Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №8**

**«МДК 05.02 Разработка кода информационных систем»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Кононов Степан Андреевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

Цель работы

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

Формулировка задания

Вариант 12.

1. Написатть программу, реализующую двусвязный список.

2. Добавить реализацию статическим массивом.

3. Добавить реализацию указателями.

4. Взаимодействие через case-menu с наглядной реализацией структуры.

Описание алгоритма

Алгоритм создаёт дву пустые структуры данных с помощью статического массива и с помощью указателей, после пользователь взаимодействует с ними при помощи case-menu.

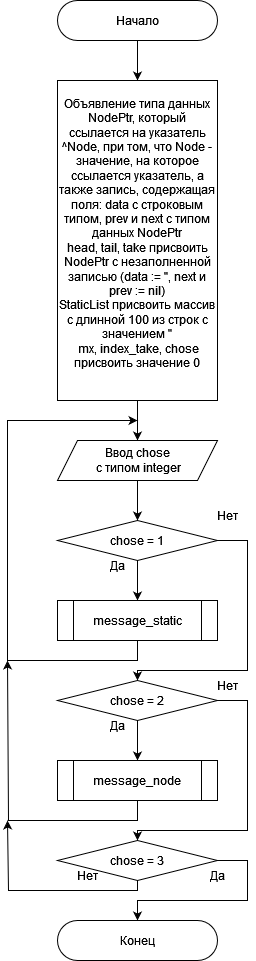


Рисунок 1 – Схема главного алгоритма

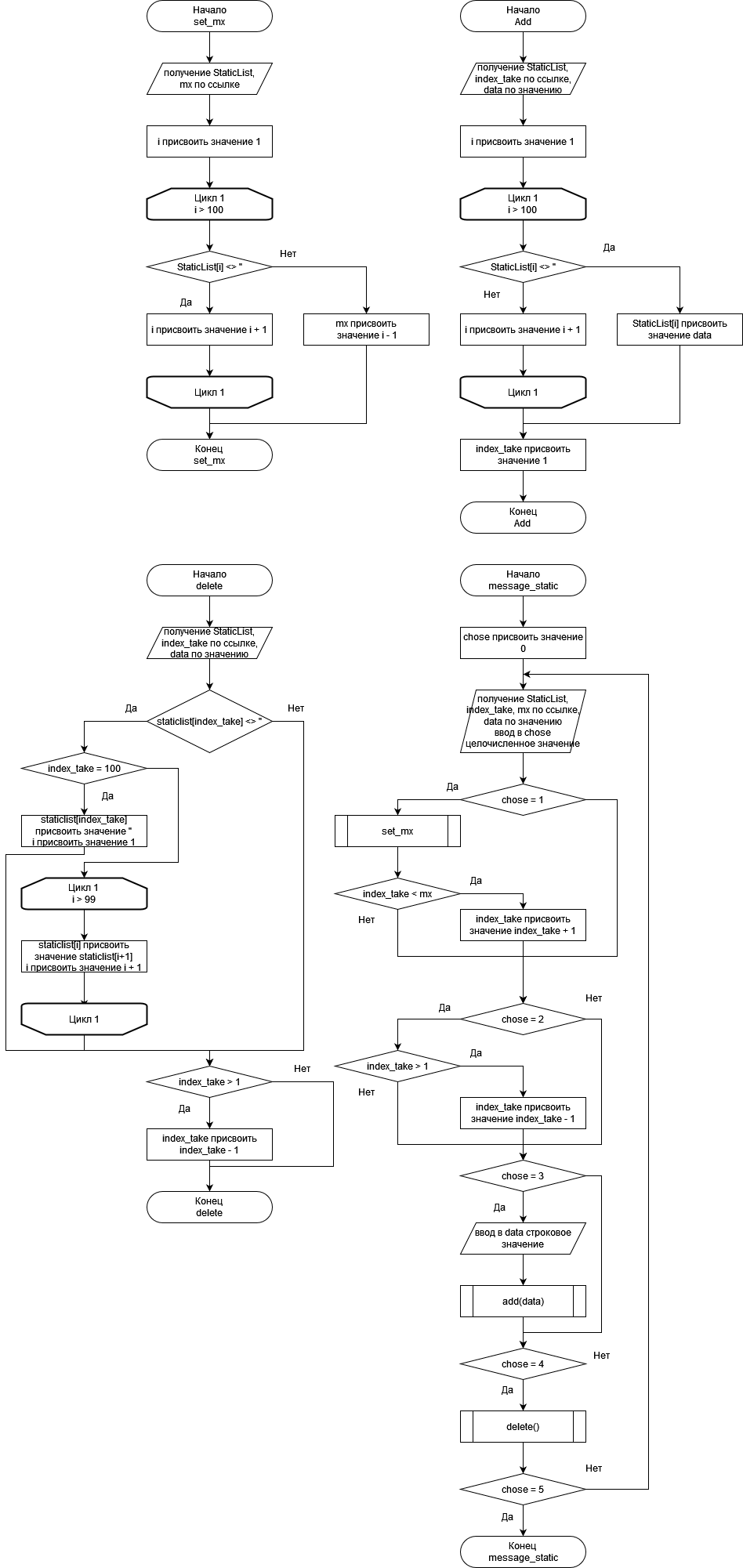


Рисунок 2 – Схема алгоритма процедур статической памяти

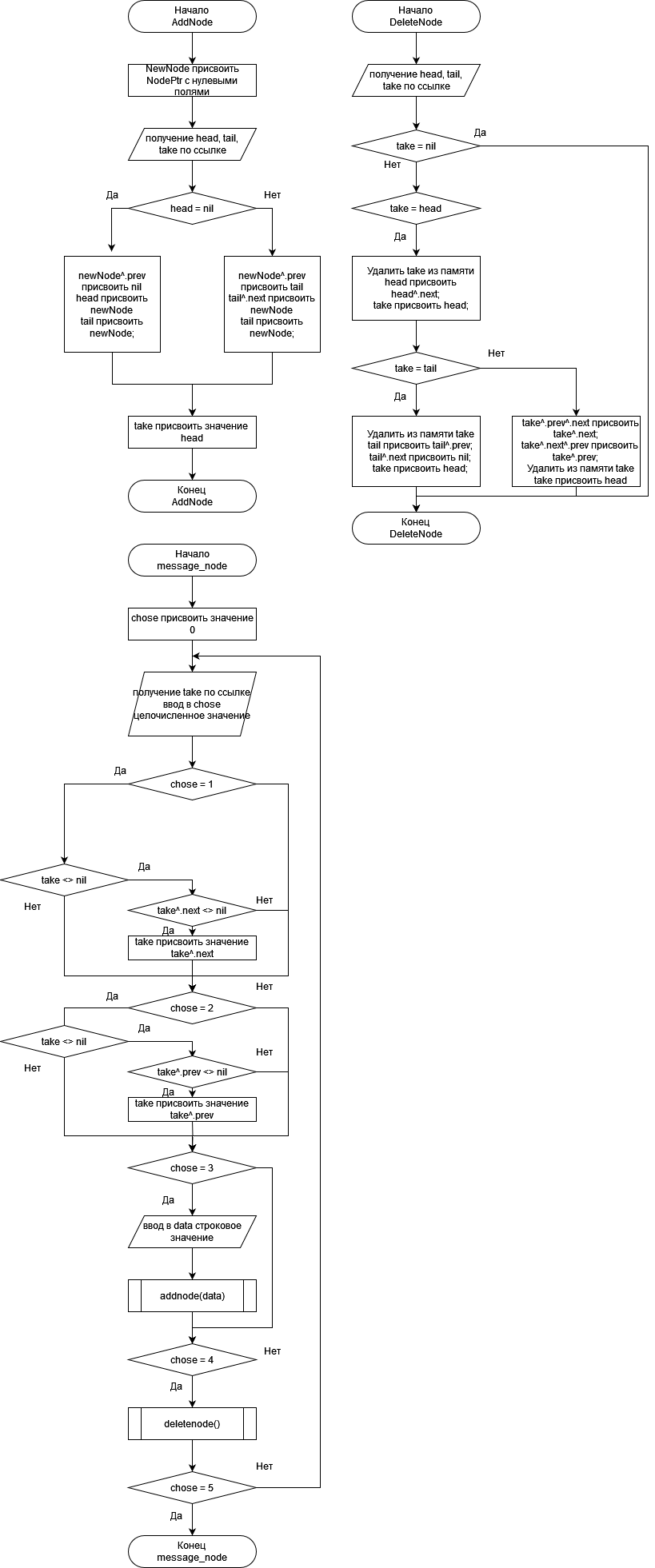


Рисунок 3 – Схема алгоритма процедур с использованием указателей

Код программы

**program** DoublyLinkedList;

**uses** CRT;

**type**

NodePtr = ^Node;

Node = **record**

data: string;

prev, next: NodePtr;

**end**;

**var**

head, tail, take: NodePtr;

StaticList: **Array**[1..100] **of** string;

index\_take, mx: integer;

chose: integer;

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

**var**

newNode: NodePtr;

**begin**

**New**(newNode);

newNode^.data := data;

newNode^.next := nil;

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

**begin**

newNode^.prev := nil;

head := newNode;

tail := newNode;

**end**

**else**

**begin**

newNode^.prev := tail;

tail^.next := newNode;

tail := newNode;

**end**;

take := head;

**end**;

**procedure** DeleteNode;

**begin**

**if** take = nil **then begin** writeln('Нечего удалять'); readln; **end**

**else**

**if** take = head **then begin**

Dispose(take);

head := head^.next;

take := head;

**end**

**else**

**if** take = tail **then begin**

Dispose(take);

tail := tail^.prev;

tail^.next := nil;

take := head;

**end**

**else begin**

take^.prev^.next := take^.next;

take^.next^.prev := take^.prev;

Dispose(take);

take := head;

**end**;

**end**;

**procedure** DisplayList;

**var**

current: NodePtr;

**begin**

current := head;

**repeat**

write(current^.data, ' ');

current := current^.next;

**until** current = nil;

writeln;

current := head;

**repeat**

**if** current <> take **then begin for** i:integer:=1 **to** length(current^.data) **do** write(' '); write(' '); **end**

**else** write('^');

current := current^.next;

**if** current = nil **then break**;

**until** False;

writeln;

**end**;

**procedure** message\_node;

**var**

chose: integer;

data: string;

**begin**

**while** True **do begin**

clrscr;

writeln('1 - Следующий объект');

writeln('2 - Предыдущий объект');

writeln('3 - Добавить объект');

writeln('4 - Удалить объект');

writeln('5 - Выйти');

writeln;

**if** head <> nil **then begin**

writeln('Элементы:');

DisplayList;

**if** take = nil **then** take := head

**end**;

readln(chose);

**case** chose **of**

1: **begin if** take <> nil **then if** take^.next <> nil **then** take := take^.next; **end**;

2: **begin if** take <> nil **then if** take^.prev <> nil **then** take := take^.prev; **end**;

3: **begin** writeln; write('Введите данные: '); readln(data); Addnode(data); **end**;

4: **begin** DeleteNode; **end**;

5: **begin** clrscr; **break**; **end**;

**end**;

**end**;

**end**;

**procedure** DisplayStaticList;

**begin**

**for** i:integer:=1 **to** 100 **do begin**

**if** staticlist[i] <> '' **then**

write(staticlist[i] + ' ')

**else break**;

**end**;

writeln;

**for** i:integer:=1 **to** 100 **do begin**

**if** i <> index\_take **then begin for** j:integer:=1 **to** length(staticlist[i]) **do** write(' '); write(' '); **end**

**else begin** write('^'); **break**; **end**;

**end**;

writeln;

**end**;

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

**begin**

**for** i:integer:=1 **to** 100 **do**

**if** staticlist[i] = '' **then begin** staticlist[i] := data; **break end**;

index\_take := 1;

**end**;

**procedure** set\_mx;

**begin**

**for** i:integer:=1 **to** 100 **do if** staticlist[i] <> '' **then continue**

**else begin** mx := i - 1; **break**; **end**;

**end**;

**procedure** Delete;

**begin**

**if** staticlist[index\_take] = '' **then begin** writeln('Нечего удалять'); readln; **end**

**else**

**if** index\_take = 100 **then** staticlist[index\_take]:='' **else**

**for** i:integer:=index\_take **to** 99 **do**

staticlist[i] := staticlist[i+1];

**if** index\_take > 1 **then**

dec(index\_take);

**end**;

**procedure** message\_static;

**var**

chose: integer;

data: string;

**begin**

**while** True **do begin**

clrscr;

writeln('1 - Следующий объект');

writeln('2 - Предыдущий объект');

writeln('3 - Добавить объект');

writeln('4 - Удалить объект');

writeln('5 - Выйти');

writeln;

**if** staticlist[1] <> '' **then begin**

writeln('Элементы:');

DisplayStaticList;

**end**;

readln(chose);

**case** chose **of**

1: **begin** set\_mx; **if** index\_take < mx **then** inc(index\_take); **end**;

2: **begin if** index\_take > 1 **then** dec(index\_take); **end**;

3: **begin** writeln; writeln('Введите данные:'); readln(data); add(data); **end**;

4: **begin** delete; **end**;

5: **begin** clrscr; **break**; **end**;

**end**;

**end**;

**end**;

**begin**

**while** True **do begin**

writeln('Выбирите метод работы двусвязного списка:');

writeln('1 - Статическая память');

writeln('2 - Указатели');

writeln('3 - Выйти');

read(chose);

**case** chose **of**

1: **begin** message\_static; **end**;

2: **begin** message\_node; **end**;

3: **break**;

**end**;

**end**;

**end**.

Результат выполнения программы

На рисунке 6 представлен результат выполнения программы.

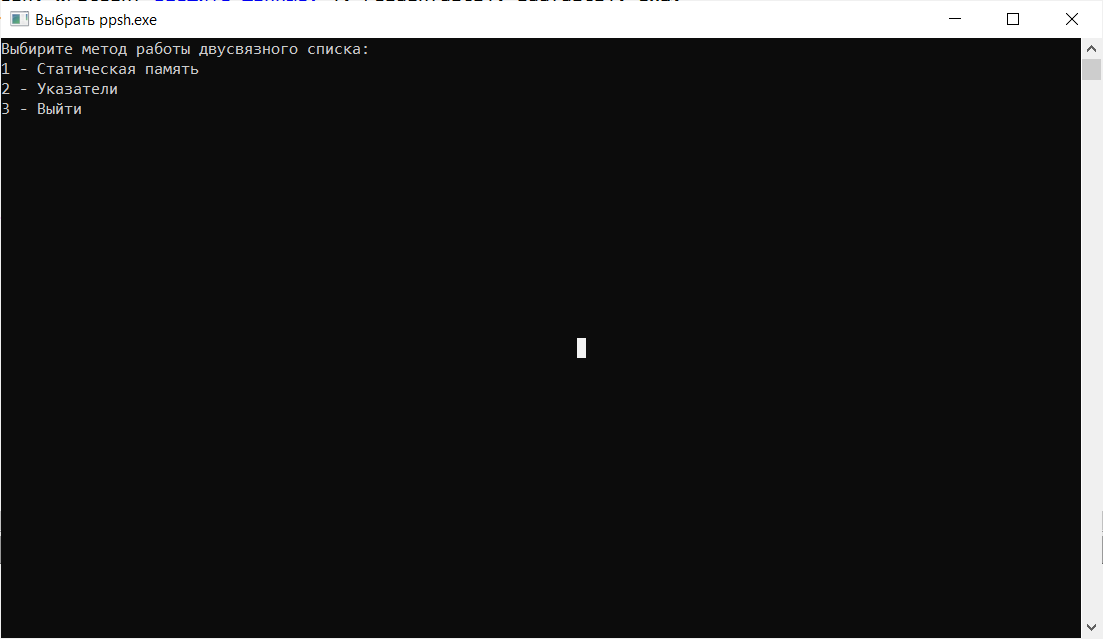


Рисунок 6 - Результат выполнения программы

Вывод

В результате выполнения домашней контрольной работы поставленная цель была достигнута. Были изучены принципы работы с базовыми структурами данных, получены навыки организации case-меню.

В ходе ДКР была разработана программа для реализации структуры данных – двусвязный список – с помощью статических и динамических структур. Были также развиты навыки работы с собственными модулями. Программа реализует два способа работы со двусвязными списками. Пользователь может выбрать какую структуру он хочет использовать для реализации стека и какие действия он хочет над ним провести.

Таким образом, все поставленные цели и задачи были успешно достигнуты и выполнены. Получены удовлетворяющие результаты.