**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: «Стек, очередь, дек»**

Студент гр. 7381 Лукашев Р.С.

Преподаватель Фирсов М. А.

Санкт-Петербург

2018

## Задание.

4-д) Содержимое заданного текстового файла F, разделенного на строки, переписать в текстовый файл G, выписывая литеры каждой строки в обратном порядке. В задании необходимо использовать стек, реализованный на базе указателей.

## Пояснение задания.

На вход программе подаётся последовательность строк, разделенных знаком перевода строки. Необходимо выписать литеры каждой строки в обратном порядке.

## Описание алгоритма.

Последовательно считываются символы входного потока и записываются в стек. В случае, если последний записанный символ оказался символом перевода строки, происходит печать символов, записанных в стек, не считая верхнего символа перевода строки, после чего гарантируется, что стек пуст. Печатается символ перевода строки, и продолжается считывание. Если был достигнут конец файла, происходит печать оставшихся в стеке символов, и алгоритм завершает работу.

## Описание функций

| Название | Выходной параметр | Входные данные | Описание |
|---|---|---|---|
| `printLine` | `void` | `stack<char>* s` – указатель на стек, содержащий считанные на данный момент символы. | Печатает все символы из s, кроме ‘\n’. После работы данной функции можно гарантировать, что стек будет пуст. |
| `displayFileContents` | `void` | `std::istream* in` – указатель на входной поток. | Печатает все символы из входного потока in. |

## Описание структур данных

| Название | Поле | Описание |
|---|---|---|
| template <class base> class stack | struct node | Пара указателей base* head и node* tail. Узел стека, head – значение узла, tail – указатель на предыдущий элемент стека. |
| | void push() | Добавляет элемент в стек. |
| | base top() | Возвращает значение элемента, находящегося наверху стека. |
| | base pop() | Возвращает значение элемента, находящегося наверху стека, и удаляет его. |
| | bool empty() | Возвращает true, если стек пуст. |
| | void destroy() | Удаляет все элементы из стека. |
| | node* stackTop | Указатель на верхний элемент стека. |

## Тестирование.

| № теста | Исходное выражение: | Результат: |
|---|---|---|
| 1 | Hello,<br>World! | ,olleH<br>!dlroW |
| 2 | My name is Jeff.<br>Jeff is my name.<br><br>I like peanuts,<br>How about you? | .ffeJ si eman yM<br>.eman ym si ffeJ<br><br>,stunaep ekil I<br>?uoy tuoba woH |
| 3 | Heyyy, that’s pretty good! | !doog ytterp s’taht ,yyyeH |
| 4 | | |

## Тест 1.

Последовательно считываются символы ‘Hello,\nW’, в стек на данный момент записаны символы строки ‘Hello,\n’. После чего вызывается printLine, и из стека поочередно удаляются символы ‘\n’, ‘,’, ‘o’, ‘l’, ‘l’, ‘e’, ‘H’. Эти символы, не считая символа перевода строки, выводятся на экран и в файл в том же порядке, после чего, поскольку стек пуст, происходит выход из цикла, выводится символ перевода строки, и управление передается главной функции. Затем считываются символы строки ‘orld!’, в стек записываются символы строки ‘world!’, происходит выход из цикла, поскольку был достигнут конец файла, затем вызывается printLine, и из стека удаляются записанные в него символы, они выводятся в оба потока вывода, функция завершает работу, стек пуст, программа завершает работу.

## Вывод.

В процессе выполнения лабораторной работы были продуманы, созданы и реализованы на практике алгоритмы и методы работы со стеком. Стек реализован на базе указателей (ссылочная реализация). С помощью созданного стека была написана программа, выводящая в отдельный файл отзеркаленную версию текста из заданного файла. Проект создан на языке C++.

## Приложение 1. Код программы (lab3.cpp).

```
#include "stack.h"
#include <fstream>
#include <iostream>

std::ofstream out("output.txt", std::ofstream::out);
std::ofstream inter_data("interData.txt", std::ofstream::out);
std::ifstream input;

void printLine(stack<char>* s) {
	char c;

	while (!s->empty()) {
		c = s->pop();
		inter_data << "pop [" << c << ']' << std::endl;
		if (c == '\n') continue;
		std::cout << c;
		out << c;
	}

	std::cout << '\n';
	out << '\n';
}

void displayFileContents(std::istream* in) {
	std::cout << "File contents:\n";
	out << "File contents:\n";
	std::cout << "-begin-\n";
	out << "-begin-\n";
	char c;
	c = in->get();
	while (!in->eof()) {
		std::cout << c;
		out << c;
		c = in->get();
	}
	std::cout << "\n-end-\n";
	out << "\n-end-\n";
}

int main(int argc, char* argv[]) {
	if (argc == 1) {
		std::cout << "Please, open this application with input file as an
argument (drag the input file onto executable file).\n";
		return 0;
	}
	else {
		input.open(argv[1], std::ifstream::in);
		displayFileContents(&input);
```

```
		input.clear();
		input.seekg(0);
	}
	//input.open("test.txt", std::ifstream::in);
	stack<char> s;

	char c;

	c = input.get();
	while (!input.eof()) {
		s.push(c);
		inter_data << "push [" << c << ']' << std::endl;
		c = input.get();
		if (s.top() == '\n') printLine(&s);
	}

	printLine(&s);
#ifdef _WIN32
	system("PAUSE");
#endif
	return 0;
	}
```

## Приложение 2. Реализация стека (stack.h).

```
#pragma once

template <class base>
class stack {
public:

	struct node {
		base* head;
		node* tail;
		node() {
			tail = NULL;
			head = NULL;
		};
	};

	void push(base element) {
		node* prev;
		prev = stackTop;
		stackTop = new node();
		stackTop->head = new base();
		*stackTop->head = element;
		stackTop->tail = prev;
	}
```

```
	base top() {
		if (stackTop && stackTop->head)
			return *(stackTop->head);
	}

	base pop() {
		base ret = top();
		node* old = stackTop;
		stackTop = stackTop->tail;
		delete old->head;
		delete old;
		return ret;
	}

	bool empty() {
		return !stackTop;
	}

	void destroy() {
		while (stackTop) { pop(); }
	}

private:
	node * stackTop = NULL;
};
```

## Приложение 3. Содержимое файла stack.cpp.

```
#include "stack.h"
```

## Приложение 4. Содержимое файла Makefile.

```
	CODE  = ./Source/
	OBJ   = lab3.o stack.o
	EXE   = lab3
	CXX        = g++
	CFLAGS     = -Wall -Wextra -fpermissive -c -static

	all:  $(OBJ)
		$(CXX) $(OBJ) -o $(EXE)

	lab3.o: $(CODE)lab3.cpp
		$(CXX) $(CFLAGS) $(CODE)lab3.cpp

	stack.o: $(CODE)stack.cpp $(CODE)stack.h
```

```
	$(CXX) $(CFLAGS) $(CODE)stack.cpp

clean:
	rm $(OBJ) $(EXE)
```

## Приложение 5. Содержимое файла checker.sh.

```
#!/bin/bash
touch checker_res.txt
cp /dev/null checker_res.txt
for cfile in Tests/Correct/*; do
    echo "running test: \"Tests/Correct/$cfile\" ";
     echo "correct test $cfile:" >>checker_res.txt;
    ./lab3 $cfile >>checker_res.txt;
     echo " ">>checker_res.txt;
done;
```