

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

ОТЧЕТ
по практической работе №3
по дисциплине «Алгоритмы и структуры данных»
Тема: Линейные структуры данных

Студент гр. 7382

Находько А.Ю.

Преподаватель

Фирсов М.А.

Санкт-Петербург

2018

Цель работы.

Познакомиться с линейными структурами данных, обеспечивающими доступ к элементам последовательности только через её начало и конец, и способами реализации этих структур. Освоить на практике работу стека и очереди.

Задание.

Вариант 2. Содержимое заданного текстового файла F , разделенного на строки, переписать в текстовый файл G , перенося при этом в конец каждой строки все входящие в нее цифры (с сохранением исходного взаимного порядка как среди цифр, так и среди остальных литер строки).

Пояснение задачи.

Требуется чтобы программа осуществляла поиск строк для преобразования в файле input.txt. Осуществляла перенос всех цифр в конец строки с сохранением взаимного порядка, а остальные символы переносила в начало также с сохранением исходного взаимного порядка. Результат программа должна записывать в файл output.txt

Основные теоретические положения.

Стеком называется упорядоченный набор элементов, в котором размещение новых и удаление существующих происходит с одного конца, называемого вершиной.

В стеке реализуется дисциплина обслуживания LIFO:

- LAST — последний
- INPUT — вошел
- FIRST — первый
- OUTPUT — вышел

Различают аппаратный и программный стек.

Аппаратный стек используется для хранения адресов возврата из функций и их аргументов. Программный стек – это пользовательская модель (структура) данных.

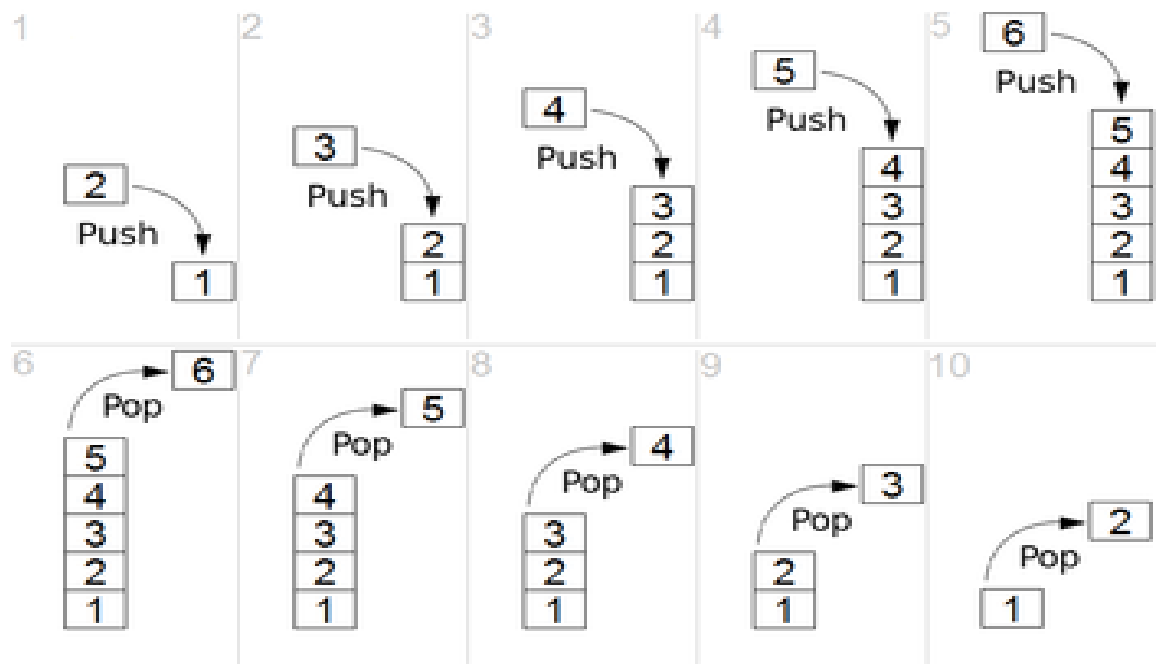


Рисунок 1. Работа стека

Очередью называется упорядоченный набор элементов, которые могут удаляться с её начала и помещаться в её конец.

Очередь организована, в отличие от стека, согласно дисциплине обслуживания FIFO:

- FIRST — первый
- INPUT — вошел
- FIRST — первый
- OUTPUT — вышел

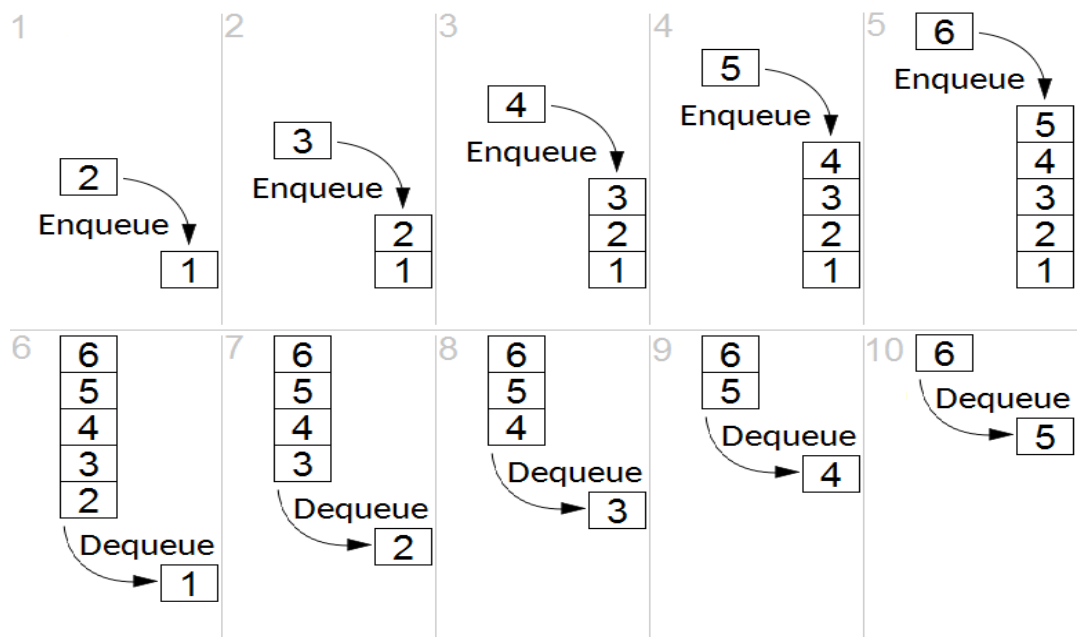


Рисунок 2. Работа очереди

Существует несколько способов реализации очереди и стека:

- с помощью одномерного массива;
- с помощью связанного списка;
- с помощью класса объектно-ориентированного программирования.

Выполнение работы.

- 1) Создадим элемент `q_num` типа `Queue`.
- 2) Создадим строку `s`, в которую будем записывать строку из файла `input.txt` с помощью функции `getline` для осуществления преобразования.
- 3) Создадим цикл относительно переменной `i=0`, с шагом `i++` и условием пока можем взять строку для выполнения преобразования.
- 4) Создадим цикл относительно переменной `j=0`, с шагом `j++` и условием пока строка `s` не закончится.
- 5) Если встречаем элемент, номер `x` которого расположен в диапазоне $47 < x < 58$ в таблице ASCII – значит этот элемент цифра. С помощью функции `enqueue` добавляем этот элемент в очередь.
- 6) Если номер элемента `x` находится не в данном диапазоне, значит этот элемент является не цифрой. Производим его вывод на консоль и также его вывод в файл `output.txt`.

- 7) Когда все символы строки `s` будут проанализированы, произойдёт выход из описанного ранее цикла относительно переменной `j`, и впоследствии передача очереди с цифрами в функцию `print_num`, аргументами которой являются: очередь цифр `q_num`, файл в который производим вывод результата `output.txt`.
- 8) Пока очередь не окажется пустой, будем производить извлечение цифр из очереди по одной с первого элемента. Записываться они будут после символов в консоли и файле `output.txt`.
- 9) Выход из цикла относительно переменной `i` произойдёт когда в файле `input.txt` закончатся строки для анализа.

Описание функций.

`Queue` – класс, который содержит написанные функции для очереди, максимальный размер очереди, массив символов, начало и размер очереди.

`Enqueue` – функция, которая выполняет проверку на полноту очереди, добавляет новый элемент на первое место и увеличивает на единицу размер очереди. Если очередь переполнена – выход из функции.

`Dequeue` – функция, которая выполняет проверку на пустоту очереди, возвращает первый элемент в программу, устанавливает второй элемент на первую позицию и уменьшает на единицу размер очереди. Если очередь пуста – выход из функции.

`IsEmpty` – функция, которая проверяет очередь на пустоту.

`IsFull` – функция, которая проверяет очередь на полноту.

`Print_num` – функция, которая производит вывод цифр в конец строки из очереди типа `Queue`.

Тестирование.

Таблица тестирования программы.

№ теста	Входные данные:	Результат:
1	897947982hfvhurehfr4923980432ho efhoerhore	hfvhurehfrhoefhoerhore8979479824 923980432
2	34fjfd3	fjfd343
3	rhogrhierije	rhogrhierije
4	8239478693247832	8239478693247832
5	jrjoerojirgho234980980432gjkjkrigg	jrjoerojirghogjkjkrigg234980980432
6	rfjhoerhore787897	rfjhoerhore787897
7	9823497hofehjef8824hdfdvhjo44hh gtjej2	hofehjefhdfdvhjohhgtjej9823497882 4442
8	pjvp4nper	pjvpnper4
9	iienvuejehoeroerreenv983778328732kf vhjvjehorhorhore92389823489234	iienvuejehoeroerreenvkfvhjvjehorhorh ore98377832873292389823489234

Иллюстрации работы тестирования:

```
Строка №1
Строка до измененеия:
897947982hfvhurehfr4923980432hoefhoerhore
Строка после изменения:
hfvhurehfrhoefhoerhore8979479824923980432
Строка №2
Строка до измененеия:
34fjfd3
Строка после изменения:
fjfd343
Строка №3
Строка до измененеия:
rhogrhierije
Строка после изменения:
rhogrhierije
```

```

Строка №4
Строка до измененеия:
8239478693247832
Строка после изменения:
8239478693247832
Строка №5
Строка до измененеия:
jгjoegojirgho234980980432gjkjkrigg
Строка после изменения:
jгjoegojirghogjkjkrigg234980980432
Строка №6
Строка до измененеия:
rfjhoerhoge787897
Строка после изменения:
rfjhoerhoge787897
Строка №7
Строка до измененеия:
9823497hofehjef8824hdfdvjhjo44hhgtjej2
Строка после изменения:
hofehjefhdfdvjhjohhgtjej98234978824442
Строка №8
Строка до измененеия:
rjvr4nper
Строка после изменения:
rjvrnper4
Строка №9
Строка до измененеия:
iievuejehoegoergreev983778328732kfvhjvjehorhorhore92389823489234
Строка после изменения:
iievuejehoegoergreevkfvhjvjehorhorhore98377832873292389823489234

```

Описание алгоритма

На примере теста №2.

- 1) С помощью функции `getline` отделяем из файла строку `s="34fjfd3"`.
- 2) Начинаем анализацию данной строки почленным перебором символов из данной строки.
- 3) Элементы данной строки имеют следующие номера в таблице ASCII: "3" – 51; "4" – 52; "f" – 102; "j" – 106; "f" – 102; "d" – 100; "3" – 51. Согласно циклу относительно переменной `j`, мы добавим элементы с номерами `x`, расположенными в диапазоне `47<x<58` в очередь – т.к. т.к. эти элементы являются цифрами.
- 4) В очереди у нас находятся элементы 343, именно в таком порядке.

- 5) Происходит вызов функции `print_num`, в которую в качестве аргументов передаём полученную очередь и файл, в который будем записывать окончательный ответ.
- 6) Пока очередь не окажется пустой будем доставать из неё цифры – по одной. Извлечённые цифры будем выводить на консоль и файл вывода. Т.е. цифры будут записываться после букв `fjfd`.
- 7) В качестве результата получим строку `fjfd343`, которую можем увидеть на консоли, а также в файле `output.txt`.

Выводы.

В результате выполнения лабораторной работы ознакомился с линейными структурами данных, обеспечивающими доступ к элементам последовательности только через её начало и конец, и способами реализации этих структур. Освоил практически работу стека и очереди. Написал функции для корректной работы очереди.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

```
#include <iostream>
#include <fstream>
#include <string>
#include <stdlib.h>

using namespace std;

class Queue { //Класс очередь
public:
    Queue(); //Инициализация очереди
    void enqueue(const char&); //Добавление элемента в очередь
    char dequeue(); //Удаление первого элемента из очереди
    int isEmpty(); //Проверка очереди на пустоту
    int isFull(); //Проверка очереди на переполненность
private:
    static const int maxSize=300; //Максимальный размер очереди
    char arr[maxSize]; //Массив для хранения найденных чисел
    int start, size; //Начало и конец очереди
};

Queue::Queue() : size(0), start(0)
{}

void Queue::enqueue(const char &a)
{
    if (isFull()!=0) //Если очередь не полна, значит можем вставить элемент
    {
        arr[(start+size)]=a; //Ставим на ближайшее свободное место, после этого
        увеличиваем размер очереди на одну единицу
        ++size;
    } else
    {
        printf("Очередь переполнена!\n");
        exit(1);
    }
}

char Queue::dequeue()
{
    if (isEmpty()!=0) //Если очередь не пуста, значит можем достать из неё
    элемент
    {
        char b = arr[start]; //Присвоим b первый элемент очереди, который будем
        извлекать
        start=(++start); //Теперь первым элементом станет второй элемент
        очереди
        --size; //Уменьшаем размер очереди и возвращаем элемент b
        return b;
    }
}
```

```

    } else {
    printf("Стек пуст! Невозможно вернуть элемент!\n");
    exit(1);
    }
}

int Queue::isEmpty()
{
if (size==0) return 0; //Если размер равен 0
return 1;
}

int Queue::isFull()
{
if(size==maxSize) return 0; //Если размер равен максиммально
возможному
return 1;
}

//Функия, которая выполняет вывод цифр из очереди на консоль в конец
строки и запись их в текстовый файл выхода
void print_num(Queue &q_num, ofstream &fout)
{
char c;
while (q_num.isEmpty()!=0) //Если очередь текущая очередь цифр не
пуста
{
с = q_num.dequeue(); //Достанем из неё первый элемент и выведем его в
конец строки в консоли и текстового файла
fout << c;
cout << c;
}
fout << endl;
cout << endl;
}

int main()
{
Queue q_num; //Очередь цифр, извлечённых из первоначальной строки
string s; //Строка которую будем преобразовывать
ifstream fin("input.txt"); //Файл содержимое которого изменяем
ofstream fout("output.txt"); //Файл в который записываем результат
работы программы

for (int i = 0; (getline(fin, s)); i++) //С помощью функции getline разбиваем
содержимое файла input.txt на строки для обработки
{
cout << "Строка №" << i+1 << endl;
cout << "Строка до измененеия:" << endl << s << endl;
cout << "Строка после изменения:" << endl;
for (int j = 0; j < s.length(); j++) //Цикл прохода строки для её

```

```
поэлементной проверки
{
if (((int)s[j]>47) && ((int)s[j]<58)) //Если номеру текущего символа
соответствует цифра согласно таблице ASCII
{
q_num.enqueue(s[j]); //Добавляем эту цифру в очередь
}
else
{
cout << s[j]; //Если элемент не оказался цифрой, то выводим его на
консоль, либо записываем в файл выхода
fout << s[j];
}
}
print_num(q_num, fout); //Передаём очереди цифр в функцию print_num
для их вывода
}
}
```