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»**

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

Выполнил: студент учебной группы

ИСПк-206-52-00

Онучин Кирилл Сергеевич

Преподаватель:

Сергеева Е. Г.

Киров

2024

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

Описание алгоритма:

Дек (от англ. deque — double ended queue) — структура данных, представляющая из себя список элементов, в которой добавление новых элементов и удаление существующих производится с обоих концов. Эта структура поддерживает как FIFO, так и LIFO, поэтому на ней можно реализовать как стек, так и очередь. В первом случае нужно использовать только методы головы или хвоста, во втором — методы push и pop двух разных концов. Дек можно воспринимать как двустороннюю очередь

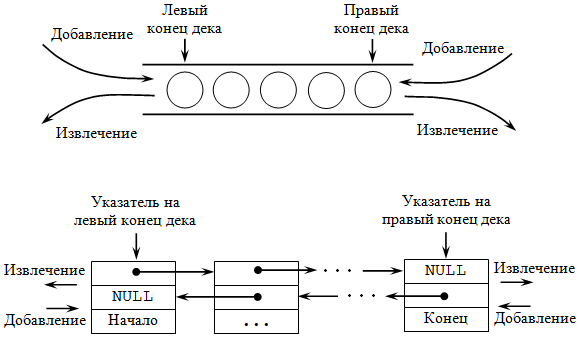


Рисунок 1 – Описание алгоритма “Дек”

**Схема алгоритма:**

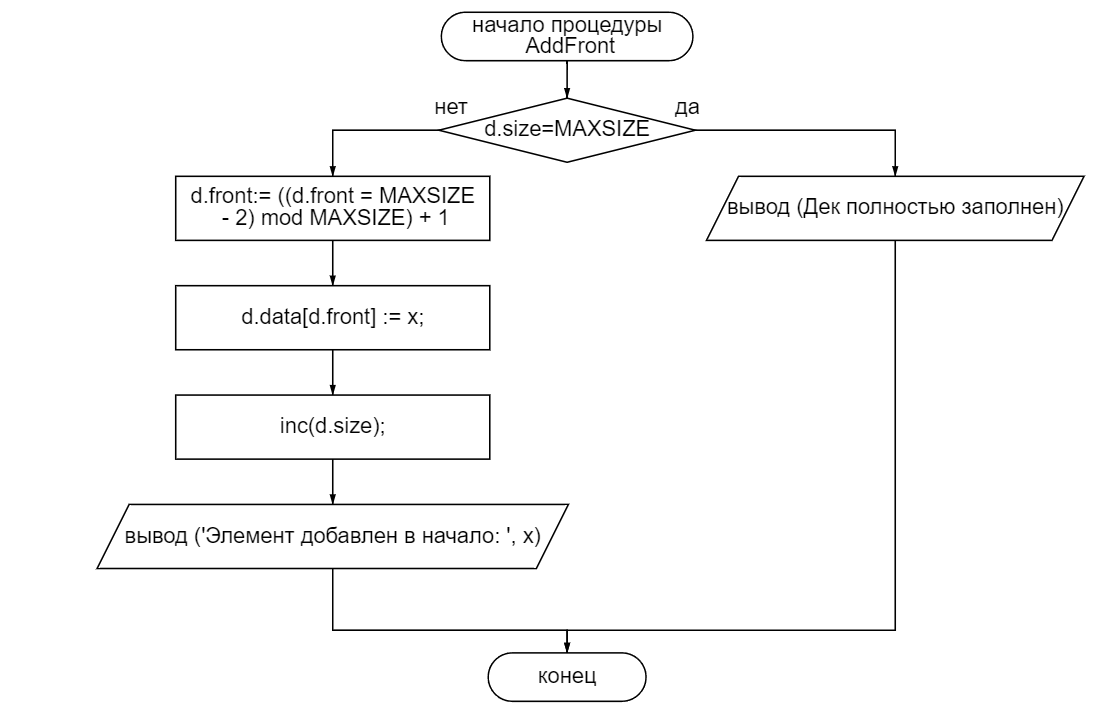


Рисунок 2 – Процедура AddFront

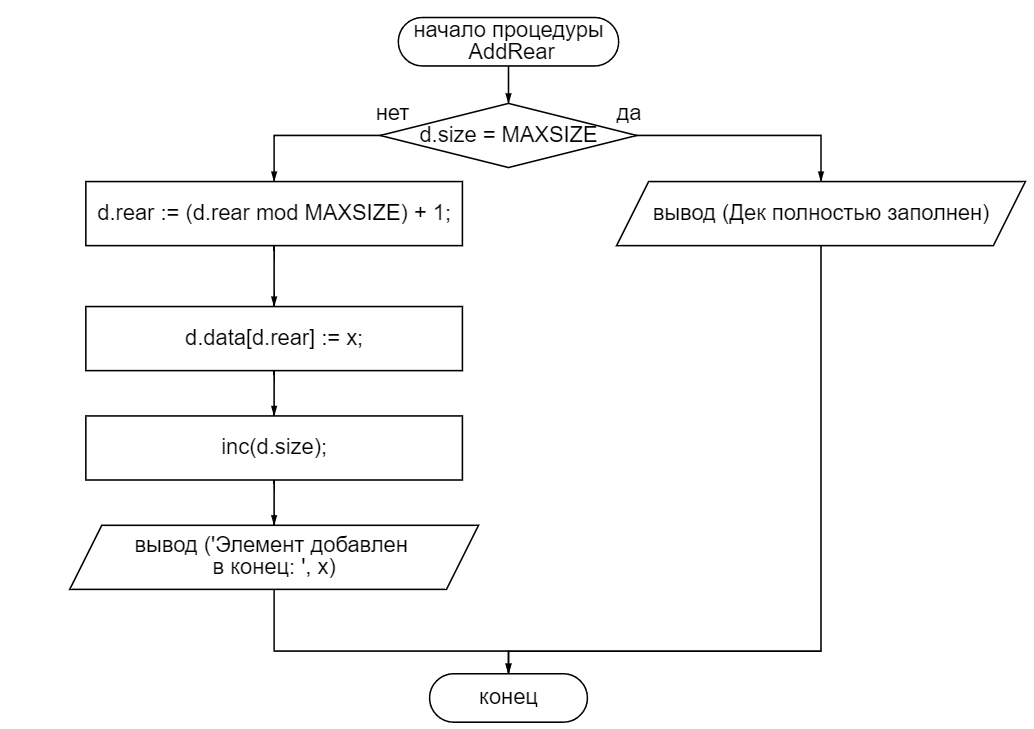


Рисунок 3 – Процедура AddRear

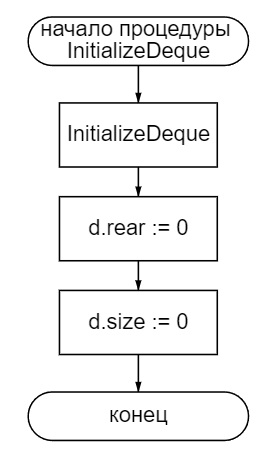


Рисунок 4 – Процедура InitializeDeque

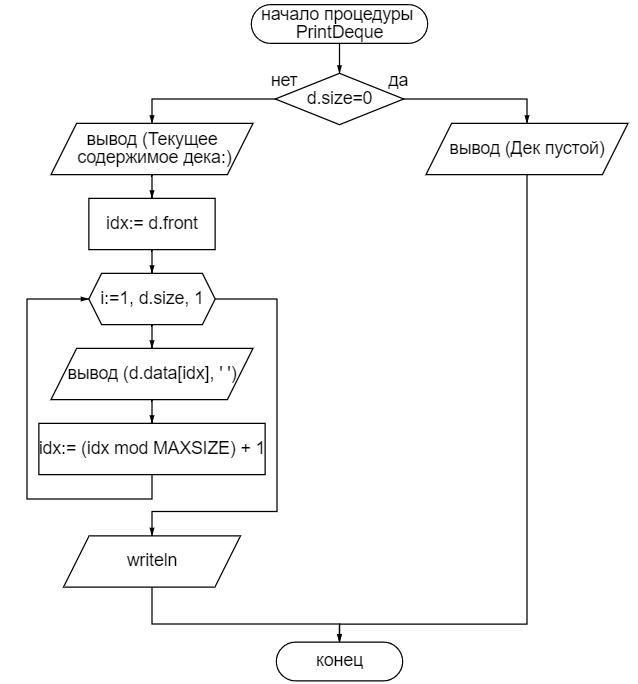


Рисунок 5 – Процедура PrintDeque

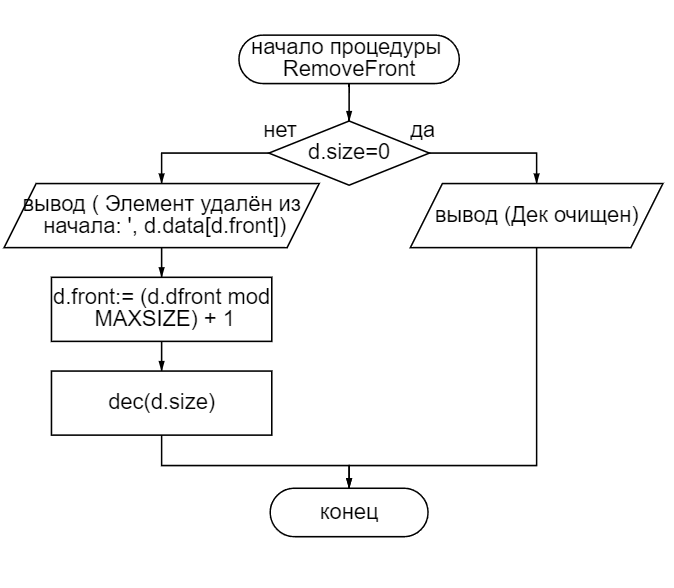


Рисунок 6 – Процедура RemoveFront

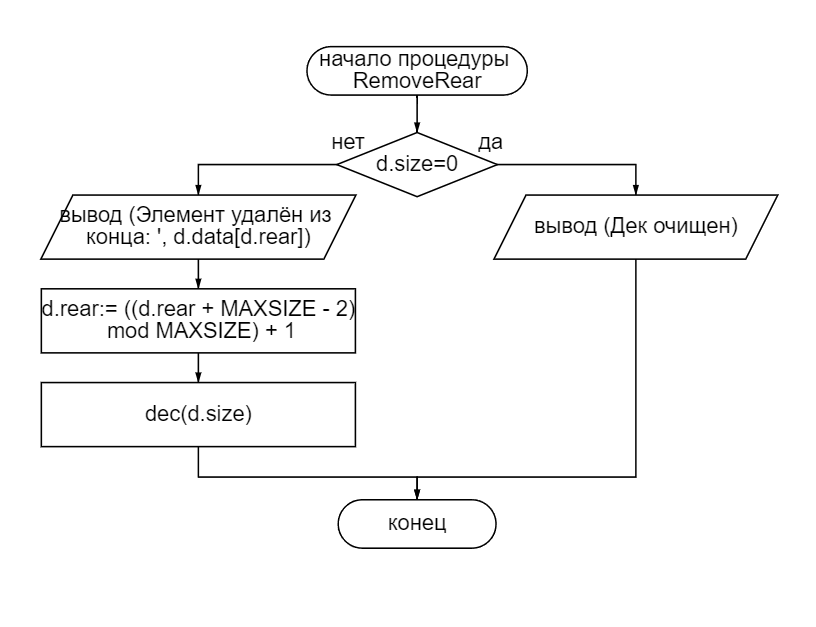


Рисунок 7 – Процедура RemoveRear

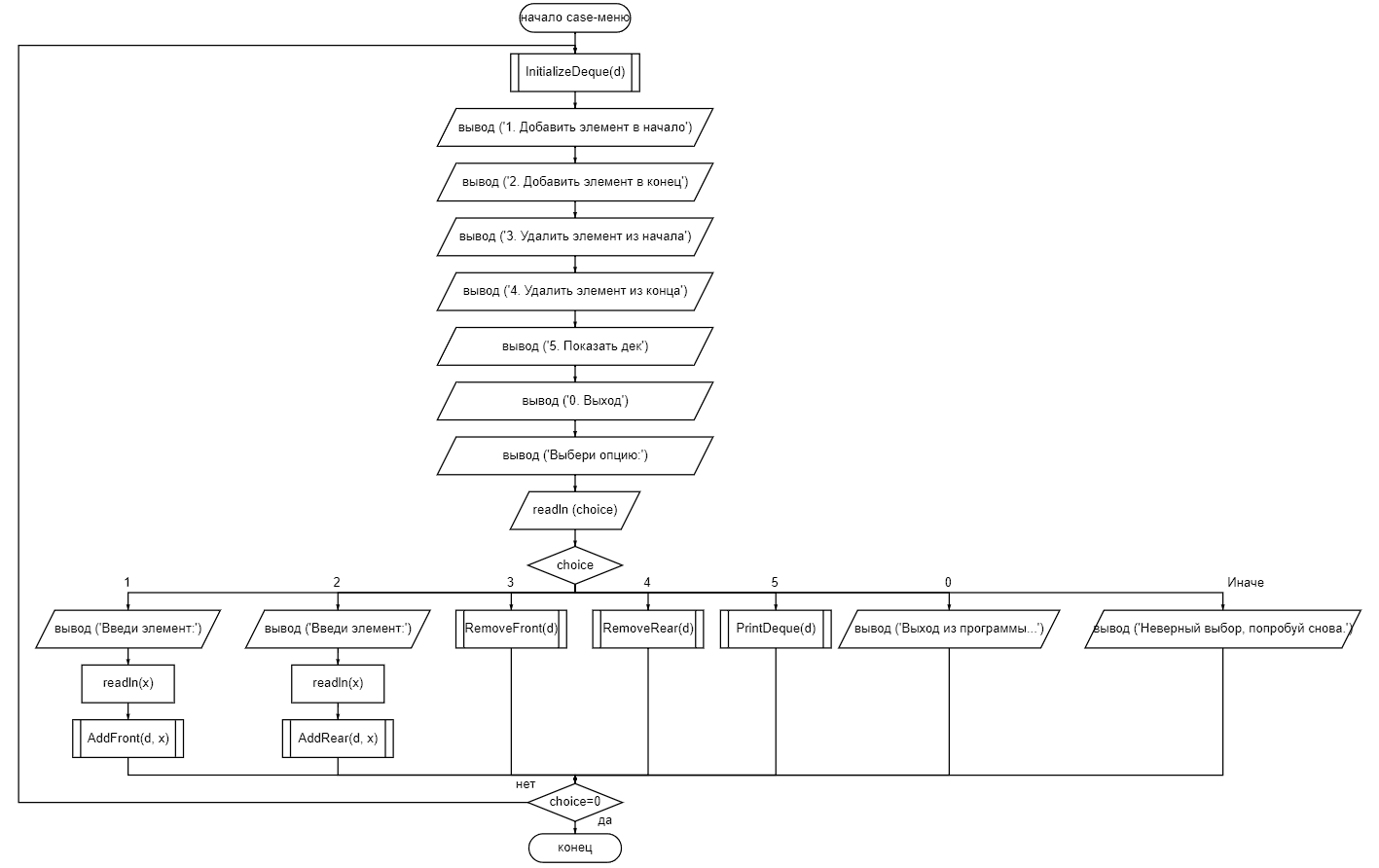


Рисунок 8 – Case-меню

**Задание:**

1. Написать программу для работы со структурой данных “Дек”
2. Структура данных должна быть реализована на основе статической памяти.
3. Работа со структурой должна осуществляться с помощью case-меню. Предусмотреть наглядную визуализацию содержимого структуры

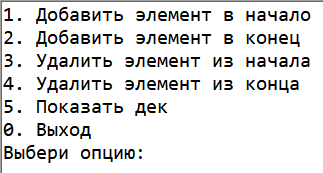


Рисунок 9 – Программа осуществляется с помощью case-меню

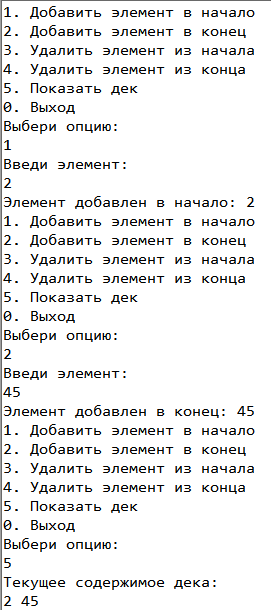


Рисунок 10 – Добавление элементов в дек

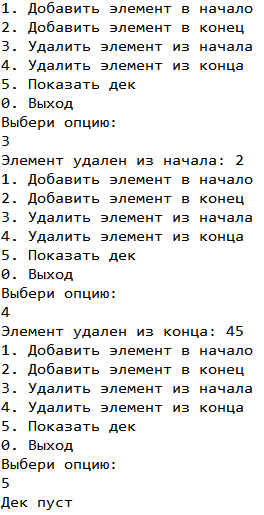


Рисунок 11 – Удаление элементов из дека

Код программы:

**program** DequeExample;

**const**

MAXSIZE = 5; // Максимальный размер дека

**type**

Deque = **record**

data: **array**[1..MAXSIZE] **of** integer;

front: integer;

rear: integer;

size: integer;

**end**;

**var**

d: Deque;

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

**begin**

d.front := 1;

d.rear := 0;

d.size := 0;

**end**;

**procedure** PrintDeque(**var** d: Deque);

**var**

i, idx: integer;

**begin**

**if** d.size = 0 **then**

writeln('Дек пустой')

**else**

**begin**

writeln('Текущее содержимое дека:');

idx := d.front;

**for** i := 1 **to** d.size **do**

**begin**

write(d.data[idx], ' ');

idx := (idx **mod** MAXSIZE) + 1;

**end**;

writeln;

**end**;

**end**;

**procedure** AddFront(**var** d: Deque; x: integer);

**begin**

**if** d.size = MAXSIZE **then**

writeln('Дек полностью заполнен')

**else**

**begin**

d.front := ((d.front + MAXSIZE - 2) **mod** MAXSIZE) + 1;

d.data[d.front] := x;

inc(d.size);

writeln('Элемент добавлен в начало: ', x);

**end**;

**end**;

**procedure** AddRear(**var** d: Deque; x: integer);

**begin**

**if** d.size = MAXSIZE **then**

writeln('Дек полностью заполнен')

**else**

**begin**

d.rear := (d.rear **mod** MAXSIZE) + 1;

d.data[d.rear] := x;

inc(d.size);

writeln('Элемент добавлен в конец: ', x);

**end**;

**end**;

**procedure** RemoveFront(**var** d: Deque);

**begin**

**if** d.size = 0 **then**

writeln('Дек очищен')

**else**

**begin**

writeln('Элемент удален из начала: ', d.data[d.front]);

d.front := (d.front **mod** MAXSIZE) + 1;

dec(d.size);

**end**;

**end**;

**procedure** RemoveRear(**var** d: Deque);

**begin**

**if** d.size = 0 **then**

writeln('Дек очищен')

**else**

**begin**

writeln('Элемент удален из конца: ', d.data[d.rear]);

d.rear := ((d.rear + MAXSIZE - 2) **mod** MAXSIZE) + 1;

dec(d.size);

**end**;

**end**;

**var**

choice, x: integer;

**begin**

InitializeDeque(d);

**repeat**

writeln('1. Добавить элемент в начало');

writeln('2. Добавить элемент в конец');

writeln('3. Удалить элемент из начала');

writeln('4. Удалить элемент из конца');

writeln('5. Показать дек');

writeln('0. Выход');

writeln('Выбери опцию:');

readln(choice);

**case** choice **of**

1: **begin**

writeln('Введи элемент:');

readln(x);

AddFront(d, x);

**end**;

2: **begin**

writeln('Введи элемент:');

readln(x);

AddRear(d, x);

**end**;

3: RemoveFront(d);

4: RemoveRear(d);

5: PrintDeque(d);

0: writeln('Выход из программы...');

**else**

writeln('Неверный выбор, попробуй снова.');

**end**;

**until** choice = 0;

**end**.

Ответы на вопросы:

1. Данный код представляет динамическую структуру данных - дек, описанную массивом с фиксированным размером. Дек обладает свойствами массива с двумя указателями на начало и конец, а также переменной size для отслеживания количества элементов.
2. Дек хранится при помощи массива data, где каждый элемент имеет индекс от 1 до MAXSIZE. Узлы связаны так, что указатель front указывает на начало, а rear на конец дека, и элементы добавляются и удаляются с обеих сторон.
3. Основные операции со структурой включают добавление элемента в начало или конец, удаление элемента из начала или конца, и вывод дека на экран.
4. Кейс-меню организовано с помощью цикла repeat-until, где пользователь выбирает опцию, вводя соответствующее число. Каждая опция вызывает соответствующую процедуру в зависимости от выбора.

Вывод:

В ходе выполнения данной работы вспомнил что такое case-меню, изучил структуру под названием “Дек”, узнал, как он работает и как работать с этой структурой. Так же понял, как работать со статической памятью на Pascal и написание наглядной визуализации содержимого структуры.