Cours de programmation séquentielle

Le nombre secret

1 Buts

- Introduction au langage C
- Utilisation des boucles et des tests
- Utilisation des entrées-sorties
- Utilisation de nombres aléatoires

2 Énoncé

Écrire un programme permettant de jouer au nombre secret. Le but est de faire chercher à l'utilisateur · trice un nombre déterminé aléatoirement par l'ordinateur. Le nombre de coups mis pour trouver la solution sera compté et affiché en fin de la partie. La borne maximum pour le choix du nombre caché sera demandée à l'utilisateur · trice.

Pour générer des nombres aléatoires, utiliser la fonction `rand()` qui retourne un nombre aléatoire entre 0 et l'entier maximum (cf. la commande `man 3 rand`). Pour initialiser le générateur denombre aléatoire, appeler la fonction `srand()` qui prend en paramètre un entier non-signé(la graine). On peut par exemple passer `time(NULL)` en paramètre à `srand()`. L'appel `time(NULL)` retourne l'horloge système en secondes.

Il est nécessaire d'inclure les bibliothèques suivantes dans le programme:

```
#include <stdio.h>      /* printf(), scanf() */
#include <stdlib.h>     /* srand(), rand() */
#include <time.h>       /* time() */
#include <math.h>       /* log() */
```

A la compilation il faut également ajouter le flag `-lm` pour générer correctement l'exécutable.

3 Déroulement du programme

- Au début du programme, l'utilisateur · trice entre le nombre maximum avec lequel il veut jouer.
- Le programme générera un nombre pris au hasard entre 0 et ce maximum.
- Dans une boucle, le programme demandera à l'utilisateur · trice d'entrer un nombre, puis lui répondra par plus petit ou plus grand et ceci jusqu'à la découverte du nombre secret.

- Afin de s'assurer que l'utilisateur · trice joue selon les règles, le programme doit vérifier que le nombre entré par l'utilisateur · trice n'est pas plus petit que zéro ou plus grand que le nombre maximal autorisé. Si un de ces deux cas se présente, le programme afficher un message d'avertissement à l'utilisateur · trice[1] afin de lui rappeler les règles.
- Chercher le moyen de trouver le nombre caché de manière optimale en moyenne.
- Afficher en fin de partie le nombre de coups mis par le joueur pour découvrir le nombre secret, ainsi que le nombre de coups optimal maximal théorique permettant de déterminer le nombre secret.

4 Exercice supplémentaire

Une fois ce qui précède terminé, échangez les rôles!

Écrire un programme où l'ordinateur et l'utilisateur · trice échangent leurs rôles. Afin de signifier à l'ordinateur que le nombre est plus grand ou plus petit, l'utilisateur · trice rentre `<` ou `>` respectivement et `=` en cas de victoire de l'ordinateur (il se pourrait que le type `char` vous soit utile). Implémentez différentes stratégies pour l'ordinateur (recherche exhaustive, bisection, etc.).

L'ordinateur doit pouvoir détecter si vous avez triché!

5 Remarques

5.1 Générateur de nombre aléatoire

Afin d'avoir des aléatoires différents à chaque excéuction, il faut utiliser la fonction `srand()` qui vous permet de fixer la *graine* du générateur de nombres pseu-do aléatoires `rand()`. Pour plus d'information:

```
man 3 rand
```

5.2 Boucle infinie

Il se peut que vous décidiez de vérifier les entrées utilisateurs · trices. Pour ce faire, vous utiliserez probablement la fonction `scanf()`. Lorsque les entrées sont mal formées vous pourriez voir apparaître une boucle infinie. Pour corriger ce problème, vous pouvez utiliser les lignes magiques après avoir appelé la fonction `scanf()`:

```
int c;
while ((c = getchar()) != '\n' && c != EOF) { }
```

voir le lien suivant pour plus d'informations.

[1] Par exemple: "Vous devez entrer un nombre plus grand ou égal à zéro." ou "Vous devez entrer un nombre plus petit que 'max'.".