[Math-L312] TP 1 : Premiers pas en C

Adrien Semin
adrien.semin@math.u-psud.fr

1 Rappels et explications sur les fonctions `printf` et `scanf`

Ces deux fonctions nécessitent d'importer l'header `<stdio.h>` au début de vos programmes lorsqu'ils les utilisent :

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

La fonction `printf` permet d'écrire une chaîne de caractères dans le terminal. Cette fonction particulière admet un nombre de paramètres variable, et permet d'insérer les valeurs de plusieurs types possibles (entiers, flottants...) à l'intérieur de la chaîne affichée. La substitution de chacune des valeurs est spécifiée par une séquence de caractères commençant par `%` et finissant par une lettre pour indiquer le type de conversion à réaliser. Les substitutions les plus courantes sont :

- `%i` ou `%d` : `int` décimal ;
- `%x` : `int` hexadécimal ;
- `%f` : `float` ;
- `%c` : `char` ; caractère
- `%s` : chaîne de caractères (`char *`).

Par exemple :

```
int a=2;
float pi=3.14159265;
printf("a vaut %i et pi vaut %f\n",a,pi);
```

affichera :

```
a vaut 2 et pi vaut 3.141593
```

Vous pourrez retrouver les spécifications en français de la fonction `printf` à cette adresse :
`http://www.linux-france.org/article/man-fr/man3/printf-3.html` ou en entrant la commande :

```
> man 3 printf
```

dans un terminal.

La fonction `scanf` permet d'effectuer l'opération inverse : lire des valeurs depuis l'entrée standard. **Attention** : son résultat est un `int` qui vaut le nombre de variables lues avec succès, et non leur valeur. Ceci permet au programme de s'assurer que l'utilisateur a tapé une suite de caractères qui respecte bien le format demandé. Par exemple :

```
int a,b;
printf("Veuillez entrer un entier: ");
b=scanf("%i",&a);
assert(b==1);[1]
```

affichera :

```
Veuillez entrer un entier:
```

et attendra que vous tapiez un entier suivi de la touche `<Enter>`. Cet entier sera alors stocké dans la variable `a`. Si vous tapez autre chose qu'un entier (par exemple une suite de lettres), alors l'assertion `b==1` ne sera pas vérifiée et vous obtiendrez un message d'erreur. Vous pourrez retrouver les spécifications de la fonction `scanf` à l'adresse : `http://www.linux-france.org/article/man-fr/man3/scanf-3.html`.

[1] La fonction `assert` nécessite l'importation de l'header standard `<assert.h>`

2 Premiers programmes simples

On demande les programmes suivants :

1. `Calcul.c` ∗ ***À Rendre!*** ∗ : le programme `Calcul` affiche la valeur de $365 \times 24 \times 3600$;
2. `ConvertToHex.c` ∗ ***À Rendre!*** ∗ : le programme `ConvertToHex` demande à l'utilisateur d'entrer un entier, puis affiche cet entier en base hexadécimale ;
3. `RaiseToSqr` ∗ ***À Rendre!*** ∗ : le programme `RaiseToSqr` demande à l'utilisateur d'entrer un entier, puis affiche le carré de cet entier ;
4. `ExtractSqrt` ∗ ***À Rendre!*** ∗ : le programme `ExtractSqrt` demande à l'utilisateur d'entrer un nombre flottant, puis affiche la racine carrée de ce nombre.
   On vérifiera d'abord que le nombre entré est valide et positif avec `assert`, et on pourra utiliser la fonction `sqrt` définie dans `<math.h>`, qui renvoie la racine carrée d'un nombre flottant. Cette dernière fonction nécessite de lier le programme avec la librairie mathématique lors de la compilation avec `gcc`, en utilisant l'option `-lm`. Par exemple :

   ```
   > gcc ExtractSqrt.c -lm -o ExtractSqrt
   ```

5. `Dice.c` ∗ ***À Rendre!*** ∗ : le programme `Dice` affiche un nombre aléatoire compris entre 1 et 6. Le code suivant permet de générer un tel nombre — en utilisant des fonctions définies dans les librairies standards `<stdlib.h>` et `<time.h>` :

   ```
   int Nombre;
   srand(time(NULL)); /* On initialise le générateur de nombres aléatoires
       .*/
   Nombre=rand() % 6; /* On prend le résultat de rand() modulo 6: le
       nombre est compris entre 0 et 5*/
   Nombre++;          /* On incrémente: le nombre est compris entre 1 et 6
       */
   ```

6. `Moyenne.c` ∗ ***À Rendre!*** ∗ : le programme `Moyenne` demande un nombre de notes `n` à l'utilisateur. Puis, il demande `n` valeurs flottantes, en calcule la moyenne et l'affiche. Pour répéter `n` fois une série d'instructions, on utilisera en C la séquence suivante :

   ```
   int i,n; /* Déclaration des variables utilisées dans la boucle */

   /* ... */

   for (i=1;i<=n;i++){
     /* Tout ce qui est ici sera répété n fois, pour i variant de 1 à n */
   }
   ```