INF304 — TP6

Curiosity Revolutions (1) — Structures de base, terrains

Récupérer les fichiers nécessaires au TP6 :

```
cp -r /Public/304_INF_Public/TP6 .
```

Au cours de ce TP et des suivants, vous serez amenés à compléter des programmes fournis. Il est important de vous les approprier.

Le terrain

Le robot se déplace dans un terrain rectangulaire composé de cases. Ces cases sont soit *libres*, soit occupées par un obstacle.

Les terrains seront lus depuis des fichiers textes dont le format est le suivant :

- sur la première ligne, un entier `L` : nombre de cases en *largeur* (ou nombre de colonnes),
- sur la deuxième ligne, un entier `H`: nombre de cases en *hauteur* (ou nombre de lignes),
- puis `H` lignes de `L` caractères qui peuvent être :
  - soit `'.'` pour représenter une case libre,
  - soit `'#'` pour représenter une case «rocher»,
  - soit `'~'` pour représenter une case «eau»,
  - soit `'C'` pour représenter la position initiale du robot.

Observez l'exemple de fichier de terrain qui se trouve dans votre répertoire.

On propose une représentation des terrains par des variables de type `Terrain`, défini dans `terrain.h` :

```
#ifndef _TERRAIN_H_
#define _TERRAIN_H_

typedef enum { LIBRE, EAU, ROCHER } Case;

#define DIM_MAX 256

// indexation utilisée :
//  1er  indice : abscisse = colonne (colonne de gauche : abscisse = 0)
//  2ème indice : ordonnée = ligne   (ligne du haut     : ordonnée = 0)

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

```
} Terrain;

void lire_terrain(FILE *f, Terrain *t, int *x, int *y);

void afficher_terrain(Terrain *t);

#endif
```

▶ **Exercice 1.**

1. Complétez le corps du paquetage ***terrain***, c'est à dire le fichier `terrain.c`.

   Vous supposerez dans un premier temps que tout se passe normalement (le fichier existe, le nombre de cases est bien celui spécifié ...).

2. Utilisez la procédure `afficher_terrain` pour vérifier que la lecture s'est bien passée : utilisez le programme de test (`test_terrain.c`) qui lit un terrain dans un fichier, puis l'affiche à l'écran.

3. Testez les fonctions de lecture et d'affichage de terrains.

Gestion des erreurs

Différents problèmes peuvent se rencontrer lors de la lecture d'un fichier de terrains :

- le fichier dont le nom est passé en paramètre ne peut être ouvert (n'existe pas, n'a pas les permissions requises,...),
- la largeur ou la hauteur ne sont pas dans l'ensemble de valeurs possibles d'après la définition du type `terrain`,
- la longueur d'une ligne n'est pas égale à la largeur (trop courte ou trop longue),
- le nombre de lignes n'est pas égal à la hauteur (trop petit ou trop grand).
- ...

▶ **Exercice 2.**

En vous inspirant de ce qui a été vu en TD :

1. Modifiez votre fonction de lecture d'un terrain pour *retourner une erreur*, parmi une liste de valeurs à déclarer dans `terrain.h`. N'oubliez pas de tester en créant des fichiers-terrains *incorrects*.
2. Modifiez le programme `test_terrain.c` de façon à ce que, lorsque l'ouverture du fichier se passe mal, c'est à dire lorsque l'erreur `ERREUR_FICHIER` est retournée, un nouveau nom de fichier soit demandé à l'utilisateur, et une nouvelle lecture de terrain tentée avec ce nom de fichier.

Testez le programme ainsi modifié.

NB : On peut se servir des mécanismes de gestion d'erreurs des fonctions `fopen`, `fscanf` et `fgets` :

- **`fopen`** renvoie `NULL` en cas d'erreur

- **`fscanf`** renvoie le nombre de valeurs lues, et la valeur `EOF` si la fin du fichier est atteinte avant la première valeur lue
- **`fgets`** renvoie le pointeur vers la chaîne lue en cas de succès, et la valeur `NULL` en cas d'erreur ou si la fin du fichier est atteinte avant d'avoir pu lire un caractère.

Le robot

Le robot est défini par sa position dans le terrain, et son orientation :

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

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

On modifie l'état du robot en le faisant *avancer*, *tourner_a_gauche* et *tourner_a_droite*.

▶ Exercice 3.

- Complétez le corps du paquetage **robot**.
- Complétez la procédure de test `test_robot` du fichier `test_robot.c` : dans le corps de la boucle, pour :
  1. afficher des informations du robot,
  2. lire au clavier un caractère (par exemple parmi `'A'`, `'G'`, `'D'` pour *avancer*, *tourner à gauche*, *tourner à droite*), et selon le caractère, modifier l'état du robot en conséquence.

Le robot dans le terrain

▶ Exercice 4.

Lisez le fichier `robot_terrain.c`. Complétez la fonction `robot_peut_avancer` : le robot peut avancer si la case devant lui est libre ou s'il est au bord du terrain.

Testez le programme modifié.

3. Compte-rendu

1. Créez deux répertoires `Terrains_Corrects` et `Terrains_Incorrects` et placez-y les terrains testés lors des exercices 2 et 3.
2. Créez un répertoire nommé `TP6` contenant vos fichiers source, le fichier Makefile, ainsi que les deux répertoires `Terrains_Corrects` et `Terrains_Incorrects`
3. Compactez le répertoire `TP6` :

```
tar czvf TP6.tar.gz TP6
```

Déposez le fichier `TP6.tar.gz` sur la page moodle du cours.