ISM MPSI/PCSI, Informatique
Année 2024/2025

TP Informatique 18

Définition 1. *Une* pile *(stack en anglais) est une structure de données correspondant à un empilement d'éléments régi par la règle* « dernier arrivé, premier sorti » *(LIFO en anglais, Last In First Out).*

La dernière information rangée est la première disponible. La pile correspond donc exactement à l'image usuelle d'un empilement de livres ou d'assiettes : on ajoute des assiettes à la pile et on ne peut accéder directement qu'à la dernière assiette.

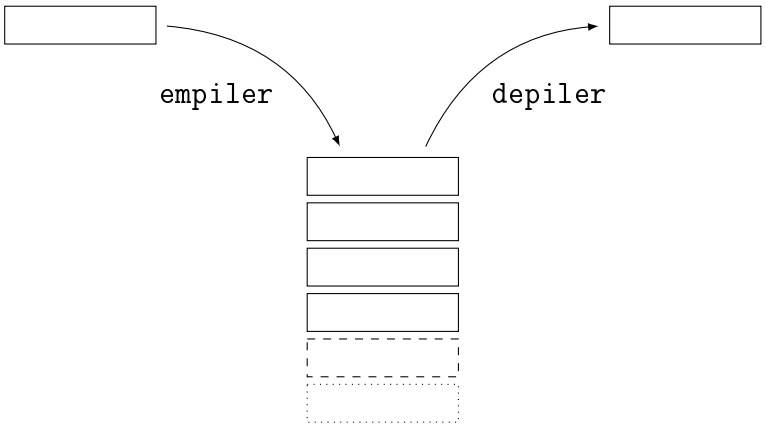

FIGURE 1 – Schéma d'une pile

Une pile peut être vue comme une structure abstraite que l'on va munir d'opérations permettant d'agir sur sa composition. Avec ces opérations, on pourra alors écrire des algorithmes utilisant des piles sans avoir besoin d'en connaître l'implémentation.

Copier le fichier `ClassePile.py` dans le répertoire de travail puis, pour chaque programme, commencer par réaliser l'importation `from ClassePile import *` pour manipuler la classe `ClassePile`. Celle-ci est munie des opérations primitives suivantes :

- `Pile()` qui crée une pile vide ;
- `P.vide()` qui renvoie `True` si la pile `P` est vide et `False` sinon ;
- `P.empiler(x)` qui empile l'élément `x` dans la pile `P` ;
- `P.depiler()` qui dépile le sommet de la pile non vide `P`.

On dispose également de l'opération `P.affiche()` pour afficher le contenu de la pile `P`.

Exercice 1

1. Écrire une fonction `taille(P)` d'argument `P` une pile et qui renvoie sa taille. La pile `P` ne doit pas être modifiée.
2. Écrire une fonction `cut(P)` d'argument `P` une pile et qui supprime l'élément tout en bas de la pile. La pile `P` est directement modifiée.

Exercice 2

On s'intéresse au problème du remplissage d'un contour dans une image. Une image sera représentée par une liste de listes contenant des zéros (cases vides) et des uns (cases pleines). Un point sera représenté par la liste de ses coordonnées `[x,y]` avec `x` le numéro de ligne et `y` le numéro de colonne. Le point en haut à gauche de l'image est le point de coordonnées `[0,0]`. Si la variable `img` contient une image, on accède au point en haut à gauche avec `img[0][0]`.

On souhaite programmer le remplissage d'un contour depuis un point situé à l'intérieur de ce contour.

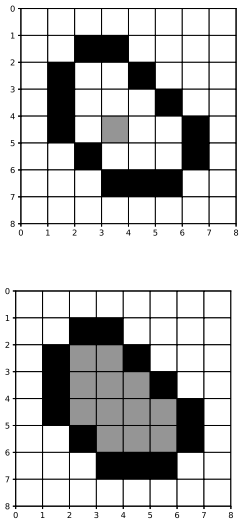

FIGURE 2 – Remplissage d'un contour

1. Copier le contenu du fichier `contour.py` dans votre éditeur puis exécuter ce code.
2. Écrire une fonction `est_blanc(image,pt)` d'arguments une image et un point et qui renvoie `True` si le point de l'image est une case vide, `False` sinon.
3. Écrire une fonction `noircit(image,pt)` d'arguments une image et un point qui remplit la case repérée par le point dans l'image. L'argument `image` sera directement modifié.
4. Écrire une fonction `remplir(image,pt)` d'argument une image contenant un contour à remplir et un point à l'intérieur du contour. On utilisera une pile pour empiler le point de départ puis, tant que la pile n'est pas vide, on dépile le sommet, on le noircit puis on empile les points voisins pas encore noircis. L'argument `image` sera directement modifié. On pourra utiliser les fonctions `haut(pt)`, `bas(pt)`, `gauche(pt)` et `droite(pt)` qui renvoient respectivement les coordonnées situés en haut, en bas, à gauche, à droite du point argument.
5. Tester la fonction `remplir` sur le contour proposé en exemple.
6. Écrire une nouvelle fonction `remplir2(image,pt)` qui remplit le contour de l'image et qui renvoie une liste permettant de visualiser l'évolution de la taille de la pile utilisée pour l'empilement des points. Afficher cette évolution pour le contour proposé en exemple.

Exercice 3

On s'intéresse au problème de parenthésage d'une expression.

1. Écrire une fonction `parenth1(expr)` d'argument une chaîne de caractère `expr` contenant une expression algébrique qui teste la validité du parenthésage pour les caractères `"("` et `")"`.
   Par exemple, la chaîne est `"((1+2)*3)-1"` est valide mais `"(1+2"` ou `"(1+2))"` ou `"((1+2)*3"` ne le sont pas.
2. Écrire une fonction `parenth2(expr)` d'argument une chaîne de caractère `expr` contenant une expression algébrique qui teste la validité du parenthésage pour les caractères `"("` et `")"`, `"["` et `"]"`, `"{"` et `"}"`.