

LISTES

1. **Listes chaînées par pointeurs**

Une cellule contient un champ *element* et un champ *suivant*, qui est un pointeur sur une cellule. Une liste est un pointeur sur une cellule. Une position est un pointeur sur une cellule.

L liste simple chaînée par pointeurs, *a* un élément, *p* une position.

Algorithme 1 Fonction Initialiser(*L*)

```
L := CreerCellule( );  
L := NULL;  
retourner(L)
```

Algorithme 2 Fonction Insérer(*a, p, L*)

```
si p ≠ NULL alors  
  q := CreerCellule( );  
  q → element := p → element;  
  q → suivant := p → suivant;  
  p → element := a;  
  p → suivant := q  
sinon  
  AjouterFin(a, L)  
fin si  
retourner(L)
```

Algorithme 3 Fonction AjouterFin(*a, L*)

```
si L = NULL alors  
  L := CreerCellule( );  
  L → element := a;  
  L → suivant := NULL  
sinon  
  q := L;  
  tant que q → suivant ≠ NULL faire  
    q := q → suivant  
  fin tant que  
  q → suivant := CreerCellule( );  
  q → suivant → element := a;  
  q → suivant → suivant := NULL  
fin si  
retourner(L)
```

Algorithme 4 Fonction AjouterTete(a, L)

```
 $q := \text{CreerCellule}(\ );$   
 $q \rightarrow \text{element} := a;$   
 $q \rightarrow \text{suivant} := L;$   
 $L := q;$   
retourner( $L$ )
```

Algorithme 5 Fonction Precedent(p, L)

```
si  $p = L$  alors  
  Erreur  
sinon  
   $q := L;$   
  tant que  $q \rightarrow \text{suivant} \neq p$  faire  
     $q := q \rightarrow \text{suivant}$   
  fin tant que  
fin si  
retourner( $q$ )
```

Algorithme 6 Fonction Supprimer(p, L)

```
si  $p \rightarrow \text{suivant} \neq \text{NULL}$  alors  
   $q := p \rightarrow \text{suivant};$   
   $p \rightarrow \text{element} := q \rightarrow \text{element};$   
   $p \rightarrow \text{suivant} := q \rightarrow \text{suivant};$   
   $\text{Free}(q)$   
sinon si  $p = L$  alors  
   $L := \text{NULL};$   
   $\text{Free}(p)$   
sinon  
   $q := \text{Precedent}(p, L);$   
   $q \rightarrow \text{suivant} := \text{NULL};$   
   $\text{Free}(p);$   
   $p := q$   
fin si  
retourner( $L$ )
```

2. Piles

P une pile, a un élément.

P est implantée comme une structure avec un champ *table*, tableau de *max* éléments, et un champ *sommet*, indice du dernier élément.

Algorithme 7 Fonction Initialiser(P)

```
 $P.\text{sommet} := 0$ 
```

Algorithme 8 Fonction Vide(P)

```
retourner( $P.\text{sommet} = 0$ )
```

Algorithme 9 Fonction Empiler(a, P)

```
si  $P.sommet = MAX$  alors
  Erreur
sinon
   $P.sommet := P.sommet + 1$ ;
   $P.table[P.sommet] := a$ ;
fin si
```

Algorithme 10 Fonction Depiler(P)

```
si Vide( $P$ ) alors
  Erreur
sinon
   $P.sommet := P.sommet - 1$ ;
fin si
```

Algorithme 11 Fonction Sommet(P)

```
retourner( $P.table[P.sommet]$ )
```

3. Queues

Q queue implantée comme une structure comportant un pointeur *avant* et pointeur *arriere* sur une cellule. Chaque cellule contient un champ *element* et un champ *suivant*.
 a un élément.

Algorithme 12 Fonction Initialiser(Q)

```
 $p := CreerCellule( )$ ;  
 $p := NULL$ ;  
 $Q.avant := p$ ;  
 $Q.arriere := Q.avant$ ;  
retourner( $Q$ )
```

Algorithme 13 Fonction Vide(Q)

```
retourner( $Q.avant = NULL$ )
```

Algorithme 14 Fonction Tete(Q)

```
si  $Q.avant \neq NULL$  alors
  retourner( $Q.avant \rightarrow element$ )
fin si
```

Algorithme 16 Fonction EnQueue(Q)

```
si  $Q.avant \neq NULL$  alors
   $Q.avant := Q.avant \rightarrow suivant$ ;  
retourner( $Q$ )
fin si
```

Algorithme 15 Fonction AjouterQueue(a, Q)

H

si Vide(Q) **alors**

$Q.avant :=$ CreerCellule();

$Q.arriere := Q.avant$;

sinon

$Q.arriere \rightarrow suivant :=$ CreerCellule();

$Q.arriere := Q.arriere \rightarrow suivant$

fin si

$Q.arriere \rightarrow element := a$;

$Q.arriere \rightarrow suivant :=$ NULL;

retourner(Q)
