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 «РАЗРАБОТКА КОДА ИНФЛОРМАЦИОННЫХ СИСТЕМ»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Носкова Анастасия Владимировна

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

1. **Цель работы**

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

1. **Формулировка задания**

Вариант: 16

1. Написать программу для работы со структурой данных "Кольцевой односвязный список".
2. Структура данных должна быть реализована на основе статической памяти и динамической.
3. Работа со структурой должна осуществляться с помощью case-меню. Предусмотреть наглядную визуализацию содержимого структуры.
4. **Схемы алгоритма**

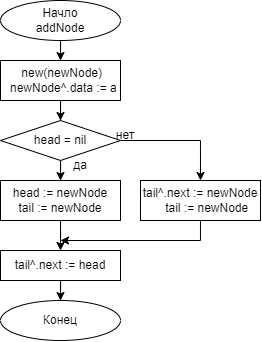


Рисунок 1 – Процедура addNode

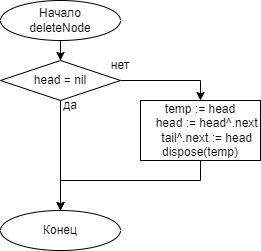


Рисунок 2 – Процедура deleteNode

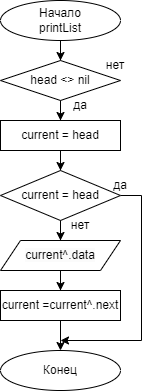


Рисунок 3 – Процедура printList

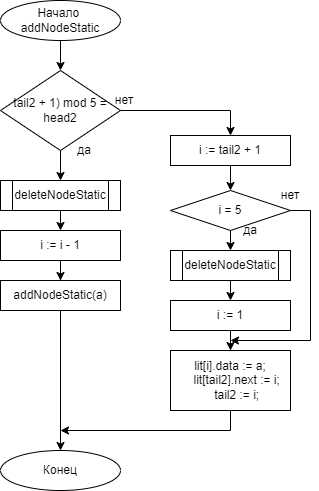


Рисунок 4 – Процедура addNodeStatic

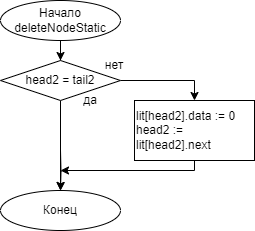


Рисунок 5 – Процедура deleteNodeStatic

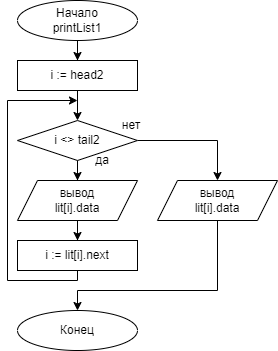


Рисунок 6 – Процедура printList1

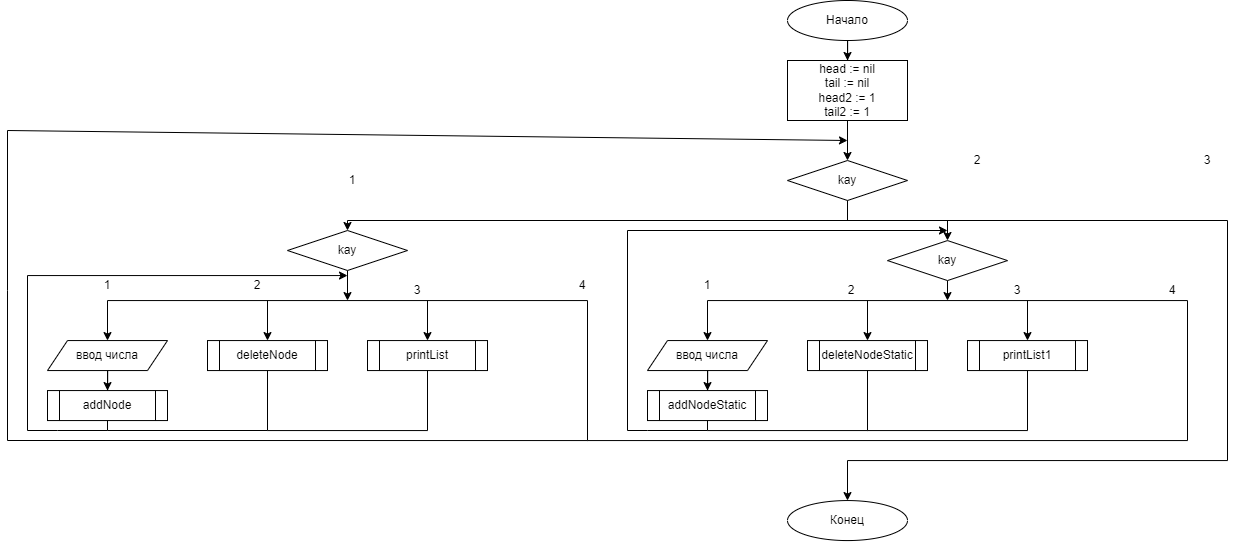


Рисунок 7 – Программа кольцевого односвязного списка

1. **Результаты выполнения программы**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 8 – Выбор способа работы со структурой данных

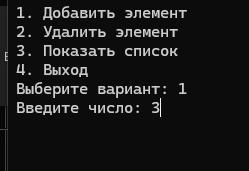


Рисунок 9 – Добавление элемента в список

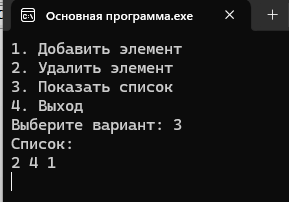


Рисунок 10 – Вывод списка

1. **Код программы**

**Основная программа:**

**program** main;

**uses** crt, dynamic\_list, static\_list;

**var**

choice: integer;

value: integer;

**begin**

head := nil;

tail := nil;

head2 := 1;

tail2 := 1;

**repeat**

writeln('1. Динамический массив');

writeln('2. Статический массив');

writeln('3. Выход');

write('Выберите вариант: ');

readln(choice);

clrscr();

**case** choice **of**

1:

**begin**

**repeat**

writeln('1. Добавить элемент');

writeln('2. Удалить элемент');

writeln('3. Показать список');

writeln('4. Выход');

write('Выберите вариант: ');

readln(choice);

**case** choice **of**

1:

**begin**

write('Введите число: ');

readln(value);

addNode(value);

**end**;

2: deleteNode;

3: printList;

**else**

writeln('Неправильно выбран вариант');

**end**;

clrscr();

**until** choice = 4;

**end**;

2:

**begin**

**repeat**

writeln('1. Добавить элемент');

writeln('2. Удалить элемент');

writeln('3. Показать список');

writeln('4. Выход');

write('Выберите вариант: ');

readln(choice);

**case** choice **of**

1:

**begin**

write('Введите число: ');

readln(value);

addNodeStatic(value);

**end**;

2: deleteNodeStatic;

3: printList1;

**else**

writeln('Неправильно выбран вариант');

**end**;

clrscr();

**until** choice = 4;

**end**;

3: ;

**else**

writeln('Неправильно выбран вариант');

**end**;

**until** choice = 3;

**end**.

**Модуль dynamic\_list:**

**unit** dynamic\_list;

**interface**

**type**

PNode = ^Node;

Node = **record**

data: integer;

next: PNode;

**end**;

**var**

head, tail: PNode;

**procedure** addNode(a: integer);

**procedure** deleteNode;

**procedure** printList;

**implementation**

**procedure** addNode(a: integer);

**var**

newNode: PNode;

**begin**

**new**(newNode);

newNode^.data := a;

newNode^.next := nil;

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

**begin**

head := newNode;

tail := newNode;

**end**

**else**

**begin**

tail^.next := newNode;

tail := newNode;

**end**;

**end**;

**procedure** deleteNode;

**var**

temp: PNode;

**begin**

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

writeln('Список пуст')

**else**

**begin**

temp := head;

head := head^.next;

dispose(temp);

**end**;

**end**;

**procedure** printList;

**var**

current: PNode;

**begin**

writeln('Список:');

current := head;

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

**begin**

write(current^.data, ' ');

current := current^.next;

**end**;

writeln;

readln;

**end**;

**end**.

**Модуль static\_list:**

**unit** static\_list;

**interface**

**type**

List = **array**[1..10] **of record**

next: integer;

data: integer;

**end**;

**var**

head2, tail2: integer;

lit: List;

**procedure** addNodeStatic(a: integer);

**procedure** deleteNodeStatic;

**procedure** printList1;

**implementation**

**procedure** addNodeStatic(a: integer);

**var**

i: integer;

**begin**

**if** (tail2 + 1) **mod** 5 = head2 **then**

**begin**

deleteNodeStatic;

i := i - 1;

addNodeStatic(a);

**end**

**else**

**begin**

i := tail2 + 1;

**if** i = 5 **then**

**begin**

deleteNodeStatic;

i := 1;

**end**;

lit[i].data := a;

lit[tail2].next := i;

tail2 := i;

**end**;

**end**;

**procedure** deleteNodeStatic;

**begin**

**if** head2 = tail2 **then**

writeln('Список пуст')

**else**

**begin**

lit[head2].data := 0;

head2 := lit[head2].next;

**end**;

**end**;

**procedure** printList1;

**var**

i: integer;

**begin**

writeln('Кольцевой список:');

i := head2;

**while** i <> tail2 **do**

**begin**

write(lit[i].data, ' ');

i := lit[i].next;

**end**;

write(lit[i].data);

writeln;

readln;

**end**;

**end**.

**Вывод:**

В результате проделанной работы, были изучены принципы работы с базовыми структурами данных, получены навыки организации case-меню.

В ходе ДКР была разработана программа для реализации структуры данных – кольцевой односвязный список – с помощью статических и динамических структур. Были также развиты навыки работы с собственными модулями. Программа реализует два способа работы с кольцевым односвязным списком. Пользователь может выбрать какую структуру он хочет использовать для реализации списка и какие действия он хочет над ним провести.

Таким образом, все поставленные цели и задачи были успешно достигнуты и выполнены.