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-204-52-00

Беляев Макар Леонидович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

**Цель работы**

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

**Формулировка задания**

Фотка задания

**Описание алгоритма**

1. Инициализация списка (InitList)

* устанавливает head и tail в nil, указывая на пустой список.

1. Добавление элемента в пустой список (AddToEmptyList)

* создает новый узел с данными и устанавливает его ссылки на себя, так как это первый узел в списке.
* назначает head и tail этому узлу.

1. Добавление элемента в начало списка (AddToBeginning)

* создает новый узел с данными.
* устанавливает ссылки нового узла на предыдущий хвост и следующий новый узел.
* переназначает ссылки предыдущего хвоста и текущего головы на новый узел.
* обновляет голову на новый узел.

1. Добавление элемента в конец списка (AddToEnd)

* создает новый узел с данными.
* устанавливает ссылки нового узла на предыдущий хвост и следующий голову.
* переназначает ссылку предыдущего хвоста на новый узел.
* обновляет хвост на новый узел.

1. Вывод списка на экран (DisplayList)

* если список пуст, выводит сообщение "Список пуст".
* в противном случае выполняет цикл, выводит данные каждого узла и переходит к следующему до тех пор, пока не достигнет головы (указатель current вернется к head).

1. Пользовательский интерфейс

* выводит меню с вариантами добавления элементов в начало, конец, отображения списка и выхода.
* запрашивает выбор от пользователя.

**Схемы алгоритма**

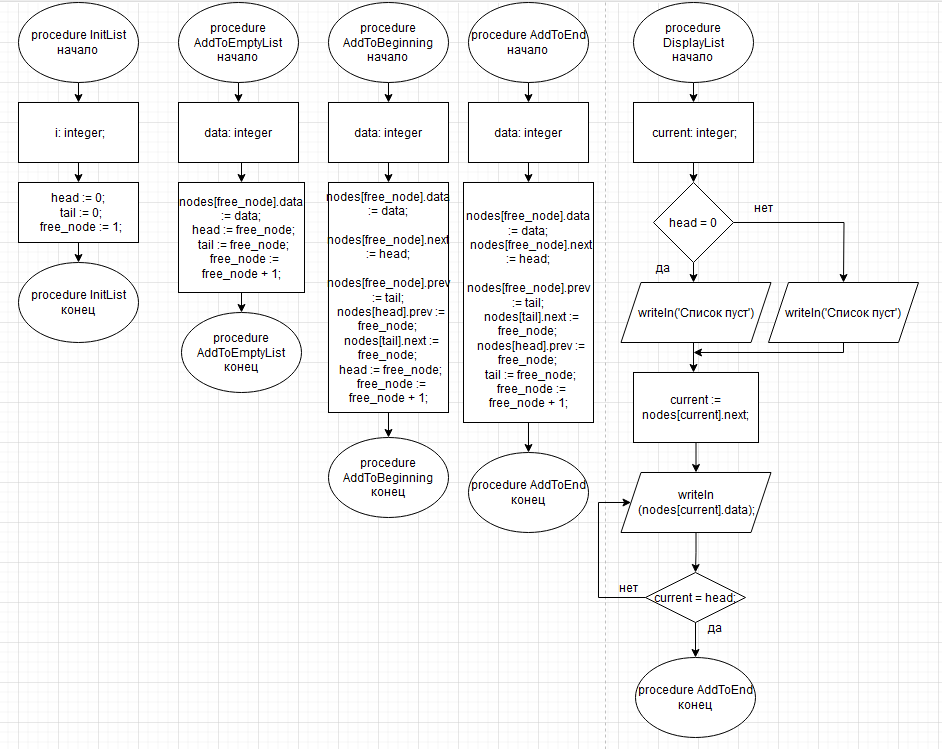
****

Рисунок 1 – procedures

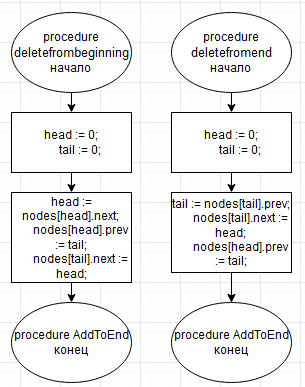


Рисунок 2 – procedures

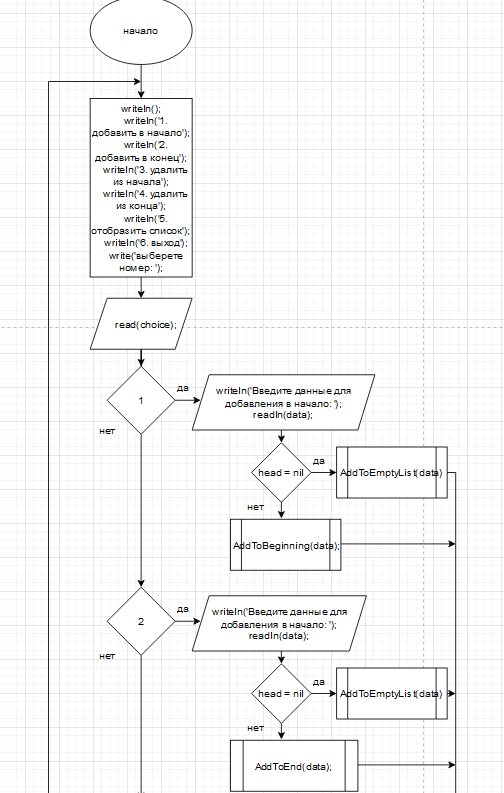
****

Рисунок 3 – схема алгоритма

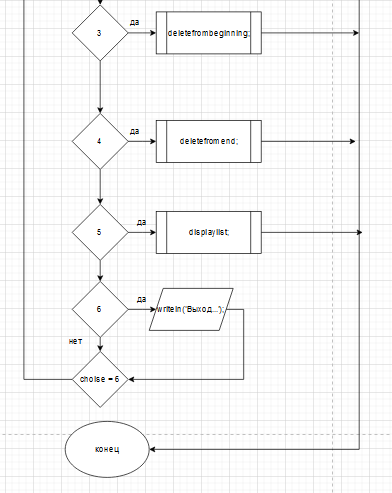


Рисунок 4 – схема алгоритма продолжение

**Код программы**

**const**

max\_size = 10;

**type**

node = **record**

data: integer;

prev, next: integer;

**end**;

**var**

head: integer;

tail: integer;

nodes: **array**[1..max\_size] **of** node;

free\_node: integer;

**procedure** initlist;

**var**

i: integer;

**begin**

head := 0;

tail := 0;

free\_node := 1;

**for** i := 1 **to** max\_size **do**

**begin**

nodes[i].prev := 0;

nodes[i].next := 0;

**end**;

**end**;

**procedure** addtoemptylist(data: integer);

**begin**

nodes[free\_node].data := data;

head := free\_node;

tail := free\_node;

free\_node := free\_node + 1;

**end**;

**procedure** addtobeginning(data: integer);

**begin**

nodes[free\_node].data := data;

nodes[free\_node].next := head;

nodes[free\_node].prev := tail;

nodes[head].prev := free\_node;

nodes[tail].next := free\_node;

head := free\_node;

free\_node := free\_node + 1;

**end**;

**procedure** addtoend(data: integer);

**begin**

nodes[free\_node].data := data;

nodes[free\_node].next := head;

nodes[free\_node].prev := tail;

nodes[tail].next := free\_node;

nodes[head].prev := free\_node;

tail := free\_node;

free\_node := free\_node + 1;

**end**;

**procedure** displaylist;

**var**

current: integer;

**begin**

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

writeln('список пуст')

**else**

**begin**

current := head;

**repeat**

writeln(nodes[current].data);

current := nodes[current].next;

**until** current = head;

**end**;

**end**;

**procedure** deletefrombeginning;

**begin**

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

writeln('Список пуст')

**else**

**begin**

**if** head = tail **then**

**begin**

head := 0;

tail := 0;

**end**

**else**

**begin**

head := nodes[head].next;

nodes[head].prev := tail;

nodes[tail].next := head;

**end**;

**end**;

**end**;

**procedure** deletefromend;

**begin**

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

writeln('Список пуст')

**else**

**begin**

**if** head = tail **then**

**begin**

head := 0;

tail := 0;

**end**

**else**

**begin**

tail := nodes[tail].prev;

nodes[tail].next := head;

nodes[head].prev := tail;

**end**;

**end**;

**end**;

**var**

choice, data: integer;

**begin**

initlist;

**repeat**

writeln();

writeln('1. добавить в начало');

writeln('2. добавить в конец');

writeln('3. удалить из начала');

writeln('4. удалить из конца');

writeln('5. отобразить список');

writeln('6. выход');

write('выберете номер: ');

read(choice);

**case** choice **of**

1: **begin**

writeln('введите данные для добавления в начало: ');

readln(data);

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

addtoemptylist(data)

**else**

addtobeginning(data);

**end**;

2: **begin**

writeln('введите данные для добавления в конец: ');

readln(data);

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

addtoemptylist(data)

**else**

addtoend(data);

**end**;

3: **begin**

deletefrombeginning;

**end**;

4: **begin**

deletefromend;

**end**;

5: **begin**

displaylist;

**end**;

6: writeln('выход...');

**end**;

**until** choice = 6;

**end**.

**Результат выполнения программы**

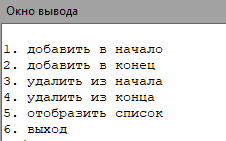
****

Рисунок 3 – Список команд

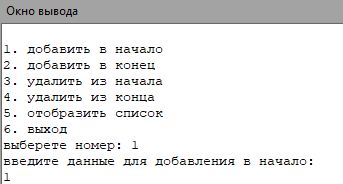
****

Рисунок 4 – 1. Добавить в начало

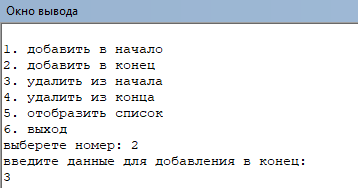
****

Рисунок 5 – 2. Добавить в конец

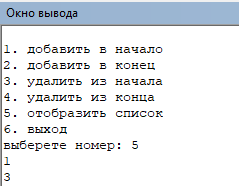
****

Рисунок 6 – 5. Отобразить список

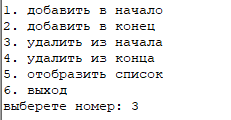


Рисунок 7 – 3. Отобразить список

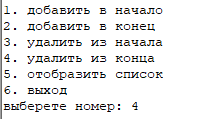


Рисунок 8 – 4. Отобразить список

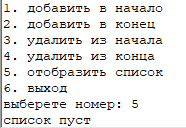


Рисунок 9 – 5. Отобразить список

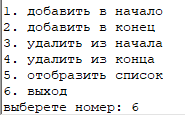


Рисунок 10 – 6. Выход

**Вывод**

В ходе выполнения домашней лабораторной работы мы закрепили знания полученные при выполнение лабораторной работы №15-16. Научились реализовывать рекурсивные алгоритмы, изучили фракталы, их свойства, освоили использование кнопок для изменения параметров фрактала. Таким образом, полученные знания могут быть полезны в дальнейшем.

Так же для написания схемы алгоритма был использован интернет -ресурс «draw.io». Его использование упрощает написание схем алгоритмов, уменьшает вероятность ошибок, а также в случае пропуска какого-либо цикла его можно без каких-либо проблем вставить в любое место схемы. Опыт написания схем-алгоритмов будет полезен в будущем для решения новых задач а так же понимание программ, написанных другими пользователями