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 ИНФОРМАЦИОННЫХ СИСТЕМ»**

Выполнила: студентка учебной группы

ИСПк-202-52-00

Широнина Анна Андреевна

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2025

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

1. **Формулировка заданий. Вариант 19**

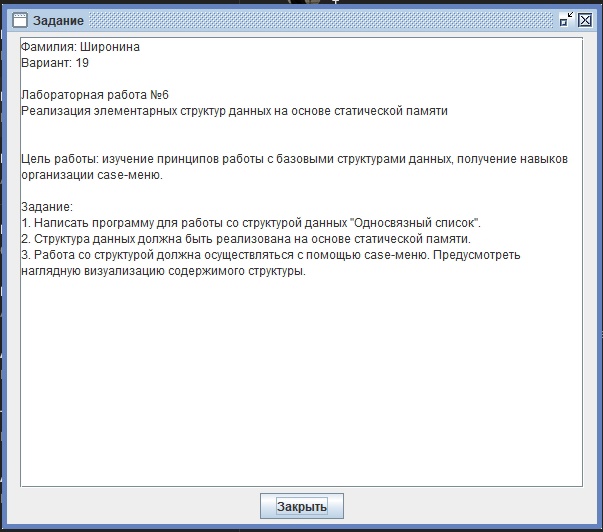
****

Рисунок 1 – Постановка задачки

1. **Описание алгоритма**

В основной программе мы вызываем процедуру Menu. В ней мы инициализируем список. Далее отображается меню, благодаря которому пользователь может добавить элемент в список, отобразить содержимое структуры, а также удалить элемент из списка. Если пользователь нажимает цифру 1, то в действие приходит процедура Append, которая добавляет новый элемент в конец списка. Если пользователь нажимает цифру 2, то в действие приходит процедура Visual, которая отображает содержимое списка. Если пользователь нажимает на цифру 3, то в действие приходит процедура Delete, которая удаляет элемент списка по заданному ключу. А цифра 4 соответствует действию «Выход из программы». Таким образом, цикл выполняется до того, пока пользователь не введет 4.

1. **Схема алгоритма**

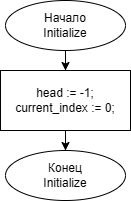
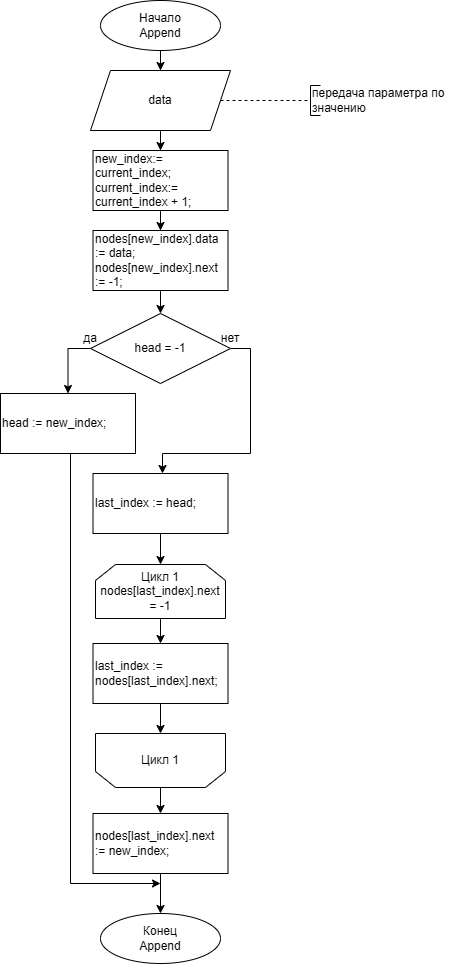
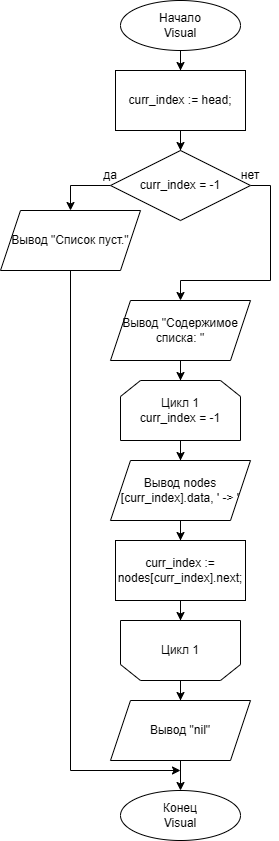
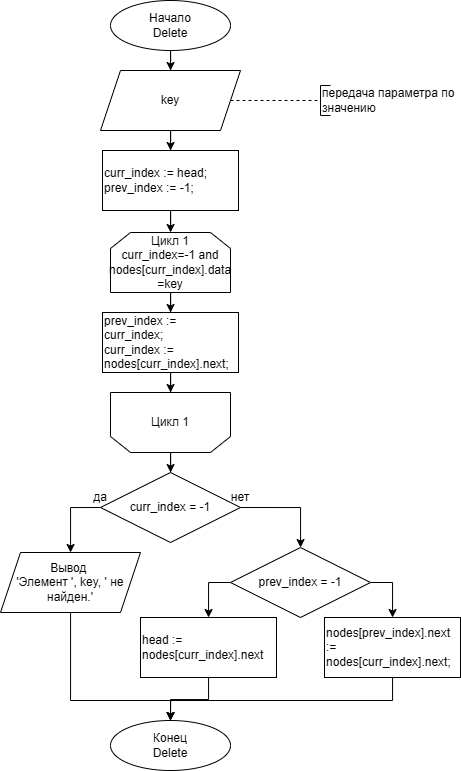
****

Рисунок 2 – Схема алгоритма

****

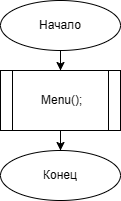
****

Рисунок 3 – Продолжение рисунка 2

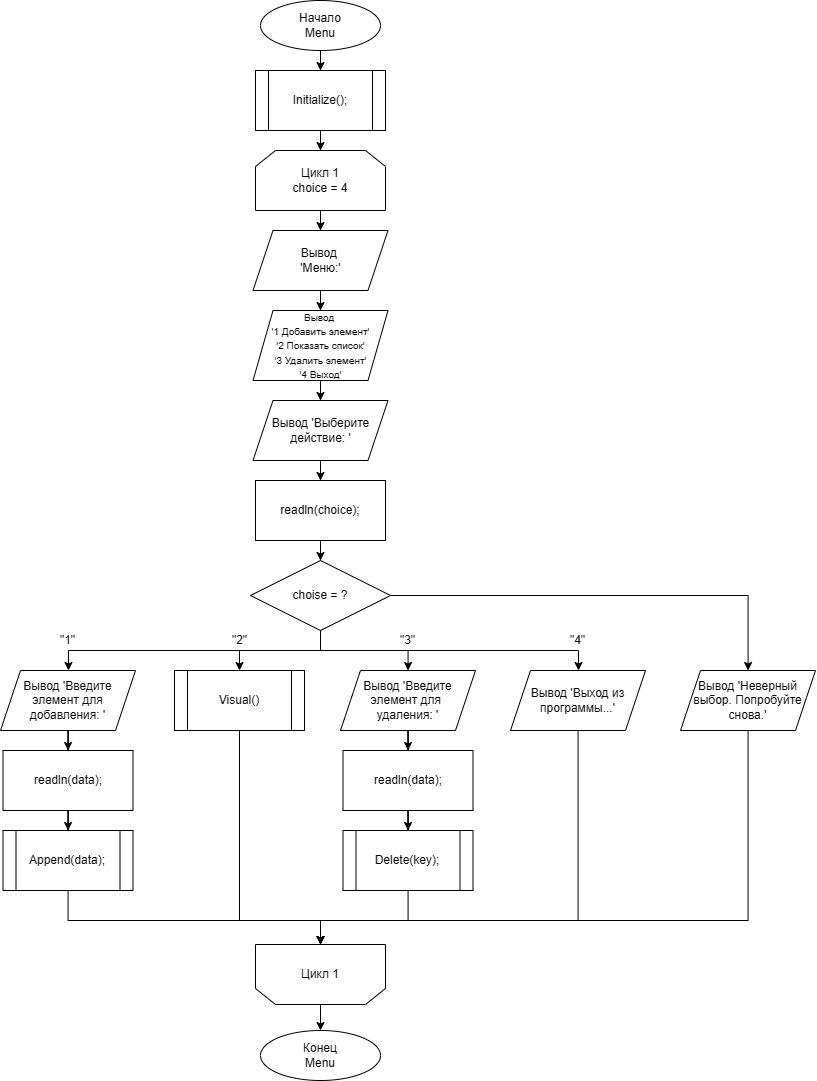
****

Рисунок 4 – Продолжение рисунка 3

1. **Код программы**

**const**

max\_nodes = 100;

**type**

Node = **record**

data: string;

next: integer;

**end**;

**var**

nodes: **array**[0..max\_nodes-1] **of** Node;

head: integer; // Индекс головы списка

current\_index: integer; // Текущий индекс для добавления новых элементов

**procedure** Initialize();

**begin**

head := -1; // Пустой список

current\_index := 0; // Начинаем с первого элемента массива

**end**;

**procedure** Append(data: string);

**var**

new\_index, last\_index: integer;

**begin**

new\_index := current\_index;

current\_index := current\_index + 1;

nodes[new\_index].data := data;

nodes[new\_index].next := -1;

**if** head = -1 **then**

**begin**

head := new\_index;

**exit**;

**end**;

// Находим последний узел

last\_index := head;

**while** nodes[last\_index].next <> -1 **do**

last\_index := nodes[last\_index].next;

nodes[last\_index].next := new\_index;

**end**;

**procedure** Visual();

**var**

curr\_index: integer;

**begin**

curr\_index := head;

**if** curr\_index = -1 **then**

**begin**

writeln('Список пуст.');

**exit**;

**end**;

write('Содержимое списка: ');

**while** curr\_index <> -1 **do**

**begin**

write(nodes[curr\_index].data, ' -> ');

curr\_index := nodes[curr\_index].next;

**end**;

writeln('nil');

**end**;

**procedure** Delete(key: string);

**var**

curr\_index, prev\_index: integer;

**begin**

curr\_index := head;

prev\_index := -1;

// Поиск узла с указанным ключом

**while** (curr\_index <> -1) **and** (nodes[curr\_index].data <> key) **do**

**begin**

prev\_index := curr\_index;

curr\_index := nodes[curr\_index].next;

**end**;

**if** curr\_index = -1 **then**

**begin**

writeln('Элемент ', key, ' не найден.');

**exit**;

**end**;

**if** prev\_index = -1 **then**

head := nodes[curr\_index].next // Удаляем голову

**else**

nodes[prev\_index].next := nodes[curr\_index].next;

**end**;

**procedure** Menu();

**var**

choice: integer;

data, key: string;

**begin**

Initialize();

**repeat**

writeln('Меню:');

writeln('1 Добавить элемент');

writeln('2 Показать список');

writeln('3 Удалить элемент');

writeln('4 Выход');

write('Выберите действие: ');

readln(choice);

**case** choice **of**

1:

**begin**

write('Введите элемент для добавления: ');

readln(data);

Append(data);

**end**;

2: Visual();

3:

**begin**

write('Введите элемент для удаления: ');

readln(key);

Delete(key);

**end**;

4: writeln('Выход из программы...');

**else** writeln('Неверный выбор. Попробуйте снова.');

**end**;

**until** choice = 4;

**end**;

**begin**

Menu();

**end**.

1. **Результат выполнения программы**

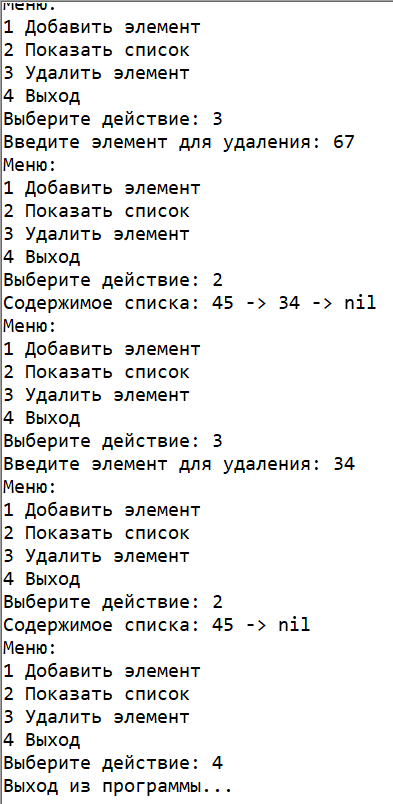
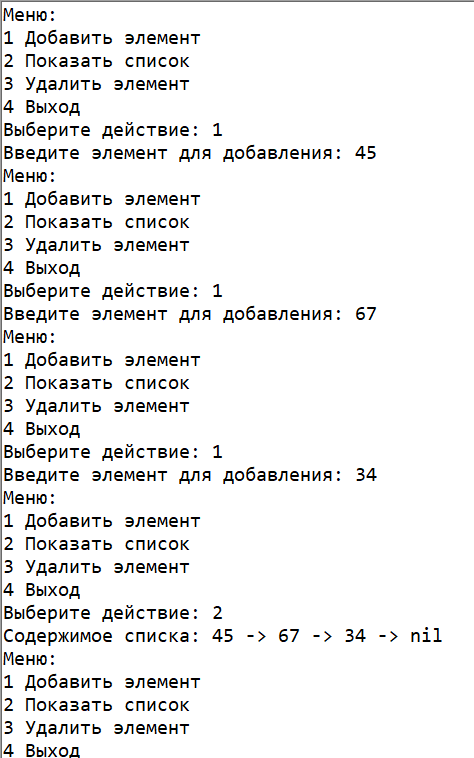
****

Рисунок 5 – Результат выполнения программы

1. **Вывод**

В ходе выполнения лабораторной работы были изучены принципы работы с базовыми структурами данных, а именно с односвязным списком, реализованным на основе статической памяти. Была разработана программа, позволяющая эффективно управлять односвязным списком, используя статическую память. Работа программы организована через интуитивно понятное case-меню, которое позволяет пользователю легко взаимодействовать с программой и выполнять различные операции над списком. Особое внимание было уделено визуализации содержимого списка, что делает процесс работы с ним более наглядным и удобным.

Для работы над схемами алгоритма необходимо четко сформулировать проблему, которую необходимо решить, собрать входные данные, необходимые для алгоритма, описать последовательность шагов, которые необходимо выполнить для решения задачи, использовать стандартные символы для построения схемы алгоритма (овалы – начало и конец, прямоугольники – действия или процессы, ромбы – условия или проверки и т.д.), проверить алгоритм на разных входных данных для выявления возможных ошибок, проанализировать алгоритм и внести изменения для повышения его эффективности, записать и оформить полученные результаты. Работа над схемами алгоритма помогает визуализировать процесс и обеспечить более легкое понимание и оптимизацию решения задачи.