S, typename T>
void Pair<S,T>::setFirst(S x)
{
    if (f==NULL)
        f = new S;
    *f = x;
    active = true;
    deleted = false;
}

template <typename S, typename T>
void Pair<S,T>::setSecond(T y)
{
    if (s == NULL)
        s = new T;
    *s = y;
    active = true;
    deleted = false;
}

template <typename S, typename T>
string Pair<S,T>::toString()
{
    stringstream ss;
```

```
    if (active)
    {
        ss << "(";
        if (f == NULL)
            ss << "NULL";
        else
            ss << (*f);
        ss<<",";
        if (s == NULL)
            ss << "NULL";
        else
            ss << (*s);
        ss << ")";
    }
    else
    {
        ss << "NULL";
    }
    return ss.str();
}

template <typename S, typename T>
int Pair<S,T>::isActive()
{
    return active;
}

template <typename S, typename T>
int Pair<S,T>::wasDeleted()
{
    return deleted;
}

template <typename S, typename T>
Pair<S,T>::Pair(Pair &other)
{
    f = NULL; s = NULL;
    if (other.f != NULL)
        f = new S(*other.f);
    if (other.s != NULL)
        s = new T(*other.s);
}

template <typename S, typename T>
Pair<S,T>::~Pair()
{
    if (f != NULL)
```

```
        delete f;
    if (s != NULL)
        delete s;
    f = NULL;
    s = NULL;
    deleted = false;
    active = false;
}

template <typename S, typename T>
void Pair<S,T>::setDeleted()
{
    deleted = true;
}

template <typename S, typename T>
const Pair<S,T> & Pair<S,T>::operator=(const Pair<S,T> &other)
{
    if(this != &other)
    {
        if(f != NULL)
        delete f;
        if(s != NULL)
        delete s;
        f = NULL; s = NULL;
        if (other.f != NULL)
        f = new S(*other.f);
        if (other.s != NULL)
        s = new T(*other.s);
        deleted = other.deleted;
        active = other.active;
    }
    return *this;
}

#endif // PAIR_H
```

ПРИЛОЖЕНИЕ Д

ИСХОДНЫЙ КОД ПРОГРАММЫ. CVECTOR.H

```
#define PAIRLIST_H
#include "pair.h"

template <typename T>
class CVector
{
public:
    CVector();
    CVector(unsigned int nsize);
    unsigned int getCapacity();
    int length();
    int resize(unsigned int nsize);
    T & operator[](unsigned int index);
    CVector<T> & operator=(const CVector<T> &);
private:
    T *tail;
    unsigned int capacity;
};

//constructors

template <typename T>
CVector<T>::CVector()
{
    tail = new T[1];
    capacity = 1;
};

template <typename T>
CVector<T>::CVector(unsigned int nsize)
{
    capacity = nsize;
    tail = new T[capacity];
}

//get info

template <typename T>
unsigned int CVector<T>::getCapacity()
{
    return capacity;
}

template <typename T>
```

```
int CVector<T>::length()
{
    return capacity;
}

//

template <typename T>
int CVector<T>::resize(unsigned int nsize)
{
    if (nsize <= capacity)
        return 1;
    T *temp = new T[nsize];
    for (unsigned int i=0; i<capacity; i++)
    {
        temp[i] = tail[i];
    }
    capacity = nsize;
    delete [] tail;
    tail = temp;
    return 0;
}

template <typename T>
T& CVector<T>::operator[](unsigned int index)
{
    return tail[index];
}

template<class T>
CVector<T> & CVector<T>::operator = (const CVector<T> & v)
{
    delete[] tail;
    capacity = v.capacity;
    tail = new T[capacity];
    for (unsigned int i = 0; i < capacity; i++)
        tail[i] = v.tail[i];
    return *this;
}

#endif // PAIRLIST_H
```

ПРИЛОЖЕНИЕ Е

ИСХОДНЫЙ КОД ПРОГРАММЫ. HASHTABLE.H

```
#ifndef HASHTABLE_H
#define HASHTABLE_H
#include "cvector.h"
#define SIZE 256

template <typename S, typename T>
class HashTable
{
public:
    HashTable();
    HashTable(unsigned int size);

    void setSize(int size);
    int getSize();
    unsigned int getItemCounter();

    Pair<S,T> & operator[](unsigned int index);
    unsigned int set(S x, T y, unsigned int index);
    Pair<S,T> get(S key, unsigned int index);
    unsigned int remove(S key, unsigned int index);
    void clear();

    string getString(S key, unsigned int index);
    string print();
private:
    CVector< Pair<S,T> > table;
    unsigned int itemCounter;
};

template <typename S, typename T>
HashTable<S,T>::HashTable()
{
    itemCounter = 0;
    setSize(SIZE);
}

template <typename S, typename T>
HashTable<S,T>::HashTable(unsigned int size)
{
    itemCounter = 0;
    setSize(size);
}

template <typename S, typename T>
```

```
void HashTable<S,T>::setSize(int size)
{
    table.resize(size);
}

template <typename S, typename T>
int HashTable<S,T>::getSize()
{
    return table.getCapacity();
}

template <typename S, typename T>
unsigned int HashTable<S,T>::getItemCounter()
{
    return itemCounter;
}

template <typename S, typename T>
unsigned int HashTable<S,T>::set(S x, T y, unsigned int index)
{
    unsigned int counter = 1;
    if (index + 5 >= table.getCapacity())
        table.resize((index + 5) * 2);
    while (table[index].isActive())
    {
        index++;
        counter++;
        if (index + 5 >= table.getCapacity())
            table.resize((index + 5) * 2);
    }
    table[index].setFirst(x);
    table[index].setSecond(y);
    itemCounter++;
    return counter;
}

template <typename S, typename T>
Pair<S,T> HashTable<S,T>::get(S key, unsigned int index)
{
    while (table[index].isActive())
    {
        if (table[index].wasDeleted())
        {
            index++;
            continue;
        }
        if (table[index].getFirst() != key)
```

```
            index++;
        else
            break;
    }
    return table[index];
}

template <typename S, typename T>
unsigned int HashTable<S,T>::remove(S key, unsigned int index)
{
    unsigned int counter = 1;
    while (table[index].isActive())
    {
        if (table[index].wasDeleted())
        {
            index++;
            counter++;
            continue;
        }
        if (table[index].getFirst() != key)
        {
            index++;
            counter++;
        }
        else
            break;
    }
    if (table[index].isActive())
    {
        table[index].~Pair();
        table[index].setDeleted();
        itemCounter -= 1;
        return counter;
    }
    else
        return 0;

}

template <typename S, typename T>
void HashTable<S,T>::clear()
{
    for (unsigned int i=0; i<table.getCapacity(); i++)
    {
        table[i].~Pair();
    }
    itemCounter = 0;
```

```
}

template <typename S, typename T>
string HashTable<S,T>::getString(S key, unsigned int index)
{
    string output;
    while (table[index].isActive())
    {
        if (table[index].getFirst() != key)
            index++;
        else
            break;
    }
    if (table[index].isActive())
        output = table[index].toString();
    else
        output = "No key in table";
    return output;
}

template <typename S, typename T>
string HashTable<S,T>::print()
{
    string output;
    for (unsigned int i=0; i<table.getCapacity(); i++)
    {
        if (table[i].isActive() && !table[i].wasDeleted())
        {
            output += to_string(i);
            output += "\t";
            output += table[i].toString();
            output += "\n";
        }
    }
    return output;
}

template <typename S, typename T>
Pair<S,T> & HashTable<S,T>::operator[](unsigned int index)
{
    return table[index];
}

#endif // HASHTABLE_H
```

ПРИЛОЖЕНИЕ Ж

ИСХОДНЫЙ КОД ПРОГРАММЫ. STRSTRWORKER.H

```
#ifndef STRSTRWORKER_H
#define STRSTRWORKER_H
#include "hashtable.h"


class StrStrWorker
{
public:
    StrStrWorker();
    QString getFromFile(QString fileName);
    QString createHashTable(QString input);
    QString deleteElem(QString input);
    void saveStrToFile(QString output, QString fileName);
private:
    HashTable <string, string> hTable;
    unsigned int hFunc(string key);
    int iter;
};

#endif // STRSTRWORKER_H
```

ПРИЛОЖЕНИЕ И

ИСХОДНЫЙ КОД ПРОГРАММЫ. STRSTRWORKER.CPP

```
#include "strstrworker.h"

StrStrWorker::StrStrWorker()
{
    iter = 0;
}

QString StrStrWorker::getFromFile(QString fileName)
{
    ifstream fin(qPrintable(fileName), ios::in);
    string out;
    string temp;
    while (true)
    {
        getline(fin, temp);
        out += temp;
        if (!fin.eof())
            out += "\n";
        else
            break;
    }
    fin.close();
    return QString::fromStdString(out);
}

QString StrStrWorker::createHashTable(QString input)
{
    hTable.setSize(SIZE);
    hTable.clear();
    QStringList pairs = input.split("\n");
    QStringList temp;
    QString output;
    unsigned int iter = 0;
    unsigned int sumIter = 0;
    unsigned int maxIter = 0;
    for (int i=0; i<pairs.length(); i++)
    {
        temp = pairs[i].split(" ");
        if (temp.length() != 2)
        {
            output += "Error! Incorrect file content in line:\n\"";
            output += pairs[i];
            output += "\"";
            return output;
```

```
        }
        iter    =    hTable.set(temp[0].toStdString(),    temp[1].toStdString(),
hFunc(temp[0].toStdString()));
        if (iter > maxIter)
            maxIter = iter;
        sumIter += iter;
    }
    output += "Table length:\t";
    output += QString::number(hTable.getSize());
    output += "\nNumber of items:\t";
    output += QString::number(hTable.getItemCounter());
    output += "\nAll inserts iterations:\t";
    output += QString::number(sumIter);
    output += "\nMax insert iterations:\t";
    output += QString::number(maxIter);
    output += "\n\n";
    output += QString::fromStdString(hTable.print());
    return output;
}

QString StrStrWorker::deleteElem(QString input)
{
    unsigned int iter = 0;
    QString output;
    string in = input.toStdString();
    iter = hTable.remove(in, hFunc(in));
    output += "Table length:\t";
    output += QString::number(hTable.getSize());
    output += "\nNumber of items:\t";
    output += QString::number(hTable.getItemCounter());
    if (iter == 0)
    {
        output += "\nError! No element in hash table";
    }
    else
    {
        output += "\nRemove iterations:\t";
        output += QString::number(iter);
    }
    output += "\n\n";
    output += QString::fromStdString(hTable.print());
    return output;
}

unsigned int StrStrWorker::hFunc(string key)
{
    unsigned int result = 0;
```

```
    for (unsigned int i=0; i<key.length(); i++)
    {
        result += static_cast<unsigned char>(key[i]);
    }
    result /= key.length();
    return result;
}

void StrStrWorker::saveStrToFile(QString output, QString fileName)
{
    ofstream fout(qPrintable(fileName));
    fout << qPrintable(output);
    fout.close();
}
```

ПРИЛОЖЕНИЕ К

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR5.PRO

```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11

# The following define makes your compiler emit warnings if you use
# any Qt feature that has been marked deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of
Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000     # disables all the APIs
deprecated before Qt 6.0.0

SOURCES += \
    main.cpp \
    mainwindow.cpp \
    strstrworker.cpp

HEADERS += \
    cvector.h \
    hashtable.h \
    libraries.h \
    mainwindow.h \
    pair.h \
    strstrworker.h

FORMS += \
    mainwindow.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

RESOURCES += \
    images/images.qrc
```

ПРИЛОЖЕНИЕ Л

ИСХОДНЫЙ КОД ПРОГРАММЫ. 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>860</width>
    <height>620</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <property name="styleSheet">
   <string notr="true">background-color: rgb(30, 30, 30)</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <widget class="QPushButton" name="pushOk">
    <property name="geometry">
     <rect>
      <x>790</x>
      <y>160</y>
      <width>50</width>
      <height>50</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border-image: url(:/ok.png);</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QPushButton" name="pushOpenFile">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>90</y>
      <width>50</width>
      <height>50</height>
     </rect>
    </property>
```

```
    <property name="styleSheet">
     <string notr="true">border-image: url(:/open.png);</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QPushButton" name="pushSave">
    <property name="geometry">
     <rect>
      <x>790</x>
      <y>510</y>
      <width>50</width>
      <height>50</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border-image: url(:/savef.png);</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QTextEdit" name="textInput">
    <property name="geometry">
     <rect>
      <x>510</x>
      <y>95</y>
      <width>331</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: rgb(50, 50, 50);
color: white;
border-radius: 15px;
font: 10pt &quot;Tahoma&quot;;
padding: 4px 5px 0px 7px;
border: 2px solid rgb(80,80,80);</string>
    </property>
    <property name="html">
     <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot;
/&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
```

```
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'Tahoma'; font-
size:10pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px;       margin-right:0px;       -qt-block-indent:0;       text-
indent:0px;&quot;&gt;&lt;br /&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
   </widget>
   <widget class="QLabel" name="nameApp">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>10</y>
      <width>241</width>
      <height>31</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 12pt &quot;Tahoma&quot;;
font-weight: bold;
color: rgb(225, 225, 225);</string>
    </property>
    <property name="text">
     <string>LR5 | Hash Table</string>
    </property>
   </widget>
   <widget class="QLabel" name="nameInputText">
    <property name="geometry">
     <rect>
      <x>450</x>
      <y>55</y>
      <width>81</width>
      <height>31</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 11pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Input:</string>
    </property>
   </widget>
   <widget class="QTextEdit" name="textOutput">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>220</y>
```

```
      <width>751</width>
      <height>341</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: rgb(50, 50, 50);
color: white;
border-radius: 15px;
font: 10pt &amp;quot;Tahoma&amp;quot;;
padding: 0px 10px 0px 10px;</string>
    </property>
   </widget>
   <widget class="QLabel" name="nameInputFile">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>55</y>
      <width>121</width>
      <height>31</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 11pt &amp;quot;Tahoma&amp;quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Hash table file:</string>
    </property>
   </widget>
   <widget class="QTextEdit" name="fileInput">
    <property name="geometry">
     <rect>
      <x>80</x>
      <y>95</y>
      <width>331</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: rgb(50, 50, 50);
color: white;
border-radius: 15px;
font: 10pt &amp;quot;Tahoma&amp;quot;;
padding: 4px 5px 0px 7px;</string>
    </property>
    <property name="html">
```

```
     <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot;
/&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body  style=&quot;  font-family:'Tahoma';  font-
size:10pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p  style=&quot;-qt-paragraph-type:empty;  margin-top:0px;  margin-bottom:0px;
margin-left:0px;       margin-right:0px;       -qt-block-indent:0;       text-
indent:0px;&quot;&gt;&lt;br /&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
   </widget>
   <widget class="QPushButton" name="pushOpenText">
    <property name="geometry">
     <rect>
      <x>450</x>
      <y>90</y>
      <width>50</width>
      <height>50</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border-image: url(:/open.png);</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="Line" name="line">
    <property name="geometry">
     <rect>
      <x>429</x>
      <y>50</y>
      <width>3</width>
      <height>100</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: rgb(62, 62, 62);</string>
    </property>
    <property name="orientation">
     <enum>Qt::Vertical</enum>
    </property>
   </widget>
   <widget class="QLabel" name="nameOutput">
    <property name="geometry">
     <rect>
```

```
      <x>20</x>
      <y>185</y>
      <width>81</width>
      <height>31</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 11pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Output:</string>
    </property>
   </widget>
  </widget>
  <widget class="QMenuBar" name="menubar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>860</width>
     <height>25</height>
    </rect>
   </property>
  </widget>
  <widget class="QStatusBar" name="statusbar"/>
 </widget>
 <resources/>
 <connections/>
</ui>
```