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-101-51-00

Костин Борис Сергеевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

**Цель работы**

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

**Задание**

Вариант 15

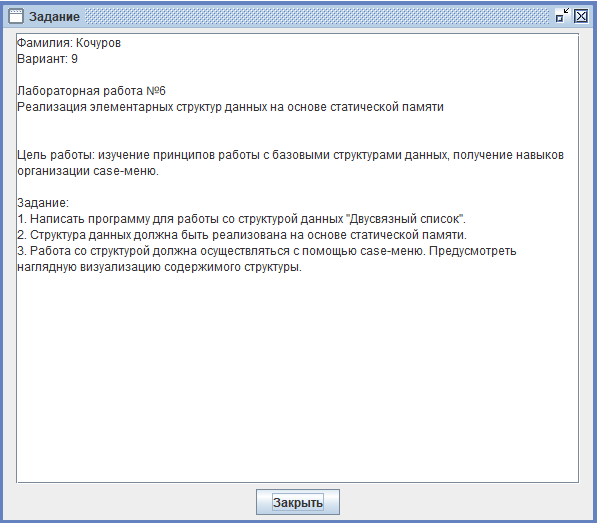
****

Рисунок 1 – Задание

**Описание**

Формируется двусвязный список посредством использования массива, список имеет ограниченное количество элементов (5-10 случайных значений). Программа реализует работу со списком: возможность добавить элемент в начало и конец списка, добавить элемент перед или после существующего элемента, удаление элементов.

**Код программы**

**uses** CRT;

**const**

max = 100;

**type**

ElementType = Integer;

Node = **record**

data: ElementType;

prev: Integer;

next: Integer;

**end**;

**var**

list: **array** [1..max] **of** Node;

head: Integer = 0;

tail: Integer = 0;

free: Integer = 1;

**procedure** initializeList;

**var**

i: Integer;

**begin**

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

**begin**

list[i].prev := i - 1;

list[i].next := i + 1;

**end**;

list[max].next := 0;

**end**;

//добавить пространство

**function** getFreeNode: Integer;

**begin**

**if** free = 0 **then**

getFreeNode := 0

**else**

**begin**

getFreeNode := free;

free := list[free].next;

**end**;

**end**;

**procedure** releaseNode(index: Integer);

**begin**

list[index].prev := 0;

list[index].next := free;

free := index;

**end**;

//добавить в начало

**procedure** addToBeginning(value: ElementType);

**var**

index: Integer;

**begin**

index := getFreeNode; //добавить в новую ячейку

**if** index <> 0 **then**

**begin**

list[index].data := value; // присвоение значения

list[index].prev := 0; //предыдущий элемент

list[index].next := head; // ссылка на следующий элемент = 1 элементу всего списка

**if** head <> 0 **then**

list[head].prev := index;

head := index;

**if** tail = 0 **then**

tail := head;

**end**;

**end**;

//добавить в конец

**procedure** addToEnd(value: ElementType);

**var**

index: Integer;

**begin**

index := getFreeNode;//добавить в новую ячейку

**if** index <> 0 **then**

**begin**

list[index].data := value;

list[index].prev := tail;

list[index].next := 0;

**if** tail <> 0 **then**

list[tail].next := index; // суём в ячейку индекс

tail := index;

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

head := tail;

**end**;

**end**;

//вставить перед

**procedure** insertBefore(existingValue: ElementType; newValue: ElementType);

**var**

index, currentNode: Integer;

**begin**

currentNode := head;

**while** (currentNode <> 0) **and** (list[currentNode].data <> existingValue) **do**

currentNode := list[currentNode].next;

**if** currentNode <> 0 **then**

**begin**

index := getFreeNode;

**if** index <> 0 **then**

**begin**

list[index].data := newValue;

list[index].prev := list[currentNode].prev;

list[index].next := currentNode;

**if** list[currentNode].prev <> 0 **then**

list[list[currentNode].prev].next := index

**else**

head := index;

list[currentNode].prev := index;

**end**;

**end**;

**end**;

//вставить после

**procedure** insertAfter(existingValue: ElementType; newValue: ElementType);

**var**

index, currentNode: Integer;

**begin**

currentNode := head;

**while** (currentNode <> 0) **and** (list[currentNode].data <> existingValue) **do**

currentNode := list[currentNode].next;

**if** currentNode <> 0 **then**

**begin**

index := getFreeNode;

**if** index <> 0 **then**

**begin**

list[index].data := newValue;

list[index].prev := currentNode;

list[index].next := list[currentNode].next;

**if** list[currentNode].next <> 0 **then** list[list[currentNode].next].prev := index

**else** tail := index;

list[currentNode].next := index;

**end**;

**end**;

**end**;

**procedure** delete(value: ElementType);

**var**

currentNode: Integer;

**begin**

currentNode := head;

**while** (currentNode <> 0) **and** (list[currentNode].data <> value) **do**

currentNode := list[currentNode].next; //принимает значение элемента

**if** currentNode <> 0 **then**

**begin**

**if** list[currentNode].prev <> 0 **then** //Если ук-тель на пред. эл-нт не равен 0, то ук-тель на след. эл-нт пред. эл-нта равен ук-телю на след. эл-нт удаляемого эл-нта

list[list[currentNode].prev].next := list[currentNode].next

**else**

head := list[currentNode].next;//елси =0

**if** list[currentNode].next <> 0 **then** //то же самое но в обратную сторону

list[list[currentNode].next].prev := list[currentNode].prev

**else**

tail := list[currentNode].prev;

releaseNode(currentNode);

**end**;

**end**;

//показ списка

**procedure** displayList;

**var**

currentNode: Integer;

**begin**

println('Элементы списка:');

currentNode := head;

**while** currentNode <> 0 **do**

**begin**

print(list[currentNode].data);

currentNode := list[currentNode].next;

**end**;

println();

**end**;

**begin**

initializeList;

**var** r: integer;

**var** g: integer; g := random(5, 10);

**for var** i := 1 **to** g **do**

**begin**

r := random(-15, 45);

addtobeginning(r);

**end**;

displayList;

**var** c: byte;

**repeat**

println('1 - Добавить элемент в начало');

println('2 - Добавить элемент в конец');

println('3 - Вставить после элемента');

println('4 - Вставить перед элементом');

println('5 - Удалить элемент');

println('0 - Выход');

read(c);

**case** c **of**

1:

**begin**

**var** el: integer;

el := readinteger('Введите элемент: ');

addtobeginning(el);

displayList;

**end**;

2:

**begin**

**var** el: integer;

el := readinteger('Введите элемент: ');

addtoend(el);

displayList;

**end**;

3:

**begin**

**var** el, aft: integer;

el := readinteger('Введите элемент: ');

aft := readinteger('Введте элемент, после которого вставить: ');

insertAfter(aft, el);

displayList;

**end**;

4:

**begin**

**var** el, bef: integer;

el := readinteger('Введите элемент: ');

bef := readinteger('Введте элемент, перед которым вставить: ');

insertBefore(bef, el);

displayList;

**end**;

5:

**begin**

**var** el: integer;

el := readinteger('Введите элемент: ');

delete(el);

displayList;

**end**;

**end**;

**until** c = 0;

**end**.

**Результат выполнения программы**

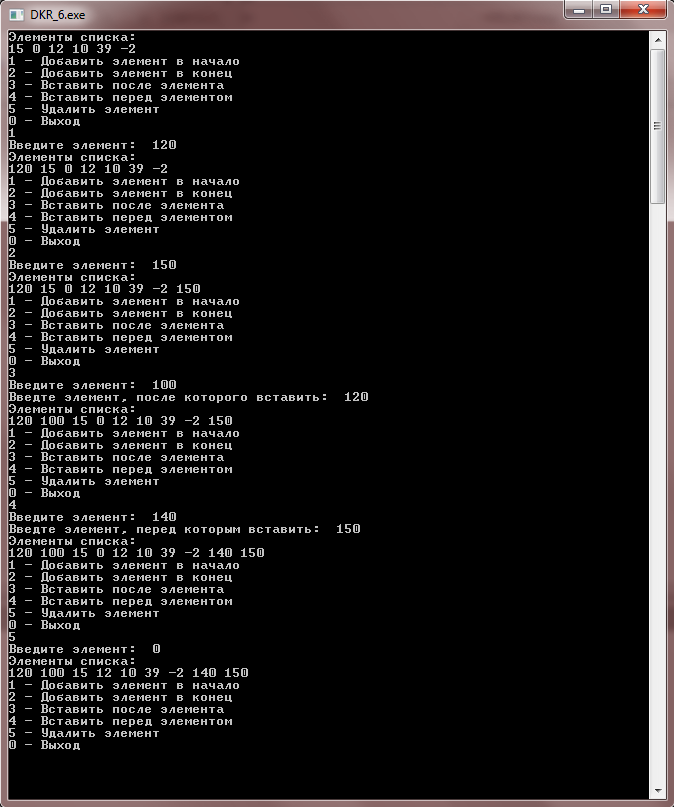


Рисунок 1 – Результат выполнения программы

**Вывод**

Выполняя данную домашнюю контрольную работу, я научился реализовывать структуры данных в двусвязном списке, столкнулся со сложностями, связанными с case-меню и его использованием.