SORBONNE UNIVERSITÉ

TD : Programmation en C — semaine 2

16 février 2018

Objectif(s)

- ✯ Premières bases de programmation : variables, et structures de contrôle (conditionnel, boucle).
- ✯ Gestion des entrées/sorties clavier.

Exercice 1 – Préliminaires

1. Régler l'éditeur de texte Gedit comme suit :
   — *Préférences* > *Éditeur* et cocher *Insérer des espaces au lieu des tabulations* ;
   — *Préférences* > *Éditeur* et réglez la largeur des tabulations à 4 espaces ;
   — *Préférences* > *Vue* et cocher *Afficher les numéros de lignes*
   Si vous utilisez un autre éditeur, soyez sûr de faire de même.
2. Télécharger le fichier *ready.c* sur le site des TDs, et en console, effectuer les étapes suivantes :
   — Créer un dossier avec votre nom, un sous-dossier *td2* et placer le fichier *ready.c* dans ce répertoire.
   — Vous pouvez compiler le fichier avec `gcc` et l'exécuter avec `./ready` :
   ```
   gcc ready.c -o ready
   ./ready
   ```
   Il y a un certain nombre d'options que l'on doit fournir à `gcc`, la commande complète que nous utiliserons pour tous nos programmes sera :
   ```
   gcc ready.c -o ready -Wall -ansi -pedantic -Wall -Werror -std=c99
   ```
   Pour ne pas retaper cette longue ligne à chaque fois, vous pouvez récupérer le script *run.sh*, le mettre dans le même répertoire que *ready.c* et taper :
   ```
   ./run.sh ready
   ```
   Rappel : Pour rendre un script exécutable il faut taper `chmod +x run.sh`. Si vous remplacez "ready" par un autre nom de programme, le script `run.sh` marchera tout aussi bien !
3. Lisez les deux fichiers que vous avez téléchargez. Que fait le programme `ready.c` ?

Exercice 2 – À votre tour !

Tous les programmes suivants seront stockés dans des fichiers différents, pour démarrer n'hésitez pas à vous inspirez de `ready.c`.

1. Affichez "Bonjour Monde" à l'écran.
2. Écrivez un programme qui demande deux réels à l'utilisateur et affiche lequel est le plus petit et lequel est le plus grand. Utiliser les variables de types `float v;` et le spécificateur `%f` dans `scanf("%f", &v);` ou `printf("%f", v);`.
3. Demandez à l'utilisateur la note n qu'il a obtenu au dernier examen et affichez sa mention sachant que :
   — $n < 10$ = Echec
   — $10 \leq n < 12$ = Peut mieux faire

- $12 \leq n < 14$ = Presque bien
- $14 \leq n < 17$ = Bien
- $17 \leq n$ = Votre enfant a t'il une vie ?

4. Choisissez 3 entiers. Demandez un entier à l'utilisateur. Si il est différent des 3 choisis, il gagne un point. Si il est supérieur à au moins un des 3 choisis, il gagne deux points. Sinon il ne gagne rien. Choisissez un nombre avec la fonction `rand` :

   ```
   int r = rand() % 10;
   ```

   De plus, vous devez ajouter la bibliothèque `#include <stdlib.h>` en début de fichier.
5. Demandez un entier n à l'utilisateur. Affichez tous les entiers de 1 à n (un par ligne).
6. Demandez deux entiers n et m à l'utilisateur. Affichez un rectangle $n \times m$ uniquement composé d'étoiles (*).

Exercice 3 – Un mini-jeu

1. Générez un entier aléatoire entre 0 et 1000. Demandez des entiers à l'utilisateur :
   - Si il est au dessus, dites le lui et redemandez ;
   - Si il est en dessous, dites le lui et redemandez ;
   - Si il trouve, c'est gagné !
2. Étendez la question précédente pour limiter l'utilisateur à un certain nombre d'essais (combien d'essais faut-il minimum pour devinez tout nombre entre 0 et 1000 ?)

Exercice 4 – Challenge Time !

1. Il y a trois sujets de challenges en ligne : "Cutting cost", "Division of Nlogonia" et "The Snail" (plus ou moins dans cet ordre de difficulté). Afin de résoudre ces trois sujets, quelques informations supplémentaires sur `scanf` :
   - `scanf` ignore les espaces, si vous écrivez `scanf("%d", &v)` et rentrez " 12", pas de soucis.
   - Pour récupérer deux éléments de l'utilisateur d'affilé, vous pouvez écrire `scanf("%d %d", &v1, &v2)` ou simplement faire deux `scanf` à la suite.
   - Mettez l'exemple du problème dans un fichier `input.txt`, vous pouvez tester votre programme avec :
     `./snail < input.txt`
     (Rappelez-vous, < redirige le contenu du fichier vers l'entrée standard du programme, par conséquent on peut lire le contenu du fichier avec `scanf`.)