# TD 1 — Éléments de programmation en C

## Préambule

Un programme en langage C est donné sous forme de texte, le code source, et il doit passer par une étape de traduction avant de pouvoir être exécuté par le processeur. Cette traduction s'appelle la *compilation*.

Si le code source contient des erreurs de syntaxe, la compilation échoue et le compilateur indiquera les erreurs qu'il a rencontrées.

**Travail à rendre**

Les exercices 1, 2 et 8 sont à faire *sans utiliser l'ordinateur*. Ils ne sont pas à rendre.

Pour les autres exercices, vous créerez un projet sur `Code::Blocks` pour chaque exercice. Vous devrez déposer sur EPREL une archive nommée `R101_TD1_<nomDeFamille>.zip`. Cette archive contiendra un dossier par exercice, avec à l'intérieur le contenu de chaque projet `Code::Blocks`.

## 1 Déclaration et affectation de variables

**Exercice 1** (Un premier programme `C`)**.** On considère le code source suivant :

```c
#include <stdlib.h> // Permet d'avoir la valeur de retour EXIT_SUCCESS

/* Fonction main
   Porte d'entrée du programme. Tous les programmes en C
   commencent par la fonction main
*/
int main(void) {
    /* Déclaration et initialisation des variables */
    int x = 5;
    int y;

    y = 2;
    x = y;

    return EXIT_SUCCESS; // Valeur de retour de la fonction main
}
```

Les textes contenus `/* entre ces symboles */` ou `// après deux slash` sont des commentaires. Ils sont ignorés lors de la compilation, et permettent d'apporter des explications sur ce que fait le code source. Savoir commenter correctement un code source est une compétence *essentielle* de tout·e programmeur ou programmeuse.

1) Expliquer ce que fait chaque ligne du code source.

2) On observe que les lignes 8 à 15 sont décalées vers la droite. Cela s'appelle *l'indentation*. À quoi cela sert-il ?

3) À la ligne 12, on remplace l'instruction `y = 2;` par l'instruction `y = x + 2;`. Quelles sont les valeurs prises par les variables à chaque étape de l'exécution ?

**Exercice 2** (Indentation)**.** Indenter correctement le code source suivant :

```c
#include <stdlib.h> int main (
) { int x ; int y ; int z ; x=10;
y = 3 0 ; z = x + y ; y = y / 3 ;
x = x * x ; return
EXIT_SUCCESS; }
```

**Exercice 3** (La fonction `printf`)**.** On considère le programme suivant :

```c
#include <stdlib.h> /* pour EXIT_SUCCESS */
#include <stdio.h> /* pour la fonction printf */
/* fonction principale */
int main() {
    int x; // Déclaration de la variable x, sans initinalisation
    x = 3; // Initialisation de x
    x = x + 1;

    printf("x = %d\n", x); // Que fait cette instruction ?
    return EXIT_SUCCESS;
}
```

1) Que fait ce programme ?

2) En particulier, quel est le rôle de la ligne 9 ? Que signifie le caractère `'\n'` ?

3) Donner la trace du programme C. Pour cela vous utiliserez un tableau comportant une colonne pour le numéro de ligne, autant de colonnes que de variables utilisées dans le programme et une colonne pour les affichages éventuels du programme sur la console.

**Exercice 4** (Échange de deux variables)**.** Écrire un programme en C qui crée deux variables entières `x` et `y`. Initialiser `x` à 17 et `y` à 42. Votre programme devra échanger les valeurs des deux variables. Vous afficherez les valeurs avant et après l'échange pour vérifier votre résultat.

## 2 Lecture et affichage de variables

La fonction `printf` utilise des indications de formats avec le symbole `%`, comme `%d` indiquant le format pour afficher un nombre entier. Il existe d'autres formats reconnus par `printf` comme le format `%x`, pour afficher des nombres entier en hexadécimal. Le tableau suivant indique divers formats que vous pourrez tester dans un programme :

| Formats | Sorties | Exemples |
|---|---|---|
| %d ou %i | Entier decimal signe | 392 |
| %u | Entier decimal non signe | 7235 |
| %o | Octal non signe | 610 |
| %x | Entier hexadecimal non signe | 7fa |
| %X | Entier hexadecimal non signe | 7FA |
| %f | Decimal virgule flottante, minuscules | 392.65 |
| %F | Decimal virgule flottante, majuscules | 392.65 |
| %e | Notation scientifique (mantisse/exposant min) | 3.9265e+2 |
| %E | Notation scientifique (mantisse/exposant maj) | 3.9265E+2 |
| %g | Utilise la representation la + courte: %e ou %f | 392.65 |
| %G | Utilise la representation la + courte: %E ou %F | 392.65 |
| %c | Caractere (char) | a |
| %s | Chaine de caracteres | bonjour |
| %p | Adresse (pointeur) | b8000000 |

Notez que pour les flottants (type `float`), on peut indiquer le nombre de caractères figurant avant et après la virgule, comme avec `%3.1f` pour afficher `392.6` au lieu de `392.65`.

La fonction qui permet de lire des caractères du clavier pour initialiser une variable s'appelle `scanf`. Elle prend (comme `printf`) en premier argument une chaine de caractères contenant des indications de format, et en second argument l'adresse en mémoire d'une variable. Pour connaitre l'adresse d'une variable, on lui applique l'opérateur `&` (se lit « et commercial » ou « esperluette »).

**Exercice 5.** Les lignes de code suivantes permettent de lire une variable, initialisée d'abord à `0`, à partir des caractères entrés au clavier par l'utilisateur·rice du programme :

```c
1 int valeurSaisie = 0;
2 scanf("%d", &valeurSaisie); // Ne pas oublier & pour l'adresse de la variable */
3 printf("%d", valeurSaisie);
```

**1)** Intégrer l'extrait de code ci-dessous dans un programme C qui compile.

**2)** Que se passe-t-il si l'on retire l'esperluette à la ligne 2 ?

**3)** Modifier la ligne 3 pour afficher l'adresse de la variable `valeurSaisie`.

**Attention avec la fonction `scanf`**

La fonction `scanf` est une fonction riche et complexe. Il est par exemple possible de préciser le nombre de caractères qui doivent être lus dans l'indicateur de format :

```c
1 int nbDeCinqChiffres = 0;
2 scanf("%5d", &nbDeCinqChiffres); // Notez la présence du nombre 5
3 print("nbDeCinqChiffres = %d\n", nbDeCinqChiffres);
```

Si l'utilisateur ou utilisatrice saisit plus de caractères que prévu, ils seront ignorés *mais seront conservés dans la mémoire* (appelée *tampon*) et seront lus automatiquement au prochain appel de la fonction `scanf`.

**Exercice 6.** **1)** Écrire un programme qui lit une variable de type `unsigned long int` avec la fonction `scanf` et l'imprime ensuite dans les formats `%u`, `%o` et `%X`.

**2)** Tester le résultat obtenu avec d'autres formats.

**Exercice 7.** Écrire un programme qui demande la saisie d'un nombre (pas nécessairement entier, pas nécessairement positif), puis calcule et affiche son carré et son cube.

# 3 Un peu de logique booléenne

En programmation, il est fréquent de devoir manipuler des quantités qui ne peuvent prendre que deux valeurs : `true` et `false`. Ce sont des valeurs *booléennes*.

Le langage C ne propose pas de type booléen par défaut, donc on adopte la convention suivante : lorsqu'une variable ou expression prend la valeur `0`, elle est considérée comme `false`. Toute autre valeur correspond à `true`.

**Exercice 8.** On considère le programme suivant :

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

int main(void) {
    int beau_temps = 1; //  1 : true
    int pas_de_vent = 0; // 0 : false

    printf("%d\n",beau_temps && pas_de_vent);
    printf("%d\n",beau_temps || pas_de_vent);
    printf("%d\n", !beau_temps || pas_de_vent);
    printf("%d\n",! (! beau_temps || pas_de_vent));

    return EXIT_SUCCESS;
}
```

Qu'affiche ce programme ?

**Exercice 9.** On considère une variable `x` initialisée à `-230` et une variable `y` initialisée à `0`.

**1)** Quelle est la valeur (`true` ou `false`) des expressions logiques données page 5 ?

**2)** Écrire un programme qui vérifie vos calculs.

```c
(x>0)
(x<=0)
(!(x>=0))
(!(y>=0))
(!(x))
(x==1)
(y==0)
((x>0) || (x<0))
((x<0) && (x>0))
((x<0) && ((x>0) || (y<=0)))
(((x<0) && ((x>0)) || (y<=0)))
(!(x==0))
(x!=0)
((x<-100) || (y<0))
((x<-100) && (y<0))
!y
```

**À retenir**

- Vos codes sources doivent être correctement indentés pour être lisibles et commentés pour être compréhensibles.
- Les variables en C sont *typées* et il faut connaitre les types principaux.
- En C, toutes les instructions se terminent par un point-virgule.
- On utilise la fonction `printf` pour afficher la valeur d'une variable dans la console et la fonction `scanf` pour lire une la valeur d'une variable saisie au clavier.