Travaux Dirigés d'algorithmique n°6

Cours d'Informatique de Deuxième Année

—DEUG Sciences S3-MI—

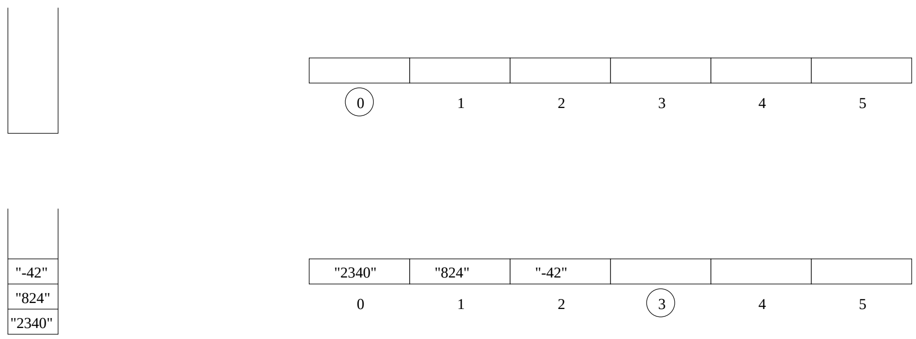
Les piles

Dans ce TD, nous introduisons un type abstrait fondamental en informatique : les piles. Ce type est entre autres à la base de l'évaluation de fonction récursives : il est utilisé par la machine pour gérer les appels de fonctions. Nous l'implémenterons de deux manières, par des listes et des tableaux, et nous comparerons les deux implémentations. Nous utiliserons ensuite les piles pour réaliser une petite calculatrice qui évalue des expressions arithmétiques écrites en notation postfixée.

Une *pile* (d'entiers, par exemple) est un type abstrait qui fonctionne comme une pile d'assiettes : on ne peut accéder qu'à l'élément du dessus. Les opérations qu'on peut faire sur une pile d'entiers sont :

- mettre un entier sur le sommet de la pile ;
- récupérer la valeur de l'entier qui est sur le sommet de la pile, et retirer cet entier de la pile.

▸ **Exercice 1.** *Nous allons commencer par implémenter les piles d'entiers par des tableaux. Par soucis de simplification on suppose que le nombre d'éléments d'une pile est majoré par une constante* `MAXPILE`*, et les éléments de la pile seront stockés dans un tableau de* `MAXPILE` *éléments.*



- *Donner un type C pour représenter les piles dont les éléments sont du type* `EltPile`*.*
- *Écrire une fonction* `InitPile` *qui initialise une telle pile.*

- *Écrire une fonction* `EstVidePile` *qui teste si une telle pile est vide.*

- *Écrire une fonction* `Empiler` *qui ajoute un élément sur le sommet d'une telle pile. Cette fonction retourne un entier qui indique si oui ou non l'opération s'est bien passée.*

- *Écrire une fonction* `Depiler` *qui effectue l'opération inverse d'*`Empiler`*. Cette fonction retourne un entier qui indique si oui ou non l'opération s'est bien passée, et met la valeur de l'entier dépilé dans un entier passé en argument à la fonction par adresse.*

▸ **Exercice 2.** *Refaire l'exercice précédent, mais en utilisant une liste pour représenter la pile plutôt qu'un tableau. Discutez les avantages et inconvénients de chacune des deux représentations.*

▸ **Exercice 3.** *On souhaite maintenant réaliser en C une calculatrice en notation postfixée (l'opérateur se trouve après les opérandes), qui dispose des opérations usuelles* $+, -, /, *$. *Pour simplifier les opérations de saisie au clavier on se limitera aux chiffres. Par exemple, l'expression* `123*4++` *vaut* $1 + ((2 \times 3) + 4) = 11$.

Écrire une fonction qui calcule la valeur d'une expression arithmétique en notation postfixée. L'expression sera fournie à la fonction sous forme d'une chaîne de caractères. Le résultat du calcul sera mis dans une variable passée (par adresse) en argument à la fonction. La fonction retournera un entier qui indique si oui ou non le calcul s'est bien passé.

Indication : il faut employer une pile, dont l'utilisation est illustrée par l'exemple suivant. On veut évaluer l'expression postfixée `123*4++`*. Au départ la pile est vide. On lit le premier caractère de l'expression. C'est le chiffre 1, on le met dans la pile. On passe au caractère suivant, c'est encore un chiffre qu'on empile, et pareil pour le suivant. À la lecture du* $*$, *la pile contient* $1, 2, 3$ *(le sommet de la pile est à la droite). On voit qu'on veut faire un produit, on dépile le 3, puis le 2, on en fait le produit et on met le résultat sur la pile, qui contient donc* $1, 6$. *On continue jusqu'à ce qu'on arrive en fin d'expression. La pile ne contient alors plus qu'un entier, 11, qui est la valeur de l'expression.*

▸ **Exercice 4.** *Une file est un type abstrait qui fonctionne comme une file d'attente à un guichet. On supprime toujours le premier élément de la file et on rajoute toujours après le dernier élément. Appelons respectivement* `Enfiler` *et* `Defiler` *ces deux opérations.*

- *Donner un type C pour représenter les files dont les éléments sont de type* `EltFile`.

- *Écrire les fonctions* `InitFile`, `EstVideFile`, `Enfiler` *et* `Defiler`.