Cycle de Transition
1A S1 INF1031
03/11/2018

ESIEA Paris/Ivry-sur-Seine
Année 2018/2019

Projet de programmation C

Intervenants : A. BRIERE & M. FRANÇOIS & N. KHODOR & E. MARTINS

---LABYRINTHE 2D---

Présentation générale :

L'objectif de ce projet est de modéliser en C, un labyrinthe en 2D. Le travail demandé est le suivant :

1. Aider Billy qui est à l'entrée du labyrinthe à trouver la sortie finale en le déplaçant manuellement au fur et à mesure dans le labyrinthe.
2. Aider Billy automatiquement à l'aide d'une IA (Intelligence Artificielle) à trouver un chemin quelconque entre l'entrée et la sortie.

On considère le labyrinthe comme un tableau tridimensionnel où chaque cellule est une chaîne de caractères ayant un sens en fonction des murs correspondant à la cellule. Le labyrinthe doit être chargé depuis un ficher texte contenant toutes les caractéristiques nécessaires. Le format de fichier choisi est le suivant :

- La 1ère ligne contiendra le nombre de lignes et de colonnes du labyrinthe, puis les coordonnées de l'entrée et de la sortie du labyrinthe.
- Ensuite, le fichier contiendra un tableau de chaînes de caractères indiquant la configuration initiale de chacune des cellules.

Création du labyrinthe :

On représentera le labyrinthe par des variables globales, contenant les éléments suivants :

- le tableau 3D de chaînes de caractères ;
- le nombre de lignes et de colonnes du labyrinthe ;
- la position en x et y de l'entrée et aussi de la sortie du labyrinthe.

Un exemple de labyrinthe de taille 4×4 est donné à la FIG. 1.

Représentation et configuration d'une cellule :

Une cellule du tableau sera donc représentée par une chaîne de caractères par exemple de taille 10 (vous pouvez prendre une plus grande taille si nécessaire) :

- Les quatre premiers caractères permettront de stocker la configuration initiale de la cellule, c'est-à-dire les 4 murs autour de la cellule.
- Les quatre prochains caractères pour les murs virtuels, c'est-à-dire les murs par lesquels on est rentré dans la cellule concernée. Cela permet de ne pas rentrer une fois à nouveau dans cette cellule.
- Les autres caractères pour stocker des informations telles que « est-ce que la cellule a déjà été visitée ? » ou « la cellule fait-elle partie du chemin final ? »

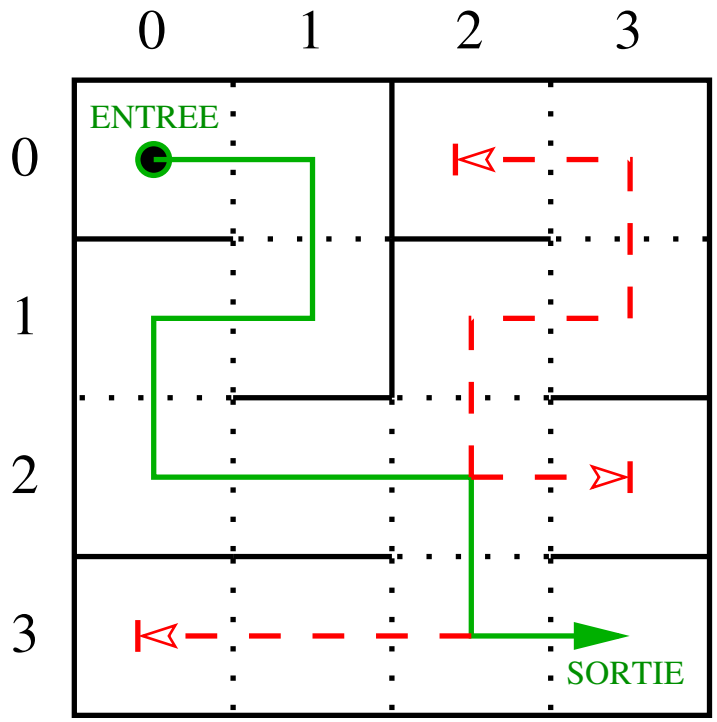

FIGURE 1 – Exemple de labyrinthe 4×4, avec une entrée (resp. sortie) en (0,0) (resp. (3,3)).

La configuration d'une cellule est représentée sur les quatre premiers caractères, de la façon suivante :

Le caractère d'indice 0 sera égal à M si la cellule contient un mur infranchissable vers le haut et $-$ sinon. Les caractères d'indices 1 , 2 , et 3 indiqueront la présence d'un mur à droite, en bas, et à gauche respectivement. Par exemple, la cellule $(0, 0)$ de la FIG. 1, sera configurée initialement à l'aide de M$-$MM, alors que la cellule $(2, 3)$ sera représentée par MMM$-$.

Pendant le chargement du labyrinthe, il faudra faire attention à ce que les cellules soient cohérentes entre elles. Par exemple, lorsqu'une cellule est initialisée sans mur vers le bas, il faudra s'assurer que celle en dessous dans le labyrinthe n'ait pas de mur vers le haut.

Voilà un exemple de fichier texte contenant une configuration de labyrinthe de taille 10×10 ayant comme entrée la position $(0, 0)$ et en sortie la position $(9, 9)$:

```
10 10 0 0 9 9
M−−M MM−− MM−M M−MM M− − − M− − − MM−− MM−M M−−M MM−−
−−MM −−M− −−M- M− − − − − − −− −−M− −−M− − − − −− −MM− −MMM
M−−M MMM− M−−M − − −− − − −− M− − − M−M− −M−− M−MM MMM−
− − −M MM−− −−MM −MM− − − −M −M−− M−−M − − −− MMM− MM−M
− − −M −M−− M−−M M− − − − − − −− −−M− − − − −− −M−− MMMM −M−M
−M−M − − −M − − − −− −M−− − − −M M−M− −−M− −M−− M−−M −MM−
− − −M −M−− −M−M − − −M − − − −− MMM− MM−M −M−M −M−M MM−M
−MMM −M−M −M−M − − −M −MM− MMMM −M−M − − −M − − −− −MM−
M−−M − − −− −−M− −M−− M−−M MM−− −MMM − − −M −M−− MM−M
−MMM −−MM M−M− −−M− −MM− −MMM MMMM −MMM −−MM −MM−
```

Sur chaque ligne les blocs de 4 caractères sont séparés par un espace. Il y a autant de blocs de 4 caractères que de cellules dans le labyrinthe.

Recherche d'un chemin :

Dans le menu de départ, on doit avoir les différents choix possibles à savoir :

- recherche manuelle d'un chemin ;
- recherche automatique d'un chemin.

Pour la recherche d'un chemin, à chaque fois que Billy se déplacera sur l'écran on met un symbole dans la cellule (par exemple une $*$). Donc à la fin du parcours, toutes les cellules appartenant au chemin final doivent pouvoir être distinguables sur l'écran.

Concernant la recherche d'un chemin quelconque via l'IA, un parcours de l'espace de recherche en profondeur permettra de trouver un chemin possible dans le labyrinthe. Comme évoqué au dessus, les autres caractères peuvent être utilisés pour stocker l'évolution de la configuration des cellules pour la recherche d'un chemin. Il est possible parfois que le labyrinthe ne présente pas de chemin entre l'entrée et la sortie, dans ce cas, le résultat de la recherche de chemin doit indiquer cela.

Pour faire vos choix dans le menu ou même pendant le déroulement, vous pouvez utiliser la fonction `key_pressed()` indiquée ci-dessous :

```
-------------------------------------------------------------------
char key_pressed()
{
    struct termios oldterm, newterm;
    int oldfd; char c, result = 0;
    tcgetattr (STDIN_FILENO, &oldterm);
    newterm = oldterm; newterm.c_lflag &= ~(ICANON | ECHO);
    tcsetattr (STDIN_FILENO, TCSANOW, &newterm);
    oldfd = fcntl(STDIN_FILENO, F_GETFL, 0);
    fcntl (STDIN_FILENO, F_SETFL, oldfd | O_NONBLOCK);
    c = getchar();
    tcsetattr (STDIN_FILENO, TCSANOW, &oldterm);
    fcntl (STDIN_FILENO, F_SETFL, oldfd);
    if (c != EOF) {ungetc(c, stdin); result = getchar();}
    return result;
}
-------------------------------------------------------------------
```

Cette fonction vous permet de récupérer la touche saisie par l'utilisateur sans pour autant retarder l'affichage car, le plan doit être dynamique. Pour cela vous aurez besoin des librairies supplémentaires suivantes :

`signal.h`, `string.h`, `termios.h`, `unistd.h` et `fcntl.h`.

Consignes et date de rendu du projet :

Le projet est à réaliser en binôme sous environnement GNU/Linux, et doit être déposé sur la plate-forme pédagogique *Moodle* au plus tard le Dimanche 20/01/2019 à 23h55, sous la forme d'une archive `.zip` à vos noms et prénoms (*i.e.* `NOM_prenom.zip`), contenant tous les fichiers sources du projet.

Votre programme doit obligatoirement présenter les différentes thématiques suivantes :

- tableaux et pointeurs ;
- chaînes de caractères ;
- des lignes de codes commentées, lisibles et bien agencées ;
- un bon choix de noms de variables.

L'évaluation sera globalement scindée en trois parties :

1. la façon de charger et afficher le labyrinthe,
2. la recherche de chemin manuelle ;
3. la recherche de chemin automatique via l'IA.

Conseils pour la réalisation technique :

Il faut commencer par écrire la fonction qui va vous permettre d'afficher un labyrinhte y compris les contours, en utilisant un fichier en entrée adapté.

Vous pouvez rafraîchir à chaque fois l'affichage du labyrinthe sur l'écran, car les cellules vont être modifiées au fur et à mesure.

Vous pouvez faire un affichage plus dynamique en vous positionnant directement sur la cellule dont le contenu doit être modifié. Par exemple, en utilisant l'instruction :

```
printf("\033[%d;%dH", 15, 29);
```

Cela déplace le curseur sur l'écran à la position $(15, 29)$ ($i.e$ ligne 15 et colonne 29). Une fois à la position souhaitée, on peut afficher un symbole dans la cellule ou même effacer le contenu déjà existant de cette dernière.