

**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: Рекурсия**

Студент гр. 8304

\_\_\_\_\_

Мухин А. М.

Преподаватель

\_\_\_\_\_

Фирсов М. А.

Санкт-Петербург

2019

## Цель работы.

Получить опыт работы со стэком, реализовав его на основе массива.

## Задание.

Содержимое заданного текстового файла  $F$ , разделенного на строки, переписать в текстовый файл  $G$ , выписывая литеры каждой строки в обратном порядке.

Вариант 4-в.

## Выполнение работы.

В заголовочном файле объявлен класс `Stack`. В частных полях которого есть указатель типа `type` и порядковый номер текущего элемента. Методы описаны в открытой части класса. Конструктор выделяет память в указатель типа `type`. Деструктор удаляет содержимое этого указателя и зануляет его ссылку.

Метод `void push(type)` записывает в массив элемент типа `type`.

Метод `type pop()` возвращает верхний элемент стэка.

Метод `bool isEmpty()` возвращает истину, если стэк пуст.

Функция `void Reverse(Stack<type>&, std::ifstream&, std::ofstream&, char&)` производит чтение из одного файла и запись в другой согласно заданию.

Функция `int main(int, char*)` выполняет считывание файла первый раз, для того, чтобы выяснить количество строк и длину максимальной строки, для дальнейшей инициализации стэка. После этого считывания, поток возвращается в начало файла и в цикле `for`, количество строк в файле раз объявляется переменная типа `Stack` и вызывается функция `void Reverse(Stack<type>&, std::ifstream&, std::ofstream&, char&)`. В конце происходит закрытие потока ввода и вывода. Стоит отметить, что поток вывода открывался с флагом `std::ios::app`, благодаря которому, запись в файл происходила в его конец.

Алгоритм рекурсивной записи строки из одного файла в обратном

порядке в другой файл реализован в функции `void Reverse(Stack<type>&, std::ifstream&, std::ofstream&, char&)`. Если текущий символ потока это не перенос строки, тогда мы его записываем в стэк и снова вызываем эту функцию. Если это символ переноса строки, тогда мы достаём последний элемент из стэка и записываем его в поток вывода.

Разработанный программный код см. в приложении А.

### Тестирование.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	kjhgvbjhlb nljhnk nluikhn	blhjbvghjk knhjln nhkiuln
2.	sfgsdfgsdf gs dfgsdsgsdfgs fgsdfg  sdfgs sdf s dfgsdfgsdfg	fdsgfdsgfs sg sgfdsgsdsgfd gfdsgf  sgfds fds s gfdsgfdsgfd
3.	sfgsdfgsdf gssdfgsdfgsdfg dfgsdsgsdfgs fgsdfg fgsdfgsdfgsdfg sdfgssdfgsdfgsdfg sdf sgfsfgsfgsdfgsdf	fdsgfdsgfs gfdsgfdsgfdssg sgfdsgsdsgfd gfdsgf gfdsgfdsgfdsgf gfdsgfdsgfdssgfd fds fdsgfdsgfsgfsfgs

	dfgsdfgsdfg	gfdsgfdsgfd
4.		
5.	a b c	a b c
6.	4a1234 bfadsfasef cflksufdhgnldf sd;ifuhglsdf adflgjhbsdljfg sdflgisbdfnl sdfmglsidnfgls fngouayhbdfgklj adfmglisdfhunlgs alfnidugnaslifd afmgiluafhnglaidf afdnlgiauabnldfg dflnugnsolf	4321a4 fesafsdafb fdlnghdfusklf fdslghufi;ds gfjldsbbhjglfda lnfdbsiglfd slgfndsilgmfd jlkghdbhyauognf sglnuhfdisglmfda dfilsangudinfla fdialgnhfauligmfa gfdlnbauiglndfa flosngunlfd

## Выводы.

Ознакомились с основными понятиями и приёмами рекурсивного программирования, получили навыки программирования рекурсивных процедур и функций на языке программирования C++ и создания стэка на базе вектора. Научились работать с потоками ввода и вывода и создавать шаблоны.

## ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.hpp

```

#include "lab3.h"
#include <fstream>

template <class type>
Stack<type>::Stack(size_t max_size) {
    static int spare_memory = 2;
    st = new type[max_size + spare_memory];
    len = 0;
}

template <class type>
Stack<type>::~~Stack() {
    delete [] st;
    st = nullptr;
}

template <class type>
void Stack<type>::push(type elem) {
    st[len] = elem;
    len++;
}

template <class type>
type Stack<type>::pop() {
    return st[--len];
}

template <class type>
bool Stack<type>::isEmpty() {
    return len == 0;
}

template <class type>
void Reverse(Stack<type>& stack, std::ifstream& in,
std::ofstream& out, char& current_elem) {
    if (current_elem != EOF) {
        if (current_elem != '\n') {
            stack.push(current_elem);
            in.get(current_elem);
            if (!in.eof())
                Reverse(stack, in, out, current_elem);
        }
        if (!stack.isEmpty()) {
            out.put(stack.pop());
            return;
        }
    }
}

```

```
    }  
}
```

### Название файла: lab3.h

```
#pragma once  
#include <cstdlib>  
#include <fstream>  
  
template <class type>  
class Stack{  
    private:  
        type* st;  
        size_t len;  
    public:  
        Stack(size_t max_size);  
        ~Stack();  
        void push(type elem);  
        type pop();  
        bool isEmpty();  
};
```

### Название файла: main.cpp

```
#include <iostream>  
#include <fstream>  
#include "lab3.h"  
#include "lab3.hpp"  
  
int main(int argc, char* argv[]) {  
    std::ifstream in(argv[1]);  
    std::ofstream out(argv[2], std::ios::app);  
    int max_len = 0;  
    int count_string = 0;  
    std::string current_line;  
    while (getline(in, current_line)) {  
        count_string++;  
        if (current_line.length() > max_len)  
            max_len = current_line.length();  
    }  
    in.clear();  
    in.seekg(0, std::ios::beg);  
    char current_elem = '1';  
    for (int i = 0; i < count_string; i++) {  
        Stack<char> stack(max_len);  
        in.get(current_elem);  
        if (current_elem != EOF) {
```

```
        Reverse(stack, in, out, current_elem);
        out.put('\n');
    }
}
in.close();
out.close();
return 0;
}
```