TP Projet de Langage C
`Snake`

François Goasdoué
ENSSAT – Université de Rennes

Le jeu `Snake`

`Snake` est un célèbre jeu où un serpent, Snake, doit aller à la recherche de nourriture au sein d'un enclos. À mesure qu'il mange, il grandit et occupe ainsi de plus en plus de place dans son enclos. La règle du jeu consiste à déplacer Snake vers sa nourriture sans passer sur un obstacle : la clôture de l'enclos ou le corps de Snake. La difficulté augmente donc plus Snake se nourrit. La partie est gagnée si Snake, à force de grandir tout en respectant la règle du jeu, occupe tout l'enclos.

Vous devez concevoir et programmer l'Intelligence Artificielle de Snake, c'est-à-dire sa stratégie pour arriver à gagner la partie, ou au moins maximiser le score (nombre de fois où Snake s'est nourri avant de perdre).

Un niveau de jeu est codé de la façon suivante en ASCII :

```
********************
*..................*
*..............+..*
*..............#..*
*..............#..*
*..............#..*
*...........#####..*
*.$....@#####......*
*..................*
********************
```

où

- '*' est la clôture de l'enclos
- '.' est la surface libre de l'enclos
- '@' est la tête de Snake
- '#' est le corps de Snake
- '+' est la queue de Snake
- '$' est de la nourriture

À chaque tour de jeu, Snake doit prendre la décision d'aller soit au nord (haut), soit à l'est (droite), soit au sud (bas), soit à l'ouest (gauche) ; il ne peut pas aller en diagonale. Pour cela, le contrôleur de jeu vous demande uniquement la prochaine action à effectuer grâce à la fonction `snake` que vous devez développer :

```
action snake(
   char * * map, // dynamic array of chars modelling the game map
   int mapxsize, // x size of the map
   int mapysize, // y size of the map
   snake_list s, // snake
   action last_action // last action made, -1 in the beginning
   ) {...}
```

où l'action demandée est codée par

```
/*
  Enumeration for the different actions that the player may choose
 */
enum actions {NORTH, EAST, SOUTH, WEST};
typedef enum actions action;
```

et où Snake est modélisé par la liste chaînée

```
struct snake_link { // models a piece of Snake
   char c; // SNAKE_HEAD, SNAKE_BODY, or SNAKE_TAIL char
   int x; // x position for this piece of Snake within the map
   int y; // y position for this piece of Snake within the map
   struct snake_link * next; // pointer to the next piece of Snake
   };
typedef struct snake_link * snake_list;
```

En sortie de fonction, vous devez retourner l'action choisie parmi les quatre possibilités : NORTH, EAST, SOUTH, WEST. Attention, vous n'avez pas le droit d'aller sur la clôture ou sur une partie du corps de Snake, sinon vous perdez immédiatement !

Votre objectif est de concevoir l'Intelligence Artificielle de Snake afin qu'il gagne le jeu, ou à défaut qu'il fasse le meilleur score possible avant de perdre !

Travail à réaliser <u>en monôme</u>

Récupérer l'archive au format ZIP de travail sur moodle, en bas de la page du cours de *Langage C* dans la section TP projet. Cette archive contient :

1. `snake_def.h, snake_dec.h et player.h` : les fichiers d'entêtes nécessaires à la compilation du jeu. CES FICHIERS NE DOIVENT PAS ÊTRE MODIFIÉS.
2. `snake-architecture-os.o` : le moteur précompilé du jeu Snake, où architecture peut être arm ou intel et où os peut être linux ou macos. Pour les machines de TP ENSSAT, il faut utiliser snake-intel-linux.o.
3. `player.c` : le joueur aléatoire que vous devez modifier pour faire votre propre joueur. Pour cela, vous devez modifier la fonction `snake`. Celle-ci pourra faire appel à autant de sous-modules que nécessaires, ceux-ci devant être développés UNIQUEMENT dans le *seul et unique fichier* `player.c`. Chaque module développé devra être **impérativement** commenté de la façon suivante : (i) avant chaque module, la stratégie de résolution du problème dévolu au module doit être décrite (càd. l'idée générale de ce que fait le module et comment il le fait), puis, (ii) dans le code du module, la mise en oeuvre de la stratégie de résolution doit être expliquée (càd. les détails du codage de la stratégie utilisée sont expliqués au fur et à mesure du code). ATTENTION !!! Votre code ne doit utiliser AUCUNE nouvelle variable globale, fichier sur disque ou autre moyen visant à stocker de l'information entre deux appels à la fonction `snake` : c'est interdit ! De la même façon, vous n'avez pas le droit d'utiliser du code issue d'autres bibliothèques que celles du compilateur C utilisé (gcc ou clang). L'intégralité du code ne devra être écrit que par vous-même !
4. `level-CxL.map` : niveaux du jeu Snake pour un enclos de C colonnes par L lignes.

La commande de compilation pour fabriquer le jeu snake à partir des fichiers fournis est :

```
gcc -std=c99 -Wall -o snake snake-architecture-os.o player.c
```

Commencez par compiler le programme avec le joueur « Random » fourni dans le fichier player.c et lancez `./snake` sans niveau de jeu. Ainsi, vous découvrirez les options du programme et pourrez vous familiariser avec : testez les !

Évaluation du travail réalisé

Lors de la dernière séance de TP, vous devrez présenter votre solution à votre chargé de TP et répondre à ses questions durant 5 minutes.

Votre appréciation de projet servira à moduler votre note de DS entre -3 et +3 points (un projet standard aura 0). Attention, la note de projet sera reportée en seconde session pour ceux qui passent le rattrapage !

L'évaluation se déroulera de la façon suivante :

1. Votre solution devra être compilée et exécutée devant votre chargé de TP.
2. Pour chaque question posée, il s'agira de montrer le code C correspondant et de l'expliquer.

Le fichier `player.c` devra être parfaitement indenté, commenté, etc. Il devra aussi compiler sans erreur ni warning.

À l'issue de l'évaluation, durant la dernière séance de TP, votre fichier C devra être déposé sur moodle dans l'emplacement prévu pour votre groupe de TP. **Pour cela, il devra être préalablement renommé en : nom-prenom.c**. Le non respect de cette consigne entraînera une évaluation à -3 points, peu importe la note donnée par le chargé de TP.

Enfin, votre code sera passé au détecteur de plagiat afin de le comparer à ceux de la promotion. Un plagiat confirmé par les enseignants du Pôle Informatique mènera à une sanction de -3 points sur la note de DS en 1e et 2e session, ainsi qu'à la saisie des directions des Études et de l'ENSSAT.

Bon travail !