Structures de données abstraites

Sandrine Vial
sandrine.vial@uvsq.fr

Octobre 2020

Structures de Données Abstraites

- Mise en œuvre d'un ensemble dynamique
- Définition de données (**structuration** et **propriétés**)
- Définition des opérations pour **manipuler** les données.

Quelques structures classiques

1. Pile *" Dernier arrivé Premier Servi"*
2. File *"Premier arrivé Premier Servi"*
3. Tables de hachage *Généralisation des tableaux*
4. Dictionnaire *Et si l'index était une chaine de caractères ?*
5. Tas *Je sais où est le maximum (ou le minimum)*
6. Files de priorité *On suit les priorités*
7. Arbres *Et si on mémorisait les liens entre les éléments ?*
8. .....

Opérations Classiques

- Insérer un nouvel élément
- Supprimer un élément
- Rechercher un élément
- Afficher les éléments d'une structure
- ....

Une Pile

Définition

Analogie avec une pile d'assiette :
LIFO (Last In First Out ou **Dernier Arrivé Premier Servi**)

- On ne peut rajouter un élément qu'au dessus de la pile
- On ne peut prendre que l'élément qui est au dessus de la pile (élément le plus récemment inséré).

Exemple d'utilisation d'une pile

- Détecter une chaine de caractères définie par la règle suivante : $S * inverse(S)$
- Par exemple : `abc*cba`

Comment faire ?

Exemple d'utilisation d'une pile

Pour déterminer si la chaine est valide :

1. Lire les caractères jusqu'à $*$ un à un en les empilant ;
2. Après $*$, jusqu'à la fin de la chaîne, lire un caractère, dépiler le caractère suivant et comparant les 2.
3. Si les 2 caractères sont différents, afficher un message d'erreur et terminer sinon continuer.
4. Lorsque la pile est vide et qu'il n'y a plus de caractères dans la chaine, la chaine est valide.

Exemple d'utilisation d'une pile : Mots de Dyck

Le problème des mots bien parenthésés :

- (), (()), ((()()())) sont des mots valides.
- (((),)(,())() sont des mots invalides.

Comment faire ?

Exemple d'utilisation d'une pile : Mots de Dyck

Le programme lit caractère par caractère le mot entré :

1. Si c'est une parenthèse ouvrante, elle est empilée
2. Si c'est une parenthèse fermante, on dépile une parenthèse ouvrante

Le mot est accepté si :

- La pile n'est jamais vide à la lecture d'une parenthèse fermante.
- La pile est vide lorsque le mot a été lu.

Une Pile

Opérations Principales

1. Insertion d'un élément dans une pile
2. Suppression d'un élément d'une pile
3. Création d'une pile vide
4. Tester si une pile est vide
5. ...

Mise en œuvre

1. A l'aide d'un tableau (*nombre maximum d'éléments dans la pile fixé*)

Type de données

```
Enregistrement Pile {
    T[NMAX] : entier;
    Sommet : entier;
}
```

Mise en œuvre : un tableau

Algorithme 1 La pile est-elle vide ?

```
PileVide(p : Pile) : booléen
▷ Entrée : P (une pile)
▷ Sortie : vrai si la pile est vide, faux sinon.
 Debut
 si (p.Sommet = -1)
      retourner  vrai ;
 sinon
      retourner  faux ;
 fin si
 Fin
```

Complexité : $O(1)$

Mise en œuvre : un tableau

Algorithme 2 La pile est-elle pleine ?

```
PilePleine(p : Pile) : booléen
▷ Entrée : P (une pile)
▷ Sortie : vrai si la pile est pleine, faux sinon.
 Debut
 si (p.Sommet = NMAX-1)
      retourner  vrai ;
 sinon
      retourner  faux ;
 fin si
 Fin
```

Complexité : $O(1)$

Mise en œuvre : un tableau

Algorithme 3 Insertion d'un élément

Insertion(p : Pile, elt : entier)
▷ ***Entrée : p (une pile) et elt (un entier)***
▷ ***Sortie : la pile p dans laquelle elt a été inséré***
Debut
si (PilePleine(p) = faux)
    p.Sommet ← p.Sommet + 1 ;
    p.T[p.Sommet] ← elt ;
sinon
    Afficher un message d'erreur
fin si
Fin

Complexité : $O(1)$

Mise en œuvre : un tableau

Algorithme 4 Suppression d'un élément

Suppression(p : Pile) : entier
▷ ***Entrée : p (une pile) e***
▷ ***Sortie : renvoie l'élément qui était au sommet de la pile p et supprime l'élément de la pile***
▷ ***Variable locale :***
elt : entier ;
Debut
si (PileVide(p) = faux)
elt ← p.T[p.Sommet] ;
p.Sommet ← p.Sommet - 1 ;
retourner elt ;
sinon
Afficher un message d'erreur
fin si
Fin

Complexité : $O(1)$

Utilisation d'une pile

Exemple

```
Pile p ; Entier x ;
        p ← InsererPile(p,12) ;
        p ← InsererPile(p,34) ;
        p ← InsererPile(p,23) ;
        x ← SupprimerPile(p) ;
        x ← SupprimerPile(p) ;
```

Utilisation d'une pile

Exemple

```
Pile p ; Entier x ;
        p ← InsererPile(p,12) ;
        p ← InsererPile(p,34) ;
        p ← InsererPile(p,23) ;
        x ← SupprimerPile(p) ;
        x ← SupprimerPile(p) ;
```

Utilisation d'une pile

Exemple

```
Pile p ; Entier x ;
        p ← InsererPile(p,12) ;
        p ← InsererPile(p,34) ;
        p ← InsererPile(p,23) ;
        x ← SupprimerPile(p) ;
        x ← SupprimerPile(p) ;
```

12 Sommet

Utilisation d'une pile

Exemple

```
Pile p ; Entier x ;
        p ← InsererPile(p,12) ;
        p ← InsererPile(p,34) ;
        p ← InsererPile(p,23) ;
        x ← SupprimerPile(p) ;
        x ← SupprimerPile(p) ;
```

34 Sommet
12

Utilisation d'une pile

Exemple

```
Pile p ; Entier x ;
        p ← InsererPile(p,12) ;
        p ← InsererPile(p,34) ;
        p ← InsererPile(p,23) ;
        x ← SupprimerPile(p) ;
        x ← SupprimerPile(p) ;
```

23 Sommet
34
12

Utilisation d'une pile

Exemple

```
Pile p ; Entier x ;
        p ← InsererPile(p,12) ;
        p ← InsererPile(p,34) ;
        p ← InsererPile(p,23) ;
        x ← SupprimerPile(p) ;
        x ← SupprimerPile(p) ;
```

34 Sommet
12

Utilisation d'une pile

Exemple

```
Pile p ; Entier x ;
        p ← InsererPile(p,12) ;
        p ← InsererPile(p,34) ;
        p ← InsererPile(p,23) ;
        x ← SupprimerPile(p) ;
        x ← SupprimerPile(p) ;
```

12 Sommet