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»**

**ПО «МДК 05.02 Разработка кода информационных систем»**

Выполнил: студент учебной группы

ИСПк-206-52-00

Шерстобитов Михаил Ромнаович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2025

1. Цель работы
2. Формулировка задания (с вариантом)
3. Описание алгоритма и ответы на вопросы
4. Схема алгоритма с комментариями
5. Код программы
6. Результат выполнения программы
7. Вывод
8. Цель работы: изучения принципов работы с баззовыми структурами данных получения навыков организации case-меню

2.Формулировка заданий:

Вариант №13

Задание.

1. Написать программу для работы со сруктурой данных очередь.
2. Сруктура данных должан быть реализовна на основе статической памяти
3. Работа со структорой должна осуществляться на основе case- меню ПРЕДУСМОТРЕТЬ ВИЗУАЛИЗАЦИЮ

* указатель – это переменная, в которой можно хранить адрес другой переменной;
* при объявлении указателя надо указать тип переменных, на которых он будет указывать, а перед типом поставить знак **^** ;
* знак **@** перед именем переменной обозначает ее адрес;
* запись **p^** обозначает *значение* ячейки, на которую указывает указатель **p**;
* **nil** – это *нулевой указатель*, он никуда не указывает
* при изменении значения указателя на **n** он в самом деле сдвигается к **n***-*ому следующему числу данного типа (для указателей на целые числа – на **n\*sizeof(integer)** байт)*.*

**Очередь** – это линейная структура данных, в которой   
добавление элементов возможно только с одного конца   
(**конца очереди**), а удаление элементов – только с другого конца (**начала очереди**)**.**

**FIFO = *First In – First Out***

«Кто первым вошел, тот первым вышел».

**Операции с очередью:**

* 1. добавить элемент в конец очереди (*PushTail =* втолкнуть   
     в конец)*;*
  2. удалить элемент с начала очереди *(Pop).*

Код программы

**program** Project1;

**const**

MAX\_SIZE = 100; // Максимальный размер очереди

**type**

TData = Integer; // Тип данных в очереди

TQueue = **record**

Data: **array**[1..MAX\_SIZE] **of** TData; // Статический массив для хранения данных

Front, Rear: Integer; // Индексы начала и конца очереди

Size: Integer; // Текущий размер очереди

**end**;

{ Инициализация очереди }

**procedure** InitQueue(**var** Q: TQueue);

**begin**

Q.Front := 1;

Q.Rear := 0;

Q.Size := 0;

**end**;

{ Проверка, пуста ли очередь }

**function** IsEmpty(**const** Q: TQueue): Boolean;

**begin**

IsEmpty := (Q.Size = 0);

**end**;

{ Проверка, заполнена ли очередь }

**function** IsFull(**const** Q: TQueue): Boolean;

**begin**

IsFull := (Q.Size = MAX\_SIZE);

**end**;

{ Добавление элемента в очередь }

**procedure** Enqueue(**var** Q: TQueue; Value: TData);

**begin**

**if** IsFull(Q) **then**

**begin**

Writeln('Очередь переполнена!');

**Exit**;

**end**;

Q.Rear := Q.Rear **mod** MAX\_SIZE + 1; // Кольцевой буфер

Q.Data[Q.Rear] := Value;

Q.Size := Q.Size + 1;

**end**;

{ Извлечение элемента из очереди }

**function** Dequeue(**var** Q: TQueue; **var** Value: TData): Boolean;

**begin**

Dequeue := False;

**if** IsEmpty(Q) **then Exit**;

Value := Q.Data[Q.Front];

Q.Front := Q.Front **mod** MAX\_SIZE + 1; // Кольцевой буфер

Q.Size := Q.Size - 1;

Dequeue := True;

**end**;

{ Вывод содержимого очереди }

**procedure** PrintQueue(**const** Q: TQueue);

**var**

i, Pos: Integer;

**begin**

**if** IsEmpty(Q) **then**

**begin**

Writeln('Очередь пуста');

**Exit**;

**end**;

Write('Очередь: ');

Pos := Q.Front;

**for** i := 1 **to** Q.Size **do**

**begin**

Write(Q.Data[Pos], ' ');

Pos := Pos **mod** MAX\_SIZE + 1;

**end**;

Writeln;

**end**;

**procedure** ShowMenu;

**begin**

Writeln('Меню управления очередью:');

Writeln('1. Добавить элемент');

Writeln('2. Удалить элемент');

Writeln('3. Показать очередь');

Writeln('4. Найти максимальный элемент');

Writeln('5. Очистить очередь');

Writeln('0. Выход');

Write('Выберите действие: ');

**end**;

**var**

Q: TQueue;

Data, MaxVal: TData;

Count: Integer;

Choice: Char;

InputValue: Integer;

**begin**

InitQueue(Q);

Randomize;

**repeat**

ShowMenu;

Readln(Choice);

Writeln;

**case** Choice **of**

'1': **begin**

**if** IsFull(Q) **then**

Writeln('Очередь переполнена!')

**else**

**begin**

Write('Введите целое число для добавления: ');

Readln(InputValue);

Enqueue(Q, InputValue);

Writeln('Элемент ', InputValue, ' добавлен в очередь');

**end**;

**end**;

'2': **begin**

**if** Dequeue(Q, Data) **then**

Writeln('Извлечен элемент: ', Data)

**else**

Writeln('Очередь пуста!');

**end**;

'3': PrintQueue(Q);

'4': **begin**

**if** IsEmpty(Q) **then**

**begin**

Writeln('Очередь пуста!');

**Continue**;

**end**;

// Создаем временную очередь для поиска максимума

Count := 0;

MaxVal := Q.Data[Q.Front];

// Проходим по всем элементам очереди

**for** Data := Q.Front **to** Q.Front + Q.Size - 1 **do**

**begin**

**if** Q.Data[Data **mod** MAX\_SIZE + 1] > MaxVal **then**

**begin**

MaxVal := Q.Data[Data **mod** MAX\_SIZE + 1];

Count := 1;

**end**

**else if** Q.Data[Data **mod** MAX\_SIZE + 1] = MaxVal **then**

Inc(Count);

**end**;

Writeln('Максимальное значение: ', MaxVal);

Writeln('Количество максимальных элементов: ', Count);

**end**;

'5': **begin**

InitQueue(Q);

Writeln('Очередь очищена');

**end**;

'0': Writeln('Выход из программы');

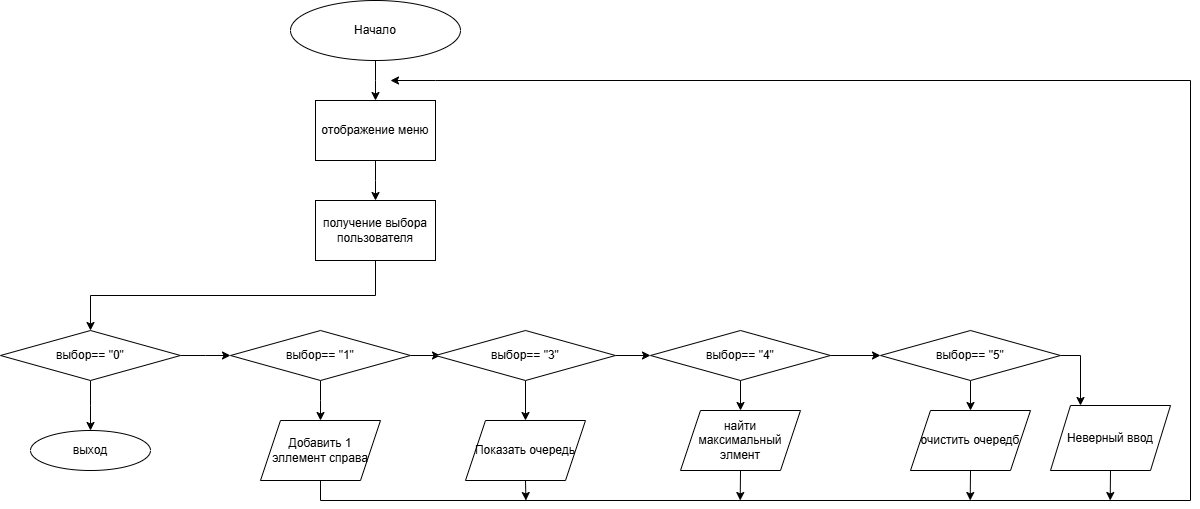
**else** Writeln('Неверный выбор!');

**end**;

Writeln;

**until** Choice = '0';

**end**.



Вывод

В ходе выполнения лабораторной работы были получены практические навыки работы с динамическими структурами данных, а также понимание принципов работы с очередями.

Программа продемонстрировала, как можно эффективно управлять памятью и использовать динамические структуры для решения задач, связанных с обработкой данных.