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 ЭЛЕМЕНТАРЫНХ СТРУКТУР ДАННЫХ НА ОСНОВЕ СТАТИЧЕСКОЙ ПАМЯТИ»**

**ПО «МДК 05.02 РАЗРАБОТКА КОДА ИНФОРМАЦИОННЫХ СИСТЕМ»**

Выполнил: студент учебной группы ИСПк- 204-52-00

Кривошеин Дмитрий Васильевич

Преподаватель:

Сергеева Елизавета Григорьевна

г. Киров

2024

1. **Цель работы**: изучение принципов работы с базовыми структурами данных, полученных навыков организации case-меню.
2. **Задание:**
3. Написать программу для работы со структурой данных “Кольцевой двусвязный список”
4. Структура данных должна быть реализована на основе статической памяти
5. Работа со структурой должна осуществляться с помощью case-меню. Предусмотреть наглядную визуализацию содержимого структуры.
6. **Описание алгоритма:**

Круговой двусвязный список — это круговой связанный список, в котором каждый узел имеет две ссылки, соединяющие его с предыдущим узлом и следующим узлом.

Свойства кругового двусвязного списка:

* Гибкое выделение и освобождение памяти для элементов списка.
* Возможность динамического добавления и удаления элементов.
* Возможность быстрого доступа к элементам списка с помощью указателей.

Хранение массива будет осуществляться с помощью

массива List, где каждый элемент массива представляет один узел списка. Узлы связаны между собой с помощью поля next, которое указывает на индекс следующего элемента в списке. Голова списка хранится в переменной Head.

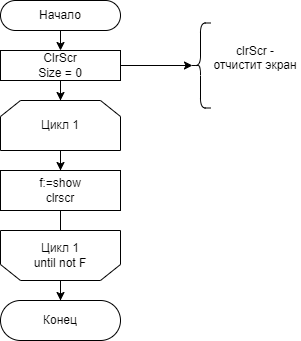
Основные операции со структурой:

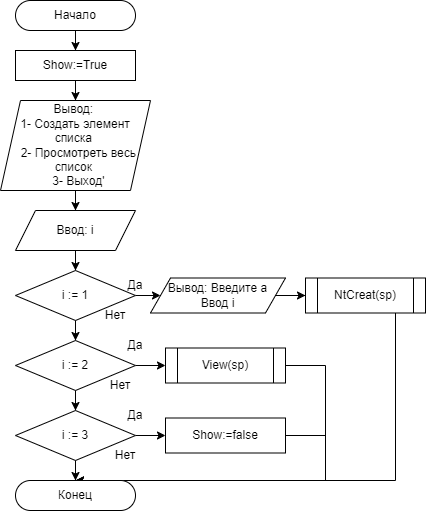
* Добавление элемента: Процедура NtCreat добавляет новый элемент в список. Если список пуст, созданный элемент становится головой списка. Если список не пуст, новый элемент добавляется в конец списка.
* Взятие элемента: Процедура View позволяет просматривать все элементы списка. Пользователь может перемещаться по списку и выбирать действия: далее, назад, закончить просмотр.
* Вывод на экран: Вывод элементов списка на экран осуществляется при просмотре списка в процедуре View.

Организация case-меню:

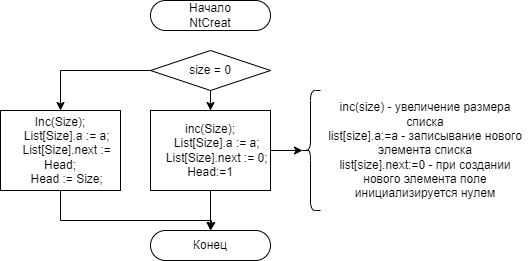
В коде функция Show, которая отображает меню с возможными действиями: создать элемент списка, просмотреть список, выход. Меню реализовано с помощью case для обработки выбора пользователя. Каждый выбор вызывает соответствующую процедуру или завершает программу.

1. **Схема алгоритма:**

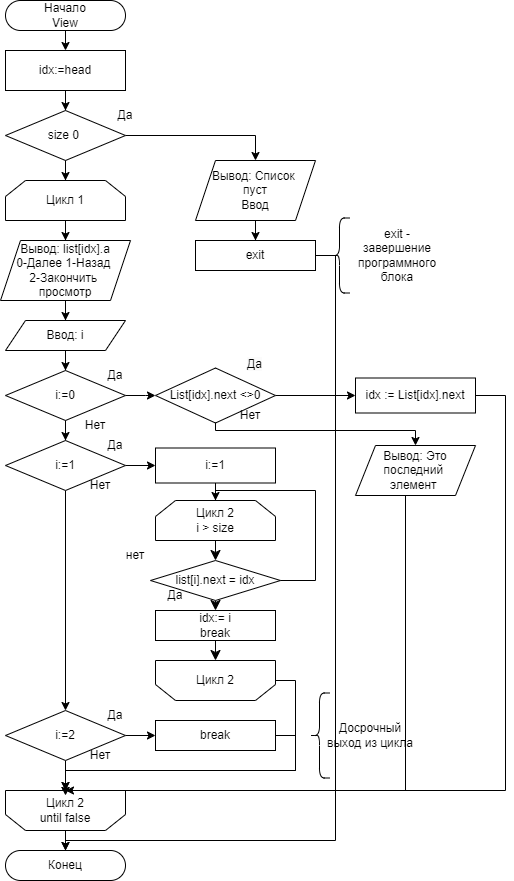
 **Рисунок 1 – алгоритм решения**



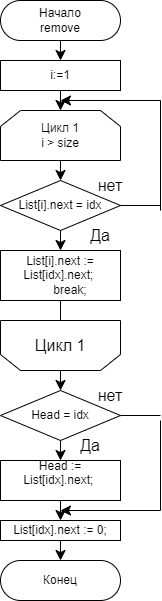
**Рисунок 2 -процедура show**



**Рисунок 3 – процедура NtCreat**



**Рисунок 4 – процедура View**



**Рисунок 5 – процедура remove**

**Код программ:**

**Program** List1;

**Uses** crt;

**Const** max = 100;

**Type**

intr = **record**

a: integer;

next: integer;

**end**;

**Var**

List: **array**[1..max] **of** intr;

Head: integer;

Size: integer;

**Procedure** NtCreat(a: integer);

**Begin**

**if** Size = 0 **then**

**begin**

Inc(Size);

List[Size].a := a;

List[Size].next := 0;

Head := 1;

**end**

**else**

**begin**

Inc(Size);

List[Size].a := a;

List[Size].next := Head;

Head := Size;

**end**;

**end**;

**Procedure** View;

**Var**

i, idx: integer;

**Begin**

idx := Head;

**if** Size = 0 **then**

**begin**

WriteLn('Список пуст! ');

readLn;

**exit**;

**end**

**else**

**repeat**

WriteLn(List[idx].a);

Write('0-Далее 1-Назад 2-Закончить просмотр ');

readLn(i);

**case** i **of**

0: **begin**

**if** List[idx].next <>0 **then**

idx := List[idx].next

**else**

**begin**

Writeln('Это последний элемент');

readln;

**end**;

**end**;

1: **begin**

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

**begin**

**if** List[i].next = idx **then**

**begin**

idx := i;

**break**;

**end**;

**end**;

**end**;

2: **break**;

**end**;

**until** false;

**end**;

**Procedure** remove(idx: integer);

**Var**

i: integer;

**Begin**

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

**begin**

**if** List[i].next = idx **then**

**begin**

List[i].next := List[idx].next;

**break**;

**end**;

**end**;

**if** Head = idx **then**

Head := List[idx].next;

List[idx].next := 0;

**end**;

**Function** Show: boolean;

**Var**

i: string;

**Begin**

Show := true;

WriteLn('1- Создать элемент списка ');

Writeln('2- Просмотреть весь список ');

WriteLn('3- Удаление элемента списка');

Writeln('4- Выход');

readLn(i);

**case** i **of**

'1': **begin**

Write('Введите а: ');

readLn(i);

NtCreat(StrToInt(i));

**end**;

'2': View;

'3': **begin**

WriteLn('Введите номер элемента для удаления: ');

readLn(i);

**if** i = '' **then** write('Нечего удалять')

**else**

remove(StrToInt(i));

**end**;

'4': Show := false;

**end**;

**end**;

**Var**

F: boolean;

**Begin**

ClrScr;

Size := 0;

**repeat**

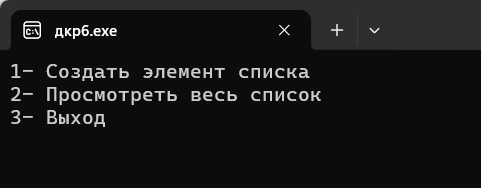
F := Show;

clrscr;

**until not** F;

**End**.

**Результат выполнения программ:**



1. **Вывод:**

Программы были успешно разработаны и выполнены согласно поставленной задаче. Вот основные выводы:

Мы изучили принципы работы с базовыми структурами данных, а именно с “Кольцевым двусвязным списком”, освежили свои знания в работе с case-меню, а также в работе с указателями и списками, в будущем это может помочь нам при оптимизации и улучшении код. Попрактиковались в реализации структуры данных на основе статической памяти