**LABYRINTHE PARFAIT**

#### Algorithme Exploration exhaustive

On part d'un labyrinthe où tous les murs sont fermés. Chaque **case** (élément de la classe case) contient une variable booléenne (case.visited) qui indique si la case a déjà été visitée ou non (i.e. les cases visitées sont celles qui appartiennent au chemin du labyrinthe en cours de construction).

**Au départ**: toutes les cases sont marquées comme non visitées (*0*).

L’utilisateur choisit une case et elle sera marquée comme visitée (*1*), puis on regarde quelles sont les cellules voisines possibles et non visitées et on stocke la position en cours.

* S'il y a au moins une possibilité, on en choisit une au hasard, on ouvre le mur et on recommence avec la nouvelle case.
* S'il n'y en pas, on revient à la dernière case enregistrée dans la mémoire de la taupe (taupe.memory) comme étant une case contenant des murs encore cassable et on recommence.

Lorsque l'on est arrivée à une case ou il n’y a plus de mur cassable et que la mémoire de la taupe est vide, le labyrinthe est terminé.

L'historique des emplacements des cellules précédentes peut être géré de deux façons équivalentes :

* par la sauvegarde dans un tableau de taille *mn* − 1
* par la sauvegarde dans la pile (class mémoire dans mon programme)