**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**

Выполнила: студент учебной группы

ИСПк-206-52-00

Ленский Арсений Андреевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

**Цель работы**

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

**Формулировка задания**

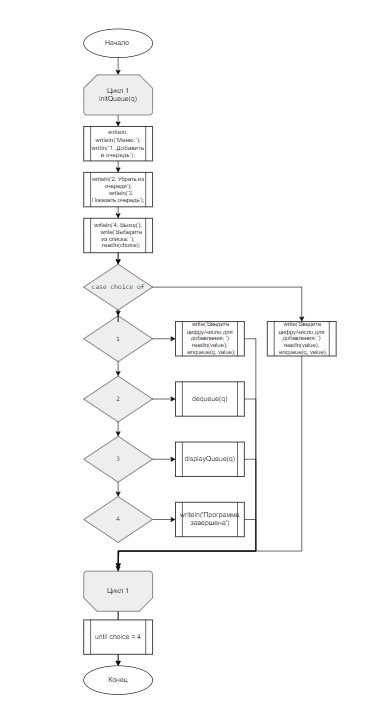
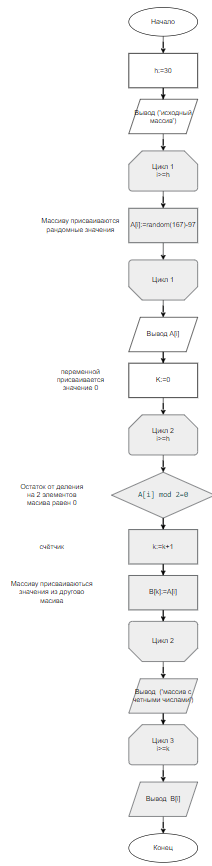
Вариант 4

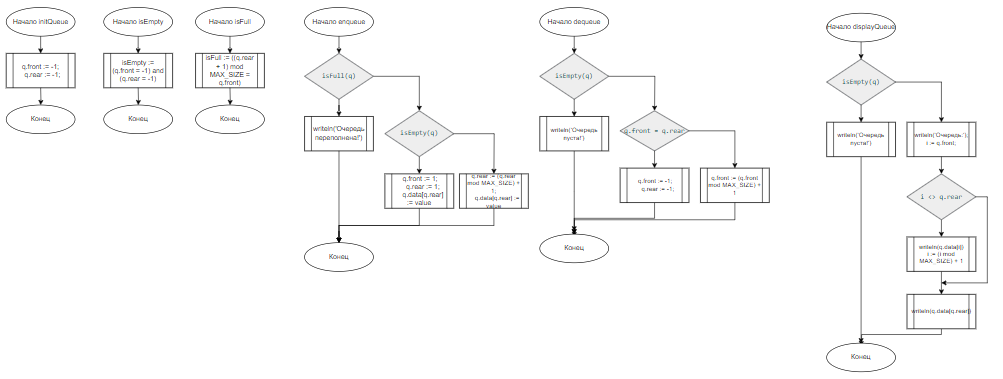
1. Написать программу для работы со структурой данных «Очередь».
2. Структура данных должна быть реализованная на основе статической памяти.

**Описание алгоритма**

1. Инициализация очереди, проверка на пустоту и полноту, добавление и удаление элементов, а также отображение содержимого очереди.
2. Главная часть программы предлагает пользователю выбрать действие из меню: добавить элемент, удалить элемент, отобразить очередь или завершить программу.

**Схемы алгоритма**

****

****

**Код программы**

**program** QueueImplementation;

**const**

MAX\_SIZE = 10;

**type**

Queue = **record**

data: **array**[1..MAX\_SIZE] **of** Integer;

front: Integer;

rear: Integer;

**end**;

**procedure** initQueue(**var** q: Queue);

**begin**

q.front := -1;

q.rear := -1;

**end**;

**function** isEmpty(**var** q: Queue): Boolean;

**begin**

isEmpty := (q.front = -1) **and** (q.rear = -1);

**end**;

**function** isFull(**var** q: Queue): Boolean;

**begin**

isFull := ((q.rear + 1) **mod** MAX\_SIZE = q.front);

**end**;

**procedure** enqueue(**var** q: Queue; value: Integer);

**begin**

**if** isFull(q) **then**

writeln('Очередь переполнена!')

**else if** isEmpty(q) **then**

**begin**

q.front := 1;

q.rear := 1;

q.data[q.rear] := value;

**end**

**else**

**begin**

q.rear := (q.rear **mod** MAX\_SIZE) + 1;

q.data[q.rear] := value;

**end**;

**end**;

**procedure** dequeue(**var** q: Queue);

**begin**

**if** isEmpty(q) **then**

writeln('Очередь пуста!')

**else if** q.front = q.rear **then**

**begin**

q.front := -1;

q.rear := -1;

**end**

**else**

q.front := (q.front **mod** MAX\_SIZE) + 1;

**end**;

**procedure** displayQueue(**var** q: Queue);

**var**

i: Integer;

**begin**

**if** isEmpty(q) **then**

writeln('Очередь пуста!')

**else**

**begin**

writeln('Очередь: ');

i := q.front;

**while** i <> q.rear **do**

**begin**

writeln(q.data[i]);

i := (i **mod** MAX\_SIZE) + 1;

**end**;

writeln(q.data[q.rear]);

**end**;

**end**;

**var**

q: Queue;

choice, value: Integer;

**begin**

initQueue(q);

**repeat**

writeln;

writeln('Меню:');

writeln('1. Добавить в очередь');

writeln('2. Убрать из очереди');

writeln('3. Показать очередь');

writeln('4. Выход');

write('Выберите из списка: ');

readln(choice);

**case** choice **of**

1:

**begin**

write('Введите цифру/число для добавления: ');

readln(value);

enqueue(q, value);

**end**;

2: dequeue(q);

3: displayQueue(q);

4: writeln('Программа завершена');

**else**

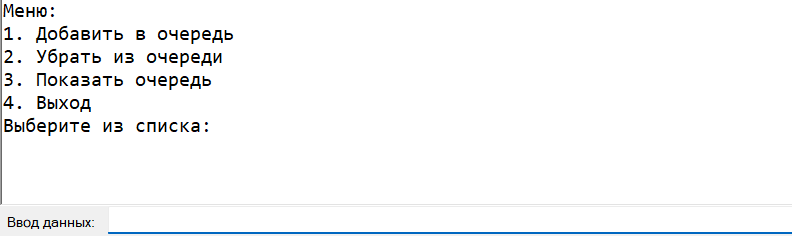
writeln('Неверное значение!');

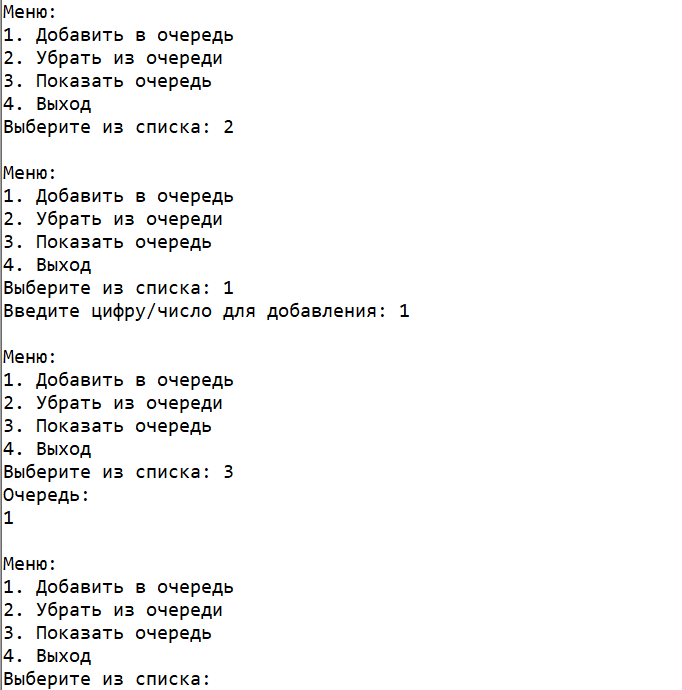
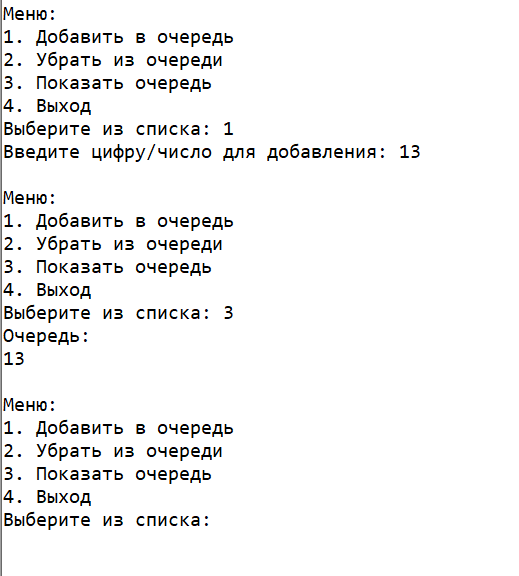
**end**;

**until** choice = 4;

**end**.

**Результат выполнения программы**

****

****

**Ответы на вопросы**

1. Дана структура данных "очередь". Очередь представляет собой коллекцию элементов, где элементы добавляются в конец очереди и удаляются из начала очереди. Основные свойства очереди:

- FIFO (First-In-First-Out) - элементы удаляются в том же порядке, в котором они были добавлены.

- Ограниченная ёмкость - размер очереди ограничен, и добавление элементов в полную очередь вызывает переполнение.

2. Очередь хранится с помощью массива. В данном случае массив data используется для хранения элементов очереди. Указатели front и rear используются для указания на начало и конец очереди соответственно. Узлы в этой структуре – это элементы массива, где каждый элемент представляет отдельный элемент очереди. Указатель front указывает на первый элемент очереди, а rear указывает на последний добавленный элемент.

3. Основные операции со структурой:

- Добавление элемента: новый элемент добавляется в конец очереди. Если очередь полна, выводится сообщение об ошибке.

- Взятие элемента: элемент удаляется из начала очереди. Если очередь пуста, выводится сообщение об ошибке.

- Отображение элементов: все элементы очереди выводятся на экран в порядке их добавления.

4. Кейс-меню организовано с помощью главной части программы, которая предлагает пользователю выбрать действие из меню. Каждая опция меню соответствует одной из операций со структурой: добавить элемент, удалить элемент, отобразить очередь или завершить программу. После выполнения выбранной операции пользователь снова попадает в меню, где может выбрать следующее действие или завершить программу.

**Вывод**

В результате выполнения данного задания, я получил навыки работы с базовыми структурами данных, получил навыки организации case-меню.

Разработав программу для работы со структурой данных «Очередь», которая позволяет добавить в очередь, удалить из очереди и показать очередь. Это позволило мне попрактиковаться в работе со структурами данных изучить их основы. Одновременно я улучшил свои навыки программирования.