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»**

Выполнил: студент учебной группы

ИСПк-202-52-00

Суслов Павел Андреевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

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

2.Формулировка задания (с вариантом):

Вариант 15

1) Написать программу для работы со структурой данных "Кольцевой 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.

3. Код программы

**uses** Crt;

**const**

MAX\_NODES = 100;

**type**

TNode = **record**

data: integer;

next: integer;

prev: integer;

**end**;

**var**

head, tail: integer;

nodeArray: **array** [1..MAX\_NODES] **of** TNode;

nodeCounter: integer;

**procedure** InsertNode(**var** head, tail: integer; data: integer);

**var**

node: integer;

**begin**

Inc(nodeCounter);

node := nodeCounter;

nodeArray[node].data := data;

nodeArray[node].next := 0;

nodeArray[node].prev := 0;

**if** head = 0 **then**

**begin**

head := node;

tail := node;

**end**

**else**

**begin**

nodeArray[tail].next := node;

nodeArray[node].prev := tail;

tail := node;

**end**;

**end**;

**procedure** DeleteNode(**var** head, tail: integer; data: integer);

**var**

node, prevNode, nextNode: integer;

**begin**

node := head;

**while** (node <> 0) **and** (nodeArray[node].data <> data) **do**

**begin**

node := nodeArray[node].next;

**end**;

**if** node <> 0 **then**

**begin**

prevNode := nodeArray[node].prev;

nextNode := nodeArray[node].next;

**if** prevNode <> 0 **then**

**begin**

nodeArray[prevNode].next := nextNode;

**end**

**else**

**begin**

head := nextNode;

**end**;

**if** nextNode <> 0 **then**

**begin**

nodeArray[nextNode].prev := prevNode;

**end**

**else**

**begin**

tail := prevNode;

**end**;

Dec(nodeCounter);

**end**;

**end**;

**procedure** PrintList(head: integer);

**var**

node: integer;

**begin**

node := head;

**while** node <> 0 **do**

**begin**

Write(nodeArray[node].data, ' ');

node := nodeArray[node].next;

**end**;

Writeln;

**end**;

**var**

choice: integer;

data: integer;

**begin**

head := 0;

tail := 0;

nodeCounter := 0;

**repeat**

Writeln('Введите число: ');

Writeln('1. Ввод данных');

Writeln('2. Удалить данные');

Writeln('3. Вывести данные списка');

Writeln('4. Выход');

Readln(choice);

**case** choice **of**

1: **begin**

Write('Введите данные: ');

Readln(data);

InsertNode(head, tail, data);

**end**;

2: **begin**

Write('Введите данные: ');

Readln(data);

DeleteNode(head, tail, data);

**end**;

3: **begin**

Writeln('Список данных:');

PrintList(head);

**end**;

**end**;

**until** (choice = 4);

**end**.

4. Алгоритм построения программы

Объявляем массив элементов списка и два указателя на голову и хвост списка. Инициализируем указатели значением nil и текущий размер списка - 0. Для создания нового узла используем функцию NewNode, которая выделяет память под новый узел и присваивает ему переданное значение data.  
Для добавления элемента в конец списка используем процедуру AddToEnd.  
Проверяем, что список не заполнен (size < MAX\_SIZE), иначе выводим сообщение об ошибке и выходим из процедуры.  
Создаем новый узел с переданным значением data.  
Если список пуст (size = 0), то голова и хвост списка указывают на новый узел, иначе: Устанавливаем указатель next на последний элемент списка (tail).  
Обновляем указатель хвоста списка на новый узел.  
Устанавливаем указатель next нового узла на голову списка.  
Увеличиваем размер списка на 1.  
Для вывода списка на экран используем процедуру PrintList.  
Проверяем, что список не пуст (size > 0), иначе выводим сообщение об ошибке и выходим из процедуры.  
Устанавливаем указатель на голову списка.  
Используем цикл repeat-until для прохода по всем элементам списка, начиная с головы:  
Выводим значение текущего узла на экран.  
Переходим к следующему узлу списка.  
Повторяем цикл до тех пор, пока не дойдем до головы списка..

5. Результаты выполнения программы

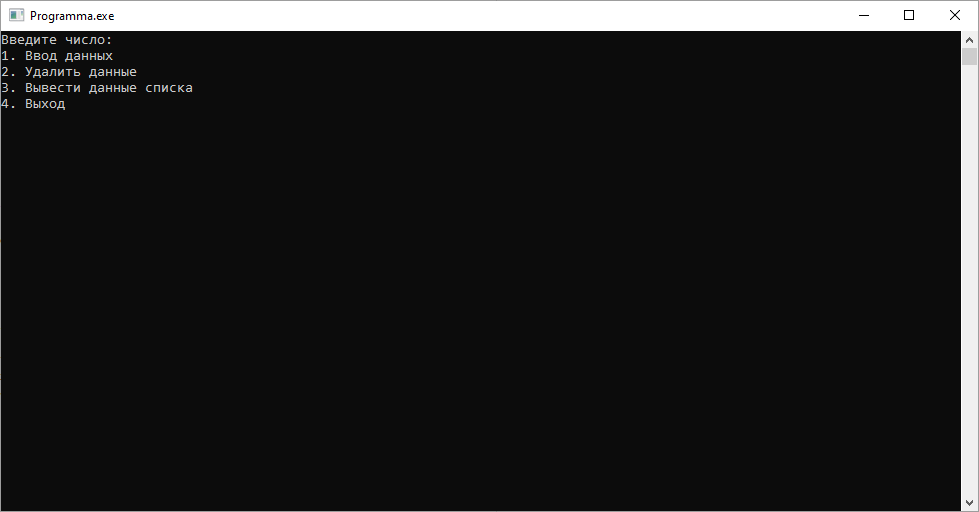


Рисунок 1 - Результаты программы (1)

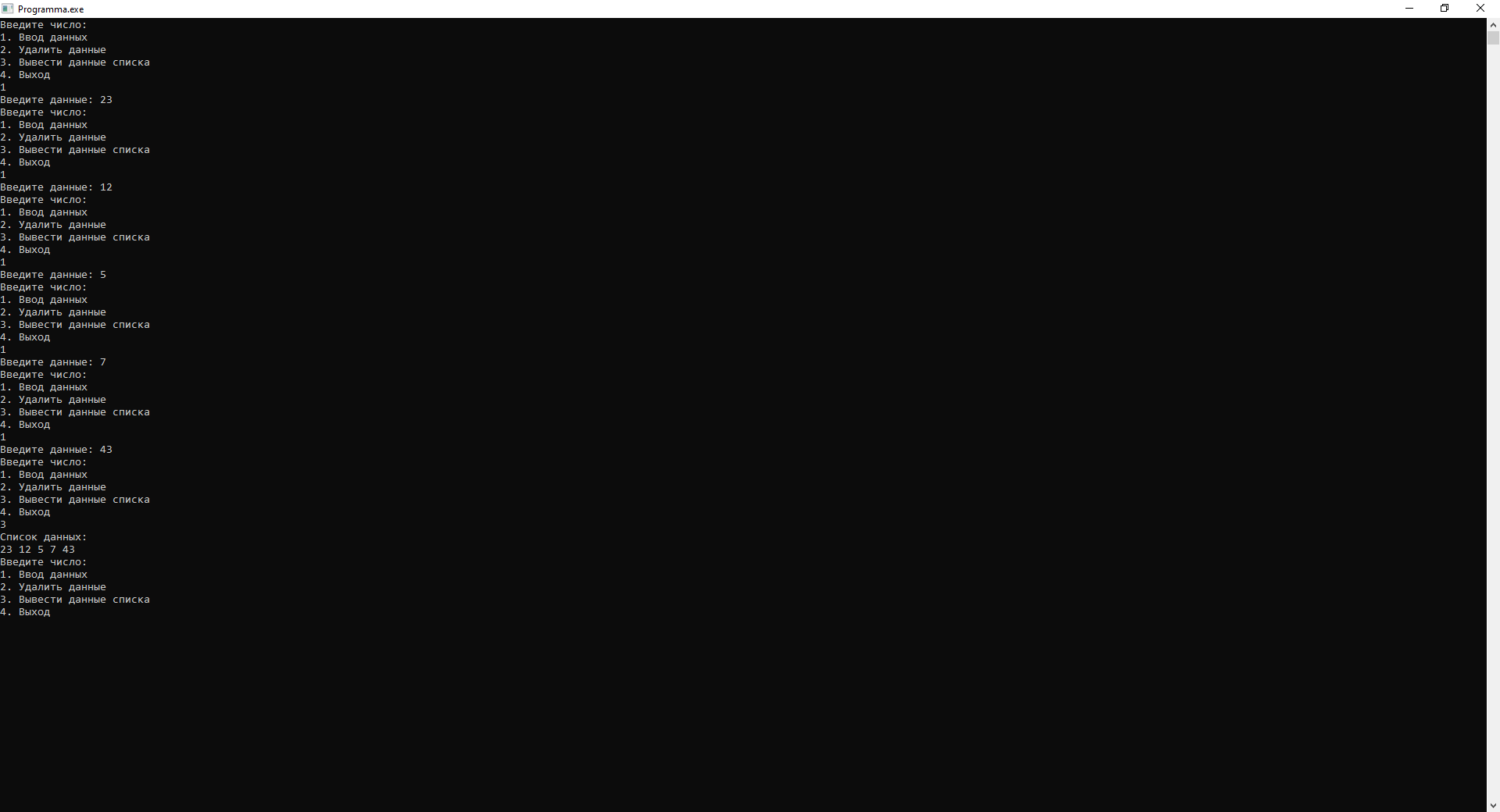


Рисунок 2 - Результаты программы (2)

6. Вывод

В ходе выполнения домашней контрольной работы №6 «реализация элементарных структур данных на основе статической памяти». Изучили принципы работы с базовыми структурами данных, получили навыки организации case-меню и закрепили знания во время работы.

Данная программа реализует кольцевой односвязный список. Пользователь может выбрать действие из меню: вставка данных, удаление данных, вывод списка данных или выход из меню. Программа использует статическую память для создания новых узлов в списке с помощью массива. Удаление узла также реализовано с помощью статической памяти. Вывод списка данных происходит циклически, проходя по всем узлам списка и выводя значение каждого узла.

Во время работы возникали трудности, по типу не правильном выборе памяти, не статической, а динамической. Проблемы удалось решить и с работой справились в полном объеме.