Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №6**

**«РЕАЛИЗАЦИЯ ЭЛЕМЕНТАРНЫХ СТРУКТУР ДАННЫХ НА ОСНОВЕ СТАТИЧЕСКОЙ ПАМЯТИ»**

**ПО МДК 05.02 РАЗРАБОТКА КОДА ИНФОРМАЦИОННЫХ СИСТЕМ**

Выполнил: студент учебной группы

ИСПк-101-51-00

Сунцов Александр Андреевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

**Цель работы:**

Изучение принципов работы с базовыми структурами данных, получение навыков организации case-меню.

**Задания:**

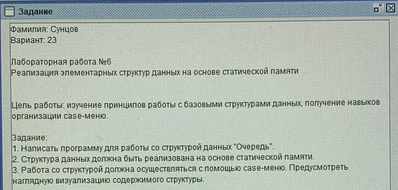
****

Рисунок 1 – Скриншот задания

**Описание алгоритма:**

Очередь – это структура данных, добавление и удаление элементов в которой происходит путём операций и соответственно. Притом первым из очереди удаляется элемент, который был помещен туда первым, то есть в очереди реализуется принцип “первым вошел — первым вышел” (англ. first-in, first-out — FIFO).

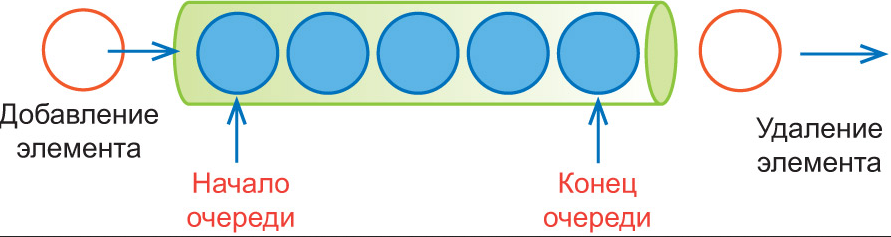
****

Рисунок 2 – Описание алгоритма

**Код программы:**

**uses** crt;

**const**

queue\_size = 11;

**type**

QueueElement = Integer;

QueueArray = **array**[1..queue\_size] **of** QueueElement;

QueuePointer = 1..queue\_size;

**var**

Que: QueueArray;

Front, Rear: QueuePointer;

Element: QueueElement;

a, begin\_queue, end\_queue: Integer;

**procedure** Queue;

**begin**

begin\_queue := 1;

end\_queue := 1;

**end**;

**function** IsEmpty: Boolean;

**begin**

IsEmpty := begin\_queue = end\_queue;

**end**;

**function** IsFull: Boolean;

**begin**

IsFull := end\_queue = queue\_size;

**end**;

**procedure** Inqueue(Element: QueueElement);

**begin**

ClrScr;

**if** IsFull **then**

WriteLn('Очердь заполнена!')

**else**

**begin**

Que[end\_queue] := Element;

end\_queue := end\_queue + 1;

**end**;

**end**;

**procedure** Delqueue;

**var**

i: QueuePointer;

**begin**

ClrScr;

**if** IsEmpty **then**

WriteLn('Очередь пустая!')

**else**

**begin**

Element := Que[begin\_queue];

**for** i := begin\_queue **to** end\_queue - 1 **do**

Que[i] := Que[i + 1];

end\_queue := end\_queue - 1;

**end**;

**end**;

**procedure** PrintQueue;

**var**

i: QueuePointer;

**begin**

ClrScr;

Print('Очередь: ');

**for** i := begin\_queue **to** end\_queue - 1 **do**

Write(Que[i], ' ');

WriteLn;

**end**;

**begin**

Queue;

**repeat**

WriteLn('1 - Добавить элемент');

WriteLn('2 - Удалить элемент');

WriteLn('3 - Показать элементы очереди');

WriteLn('0 - Выйти');

Write('Выберите действие: ');

ReadLn(a);

**case** a **of**

1: **begin**

ClrScr;

Print('Введите элемент для добавления в очередь: ');

ReadLn(Element);

Inqueue(Element);

**end**;

2: Delqueue;

3: PrintQueue;

0: **exit**;

**end**;

**until** a = 0;

**end**.

**Результат выполнения программы:**

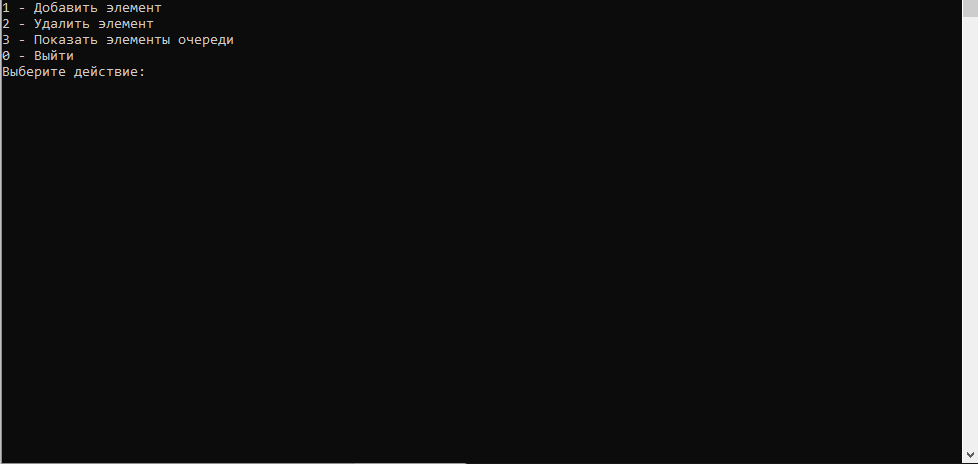
****

Рисунок 2 – Case-меню программы

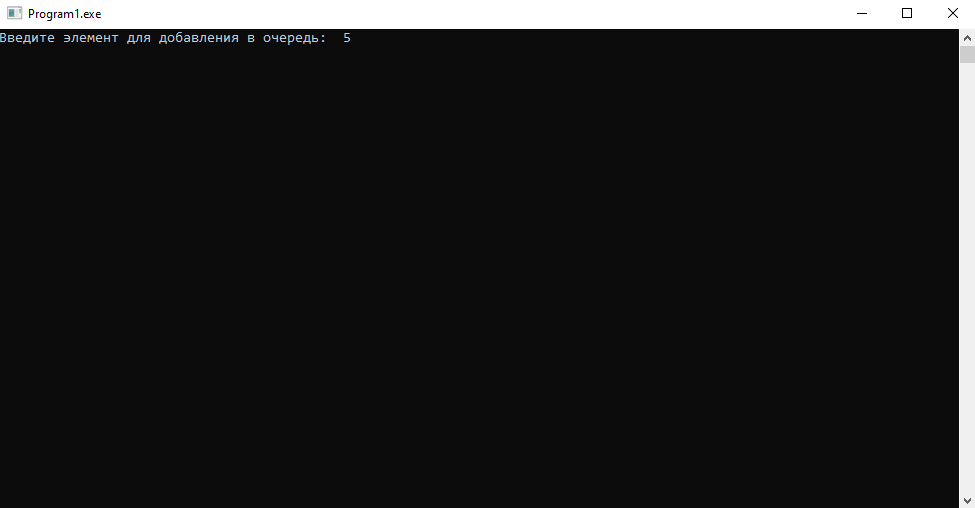
****

Рисунок 3 – Добавление элемента в очередь

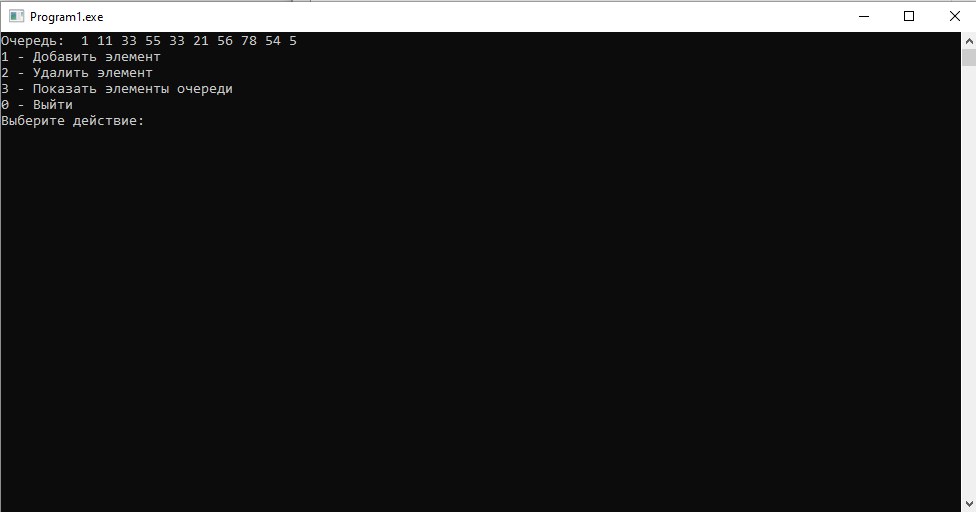


Рисунок 4 – Визуализация содержимого структуры



Рисунок 5 – Удаление элемента

**Вывод:**

В ходе выполнения данной домашней контрольной работы, были получены знания об базовой структуре данных. Был получен опыт в реализации динамической структуры данных “Очередь”. Также в ходе домашней контрольной работы пригодились полученные ранее знания об организации case-меню.

В ходе выполнения домашней контрольной работы №6 возникли некоторые трудности, но все они решились, просмотрев заново весь лекционный материал.

В итоге были изучены принципы реализации элементарных структур данных на основе статической памяти, и пригодились знания о организации case-меню