Informatique pour tous - MP*

Chapitre 1 : Structure de pile

Florent Pompigne
pompigne@crans.org

Lycée Buffon

année 2017/2018

Plan

I Structures de données

Une **structure de données** est une manière d'organiser des données en un objet informatique.

I Structures de données

Une **structure de données** est une manière d'organiser des données en un objet informatique.

Le choix de cette organisation dépend des opérations que l'on souhaite pouvoir effectuer sur ces données.

I Structures de données

Une **structure de données** est une manière d'organiser des données en un objet informatique.

Le choix de cette organisation dépend des opérations que l'on souhaite pouvoir effectuer sur ces données.

Exemples :

I Structures de données

Une **structure de données** est une manière d'organiser des données en un objet informatique.

Le choix de cette organisation dépend des opérations que l'on souhaite pouvoir effectuer sur ces données.

Exemples :

- les tableaux

I Structures de données

Une **structure de données** est une manière d'organiser des données en un objet informatique.

Le choix de cette organisation dépend des opérations que l'on souhaite pouvoir effectuer sur ces données.

Exemples :

- les tableaux
- les piles

I Structures de données

Une **structure de données** est une manière d'organiser des données en un objet informatique.

Le choix de cette organisation dépend des opérations que l'on souhaite pouvoir effectuer sur ces données.

Exemples :

- les tableaux
- les piles
- les files

I Structures de données

Une **structure de données** est une manière d'organiser des données en un objet informatique.

Le choix de cette organisation dépend des opérations que l'on souhaite pouvoir effectuer sur ces données.

Exemples :

- les tableaux
- les piles
- les files
- les arbres

Type abstrait

Une structure de données est caractérisée par les opérations disponibles. Ces opérations permettent de lire, d'ajouter ou de modifier des données dans la structure.

Type abstrait

Une structure de données est caractérisée par les opérations disponibles. Ces opérations permettent de lire, d'ajouter ou de modifier des données dans la structure.

Le **type abstrait** d'une structure de données liste ces opérations, et définit leur comportement. Le type abstrait **ne** précise **pas** comment ces opérations sont implémentées, mais seulement leurs spécifications.

Type abstrait

Une structure de données est caractérisée par les opérations disponibles. Ces opérations permettent de lire, d'ajouter ou de modifier des données dans la structure.

Le **type abstrait** d'une structure de données liste ces opérations, et définit leur comportement. Le type abstrait **ne** précise **pas** comment ces opérations sont implémentées, mais seulement leurs spécifications.

Autrement dit, le type abstrait dit ce que font les opérations, mais pas *comment* elles le font.

Exemple : le type abstrait des tableaux

- **créer_tableau(n)** renvoie un nouveau tableau de n cases.

Exemple : le type abstrait des tableaux

- **créer_tableau(n)** renvoie un nouveau tableau de n cases.
- **longueur(t)** renvoie le nombre de cases du tableau t

Exemple : le type abstrait des tableaux

- **créer_tableau(n)** renvoie un nouveau tableau de n cases.
- **longueur(t)** renvoie le nombre de cases du tableau t
- **écrire(t,i,v)** remplace dans le tableau t le contenu de la case d'indice i par la valeur v

Exemple : le type abstrait des tableaux

- **créer_tableau(n)** renvoie un nouveau tableau de n cases.
- **longueur(t)** renvoie le nombre de cases du tableau t
- **écrire(t,i,v)** remplace dans le tableau t le contenu de la case d'indice i par la valeur v
- **lire(t,i)** renvoie la valeur contenue dans la case d'indice i du tableau t

Exemple : le type abstrait des tableaux

- **créer_tableau(n)** renvoie un nouveau tableau de n cases.
- **longueur(t)** renvoie le nombre de cases du tableau t
- **écrire(t,i,v)** remplace dans le tableau t le contenu de la case d'indice i par la valeur v
- **lire(t,i)** renvoie la valeur contenue dans la case d'indice i du tableau t

L'immense majorité des langages de programmation implémente cette structure de données.

Exemple : le type abstrait des tableaux

- **créer_tableau(n)** renvoie un nouveau tableau de n cases.
- **longueur(t)** renvoie le nombre de cases du tableau t
- **écrire(t,i,v)** remplace dans le tableau t le contenu de la case d'indice i par la valeur v
- **lire(t,i)** renvoie la valeur contenue dans la case d'indice i du tableau t

L'immense majorité des langages de programmation implémente cette structure de données.

En Python, la structure *List* étend cette structure de données. Il est par exemple possible d'ajouter une nouvelle case avec l'opération *append*.

Plan

II Type abstrait des piles

II Type abstrait des piles

Principe général : une pile est une structure de données linéaire dans laquelle les éléments sont dans l'ordre dans lequel ils ont été **empilé** (ie ajouté), et où seul le **sommet** (ie le dernier élément empilé) est accessible.

II Type abstrait des piles

Principe général : une pile est une structure de données linéaire dans laquelle les éléments sont dans l'ordre dans lequel ils ont été **empilé** (ie ajouté), et où seul le **sommet** (ie le dernier élément empilé) est accessible.

On dit qu'elle suit l'ordre «dernier arrivé, premier sorti »(LIFO, pour *Last In First Out*).

- **créer_pile(c)** : renvoie une pile vide, de capacité maximale c

- **créer_pile(c)** : renvoie une pile vide, de capacité maximale c
- **dépiler(p)** : dépile et renvoie le sommet de la pile p

- **créer_pile(c)** : renvoie une pile vide, de capacité maximale c
- **dépiler(p)** : dépile et renvoie le sommet de la pile p
- **empiler(p,v)** : Si la capacité maximale n'est pas déjà atteinte, empile la valeur v au sommet de la pile p

- **créer_pile(c)** : renvoie une pile vide, de capacité maximale c
- **dépiler(p)** : dépile et renvoie le sommet de la pile p
- **empiler(p,v)** : Si la capacité maximale n'est pas déjà atteinte, empile la valeur v au sommet de la pile p
- **taille(p)** : renvoie le nombre d'éléments stockés dans la pile p

- **créer_pile(c)** : renvoie une pile vide, de capacité maximale c
- **dépiler(p)** : dépile et renvoie le sommet de la pile p
- **empiler(p,v)** : Si la capacité maximale n'est pas déjà atteinte, empile la valeur v au sommet de la pile p
- **taille(p)** : renvoie le nombre d'éléments stockés dans la pile p
- **est_vide(p)** : renvoie un booléen indiquant si la pile p est vide ou non

- **créer_pile(c)** : renvoie une pile vide, de capacité maximale c
- **dépiler(p)** : dépile et renvoie le sommet de la pile p
- **empiler(p,v)** : Si la capacité maximale n'est pas déjà atteinte, empile la valeur v au sommet de la pile p
- **taille(p)** : renvoie le nombre d'éléments stockés dans la pile p
- **est_vide(p)** : renvoie un booléen indiquant si la pile p est vide ou non
- **sommet(p)** : renvoie le sommet de la pile p

- **créer_pile(c)** : renvoie une pile vide, de capacité maximale c
- **dépiler(p)** : dépile et renvoie le sommet de la pile p
- **empiler(p,v)** : Si la capacité maximale n'est pas déjà atteinte, empile la valeur v au sommet de la pile p
- **taille(p)** : renvoie le nombre d'éléments stockés dans la pile p
- **est_vide(p)** : renvoie un booléen indiquant si la pile p est vide ou non
- **sommet(p)** : renvoie le sommet de la pile p

Attention à ne pas confondre **dépiler** et **sommet** : en parallèle de sa valeur de retour, l'opération **dépiler** modifie la pile passée en argument. On dit qu'elle a un **effet de bord**.

Exercices

1. Donner l'état de la mémoire après la suite d'instructions :

```
p <- créer_pile(50)
Pour i de 1 à 10 faire
|   empiler(p,i)
FinPour
dépiler(p)
n <- dépiler(p)
m <- sommet(p)
```

2. Écrire les opérations taille et sommet à partir des autres opérations.

Plan

Une possibilité

On implémente une pile de capacité maximale *c* par un tableau de taille c+1. La case d'indice 0 contient le nombre d'éléments dans la pile, qui est aussi l'indice du sommet.