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»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Вершинин Григорий Сергеевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2025

**Цель работы:**

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

**Формулировка задания:**

* написать программу для работы со структурой данных «Кольцевой односвязный список»;
* структура данных должна быть реализована на основе статической памяти;
* работа со структурой должна осуществляться с помощью case-меню. Предусмотреть наглядную визуализацию содержимого структуры.

**Описание алгоритма:**

Инициализация списка:

* Используется массив Nodes для хранения узлов списка;
* Head указывает на начало списка;
* FreeList управляет свободными узлами.

Добавление элемента:

* Элемент добавляется в конец списка;
* Если список пуст, создается новый узел, который ссылается сам на себя.

Удаление элемента:

* Элемент ищется в списке и удаляется, если найден.

Поиск элемента:

* Элемент ищется в списке, и выводится сообщение о его наличии.

Вывод списка:

* Содержимое списка выводится в виде последовательности элементов.

Меню:

* Пользователь может выбирать действия через case-меню.

**Схемы алгоритмов:**

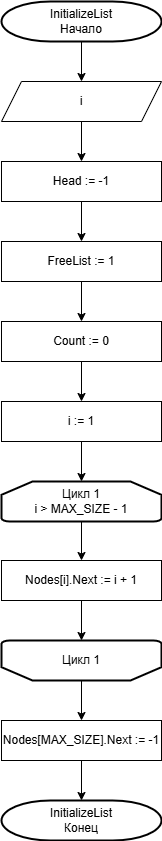
****

Рисунок 1 – Схема алгоритма процедуры InitializeList

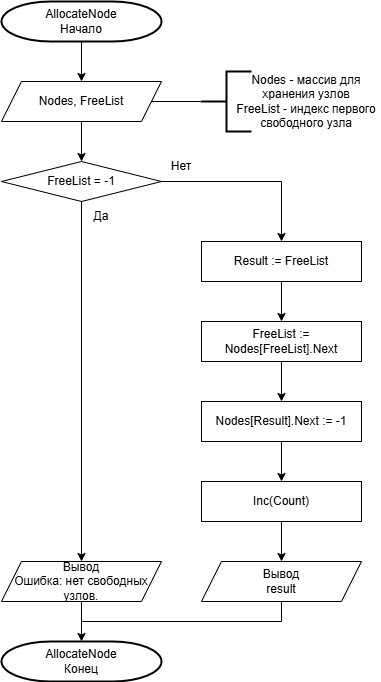


Рисунок 2 – Схема алгоритма функции AllocateNode

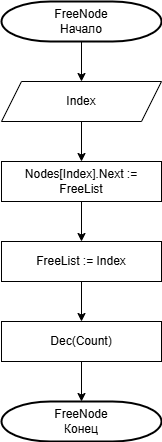


Рисунок 3 – Схема алгоритма процедуры FreeNode

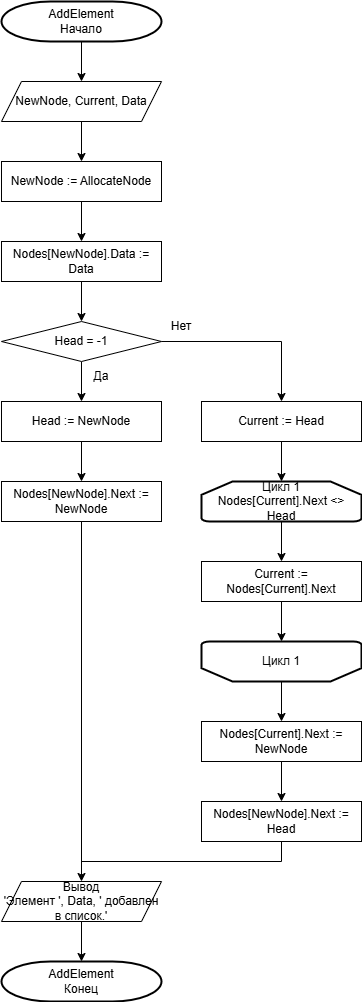


Рисунок 4 – Схема алгоритма процедуры AddElement

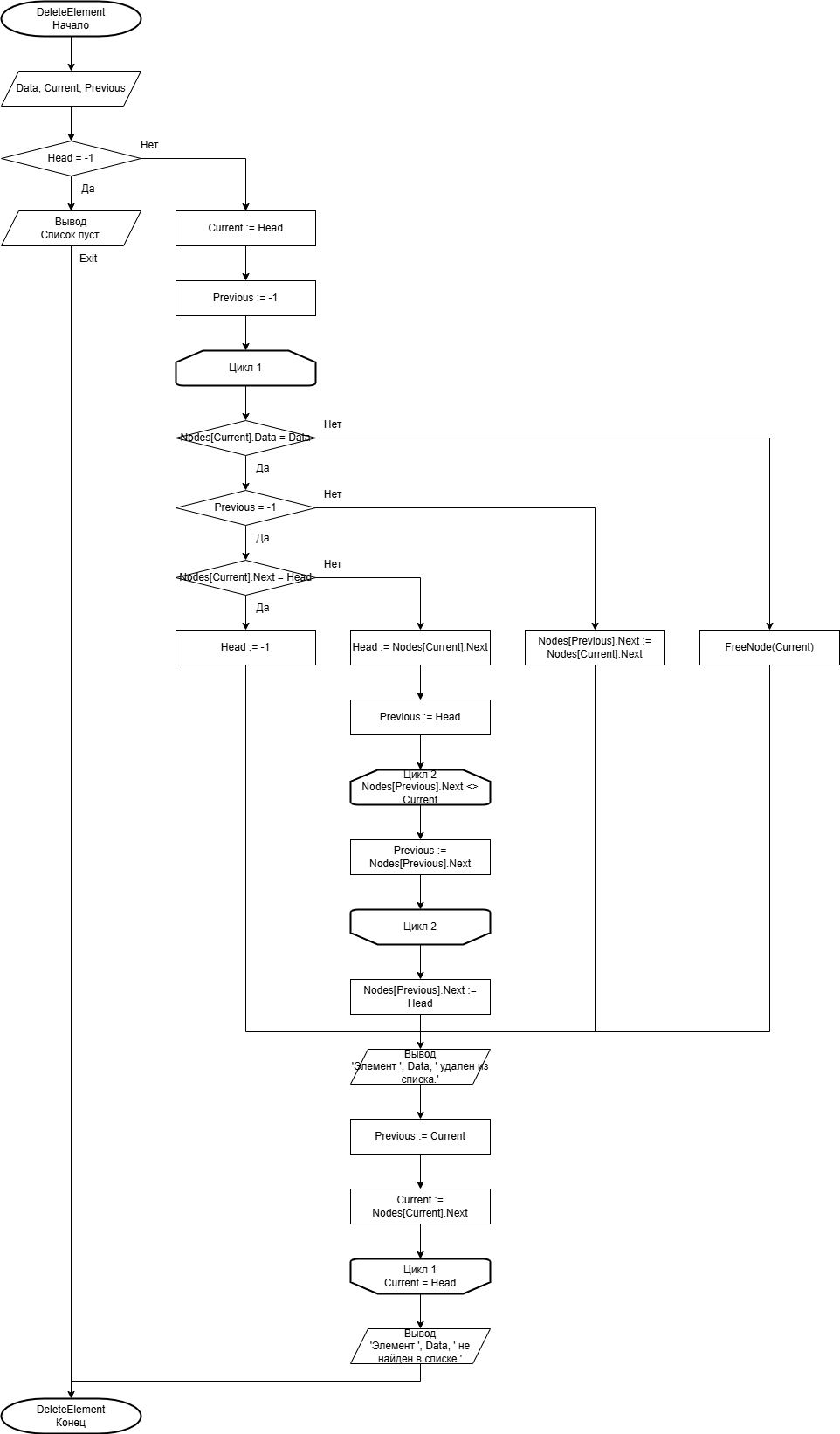


Рисунок 5 – Схема алгоритма процедуры DeleteElement

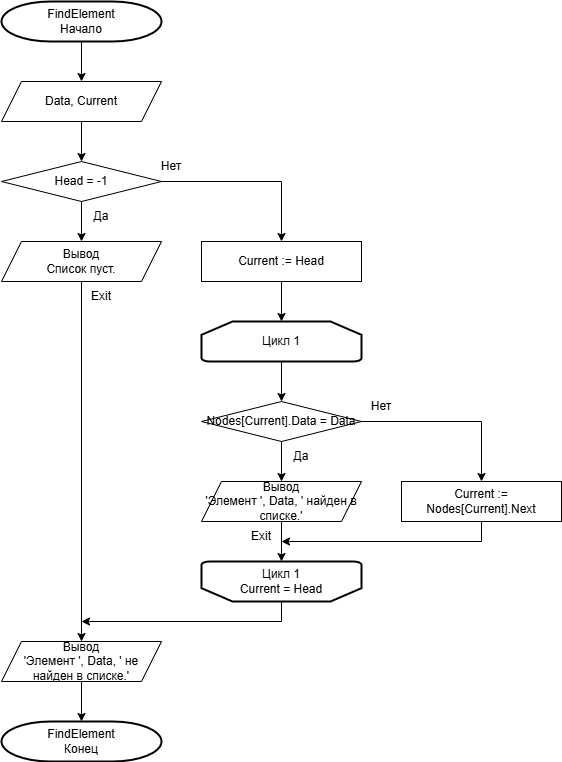


Рисунок 6 – Схема алгоритма процедуры FindElement

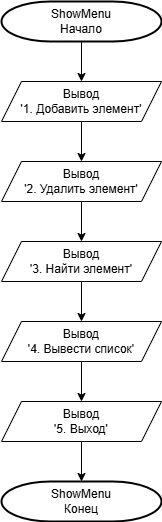


Рисунок 7 – Схема алгоритма процедуры ShowMenu

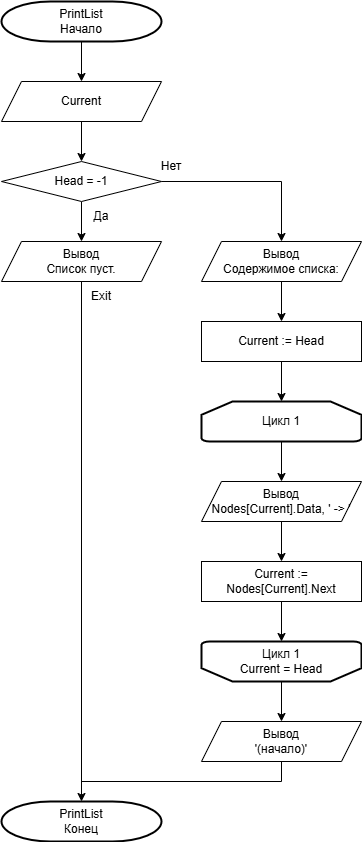


Рисунок 8 – Схема алгоритма процедуры PrintList

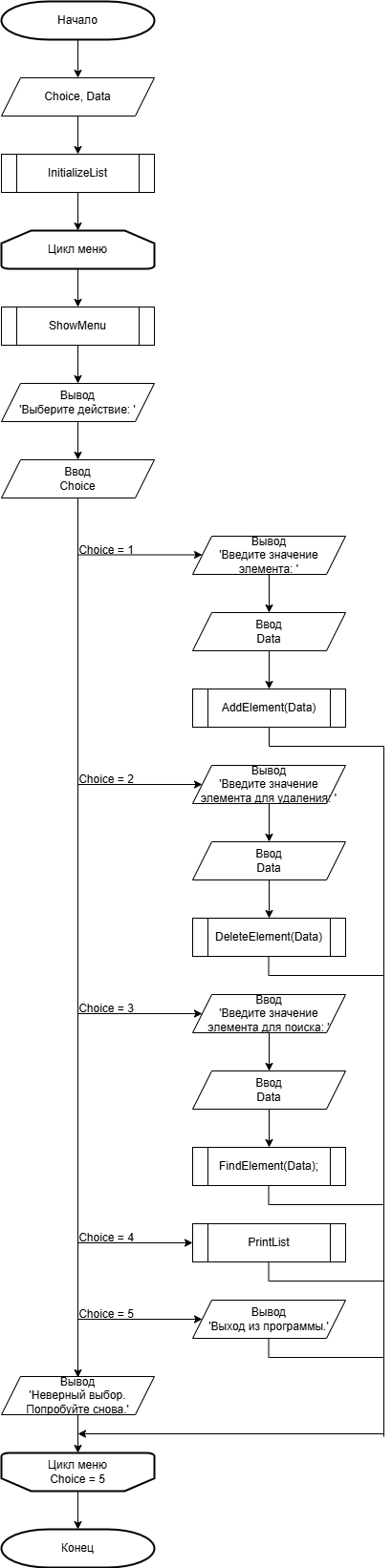


Рисунок 9 – Схема алгоритма основного кода программы

**Код программы:**

**const**

MAX\_SIZE = 100; // Максимальный размер списка

**type**

TNode = **record**

Data: Integer;

Next: Integer; // Индекс следующего элемента

**end**;

**var**

Nodes: **array**[1..MAX\_SIZE] **of** TNode; // Массив для хранения узлов

Head: Integer; // Индекс головы списка

FreeList: Integer; // Индекс первого свободного узла

Count: Integer; // Количество элементов в списке

// Инициализация списка

**procedure** InitializeList;

**var**

i: Integer;

**begin**

Head := -1; // Список пуст

FreeList := 1; // Начинаем с первого узла

Count := 0;

// Инициализация свободного списка

**for** i := 1 **to** MAX\_SIZE - 1 **do**

**begin**

Nodes[i].Next := i + 1;

**end**;

Nodes[MAX\_SIZE].Next := -1; // Конец свободного списка

**end**;

// Выделение нового узла

**function** AllocateNode: Integer;

**begin**

**if** FreeList = -1 **then**

**begin**

Writeln('Ошибка: нет свободных узлов.');

**end**;

Result := FreeList;

FreeList := Nodes[FreeList].Next;

Nodes[Result].Next := -1; // Новый узел пока ни на что не указывает

Inc(Count);

**end**;

// Освобождение узла

**procedure** FreeNode(Index: Integer);

**begin**

Nodes[Index].Next := FreeList;

FreeList := Index;

Dec(Count);

**end**;

// Добавление элемента в список

**procedure** AddElement(Data: Integer);

**var**

NewNode, Current: Integer;

**begin**

NewNode := AllocateNode;

Nodes[NewNode].Data := Data;

**if** Head = -1 **then**

**begin**

Head := NewNode;

Nodes[NewNode].Next := NewNode; // Кольцевой список

**end**

**else**

**begin**

Current := Head;

**while** Nodes[Current].Next <> Head **do**

**begin**

Current := Nodes[Current].Next;

**end**;

Nodes[Current].Next := NewNode;

Nodes[NewNode].Next := Head;

**end**;

Writeln('Элемент ', Data, ' добавлен в список.');

**end**;

// Удаление элемента из списка

**procedure** DeleteElement(Data: Integer);

**var**

Current, Previous: Integer;

**begin**

**if** Head = -1 **then**

**begin**

Writeln('Список пуст.');

**Exit**;

**end**;

Current := Head;

Previous := -1;

**repeat**

**if** Nodes[Current].Data = Data **then**

**begin**

**if** Previous = -1 **then**

**begin**

**if** Nodes[Current].Next = Head **then**

**begin**

Head := -1; // Список стал пустым

**end**

**else**

**begin**

Head := Nodes[Current].Next;

Previous := Head;

**while** Nodes[Previous].Next <> Current **do**

**begin**

Previous := Nodes[Previous].Next;

**end**;

Nodes[Previous].Next := Head;

**end**;

**end**

**else**

**begin**

Nodes[Previous].Next := Nodes[Current].Next;

**end**;

FreeNode(Current);

Writeln('Элемент ', Data, ' удален из списка.');

**end**;

Previous := Current;

Current := Nodes[Current].Next;

**until** Current = Head;

Writeln('Элемент ', Data, ' не найден в списке.');

**end**;

// Поиск элемента в списке

**procedure** FindElement(Data: Integer);

**var**

Current: Integer;

**begin**

**if** Head = -1 **then**

**begin**

Writeln('Список пуст.');

**Exit**;

**end**;

Current := Head;

**repeat**

**if** Nodes[Current].Data = Data **then**

**begin**

Writeln('Элемент ', Data, ' найден в списке.');

**Exit**;

**end**;

Current := Nodes[Current].Next;

**until** Current = Head;

Writeln('Элемент ', Data, ' не найден в списке.');

**end**;

// Вывод списка

**procedure** PrintList;

**var**

Current: Integer;

**begin**

**if** Head = -1 **then**

**begin**

Writeln('Список пуст.');

**Exit**;

**end**;

Writeln('Содержимое списка:');

Current := Head;

**repeat**

Write(Nodes[Current].Data, ' -> ');

Current := Nodes[Current].Next;

**until** Current = Head;

Writeln('(начало)');

**end**;

**procedure** ShowMenu;

**begin**

Writeln('1. Добавить элемент');

Writeln('2. Удалить элемент');

Writeln('3. Найти элемент');

Writeln('4. Вывести список');

Writeln('5. Выход');

**end**;

**var**

Choice, Data: Integer;

**begin**

InitializeList;

**repeat**

ShowMenu;

Write('Выберите действие: ');

Readln(Choice);

**case** Choice **of**

1:

**begin**

Write('Введите значение элемента: ');

Readln(Data);

AddElement(Data);

**end**;

2:

**begin**

Write('Введите значение элемента для удаления: ');

Readln(Data);

DeleteElement(Data);

**end**;

3:

**begin**

Write('Введите значение элемента для поиска: ');

Readln(Data);

FindElement(Data);

**end**;

4:

**begin**

PrintList;

**end**;

5:

**begin**

Writeln('Выход из программы.');

**end**;

**else**

Writeln('Неверный выбор. Попробуйте снова.');

**end**;

**until** Choice = 5;

**end**.

**Результат выполнения программы:**

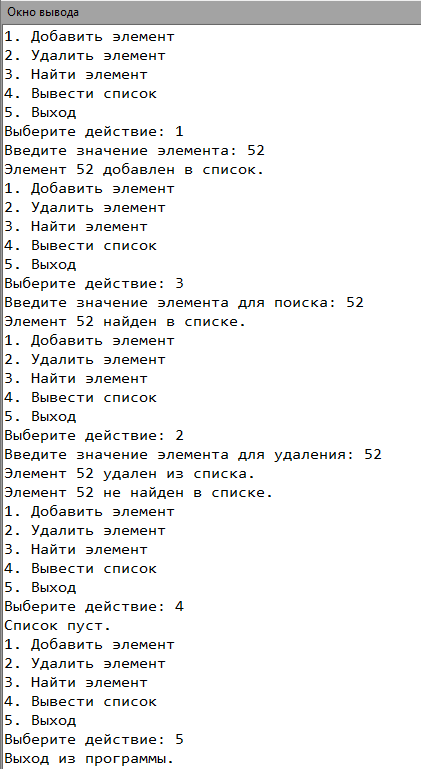
****

Рисунок 10 – Результат выполнения программы

**Вывод:** работа со структурой данных «кольцевой односвязный список» оказалась не самой легкой. Но основные функции работы с ним удалось реализовать, а именно: возможность просмотра элементов в списке, возможность добавления элементов в список и удаления из него элементов, а также присутствие элемента по его значению. Все управление программой пользователь может осуществлять с помощью case-меню.