Strutture Dati

Array

Vengono allocate n celle in fila in base a quelle che gli servono.
Nella variabile viene memorizzata l'indirizzo della prima cella.

ACCESSO DIRETTO: mi basta il nome dell'array e l'indice per accedere ad una determinata posizione; l'accesso avviene in tempo costante O(1).

Problema: nel caso in cui si voglia lavorare con dimensioni dinamiche l'array non ci permette di aggiungere o rimuovere dinamicamente elementi.

Lista

Gli elementi sono memorizzati "dove capita", memorizzando l'informazione e l'indirizzo di quello successivo, l'importante è conoscere il nome del primo elemento.
Per accedere all'i-esimo elemento devo scorrere tutti gli elementi precedenti.

> Accesso all'i-esimo elemento O(i)

Se il numero di elementi cresce non c'è problema ad aggiungerne uno.

Primitive

1.
```
NEW_LIST
   return NIL
```

2.
```
IS_EMPTY(L)
   if L = NIL
      then retrun TRUE
      else return FALSE
```

3.
```
INSERT_H(L, e)
   e.next := L
   L := e
```

4.
```
INSERT_P(L, p, e)
   e.next := L
   p.next := e
```

5.
```
SEARCH(L, i)
   p := L
   j := 0
   while p != NIL and j < i
      p := p.next
      j := j + 1
   return p
```

6.
```
INSERT_POS(L, e, i)
   p := SEARCH(L, i)
   if p != NIL
      then INSERT_P(L, p, e)
```

7.
```
DELETE(L, p)
   if p 0 L
      then L := L.next
      else tmp := L
           while tmp.next != p
               tmp := tmp.next
           tmp.next := p.next
```

Pila

Sequenza di valori tutti dello stesso tipo messi tutti uno sopra l'altro, l'estrazione avviene sull'ultimo elemento inserito, l'inserimento avviene in testa.
Strategia LIFO: Last-in First-out

Primitive

1. NEW_STACK()
   Return NEW_LIST()

2. IS_EMPTY_STACK(S)
   Retrun IS_EMPTY_LIST(S)

3. PUSH(S, x)
   e := NEW_NODE_LIST()
   e.val := x
   e.next := NIL
   INSERT_H(S, e)

4. TOP(S)
   if not IS_EMPTY_STACK(S)
      then return S.val

5. POP(S)
   if not IS_EMPTY_STACK(S)
      then
         x := TOP(S)
         DELETE(S, S)
         Retrun x

Tutte queste operazioni si fanno in $\Theta(1)$

Coda

Primitive

1. NEW_QUEUE()

```
Q := NEW_NODE_QUEUE()
Q.head := NEW_LIST()
Q.tail := NIL
Return Q
```

2. IS_EMPTY_QUEUE(Q)

```
Retrun IS_EMPTY_LIST(Q)
```

3. ENQUEUE(Q, x)

```
e := NEW_NODE_LIST()
e.val := x
e.next := NIL
if IS_EMPTY_QUEUE(Q)
   then
      INSERT_H(Q.head, e)
      else
         Q.tail.next := e
Q.tail := e
```

4. FIRST(Q)

```
if not IS_EMPTY_QUEUE(Q)
   then return Q.head.val
```

5. DEQUEUE(Q)

```
if not IS_EMPTY_QUEUE(Q)
   then
      x := FIRST(Q)
      DELETE(Q.head, Q.head)
      if IS_EMPTY_QUEUE(Q)
         then Q.tail := NULL
      Return x
```

Il costo è $\Theta(1)$ per ogni operazione.