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**

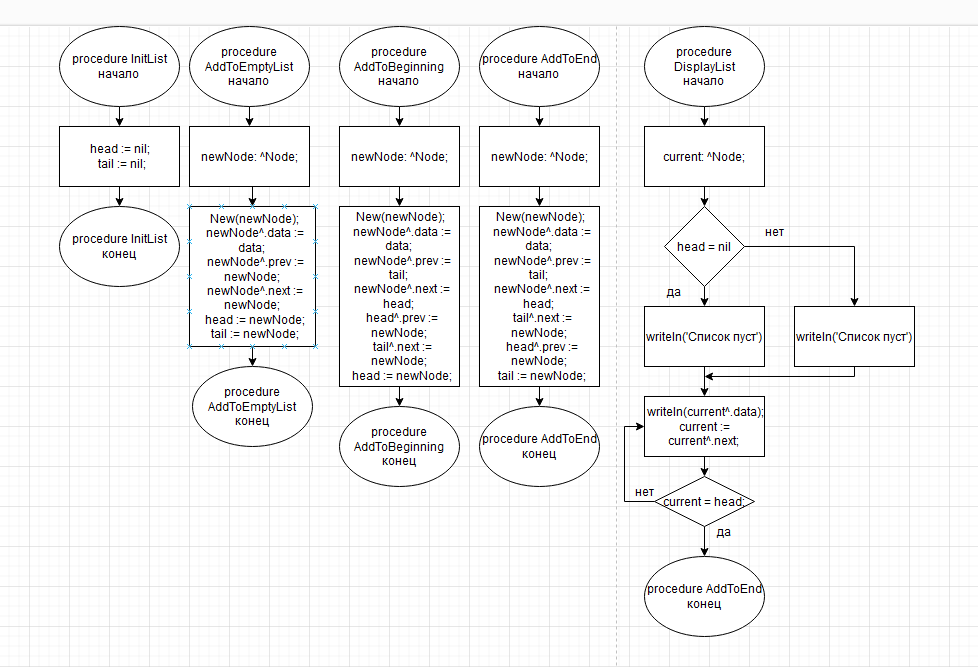
****

Рисунок 1 – procedure

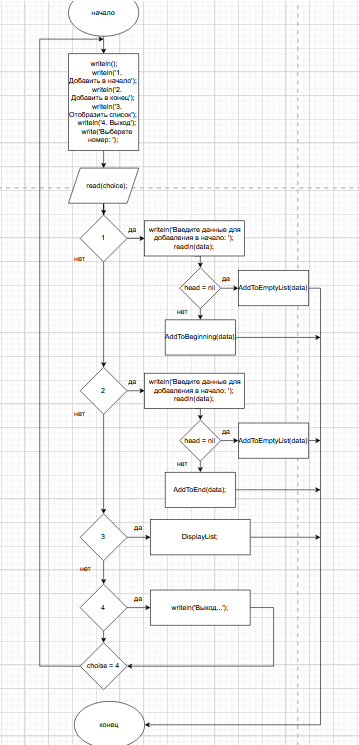
****

Рисунок 2 – схема алгоритма

**Код программы**

**const**

MAX\_SIZE = 10;

**type**

Node = **record**

data: Integer;

prev, next: ^Node;

**end**;

**var**

head: ^Node;

tail: ^Node;

**procedure** InitList;

**begin**

head := nil;

tail := nil;

**end**;

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

**var**

newNode: ^Node;

**begin**

**New**(newNode);

newNode^.data := data;

newNode^.prev := newNode;

newNode^.next := newNode;

head := newNode;

tail := newNode;

**end**;

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

**var**

newNode: ^Node;

**begin**

**New**(newNode);

newNode^.data := data;

newNode^.prev := tail;

newNode^.next := head;

head^.prev := newNode;

tail^.next := newNode;

head := newNode;

**end**;

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

**var**

newNode: ^Node;

**begin**

**New**(newNode);

newNode^.data := data;

newNode^.prev := tail;

newNode^.next := head;

tail^.next := newNode;

head^.prev := newNode;

tail := newNode;

**end**;

**procedure** DisplayList;

**var**

current: ^Node;

**begin**

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

writeln('Cписок пуст')

**else**

**begin**

current := head;

**repeat**

writeln(current^.data);

current := current^.next;

**until** current = head;

**end**;

**end**;

**var**

choice, data: Integer;

**begin**

InitList;

**repeat**

writeln();

writeln('1. Добавить в начало');

writeln('2. Добавить в конец');

writeln('3. Отобразить список');

writeln('4. Выход');

write('Выберете номер: ');

read(choice);

**case** choice **of**

1: **begin**

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

readln(data);

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

AddToEmptyList(data)

**else**

AddToBeginning(data);

**end**;

2: **begin**

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

readln(data);

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

AddToEmptyList(data)

**else**

AddToEnd(data);

**end**;

3: **begin**

DisplayList;

**end**;

4: writeln('Выход...');

**end**;

**until** choice = 4;

**end**.

**Результат выполнения программы**

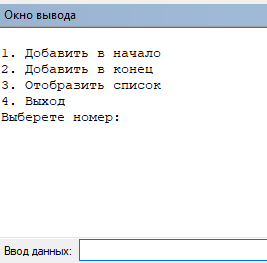
****

Рисунок 3 – Список команд

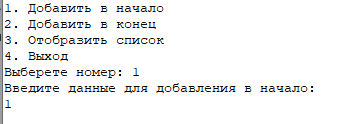
****

Рисунок 4 – 1. Добавить в начало

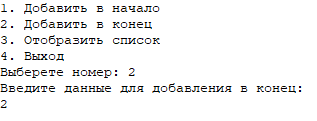
****

Рисунок 5 – 2. Добавить в конец

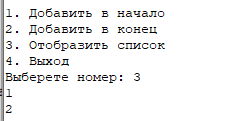
****

Рисунок 6 – 3. Отобразить список

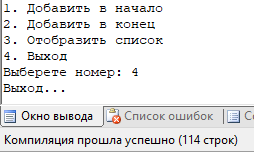


Рисунок 7 – 4. Выход

**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