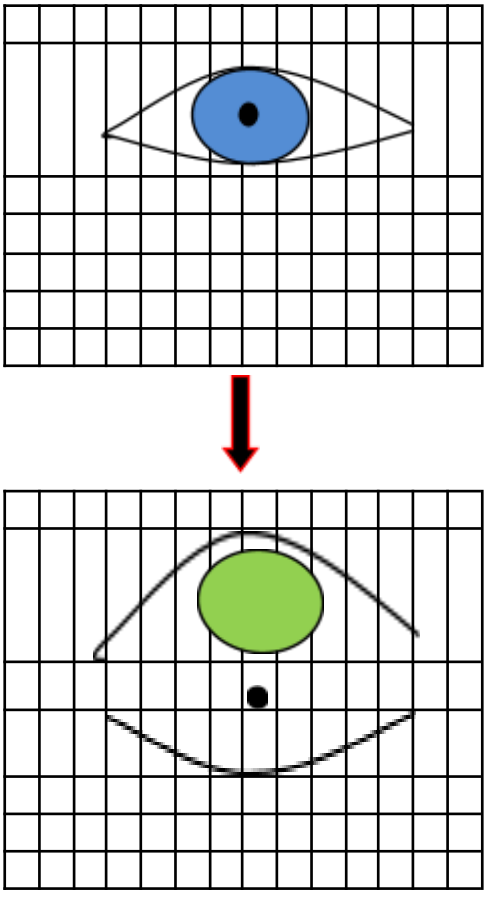
Piles

Exercice 1

Une image en informatique peut être représentée par une matrice de points ayant N lignes et M colonnes. Un élément Image[x, y] de la matrice représente la couleur du point de coordonnées (x, y). En utilisant la pile comme structure de donnée, on propose d'écrire une action paramétrée qui, à partir d'un point p donné, étale une couleur c autour de ce point. Pour effectuer le remplissage, on doit aller dans toutes les directions à partir d'un point de départ p. La progression de la couleur étalée s'arrête quand elle rencontre une couleur autre que celle du point p. Les figures suivantes illustrent ce concept, en considérant le point noir se trouvant dans la case (4, 6).



Exercice 2

Etant donné une matrice carrée T [N][N] où N > 1 et N est impair. On veut représenter un triangle dans la moitié supérieure de cette matrice en mettant à jour certaines cases de la matrice T. Pour effectuer le remplissage du triangle on procède comme suit : à chaque fois qu'on met à jour une case par un '1', on sauvegarde les coordonnées entourant cette case (gauche, droite, haut et bas) dans une **PILE**.

1. Définissez le type « coordo » représentant les coordonnées (x, y) d'une matrice et donnez le type de la pile.
2. Réécrivez les deux actions Dépiler et Empiler en tenant compte du nouveau type « coordo ».
3. Ecrivez une action (**non récursive**) qui place des '1' dans la partie supérieure de la matrice T en suivant le principe énoncé ci-dessus.

   Note : on prend comme configuration initiale les deux côtés du triangle et une position initiale à l'intérieur du triangle : T [2][4] ≠1 (voir par exemple la figure 1). Le résultat final est donné par la figure 2

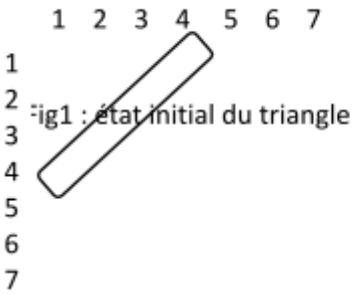


Fig1 : état initial du triangle

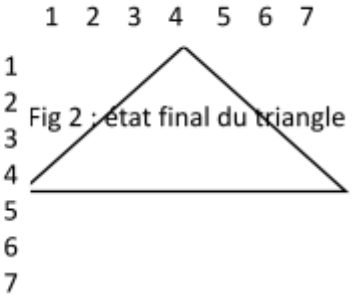


Fig 2 : état final du triangle

Exercice 3

Etant donné une matrice carrée N×N sur laquelle on a posé un objet et plusieurs barrières (voir la figure 2). L'accès d'une case aux cases adjacentes se fait par l'intermédiaire des quatre directions possibles : G : gauche, D : droite, H : haut et B : bas. Pour savoir si on peut accéder ou non à une case adjacente, chaque case utilise l'enregistrement suivant :

Type

Piece : **Enregistrement**

G : entier ; // G= 1 / 0 accéder à la case qui est à gauche ou non
D : entier ; // D= 1 / 0 accéder à la case qui est à droite ou non
H : entier ; // H= 1 / 0 accéder à la case qui est en haut ou non
B : entier ; // B= 1 / 0 accéder à la case qui est en bas ou non

Etat : entier ; // Etat = 0 / 1 / 2 non encore passé par cette case / déjà passé par cette case / objet
Fin

Exemple : la case (2, 2) est représentée par :

G=0 : accès interdit à la case (2, 1). D=0 : accès interdit à la case (2, 3). H=0 : accès interdit à la case (1, 2)]. B=1 : accès autorisé à la case (3, 2).

On veut explorer cette matrice dans le but de récupérer un objet. La démarche à suivre est donnée comme suit :

- Avant de quitter une case, il est recommandé de mettre à jour son champ Etat à 1.
- On peut consulter le champ Etat des cases adjacentes qui n'ont pas de barrière.
- Si un chemin est sans issue, on doit explorer d'autres pistes en prenant le chemin inverse. Exemple : … (1,3) (2,3) (2,4) (1,4). Emprunter …(1,4) (2,4) (2,5) ou …(1,4) (2,4) (3,4).

1. Ecrire un algorithme qui nous permet de trouver un objet sur une matrice N×N en utilisant la structure **Pile**. Sauvegarder le chemin emprunté dans une liste.

Note :

- La matrice est donnée. Initialement, le champ Etat de toutes les cases est à zéro à l'exception de la case où se trouve l'objet: Etat=2.
- Pour un exemple de déplacement : (1,1) (1,2), il est recommandé d'utiliser une pile P pour sauvegarder le couple (1,2).
- Pour le même déplacement, il est recommandé d'utiliser une autre pile PP pour sauvegarder le couple (1,1). Cette pile est très utile pour explorer d'autres pistes si un chemin est sans issue.
- Utiliser les types et les actions donnés ci-dessous.
- Aider vous de la solution de l'exercice : propagation d'une couleur dans une zone.
- ***Il est strictement interdit de parcourir la matrice avec une boucle.***

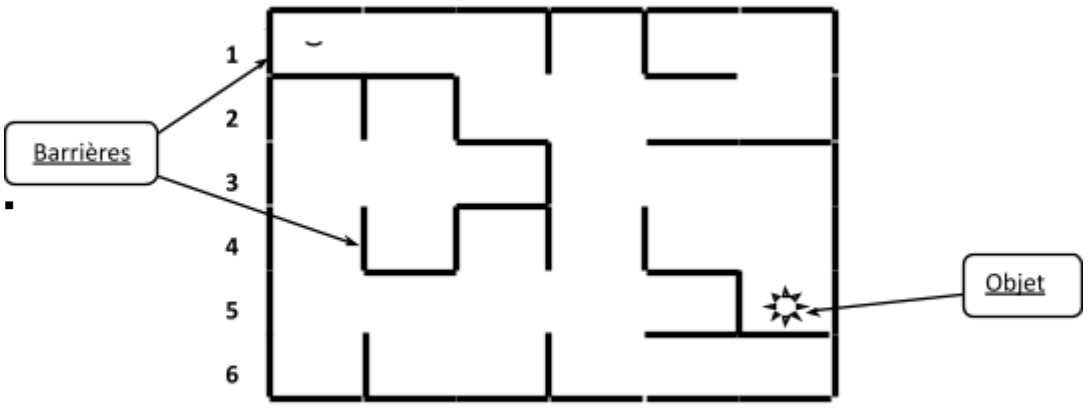


Figure 2: surface représentée par une matrice 6×6

Fonction InitPile( ) :Pile **Fonction** PileVide(E/ p :Pile) :Booléen **Fonction** SommetPile(E/ p :Pile) :Coordo **Procédure** Empiler(ES/ p :Pile, E/ x :Coodo) **Procédure** Desempler(ES/ p :Pile, ES/ x :Coodo) **Procédure** Ajout_Apres(ES/ prd :Liste, E/x :Coordo)	**Type** Coordo : **Enregistrement** Ligne : entier ; Colonne : entier ; **Fin**	**Type** Pile : ^ case case : **Enregistrement** M : Coordo ; svt : Pile ; **Fin**