Les types abstraits de données (TAD)

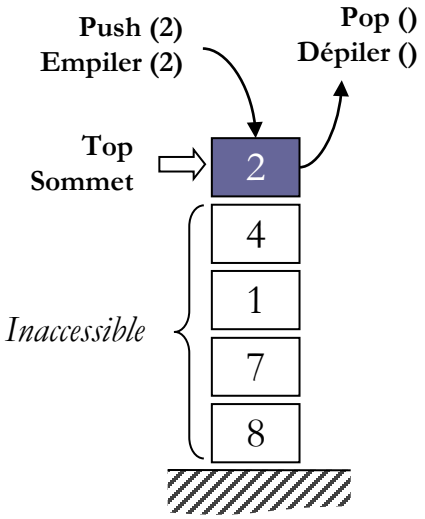
Les Piles & Les Files

Modélisation & Algorithmique

La Pile (Stack)

LIFO : last in, first out

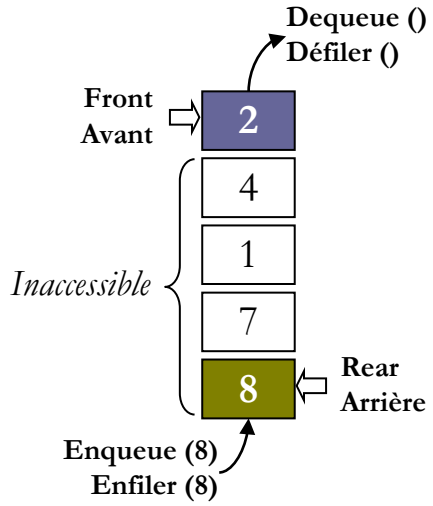
Opération	Paramètres	Le profil de l'opération		Spécification de l'opération	
		domaine	co-domaine	prés-conditions	post-condition
Sommet : *(Top)*	(P : Pile) : valeur *Fonction*	Pile	→ valeur	Pile non vide	- valeur du sommet de la Pile,
Empiler : *(Push)*	(var P : Pile, x : valeur) *Procédure*	Pile, valeur	→ Pile	Pile non pleine	- Pile résultat : pile donnée + valeur comme sommet,
Dépiler : *(Pop)*	(var P : Pile) : valeur *Fonction*	Pile	→ Pile, valeur	Pile non vide	- Pile résultat : pile donnée sans sommet, - Valeur du sommet,
Vide : *(Empty)*	(P : Pile) : booléen *Fonction*	Pile	→ booléen	~~	- Vrai si la pile et vide, - Faux si la pile et pleine,



La File (Queue)

FIFO: first in, first out

Opération	Paramètres	Le profil de l'opération		Spécification de l'opération	
		domaine	co-domaine	prés-conditions	post-condition
Avant : *(Front)*	(F : File) : valeur *Fonction*	File →	valeur	File non vide	- Valeur de l'avant de la file
Arrière : *(Rear)*	(F : File) : valeur *Fonction*	File →	valeur	File non vide	- Valeur de l'arrière de la file
Enfiler : *(Enqueue)*	(var F : File, x : valeur) *Procédure*	File, valeur →	File	File non pleine	- File résultat : File donnée + valeur comme arrière de la file
Défiler : *(Dequeue)*	(var F : File) : valeur *Fonction*	File →	File, valeur	File non vide	- File résultat : File donnée sans la valeur avant de la File - Valeur de l'avant de la File
Vide : *(Empty)*	(F : File) : booléen *Fonction*	File →	booléen	~~	- Vrai si la file et vide, - Faux si la file et pleine,



Exercice 1. *(Reconnaissance de chaîne)*

On se propose de reconnaître les chaînes de caractères définies par la règle suivante :

Une chaîne quelconque S suivie du caractère *, suivie de la chaîne S inversée. La chaîne se termine par le marqueur fin de ligne. On suppose que S ne contient pas le caractère '*'.

Exemple : ab2c*b2ca

Ecrire un algorithme qui vérifie si une chaîne est valide ou pas, en utilisant une pile.

Exercice 2. *(Palindrome)*

Un palindrome est un mot qui se lit de la même façon de gauche à droite et de droite à gauche.

Exemple : RADAR, ELLE , 1991, ETE , ..

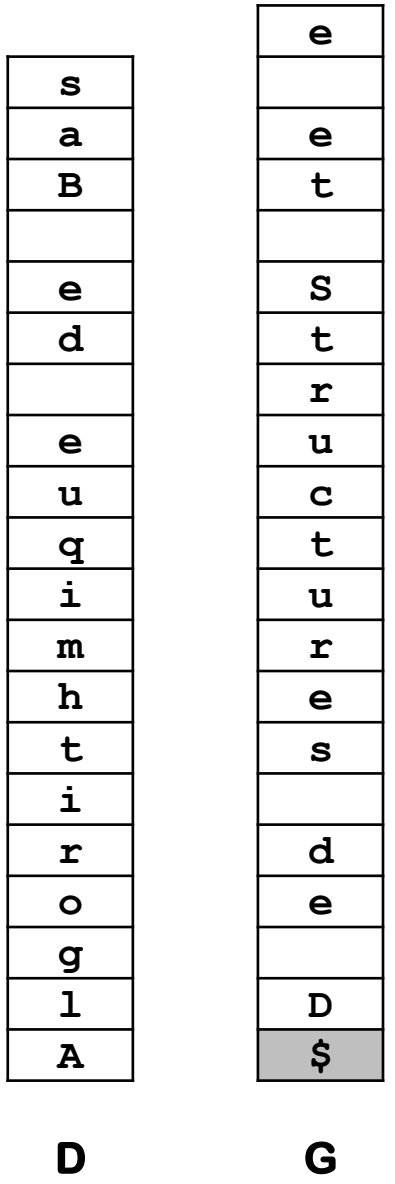
Ecrire un algorithme qui vérifie si un mot donné est un palindrome ou non, en utilisant une pile.

Exercice 3. *(Editeur de texte)*

Un éditeur de texte doit mémoriser le texte sur lequel il travaille ainsi que l'emplacement du curseur. L'ensemble texte-curseur est parfois représenté à l'aide de deux piles **G** et D :

- **G** contient les caractères situés en début de texte (avant le curseur), le dernier étant en sommet de la pile,
- **D** contient les caractères de fin de texte, le premier en sommet de pile.

On suppose de plus que l'on dispose d'un caractère de fin de ligne, soit **$** ce caractère.



Algorithmique de Bas|e et Structures de D

En utilisant les opérations du type abstrait pile (sommet, empiler, dépiler, est_vide), décrivez les opérations suivantes :

Exercice 3. *(Editeur de texte)*

1. Insérer un caractère C et positionnement du curseur après le caractère inséré,

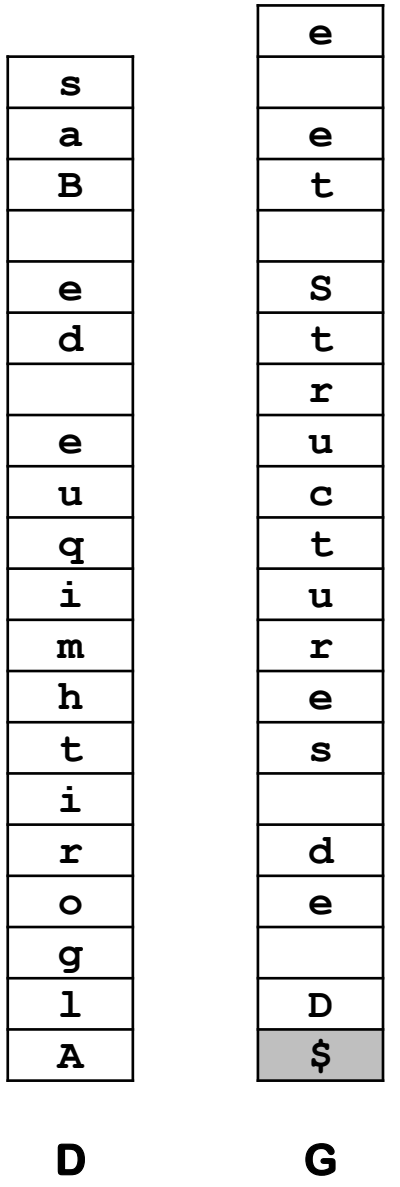
2. Effacer le caractère qui suit le curseur,

3. Avancer (vers la droite) le curseur d'un caractère,

4. Reculer (vers la gauche) le curseur d'un caractère,

5. Rechercher la première occurrence d'un caractère donné C à partir de la position courante, et positionner le curseur juste après cette occurrence si elle existe, et en fin de texte sinon,

6. Aller au début de la ligne (curseur devant le premier caractère de la ligne),



Exercice 3. *(Editeur de texte)*

7. de plus, lorsqu'on utilise un éditeur de texte, on dispose d'une touche qui permet d'effacer le caractère que l'on vient de frapper (par exemple "*BackSpace*"). Notons '#' ce caractère.

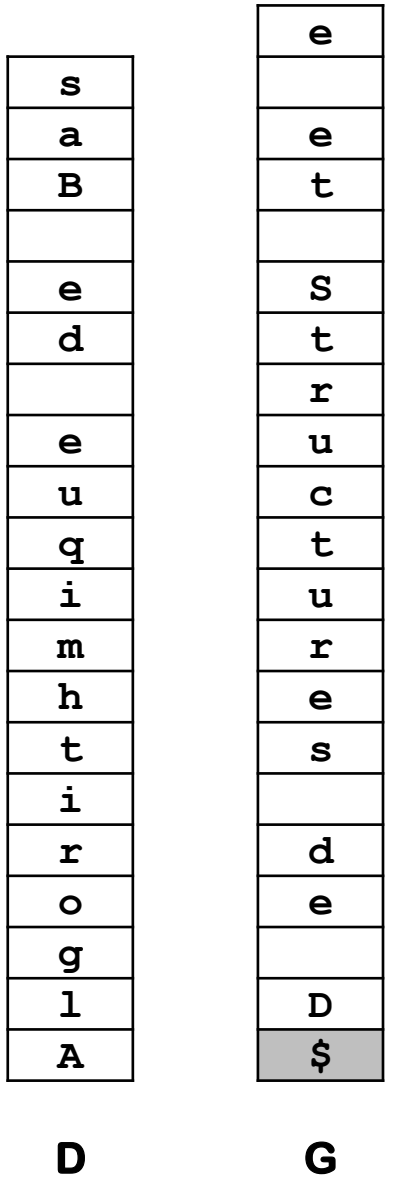
   Une autre touche permet de tout effacer jusqu'au début de la ligne. Notons '%' ce caractère.

   La fin d'une ligne sera indiquée par le caractère '$'.

   Exemple : "Jem# m'euh%Je m'#euh## suit#s trop#mp&#é$"

   sera lu comme : **"Je me suis trompé"**

   Ecrire un algorithme qui permet de lire une ligne de texte avec ce mécanisme et affiche la ligne réelle (sans les caractères d'effacement).



Exercice 4. *(Evaluation d'une expression Arithmétique)*

Etant donnée une expression arithmétique représentée par une chaîne de caractères.

1) Ecrire une procédure qui permet le passage de la forme infixée à la forme postfixée.

Exemple : l'expression "3*7" → "3 7 *"

"3*7+2*(9-6)" → "3 7 * 2 9 6 - * +"

Indication : L'ordre des opérandes est le même dans les deux notations, donc on peut les écrire directement. Tandis que pour les opérateurs et les parenthèses, il faudra les stocker dans la pile jusqu'au moment opportun pour les dépiler et les afficher.

Exercice 4. *(Evaluation d'une expression Arithmétique)*

Exemple :

a + b - c ⇨ a b + c -

A * b / c ⇨ a b * c /

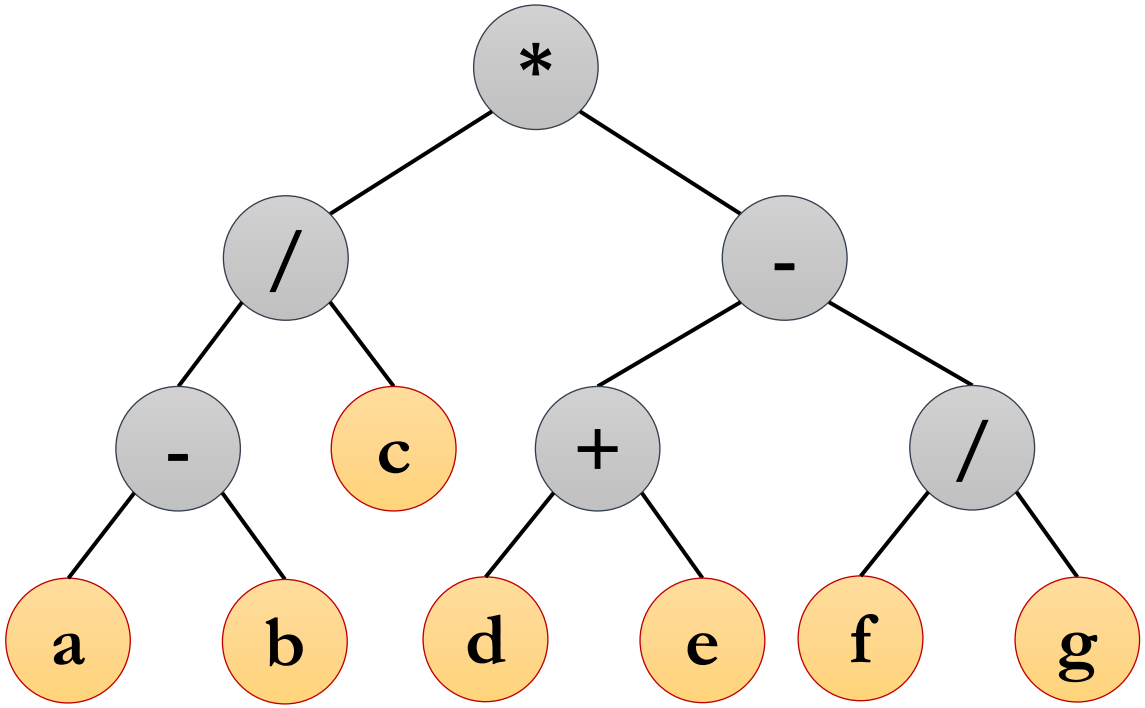
a + b * c ⇨ a b c * +

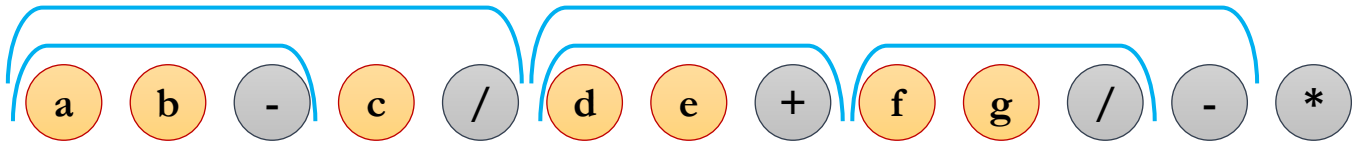
a + b * c - d ⇨ a b c * + d -

Exercice 4. *(Evaluation d'une expression Arithmétique)*

Exemple :

$$(a-b)/c*(d+e-f/g)$$





Exercice 4. *(Evaluation d'une expression Arithmétique)*

Etant donnée une expression arithmétique représentée par une chaîne de caractères.

1) Ecrire une procédure qui permet le passage de la forme infixée à la forme postfixée.

Exemple : l'expression "3*7" → "3 7 *"

"3*7+2*(9-6)" → "3 7 * 2 9 6 - * +"

2) Ecrire une procédure qui permet d'évaluer une expression postfixée

Exercice 5. *(Application File : Tri FIFO)*

On veut trier une suite de nombres entiers de la façon suivante :

1. Au départ, chaque nombre est mis dans une liste ne contenant que lui.

   Toutes ces listes sont mises dans une file (FIFO).

2. On fait sortir deux listes de la file, on les fusionne en une liste croissante.
3. On fait rentrer la liste résultat dans la file.

- On répète (2) et (3) jusqu'à ce que la file ne contienne plus qu'une seule liste, ça sera alors la suite ordonnée.

Exemple : Soit à trier la suite 4, 1, 8, 2, 3, 7, 9, 5, 6

Exercice 5. *(Application File : Tri FIFO)*

Écrire l'algorithme de ce tri, en utilisant les opérations de base sur la file :

- Init_file (F) : retourne une file vide F
- Enfiler (F, e) : ajoute l'élément e à la file F
- Défiler (F) : supprime un élément de la file F et retourne l'élément supprimé

Exemple : Soit à trier la suite 4, 1, 8, 2, 3, 7, 9, 5, 6

Merci …