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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: «Очереди и стеки»

Студентка гр. 8381 Бердникова А.А.

Преподаватель Жангиров Т.Р.

Санкт-Петербург

2019

Цель работы.

Познакомиться с часто используемыми на практике линейными структурами данных, обеспечивающими доступ к элементам последовательности только через её начало и конец, и способами реализации этих структур, освоить на практике использование стека, очереди и дека для решения задач.

Основные теоретические положения.

Ссылочная реализация стека и очереди в динамической памяти в основном аналогична ссылочной реализации линейных списков. Упрощение связано с отсутствием необходимости работать с текущим элементом списка. Для ссылочной реализации дека естественно использовать двунаправленный список.

Поскольку для стека, очереди и дека доступ к элементам осуществляется только через начало и конец последовательности, то эти структуры данных допускают и эффективную непрерывную реализацию на базе вектора. При этом используется одномерный массив. Непрерывная реализация ограниченной очереди на базе вектора требует, в отличие от стека, двух переменных Начало и Конец. Особенностью такого представления является наличие ситуации, когда последовательность элементов очереди по мере их добавления может выходить за границу вектора, продолжаясь с его начала (вектор имитирует здесь так называемый кольцевой буфер).

Задание

Вариант 4

Содержимое заданного текстового файла *F*, разделенного на строки, переписать в текстовый файл *G*, выписывая литеры каждой строки в обратном порядке. В решении задачи использовать *стек*.

Ход работы.

Программа написана на языке C++.

Исходный файл: main.cpp

Программа разбивает на строки переданный ей текст из файла, каждый символ через цикл поочерёдно передается в стек. После этого изъятые данные из стека записываются в новую строку и полученный текст сохраняется в новом файле.

Тестирование программы.

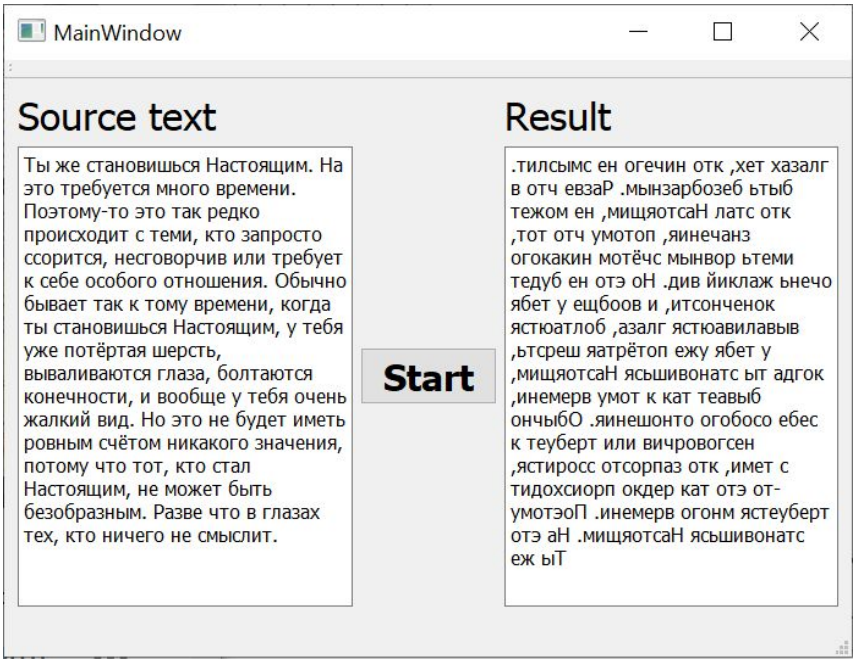
Было создано несколько тестов для проверки работы программы. (см. Приложение А).

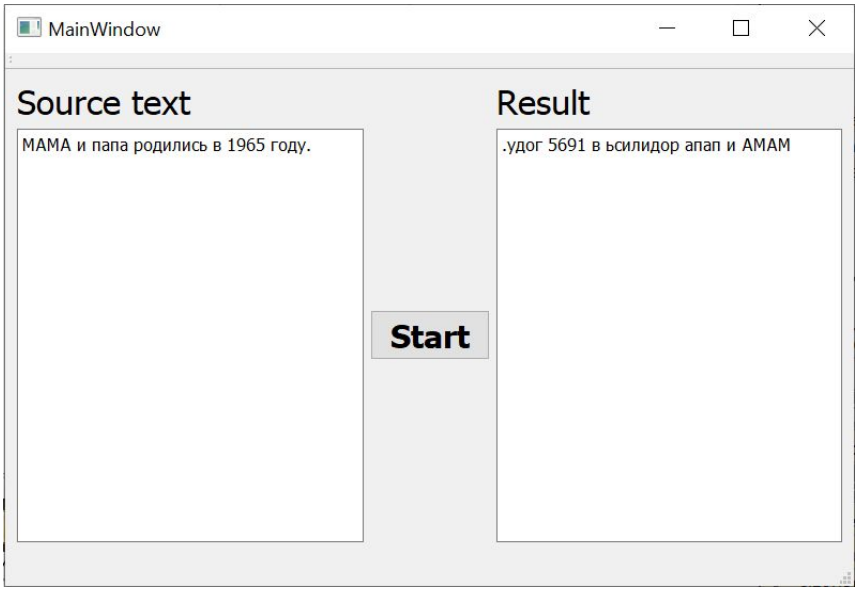
Вывод.

В ходе выполнения данной работы была написана программа с интерфейсом, которая использует такую структуру данных, как стек. Она в файле, выбранном пользователем выписывает литералы каждой строки в обратном порядке.

ПРИЛОЖЕНИЕ А

Тестирование.





ПРИЛОЖЕНИЕ Б

Исходный код программы.

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <QFileDialog>
#include <QMainWindow>
#include <QFile>
#include <QTextStream>
#include <QMessageBox>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
  Q_OBJECT

public:
  explicit MainWindow(QWidget *parent = nullptr);
  ~MainWindow();
 // bool can = false;

private slots:
  void on_pushButton_clicked();
 //  void MainWindow::on_textInput_textChanged();

private:
```

```
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

stack.h

```
#ifndef SYNTAX_H
#define SYNTAX_H

#include <cstdio>
#include <QChar>

#define NMAX 10000

struct stack {
 QChar elem[NMAX];
 int top;
};

void init(struct stack *stk);
void push(struct stack *stk, QChar f);
QChar pop(struct stack *stk);
QChar stkTop(struct stack *stk);
int gettop(struct stack *stk);
int isempty(struct stack *stk);
void stkPrint(struct stack *stk);

#endif // SYNTAX_H
```

main.cpp

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

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

  return a.exec();
}
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDebug>
#include <QStringList>
#include <QTextCodec>
#include "stack.h"

MainWindow::MainWindow(QWidget *parent) :
  QMainWindow(parent),
  ui(new Ui::MainWindow)
{
  ui->setupUi(this);
}

MainWindow::~MainWindow()
```

```
{
  delete ui;
}

void MainWindow::on_pushButton_clicked()
{

  QString readfile = QFileDialog::getOpenFileName(0, "Выберете файл для чтения", "C:/Users/nasty/source", "*.txt");

  QFile openFile(readfile);
  if(!openFile.open(QFile::ReadOnly | QFile::Text)){
     QMessageBox::information(this, "Ошибка", "Файл для чтения не удалось открыть!");
        ui->statusBar->showMessage("Ошибка");
        return;
  }
  QTextStream stream(&openFile);

  //stream.setLocale();
  QString buffer = stream.readAll();
  ui->textInput->setText(buffer);

  // Открытие файла и создание потока для записи
  QString writefile = QFileDialog::getSaveFileName(0, "Выберете файл для записи", "C:/Users/nasty/source/Desktop", "*.txt");

  QFile openFile2(writefile);
  if(!openFile2.open(QFile::WriteOnly | QFile::Text)){
```

```
        QMessageBox::information(this, "Ошибка", "Файл для записи не
удалось открыть!");
            ui->statusBar->showMessage("Ошибка");
            return;
    }
    QTextStream stream2(&openFile2);

    //разбиение на строки

    QStringList list1 = buffer.split("\n");

    for (int i=0; i<list1.size(); i++) {
        QString str = list1[i];

        stack st; //переворот
        init(&st);
        for (int i=0;i<str.size();i++) {
            push(&st, str[i]);
        }
        QString str2="";

        for (int i=0;i<str.size();i++) {
            QChar symb = pop(&st);
            str2+=symb;
        }
        ui->textOutput->append(str2);
        stream2<<str2;
    }

    openFile.flush();
```

```
    openFile.close();

    openFile2.flush();
    openFile2.close();

}
```

stack.cpp

```
#include "stack.h"

void init(struct stack *stk) {
 stk->top = 0;
}

void push(struct stack *stk, QChar f) {
 if(stk->top < NMAX) {
   stk->elem[stk->top] = f;
   stk->top++;
 } else
   printf("Стек полон, количество элементов: %d !\n", stk->top);
}

QChar pop(struct stack *stk) {
 QChar elem;
 if((stk->top) > 0) {
   stk->top--;//
   elem = stk->elem[stk->top];
   return(elem);
 } else {
```

```
  printf("Стек пуст!\n");
  return(0);
 }
}

QChar stkTop(struct stack *stk) {
 if((stk->top) > 0) {
  return( stk->elem[stk->top-1]);
 } else {
  printf("Стек пуст!\n");
  return(0);
 }
}

int gettop(struct stack *stk) {
  return(stk->top);
}

int isempty(struct stack *stk) {
 if((stk->top) == 0)    return(1);
 else return(0);
}

void stkPrint(struct stack *stk) {
 int i;
 i=stk->top;
 if(isempty(stk)==1) return;
 do {
  i--;
  printf("%f\n", stk->elem[i]);
```

```
 }while(i>0);
}
```