

Fiche n° 2 de TP

Formats et entiers

Objectifs : manipulation des entrées/sorties ; expressions entières.

Prérequis : instructions `scanf` et `printf`; formats de lecture et d'affichage.

Travail minimum : exercices 1 à 4; les solutions que vous devez envisager ne doivent pas faire appel à l'instruction conditionnelle.

Échauffement.

Exercice 1

- 1) Écrivez un programme qui affiche :

```

1
 2
 3
 4
 5
 6
 7
 8
 9

```

- 2) Si nécessaire, modifiez votre programme pour que les formats d'affichage ne contiennent aucune espace (c'est à dire supprimez les espaces de vos « printf »).

Le but de ce TP est de présenter les notions d'entrées/sorties standard au travers d'un exemple permettant de gérer les conversions durée \longleftrightarrow heures:minutes:secondes.

Exercice 2

- 1) Supposons que `d` est une variable entière contenant une durée exprimée en seconde. Donnez une séquence d'instructions permettant de calculer et de stocker dans trois variables `h`, `m` et `s` la conversion de la valeur de `d` sous la forme « heures:minutes:secondes ».
- 2) Écrivez le programme `sec_to_time` qui lit une valeur horaire exprimée en secondes sur l'entrée standard et qui affiche sur la sortie standard la même valeur sous la forme « heures:minute:seconde ».
Exemple : pour l'entrée `3637` le programme affichera `1:0:37`.
- 3) Améliorer votre programme pour que chaque nombre s'affiche avec au moins 2 chiffres (en complétant par des 0 sur la gauche si besoin).
Exemple : pour l'entrée `3637` le programme affichera `01:00:37`.
- 4) Que se passe-t-il si l'utilisateur entre une valeur négative ? Est-ce que cela a un sens ? Empêchez l'utilisateur d'entrer une valeur négative en utilisant une instruction `assert`.

Exercice 3

- 1) Écrivez un programme `time_to_sec` qui, pour toute valeur horaire exprimée sous la forme « heures:minutes:secondes » lue sur l'entrée, l'affiche sur la sortie en secondes.
Exemple : pour l'entrée `1:0:37` le programme affichera `3637`.
Attention, le programme doit imposer la lecture du caractère `:` entre la saisie de chaque nombre.
- 2) Sous quelles conditions (portant sur les valeurs lues) le programme écrit un résultat cohérent sur la sortie ?
- 3) Ajouter une ou plusieurs instructions `assert` afin de sortir du programme si ces conditions ne sont pas satisfaites.

Exercice 4

- 1) Ouvrez un terminal à partir du répertoire du TP2 et tapez

```
$ ./time_to_sec  
12:34:56
```

Qu'observez-vous ?

- 2) Que produit la commande suivante ? Expliquez.

```
echo "12:34:56" | ./time_to_sec > conv_sec.txt
```

- 3) Utiliser le programme `sec_to_time` dans le terminal.

- 4) Vérifiez que le `sec_to_time` est le réciproque de `time_to_sec` en faisant lire au second la sortie du premier et en vérifiant que la valeur affichée est bien la même que celle que vous avez rentrée.

Exercice 5 (*Pour aller plus loin*)

Que font les lignes du programme suivant ?

```
tp2-scanf.c  
-----  
#include <stdio.h>  
#include <stdlib.h>  
  
int main(void) {  
    int n1, n2;  
    int rep = scanf("%d%d", &n1, &n2);  
    rep = printf("\n%d\n", rep);  
    printf("%d\n", rep);  
    rep = scanf("%*s");  
    printf("%d\n", rep);  
    return EXIT_SUCCESS;  
}
```

tp2-scanf.c -----

Que se passe-t-il si on tape 6 Ctrl+D Ctrl+D Ctrl+D lors de l'exécution du programme ?