
Types de données abstraits

Cette séance de Travaux dirigée est consacrée à l'étude des types de données abstraits et à leurs implantations. On s'intéresse en particulier aux types Pile, File et Ensembles.

► Exercice 1. Théorie $\text{Pile}(T)$

1. Soit $(y, q) = \text{Depiler}(\text{Empiler}(x, \text{PileVide}))$. Montrer que $\text{EstVide}(q) = \text{VRAI}$ et $y = x$.

On suppose que l'on dispose d'une implantation du TDA $\text{Pile}(\text{elem})$ avec l'interface suivante :

```
typedef ... pile;
pile empty();
int is_empty(const pile);
push(pile &, elem);
elem pop(pile &);
```

2. Montrer que les états de la variable p après les deux suites d'instructions suivantes sont identiques :

<pre>// déclaration des types elem x,y,z,t; pile p; // séquence d'instructions 1 push(&p, x); push(&p, y);</pre>	<pre>// séquence d'instructions 2 push(&p, y); push(&p, x); z = pop(&p); t = pop(&p); push(&p, z); push(&p, t);</pre>
---	---

► Exercice 2. Implantation de $\text{Pile}(T)$ et $\text{File}(T)$

1. Implanter le TDA $\text{Pile}(\text{elem})$ avec la structure de donnée suivante :
2. Implanter le TDA $\text{Pile}(\text{elem})$ avec la structure de donnée suivante :

<pre>#define MAXPILE ... struct pile_s { int size; elem values[MAXPILE]; };</pre>	<pre>#define MAXFILE ... struct pile_s { int head, tail; elem values[MAXFILE]; };</pre>
---	---

► Exercice 3. Implantation de $\text{Ens}(T)$

Implanter grâce à un tableau de booléen le type $\text{Ens}(T)$ où T est l'intervalle de nombres entiers $[\text{min}, \text{max}]$.

► **Exercice 4. Le TDA $\text{Couple}(U, V)$**

Soit U et V deux types. Écrire la spécification d'un type abstrait pour représenter les couples (u, v) où $u \in U$ et $v \in V$.

► **Exercice 5. Pile avec taille**

1. Étendre la spécification $\text{Pile}(T)$ pour ajouter une fonction Hauteur .
2. Quel sont les axiomes vérifié par Hauteur ?

► **Exercice 6. Évaluation de expressions postfixées**

Une expression postfixée (EP) est

- soit une variable ou un nombre.
- soit une suite $x\ y\ \#$ où $\#$ est un opérateur et x, y deux EP.

Par exemple : L'expression $2 + 3$ est noté $2\ 3 +$. Le but est de mettre au point un algorithme permettant d'évaluer une EP. On considère ici les opérateurs $+, -, \times, /$.

1. Donner l'équivalent des expressions $x - y + 2$, $(2 + 3) * 4 + 1 * 5$, $2 + 3 + 4$.

Dans l'expression postfixée, l'ordre des opérateurs donne l'ordre dans lequel on doit faire les calculs (dès que l'on a un opérateur, on sait qu'il faut faire un calcul)

Voici l'idée de l'algorithme d'évaluation : Tant que l'on tombe sur une opérande, on empile sa valeur. Dès que l'on tombe sur un opérateur, on dépile les deux dernières opérandes, on effectue le calcul et on place le résultat dans la pile. À la fin, la seule valeur présente dans la pile est le résultat de l'expression qu'il fallait évaluer.

2. Écrire l'algorithme `evalEP`