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 элементов структурных данных на основе статической памяти»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Коробейникова Софья Николаевна

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

В отчете должны отображаться:

1. **Цель работы:** изучение принципов работы с базовыми структурами данных, получение навыков организации case-меню.
2. **Формулировка задания (с вариантом)**

Вариант 20

Задание

1. ﻿﻿﻿ Написать программу для работы со структурой данных "Односвязный список.
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.
4. **Описание алгоритма**

1. Объявление типа Node для представления узла списка. Каждый узел содержит поле data для хранения данных и указатель next на следующий узел.

2. Объявление переменной head, которая будет указывать на начало списка.

3. Создание процедуры InsertNode(data: Integer), которая добавляет новый узел в конец списка. В процедуре создается новый узел с переданными данными, затем проверяется, пуст ли список. Если список пуст, то новый узел становится началом списка, иначе новый узел добавляется в конец списка.

4. Создание процедуры DisplayList, которая отображает содержимое списка. Процедура проходит по всем узлам списка, начиная с начала, и выводит данные каждого узла.

5. Основная часть программы начинается с инициализации переменной head как nil и запуска цикла repeat-until, который позволяет пользователю выбирать действия из меню.

6. В меню предоставлены три варианта действий: добавить элемент в список, отобразить список и завершить программу. Пользователь выбирает действие, после чего программа выполняет соответствующую операцию.

7. При выборе добавления элемента в список пользователь вводит данные для нового элемента, которые затем передаются в процедуру InsertNode.

8. При выборе отображения списка вызывается процедура DisplayList, которая выводит содержимое списка на экран.

9. При выборе завершения программы программа выводит сообщение о завершении и завершает свою работу.

1. **Схема алгоритма с комментариями**

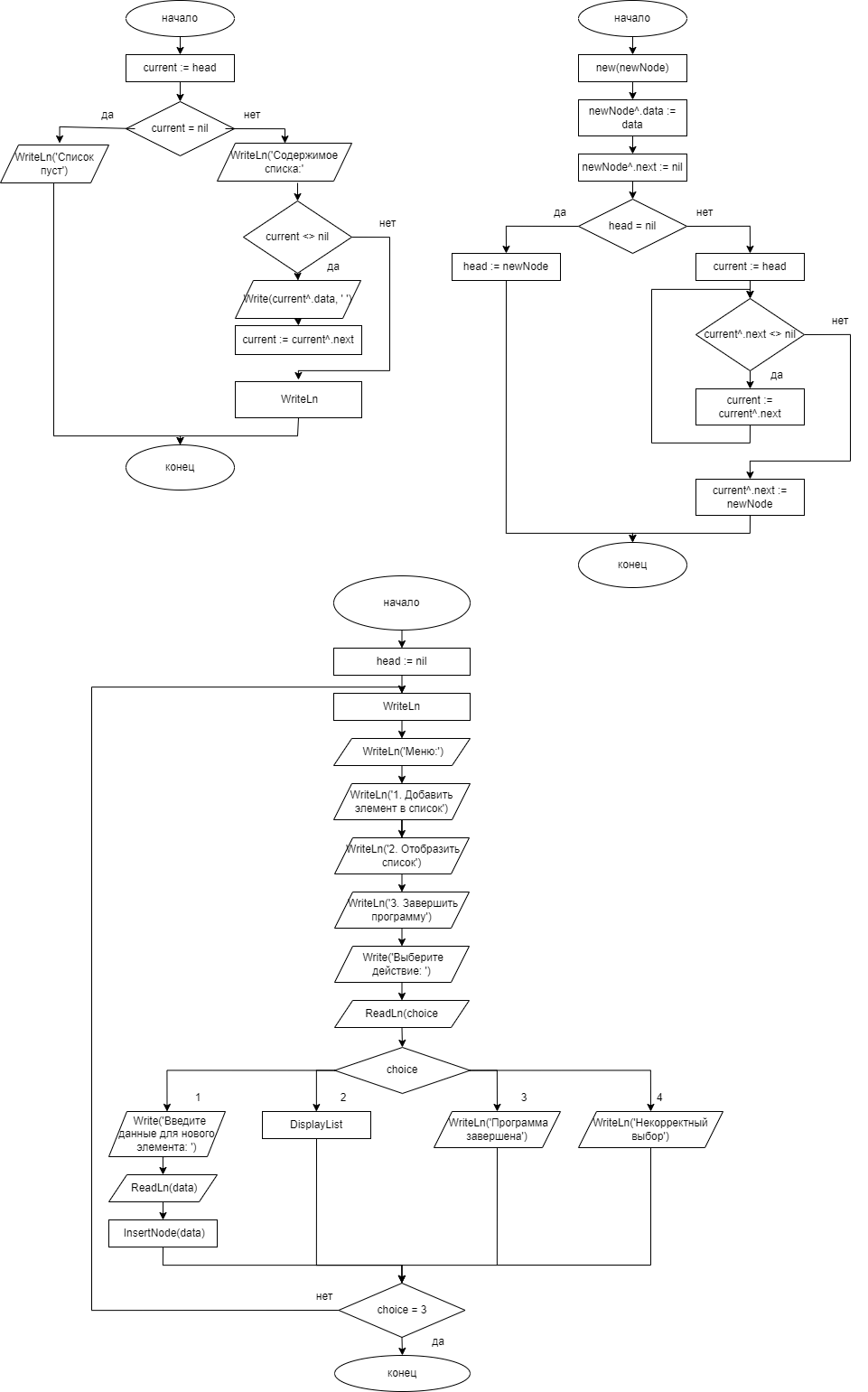
****

Рисунок 3

1. **Код программы**

**type**

Node = **record**

data: Integer;

next: ^Node;

**end**;

**var**

head: ^Node;

// Процедура для добавления нового узла в конец списка

**procedure** InsertNode(data: Integer);

**var**

newNode, current: ^Node;

**begin**

**new**(newNode);

newNode^.data := data;

newNode^.next := nil;

**if** head = nil **then**

head := newNode

**else begin**

current := head;

**while** current^.next <> nil **do**

current := current^.next;

current^.next := newNode;

**end**;

**end**;

// Процедура для отображения содержимого списка

**procedure** DisplayList;

**var**

current: ^Node;

**begin**

current := head;

**if** current = nil **then**

WriteLn('Список пуст')

**else begin**

WriteLn('Содержимое списка:');

**while** current <> nil **do begin**

Write(current^.data, ' ');

current := current^.next;

**end**;

WriteLn;

**end**;

**end**;

**var**

choice, data: Integer;

**begin**

head := nil;

**repeat**

WriteLn;

WriteLn('Меню:');

WriteLn('1. Добавить элемент в список');

WriteLn('2. Отобразить список');

WriteLn('3. Завершить программу');

Write('Выберите действие: ');

ReadLn(choice);

**case** choice **of**

1: **begin**

Write('Введите данные для нового элемента: ');

ReadLn(data);

InsertNode(data);

**end**;

2: DisplayList;

3: WriteLn('Программа завершена');

**else**

WriteLn('Некорректный выбор');

**end**;

**until** choice = 3;

**end**.

1. **Результат выполнения программы**

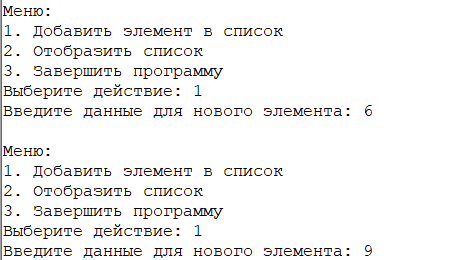


Рисунок 2

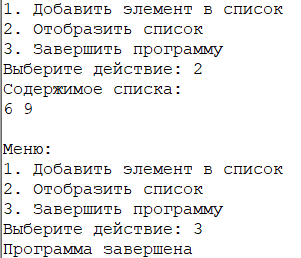


Рисунок 3

1. **Вывод**

Программа работает корректно и выполняет все указанные шаги. Пользователю предоставляется возможность добавлять элементы в список, отображать его содержимое, а также завершить программу. Все операции выполняются успешно, что позволяет эффективно управлять списком данных.