Cycle : Fondamental
Niveau : 3ème année
Semestre : 1

ESIEA Paris-Ivry/Laval
Année 2014/2015
26/09/2014

Projet programmation C

(Encadrement : Michael FRANÇOIS)
michael.francois@esiea.fr

---LABYRINTHE---

Présentation générale :

L'objectif de ce projet est de modéliser en C, un labyrinthe en 2D. Ensuite, faire une recherche d'un chemin quelconque entre l'entrée et la sortie, puis le plus court chemin dans ce labyrinthe. On considère le labyrinthe comme un tableau bidimensionnel d'entiers courts, de taille donnée. On doit pouvoir générer le labyrinthe de plusieurs façons :

1. En fixant dès le départ tous les paramètres du labyrinthe (taille du tableau, entiers, position de l'entrée, etc.), afin de bien débuter et bien contrôler votre programme.
2. Générer aléatoirement un labyrinthe avec toutes ses caractéristiques de base.
3. On doit pouvoir également charger un labyrinthe depuis un fichier texte, contenant toutes les caractéristiques nécessaires. Le format de fichier choisi sera 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 valeurs indiquant la configuration initiale de chaque cellule.

Création du labyrinthe :

On représentera le labyrinthe par une structure, contenant les éléments suivants :

- le tableau 2D d'entiers courts (`unsigned short`),
- le nombre de lignes et de colonnes du labyrinthe,
- la position en x et y de l'entrée/sortie du labyrinthe,
- ainsi que la position en x et y du chercheur de chemin.

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 un entier court de type `unsigned short` ($i.e.$ sur 16 bits : $b_{15}b_{14}b_{13}b_{12}b_{11}b_{10}b_9b_8b_7b_6b_5b_4b_3b_2b_1b_0$) :

— Les quatre bits de poids faibles ($b_3b_2b_1b_0$) permettront de stocker la configuration initiale de la cellule, c'est-à-dire les 4 murs autour de la cellule.
— Les quatre bits de poids suivants permettront de stocker l'évolution de la configuration de la cellule lors de la recherche d'un chemin.
— Les huit bits restants serviront à stocker par exemple, la distance de la cellule à l'entrée du labyrinthe. Ces huit bits ne serviront que dans le cas de la recherche d'un chemin.

La configuration d'une cellule est représentée sur les quatre bits de poids faibles ($b_3b_2b_1b_0$), de la façon suivante :
Le bit b_3 sera égal à 1 si la cellule contient un mur infranchissable vers le haut. Les bits b_2 ,

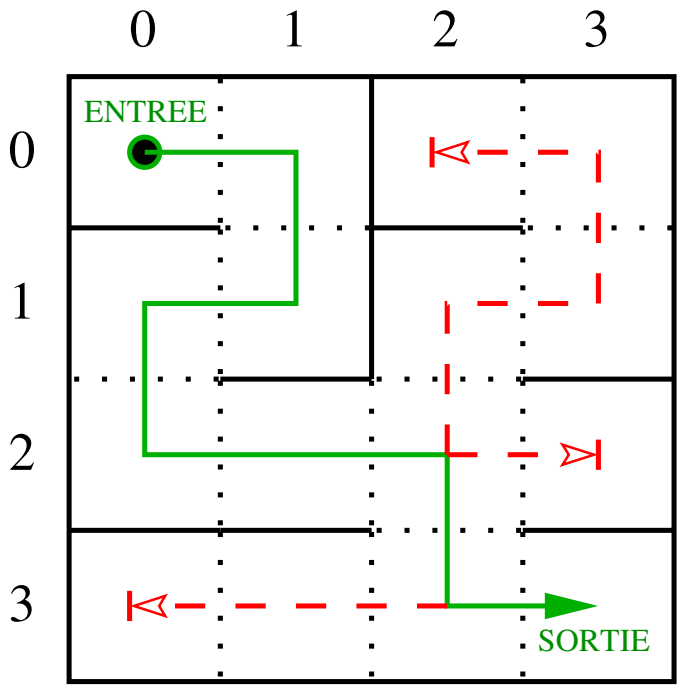

FIGURE 1 – Exemple de labyrinthe 4×4, avec une entrée (resp. sortie) en (0,0) (resp. (3,3)).

b_1 , et b_0 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 1011, alors que la cellule $(2, 3)$ sera représentée par 1110.

Pendant la génération du labyrinthe, il faudra faire attention 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
9 12 13 11 8 8 12 13 9 12
3 2 2 8 0 2 2 0 6 7
9 14 9 0 0 8 10 4 11 14
1 12 3 6 1 4 9 0 14 13
1 4 9 8 0 2 0 4 15 5
5 1 0 4 1 10 2 4 9 6
1 4 5 1 0 14 13 5 5 13
7 5 5 1 6 15 5 1 0 6
9 0 2 4 9 12 7 1 4 13
7 3 10 2 6 7 15 7 3 6
```

Les valeurs sont petites ici, car les 12 bits de poids forts ont été initialisés à 0.

Recherche d'un chemin :

La recherche d'un chemin quelconque ou même du plus court, sera accessible à travers le menu principal de votre programme. Un parcours de l'espace de recherche en profondeur permettra de trouver un chemin possible dans le labyrinthe. Comme évoqué au dessus, les

autres bits peuvent être utilisés pour stocker l'évolution de la configuration des cellules pour la recherche d'un chemin. Vous devrez réfléchir sur la façon d'optimiser votre stratégie dans le cas de la recherche du plus court chemin. Il est possible parfois que le labyrinthe ne présente pas de chemin entre l'entrée et la sortie, notamment en cas de génération aléatoire. Dans ce cas, le résultat de la recherche de chemin doit indiquer cela.

NB :
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 21/11/2014 à 23h59, sous la forme d'une archive `.tar.gz` à vos noms et prénoms (*i.e.* `NOM_prenom.tar.gz`), contenant tous les fichiers sources du projet et également un rapport `pdf`. Vous expliquerez toutes les démarches effectuées sur chaque partie et conclure en insistant notamment sur les difficultés rencontrées et les problèmes non résolus (s'il en reste).

Votre programme doit obligatoirement présenter les différentes thématiques suivantes :

- tableaux, pointeurs et allocation dynamique,
- structures,
- manipulation de bits en cases mémoires,
- traitement de fichiers,
- fichier *Makefile* contenant les commandes de compilation,
- 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 créer, charger et afficher les labyrinthes,
2. la recherche de chemin quelconque et celui du plus court chemin,
3. le rapport `pdf` doit être soigneux, complet et bien détaillé.

Quelques liens utiles sur les opérateurs bit à bit en langage C :

`http://progdupeu.pl/tutoriels/15/le-langage-c/pour-aller-plus-loin/operations-bit-a-bit-et-champs-binaires/`

`http://emmanuel-delahaye.developpez.com/tutoriels/c/operateurs-bit-bit-c/`

Quelques liens utiles sur les fichiers `Makefile` en C :

`http://gl.developpez.com/tutoriel/outil/makefile/`

`http://www.cmi.univ-mrs.fr/~contensi/coursC/index.php?section=env&page=make`