**“ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ**

**УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ” (ТУСУР)**

Кафедра автоматизированных систем управления (АСУ)

Отчет по лабораторной работе №2

«Стеки и очереди»

По дисциплине СиАОДвЭВМ

Выполнил студент гр. 438-3:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Канаев О.А.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Проверил

Доктор технических наук,

доцент каф. АСУ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Горитов А.Н

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Томск 2019

**Цель работы:**

Получить первичные навыки методов обработки абстрактных данных: стеки и очереди.

**Задание:**

Вариант 2

Используя очередь, решить следующую задачу.

Содержимое текстового файла F, разделенное на строки, переписать в текстовый файл g, перенося при этом в конец каждой строки все входящие в неё цифры(с сохранением исходного порядка как цифр, так и литер).

Для реализации АТД "Очередь" использовать динамическое распределение памяти.

**Алгоритм:**

Для реализации нам потребуются счетчики: i,j,k,count, а также буфер для обрабатываемой строки: buf[100] и массив цифр: num[10], чтобы сравнивать с элементами буфера. После считывания строки идет её обработка посредством сравнения с числами от 0-9, перестановка их в конец файла и запись в выходной файл g. Итерации продолжаются до конца файла: while (!feof(file)).

**Решение:**#include <iostream>

#include <ostream>

using namespace std;

struct Node //элемент/узел очереди

{

char \*data;

Node \*next;

};

struct Queue //описание очереди

{

int size;

Node \*first;

Node \*last;

};

void Add(Queue \*Q, char \*value);

void Delete(Queue \*Q);

char \*Top(Queue \*Q);

int Size(Queue \*Q);

void Init(Queue \*Q);

bool Full(Queue \*Q);

void func(Queue \*Q);

int main()

{

system("chcp 1251");

Queue Q;

Init(&Q);

int number;

do

{

cout << "1) Добавить эл-т" << endl;

cout << "2) Удалить эл-т" << endl;

cout << "3) Вывести верхний эл-т" << endl;

cout << "4) Узнать размер очереди" << endl;

cout << "5) Функция с карточки" << endl;

cout << "0) Stop\n\n";

cout << "Команда №: ";

cin >> number;

switch (number)

{

case 1: {

char \*txt= new char;

cin >> txt;

Add(&Q, txt);

break;

}

case 2: {

if (Full(&Q)) cout << endl << "Очередь пуста\n\n";

else Delete(&Q);

break;

}

case 3: {

if (Full(&Q)) cout << endl << "Очередь пуста\n\n";

else {

cout << "\nНачальный элемент: " << Top(&Q) << "\n\n";

}

break;

}

case 4: {

if (Full(&Q)) cout << endl << "Очередь пуста\n\n";

else cout << "\nРазмер очереди: " << Size(&Q) << "\n\n";

break;

}

case 5: {

func(&Q);

break;

}

case 0: break;

}

} while (number != 0);

while (Q.size != 0) //Отчистка памяти от очереди

{

if ((Q.first != Q.last))

Delete(&Q);

else

{

delete Q.first->next;

delete Q.first;

Q.size--;

}

}

return 0;

system("pause");

}

void Init(Queue \*Q) //создание очереди

{

Q->last = Q->first = new Node;

Q->last->next = Q->first->next = NULL;

Q->size = 0;

}

bool Full(Queue \*Q) //проверка очереди на пустоту

{

if ((Q->first == Q->last) && (Q->size == 0)) return true;

else return false;

}

char \*Top(Queue \*Q) //вывод начального элемента

{

return Q->first->data;

}

void Add(Queue \*Q, char \*value) //добавление элемента

{

Node \*temp = new Node;

Q->last->next = temp;

temp->data = value;

temp->next = NULL;

Q->last = temp;

//temp->next = Q->first;

//temp->data = value;

//Q->first = temp;

if (Q->size == 0) Q->last = Q->first;

Q->size++;

cout << "\nЭлемент добавлен\n\n";

}

void Delete(Queue \*Q) //удаление элемента

{

Node \*temp;

temp = Q->first;

//while (temp->next != Q->last) temp = temp->next;

//delete Q->last;

////Q->last = new Node;

//temp->next->next = NULL;

//Q->last = temp;

Q->first = Q->first->next;

delete temp;

Q->size--;

cout << "\nЭлемент удален\n\n";

}

int Size(Queue \*Q) //размер очереди

{

return Q->size;

}

void func(Queue \*Q) {

FILE \*file = fopen("F.txt", "r+"), \*fileG= fopen("g.txt", "w+");

char buf[100], temp, num[10] = {'0','1','2','3','4','5','6','7','8','9'};

int j = 0, k, count;

while (!feof(file))

{

fgets(buf, 100, file);

count = strlen(buf);

while (count>0)

{

for (int i = 0; i < 10; i++)

{

if (buf[j] == num[i])

{

temp = buf[j];

for (k = j; k < strlen(buf)-1; k++)

buf[k] = buf[k + 1];

buf[strlen(buf) - 2] = temp;

count = count - 2;

//buf[k] = '\0';

}

}

j++;

count--;

}

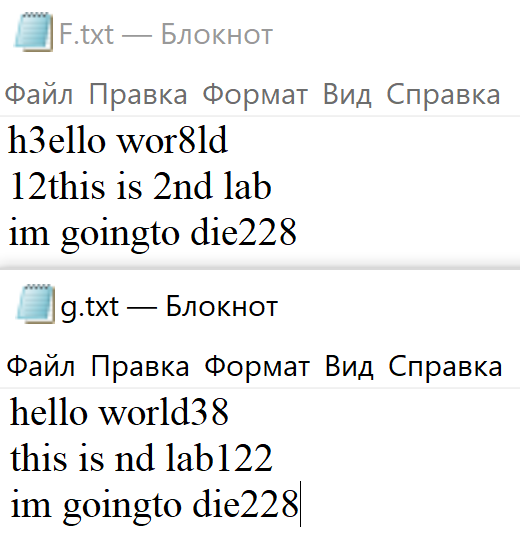
fprintf(fileG, "%s", buf);

j = 0;

}

fclose(file);

}}**Результат работы:**



Исходный файл

Обработанный файл

**Вывод:**Научился использовать абстрактный тип данных, очереди, с учетом плюсов и минусов. Плюсы очередей — это возможность использовать два конца: в один помещать, из другого доставать; и скорость выполнения операций. Минусы — оба конца имеют только одну собственную функцию и нельзя вынимать элемент сразу из центра.