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-203-52-00

Макаров Никита Андреевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2025

**Цель работы:**

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

**Формулировка задания:**

1.Написать программу для работы со структурой данных “Дек”.

2. Структура данных должна быть реализована на основе статической памяти.

3. Работа со структурой данных должна осуществляться с помощью case-меню. Предусмотреть наглядную визуализацию содержимого структуры.

**Описание алгоритма:**

Алгоритм:

Сначала происходит инициализациядека procedure InitializeDeque(var Deque: TDeque);

begin

Deque.Front := 0;

Deque.Rear := 0;

end;

После инициализации ,происходит проверка на пустоту function IsEmpty(Deque: TDeque): Boolean;

begin

IsEmpty := (Deque.Front = 0);

end;

Если Front равен 0, дек считается пустым.

Дек проверяется на заполненность function IsFull(Deque: TDeque): Boolean;

begin

IsFull := (Deque.Rear = MAX\_SIZE);

end;

Если Rear равен MAX\_SIZE, дек считается заполненным.

Добавление элемента в начало ,если дек заполнен то перед первым числом ставится вводимое пользователем, если же он был пуст то параметр front устанавливает на 1 и число добавляется .

Добавление элемента в конец происходит аналогично как и вперед , если дек пуст то число просто добавляется, если он полон то число вводимое пользователем добавляется после последнего в массиве.

Удаление элемента спереди происходит через задачи переменной Deque.Front := 0 после этого следующей переменной в деке передается значение первой.

Удаление последнего элемента делается аналогично переменной Deque.Rear :=0 , и предпоследний элемент становится последним.

**Схемы алгоритмов**:

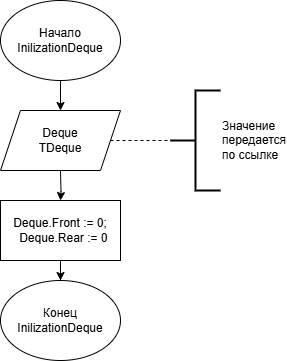


Рисунок 1- схема алгоритма процедуры InitializeDeque

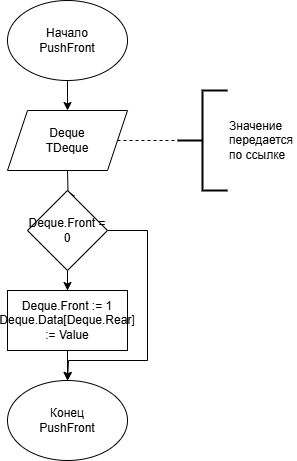


Рисунок 2 – схема алгоритма процедуры PushFront

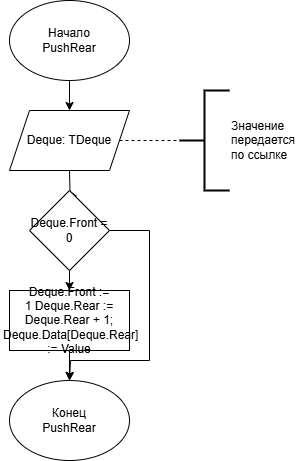


Рисунок 3 – схема алгоритма процедуры PushRear

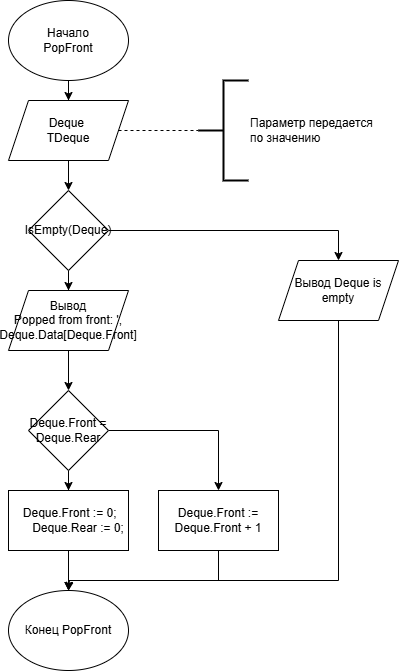


Рисунок 4 – схема алгоритма процедуры PopFront

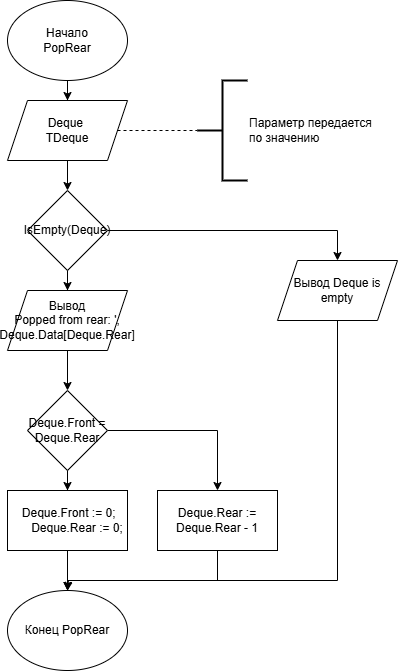


Рисунок 5 – схема алгоритма процедуры PopRear

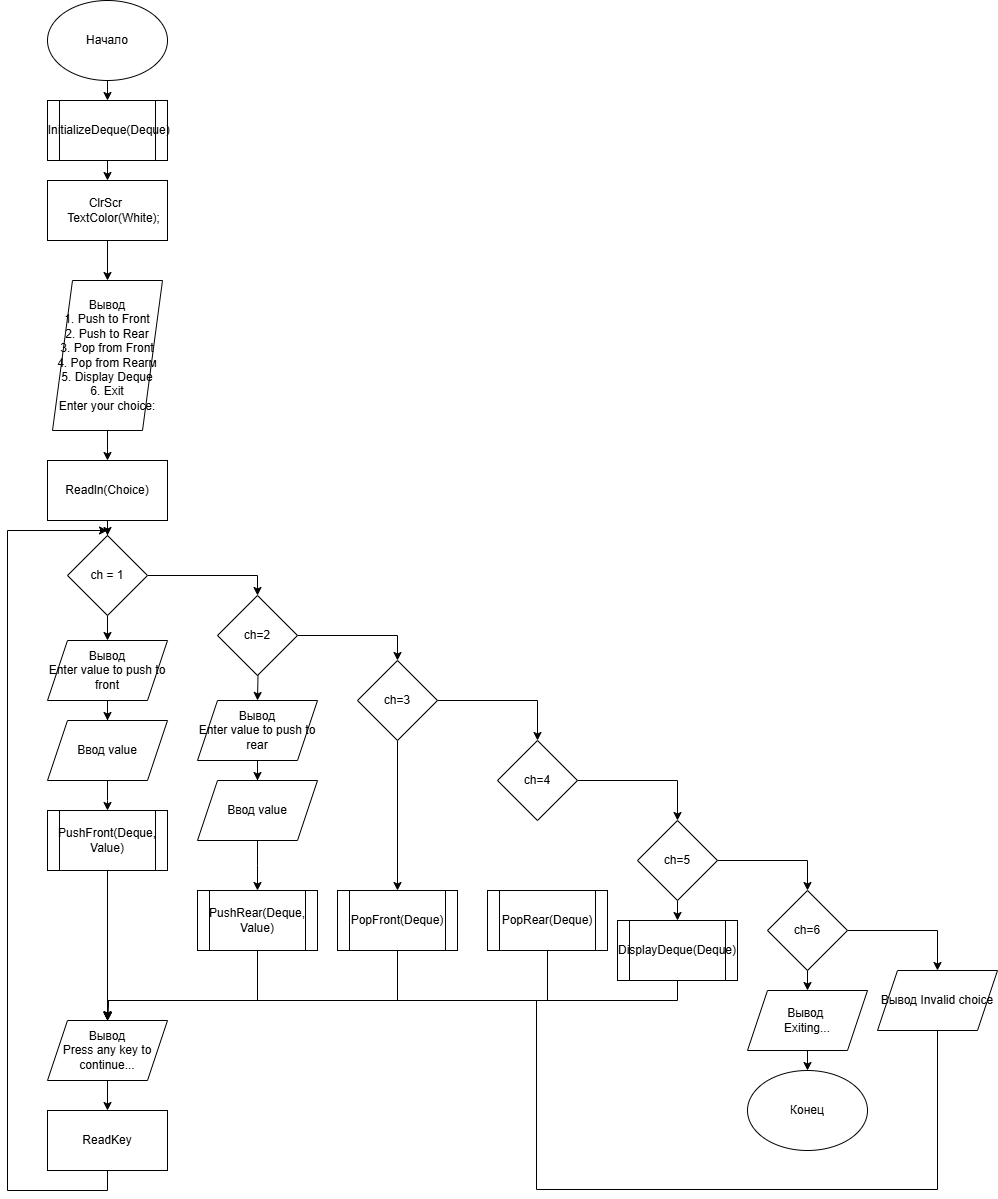
****

Рисунок 6 – схема алгоритма case-меню

**Код программы:**

**program DequeProgram;**

**uses**

**crt;**

**const**

**MAX\_SIZE = 10; // Максимальный размер дека**

**type**

**TDeque = record**

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

**Front, Rear: Integer;**

**end;**

**procedure InitializeDeque(var Deque: TDeque);**

**begin**

**Deque.Front := 0;**

**Deque.Rear := 0;**

**end;**

**function IsEmpty(Deque: TDeque): Boolean;**

**begin**

**IsEmpty := (Deque.Front = 0);**

**end;**

**function IsFull(Deque: TDeque): Boolean;**

**begin**

**IsFull := (Deque.Rear = MAX\_SIZE);**

**end;**

**procedure PushFront(var Deque: TDeque; Value: Integer);**

**begin**

**if IsFull(Deque) then**

**Writeln('Deque is full')**

**else**

**begin**

**if Deque.Front = 0 then**

**Deque.Front := 1;**

**Deque.Rear := Deque.Rear + 1;**

**Deque.Data[Deque.Rear] := Value;**

**end;**

**end;**

**procedure PushRear(var Deque: TDeque; Value: Integer);**

**begin**

**if IsFull(Deque) then**

**Writeln('Deque is full')**

**else**

**begin**

**if Deque.Front = 0 then**

**Deque.Front := 1;**

**Deque.Rear := Deque.Rear + 1;**

**Deque.Data[Deque.Rear] := Value;**

**end;**

**end;**

**procedure PopFront(var Deque: TDeque);**

**begin**

**if IsEmpty(Deque) then**

**Writeln('Deque is empty')**

**else**

**begin**

**Writeln('Popped from front: ', Deque.Data[Deque.Front]);**

**if Deque.Front = Deque.Rear then**

**begin**

**Deque.Front := 0;**

**Deque.Rear := 0;**

**end**

**else**

**Deque.Front := Deque.Front + 1;**

**end;**

**end;**

**procedure PopRear(var Deque: TDeque);**

**begin**

**if IsEmpty(Deque) then**

**Writeln('Deque is empty')**

**else**

**begin**

**Writeln('Popped from rear: ', Deque.Data[Deque.Rear]);**

**if Deque.Front = Deque.Rear then**

**begin**

**Deque.Front := 0;**

**Deque.Rear := 0;**

**end**

**else**

**Deque.Rear := Deque.Rear - 1;**

**end;**

**end;**

**procedure DisplayDeque(Deque: TDeque);**

**var**

**i: Integer;**

**begin**

**if IsEmpty(Deque) then**

**Writeln('Deque is empty')**

**else**

**begin**

**Writeln('Deque contents:');**

**for i := Deque.Front to Deque.Rear do**

**Write(Deque.Data[i], ' ');**

**Writeln;**

**end;**

**end;**

**var**

**Deque: TDeque;**

**Choice, Value: Integer;**

**begin**

**InitializeDeque(Deque);**

**repeat**

**ClrScr; // Очистка экрана**

**TextColor(White); // Установка цвета текста**

**Writeln('1. Push to Front');**

**Writeln('2. Push to Rear');**

**Writeln('3. Pop from Front');**

**Writeln('4. Pop from Rear');**

**Writeln('5. Display Deque');**

**Writeln('6. Exit');**

**Write('Enter your choice: ');**

**Readln(Choice);**

**case Choice of**

**1: begin**

**Write('Enter value to push to front: ');**

**Readln(Value);**

**PushFront(Deque, Value);**

**end;**

**2: begin**

**Write('Enter value to push to rear: ');**

**Readln(Value);**

**PushRear(Deque, Value);**

**end;**

**3: PopFront(Deque);**

**4: PopRear(Deque);**

**5: DisplayDeque(Deque);**

**6: Writeln('Exiting...');**

**else**

**Writeln('Invalid choice');**

**end;**

**Write('Press any key to continue...');**

**ReadKey; // Ожидание нажатия клавиши**

**until Choice = 6;**

**end.**

**Результат выполнения программы:**

**Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.**

**Рисунок – результат выполнения программы**

**Вывод:**

В ходе работы была изучена структура данных дек – как он работает, строиться и в каких случаях используется ,также была написана программа реализующая простой дек, также был получен опыт работы с case-меню , которое также реализовано в программе.