

Mini-projet «Curiosity Revolutions»

TD6 — Structure de base, lecture du terrain

NB : ce projet utilise le même environnement que l'APP2 «Curiosity Reloaded» de l'UE INF301; cependant les problèmes posés et les exercices demandés sont orthogonaux. Il n'est pas nécessaire d'avoir suivi l'UE INF301 pour réaliser ce projet.

1. Contexte

« Liquid water on Mars! » Maintenant, on en est sûrs, il y a de l'eau liquide sur Mars. Petit problème : le plan d'exploration du robot Curiosity est complètement perturbé, il ne faudrait pas qu'il s'embourbe dans de la boue martienne ou tombe au fond d'une flaque par mégarde...

De nouveaux programmes ont été développés pour ce robot, pour tenir compte de cette nouvelle situation. Vous allez travailler sur un simulateur permettant de tester ces programmes : il vous faudra notamment vérifier que ce simulateur a bien le comportement attendu.

2. Organisation générale

Le programme sur lequel on travaille est un simulateur du robot Curiosity. Ce robot évolue sur un *terrain*, en exécutant un *programme* écrit dans un langage spécifique.

Exercice 1. Réflechissez à la structure globale de ce simulateur : quels peuvent être les modules du programme? Quels types et/ou fonctions sont définies dans chacun de ces modules?

3. Le robot

Le robot va être amené à se déplacer dans le terrain à partir d'une case initiale pour si possible atteindre une case but, ou bien en sortir (la sortie est l'extérieur du terrain).

Le robot est défini par :

- sa position (x,y) en coordonnées entières,
- son orientation o pouvant prendre 4 valeurs possibles : $|NORD:\uparrow||EST:\rightarrow||SUD:\downarrow||OUEST:\leftarrow|$

Le type robot peut donc être défini ainsi :

```
typedef enum { Nord, Est, Sud, Ouest } Orientation;

typedef struct {
  int x, y;
  Orientation o;
} Robot;
```

Le robot est capable des actions suivantes :

- avancer,
- tourner d'un quart de tour à droite sur lui-même,
- tourner d'un quart de tour à gauche sur lui-même,
- effectuer une mesure.

Pour ce projet, l'interprétation des programmes des robots (fournis dans un langage spécifique — cf INF301) sera fournie.

INF304 2023/24 TD6 1/2

4. Le terrain

Un *terrain* est un rectangle composé de cases carrées (L en largeur et H en hauteur), chaque case pouvant être libre (caractère '.'), occupée par de l'eau (caractère '~'), ou occupée par un rocher (caractère '#'). La case initiale du robot est une case libre, marquée par le caractère 'C'.

Préconditions pour le terrain :

- les dimensions L (largeur) et H (hauteur) doivent être inférieures à une dimension maximale DIM_MAX,
- il existe un chemin (formé de cases libres) entre la case centrale et l'extérieur du terrain (la sortie).

Un fichier terrain est un fichier composé:

- 1. d'un entier L, la largeur du terrain
- 2. d'un entier H, la hauteur du terrain
- 3. de H lignes de L caractères dans l'ensemble { '#', '.', '~', 'C'}.

Exemple de fichier «terrain» :

```
11
9
#..#...#
.....C....#
#..#....#
#..#....#
#..##.....#
#..##.....#
....##
```

Déclaration du type terrain en C :

```
typedef enum { LIBRE, EAU, ROCHER } Case;

#define DIM_MAX 256

// indexation utilisée :
    // ler indice : abscisse = colonne (colonne de gauche : abscisse = 0)
    // 2è me indice : ordonnée = ligne (ligne du haut : ordonnée = 0)

typedef struct {
    int largeur, hauteur;
    Case tab[DIM_MAX][DIM_MAX];
} Terrain;
```

Exercice 2. Écrire une fonction lire_terrain, permettant de lire un terrain dans un fichier (on suppose dans un premier temps que le format du fichier à lire est correct et les préconditions remplies).

Exercice 3. Si le fichier n'existe pas ou est incorrect, quelles erreurs peuvent se produire à l'exécution de la fonction lire_terrain?

Compléter la fonction lire_terrain afin de renvoyer une erreur si le format du fichier n'est pas correct.

INF304 2023/24 TD6 2/2