# Séance 3 : Fonctions d'entrées/sorties

## *Introduction, fonction printf*

Le langage C définit uniquement le langage en lui-même et ne propose pas de bibliothèque. La norme ANSI propose une bibliothèque standard permettant de gérer les entrées/sorties. Nous avons déjà vu un exemple de ces fonctions, la fonction `printf`. Nous allons voir un nouvel exemple utilisant cette fonction :

```
#include <stdio.h>
#include <stdlib.h>

/* Programme de conversion des noeuds vers des km/h */

int main(void) {
     int vitesse_max     = 60;
     int vitesse_noeud   = 1;
     int vitesse_kmh;

     printf( "Conversion vitesses\n");

     while( vitesse_noeud <= vitesse_max)
     {
          vitesse_kmh    = (vitesse_noeud*1852)/1000;
          printf( "%d nds\t = %d km/h\n", vitesse_noeud,
                                vitesse_kmh);
          vitesse_noeud++;
     }

     return(EXIT_SUCCESS);
}
```

Ce programme permet la conversion entre une vitesse en noeud et une vitesse en km/h. Dans le premier argument de la fonction `printf`, une chaîne de caractères où chaque `%` indique une substitution avec les arguments qui suivent. Voici les principales possibilités de substitution :

| ***substitution*** | ***type*** |
|---|---|
| `%d` | entier |
| `%f` | flottant |
| `%g` | flottant en notation avec puissance de 10 éventuelle et sans décimales nulles à droite du nombre |
| `%c` | caractère |
| `%s` | chaîne de caractères |

Nous avons déjà vu de plus que dans une chaîne de caractères, il était possible de placer des séquences d'échappement commençant par le caractère « \ ». Le tableau suivant récapitule les principales séquences :

| *séquence* | *effet* |
|---|---|
| \n | retour à la ligne |
| \t | tabulation |
| \\ | anti-slash |
| \'<br>\" | apostrophe<br>guillemets |

## *Fonction scanf*

Nous allons voir maintenant une fonction permettant au programmeur de récupérer ce que rentre l'utilisateur dans la console. C'est la fonction `scanf`, ainsi :

```
scanf( "%d", &variable);
```

permet de récupérer une valeur entrée au clavier par l'utilisateur, ce dernier devant valider sa saisie par la touche « Entrée ».
**Bien noter la présence du caractère « `&` » devant le nom de la variable : il signifie que scanf a besoin de l'adresse de la variable.**

Les caractères suivant le % dans le premier argument de la fonction scanf suit la même logique que pour la fonction printf (éviter %g dans un scanf)

Voici maintenant un exemple plus élaboré :

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
     int vitesse_noeud;
     int vitesse_kmh;

     printf( "Entrez une vitesse en noeuds\n");
     scanf( "%d", &vitesse_noeud);

     printf( "%d nds\t = %d km/h\n", vitesse_noeud,
                         vitesse_kmh);
     }

     return(EXIT_SUCCESS);
}
```

## *Fonction getchar et putchar*

Les fonctions `getchar` et `putchar` sont aussi des fonctions de la bibliothèque d'entrées/sorties standard. Mais contrairement aux fonctions `printf` et `scanf`, elles n'opèrent que sur un caractère à la fois. Ainsi après

```
c    = getchar();
```

La variable `c` contient le caractère reçu en entrée, et

```
putchar(c);
```

affiche le contenu de la variable `c` sous la forme d'un caractère.

Voici un exemple d'utilisation :

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
     int c;

     c = getchar();
     while( c != EOF)
     {
          putchar( c);
          c = getchar();
     }

     return(EXIT_SUCCESS);
}
```

Ce programme recopie simplement l'entrée sur la sortie standard.

## *Exercice 1*

Ecrire un programme qui reprend le premier programme de cette fiche en demandant à l'utilisateur la borne maximum et la borne minimum pour lesquelles la conversion doit être réalisée.

## *Exercice 2*

Ecrire un programme qui prend le pourcentage, saisi par l'utilisateur, d'une somme saisie aussi par l'utilisateur. Dans ce programme vous utiliserez des variables flottantes et donc le caractère de substitution associé dans les fonctions `printf` et `scanf` sera le «`%f`».

## *Exercice 3*

Ecrire un programme réalisant la moyenne des notes de 10 élèves en utilisant l'instruction `while`. Compléter ensuite le programme pour qu'il calcule la moyenne des notes pour un nombre d'élèves qui sera défini par l'utilisateur.