Structures de données abstraites

Sandrine Vial
sandrine.vial@uvsq.fr

Avril 2016

Structures de Données Abstraites

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

Quelques structures classiques

1. Pile
2. File
3. *Tables de hachage*
4. Tas
5. Files de priorité
6. Arbres
7. .....

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).

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. Quel est l'élément du sommet d'une pile ?
6. ...

Mise en œuvre

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

Type de données

```
typedef struct pile {
    int T[NMAX];
    int sommet;
} Pile;
```

Mise en œuvre : un tableau

- Créer une pile vide :
  - *Entrée* : rien
  - *Sortie* : Une pile avec aucun élément stocké.

```
Pile CreerPile()
{
  Pile P;

  P.sommet = -1;
  return P;
}
```

Mise en œuvre : un tableau

- La pile est-elle vide :
  - *Entrée :* P (une pile)
  - *Sortie :* vrai si la pile est vide, faux sinon.

```
int PileVide(Pile P)
{
  if (P.sommet == -1)
    return 1;
  else
    return 0;
}
```

Mise en œuvre : un tableau

- La pile est-elle pleine :
  - *Entrée :* P (une pile)
  - *Sortie :* vrai si la pile est pleine, faux sinon.

```
int PilePleine(Pile P)
{
  if (P.sommet == NMAX -1)
    return 1;
  else
    return 0;
}
```

Mise en œuvre : un tableau

- Insertion d'un élément :
  - *Entrée :* p (une pile) et elt (un entier)
  - *Sortie :* la pile p dans laquelle elt a été inséré

```
Pile InsererPile (Pile P, int e)
{
  if (!PilePleine(P))
    {
      P.sommet = P.sommet + 1;
      P.T[P.sommet] = e;
    }
  return P;
}
```

Mise en œuvre : un tableau

- Suppression d'un élément :
  - *Entrée :* p (une pile)
  - *Sortie :* renvoie l'élément qui était au sommet de la pile p et supprime l'élément de la pile

```
int SupprimerPile(Pile *P)
{
  int e = -1;

  if (!PileVide(*P))
    {
      e = (*P).T[(*P).sommet];
      (*P).sommet = (*P).sommet - 1;
    }
  return e;
}
```

Utilisation d'une pile

Exemple

```
Pile p ; int x ;

    p = CreerPile() ; ;
    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 ; int x ;

    p = CreerPile() ; ;
    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 ; int x ;
    p = CreerPile() ; ;
    p = InsererPile(p,12) ;
    p = InsererPile(p,34) ;
    p = InsererPile(p,23) ;
    x =
    SupprimerPile(&p) ;
    x = SupprimerPile(&p) ;
```

12

Utilisation d'une pile

Exemple

```
Pile p ; int x ;

    p = CreerPile() ; ;
    p = InsererPile(p,12) ;
    p = InsererPile(p,34) ;
    p = InsererPile(p,23) ;
    x =
    SupprimerPile(&p) ;
    x = SupprimerPile(&p) ;
```

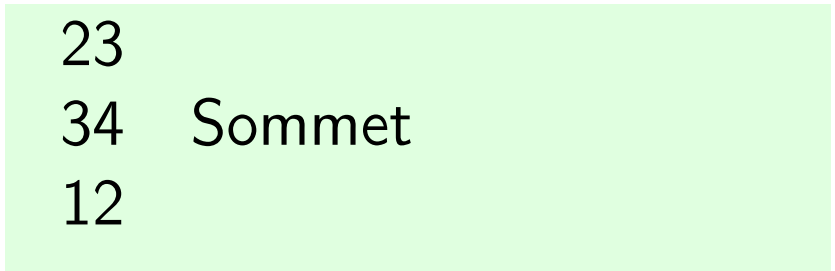
Sommet
34
12

Utilisation d'une pile

Exemple

```
Pile p ; int x ;

    p = CreerPile() ; ;
    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 ; int x ;

    p = CreerPile() ; ;
    p = InsererPile(p,12) ;
    p = InsererPile(p,34) ;
    p = InsererPile(p,23) ;
    x =
    SupprimerPile(&p) ;
    x = SupprimerPile(&p) ;
```

34
12 Sommet

Utilisation d'une pile

Exemple

```
Pile p ; int x ;
    p = CreerPile() ; ;
    p = InsererPile(p,12) ;
    p = InsererPile(p,34) ;
    p = InsererPile(p,23) ;
    x =
    SupprimerPile(&p) ;
    x = SupprimerPile(&p) ;
```

12

Exemple d'utilisation d'une pile

- Soit 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 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.

Exemple d'utilisation d'une pile : calculatrice postfixée.

On des expressions arithmétiques de la forme : $AB*C+$. On veut calculer le résultat.

1. On a une pile initialement vide
2. Pour chaque valeur numérique, on empile la valeur
3. Pour chaque opérateur, depiler une valeur que l'on stocke dans x, dépiler une valeur que l'on stocke dans y, on effectue le calcul y *operateur* x et on empile le résultat.

Que donne cet algorithme sur $6\ 5\ 2\ 3 + 8 * +3 + *$?

Exemple d'utilisation d'une pile : calculatrice postfixée.

Pb : comment passer d'une expression arithmétique sous forme infixe à une expression arithmétique sous forme postfixe ?
Pour cela on se fixe un ordre sur les opérateurs :

1. ()
2. $/*$
3. $+-$

Exemple d'utilisation d'une pile

Répéter tant que l'expression lue n'est pas vide :

1. Si le caractère lu est une valeur, afficher la valeur
2. sinon si le caractère lu est (, empiler (
3. sinon si le caractère lu est), dépiler un caractère et le stocker dans x tant que x est différent de (, afficher x et dépiler un caractère et le stocker dans x.
4. sinon tant que l'élément au sommet de la pile est plus prioritaire que le caractère lu, dépiler l'élément de sommet de pile et l'afficher. Ensuite empiler le caractère lu.

Dépiler les éléments restants de la pile et les afficher.
Exemple : $1 + 2 * 3 + (4 * 5 + 6) * 7$