1.8. ОРГАНІЗАЦІЯ СТЕКІВ І ЧЕРГ ЗА ДОПОМОГОЮ ДИНАМІЧНИХ СПИСКІВ

Принципово організація та робота зі стеком і чергою на базі динамічних списків не відрізняється від роботи із звичайними динамічними списками, як вже було описано у п. 1.6. Різниця полягає лише у тому, що додавання та зчитування елементів зі стеку організується з однієї сторони («голови» або «хвоста»), а для черги елементи додаються з однієї сторони, а обираються із протилежної (як у звичайній живій черзі). Яку частину списку рахувати за «голову» та «хвіст» – залежить від конкретної програмної реалізації і принципового значення не має. Нижче наведено фрагмент програмного коду для роботи зі стеком на базі списків.

```
const
  N=11;
type
  TNode=^Node;
  Node=record //структура елемента списку
    p,q: integer; //пріоритет та обсяг заявки
    Next: TNode; //адреса наступного елемента
  end;
  Mas1=array[1..N] of Node;
var
  First,Extract: TNode;
  I: byte;
  Info: Node;
const //ініціалізація масиву елементів стеку, або черги
  M: Mas1=((p:3; q:2),(p:1; q:5),(p:1; q:2),
           (p:2; q:3),(p:2; q:1),(p:4; q:4),
           (p:4; q:2),(p:3; q:1),(p:2; q:1),
           (p:4; q:5),(p:0; q:5));
procedure Show();
{Виведення на екран списку, перший елемент якого
має адресу First}
var
  NewP: TNode;
begin
  NewP:=First;
  while NewP<>nil do
    begin
      write(NewP^.p, ' ', NewP^.q, ' ');
```

```
        NewP:=NewP^.Next;
      end;
   writeln;
end;

procedure InsertNodeStack(var First:TNode; Info:Node);
{Вставка одного елемента у стек:
First - адреса першого елемента (голови) стеку;
Info - дані для елемента (структура запис), який
вставляється}
var
   NewP: TNode;
begin
   New(NewP); {формування елемента стеку і вставка
його у стек}
   NewP^.q:=Info.q;
   NewP^.p:=Info.p;
   NewP^.Next:=nil;
   if First=nil then
     begin //якщо стек порожній
        NewP^.Next:=nil;
        First:=NewP;
     end
   else
     begin //якщо стек не порожній
        NewP^.Next:=First;
        First:=NewP;
     end;
end; //procedure InsertNodeStack

procedure ExtractNodeStack(var First,Extract: TNode);
{Отримання одного елемента зі стеку:
First - адреса першого елемента (голови) стеку;
Extract - дані для елемента (структура запис),
який забирається із стеку}
begin
   if First=nil then //якщо стек порожній
     Extract:=nil
   else
     begin //якщо стек не порожній
        Extract:=First;
```

```
        First:=First^.Next;
      end ;
end; //procedure ExtractNodeStack
```

Приклад програмного коду для організації простої черги: з масиву у циклі читаються заявки, які розрізняються за пріоритетами та терміном виконання. При формуванні черги заявка вставляється у «голову» (адреса «голови» – `First`). Із «хвоста» черги (останній елемент, для якого `Next=nil`) обирається чергова заявка. У прикладі коду наведені процедури вставки та вибірки заявок із черги.

```
const N=11;
type
   TNode=^Node;
   Node=record
      p,q: integer;
      Next: TNode;
   end;
   Mas1=array[1..N] of Node;
var
   First,CurrP,GetElem: TNode;
   I,Count,count1: byte;
   Info: Node;
const
   M:Mas1=((p:3; q:2),(p:1; q:5),(p:1; q:2),
           (p:2; q:3),(p:2; q:1),(p:4; q:4),
           (p:4; q:2),(p:3; q:1),(p:2; q:1),
           (p:4; q:5),(p:0; q:5));
procedure Insert(var First:TNode; Info:Node);
{Вставка у чергу елемента в початок списку:
First - вказівник на перший елемент черги;
Info - дані елемента(p i q)}
var
   NewP: TNode;
begin
   New(NewP); //формування елемента списку
   NewP^.q:=Info.q;
   NewP^.p:=Info.p;
   NewP^.Next:=nil;
   if First = nil then //якщо список порожній
      First:=NewP
```

```
   else
     begin //якщо список має хоча б один елемент
       NewP^.Next:=First;
       First:=NewP;
     end;
end; //procedure Insert

procedure Extract(var First: TNode;
                  var GetElem: TNode);
{Забрати елемент з кінця черги:
First - вказівник на перший елемент черги;
GetElem - вказівник на останній елемент черги}
var
   P: TNode;
begin
   if First=nil then //якщо черга порожня
     GetElem:=nil
   else
     if First^.Next=nil then
       begin //якщо черга має хоча б один елемент
         GetElem:=First;
         First:=nil;
       end
     else
       begin {якщо черга має більше, ніж один елемент}
         P:=First;
         while P^.Next^.Next<>nil do P:=P^.Next;
         GetElem:=P^.Next;
         P^.Next:=nil;
       end;
end; //procedure Extract
```