Динамічні структури даних (мова Паскаль)

Ч

ерги

Переклад: Р. М. Васильчик

Стек





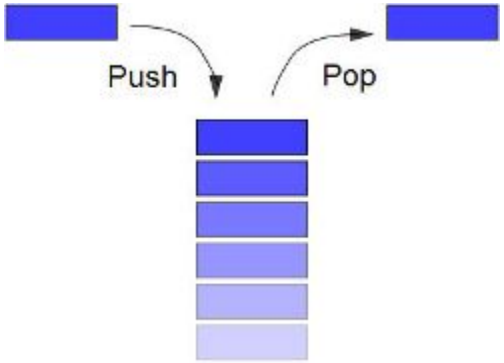
Стек – це лінійна структура даних, в якій додавання і видалення елементів можливо тільки з одного кінця (**вершини стека**). *Stack* = кипа, куча, стопка (англ.)

LIFO = *Last In – First Out*

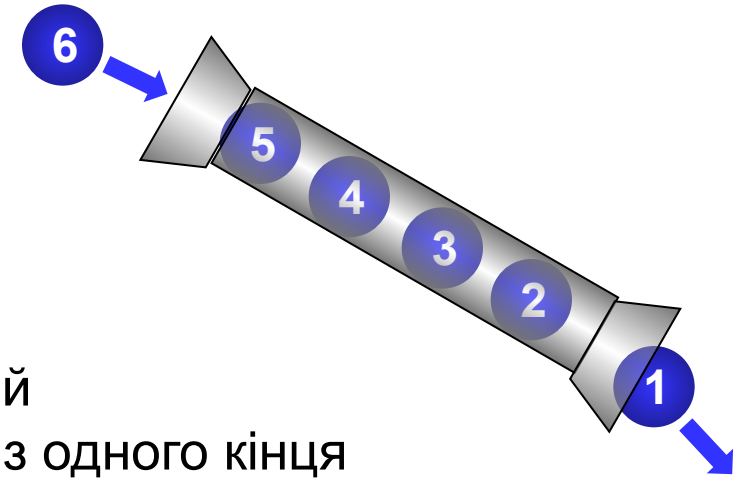
«Хто останнім увійшов, той першим вийшов».

Операції зі стеком:

1) додати елемент на вершину (*Push* = заштовхнути);
2) зняти елемент з вершини (*Pop* = вилетіти зі звуком).



Черга



Черга – це лінійна структура даних, в якій додавання елементів можливе тільки з одного кінця (**кінця черги**), а видалення елементів – тільки з іншого кінця (**початку черги**).

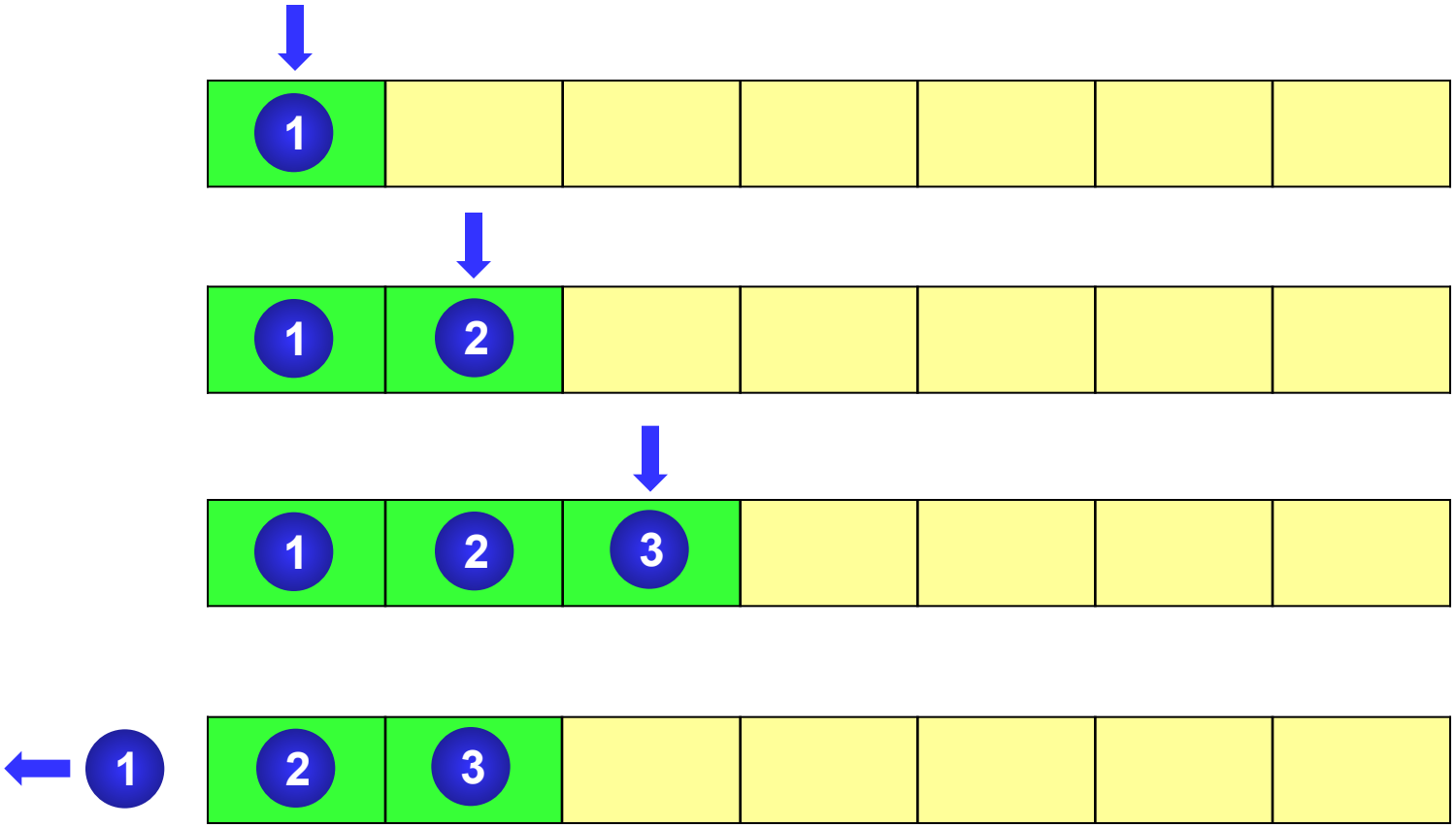
FIFO = *First In – First Out*

«Хто першим увійшов, той першим вийшов».

Операції з чергою:

1) додати елемент в кінець черги (*PushTail* = заштовхнути в кінець);
2) видалити елемент з початку черги *(Pop).*

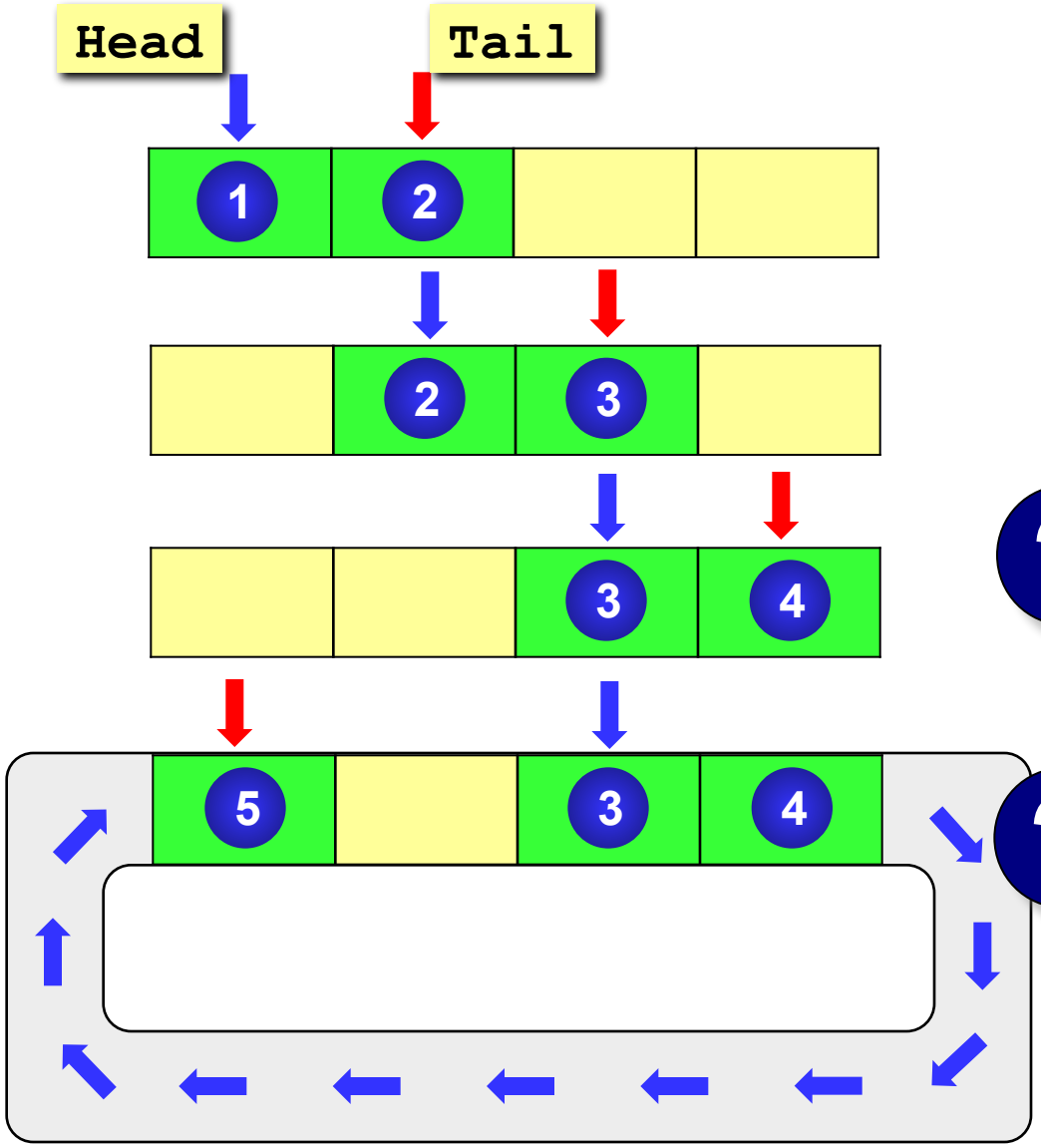
Реалізація черги (масив)

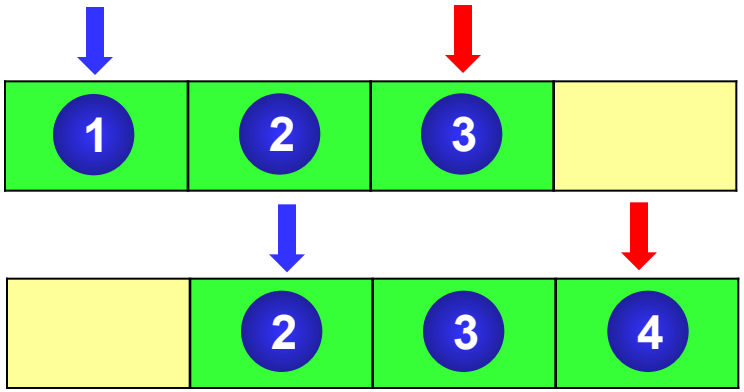


найпростіший спосіб

1) потрібно заздалегідь виділити масив;
2) при вибірці з черги потрібно зрушувати всі елементи.

Реалізація черги (кільцевий масив)



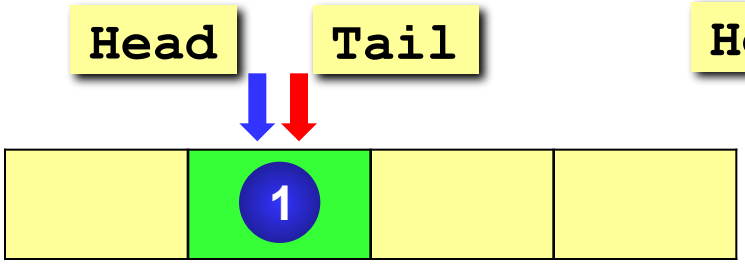


Скільки елементів можна зберігати в такій черзі?

Як розрізнити стани «черга порожня» і «черга повна»?

Реалізація черги (кільцевий масив)

В черзі 1 елемент:

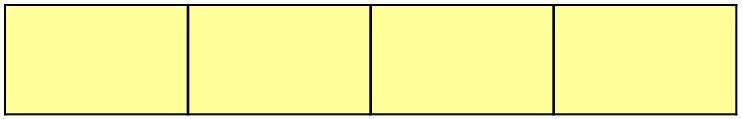


`Head = Tail`

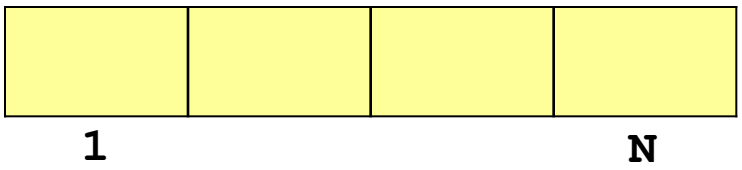
розмір масиву

Черга порожня:

`Head = Tail + 1`

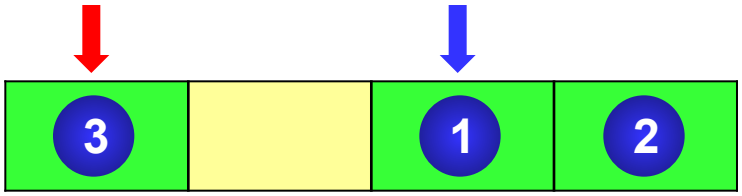


`Head = (Tail mod N) + 1`

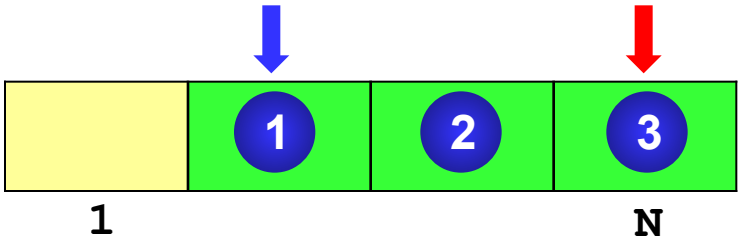


Черга повна:

`Head = Tail + 2`



`Head = (Tail+1) mod N + 1`



Реалізація черги (кільцевий масив)

Структура даних:

```
type Queue = record
       data: array[1..MAXSIZE] of integer;
       head, tail: integer;
     end;
```

Додавання в чергу:

```
procedure PushTail( var Q: Queue; x: integer);
begin
  if Q.head = (Q.tail+1) mod MAXSIZE + 1
    then Exit; { черга повна, не додавати }
  Q.tail := Q.tail mod MAXSIZE + 1;
  Q.data[Q.tail] := x;
end;
```

замкнути в кільце

Реалізація черги (кільцевий масив)

Вибірка з черги:

```
function Pop ( var S: Queue ): integer;
begin
  if Q.head = Q.tail mod MAXSIZE + 1 then begin
    Result := MaxInt;
    Exit;
  end;
  Result := Q.data[Q.head];
  Q.head := Q.head mod MAXSIZE + 1;
end;
```

черга порожня

максимальне ціле число

взяти перший елемент

видалити його з черги

Реалізація черги (списки)

Структура вузла:

```
type PNode = ^Node;
     Node = record
       data: integer;
       next: PNode;
     end;
```

Тип даних «черга»:

```
type Queue = record
       head, tail: PNode;
     end;
```

Реалізація черги (списки)

Додавання елемента:

```
procedure PushTail( var Q: Queue; x: integer );
var NewNode: PNode;
begin
  New(NewNode);
  NewNode^.data := x;
  NewNode^.next := nil;
  if Q.tail <> nil then
    Q.tail^.next := NewNode;
  Q.tail := NewNode; { новий вузол – в кінець}
  if Q.head = nil then Q.head := Q.tail;
end;
```

створюємо новий вузол

якщо в списку вже щось було, додаємо в кінець

якщо в списку нічого не було, …

Реалізація черги (списки)

Вибірка елемента:

```
function Pop ( var S: Queue ): integer;
var top: PNode;
begin
  if Q.head = nil then begin
    Result := MaxInt;
    Exit;
  end;
  top := Q.head;
  Result := top^.data;
  Q.head := top^.next;
  if Q.head = nil then Q.tail := nil;
  Dispose(top);
end;
```

якщо список порожній, …

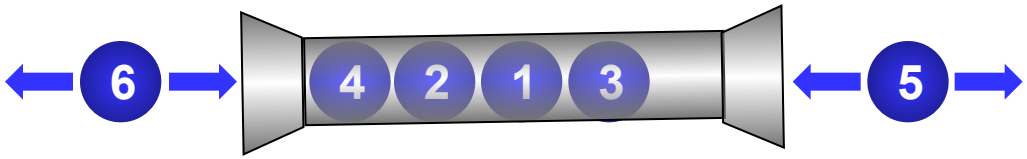
запам'ятали перший елемент

якщо в списку нічого не залишилося, …

звільнити пам'ять

Дека

Дека (*deque* = ***d****ouble* ***e****nded* ***que****ue*, черга з двома кінцями)– це лінійна структура даних, в якій додавання і видалення елементів можливе з обох кінців.



Операції з декою:

1) додавання елемента в початок *(Push);*
2) видалення елемента з початку *(Pop)*;
3) додавання елемента в кінець *(PushTail)*;
4) видалення елемента з кінця *(PopTail)*.

Реалізація:

1) кільцевий масив*;*
2) двохзв'язний список.