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-204-52-00

Малахов Егор Романович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

1.Цель работы

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

2. Вариант 1

3. Описание алгоритма

В данной программе используются такие функции как: dobavitdo (добавляет число до выбранного элемента), dobavitposle (добавляет число после выбранного элемента), udalenie(Удаление записей из списка), poisk (поиск элемента списка) для работы с двухсвязным списком. Так же в ней используется case-menu исходя из условии выполнения задания. Она упрощает работу с программой.

4.Код программы

**uses** CRT;

**const**

MAX\_SIZE = 100;

**type**

ElementType = Integer;

Node = **record**

data: ElementType;

prev: Integer;

next: Integer;

**end**;

**var**

list: **array** [1..MAX\_SIZE] **of** Node;

head: Integer = 0;

tail: Integer = 0;

free: Integer = 1;

**procedure** initializeList;

**var**

i: Integer;

**begin**

**for** i := 1 **to** MAX\_SIZE **do**

**begin**

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

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

**end**;

list[MAX\_SIZE].next := 0;

**end**;

**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**;

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

**var**

index: Integer;

**begin**

index := getFreeNode;

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

**begin**

list[index].data := value;

list[index].prev := 0;

list[index].next := head;

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

list[head].prev := index;

head := index;

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

tail := head;

**end**;

**end**;

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

**var**

index: Integer;

**begin**

index := getFreeNode;

**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;

tail := index;

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

head := tail;

**end**;

**end**;

**procedure** dobavitdo(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**;

**procedure** dobavitPosle(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** udalenie(value: ElementType);

**var**

currentNode: Integer;

**begin**

currentNode := head;

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

currentNode := list[currentNode].next;

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

**begin**

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

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

**else**

head := list[currentNode].next;

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

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

**else**

tail := list[currentNode].prev;

releaseNode(currentNode);

**end**;

**end**;

**function** poisk(value: ElementType): Boolean;

**var**

currentNode: Integer;

**begin**

currentNode := head;

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

currentNode := list[currentNode].next;

poisk := currentNode <> 0;

**end**;

**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.Найти элемент в списке');

println('6.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('Введите элемент, после которого вставить: ');

dobavitPosle(aft, el);

displayList;

**end**;

4:**begin**

**var** el,bef:integer;

el:=readinteger('Введите элемент: ');

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

dobavitdo(bef, el);

displayList;

**end**;

5:**begin**

**var** el:integer;

**var** fl:boolean;

el:=readinteger('Введите элемент: ');

fl:=poisk(el);

**if** fl **then** print('Элемент в списке присутствует')

**else** print('Элемент в списке отсутствует');

println();

**end**;

6:**begin**

**var** el:integer;

el:=readinteger('Введите элемент: ');

udalenie(el);

displayList;

**end**;

**end**;

**until** c=0;

**end**.

5.Результат выполнение программы

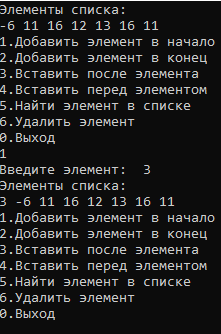


Рисунок 1 – Результат выполнения программы

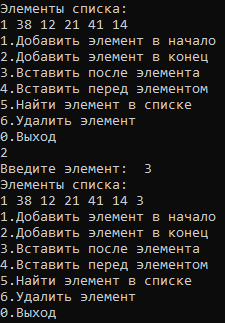


Рисунок 2 – Результат выполнения программы

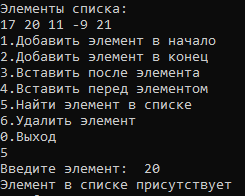


Рисунок 3 – Результат выполнения программы

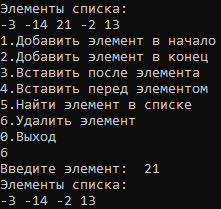


Рисунок 4 – Результат выполнения программы

6. Вывод

В ходе выполнения данной домашней контрольной работы, с помощью доп. Материала, интернета и помощью учителя, мы получили знания о принципах работы с базовыми структурами данных, таких как двухсвязный список и навыки организации case-menu.