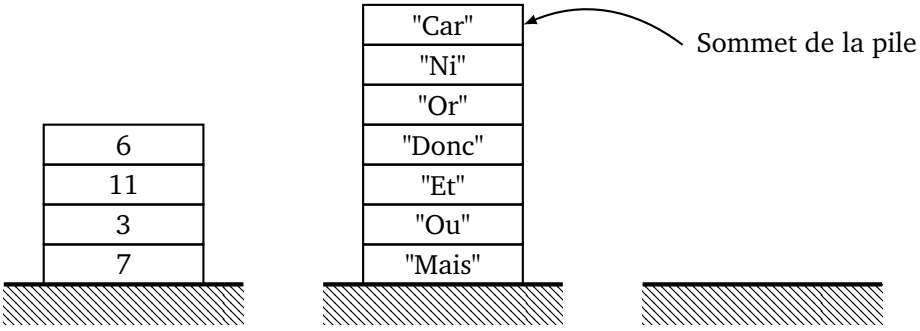
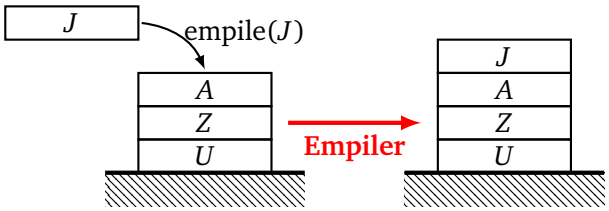
Calculatrice polonaise – Piles

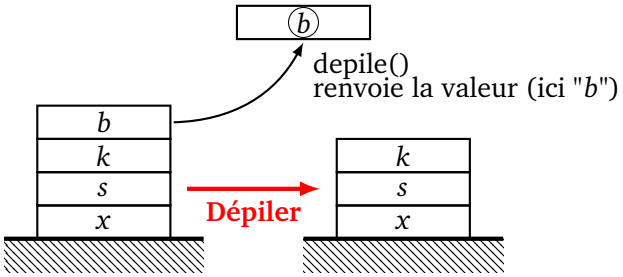
Une ***pile*** est une suite de données munie de trois opérations de base :

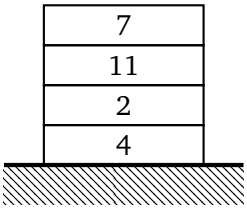
- ***empiler*** : on ajoute un élément au sommet de la pile,
- ***dépiler*** : on lit la valeur de l'élément au sommet de la pile et on retire cet élément de la pile,
- et enfin, on peut tester si la pile est vide.



Une pile d'entiers **Une pile de chaînes** **Une pile vide**


J
empile(J)
A
Z
U
Empiler
J
A
Z
U


b
depile()
renvoie la valeur (ici "b")
b
k
s
x
Dépiler
k
s
x



Une pile

```
pile = [4,2,11,7]
```

Modèle sous forme de liste

Variable globale `pile`

`empile()`

Usage : `empile(element)`
Entrée : un entier, une chaîne…
Sortie : rien
Action : la pile contient un élément en plus

Exemple : si au départ `pile = [5,1,3]` alors, après l'instruction `empile(8)`, la pile vaut `[5,1,3,8]` et si on continue avec l'instruction `empile(6)`, la pile vaut maintenant `[5,1,3,8,6]`.

`depile()`

Usage : `depile()`
Entrée : rien
Sortie : l'élément du sommet de la pile
Action : la pile contient un élément de moins

Exemple : si au départ `pile = [13,4,9]` alors l'instruction `depile()` renvoie la valeur `9` et la pile vaut maintenant `[13,4]` ; si on exécute une nouvelle instruction `depile()`, elle renvoie cette fois la valeur `4` et la pile vaut maintenant `[13]`.

`pile_est_vide()`

Usage : `pile_est_vide()`
Entrée : rien
Sortie : vrai ou faux
Action : ne fait rien sur la pile

Exemple :

- si `pile = [13,4,9]` alors l'instruction `pile_est_vide()` renvoie `False`,
- si `pile = []` alors l'instruction `pile_est_vide()` renvoie `True`.

Calculatrice polonaise

- classique : $7 + 6$; polonaise : $7\ \ 6\ \ +$
- classique : $(10 + 5) \times 3$; polonaise : $10\ \ 5\ \ +\ \ 3\ \ \times$
- classique : $10 + 2 \times 3$; polonaise : $10\ \ 2\ \ 3\ \ \times\ \ +$
- classique : $(2 + 8) \times (6 + 11)$; polonaise : $2\ \ 8\ \ +\ \ 6\ \ 11\ \ +\ \ \times$

- On lit l'expression de gauche à droite :

$$2\ \ 8\ \ +\ \ 6\ \ 11\ \ +\ \ \times \longrightarrow$$

- Lorsque l'on rencontre un premier opérateur ($+$, $\times$,…) on calcule l'opération *avec les deux membres juste avant cet opérateur* :

$$\underbrace{2\ \ 8\ \ +}_{2+8}\ \ 6\ \ 11\ \ +\ \ \times$$

- On remplace cette opération par le résultat :

$$\underbrace{10}_{\substack{\text{résultat de}\\ 2+8}}\ \ 6\ \ 11\ \ +\ \ \times$$

- On continue la lecture de l'expression (on cherche le premier opérateur et les deux termes juste avant) :

$$10\ \ \underbrace{6\ \ 11\ \ +}_{6+11=17}\ \ \times \qquad \text{devient} \qquad 10\ \ 17\ \ \times \qquad \text{qui vaut} \qquad 170$$

- À la fin il ne reste qu'une valeur, c'est le résultat ! (Ici 170.)

- 8 2 ÷ 3 × 7 +

  $\underbrace{8\ 2\ \div}_{8\div2=4}\ 3\ \times\ 7\ +$ devient $\underbrace{4\ 3\ \times}_{4\times3=12}\ 7\ +$ devient 12 7 + qui vaut 19

- 11 9 4 3 + − ×

  11 9 $\underbrace{4\ 3\ +}_{4+3=7}$ − × devient 11 $\underbrace{9\ 7\ -}_{9-7=2}$ × devient 11 2 × qui vaut 22

Algorithme.

- — Entrée : une expression en écriture polonaise (une chaîne de caractères).
  — Sortie : la valeur de cette expression.
  — Exemple : `"2 3 + 4 *"` (le calcul $(2+3)\times 4$) donne 20.
- Partir avec une pile vide.
- Pour chaque élément de l'expression (lue de gauche à droite) :
  — si l'élément est un nombre, alors ajouter ce nombre à la pile,
  — si l'élément est une opération, alors :
    — dépiler une fois pour obtenir un nombre b,
    — dépiler une seconde fois pour obtenir un nombre a,
    — calculer $a+b$ ou $a\times b$ selon l'opération,
    — ajouter ce résultat à la pile.
- À la fin, la pile ne contient qu'un seul élément, c'est le résultat du calcul.

`calculatrice_polonaise()`

Usage : `calculatrice_polonaise(expression)`
Entrée : une expression en notation polonaise (chaîne de caractères)
Sortie : le résultat du calcul
Action : utilise une pile

Exemple :

- `calculatrice_polonaise("2 3 4 + +")` renvoie 9
- `calculatrice_polonaise("2 3 + 5 *")` renvoie 25