Labyrinthe

Le projet sera constitué de 4 parties indépendantes réalisées dans au moins 4 fichiers séparés, compilés séparément et liés entre eux .

Parties :

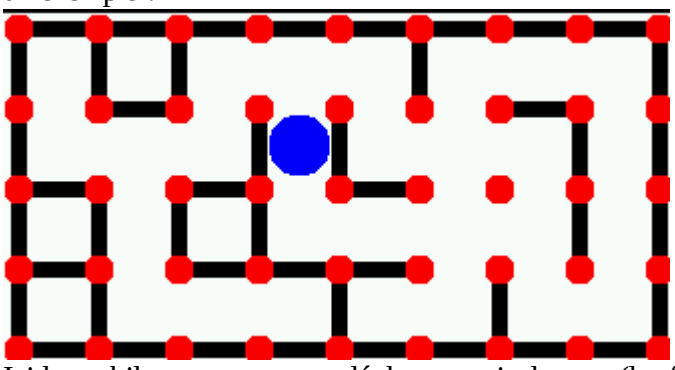
1) Génération du labyrinthe (l'initialisation)

2) Déplacement d'un mobile dans le labyrinthe (Le moteur)

l'utilisateur déplace un mobile dans un labyrinthe à l'aide des flèches de direction depuis l'entrée vers la sortie

3) Interface graphique (IG)

un exemple :



Ici le mobile ne peut que se déplacer verticalement (les flèches droite-gauche doivent être inopérantes (rôle du « moteur »)

4) Interface texte (IT)

un autre exemple

```
   0 1 2 3 4 5 6 7 8 9
    +-+-+-+-+-+-+-+-+-+
 1    | |       |     |
    + +-+ + + + + +-+ +
 2  |     | |   X   | |
    +-+ +-+ +-+ + + +
 3  | | | |         | |
    +-+ +-+-+-+ + + +
 4  | |     |   |
    +-+-+-+-+-+-+-+-+-+
```

Le mobile est en 2-6(X)

Partie 1 : Représentation et construction du labyrinthe

Le dessin figure 1 nous montre un exemple de labyrinthe.

```
# # # # # # # # #
#   /   /   /   #
# / # / # # # # #
#   /   #   /   #
# / # / # # # / #
#   #   /   #   #
# / # # # / # # #
#   #   #   /   #
# # # # # # # # #
```

Figure 1.

```
Le caractère '#' représente un mur.
Le caractère '/' représente une porte.
Le caractère ' ' représente une case du labyrinthe.
```

```
Ce labyrinthe comporte 4x4=16 cases.
```

Dans un labyrinthe, le caractère '#' représente un mur, le caractere '/' représente une porte, le caractère ' ' représente une case du labyrinthe. On se déplace dans un labyrinthe de case en case en passant par les portes. Le déplacement est horizontal ou vertical.
Ainsi, entre deux cases adjacentes d'un labyrinthe (horizontalement ou verticalement), il existe soit un mur soit une porte. Le labyrinthe, comme on le voit sur le dessin, est entouré de murs. On décide aussi de mettre des murs à l'intersection de quatre cases adjacentes.
Pour représenter un labyrinthe de `N*N` cases, on utilise un tableau de type :
`Plateau = tableau[0..2*N,0..2*N] d'entier`
Ici `N` est une constante du programme.
Notez bien que la représentation informatique d'un labyrinthe NxN est un tableau de taille (2N+1)x(2N+1).

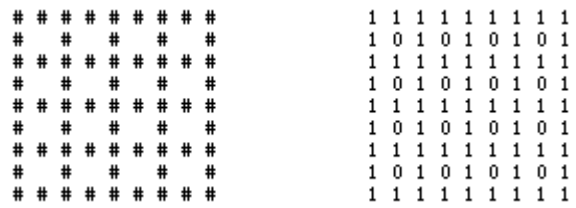
```
# # # # # # # # #          1 1 1 1 1 1 1 1 1
#   /   /   /   #          1 0 0 0 0 0 0 0 1
# / # / # # # # #          1 0 1 0 1 1 1 1 1
#   /   #   /   #          1 0 0 0 1 0 0 0 1
# / # / # # # / #          1 0 1 0 1 1 1 0 1
#   #   /   #   #          1 0 1 0 0 0 1 0 1
# / # # # / # # #          1 0 1 1 1 0 1 1 1
#   #   #   /   #          1 0 1 0 1 0 0 0 1
# # # # # # # # #          1 1 1 1 1 1 1 1 1
```

Un exemple de labyrinthe 4x4 (N=4). Sa représentation à l'écran (à gauche) et le tableau de type Plateau 9x9 lui correspondant (à droite). Ici, P[1,1] est une case du labyrinthe, P[1,2] est une porte, et P[5,2] est un mur.
Figure 2.

1) Les éléments du tableau qui ont leurs deux indices impairs représentent les cases du labyrinthe.
2) Les éléments d'indices compris entre 1 et 2*N-1 et dont l'un des indices est pair et l'autre impair représentent les murs ou les portes qui séparent deux cases du labyrinthe (horizontalement ou verticalement).
3) Les éléments du bord du tableau et ceux qui ont leurs deux indices pairs sont invariablement des murs du labyrinthe.
On décide de donner la valeur 1 aux éléments du tableau qui représentent des murs. La valeur 0 représente une case du labyrinthe ou une porte (suivant son emplacement). La figure 2 représente un labyrinthe et le tableau qui lui est associé.

Dans cette première partie du problème, on donne deux méthodes différentes de création de labyrinthe.
Pour les deux méthodes les éléments du bord du tableau sont toujours des murs (et donc à 1), de même que les éléments dont les deux indices sont pairs.
Les éléments dont les deux indices sont impairs, sont les cases du labyrinthe et sont donc à 0.
Les autres éléments d'indices compris entre 1 et 2*N-1 et dont l'un des indices est pair et l'autre impair représentent les murs ou les portes qui séparent deux cases. C'est en décidant de mettre ces derniers à 1 ou à 0 que notre programme va construire un labyrinthe. Quand l'on décide de mettre un mur cet élément est mis à 1, sinon il est mis à 0.

Question 1 : On demande d'écrire une procédure d'initialisation :
Cette procédure initialise à 0 les éléments représentant les cases du labyrinthe P et à 1 tous les autres (voir un exemple figure 3).



`Valeur du labyrinthe, après la procédure Initialisation.`
figure 3.

Question 2 : On demande d'écrire une première procédure de création (qui décide des positions des murs et des portes, à l'intérieur du labyrinthe).

Cette procédure utilise un tableau représentant un labyrinthe, préalablement initialisé grâce à la procédure Initialisation. Le labyrinthe `P` a donc au départ toutes ses cases séparées par un mur (voir figure 3).
La procédure ***`CreationAmateur`*** retourne le tableau `P` correspondant au labyrinthe obtenu en transformant exactement N*N-1 de ces murs de séparation en portes.
Pour cela, elle tire deux entiers aléatoirement et équitablement entre 1 et 2*N-1. Si la somme de ces deux entiers est impaire et si l'élément du tableau correspondant est encore à 1, celui-ci est mis à 0. Elle réitère cette opération jusqu'à ce que N*N-1 murs aient été transformés en portes.

Question 3 : On demande de programmer la fonction ***`DansPlateau`***
Cette fonction retourne vrai si `i` et `j` représentent les indices d'un élément d'un tableau de type `Plateau` qui n'est pas un élément du bord du tableau.
Question 4 : On demande d'écrire une deuxième procédure de création ***`CreationProfessionnel`***. Cette procédure plus difficile, crée des labyrinthes plus intéressants. Comme celle de la question 2, elle prend un tableau représentant un labyrinthe où toutes les séparations sont des murs, puis transforme certains de ces murs en portes.
Le choix des murs à transformer en portes est déterminé par une marche aléatoire, simulée, d'un petit bonhomme. Le petit bonhomme part de la case représentée par l'élément `1,1` et ne se déplace que de case en case adjacente, sans s'occuper des murs.
Il tire aléatoirement une des quatre directions possibles : Nord, Est, Sud, Ouest. Si ce déplacement le laisse dans le labyrinthe, il progresse indépendamment des murs dans la direction ainsi déterminée. Sinon il reste sur place. Si son déplacement le conduit sur une case pour la première fois, il transforme en porte le mur par lequel il est passé.
Il réitère cette démarche jusqu'à avoir visité chacune des cases.
Indications pour l'écriture de cette procédure :
Cette procédure commence par mettre `P[1,1]` à 2. Puis chaque fois que la marche conduit notre petit bonhomme sur une case nouvelle, l'élément correspondant du tableau devient 2. Ces valeurs 2 ne servent qu'à cette procédure. Lorsque la marche est terminée, la procédure remet les éléments correspondant à des cases du labyrinthe à 0.
On remarque que cette procédure s'arrête soit parce que toutes les cases valent 2 (mais c'est un test lourd) soit parce qu'elle a transformé exactement N*N-1 murs. Ces deux conditions d'arrêt sont équivalentes (on l'admet).

Il existe d'autres façons de modéliser un labyrinthe, vous pouvez utiliser celle que vous avez imaginé à condition de l'expliquer. Il existe aussi d'autres façons de générer les labyrinthes parmi les 2 méthodes exposées, ***`CreationProfessionnel`*** semble la meilleure, mais

CreationAmateur peut être utilisée si vous n'arrivez pas à mettre l'autre en place (Il est inutile de rendre les 2 versions).

Pour la génération de nombres aléatoires, il existe des méthodes mathématiques permettant de générer des nombres pseudo-aléatoires équiprobables. Ces méthodes sont basées sur des suites de nombres entiers. Le logiciel Maple, de calcul formel, utilise la formule suivante :
x:= x * 427419669081 mod 999999999989 (attention : il y a un 8)
il suffit d'initialiser x avec un entier quelconque pour que chaque calcul donne un nouvel entier (pseudo-aléatoire) compris entre 0 et 10^{12} .
pour initialiser x, on peut prendre l'heure system que l'on obtient grâce à une fonction de l'interruption dos (21h / 2Ch).

Partie 2 : Déplacements
L'appuis sur certaines touches (c'est le cas des touches fléchées) provoquent l'émission de 2 codes : le premier étant 0, le second dépend de la touche.
→ : 0-77
← : 0-75
↓ : 0-80
↑ : 0-72
Il convient d'exécuter l'interruption permettant la lecture d'une touche au clavier , si le code renvoyé est 0 , c'est que une touche spéciale a été enfoncée et on rappelle donc cette interruption pour identifier la touche.

Partie 3 et 4 : Affichages
2 versions interface du labyrinthe seront proposées

Remarque :
Les différentes parties étant indépendantes, elles peuvent être réparties entre les 2 membres du binôme.