Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №2**

**«Реализация элементарных структур данных на основе статической памяти»**

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

Выполнил: студент учебной группы

ИСПк-202-52-00

Ситников Иван Владимирович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

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

**Задание**

1. Написать программу для работы со структурой данных «Кольцевой двусвязный список».
2. Структура должна быть реализована на основе статической и динамической памяти.
3. Работа со структурой должна осуществляться с помощью case-меню. Предусмотреть наглядную визуализацию содержимого структуры.

**Описание алгоритма.**

Кольцевой двусвязный список, в котором каждый узел хранит в себе три значения: адрес предыдущего узла, информацию, а также адрес следующего узла. Стоит отметить, что последний элемент списка ссылается на первый.

Изменение данной структуры (добавление и удаление элементов) возможно только при её организации на основе динамической памяти.

Реализация кольцевого двусвязного списка на основе статической памяти, возможна, при создании массива, в котором каждый элемент – узел списка.

**Схема алгоритма**

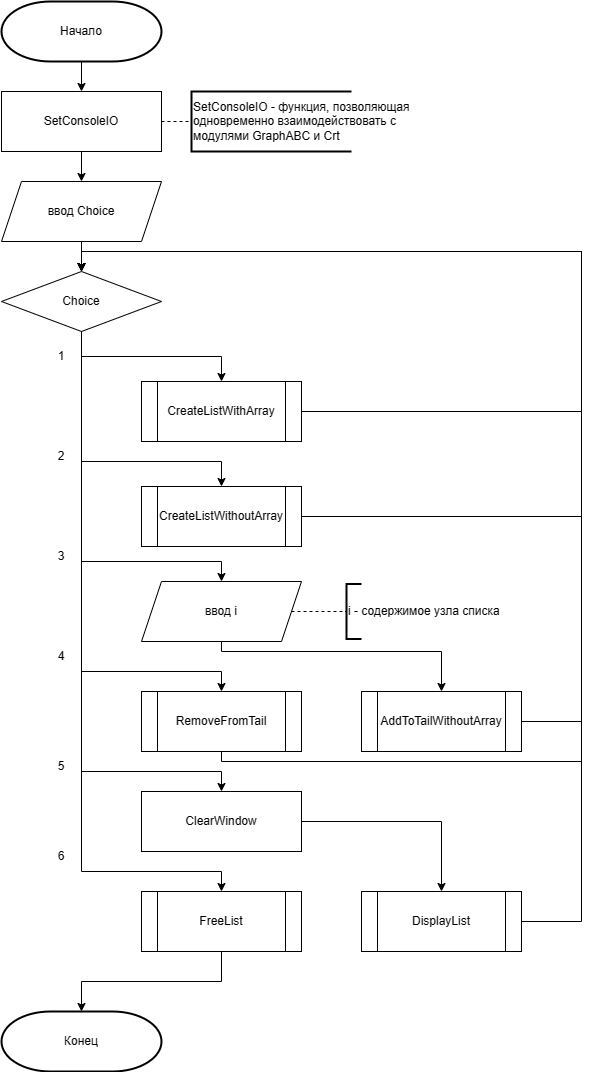


Рисунок 1 – Основное тело программы

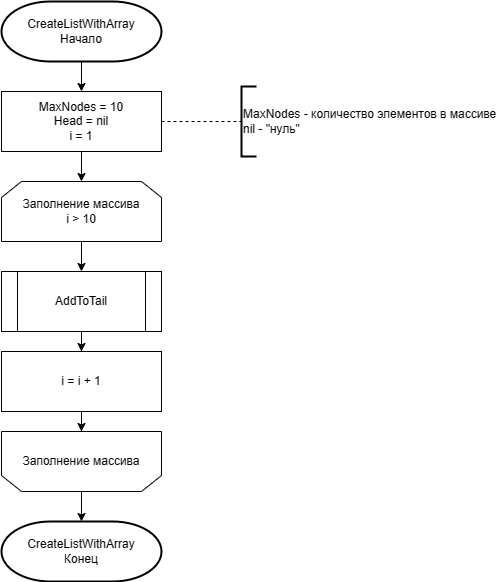


Рисунок 2 – Процедура заполнения массива

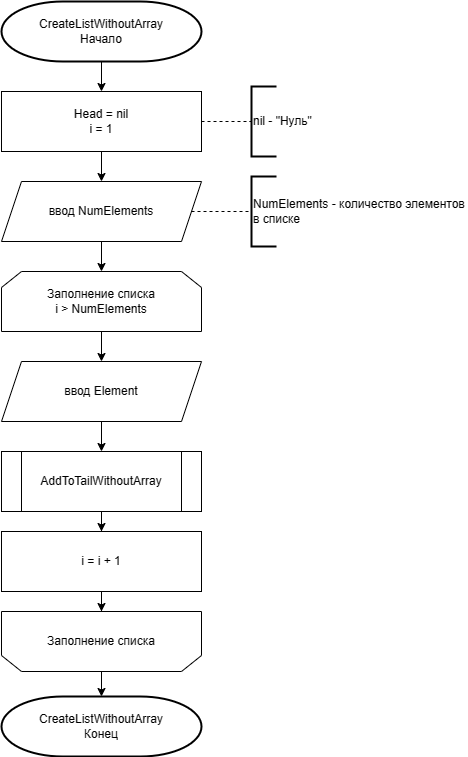


Рисунок 3 – Процедура заполнения списка

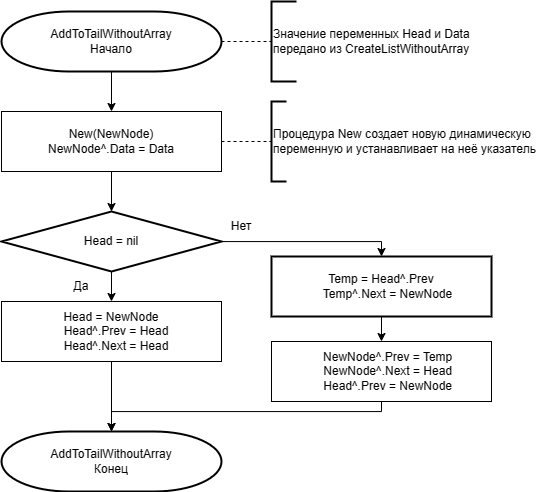


Рисунок 4 – Добавление узла списка без массива

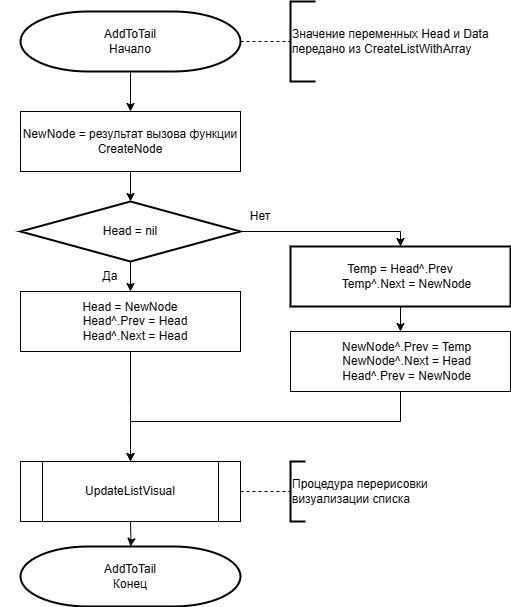


Рисунок 5 – Добавление узла списка в массив

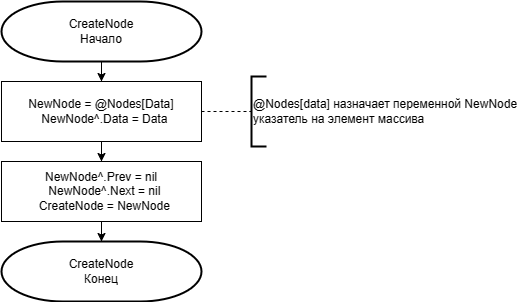


Рисунок 6 – Создание узла

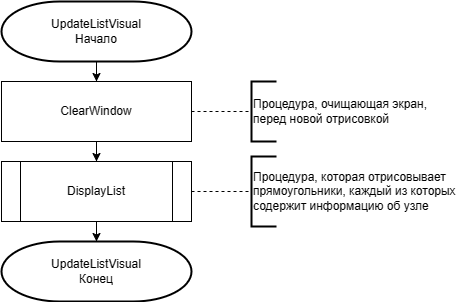


Рисунок 7 – Процедура перерисовки списка

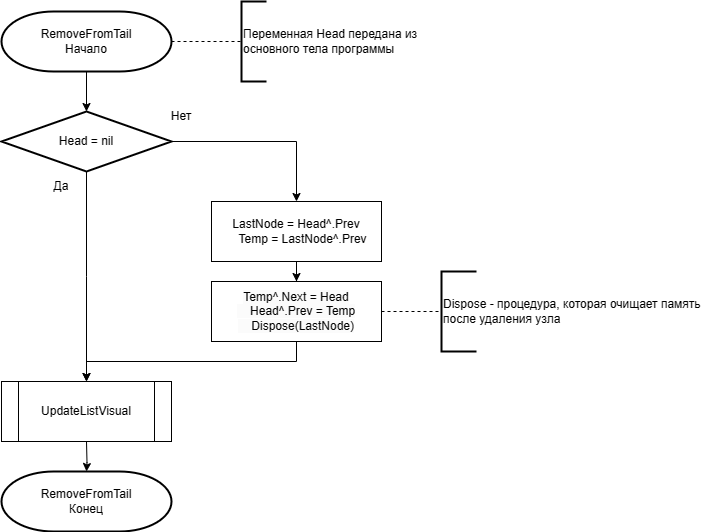


Рисунок 8 – Процедура удаления узла

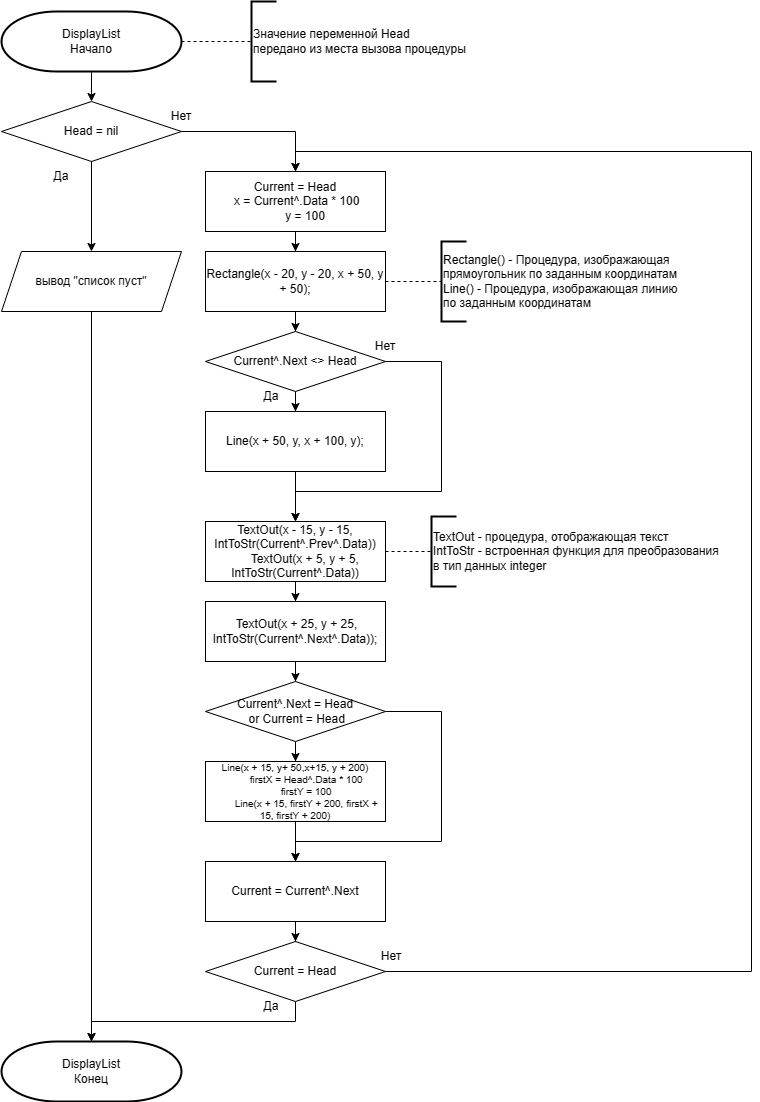


Рисунок 9 – Процедура визуализации списка

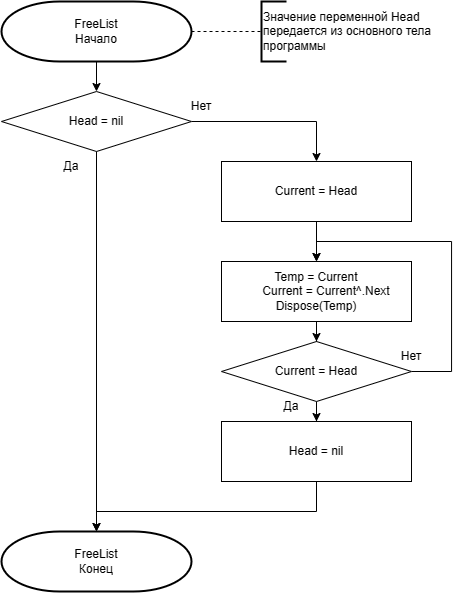


Рисунок 10 – Процедура полной очистки списка

**Результат выполнения программы**

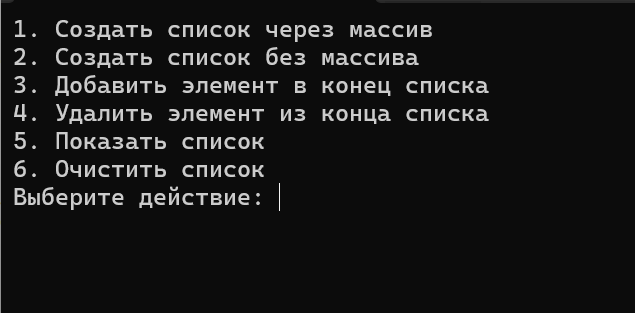
****

Рисунок 11 – Case-меню

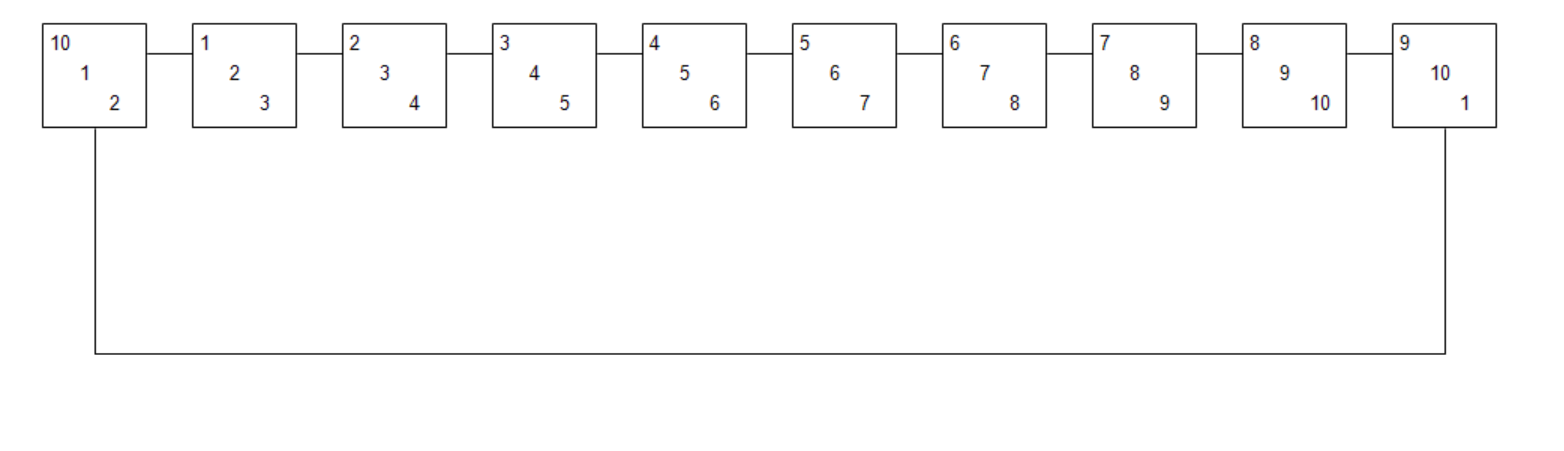
****

Рисунок 12 – Список на основе статической памяти (фиксированный размер 10)

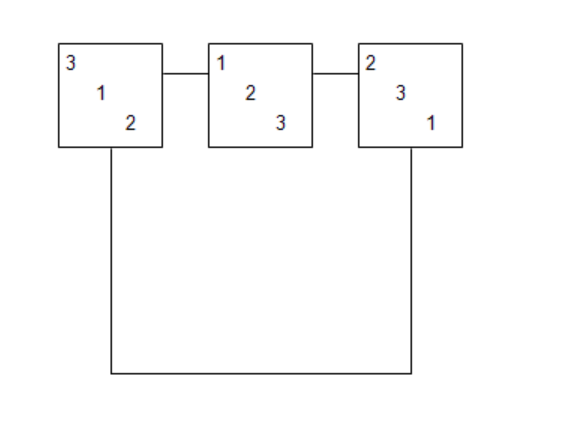


Рисунок 13 – Список на основе динамической памяти

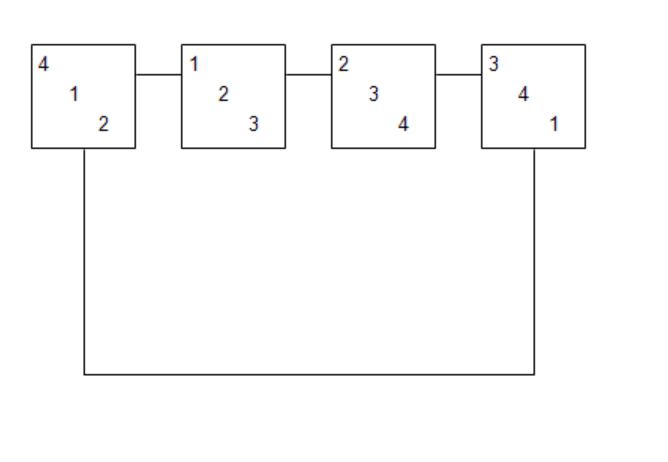


Рисунок 14 – Добавлен новый элемент списка (4)

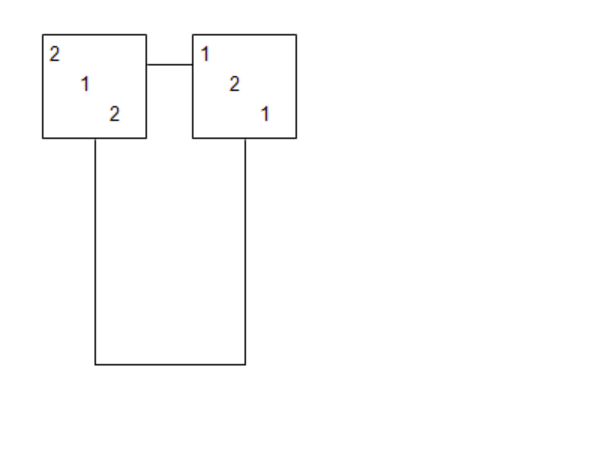


Рисунок 15 – Удалены 2 элемента списка

**Код программы**

**program** CircularDoublyLinkedList;

**uses** Crt, GraphABC;

**const**

MaxNodes = 10;

**type**

Node = **record**

Data: Integer;

Prev, Next: ^Node;

**end**;

**var**

Nodes: **array**[1..MaxNodes] **of** Node;

Head: ^Node;

i: Integer;

**procedure** DisplayList(Head: ^Node);

**var**

Current: ^Node;

x, y: Integer;

firstX, firstY: Integer;

**begin**

**if** Head <> nil **then**

**begin**

Current := Head;

**repeat**

x := Current^.Data \* 100;

y := 100;

Rectangle(x - 20, y - 20, x + 50, y + 50);

**if** Current^.Next <> Head **then**

**begin**

Line(x + 50, y, x + 100, y);

**end**;

TextOut(x - 15, y - 15, IntToStr(Current^.Prev^.Data));

TextOut(x + 5, y + 5, IntToStr(Current^.Data));

TextOut(x + 25, y + 25, IntToStr(Current^.Next^.Data));

**if** (Current^.Next = Head) **or** (Current = Head) **then**

**begin**

Line(x + 15, y+ 50,x+15, y + 200);

firstX := Head^.Data \* 100;

firstY := 100;

Line(x + 15, firstY + 200, firstX + 15, firstY + 200);

**end**;

Current := Current^.Next;

**until** Current = Head;

**end**

**else**

Writeln('Список пуст.');

**end**;

**function** CreateNode(Data: Integer): ^Node;

**var**

NewNode: ^Node;

**begin**

NewNode := @Nodes[Data];

NewNode^.Data := Data;

NewNode^.Prev := nil;

NewNode^.Next := nil;

CreateNode := NewNode;

**end**;

**procedure** UpdateListVisual(Head: ^Node);

**begin**

ClearWindow;

DisplayList(Head);

**end**;

**procedure** AddToTail(**var** Head: ^Node; Data: Integer);

**var**

NewNode, Temp: ^Node;

**begin**

NewNode := CreateNode(Data);

**if** Head = nil **then**

**begin**

Head := NewNode;

Head^.Prev := Head;

Head^.Next := Head;

**end**

**else**

**begin**

Temp := Head^.Prev;

Temp^.Next := NewNode;

NewNode^.Prev := Temp;

NewNode^.Next := Head;

Head^.Prev := NewNode;

**end**;

UpdateListVisual(Head);

**end**;

**procedure** RemoveFromTail(**var** Head: ^Node);

**var**

Temp, LastNode: ^Node;

**begin**

**if** Head <> nil **then**

**begin**

LastNode := Head^.Prev;

Temp := LastNode^.Prev;

Temp^.Next := Head;

Head^.Prev := Temp;

Dispose(LastNode);

**end**;

UpdateListVisual(Head);

**end**;

**procedure** AddToTailWithoutArray(**var** Head: ^Node; Data: Integer);

**var**

NewNode, Temp: ^Node;

**begin**

**New**(NewNode);

NewNode^.Data := Data;

**if** Head = nil **then**

**begin**

Head := NewNode;

Head^.Prev := Head;

Head^.Next := Head;

**end**

**else**

**begin**

Temp := Head^.Prev;

Temp^.Next := NewNode;

NewNode^.Prev := Temp;

NewNode^.Next := Head;

Head^.Prev := NewNode;

**end**;

**end**;

**procedure** FreeList(**var** Head: ^Node);

**var**

Current, Temp: ^Node;

**begin**

**if** Head <> nil **then**

**begin**

Current := Head;

**repeat**

Temp := Current;

Current := Current^.Next;

Dispose(Temp);

**until** Current = Head;

Head := nil;

**end**;

**end**;

**procedure** CreateListWithArray;

**var** i:integer;

**begin**

Head := nil;

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

AddToTail(Head, i);

**end**;

**procedure** CreateListWithoutArray;

**var**

NumElements, Element,i: Integer;

**begin**

Head := nil;

Write('Введите количество элементов в списке: ');

Readln(NumElements);

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

**begin**

Write('Введите элемент ', i, ': ');

Readln(Element);

AddToTailWithoutArray(Head, Element);

**end**;

**end**;

**procedure** MainMenu;

**var**

Choice: Integer;

**begin**

SetConsoleIO;

**repeat**

Writeln('1. Создать список через массив');

Writeln('2. Создать список без массива');

Writeln('3. Добавить элемент в конец списка');

Writeln('4. Удалить элемент из конца списка');

Writeln('5. Показать список');

Writeln('6. Очистить список');

Write('Выберите действие: ');

Readln(Choice);

ClrScr;

**case** Choice **of**

1: CreateListWithArray;

2: CreateListWithoutArray;

3: **begin**

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

Readln(i);

AddToTailWithoutArray(Head, i);

**end**;

4: RemoveFromTail(Head);

5:**begin**

ClearWindow;

DisplayList(Head);

**end**;

6: FreeList(Head);

**end**;

**until** Choice = 6;

**end**;

**begin**

SetWindowSize(1000, 600);

MainMenu;

**end**.

**Вывод**

В ходе написания программы, реализующей структуру кольцевой двусвязный список, были изучены основные структуры данных, такие как список и его виды, а также стек, дек и очередь.

В итоге, была разработана программа, которая позволяет создать кольцевой двусвязный список на основе статической либо динамической памяти, управление которым осуществляется с помощью case-меню. А также реализована наглядная визуализация содержимого структуры.