المدرسة العليا للتجارة
بصفاقس

Note de cours 2

Types de données - Fonctions d'entrée/sortie

Résumé

Les types

- **3 types** de base (**char**, **int**, **float**) que l'on peut modifier par **3 spécificateurs** (short, long, unsigned).

type	taille (en bits)	plage de valeurs	
char	8	-128 à +127	Caractère
unsigned char	8	0 à 255	
short	16	-32768 à 32767	Entier (int)
unsigned short	16	0 à 65535	
long	32	-2.147.483.648 à 2.147.483.647	
unsigned long	32	0 à 4.294.967.295	
float	32	-3.4e38 à 3.4e38 (7 chiffres significatifs)	Réel
double	64	-1.7e308 à 1.7e308 (15 chiffres significatifs)	
long double (non standard)	80 ou 128	Dépend de la machine	

- int = short sur PC, et int = long sur les stations 32 bits. Désormais, plusieurs compilateurs considèrent int comme 32 bits (-2.147.483.648 à 2.147.483.647).
- 1 bit est une variable binaire de valeur 0 ou 1.

Déclaration et initialisation des variables

- **Déclaration :**

```
<type>  Nom_Variable;
```

- **Initialisation :** donner une première valeur à la variable
  → affectation.

```
#include <stdio.h>
void main()
{
    char car;
    int i;
    float r;
    car='a';
    i=4;
    i=-5;
    r=3.9 ;
}
```

Une affectation écrase l'ancienne valeur et la remplace par la nouvelle

Déclaration et initialisation des variables

- **Déclaration et initialisation combinée :**

Exemple :

```
int a=5, b, c;
```

Lecture et Affichage

- **printf() :** le programme écrit un message sur l'écran
  - Message texte seulement :
    (1 champs)

```
printf("Bonne chance monsieur!\n");
```

  - Message texte contenant les valeurs de certaines variables :
    (2 champs) : `printf("champ1", champ2);`

```
num = 1 ;
p = 2.510 ; nbr = 3 ;
printf("Le prix du produit %d est %f seulement\n",num, p);
printf("Il en reste %d\n",nbr) ;
```

Résultat affiché :

```
Le prix du produit 1 est 2.510000 seulement
Il en reste 3
_
```

Lecture et Affichage

```
printf("expression texte", variable1, variable2,…,variableN);
```

- Dans « expression texte », il y a des valeurs. On doit spécifier dans cette zone les types des variables respectives par l'intermédiaire de % suivi d'un caractère indiquant le type de la variable :
  - %d : pour les décimaux
  - %f : pour les flottants (réels) → %.nf : la valeur n est le nombre de chiffres après la virgule. Par défaut : 6 chiffres
  - %c : pour les caractères
  - %s : pour les chaînes (tableaux) de caractères

```
printf("Le prix du produit %d est %f seulement\n",num, p);
```

Lecture et Affichage

- **scanf() :** le programme attends la lecture d'une valeur à partir du clavier. Une fois que la touche ENTREE est tapée, cette valeur sera affectée à une variable indiquée dans le champ 2 de l'instruction et précédée par le symbole &.

```
scanf(champ1, champ2);
```

Le champ1 indique le type de la valeur, qui est aussi le type de la variable, par l'intermédiaire de "%" (entre guillemets et suivi d'une lettre indiquant le type):

```
#include <stdio.h>
void main()
{
    char L; float moy; int num;
    scanf("%d", &num);
    scanf("%c", &L);
    scanf("%f", &moy);
    printf("num = %d\nmoy = %f\nL = %c\n",num,moy,L);
}
```

Conversion

- **Conversion commandée (expression générale de la conversion):**

  Synataxe : <variable> = (nouveau type) <variable à convertir> ;

Exemple :

```
float x; int a=5;
x = (float) a;
```

Exemple :

```
float x= 4.3; int a;
a = (int) x;
```

- **Conversion par simple affectation :**

  Synataxe : <variable> = <variable à convertir> ;

Exemple :

```
int i; double x = 4.3;
i = x ; // i prendra la valeur 4 (partie entière de x)
```

Par ailleurs :

- la valeur résultante de x suite à la syntaxe `{int x; x = 7/2;}` est 3 (partie entière)
- la valeur résultante de x suite à la syntaxe `{float x; x = 7/2;}` est 3 (partie entière : entier / entier = entier, même si la variable résultat est déclarée en tant que réel)

➔ il faut convertir en réel au moins l'un des deux éléments de l'opération : la valeur résultante de x suite à la syntaxe {float x; x = 7/(float)2;} est 3.5

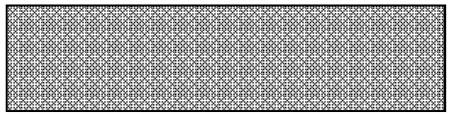
Conversion

- **Conversion automatique entre char et int:** les « char » sont automatiquement transformés en « int » (on considère leurs codes ASCII : nombre entre -128 et 127).

Exemple :

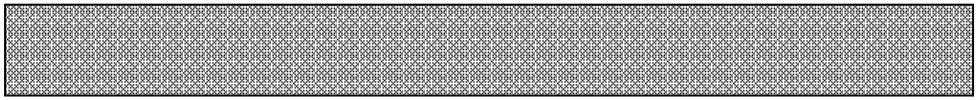
```
#include <stdio.h>
void main()
{
   int i=2, j;
   char x='a', y;
   j= x; //donne j=97 (le code ASCII de x)
   printf("j = %d \n", j);
   y = x + i; // y = 'a' + 2 : y est le caractère de code ASCII = 97 + 2
   printf("y = %c \n", y);
   printf("ASCII(y) = %d \n", y);
```

Les tableaux

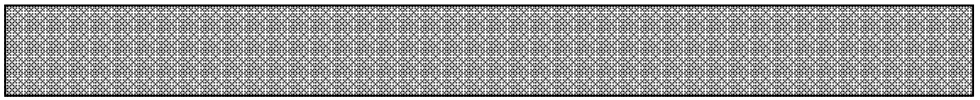
- Les tableaux sont des structures pratiques pour traiter plusieurs variables du **même type → possibilité de rangement**.
- En C, on peut définir des tableaux pour tous les types de base (entier, réel, caractère…),
- **Déclaration :** `<type> nom_tableau[Nombre_elements];`
- **Exemple :** 
- 5 est la taille du tableau (le nombre de ses éléments),
- En C, l'indexation d'un tableau commence à partir de 0,
- tab[0] représente le premier élément du tableau et tab[1] est le deuxième élément et ainsi de suite…
- Le dernier élément est tab[4] (tab[5] n'existe pas).

Autres fonctions d'entrée / sortie

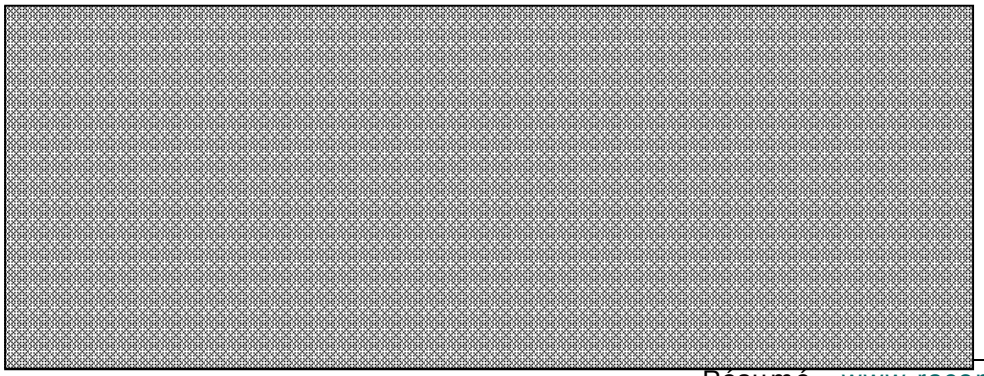
- **prinf() et scanf() .**
- **getchar() et putchar() :**
  - getchar → saisir un caractère et d'affecter sa valeur à une variable :



  - putchar → afficher la valeur d'une variable caractère à l'écran :



- **gets() et puts() :** saisir et afficher une chaîne de caractères (un texte)



Plus de détails, d’exemples et d’explications

dans la séance du cours