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 статистической памяти»**

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

Выполнил: студент учебной группы

ИСПк-203-52-00

Шишмакова Анастасия Сергеевна

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2025

**Цель работы:**

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

**Формулировка задания: Вариант 23**

* Написать программу для работы со структурой данных «Дек».
* Структура данных должна быть реализована на основе статистической памяти.
* Работа со структурой должна осуществляться с помощью case-меню. Предусмотреть наглядную визуализацию содержимого структуры.

**Описание алгоритма:**

Алгоритм программы

1. Инициализация дека:

Создается структура с массивом фиксированного размера (`MAX\_SIZE`), индексами front и rear (указывают на начало и конец), счетчиком элементов count.

2. Главный цикл меню:

Программа отображает пункты меню в цикле, пока пользователь не выберет выход:

- Добавить элемент в начало/конец.

- Удалить элемент из начала/конца.

- Показать содержимое дека.

- Завершить работу.

3. Обработка операций:

- Добавление:

- Проверка на переполнение.

- Сдвиг индекса front (для вставки в начало) или rear (для вставки в конец) с использованием модульной арифметики (кольцевой буфер).

- Увеличение счетчика элементов.

- Удаление:

- Проверка на пустоту.

- Сдвиг индекса front (удаление из начала) или rear (удаление из конца).

- Уменьшение счетчика элементов.

4. Визуализация:

- Вывод элементов в порядке их расположения в деке, начиная с front, с учетом кольцевой структуры массива.

5. Обработка ошибок:

- При попытке добавить в заполненный дек или удалить из пустого выводятся соответствующие сообщения.

**Схемы алгоритмов:**

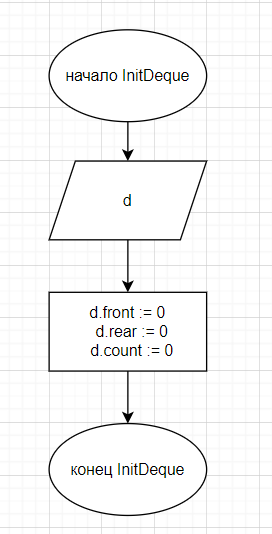


Рисунок 1 – схема алгоритма процедуры InitDeque

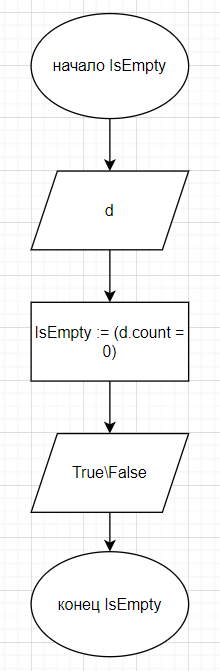


Рисунок 2 – схема алгоритма функции IsEmpty

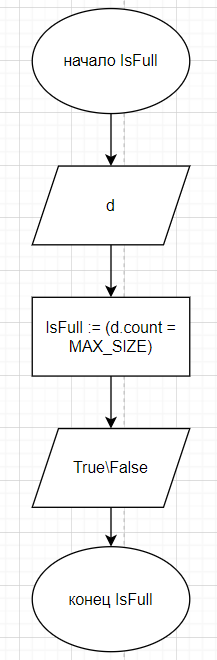


Рисунок 3 – схема алгоритма функции IsFull

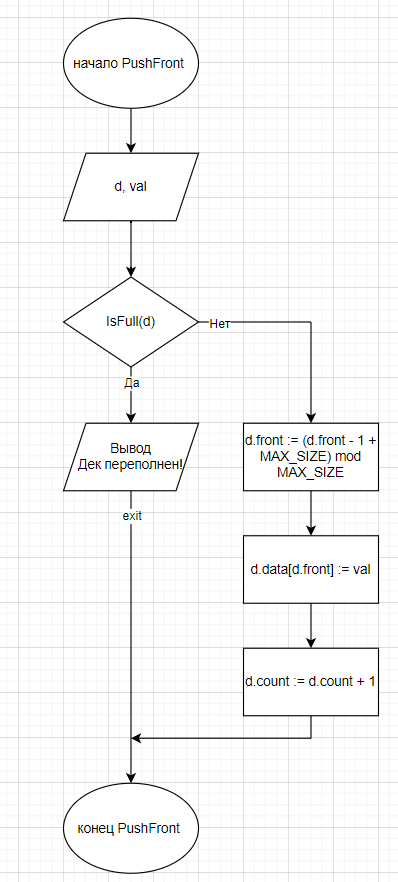


Рисунок 4 – схема алгоритма процедуры PushFront

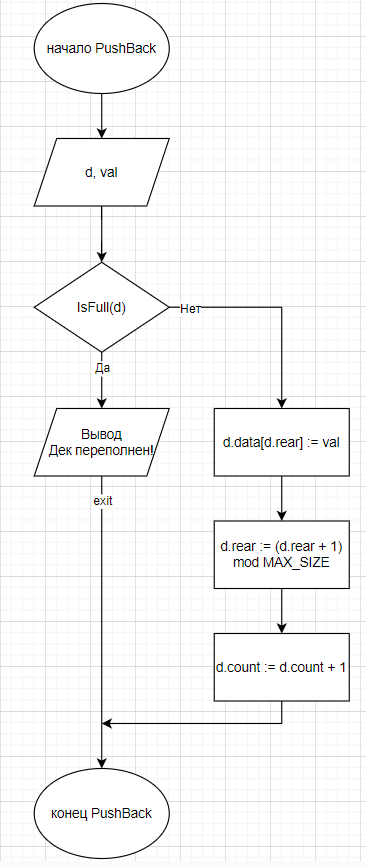


Рисунок 5 – схема алгоритма процедуры PushBack

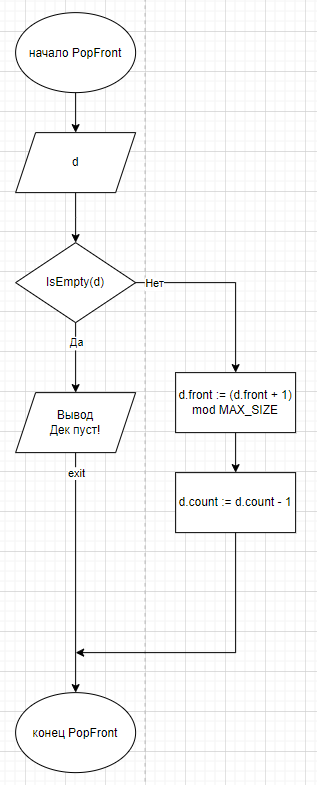


Рисунок 6 – схема алгоритма процедуры PopFront

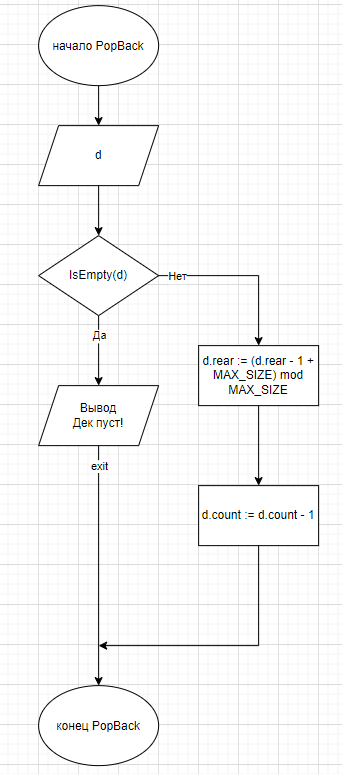


Рисунок 7 – схема алгоритма процедуры PopBack

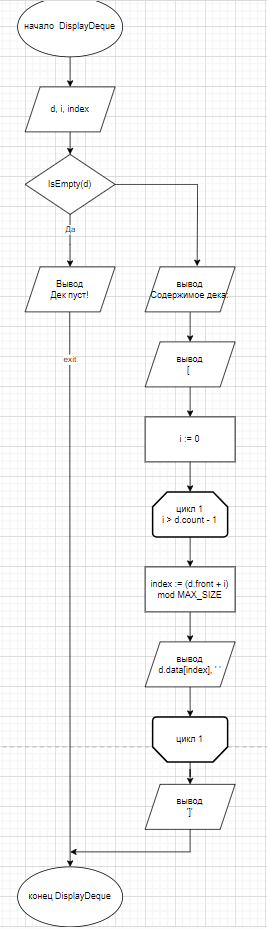


Рисунок 8 – схема алгоритма процедуры DisplayDeque

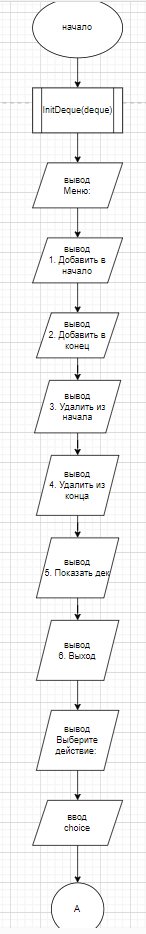


Рисунок 9.1 – схема алгоритма основной программы

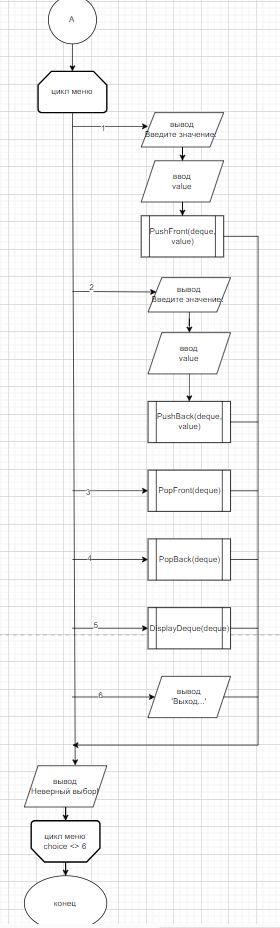


Рисунок 9.2 – продолжение схемы алгоритма основной программы

**Код программы:**

**const**

MAX\_SIZE = 10; // Максимальный размер дека

**type**

TDeque = **record**

data: **array**[0..MAX\_SIZE-1] **of** Integer;

front, rear, count: Integer;

**end**;

**var**

deque: TDeque;

choice, value: Integer;

// Инициализация дека

**procedure** InitDeque(**var** d: TDeque);

**begin**

d.front := 0;

d.rear := 0;

d.count := 0;

**end**;

// Проверка на пустоту

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

**begin**

IsEmpty := (d.count = 0);

**end**;

// Проверка на заполненность

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

**begin**

IsFull := (d.count = MAX\_SIZE);

**end**;

// Добавление в начало

**procedure** PushFront(**var** d: TDeque; val: Integer);

**begin**

**if** IsFull(d) **then**

**begin**

writeln('Дек переполнен!');

**exit**;

**end**;

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

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

d.count := d.count + 1;

**end**;

// Добавление в конец

**procedure** PushBack(**var** d: TDeque; val: Integer);

**begin**

**if** IsFull(d) **then**

**begin**

writeln('Дек переполнен!');

**exit**;

**end**;

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

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

d.count := d.count + 1;

**end**;

// Удаление из начала

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

**begin**

**if** IsEmpty(d) **then**

**begin**

writeln('Дек пуст!');

**exit**;

**end**;

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

d.count := d.count - 1;

**end**;

// Удаление из конца

**procedure** PopBack(**var** d: TDeque);

**begin**

**if** IsEmpty(d) **then**

**begin**

writeln('Дек пуст!');

**exit**;

**end**;

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

d.count := d.count - 1;

**end**;

// Визуализация дека

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

**var**

i, index: Integer;

**begin**

**if** IsEmpty(d) **then**

**begin**

writeln('Дек пуст!');

**exit**;

**end**;

writeln('Содержимое дека:');

write('[ ');

**for** i := 0 **to** d.count - 1 **do**

**begin**

index := (d.front + i) **mod** MAX\_SIZE;

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

**end**;

writeln(']');

**end**;

**begin**

InitDeque(deque);

**repeat**

writeln;

writeln('Меню:');

writeln('1. Добавить в начало');

writeln('2. Добавить в конец');

writeln('3. Удалить из начала');

writeln('4. Удалить из конца');

writeln('5. Показать дек');

writeln('6. Выход');

write('Выберите действие: ');

readln(choice);

**case** choice **of**

1: **begin**

write('Введите значение: ');

readln(value);

PushFront(deque, value);

**end**;

2: **begin**

write('Введите значение: ');

readln(value);

PushBack(deque, value);

**end**;

3: PopFront(deque);

4: PopBack(deque);

5: DisplayDeque(deque);

6: writeln('Выход...');

**else** writeln('Неверный выбор!');

**end**;

**until** choice = 6;

**end**.

**Результат выполнения программы:**

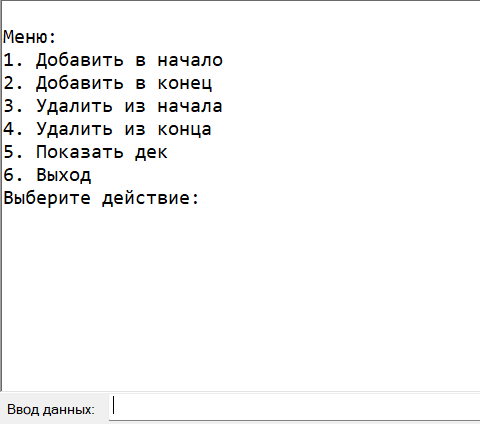


Рисунок . – меню программы

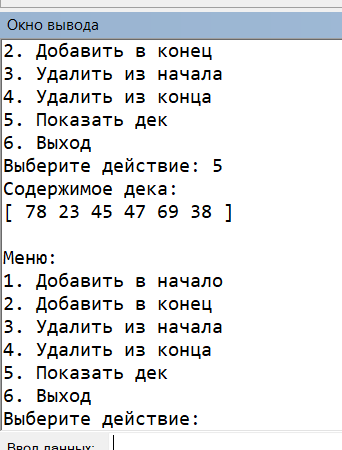


Рисунок . – выполнение программы

**Вывод:**

В ходе выполнения данной работы был изучен алгоритм для работы со структурой данных «Дек», а также работа с case-меню. Работа показалась интересной, были приобретены более глубокие знания в работе с паскалем.