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**

Выполнил: студент учебной группы

ИСПк-205-52-00

Праздников Александр Андреевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

**1. Цель работы**

Изучить принципы работы с базовыми структурами данных, получение навыков организации case-меню.

**2. Скриншот задания (с вариантом)**

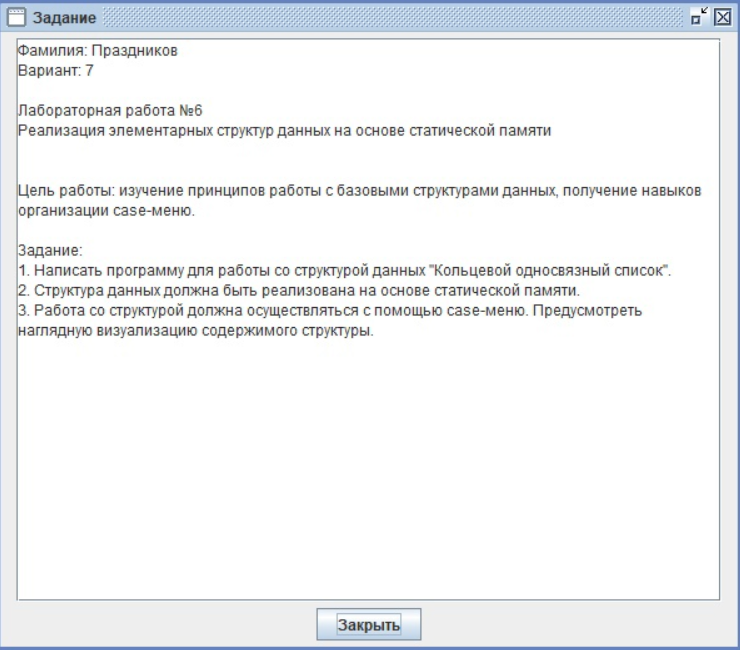


Рис 1 – Скриншот задания

**3. Описание алгоритма**

В программе есть 3 функции. Первая функция deleteNode служит для удаления узла в списке. Следующая функция addNode служит для добавления узла. Функция AddNode работает по следующему алгоритму: если список переполнен то вызывается функция deleteNode, переходит на один элемент назад в списке и вызывает функцию addNode. Иначе переходит к концу списка и добавляет то число, которое было введено с клавиатуры. И последняя функция это PrintList1 она служит для вывода всего списка. Так же в программе реализованно case-меню.

**4. Код программы**

**uses** crt;

**const**

MAX\_SIZE = 5;

**type**

List = **array**[1..MAX\_SIZE] **of record**

next: integer;

data: integer;

**end**;

**var**

lit: List;

head, tail: integer;

**procedure** deleteNode;

**begin**

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

writeln('List is empty')

**else**

**begin**

lit[head].data := 0;

head := lit[head].next;

**end**;

**end**;

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

**var**

i: integer;

**begin**

**if** (tail + 1) **mod** MAX\_SIZE = head **then begin**

deleteNode;

i:=i-1;

addNode(a);

**end**

**else**

**begin**

i := tail + 1;

**if** i = MAX\_SIZE **then**

**begin**

deleteNode;

i:=1;

**end**;

lit[i].data := a;

lit[tail].next := i;

tail := i;

**end**;

**end**;

**procedure** printList;

**var**

i: integer;

**begin**

writeln('List:');

i := head;

**while** i <> tail **do**

**begin**

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

i := lit[i].next;

**end**;

write(lit[i].data);

writeln;

**end**;

**procedure** printList1;

**var**

i: integer;

**begin**

writeln('List:');

i := head;

**while** i <> tail **do**

**begin**

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

i := lit[i].next;

**end**;

write(lit[i].data);

writeln;

**end**;

**var**

choice: integer;

a: integer;

**begin**

head := 1;

tail := 1;

**repeat**

writeln('1. Добавить узел');

writeln('2. Удалить узел');

writeln('3. Вывести');

writeln('4. Выйти');

write('Выбери на чьей стороне ты: ');

readln(choice);

**case** choice **of**

1: **begin**

write('Введите значение: ');

readln(a);

addNode(a);

{ClrScr;}

**end**;

2: **begin** deleteNode; {ClrScr;} **end**;

3: printList;

4: **exit**;

**else** writeln('Не тот выбор');

**end**;

**until** choice = 4;

**end**.

**5. Результат выполнения программы**

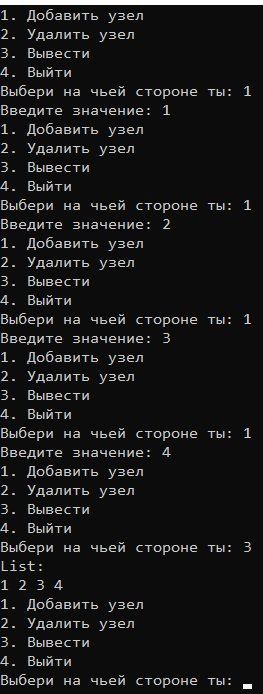
****

Рисунок 2 – Результаты выполнения программы

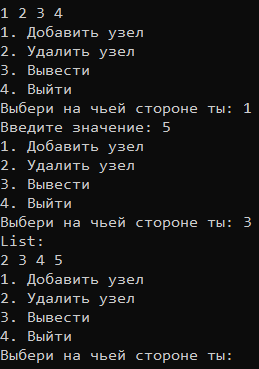


Рисунок 3 – Результаты выполнения программы

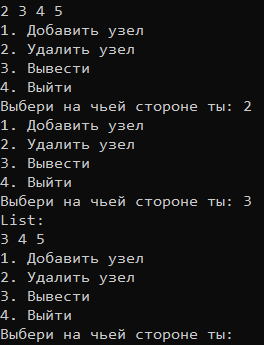
****

Рисунок 4 – Результаты выполнения программы

**6. Вывод**

В данной домашней контрольной работе была изучена работа с базовыми структурами данных и закрепление навыков организации case-меню.

Программа начинается с того что в списке обнуляется все для того чтоб не возникли ошибки в коде. После чего вызывается цикл repeat в котором реализован case-меню. В case-меню есть 4 варианта: 1 вариант для создания узла в списке, 2 вариант для удаления узла в списке, 3 вариант для просмотра всего списка, 4 вариант для выхода из цикла и в общем из программы. Если пользователь выберет неправильный вариант, то на экран выведется надпись: «Не правильно выбран вариант».

В процессе выполнения домашней контрольной работы возникли трудности:

* Реализация программы с помощью статической памяти в силу не знания
* Реализация кольцевого списка так как изначально при превышении массива список обнулялся и начинался с начала