



R1.08 Informatique 2022 – 2023	TD n°0 Bases du C	 
-----------------------------------	----------------------	---

Notions de base en langage C

Pour tous les exercices du TD (mais également des TD suivants) où vous devez travailler sur un programme à récupérer sur Moodle, vous devrez au préalable créer sur le bureau de l'ordinateur un dossier s'intitulant TD1_VotreNom et télécharger les fichiers dans ce dossier. Vous pourrez ensuite les modifier et les compiler.

A la fin de la séance, vous supprimerez le dossier dans lequel vous avez travaillé.

Exercice 1. Compilation d'un programme

Le langage C est un langage compilé : le fichier source (extension .c) doit au préalable être converti en langage machine (extension .exe) avant de pouvoir être exécuté. Lors de cette étape, le logiciel qui effectue cette traduction (le compilateur) s'arrête dès qu'une erreur est trouvée. Le programme exécutable n'est alors pas généré.

Il est nécessaire de recompiler le programme dès qu'on le modifie pour que la modification apportée se retrouve dans le programme exécutable.

- ✓ Récupérez le programme td1exo8.c sur Moodle ;
- ✓ Ouvrez-le avec le logiciel Code::Blocks ;
- ✓ Compilez-le et lisez le message d'erreur qui apparaît dans la fenêtre du bas ;
- ✓ Corrigez l'erreur, recompilez le programme puis exécutez-le.

Exercice 2. Programmation séquentielle

En langage C, comme dans tout langage de programmation, les instructions sont séquentielles : elles sont exécutées les unes après les autres dans l'ordre où elles apparaissent dans le fichier source.

Toutes les instructions doivent se trouver dans le programme principal (nous verrons plus tard le cas des fonctions et des sous-programmes).

Les seules exceptions sont les instructions de précompilation (repérées par le symbole '#') qui doivent obligatoirement se situer au tout début du fichier source.

De plus, à l'intérieur du programme principal, les instructions de déclaration des variables se situent « obligatoirement » en premier.

On souhaite écrire un programme qui permet à l'utilisateur de rentrer un nombre entier, puis de calculer et afficher son carré.

- ✓ Récupérez le programme td1exo9.c sur Moodle
- ✓ Placez les instructions dans le bon ordre, **sans en modifier aucune.**
- ✓ Compilez le programme et vérifiez sa bonne exécution
- ✓ Modifiez le programme pour que l'utilisateur puisse entrer un nombre réel codé en double précision (le résultat sera également affiché comme un nombre réel)

Exercice 3. Affectation et opérateurs arithmétiques

Les opérateurs arithmétiques classiques sont directement utilisables en C.

Le résultat de l'opération est une valeur du même type que celles des 2 opérandes (nous verrons dans l'exercice 13 le cas où les 2 opérandes sont de types différents).

Il est à noter que, pour les entiers, il existe 2 opérateurs relatifs à la division : '/' qui permet d'obtenir le quotient et '%' (appelé modulo) qui permet d'obtenir le reste.

La priorité des opérateurs est la même qu'en mathématiques, et dans le cas où plusieurs opérations ayant la même priorité doivent être exécutées dans la même instruction, elles sont exécutées de la gauche vers la droite.

- ✓ Récupérez le programme tdlxo10.c sur Moodle. **NE PAS LE COMPILER TOUT DE SUITE.**
- ✓ Essayez de deviner ce qui va être affiché à l'écran
- ✓ Compilez et exécutez le programme pour comparer avec ce que vous aviez prédit à la question précédente.

Exercice 4. Fonction printf

La fonction `printf` est une fonction qui permet d'afficher un message à l'écran.

Son utilisation la plus basique, telle que vue à l'exercice 8 est de recopier directement à l'écran la chaîne de caractères donnée en entrée de la fonction (donc à l'intérieur des parenthèses).

En C, une chaîne de caractères est un texte délimité par une paire de guillemets (").

On voit dans l'exercice 8 une première exception à cette règle, qui sont les lettres précédées du caractère « anti-slash » ('\ ou AltGr-8 sur le clavier). Il s'agit de caractères spéciaux qui ne s'afficheront pas de la même façon lors de l'exécution du programme que lors de son écriture. Le caractère '\n' permet de retourner à la ligne dans l'affichage.

Il est également possible d'insérer des « variables » à l'intérieur de la chaîne de caractère à afficher. Pour cela, on place un marqueur dans la chaîne à l'endroit où on veut afficher la valeur et on précise la valeur à afficher. Le marqueur est constitué du caractère pourcent (%) suivi d'une lettre spécifiant le type de valeur à afficher : %c pour un caractère, %d pour un entier et %f pour un réel.

Son écriture est la suivante :

```
printf("<format>",<Expr1>,<Expr2>, ... )
```

avec la partie "<format>" qui donne les informations de représentation et la partie <Expr1>,... qui contient les variables et expressions dont les valeurs sont à représenter.

Exemple 1 : je souhaite afficher du texte

```
#include <stdio.h>

int main() {
    printf("Hello world!\n"); /*je place une chaîne de caractère "Hello world!\n" en argument de printf
    return 0;
}
```

Ici, aucune variable n'est donnée en argument de la fonction. «\n» permet le retour à la ligne.

Exemple 2 : je souhaite afficher l'état d'une variable

```
#include <stdio.h>

int main() {
    int A = 10;
    printf("j'ai %i ans \n", A);
    return 0;
}
```

Je place ici une chaîne de caractère en premier argument de la fonction : "j'ai %i ans \n", avec un spécificateur « %i » qui indique que la valeur de A sera imprimée comme entier relatif. Le deuxième argument est la variable qui sera affichée.

Exemple 3 : je souhaite afficher l'état de plusieurs variables

```
#include <stdio.h>

int main() {
    int A = 10;
    int B = 6;
    printf("j'ai %i ans et %i mois\n", A, B);
    return 0;
}
```

Je place ici une chaîne de caractère en premier argument de la fonction : ("j'ai %i ans et %i mois\n", avec un premier spécificateur « %i » qui indique que la valeur de A sera imprimée comme entier relatif et un deuxième spécificateur « %i » qui indique que la valeur de B sera imprimée comme entier relatif. Le deuxième et le troisième argument sont les variables qui seront affichées.

Les différents spécificateurs de format pour printf :

SYMBOLE	TYPE	IMPRESSION COMME
%d ou %i	int	entier relatif
%u	int	entier naturel (unsigned)
%o	int	entier exprimé en octal
%x	int	entier exprimé en hexadécimal
%c	int	caractère
%f	double	rationnel en notation décimale
%e	double	rationnel en notation scientifique
%s	char*	chaîne de caractères

- ✓ Récupérez le programme `td1exo11a.c` sur Moodle. **NE PAS LE COMPILER TOUT DE SUITE.**
- ✓ Essayez de deviner ce qui va être affiché à l'écran
- ✓ Compilez et exécutez le programme pour comparer avec ce que vous aviez prédit à la question précédente.

Il est important qu'il y ait une cohérence totale entre les marqueurs situés dans la chaîne de caractère et les valeurs, que ce soit en terme de quantité qu'en terme de type.

- ✓ Récupérer le programme `td1exo11b.c` sur Moodle.
- ✓ Compilez-le sans l'exécuter. Le compilateur devrait vous afficher des avertissements (ou warnings en anglais), vous prévenant ainsi que votre programme est mal écrit et qu'il risque de ne pas s'exécuter correctement. Si ce n'est pas le cas, il faut forcer l'affichage des « warnings » dans Code::Blocks en allant dans le menu Settings, puis Compiler et cocher la case « Enable all common compiler warnings » (-Wall). Lisez les avertissements affichés et rapprochez-les des erreurs commises dans le programme
- ✓ Exécutez le programme et vérifiez que l'affichage ne correspond pas à ce qui était attendu
- ✓ Vous pouvez maintenant corriger le programme pour faire en sorte qu'il fonctionne correctement (et qu'il n'y ait plus de warning lors de la compilation)

Exercice 5. Fonction scanf

La fonction `scanf` permet de lire une valeur depuis le clavier et de la stocker dans une variable. La syntaxe telle que vue en cours doit être parfaitement respectée, à défaut on s'expose à ce que la valeur récupérée soit mal interprétée par le programme

- ✓ Récupérer le programme `td1exo12.c` sur Moodle.
- ✓ Compilez-le sans l'exécuter. Lisez les avertissements affichés et rapprochez-les des erreurs commises dans le programme
- ✓ Exécutez le programme et vérifiez que celui-ci s'arrête de fonctionner dès la lecture de `b` réalisée
- ✓ Corrigez uniquement l'erreur du `scanf` sur la lecture de `b` : la fin du programme est maintenant atteinte lorsqu'on l'exécute mais l'affichage ne correspond toujours pas à ce qui était attendu
- ✓ Vous pouvez maintenant corriger le reste du programme pour faire en sorte qu'il fonctionne

Il faut également noter que la chaîne de caractères dans le `scanf` ne doit contenir que le caractère « % » suivi de la lettre qui précise le type de valeur lue et rien d'autre. Pour vous en convaincre, vous pouvez rajouter un espace juste après le `%d` dans le premier `scanf` et visualiser le comportement de votre programme.

Une autre difficulté est que la correspondance entre le format et le type de variable est plus strict que pour la fonction `printf`, pour laquelle on pouvait utiliser `%d` pour n'importe quel type de valeur de type entier et `%f` pour tous les réels. Pour le `scanf`, il faudra différencier les sous-types de variables et utiliser un marqueur bien spécifique à chaque fois :

short int	long int	unsigned int	double
%hd	%ld	%u	%lf

- ✓ Exécutez votre programme en entrant la valeur 5.1 pour `x`. Comparez la valeur du carré affiché par votre programme à la valeur réelle issue de la calculatrice : vous pouvez constater que, même pour un calcul extrêmement simple, des approximations non négligeables apparaissent. Il est alors profitable pour les limiter d'utiliser systématiquement le type `double` à la place de `float`.
- ✓ Modifiez en conséquence la déclaration de la variable `x` sans modifier le reste du programme. Compilez-le et exécutez-le. Vous constaterez que celui-ci ne fonctionne plus du tout
- ✓ Modifiez maintenant le `scanf` et ré-exécutez le programme. Vous constaterez que l'approximation de calcul a disparu (ou n'est en tout cas plus visible).

Un autre point qui peut s'avérer délicat est la lecture d'un caractère (avec le marqueur %c). En effet, si celle-ci intervient après la lecture d'une autre variable, la touche « entrée » qui a permis de valider cette saisie antérieure a été mémorisée dans un buffer et sera automatiquement copiée dans le caractère que vous souhaitiez lire. En effet la touche entrée est elle-même un caractère et possède donc son propre code ascii.

- ✓ Pour mettre ceci en évidence, ajoutez dans le programme une variable `c` de type `char` et lisez la depuis le clavier à la suite des instructions précédentes. Vous pourrez ensuite demander au programme d'afficher le caractère saisi ainsi que son code ascii grâce à un `printf` adapté.
- ✓ Testez votre programme, vous constaterez que vous ne pouvez pas saisir de caractère et que le caractère stocké dans `c` est `'\n'` !

Pour remédier à ce problème, une solution est de vider ce buffer d'entrée juste avant la lecture du caractère.

- ✓ Ajoutez l'instruction `fflush(stdin) ;` juste avant le `scanf` permettant de lire le caractère
- ✓ Compilez et exécutez votre programme. Vous pourrez constater que celui-ci a retrouvé un comportement normal

Exercice 6. Conversion de type

Lorsque l'on souhaite effectuer une opération arithmétique sur 2 valeurs qui ne sont pas du même type, le compilateur va au préalable faire une conversion de type, c'est-à-dire qu'il va changer le type d'une des 2 valeurs pour la transformer dans celle de la 2^{ème} valeur. L'ordre des conversions est fixé par la norme du langage C : une valeur de type `char` est modifiée en `short int`, puis en `int`, puis en `long int`, puis en `float` puis en `double` (on s'arrête bien sûr dès que les 2 valeurs sont du même type).

- ✓ Récupérer le programme `td1exo13.c` sur Moodle. **NE PAS LE COMPILER TOUT DE SUITE.**
- ✓ Complétez les lignes 11 à 17 en ajoutant dans la fonction `printf`, après le `'%'`, la lettre correspondant au type du résultat de l'opération
- ✓ Essayez de deviner ce qui va être affiché à l'écran
- ✓ Compilez et exécutez le programme pour comparer avec ce que vous aviez prédit à la question précédente.

Si l'on souhaite forcer une conversion ou aller dans le sens opposé à la chaîne de conversion automatique vue ci-dessus, on peut utiliser un opérateur spécifique : l'opérateur de conversion (ou `cast` en anglais). Il s'écrit en plaçant le type visé entre parenthèses juste avant la valeur à convertir. Ainsi

```
(int) toto
```

transforme la valeur de la variable `toto` en valeur entière quel que soit le type de départ de la variable `toto`.

- ✓ Décommentez les lignes 19 à 21
- ✓ Essayez de deviner ce qui va être affiché à l'écran
- ✓ Compilez et exécutez le programme pour comparer avec ce que vous aviez prédit à la question précédente.
- ✓ Décommentez la ligne 22 et modifiez le `printf` de façon à exécuter exactement le même calcul qu'à la ligne précédente mais sans utiliser l'opérateur de conversion
- ✓ Décommentez la ligne 24, compilez le programme et analysez l'erreur de compilation générée
- ✓ Modifiez la ligne pour pouvoir exécuter l'opération demandée

Exercice 7. Permutation de 2 variables

Un algorithme très classique est de permuter le contenu de 2 variables

- a) On propose un premier algorithme « naïf » :
 - ✓ Récupérer le programme `td1exo14.c` sur Moodle. **NE PAS LE COMPILER TOUT DE SUITE.**
 - ✓ Essayez de deviner ce qui va être affiché à l'écran
 - ✓ Compilez et exécutez le programme pour comparer avec ce que vous aviez prédit à la question précédente. Que peut-on conclure ?
- b) Modifiez le programme de façon à ce qu'il permette réellement de permuter les deux variables `a` et `b`. Vous pourrez pour cela utiliser une troisième variable `c` (que l'on appellera variable tampon).

Exercice 8. Quotient familial

Reprendre l'exercice 2 du TD 1 et écrire le programme correspondant en langage C.