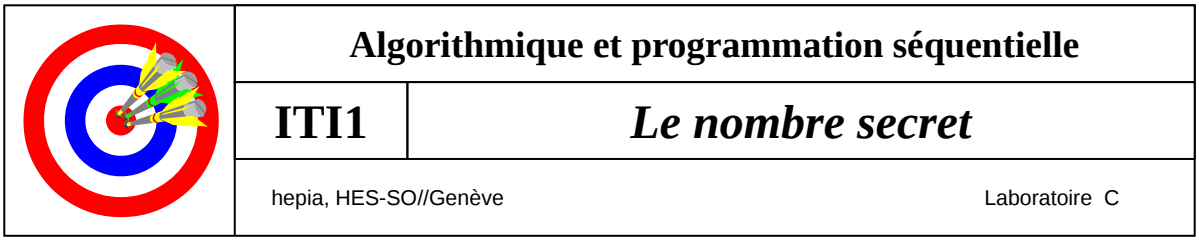

Algorithmique et programmation séquentielle	
ITI1	***Le nombre secret***
hepia, HES-SO//Genève	Laboratoire C

Buts

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

Partie 1

Enoncé

Ecrire un programme permettant de jouer au nombre secret. Le but est de faire chercher à l'utilisateur 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.

Pour générer des nombres aléatoires, utiliser la fonction `rand` qui retourne un nombre aléatoire entre 0 et l'entier maximum (cf. `man 3 rand`). Pour initialiser le générateur de nombre 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() */
```

Déroulement du programme

Au début du programme, l'utilisateur 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 d'entrer un nombre, puis lui répondra par plus petit ou plus grand et ceci jusqu'à la découverte du bon nombre.
Chercher le moyen de trouver le nombre caché de manière optimale.
Afficher en fin de partie le nombre de coups mis par le joueur pour découvrir le nombre caché ainsi que le nombre de coups minimum théorique permettant de déterminer le nombre.

Partie 2

Enoncé

Deux programmes binaires compilés, `sqrt` et `calc`, vous sont fournis sur la page Cyberlearn du travail pratique, mais sans les codes sources. Exécutez chaque programme afin d'en comprendre le comportement exact.

Une fois que vous avez une bonne compréhension de chaque programme, implémentez les fichiers sources manquants, `sqrt.c` et `calc.c`, afin d'obtenir deux programmes aux comportements exactement identiques aux binaires `sqrt` et `calc` fournis.

Au cas où vous utiliseriez des fonctions de la librairie mathématique `math.h` (cf. man `math.h` pour afficher les fonctions de la librairie), n'oubliez pas de spécifier `-lm` à la fin de la ligne de compilation.

Par exemple :

```
gcc -std=c11 -Wall -Wextra -fsanitize=address -fsanitize=leak
-fsanitize=undefined prog.c -o prog -lm
```

Partie 3

Enoncé

Ecrivez un makefile permettant de compiler le code que vous avez développé pour les parties 1 et 2 du TP. On désire que le makefile implémente au moins les cibles suivantes :

- **`debug`**. Cette cible doit s'occuper de compiler le nombre secret, `sqrt` et `calc` en mode « debug », c'est à dire avec les options `sanitize`, tous les *warnings* possibles et l'option `-g` qui permet ensuite de débogger le code dans un débogueur.
- **`release`**. Cette cible doit s'occuper de compiler les programmes en mode « release », c'est à dire sans `sanitize`, sans `-g` et avec l'option d'optimisation maximum `-O3`.
- **`clean`**. Cible qui supprime tous les fichiers intermédiaires (s'il y en a) et les fichiers exécutables générés.
- **`all`**. Cette cible est la cible par défaut. Celle-ci doit générer les exécutable des cibles `debug` et `release`.