МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмы и структуры данных»

Тема: Хэш-таблица с цепочками

Студент гр. 8304 _____ Алтухов А.Д.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2019

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Алтухов А.Д.

Группа 8304

Тема работы: хэш-таблица с цепочками

Исходные данные:
По заданному файлу *F* (типа *file of Elem*), все элементы которого различны, построить хэш-таблицу;
Выполнить следующие действия:
а) 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.
б) 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.
Визуализировать структуры данных, алгоритмы, действия. Демонстрация должна быть подробной и понятной (в том числе сопровождаться пояснениями), чтобы программу можно было использовать в обучении для объяснения используемой структуры данных и выполняемых с нею действий.

Содержание пояснительной записки:

- Содержание
- Введение
- Заключение
- Тестирование
- Исходный код

- Список использованных материалов

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 27.02.2019

Дата сдачи реферата:

Дата защиты реферата:

Студент _____ Алтухов А.Д.

Преподаватель _____ Фирсов М.А.

АННОТАЦИЯ

В данной работе была написана программа на языке программирования C++, которая реализует такую структуру данных, как хэш-таблица. Операции над ней производятся соответственно командам, отдаваемым пользователем через реализованный графический интерфейс. Созданный функционал позволяет добавлять и удалять значения, считанные из файла или введенные непосредственно, из хэш-таблицы, визуализировать структуру и отображать происходящие с ней изменения. Также была проведена работа по повышению производительности программы, уменьшению требуемых ресурсов. Приведены примеры работы программы и полное описание исходного кода.

SUMMARY

In this paper, a program was written in the C ++ programming language, which implements a data structure such as a hash table. Operations on it are carried out according to the commands given by the user through the implemented graphical interface. The created functionality allows you to add and delete values read from a file or entered directly from a hash table, visualize the structure and display the changes that occur with it. Also, work was done to increase the productivity of the program, reduce the required resources. Examples of the program and a full description of the source code are given.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной курсовой работы является закрепление навыков работы с различными структурами данных, классами, стандартными библиотеками языка программирования C++. Используя изученные средства языка C++ и средства фреймворка Qt для создания графического интерфейса, необходимо реализовать хэш-таблицу и подробно и понятно визуализировать работу этой структуры данных.

1. ХЭШ-ТАБЛИЦА

1.1. Описание алгоритма

Хэш-таблица реализована на массиве однонаправленных списков. Применена полиномиальная хэш-функция вида

$$(s_1a^1 + s_2a^2 + \ldots + s_na^n) \; mod \; len,$$

где s_i — элемент класса, который хранит хэш-таблица, а a — некоторый коэффициент. Считывание возможно как из файла, так и из окна ввода в пользовательском интерфейсе. Считанное значение записывается в хэш-таблицу, если она еще не содержит такого элемента. Удаление происходит при указании команды delete перед необходимым элементом в окне ввода.

1.2. Используемые структуры данных и функции

`class HashTable` — реализация хэш-таблицы.

`std:: list<Elem>* arr` — массив с информацией.

`size_t len` — количество выделенной памяти.

`size_t realLen` — количество элементов в массиве.

`unsigned long long* coeffs` — массив коэффициентов.

`unsigned long long hash(Elem elem)` — функция хэширования, описанная в разделе описания алгоритма.

`void expandTable()` — функция расширения и перестройки таблицы.

`bool insert(Elem elem)` — функция вставки нового элемента в таблицу. Возвращает *true*, если вставка была произведена.

`bool remove(Elem elem)` — функция удаления элемента из таблицы. Возвращает *true*, если удаление было произведено.

`bool find(Elem elem, unsigned long long myHash)` — функция проверки наличия элемента в таблице. В случае наличия высчитанного заранее хэша элемента, он передается вторым аргументом.

`size_t getLen() const` — получение количества выделенной памяти под таблицу.

2. ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ

Интерфейс реализован с помощью фреймворка Qt. Окно программы содержит приветственный текст, кнопки открытия файла и удаления всех данных, окно ввода. Интерфейс представлен на рис. 1.

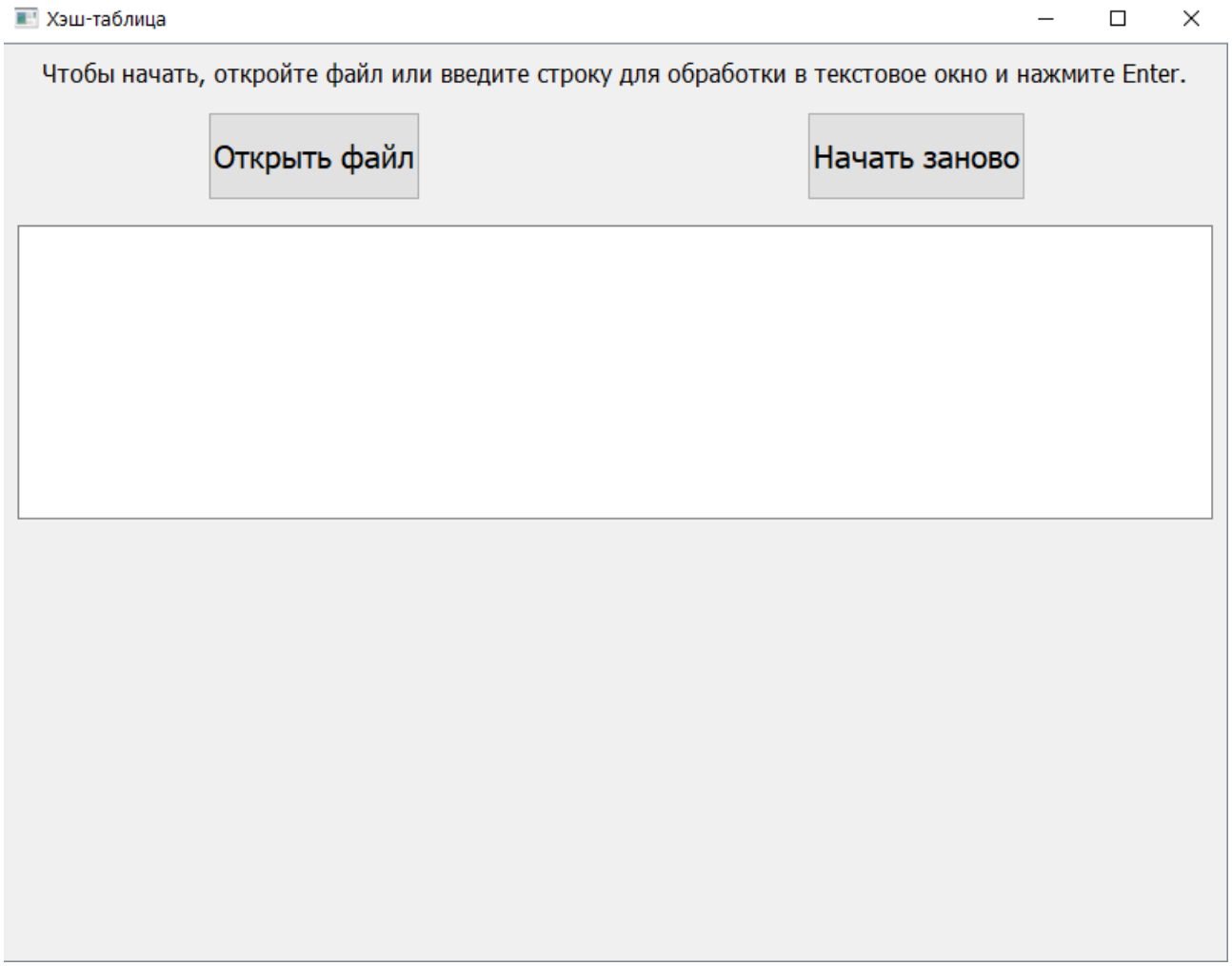

Рисунок 1 — Начальный интерфейс

Для ввода данных используется либо файл, либо непосредственный ввод. Ввод в программу по умолчанию расценивается как команда вставки написанного элемента в хэш-таблицу. Существуют два ключевых слова для явного задания команда.

insert [элемент] — вставка элемента.

delete [элемент] — удаление элемента.

После ввода каких-либо данных одним из заявленных способов, пользователю представляется визуализация текущего состояния хэш-таблицы и текстовое описание того, что с ней произошло. Последнее изменение подсвечивается. Интерфейс с отрисованной хэш-таблицей представлен на рис. 2.

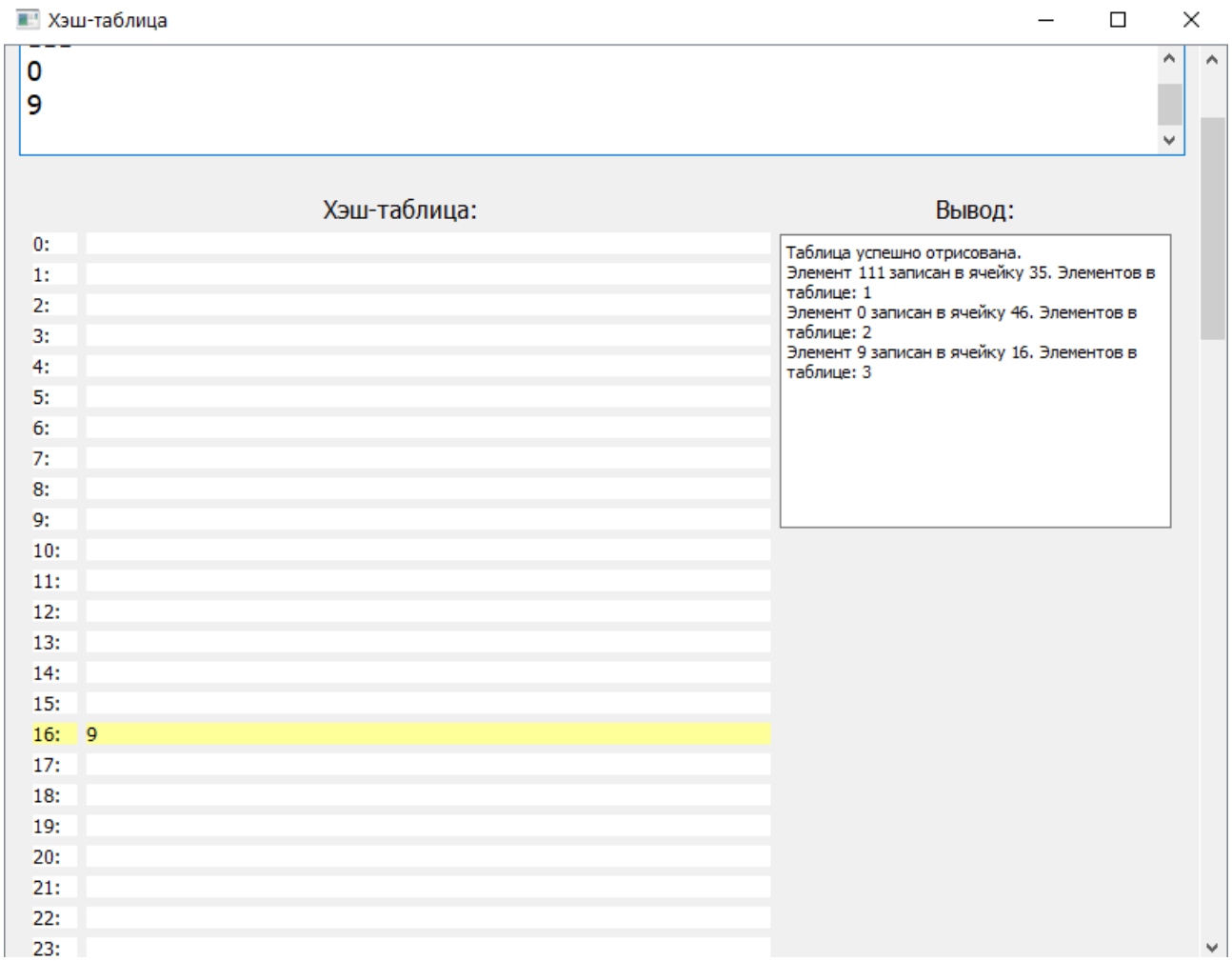

Рисунок 2 — Интерфейс с хэш-таблицей

Интерфейс реализован с помощью класса **MainWindow**, унаследованного от стандартного класса **QMainWindow**. Часть интерфейса, зависящего от хэш-таблицы, представлена в классе **HashTable**. Для управления отрисовкой служат следующие функции:

`void drawHashTable( QWidget*& tableWidget, Elem elem)` — управляет рисованием таблицы. Принимает ссылку на указатель на виджет, предназначенный для хранения отрисованной таблицы, и, как необязательный

аргумент, элемент, для случая если необходимо изменить только одну ячейку отрисованной таблицы.

`Void clearDrawnHashTable(QWidget*& tableWidget)` — уничтожает отрисованную хэш-таблицу.

`void highlight(unsigned long long i)` — управление подсветкой строк таблицы.

3. ТЕСТИРОВАНИЕ

3.1. Описание юнит-тестов

Реализованные юнит-тесты реализованы с помощью библиотеки QTest из фреймворка Qt. Они состоят из проверки работоспособности вставки элемента в таблицу и его удаления. Подробнее с кодом тестирования можно ознакомиться в приложении А.

3.2. Результаты тестирования

Результаты тестирования представлены на рис. 3.

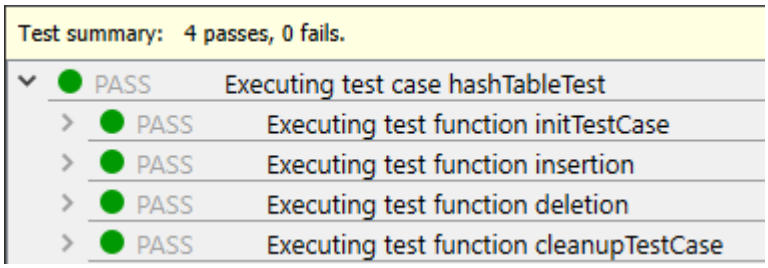

Рисунок 3 — Результаты тестирования

ЗАКЛЮЧЕНИЕ

В результате курсовой работы была написана программа на языке программирования C++, предназначенная для структурирования данных внутри хэш-таблицы и визуализацией работы алгоритма понятным для пользователя образом. Были закреплены навыки разработки программ на языке программирования C++ и использования фреймворка Qt. В ходе тестирования было выяснено, что программа работает корректно.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://doc.qt.io/
2. https://e-maxx.ru/algo/string_hashes3

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

main.cpp

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

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

    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

qtfiles.h

```
#ifndef QTFILES_H
#define QTFILES_H

#include <QMainWindow>
#include <QWidget>
#include <QApplication>
#include <QDesktopWidget>
#include <QGridLayout>
#include <QLabel>
#include <QPushButton>
#include <QFileDialog>
#include <QString>
#include <QMessageBox>
#include <QTextEdit>
#include <QScrollArea>
#include <QSpacerItem>
#include <QRadioButton>

#endif // QTFILES_H
```

HashTable.h

```
#pragma once
#include <iostream>
#include <string>
#include <vector>
#include <list>
#include "qtfiles.h"
```

```
constexpr size_t DEFAULT_LEN = 103;//стартовая длина
constexpr size_t COEFF = 31; //стартовый множитель для хэша
constexpr size_t COEFFS_COUNT = 100; //количество степеней множителя,
высчитываемых заранее
constexpr size_t PRIMES_COUNT = 24;//количество простых чисел для
расширения хэш-таблицы
constexpr size_t PRIMES[] = { 103 , 211, 331 , 449 , 587 , 709 , 853 ,
991 , 1117 , 1279 , 1433 , 1567 , 1709 ,  1873 , 2027 , 2179 , 2341 ,
2477 , 2671 , 2797 , 2963 ,        3163, 3319, 3469 };

template <class Elem>
class HashTable {
private:
    std::list<Elem>* arr; //массив с информацией
    size_t len; //количество выделенной памяти
    size_t realLen; //количество элементов в массиве
    unsigned long long* coeffs;//массив коэффициентов
    unsigned long long hash(Elem elem);
    std::string getListString(unsigned long long hash);//формирует
строку, представляющую все значения в ячейке
    QWidget* tableWidget;//главный виджет
    QGridLayout* table;//главный слой
    QLabel** numberLabels;//порядковые номера
    QLabel** textLabels;
    QLabel* hashTableText;//заголовок "хэш-таблица"
    QLabel* outputText;//заголовок "вывод"
    QTextEdit* outputBox;//окно вывода

    void expandTable();

    void out(QString output);
public:
    HashTable();
    ~HashTable();

    bool insert(Elem elem, bool withoutOutput = false);
    bool remove(Elem elem);
    bool find(Elem elem, unsigned long long myHash);
    void checkCollisions() const;
    size_t getLen() const;

    //void setStepByStepMode(bool state);
    void drawHashTable(QWidget*& tableWidget, Elem elem = "");
    void clearDrawnHashTable(QWidget*& tableWidget);
    void highlight(unsigned long long i);
    void clearHighlights();

};

template <class Elem>
```

```
HashTable<Elem>::HashTable() :arr(new
std::list<std::string>[DEFAULT_LEN]),
    len(DEFAULT_LEN),
    realLen(0),
    coeffs(new unsigned long long[COEFFS_COUNT]),
    table(nullptr),
    numberLabels(new QLabel*[DEFAULT_LEN]),
    textLabels(new QLabel*[DEFAULT_LEN]),
    outputBox(nullptr)

{ //высчитываем коэффициенты для хэша
    for (int i = 0; i < static_cast<int>(COEFFS_COUNT); i++) {
        coeffs[i] = (i > 0) ? coeffs[i - 1] * COEFF : COEFF;
    }
    for (int i=0; i<DEFAULT_LEN; i++){
        numberLabels[i] = nullptr;
        textLabels[i] = nullptr;
    }
}

template <class Elem>
HashTable<Elem>::~HashTable() {

    delete[] numberLabels;//delete содержимого не нужен, виджет удаляет
всех детей сам
    delete[] textLabels;
    delete[] coeffs;
    delete[] arr;
}
template <class Elem>
unsigned long long HashTable<Elem>::hash(Elem elem) {//полиномиальная
функция хэширования (s1*a^1 + s2*a^2 + ... + sn*a^n) mod len

    unsigned long long myHash = 0;
    for (int i = 0; i < static_cast<int>(elem.length()); i++) {
        myHash += elem[i] * coeffs[i % COEFFS_COUNT];//если нужно больше
коэффициентов, то и так сойдет
    }
    myHash %= len;
    return myHash;

}

template <class Elem>
bool HashTable<Elem>::find(Elem elem, unsigned long long myHash) {
    if (myHash == len)//если хеш еще не посчитан то передается
несуществующий
        myHash = hash(elem);
    if (!arr[myHash].empty()) {
        auto current = arr[myHash].begin();
        auto end = arr[myHash].end();
```

```
        while (current != end) {
            if (*current == elem)
                return true;
            current++;
        }
        return false;
    }
    else return false;
}

template <class Elem>
bool HashTable<Elem>::insert(Elem elem, bool withoutOutput) {

    if (elem.length()==0){
        if (!withoutOutput)
            out("Введен пустой элемент.");
        return false;
    }
    if (realLen > (0.8 * len))
        expandTable();

    unsigned long long myHash = hash(elem);
    if (find(elem, myHash)) { //а вот тут уже высчитан хэш, передадим его
чтобы не пересчитывать
        if (!withoutOutput)
            out(QString::fromStdString("Элемент " + elem + " уже
находится в таблице."));
        return false;
    }

    //std::cout << elem << "\n";
    arr[myHash].push_back(elem);
    realLen++;
    if (!withoutOutput)
        out(QString::fromStdString("Элемент " + elem + " записан в ячейку
" + std::to_string(myHash) + ". Элементов в таблице: " +
std::to_string(realLen)));
    highlight(myHash);
    return true;
}

template <class Elem>
bool HashTable<Elem>::remove(Elem elem) {
    unsigned long long myHash = hash(elem);
    if (!arr[myHash].empty()) {
        auto current = arr[myHash].begin();
        auto end = arr[myHash].end();
        while (current != end) {
            if (*current == elem) {
                arr[myHash].erase(current);
                realLen--;
```

```
                out(QString::fromStdString("Элемент " + elem + " удален
из ячейки " + std::to_string(myHash) + ". Элементов в таблице: " +
std::to_string(realLen)));
                highlight(myHash);
                return true;
            }
            current++;
        }
    }
    else {
        out(QString::fromStdString("Элемент " + elem + " не найден."));
        //std::cout << "Элемент " << elem << " не найден!\n";
        return false;
    }
}

template <class Elem>
void HashTable<Elem>::checkCollisions() const {

    std::cout << "Проверка коллизий...\n";
    for (int i = 0; i < static_cast<int>(len); i++) {
        if (!arr[i].empty()) {
            std::cout << "Занятая ячейка " << i << "\n";
            auto current = arr[i].begin();
            auto end = arr[i].end();
            size_t counter = 0;
            std::string collisions;//содержимое ячейки
            while (current != end) {
                if (!counter) {
                    collisions.clear();
                }
                collisions += *current + "\n";
                counter++;
                current++;
            }
            if (counter > 1) {
                std::cout << "Коллизия!\n" << collisions; //<< "\n";
            }
        }
    }
}

template <class Elem>
void HashTable<Elem>::expandTable() {

    //std::cout << "Расширение таблицы!\n";
    out("Таблица переполнена. Расширение таблицы.");
    clearHighlights();
    size_t oldLen = len;

    for (int i=0; i< len; i++){
```

```
            delete numberLabels[i];
            delete textLabels[i];
            numberLabels[i] = nullptr;
            textLabels[i] = nullptr;
        }
        delete[] numberLabels;
        delete[] textLabels;


        for (int i = 0; i < static_cast<int>(PRIMES_COUNT); i++) {
            if ((PRIMES[i] == oldLen) && (i + 1) <
static_cast<int>(PRIMES_COUNT)) {
                len = PRIMES[i + 1]; //за новую длину берем следующее простое
число из заданных
                break;
            }
            else if ((i + 1) >= static_cast<int>(PRIMES_COUNT))
                len += DEFAULT_LEN; //если кончились, что поделать
        }

        numberLabels = new QLabel*[len];
        textLabels = new QLabel*[len];
        for (int i=0; i<len; i++){
            numberLabels[i] = nullptr;
            textLabels[i] = nullptr;
        }

        realLen = 0; //само пересчитается в insert
        auto oldArr = arr;
        arr = new std::list<std::string>[len];

        for (int i = 0; i < static_cast<int>(oldLen); i++) {
            if (!oldArr[i].empty()) {
                auto current = oldArr[i].begin();
                auto end = oldArr[i].end();
                while (current != end) {
                    insert(*current, true);
                    current++;
                }
            }
        }

        delete[] oldArr;

        drawHashTable(tableWidget);
        out(QString::fromStdString("Расширение таблицы завершено. Количество
ячеек: " + std::to_string(len)));
}

template <class Elem>
void HashTable<Elem>::out(QString output){
```

```
    if (outputBox){
        outputBox->setText(outputBox->toPlainText()+output+"\n");
        outputBox->moveCursor(QTextCursor::End);
    }
}

template <class Elem>
void HashTable<Elem>::highlight(unsigned long long i){

    if (numberLabels[i])
        numberLabels[i]->setStyleSheet("font-size: 12px; background:
rgb(255,255,153)");
    if (textLabels[i])
        textLabels[i]->setStyleSheet("font-size: 12px; background:
rgb(255,255,153)");
}

template <class Elem>
void HashTable<Elem>::clearHighlights(){
    for (int i=0; i<len; i++){
        if (numberLabels[i])
            numberLabels[i]->setStyleSheet("font-size: 12px; background:
white");
        if (textLabels[i])
            textLabels[i]->setStyleSheet("font-size: 12px; background:
white");
    }
}

template <class Elem>
size_t HashTable<Elem>::getLen() const {
    return len;
}


template <class Elem>
std::string HashTable<Elem>::getListString(unsigned long long hash){

    unsigned long long i = hash;
    if (!arr[i].empty()) {
        std::string str = "";
        auto current = arr[i].begin();
        auto end = arr[i].end();
        while (current != end) {
            str += *current;
            current++;
            if (current != end)
                str += " --> ";
        }
        return str;
    }
```

```
    else {
        return "";
    }

}

template <class Elem>
void HashTable<Elem>::drawHashTable( QWidget*& tableWidget, Elem elem){

    bool fullRedraw = false;
    if (!table || (elem==""))
        fullRedraw = true;

    this->tableWidget = tableWidget;
    if (!table){ //инициализация содержимого
        table = new QGridLayout(tableWidget);
        table->setColumnStretch(1, 1);

        hashTableText = new QLabel("Хэш-таблица:", tableWidget);
        hashTableText->setStyleSheet("font-size: 16px");
        table->addWidget(hashTableText, 0, 0, 1, 2, Qt::AlignHCenter |
Qt::AlignVCenter);

        outputText = new QLabel("Вывод:", tableWidget);
        outputText->setStyleSheet("font-size: 16px");
        table->addWidget(outputText, 0, 2, 1, 2, Qt::AlignHCenter |
Qt::AlignVCenter);

    }
    if (!outputBox){ //инициализация текстбокса
        outputBox = new QTextEdit(tableWidget);
        table->addWidget(outputBox, 1, 2, 10, 2);
        outputBox->setText("Таблица успешно отрисована.\n");
        outputBox->setReadOnly(true);
        QSizePolicy policy(QSizePolicy::Fixed, QSizePolicy::Fixed);
        outputBox->setSizePolicy(policy);
    }

    if (fullRedraw){ //перебираем все ячейки и все рисуем
        for (unsigned i=0; i<len; i++){
            //if (!arr[i].empty()) {
                if (!numberLabels[i]){
                    numberLabels[i] = new
QLabel(QString::fromStdString(std::to_string(i)+ ": "), tableWidget);
                    numberLabels[i]->setStyleSheet("font-size: 12px;
background: white");
                    table->addWidget(numberLabels[i], i+1, 0, 1, 1);
                }

                if (!textLabels[i]){
```

```
                    textLabels[i] = new
QLabel(QString::fromStdString(getListString(i)), tableWidget);
                    if (getListString(i)!=""){
                        numberLabels[i]->setStyleSheet("font-size: 12px;
background: rgb(255,255,153)");
                        textLabels[i]->setStyleSheet("font-size: 12px;
background: rgb(255,255,153)");
                    }
                    else
                        textLabels[i]->setStyleSheet("font-size: 12px;
background: white");
                    table->addWidget(textLabels[i], i+1, 1, 1, 1);

                }
                else {
                    textLabels[i]-
>setText(QString::fromStdString(getListString(i)));
                }
            //}
        }
    }

    if (elem.length() > 0){ //если передан конкретный элемент то
перерисовываем только его ячейку
        unsigned long long myHash = hash(elem);
        if (!textLabels[myHash]){

            textLabels[myHash] = new
QLabel(QString::fromStdString(getListString(myHash)), tableWidget);
            textLabels[myHash]->setStyleSheet("font-size: 12px");
            table->addWidget(textLabels[myHash], myHash+1, 1, 1, 1);

        }
        else {
            textLabels[myHash]-
>setText(QString::fromStdString(getListString(myHash)));

        }
    }
    else{

    }
}


template <class Elem>
void HashTable<Elem>::clearDrawnHashTable(QWidget*& tableWidget){
    if (!table)
        return;
    for (int i=0; i< len; i++){
        delete numberLabels[i];
```

```
        delete textLabels[i];
        numberLabels[i] = nullptr;
        textLabels[i] = nullptr;
    }
    delete hashTableText;
    delete outputText;
    delete outputBox;
    outputBox = nullptr;
    delete table;
    table = nullptr;

}
```

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <iostream>
#include <fstream>
#include <string>
#include <algorithm>

#include "qtfiles.h"
#include "HashTable.h"
class MainWindow : public QMainWindow
{
    Q_OBJECT

private:
    QWidget* centralWidget; //почти центральный виджет
    QGridLayout* mainLayout;
    QWidget* tableWidget; //для хэштаблицы
    QLabel* mainText;
    QPushButton* openFileButton;
    QPushButton* clearAllButton;
    QTextEdit* textBox;
    QScrollArea* scroll; //скролл окна

    HashTable<std::string>* hashTable;
    void readFile(std::string fileName);
    void createErrorMessage(QString text);

private slots:
    void openFileButtonListener();
    void clearAllButtonListener();
    void enterLastLine();//ввод из текстбокса
public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
};
```

```
#endif // MAINWINDOW_H
```

mainwindow.cpp

```
#include "../Headers/mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
{

    //QRect screenSize = QApplication::desktop()->screenGeometry();
    //this->setGeometry(screenSize);
    resize(800,600);
    setWindowTitle("Хэш-таблица");
   // show();

    hashTable = nullptr;
    hashTable = new HashTable<std::string>;

    scroll = new QScrollArea(this);
    setCentralWidget(scroll);
    centralWidget = new QWidget();//запихнем в скролл попозже
    mainLayout = new QGridLayout(centralWidget);

    QSpacerItem* space = new QSpacerItem(1,10);
    mainLayout->addItem(space,4, 0, 1, 4);

    mainText = new QLabel("Чтобы начать, откройте файл или введите строку
для обработки в текстовое окно и нажмите Enter.", centralWidget);
    mainText->setStyleSheet("font-size: 16px");
    mainLayout->addWidget(mainText, 0, 0, 1, 4, Qt::AlignHCenter |
Qt::AlignTop);
   // mainText->setAlignment(Qt::AlignHCenter | Qt::AlignTop);

    space = new QSpacerItem(1,10);
    mainLayout->addItem(space,1, 0, 1, 4);

    openFileButton = new QPushButton("Открыть файл", centralWidget);
    openFileButton->setStyleSheet("font-size: 20px; height:50px");
    mainLayout->addWidget(openFileButton, 2, 0, 1, 2, Qt::AlignHCenter |
Qt::AlignTop);
    connect(openFileButton, SIGNAL(clicked()), this,
SLOT(openFileButtonListener()));

    clearAllButton = new QPushButton("Начать заново", centralWidget);
    clearAllButton->setStyleSheet("font-size: 20px; height:50px");
    mainLayout->addWidget(clearAllButton, 2, 2, 1, 2, Qt::AlignHCenter |
Qt::AlignTop);
    connect(clearAllButton, SIGNAL(clicked()), this,
SLOT(clearAllButtonListener()));
```

```
    space = new QSpacerItem(1,10);
    mainLayout->addItem(space,3, 0, 1, 4);

    textBox = new QTextEdit(centralWidget);
    mainLayout->addWidget(textBox, 4, 0, 1, 4);
    connect(textBox, SIGNAL(textChanged()), this, SLOT(enterLastLine()));
    textBox->setStyleSheet("font-size: 18px;");

    space = new QSpacerItem(1,10);
    mainLayout->addItem(space, 5, 0, 1, 4);



    tableWidget = new QWidget(centralWidget);
    mainLayout->addWidget(tableWidget, 7, 0, 1, 4);
    mainLayout->setRowStretch(7, 1);

    scroll->setWidget(centralWidget);
    scroll->setWidgetResizable(true);
}

MainWindow::~MainWindow()
{
    //все остальное в этих виджетах само удалится
    delete tableWidget;
    delete centralWidget;
    delete scroll;
    if (hashTable){
        delete hashTable;
    }
}

void MainWindow::openFileButtonListener(){

    QString fileName = QFileDialog::getOpenFileName(centralWidget,
                            QString::fromUtf8("Открыть файл"),
                            QDir::currentPath(),
                            "TXT (*.txt);;All files (*.*)");
    if (!fileName.isNull()){
       hashTable->drawHashTable(tableWidget);
       readFile(fileName.toStdString());
       hashTable->drawHashTable(tableWidget);
    }
}

void MainWindow::clearAllButtonListener(){
    if (hashTable){
        hashTable->clearDrawnHashTable(tableWidget);
        delete hashTable;
        hashTable = nullptr;
```

```
    }
    textBox->clear();
}

void MainWindow::readFile(std::string fileName){

    if (!hashTable)
        hashTable = new HashTable<std::string>;
    hashTable->clearHighlights();
    std::ifstream inputF(fileName);
    if (!inputF.is_open()) {

        createErrorMessage("Невозможно открыть файл со входными
данными");
    }

    std::string nextLine;

    while (std::getline(inputF, nextLine)) {
      //  std::cout << "Введена строка: " << nextLine << "\n";
        hashTable->insert(nextLine);
    }

    inputF.close();
}

void MainWindow::createErrorMessage(QString text){

    QMessageBox::information(centralWidget,
                QString::fromUtf8("Ошибка"),
                text);

}

void MainWindow::enterLastLine(){ //если последний символ текстбокса -
перенос, то анализируем строку и выполняем над ней нужную операцию
    QString text = textBox->toPlainText();
    if ((!text.isEmpty()) && text[text.length()-1]=='\n'){

        std::reverse(text.begin(), text.end());
        int res = text.indexOf('\n', 1);
        std::reverse(text.begin(), text.end());

        QString line;
        if (res>-1){
            line = text.right(res).section('\n', 0, -2);
        }
        else{
            line = text.section('\n', 0, -2);
        }
        if (!hashTable)
```

```
            hashTable = new HashTable<std::string>;
        hashTable->drawHashTable(tableWidget);
        hashTable->clearHighlights();
        QString mode;
        mode = line.length()>7 ? line.left(7) : "nomode ";
        if (mode == "insert "){
            hashTable->insert(line.right(line.length()-7).toStdString());
            hashTable->drawHashTable(tableWidget,
line.right(line.length()-7).toStdString());
        }
        else if (mode == "delete "){
            hashTable->remove(line.right(line.length()-7).toStdString());
            hashTable->drawHashTable(tableWidget,
line.right(line.length()-7).toStdString());
        }
        else{
            hashTable->insert(line.toStdString());
            hashTable->drawHashTable(tableWidget, line.toStdString());
        }

    }
}
```