Chapitre 2

Les notions de base du langage C

Chapitre 2 : Les notions de base du langage C

I Les types de base

Toute donnée manipulée doit être typée.

a) Entier : ***int***

En **C** il existe plusieurs types entiers.

Ils dépendent :

- du nombre d'octets sur lesquels ils sont codés
- de leur format, c'est-à-dire s'ils sont signés (possédant le signe - ou +) ou non. Par défaut les données sont signées.

Chapitre 2 : Les notions de base du langage C

Propriétes :

- codage sur 2 ou 4 octets, selon la machine utilisée
- valeurs : -32768 à 32767 sur 2 octets par exemple
- Format d'affichage: %d, %i

Le type ***int*** peut être modifié par l'un des qualificateurs suivants :

- ✓ short : entier court (2 octets); format : %d;
  - ❖ Exemple : short int i; short i;
- ✓ long : entier long (4 octets); format : %ld;
  - ❖ Exemple: long int i; long i;
- ✓ unsigned : entier non signé positif; format : %u, %lu;
  - ❖ Exemple: unsigned int i; unsigned i; unsigned long int i; unsigned long i;

Chapitre 2 : Les notions de base du langage C

b) Réel :

float

- codé sur 4 octets (1 bit → signe, 8 bits → exposant, 23 bits → mantisse : *partie du nombre sans la virgule*)
- valeurs : de $3.4*10^{-38}$ à $3.4*10^{38}$

double

- codé sur 8 octets (1 bit → signe, 11 bits → exposant, 52 bits → mantisse)
- valeurs : de $1.7*10^{-308}$ à $1.7*10^{308}$

long double

- codé sur 10 octets (1 bit → signe, 15 bits → exposant, 64 bits → mantisse)
- valeurs : de $3.4*10^{-4932}$ à $3.4*10^{4932}$

Remarque : les constantes réelles ont par défaut le type double.

Chapitre 2 : Les notions de base du langage C

c) Caractère : *char / unsigned char*

- Le caractère est utilisé à travers son code dans la table ASCII, ce qui explique la notion de caractère signé ou pas
- codé sur 1 octet
- valeurs : de -128 à 127 pour *char*; de 0 à 255 pour *unsigned char*
- format : %c
- Les valeurs caractères sont notées entre simples quotes : ' '
- Les valeurs chaînes de caractères sont notées entre doubles quotes : " "
- Les caractères *non-imprimables* de la table ASCII ou caractères *spéciaux* sont utilisables en les précédant du caractère d'échappement anti-slash : ****.

Chapitre 2 : Les notions de base du langage C

Les caractères spéciaux les plus utilisés

\n : nouvelle ligne (ou saut de ligne)

\t : tabulation horizontale

\v : tabulation verticale

\b : retour arrière

\r : retour chariot

\f : nouvelle page (ou saut de page)

\a : bip

\0 : caractère nul (null)

Chapitre 2 : Les notions de base du langage C

Remarque :

Il n'existe pas de type chaîne de caractères tel qu'il a été vu en algorithmique. Néanmoins, on manipule des constantes chaîne de caractères. Elles sont définies entre guillemets " " et ont pour format d'affichage : *%s*.

Exemple : " ceci est ma chaîne "

Toutes les notations de caractères non imprimables sont utilisables dans les chaînes.

Exemple : " si je suis affichée \n je serai sur \n 3 lignes "

A l'intérieur d'une chaîne, le caractère " est noté \" et \ est noté \\.

Chapitre 2 : Les notions de base du langage C

Renommage de type :

❖La commande ***typedef*** permet de renommer un type dans le but d'avoir une plus grande clarté.

Exemple : typedef *short* Entiercourt;

Déclaration d'une variable dans le type : Entiercourt i;

Chapitre 2 : Les notions de base du langage C

II Les constantes nommées

- En **C**, il n'est pas possible de donner un nom à une constante. Néanmoins, on peut avoir un effet équivalent grâce au pré-processeur et à la notion de **directive** donnée au pré-processeur.

Syntaxe : **#define** *identificateur valeur*

Exemple : **#define PI** 3.14159

- Lorsque le *pré-processeur* lit cette ligne, il remplace, dans le programme source, toute occurrence de l'identificateur **PI** par la valeur constante réelle 3.14159.

Chapitre 2 : Les notions de base du langage C

III Déclarations des variables dans les types de base

Syntaxe : **nom_du_type** *liste_de_noms_de_variables;*

Exemples :

```
short int i;
float f, note, moy;
char c;
double d;
```

Chapitre 2 : Les notions de base du langage C

Remarques :

✓Plusieurs variables peuvent être déclarées avec le même type. Elles seront séparées par des virgules.

✓Une ligne de déclaration est terminée par un point-virgule.

✓On peut initialiser une variable à la déclaration :

Exemple :

```
short int i = 2, nb = 10;
float note, moy = 0.0;
char c = 'e';
```


Chapitre 2 : Les notions de base du langage C

IV Opérateurs usuels

a- L'affectation

En C, ***l'affectation*** est un **opérateur** et non une instruction.

Syntaxe : variable **=** expression;

Le terme de gauche peut être une variable simple, un élément de tableau (*mais pas une valeur constante*).

Exemples :

```
i = 5;
f = 3.14;
k = j+i;
```

Chapitre 2 : Les notions de base du langage C

➢ L'affectation effectue une *conversion de type implicite* : la valeur de l'expression est convertie dans le type du terme de gauche.

Exemple :

```
int i, j = 4;
float x = 2.5;

i  = j + x; (i vaut 6)
x = x + i; (x vaut 8.5)
```


Chapitre 2 : Les notions de base du langage C

Sémantique :

L'opérateur = a deux effets :

✓ un effet de bord qui consiste à affecter la valeur de *expression* à la *variable*

✓ l'opérateur délivre la valeur affectée qui pourra être utilisée dans une expression englobant l'affectation.

Exemple :

```
int k = 2, i, j;
i = (j = k) + 1;
```

- k est affecté à j
- on ajoute 1 à cette valeur
- le résultat est affecté à i, ce qui donne : i vaut 3.

Chapitre 2 : Les notions de base du langage C

b- Les opérateurs arithmétiques

Ce sont les opérateurs arithmétiques classiques, à savoir :

- l'opérateur unaire **-** : permet de changer le signe
- les opérateurs binaires
  - **+** : addition
  - **-** : soustraction
  - ***** : multiplication
  - **/** : division
  - **%** : modulo ou reste de la division entière
- ils sont communs aux *entiers* et aux *réels* (sauf le modulo %).

Chapitre 2 : Les notions de base du langage C

Spécificités :

- le même symbole est utilisé pour la division entière et réelle : **/**
- Le résultat dépend du type des opérandes, sachant que le type réel l'emporte sur le type entier :

entier/*entier* = *entier*;
entier/*réel* = *réel*;
réel/*entier* = *réel*;
réel/*réel* = *réel*.

Exemple :

float x;
x = 5/2 affecte 2 à x
x = 5/2. affecte 2.5 à x (écrire 2. est équivalent à 2.0)

Chapitre 2 : Les notions de base du langage C

- L'opérateur **%** ne s'utilise que sur des opérandes de type entier.
- Si l'un des deux est négatif, le signe du reste dépend en général du signe du dividende.

Remarque :

✓ il n'y a pas d'opérateur qui élève un nombre ***a*** à une puissance ***b*** donnée.

✓ Il faut pour cela utiliser la fonction **pow**(a,b) de la librairie mathématique math.h.

Chapitre 2 : Les notions de base du langage C

c- Les opérateurs de comparaison (ou relationnels)

- **>** : strictement supérieur
- **<** : strictement inférieur
- **>=** : supérieur ou égal
- **<=** : inférieur ou égal
- **==** : égal
- **!=** : différent (ou non égal)

Syntaxe : expression1 ***opérateur*** expression2

Le résultat de la comparaison est de type ***int*** = 1 si la condition est vérifiée, 0 sinon.

Remarque : le type *booléen* n'existe pas en **C**, d'où le type ***int*** pour le résultat d'une comparaison.

Chapitre 2 : Les notions de base du langage C

d- Les opérateurs logiques

- **&&** : et
- **||** : ou
- **!** : non

Syntaxe : expression1 ***opérateur*** expression2

✓ L'évaluation se fait de gauche à droite
✓ Le résultat calculé est de type ***int*** = 1 si la condition est vérifiée (à *vrai*), 0 sinon (*faux*).

Remarque : **&&** et **||** ont un rôle de ***courts-circuits***.
Exemple : (a **<** 30) **&&** (a **>=** 10) **&&** (x **==** 0)
La 2° clause n'est évaluée que si la 1° est égale à *vrai*. Si elle vaut *faux*, toute l'expression vaut *faux*. L'évaluation s'arrête à la condition 1.
La 3° clause n'est évaluée que si a est compris entre 10 et 30.
L'expression : (a **<** 30) **&&** (a **>=** 10) **II** (x **==** 0) n'est évaluée que si a n'est pas compris entre 10 et 30.

(*Il existe d'autres opérateurs logiques bit à bit, traitant des bits de même poids*)

Chapitre 2 : Les notions de base du langage C

e- Les opérateurs d'affectation composée

Ce sont les opérateurs : **+=**, **-=**, ***=**, **/=**, **%=**

Syntaxe : variable ***opérateur*** expression

Pour chacun de ces opérateurs, l'expression est équivalente à :
variable **=** variable ***opérateur*** expression

Exemple : a**+=** 1 et a **=** a **+** 1 ⇒ 2 formes d'écriture pour le même résultat
b**%=** c et b **=** b **%** c : même résultat ⇒ reste de la division entière de b par c

Chapitre 2 : Les notions de base du langage C

f- Les opérateurs d'incrémentation et de décrémentation

- Ce sont les opérateurs ++ et --.
- Ils s'utilisent aussi bien en *préfixe* (*opérateur* variable) qu'en *suffixe* (variable *opérateur*) de variable.
- En *préfixe* : la valeur utilisée de la variable est la valeur après modification.
- En *suffixe* : la valeur utilisée de la variable est la valeur avant modification.

Chapitre 2 : Les notions de base du langage C

Exemple :

++i : on incrémente i de 1 avant d'utiliser sa nouvelle valeur

i++ : on utilise la valeur de i avant de l'incrémenter de 1

```
int x = 2, y, z;
y = ++x;          /* x et y valent 3 */
z = x--;          /* z vaut 3 et x vaut 2 */
x = z++;          /* x vaut 3 et z vaut 4 */
```

Chapitre 2 : Les notions de base du langage C

g- L'opérateur d'adressage

L'opérateur **&** appliqué à une variable donne l'adresse mémoire de la variable.

Syntaxe : **&** nom_variable

Remarque : Le nom de la variable tout seul fait référence au contenu.

Chapitre 2 : Les notions de base du langage C

h- L'opérateur de conversion de type

- On peut convertir de manière **explicite** le type d'un objet (variable ou valeur) : cette opération s'appelle un ***cast***.

Syntaxe : **(**type**)** objet

Exemple : **int** a = 5, b = 2;

L'expression : **(float)** a**/**b vaut 2.0 (ou 2.) au lieu de 2, après conversion du résultat de la division de deux entiers.

Chapitre 2 : Les notions de base du langage C

i- L'opérateur virgule

- Une *expression* peut être constituée de *plusieurs expressions* avec comme séparateur la virgule
- Son *évaluation* se fera de gauche à droite

Syntaxe : expression1**,** expression2**,** …., expressionN**;**

Exemple :

```
a = ((x = 8),  (x + 3));   /* a vaut 11 */
```


Chapitre 2 : Les notions de base du langage C

j- L'opérateur ternaire conditionnel

- L'opérateur **?** est un opérateur **ternaire** qui permet d'exprimer une condition.

Syntaxe : condition **?** expression1 **:** expression2;

Interprétation : si la condition est vérifiée toute l'expression vaut *expression1* sinon elle vaut *expression2*.

Exemple :

```
min =  ((x < y) ? x : y)        /* on affecte à min le minimum entre x et y */
```