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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: Очередь

Студент гр. 7383 _____ Бергалиев М.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2018

ОГЛАВЛЕНИЕ

1. Цель работы

Цель работы: познакомиться со структурой и реализацией очереди и использованием ее в практических задачах на языке программирования C++.

Формулировка задачи: 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).

2. Реализация задачи

В функции main выводится приглашение ввести название входного файла или выйти из программы вводом пустой строки. Если такого файла нет, то выводится сообщение о ошибке и предлагается ввести название файла вновь. Далее предлагается ввести название выходного файла. Если к файлу нет доступа на запись, то выводится сообщение об ошибке и программа начинается вновь. Создаются объекты типа std::istream и std::ostream. После вызывается функция rewrite, которая перезаписывает данные из входного файла в выходной файл, перемещая числа из строк в их конец. Далее процесс повторяется вновь.

Переменные, используемые в функции main:

- ifile, ofile — файловые буфера, участвующие в создании объектов типа istream и ostream соответственно.
- ifilename, ofilename — входной и выходной файлы соответственно.
- fin, fout — входной и выходной потоки.

Функция rewrite принимает на вход входной и выходной потоки. Создаются две очереди для цифр и других символов. Пока входной файл не закончится, оттуда считываются строки. Каждая строка обрабатывается посимвольно: цифры добавляются в одну очередь, остальные символы — в другую. Когда строка закончилась, символы из очередей записываются в выходной файл, причем символы из очереди цифр записываются последними.

Переменные, используемые в функции rewrite:

- numbers, other — очереди цифр и других символов соответственно.
- input — обрабатываемая строка.

Шаблонный класс Queue с шаблонным параметром T (тип хранимых элементов) представляет из себя циклическую очередь на базе динамического массива. Класс содержит следующие поля:

- vsize — длина динамического массива(вектора).
- qsize — длина очереди.
- vstart — указатель на начало массива.
- qstart — указатель на начало очереди.
- qend — указатель на конец очереди.

Методы класса Queue:

- Конструктор, принимающий длину массива. Выделяется память под массив, все указатели устанавливаются на начало массива.
- push — принимает объект типа T, который добавляется в конец очереди по указателю qend, qend сдвигается вправо с учетом цикличности очереди. Если очередь переполнена, то сначала увеличивается в два раза размер массива, а после добавляется элемент.
- resize — принимает новый размер массива. Создается новый массив и копируется в него содержимое старого массива, после чего память под старый массив высвобождается.
- isEmpty — возвращает true, если очередь пуста.
- pop — возвращает первый элемент из очереди по указателю qstart, он сдвигается вправо с учетом цикличности очереди.

3. Тестирование

Программа была собрана в компиляторе G++ с ключом -std=c++14 в OS Linux Ubuntu 16.04 LTS.

В ходе тестирования ошибки не были найдены.

Тестовые случаи представлены в приложении А.

4. Вывод

В ходе работы были получены навыки работы с очередью. Был реализован шаблонный класс, представляющий из себя циклическую очередь на базе массива. Очередь является удобной структурой данных для хранения элементов в определенном порядке.

ПРИЛОЖЕНИЕ А.

ТЕСТОВЫЕ СЛУЧАИ

Таблица 1 — Тестовые случаи

Входные данные	Результат
Файл F: asdfkkj 25 a;fd 80 askl maskl smm 98 kldsl sfd 4sffds 2256 fds 235 2fs 54sdf df5f	asdfkkj a;fd askl maskl smm kldsl258098 sfd sffds fds fs sdf dff422562352545
24324 asdsdf 1a3b5c7d9e	24324 asdsdf abcde13579
((24 ((25 82) (45 ((32 95) (53 85))))) (24 ((45 ((23 69) (34 45))) (26 46))))	(((( ) ((( ) ( ))))) ((((( ) ( ))) ( ))))2425824532955385244523693445 2646

ПРИЛОЖЕНИЕ Б.

ИСХОДНЫЙ КОД ПРОГРАММЫ

Makefile:

```
all: main.cpp queue.hpp
	g++ main.cpp -std=c++14 -o queue
```

main.cpp:

```
#include <iostream>
#include <fstream>
#include "queue.hpp"

void rewrite(std::istream &fin, std::ostream &fout){
    Queue<unsigned char> numbers(4), other(10);
    std::string input;
    while(!fin.eof()){
      getline(fin, input);
      for(auto i : input)
          if(i > '9' || i < '0')
            other.push(i);
          else
            numbers.push(i);
      while(!other.isEmpty())
          fout << other.pop();
      while(!numbers.isEmpty())
          fout << numbers.pop();
      fout << std::endl;
    }
}

int main(){
    std::filebuf ifile, ofile;
    std::string ifilename, ofilename;
    while(true){
      std::cout << "Enter input file name or nothing to exit: ";
      getline(std::cin, ifilename);
      if(ifilename == "")
          break;
      if(!ifile.open(ifilename, std::ios::in)){
          std::cout << "Incorrect file, try again" << std::endl;
          ifile.close();
          continue;
      }
      std::cout << "Enter output file name: ";
      getline(std::cin, ofilename);
      if(!ofile.open(ofilename, std::ios::out)){
          std::cout << "Incorrect file, try again" << std::endl;
          ifile.close();
          ofile.close();
          continue;
      }
      std::istream fin(&ifile);
      std::ostream fout(&ofile);
```

```
      rewrite(fin, fout);
      ifile.close();
      ofile.close();
    }
    return 0;
}
```

queue.hpp:

```
#pragma once

template <class T>
class Queue{
public:
    Queue(unsigned int);
    void push(T);
    void resize(unsigned int);
    bool isEmpty();
    T pop();
    ~Queue();
private:
    unsigned int vsize;
    unsigned int qsize;
    T* vstart;
    T* qstart;
    T* qend;
};

template <class T>
Queue<T>::Queue(unsigned int size) : vsize(size), qsize(0) {
    vstart = new T[size];
    qstart = vstart;
    qend = vstart;
}

template <class T>
void Queue<T>::push(T value) {
    if(qend != qstart || (qsize == 0 && vsize != 0))
        *qend = value;
    else {
        resize(vsize*2);
        *qend = value;
    }
    if(qend < (vstart + vsize - 1))
        ++qend;
    else
        qend = vstart;
    ++qsize;
}

template <class T>
void Queue<T>::resize(unsigned int new_size) {
    T* new_vect = new T[new_size];
    int i;
    for(i=0; i<(vsize-(qstart-vstart)); ++i)
        new_vect[i] = qstart[i];
```

```
    for(int j=0; i < vsize; ++i, ++j)
        new_vect[i] = vstart[j];
    qstart = new_vect;
    qend = new_vect + qsize;
    delete vstart;
    vstart = new_vect;
    vsize = new_size;
}

template <class T>
bool Queue<T>::isEmpty() {
    return qsize == 0;
}

template <class T>
T Queue<T>::pop() {
    T ret = *qstart;
    if(qstart != (vstart + vsize - 1))
        ++qstart;
    else
        qstart = vstart;
    --qsize;
    return ret;
}

template <class T>
Queue<T>::~Queue() {
    delete vstart;
}
```