

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: «Стеки и очереди»

Студент гр. 7382

Глазунов С.А.

Преподаватель

Фирсов М.А.

Санкт-Петербург

2018

Задание. (4В)

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

Описание алгоритма.

Чтобы вывести строку в обратном порядке был использован стек. При вводе строки каждый символ «пушился» в стек, а после при выводе использовалась функция «pop». Результатом была обратная строка потому что стек работает по принципу FILO (First In Last Out).

Описание функций и структур данных.

Class MyStack является классом стека, который реализует шаблонный стек.

```
void Mystack<T>::resize()
```

Функция увеличивает размер массива, который и реализует контейнер для стека.

```
void Mystack<T>::push(T value)
```

Функция «пуш» вставляет элемент в конец стека, то есть записывает в ячейку массива с индексом length (переменная, которая показывает длину массива)

```
bool Mystack<T>::is_empty()
```

Функция проверяет стек на : пуст ли он или нет. Если пуст, то функция возвращает true.

```
int Mystack<T>::size()
```

Функция возвращает длину массива (стека), то есть length, о котором говорилось выше.

`T Mystack<T>::top()`

Функция возвращает верхний элемент стека, не удаляя элемент из него.

`T Mystack<T>::pop()`

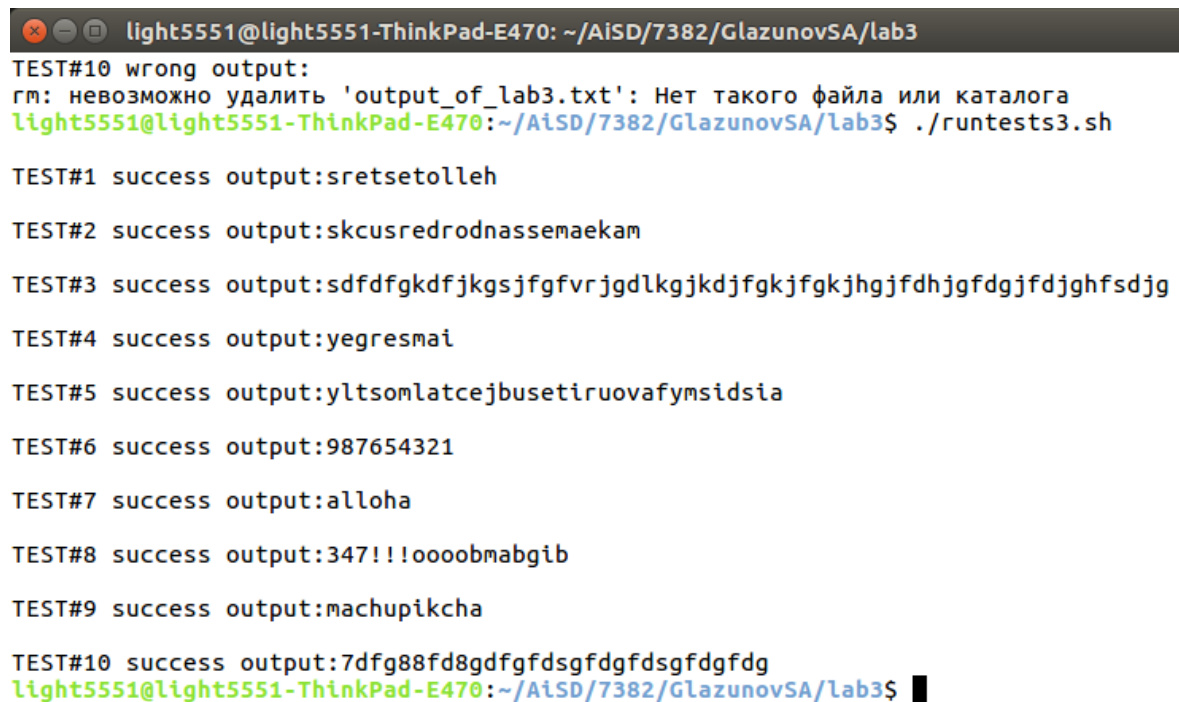
Функция возвращает верхний элемент стека, удаляя его из стека.

Благодаря этой функции и появляется обратная строка.

Тестирование.

Для тестирования использовался скрипт, унаследованный из второй лабораторной работы, перенесённый практически без изменений. В директории проекта папка с тестами содержит в себе 10 тестов. Ниже представлен рисунок с результатами тестирования. Данная программа не имеет некорректных тестов, потому что задание требует любую строку записать в обратном порядке, то есть любой ввод нас устроит.

Рассмотрим тест №4. На вход подается: «iamsergey». Программа пушит каждый элемент в стек, а потом выводит по одному символу на консоль и получаем: “yegresmai”



```
light5551@light5551-ThinkPad-E470: ~/AiSD/7382/GlazunovSA/lab3
TEST#10 wrong output:
gm: невозможно удалить 'output_of_lab3.txt': Нет такого файла или каталога
light5551@light5551-ThinkPad-E470:~/AiSD/7382/GlazunovSA/lab3$ ./runtests3.sh

TEST#1 success output:sretsetolleh
TEST#2 success output:skcusredrodnassemaekam
TEST#3 success output:sdfdfgkdfjkgsjfgfvrjgdlkgjkdjfgkjfgkjhgjfdhjgfdgjfdjghfsdjg
TEST#4 success output:yegresmai
TEST#5 success output:yltsomlatcejbusetiruovafymsidsia
TEST#6 success output:987654321
TEST#7 success output:alloha
TEST#8 success output:347!!!oooobmabgib
TEST#9 success output:machupikcha
TEST#10 success output:7dfg88fd8gdfgfdsgfdgfdsgfdgfdg
light5551@light5551-ThinkPad-E470:~/AiSD/7382/GlazunovSA/lab3$
```

Выводы.

В процессе выполнения лабораторной работы были изучены возможности шаблонов в языке C++, программная реализация такой структуры данных, как стек. Закреплены навыки работы с системой контроля версий, bash-скриптами. Систематизированы навыки полуавтоматического тестирования программ.

ПРИЛОЖЕНИЯ А

```
#include <algorithm> // for exit

#include <iostream>

#include <cstring> //for memset

#define START_SIZE_OF_STACK 10

template <typename T>

class Mystack

{

public:

    Mystack(int); //these are public fields :)

    Mystack();

    ~Mystack();

    bool is_empty();

    int size();

    T top();

    T pop();

    void push(T);

private:

    //these are private

    T *array;

    int size_of_array,

        length;

    void resize();

};

template <typename T>
```

```

void Mystack<T>::resize(){
    //this is stupid function in style C because we cant use vector!!!

    //this func increase array...

    T* time_array = new T[size_of_array];

    std::memcpy(time_array,array,( size_of_array )*sizeof(T));

    delete [] array;

    array = new T[size_of_array + START_SIZE_OF_STACK ];

    std::memcpy(array,time_array,( size_of_array )*sizeof(T));

    delete [] time_array;

    size_of_array += START_SIZE_OF_STACK;
}

```

```

template <typename T>

```

```

Mystack<T>::~~Mystack()

```

```

{
    delete [] array;
}

```

```

template <typename T>

```

```

Mystack<T>::Mystack(int size)

```

```

{
    size_of_array = size > 0 ? size : START_SIZE_OF_STACK; //if size > 0 - it's,else size
equal START_SIZE_OF_STACK

    array = new T[size_of_array];

    length = -1;
}

```

```

template <typename T>

```

```

Mystack<T>::Mystack()

```

```

{
    size_of_array = START_SIZE_OF_STACK;
    array = new T[size_of_array];
    length = -1;
}

template <typename T>
void Mystack<T>::push(T value)
{
    if(length==size_of_array-1)
    {
        resize();
    }

    array[++length]=value;//++ is prefix because in the beginning length == -1
    and we cant use this like index of array
}

template <typename T>
bool Mystack<T>::is_empty()
{
    if(length<=-1)
        return true;
    return false;
}

template <typename T>
int Mystack<T>::size()
{
    return length;
}

```

```

template <typename T>
T Mystack<T>::top()
{
    if(!is_empty())
        return array[length];

    std::cout<<std::endl<<"ERROR!!!"<<std::endl;
    exit(0);
}

```

```

template <typename T>
T Mystack<T>::pop()
{
    T total=top();
    length--;
    return total;
}

```

```

#include <iostream>

#include <fstream>

#include "stack.cpp"

//#define TEST

//#define SCRIPT

int main(){

    Mystack<char> stack(10);

    std::string text;

    #ifndef TEST

    #ifndef SCRIPT

```



```

        std::cout<<"this is stack.Enter...(for example: ados->stack-
>soda)"<<std::endl;

    #endif

    #endif

    getline(std::cin,text);


    #ifdef TEST

                                #ifndef SCRIPT

                                std::cout<<"this is stack.Enter...(for example: ados-
>stack->soda)"<<std::endl;

                                #endif

                                std::cout<<"START PUSHING!!!"<<std::endl;

                                #endif

                                for(int i=0;i<text.size();i++)

                                {

                                        #ifdef TEST

                                                std::cout<<"PUSH"<<"  "<<text[i]<<"

(" "<<i<<"")<<std::endl;

                                                #endif

                                                stack.push(text[i]);                                //add elements in stack

                                }

                                #ifdef TEST

                                std::cout<<"_____\\n"<<"STOP PUSHING
AND START POPING :)"<<std::endl;

                                #endif

                                for(int i=0;i<text.size();i++)

                                {

```

```

        #ifdef TEST

            std::cout<<"POP"<<" ";

            std::cout<<"TOP:"<<stack.top()<<" ";

            std::cout<<"SIZE:"<<stack.size()<<std::endl;

            stack.pop();

        #else

            #ifndef SCRIPT

                std::cout<<stack.pop();

            #else

                std::ofstream

file("output_of_lab3.txt",std::ios::app);

                file << stack.pop();

                //std::ofstream("1.txt", ios::app);

                //f<<"the last string";

                file.close();

            #endif

        #endif

    }

    std::cout<<std::endl;

    return 0;

}

```