INF304 — TP7
Curiosity Revolutions (2)
Tests de programmes robots

Ce TP est la suite du TP précédent ; placez-vous dans le répertoire du TP6, et copiez-y les fichiers nécessaires au TP7 :

```
cp -r /Public/304_INF_Public/TP7/* .
```

NB : vous avez normalement complété au TP6 les fichiers robot.c et terrain.c. Si vous n'avez pas eu le temps de finir, vous pouvez récupérer une version corrigée de ces deux fichiers dans le répertoire /Public/304_INF_Public/Correction-TP6.

Un programme simple d'interprétation d'un programme-robot

Le programme fourni curiosity.c prend en argument un nom de fichier terrain, et un nom de fichier programme, et interprète simplement le programme sur le terrain.

Lisez ce programme, et compilez-le à l'aide du Makefile fourni.

Exercice 1.

Testez ce programme sur les exemples de terrains et de programmes fournis, puis sur plusieurs terrains et programmes de votre cru.

Test fonctionnel de l'interprète

On souhaite maintenant tester de manière intensive le paquetage interprete.

Exercice 2.

Écrivez un programme de test, nommé curiosity-test.c, prenant en argument le nom d'un fichier de test dont le format est le suivant :

- sur une ligne, un nom de fichier terrain ;
- sur une ligne, un nom de fichier programme ;
- sur une ligne, le nombre de pas maximum d'exécution ;
- sur une ligne, un caractère e indiquant l'évènement attendu à la fin de l'exécution :
  - «N» si le robot est sur une position normale à l'intérieur du terrain,
  - «F» si le programme est terminé,
  - «S» si le robot est sorti du terrain,
  - «O» si le robot a rencontré un obstacle,
  - «P» si le robot est tombé dans l'eau ;

si e=N ou e=F :

- deux entiers x et y, la position attendue du robot à la fin de l'exécution ;
- un caractère o parmi {N,S,E,O}, l'orientation attendue du robot à la fin de l'exécution.

Ce programme de test doit initialiser l'environnement en lisant les fichiers de terrain et de programme, puis interpréter le programme jusqu'à l'événement de fin ou le nombre de pas maximum atteint. Il doit ensuite vérifier que l'événement obtenu et éventuellement la position du robot correspondent à ce qui est attendu et spécifié dans le fichier de test lu, et afficher un diagnostic simple («Ok» si tout va bien, une ou quelques lignes d'explications sinon).

NB : vous pouvez vous inspirer de la structure du programme curiosity.c.

Compilez et essayez votre programme avec quelques premiers tests.

Exercice 3.

Écrivez un jeu complet de tests permettant de tester l'interprète. Tester soigneusement chaque cas (chaque commande du langage, plusieurs enchaînements de commandes, des programmes plus compliqués, chaque type d'événement de fin).

Exercice 4.

Les fichiers interpretei.c, avec i = 0,...,9, représentent 10 implémentations légèrement différentes de l'interprète. Parmi ces 10 implémentations, certaines comportent des erreurs.

Le Makefile peut être utilisé pour produire un exécutable curiosity-testi, à partir du programme curiosity-test.c et de l'implémentation interpretei.c, en tapant simplement (pour i = 0) :

```
make curiosity-test0
```

Utilisez votre jeu de test pour identifier les implémentations correctes.

NB : vous pouvez éventuellement modifier votre programme de test, afin qu'il prenne en argument une liste de nom de fichiers de test, et qu'il affiche pour chaque test si le résultat est correct ou non.

Indiquez sur la page Moodle quelles implémentations sont selon vous correctes.

Tests de robustesse de l'interprète

Exercice 5.

Écrire des exemples de programmes déclenchant une erreur :

- à la lecture du programme ;
- à l'exécution du programme.

Vérifiez que ces erreurs sont correctement gérées.

Exercice 6. (bonus)

Du fait de l'implémentation, certaines erreurs dans les programmes-robot ne sont pas gérées par le paquetage interprete.

Autrement dit, on peut écrire des programmes incorrects qui seront lus et interprétés sans retourner d'erreur parmi celles déclarées.

Pouvez-vous identifier ces erreurs ? Écrivez un exemple de programme contenant une telle erreur.

3. Compte-rendu

Créez un répertoire Tests_TP7 et placez-y l'ensemble des fichiers tests créés lors de l'exercice 3.

Compactez le répertoire Tests_TP7 :

```
tar czvf Tests_TP7.tar.gz Tests_TP7
```

Déposez le fichier Tests_TP7.tar.gz sur la page moodle du cours.