Algorithmique et Programmation 1
IMAC 1ere année

TP 2

Entrées/sorties, variables, structures conditionnelles, types, chaines de caractères

---

Dans cette séance de travaux dirigés, les points suivants sont abordés :
— la notion de programme (en langage C) ;
— les déclarations de variables ;
— les types de base ;
— les entrées/sorties ;
— les structures conditionnelles `if` et `if`, `else` ;
— les conversions de types ;
— la manipulation basique de chaînes de caractères.

Pendant le TP, vous écrirez vos codes dans des fichiers source. Ayez le réflexe d'enregistrer régulièrement vos sources. À la fin de la séance, mettez-les dans une archive (commande `tar -zcvf nom_archive fichiers_source`) et rendez-le suivant les instructions données sur e-learning. N'oubliez pas de commenter votre code.

Le sujet est long, ne cherchez pas forcément à le finir, mais plutôt à bien comprendre les premiers exercices. Pensez à consulter les transparents de cours sur elearning.

Exercice 1. (Hello, world !)

— Créer un fichier `hello.c` et avec l'aide d'un éditeur de texte (par ex. *emacs*) composer le programme *Hello, world !* vu pendant les cours.
— Compiler avec `gcc hello.c`, puis executer le `a.out`
— Compiler avec `gcc -o monprogramme hello.c`.
— Essayer de supprimer ou ajouter quelques symboles (crochets, point-virgules,...). Compiler et essayez de comprendre les messages d'erreur obtenus.
— Compiler avec `gcc -o hello.c hello.c`. Qu'est-ce qui s'est passé ?

Exercice 2. (Comprendre un programme et lecture au clavier)

Recopier le programme qui suit dans un fichier `.c` :

```c
#include <stdio.h>
int main() {
    int nombre;

    printf("Voici un nombre : %d\n", 7);
    nombre = 11;
    printf("En voici un autre : %d\n", nombre);
    return 0;
}
```

1. Compilez le programme et exécutez-le. Qu'est-ce qui est affiché sur la sortie standard ?
2. Modifiez le fichier source pour qu'il affiche `11/2` à la place de `11`. Compilez et exécutez une deuxième fois. Quelle est la valeur de `11/2` ?
3. Modifiez votre programme pour qu'il demande à l'utilisateur de rentrer lui-même la valeur `nombre`. On rappelle que pour stocker une valeur de type `int` dans une variable entière `n`, on utilise la commande `scanf("%d", &n)`.

Exercice 3. (Manipulation de nombres)

Écrire un programme qui demande à l'utilisateur de saisir quatre entiers au clavier, affiche ensuite une valeur par ligne, puis leur somme et leur moyenne sur la ligne suivante. Par exemple, si l'utilisateur entre les valeurs `12 3 14 5`, le programme affiche

```
12
3
14
5
somme = 34, moyenne = 8.5.
```

Comment faire avec un seul `scanf` ?

Exercice 4. (Structures conditionnelles)

1. Écrire un programme qui lit un entier saisi par l'utilisateur au clavier et affiche `positif` s'il est positif et `negatif` sinon.
2. Écrire un programme qui lit un entier saisi par l'utilisateur au clavier et affiche `pair` s'il est pair et `impair` sinon.

Exercice 5. (Échange de valeurs)

1. On suppose que deux variables `a` et `b` de type `int` sont déclarées et affectées. Écrire une suite d'instructions qui permet d'échanger le contenu de `a` et de `b`.
2. En déduire une suite d'instructions qui effectue l'échange des valeurs de `a` et de `b` uniquement si `a` est inférieur à `b` et qui sinon incrémente de 10 la variable `b`. La suite d'instructions affiche dans tous les cas les valeurs de `a` et de `b`. Par exemple, si `a = 1` et `b = 2` votre programme doit afficher

```
1 a = 2 et b = 1}
```

Si `a = 2` et `b = 1`, alors votre programme doit afficher

```
1 a = 2 et b = 11
```

Exercice 6. (Mise sous forme de siècles, années, mois, jours)

1. Écrire un programme transformant un nombre de jours J entré par l'utilisateur en nombre de siècles, de mois, de semaines et de jours restant.
   Pour simplifier le problème, on considèrera que tous les mois ont 30 jours et toutes les années 360 jours.
   L'affichage se fera sous la forme :
   `J jores correspondent a:`
   `xx siecle xx annee xx mois xx semaine xx jours`
2. Si ce n'est déjà fait, appliquer les règles d'accord en nombre. ("5 annees" et non pas "5 annee")

Exercice 7. (Conversion d'un réel positif en entier)

1. Écrire un programme qui convertit un *float* positif en *int* en utilisant la règle d'arrondi classique. Par exemple, `4.4` est converti en `4` et `4.5` est arrondi en `5`.
2. Que se passe-t-il quand vous convertissez `5.499 999 99` ? Comment résoudre simplement le problème ?

Exercice 8. (Jeu sur les caractères)

1. Écrire un programme qui
   — demande à l'utilisateur de rentrer une chaîne de caractères (sans espace) ;
   — demande à l'utilisateur de rentrer une position dans cette chaîne de caractères (la numérotation de la position devra commencer à 1 pour le premier charactère) ;
   — indique si le caractère à la position indiquée existe (vous utiliserez la fonction *strlen* de la bibliothèque `<string.h>` pour calculer la longueur de la chaîne) ;
   — si ce caractère existe, indique si c'est une lettre de l'alphabet (anglais), un chiffre ou autres ;
   — si c'est une lettre, indique si elle est en majuscule ou en minuscule.
2. Modifier le programme pour qu'il y ait deux utilisateurs (un adulte et un enfant). L'adulte rentre une chaîne de caractères et une position dans cette chaîne. L'enfant devra indiquer si c'est une lettre (réponse 'l'), un chiffre (réponse 'c') ou autres (reponse 'a'). L'ordinateur affiche le message "bravo" en cas de succès. Sinon, il affichera un message d'erreur explicite à l'enfant sans donner la correction exacte.
3. (BONUS) Imaginer une extension de ce jeu.