

0.1 Type de données, tableaux, opérateurs

Le langage C est un langage algorithmique impératif. Les mots clés s'écrivent en minuscule (main, printf, scanf), en C il n'y a que des fonctions, qui ont chacune un nom, des paramètres ou pas, elles renvoient chacune quelque chose ou pas et dont le corps est contenu entre {}. Chaque programme C contient au moins une fonction int main(void), qui constitue l'entrée du programme. Chaque instruction, se termine par un ; et voici l'aspect d'une fonction :

```
type_valeur_renvoyée nom_de_la_fonction(paramètres)
{
    définition des variables
    instructions
}
```

Avant d'être utilisées les variables doivent être déclarées pour en indiquer le type, ceci permet de réserver suffisamment de mémoire pour le programme. En C, il y'a 2 types fondamentaux : les entiers et les réels.

0.1.1 1. Les types élémentaires :

0.1.2 1.1 Les entiers :

On distingue 4 sortes d'entiers :

- **char** : les caractères, un entier codé sur 1 octet.
- **short int** : entier sur 2 octets
- **int** : un entier sur 4 ou 2 octets.
- **long int** : un entier sur 4 octets.

Chacun de ces types pouvant être signé, ou non. Sauf indication contraire, ils sont signés.

Les entiers peuvent être représentés dans trois bases :

- **octale (base 8)** : Ils sont alors précédés de 0. Chaque chiffre est compris entre 0 et 7 inclus. 895 en octale s'écrit 01577.
- **décimale (base 10)** : C'est la représentation, habituelle, 895 en décimale s'écrit 895. Pour les caractères on utilise les " exemple 'a', 'b' ou 'c'.
- **hexadécimale (base 16)** : Ils sont alors précédés de 0x. Chaque nombre est compris entre 0 et 15 inclus. A ceci près que 10 s'écrit, A que 11 s'écrit B et par extension 15 s'écrira F. En hexadécimal 895 s'écrira 0x37F

0.1.3 1.2 Les réels :

[url=c_type_float][url]# On distingue 3 sortes de réels :

- **float** : réel simple sur 4 octets (6 chiffres significatifs après la virgule)
- **double** : réel double sur 8 octets (15 chiffres significatifs après la virgule)
- **long double** : long réel double sur 10 octets

Les réels sont évidemment signés, ils peuvent être représentés de différentes manières :

- **Sous forme fixée** : 22/7 ou 3.14
- **Sous forme scientifique** : 3,56.10-5. Pour cela nous écrirons : 3,56e-5

0.1.4 1.3 Les caractères spéciaux :

- \n : nouvelle ligne
- \r : retour chariot
- \t : tabulation
- \0 : caractère nul
- \" : l'apostrophe

0.1.5 1.4 Récapitulatif :

Notez qu'en C il n'y a pas de booléens, en général l'entier 0 code FAUX et un entier dif-

	Type de donnée	Signification
	char	Caractère
	unsigned char	Caractère non signé
	short int	Entier court
	unsigned short int	Entier court non signé
férent de 0 code VRAI. (3 < 2) est évaluée à 0 et ('d' > 'b') renvoie 1.	int	Entier
	unsigned int	Entier non signé
	long int	Entier long
	unsigned long int	Entier long non signé
	float	flottant (réel)
	double	flottant double
	long double	flottant double long

0.1.6 2. Les déclarations :

- **Déclaration de variables** : La déclaration d'une variable, précise le type et réserve de la mémoire, la variable n'a alors aucune valeur

```
char c ;
int un, deux, trois ;
float genre, adverbe ;
```

- **Déclaration et initialisation** : Cette fois, la variable reçoit une valeur modifiable par la suite

```
char c = 'b' ;
int un=1, deux=2, trois ;
float genre=3.14, adverbe=1.0e-7 ; complément=1.0e+6 ;
```

- **Déclaration de constante** : la valeur de la variable est fixée et non modifiable

```
const int N=20 ;
const float PI=3.1415 ;
#define N 20 ;
#define PI 3.1415 ;
```

Notez qu'il y'a double déclaration, dans l'exemple ci-dessus, c'est juste pour faire la correspondance.

0.1.7 3. Les Opérateurs :

- **Arithmétiques** : + fait l'addition, - la soustraction, * la multiplication, % la division entière et / la division réelle. Ils s'appliquent aux entiers et aux réels. Il y'a beaucoup de gadgets associés à ces opérateurs, je n'en expose que quelques-uns que vous rencontrerez souvent : ++, qui a le même effet sur a que a = a+1 ; il existe aussi --. x += a qui a le même effet sur x que x = x+a, il est donc possible d'envisager également -=, *=, /= et %=.
- **Relationnels** : == teste l'égalité, != test la différence (non-égalité) et bien-sûr <, >, <=, >=, qui ont leurs sens triviaux.
- **Logiques** : ! fait la négation, && équivaut au Et logique, || équivaut lui au OU logique.
- **Binaires** : & c'est le ET bit à bit, | le ou bit à bit, ^ le OU exclusif bit à bit, « décalage de bit vers la gauche, » décalage de bit vers la droite, - complément d'une chaîne de bits.
- **L'affectation** : Comme vous vous en doutez, on utilise = pour affecter des une valeur à une variable par exemple, ceci pouvant se faire pendant la déclaration ou dans le corps du programme.

0.1.8 4. Les Tableaux :

0.1.9 4.1 Généralité sur les tableaux :

Un tableau est une suite d'éléments de même type dont les adresses mémoires sont consécutives. Pour déclarer un tableau, il faut indiquer, le type des éléments, le nom du tableau et le nombre de cases :

```
int tab[20]; /* tableaux de 10 entiers */
char chat[15]; /*tableau de 15 caractères */
```

Un tableau de 20 cases est indicé de 0 à 19, ainsi tab[2] correspond en fait au 3ème élément du tableau. Il est bien-sûr possible de parcourir un tableau à l'aide d'une variable ainsi tab[i], renverra le i+1 ème élément. Il y'a 2 manières d'initialiser un tableau :

- **A la déclaration** : int tab[2]={ 10,20 } ;
- **En cours de programme** :

```
int tab[2];
tab[0]= 10;
tab[1]=20;
```

0.1.10 4.2 Les tableaux à plusieurs dimensions :

Un tableau à plusieurs dimensions est en fait un tableau de tableau, le formalisme associé permet cependant de les manipuler de manière transparente. On fournit d'abord la taille du tableau principal, puis la taille des tableaux sous-jacents récursivement. Par exemple dans le cas d'une matrice, on donne d'abord le nombre de ligne puis le nombre de colonne :

```
int tab[4][3]={ { 1,2,3},
{4,5,6},
{7,8,9},
{10,11,12}
}
```

```
int tab[3][4]={ {1,2,3,4},
{5,6,7,8},
{9,10,11,12}
}
```

0.1.11 4.3 Chaînes de caractères :

Il n'y a pas de type chaîne de caractère en C, mais en utilisant un tableau de caractères, on peut, les représenter et les manipuler. Lorsque j'écris : `char chat[15]` ; je déclare un tableau pouvant contenir, 14 caractères et un 15 ème caractère `\0`. Concrètement je déclare une chaîne de 14 caractères maximum. Notez que les cases du tableaux non utilisées sont remplacées par des `\0`. Voici un programme à compiler, qui donne quelques applications des chaîne de caractères :

```
#include <stdio.h>
#include <string.h>
int main(void)
{
char Chaine1[10], Chaine2[10];
char Chaine3[10] = {'t','e','x','t','e',' ',' ','3'};
char Chaine4[10]= "texte 4";
Chaine1[0]= 't';
Chaine1[1]= 'e';
Chaine1[2]= 'x';
Chaine1[3]= 't';
Chaine1[4]= 'e';
Chaine1[5]= ' ';
Chaine1[6]= '1';
Chaine1[7]= '\0';
printf("\n Entrez une chaîne de caractères : ");
scanf("%s",Chaine2);
printf("\n la chaîne 1 passée en dur dans le programme :
%s", Chaine1);
printf("\n la chaîne 2 passée depuis le clavier : %s",
Chaine2);
printf("\n la chaîne 3 passée en dur dans le programme :
%s", Chaine3);
printf("\n la chaîne 4 passée en dur dans le programme :
%s \n", Chaine4);
return(1);
}
```

Vous pouvez le compiler et l'exécuter, il vous affichera :

```
Entrez une chaîne de caractères : texte_2
la chaîne 1 passée en dur dans le programme : texte 1
```

la chaîne 2 passée depuis le clavier : texte_2
la chaîne 3 passée en dur dans le programme : texte 3
la chaîne 4 passée en dur dans le programme : texte 4

0.1.12 5. Les Structures :

Une structure comme son nom l'indique est une juxtaposition d'objets de types différents ou pas. La cas d'école sur les structures est bien-sûr les complexes. Vous pouvez définir un complexe par :

```
struct complexe { float reel;  
float imaginaire; };
```

On peut alors déclarer des complexes par : `struct complexe z, z1={3.0,4.0}, z2={1.0,2.0};`
Affecter des valeurs : `z.reel=z1.reel + z2.reel; z.imaginaire = 1.0;`
Les mettre dans un tableau : `struct complexe tab_comp[10];` Accéder à une valeur du tableau : `tab_comp[1].reel=2.0;` «[Précédent](#)¹ [Suivant](#)»²

¹<http://www.trustonme.net/didactels/149.html>

²<http://www.trustonme.net/didactels/151.html>