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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

ТЕМА: ОЧЕРЕДИ И СТЕКИ

Студент гр. 7382 _____ Гиззатов А.С.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2018

Задание.

Варинт №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).Программа была реализована на списке.

Функции и структуры данных

Были написаны 1 функция и 2 класса, в 1 из которых находится 7 методов и int main(). Функция void digit_srch(q_list *lst): принимает список, создает 2 строки в одну из которых записывает только цифры,а в другую записывает все остальные символы в обратном порядке. После записывает эти строки в результирующую строку в обратном порядке. Класс class q_El имеет 2 элемента char El и q_El *next, где char El является элементом списка, а q_El *next указатель на следующий элемент. Также в этом классе есть деструктор, который является дефолтным. 2 класс class q_list включает в себя 2 элемента которые относятся к классу q_El: q_El *head(голова списка) и q_El *tail(хвост списка). Также класс имеет коструктор q_list(std::string str) принимающий строку и создающий список, работая итеративно, дефолтный деструктор, метод char pop() который возвращает последний элемент списка, предварительно удалив его из списка, метод bool isEmpty(), который проверяет является ли список пустым или нет, метод int count(), который считает количество элементов в списке и метод q_El *prev_tail(q_El *lst), который возвращает указатель на предпоследний элемент в списке. В int main() происходит считывание строки или файла, и вызов функции void digit_srch(q_list *lst).

Описание алгоритма:

Была организована очередь для того, чтобы вывести все цифры строки после текста. То есть, записать их в конец строки, предварительно считав их или из потока данных или из файла. Программа записывала строку в очередь,

которая была организованна на списке, и записывала в 2 разные строки, в которых были записаны только цифры и остальные символы. Так как очередь работает по принципы FIFO(first in first out) то в алгоритме пришлось записывать результирующую строку в обратном порядке.

Тестирование.

Были написаны 4 теста для данной программы, а также скрипт для тестирования и компиляции программы.

Результаты тестирования приведены в табл.1

Возьмем 1 тест на рассмотрение:

Hello123 Wolrd321, I'm456 programmer789.

На 1 шаге был создан список, после этого проведен алгоритм:

1 итерация-текущая строка для символов - .

2 итерация-текущая строка для цифр - 9

3 итерация-текущая строка для цифр - 98

4 итерация-текущая строка для цифр - 987

5 итерация-текущая строка для символов - .r

6 итерация-текущая строка для символов - .re

7 итерация-текущая строка для символов - .rem

8 итерация-текущая строка для символов - .remm

9 итерация-текущая строка для символов - .remma

10 итерация-текущая строка для символов - .remmar

11 итерация-текущая строка для символов - .remmarg

12 итерация-текущая строка для символов - .remmargo

13 итерация-текущая строка для символов - .remmargor

14 итерация-текущая строка для символов - .remmargorp

15 итерация-текущая строка для символов - .remmargorp

16 итерация-текущая строка для цифр - 9876

17 итерация-текущая строка для цифр - 98765

18 итерация-текущая строка для цифр - 987654

19 итерация-текущая строка для символов - .remmargorp m

20 итерация-текущая строка для символов - .remmargorp m'

21 итерация-текущая строка для символов - .remmargorp m'I

22 итерация-текущая строка для символов - .remmargorp m'I

23 итерация-текущая строка для символов - .remmargorp m'I ,

24 итерация-текущая строка для цифр - 9876541

25 итерация-текущая строка для цифр - 98765412

26 итерация-текущая строка для цифр - 987654123

27 итерация-текущая строка для символов - .remmargorp m'I ,d

28 итерация-текущая строка для символов - .remmargorp m'I ,dr

29 итерация-текущая строка для символов - .remmargorp m'I ,drl

30 итерация-текущая строка для символов - .remmargorp m'I ,drlo

31 итерация-текущая строка для символов - .remmargorp m'I ,drloW

32 итерация-текущая строка для символов - .remmargorp m'I ,drloW

33 итерация-текущая строка для цифр - 9876541233

34 итерация-текущая строка для цифр - 98765412332

35 итерация-текущая строка для цифр - 987654123321

36 итерация-текущая строка для символов - .remmargorp m'I ,drloW o

37 итерация-текущая строка для символов - .remmargorp m'I ,drloW ol

38 итерация-текущая строка для символов - .remmargorp m'I ,drloW oll

39 итерация-текущая строка для символов - .remmargorp m'I ,drloW olle

40 итерация-текущая строка для символов - .remmargorp m'I ,drloW olleH

после всех итераций была создана результирующая строка:

Hello Wolrd, I'm programmer.123321456789

Таблица 1 - тесты

№	Входные данные	Выходные данные
1	Hello123 Wolrd321, I'm456 programmer789.	Hello Wolrd, I'm programmer.123321456789
2	abcdef123ghklmnop56789010	abcdefghklmnop12356789010
3	asdfghjkl;125678vfghdvjkdshgvbjkfglh21435674dvfkbl;'dalKJ	asdfghjkl;vfghdvjkdshgvbjkfglhdvfkbl;'dalKJ12567821435674
4	This mo123321del allows the ser456654ver to be wri789987tten as a collect111ion of sequential threads.	This model allows the server to be written as a collection of sequential threads.12332145665478998711

Выводы.

В результате работы были усвоены методы использования рекурсии, а также написана программа с использование метода рекурсии.

Исходный код

```
#include <iostream>
#include <string>
#include <cstring>
#include <ctype.h>
#include <fstream>
//#define READ
class q_El{
	public:
	char El;
	q_El *next;
	q_El(char el){
		El = el;
		next = nullptr;
	}
	~q_El() = default;

};

class q_list{
	public:
	q_El *head;
	q_El *tail;
	q_list(std::string str){
		q_El *cur=nullptr;
		q_El *ptr=nullptr;
		for(int i=0;i<str.length();i++){
			cur = new q_El(str[i]);
			if(ptr != nullptr)
				ptr->next = cur;
			if(i==0)
				head = cur;
			if(i == str.length()-1)
				tail = cur;
			ptr=cur;

		}
	}
```

```
        ~q_list() = default;
        void push(q_list *lst,char El){
                q_El *ptr = new q_El(El);                                       //pushing
an elemnt to the end of the queue
                ptr->next = head;
                head = ptr;
        }
        char pop(){
                    char cur_El = tail->El;
                  q_El *cur = prev_tail(head);
                        if(cur == nullptr){
                        delete head;
                        head = nullptr;
                        tail = nullptr;                                         //the
function is deleting the last element and returning it
                        return cur_El;
                        }
                  else{
                        delete cur->next;
                        cur->next = nullptr;
                                tail = cur;
                                return cur_El;
                  }
        }
        bool isEmpty(){
                if(head == nullptr){
                        return true;
                }
                else{
                        return false;
                }
        }
        int count(){
                int i=0;
                q_El *ptr = head;
            if(isEmpty() == true){
                  return i;
            }
                while(ptr->next != nullptr ){
                  i++;
                       // std::cout << ptr->El << " ";
                  ptr = ptr->next;
```

```
                }
            //std::cout<<ptr->El << std::endl;
                return i+1;

        }
        q_El *prev_tail(q_El *lst){
                q_El *ptr = lst;
            if(ptr->next == nullptr){
                  return ptr;                                  //searching element
that is previous to the last element
            }
                while(ptr->next->next != nullptr){
                        ptr = ptr->next;
                }
                return ptr;
        }
};

void digit_srch(q_list *lst){
      char *text = (char*)calloc(1000,sizeof(char));
      char *digits = (char*)calloc(1000,sizeof(char));
      char *str = (char*)calloc(2000,sizeof(char));
      int i=0,j=0,count;
      char el;
      std::ofstream F1;
      count = lst->count();
      while(count > 0){
            el = lst->pop();
            if(isdigit(el) > 0){
                  digits[j] = el;
      // poping all digits and write them into line named digits
                  j++;
                  std::cout <<"текущая строка для цифр - " << digits <<
std::endl;
            }
            else{
                  text[i] = el;
      //poping all other elements and write them into line named text
                  i++;
                      std::cout <<"текущая строка для символов - " << text <<
std::endl;
            }
```

```
            count--;
      }
      i=0;
//    std::cout<< text <<std::endl;
//    std::cout << digits<< std::endl;
      for(int k = strlen(text)-1;k>=0;k--){
      //making the final line by reversing text, digits lines and write them
into str
            str[i] = text[k];
              i++;
      }
      for(int l = strlen(digits)-1;l>=0;l--){
            str[i] = digits[l];
            i++;
      }
#ifdef READ
      F1.open("2.txt",std::ios_base::app);
      F1 << str << std::endl<<std::endl;
      F1.close();
#endif
      std::cout<< str << std::endl;
}


int main(){
        std::string text;
#ifndef READ
      std::getline(std::cin,text);
#endif
#ifdef READ
      std::fstream F;
      F.open("1.txt",std::ios::in);
      while(1) {
            if(!F.eof()){
                  std::getline(F, text);
                  q_list lst(text);
                  digit_srch(&lst);
      //doing the algorithm to each line that is gotten
            }
            else{
                  break;
      //breaking the cycle if it is the end of the file
```

```
            }
      }
      F.close();
#endif
        q_list lst(text);
      digit_srch(&lst);
}
```