Nous allons revenir plut tard sur les structures de données pour définir les pointeurs.

LES ENTREES ET SORTIES

1°) Affichage à l'écran

fonction : **printf**

Le C permet un affichage à l'écran avec un contrôle de la forme et du format des données à afficher.

Exemple:

- Affichage d'une chaîne de caractères:

```
printf("ceci permet l'affichage d'une chaine de caractères à l'ecran sur une seule ligne");
```

La spécification du format commence par le caractère % suivi d'une lettre clé qui identifie le type de l'argument correspondant.

Syntaxe : printf("chaine de caracteres plus les formats à la bonne place", arg1,.....,argN);

Exemples:

- Affichage d'un entier

a-

```
int i;
i = 1258;
printf("la variable entiere i a pour valeur %d .",i);
```

le résultat à l'écran : la variable entiere i a pour valeur 1258 .

b-

```
int i, j;
i = 10; j =12;
printf("la somme de %d et de %d vaut %d .", i , j, i+j);
```

le résultat à l'écran : la somme de 10 et de 12 vaut 22 .

- Affichage d'un caractère

a-

```
printf("La premiere lettre de l'alphabet est le %c .", 'a');
```

le résultat à l'écran : La premiere lettre de l'alphabet est le a .

b-

```
char vari_c;
vari_c = 'Z';
printf("la dernier lettre de l'alphabet en majuscule est le %c .", vari_c);
```

le résultat à l'écran : la dernier lettre de l'alphabet en majuscule est le Z .

c-

```
printf("%c et %c ont les codes ASCII %d et %d", 97 , 65 , 97 , 65 );
```

le résultat à l'écran : a et A ont les codes ASCII 97 et 65

- Affichage formaté des chaînes de caractères

a-

```
printf("%s", "c'est une chaine de caracteres");
```

le résultat à l'écran : c'est une chaine de caracteres

b-

```
char vari_string[] = "une variable de type chaine de caracteres";
printf(" de quelle type est la variable vari_stirng? \nreponse : %s",vari_string);
```

le résultat à l'écran :

de quelle type est la variable vari_string?
réponse : une variable de type chaine de caracteres

Le tableau suivant résume les différents formats possibles en langage C.

Format	Donnée formatée
%d	Nombre décimal
%i	Nombre Entier
%u	Nombre Décimal sans caractère précédent (signe)
%o	Nombre Entier octal
%x,%X	Nombre entier hexadécimal
%c	Caractère ASCII
%f	Nombre à virgule flottante
%e,%E	Nombre à virgule flottante en titre exponentiel
%g,%G	Nombre à virgule flottante aux formats %f ou % e
%s	Chaîne de caractères
%p	Pointeur

- Largeur minimale et précision

La largeur d'une valeur détermine le nombre de caractères ou de chiffres minimaux qu'il faudra afficher. Pour spécifier la largeur minimale, on place un nombre entier entre le caractère % et la lettre clé de formatage. Si la représentation d'une valeur nécessite mois de positions que n'en indique la largeur minimale, les positions restant avant la valeur sont complétées par des espaces. Si la représentation nécessite plus de positions que n'indique la largeur minimale, l'espace supplémentaire exigé est occupé par printf, cela indépendamment du type de données de l'argument à afficher.

Exemple:

```
printf("%s \n", "cinq");
printf("%2s \n", "cinq" );
printf("%10s \n", "cinq");
```

Le résultat à l'écran est:

```
cinq          <- affichage normal
cinq          <- affichage complété par le printf
------cinq    <- affichage complété par des espaces
```

La précision spécifie les valeurs à afficher. La spécification de précision est un nombre entier qui suit la largeur minimale et est séparé de cette dernière par un point. Selon le type de données, le terme précision peut prendre différentes significations.

Pour les données réels, il indique le nombre de chiffre après la virgule. Avec des nombres entiers, il donne en revanche le nombre minimal de chiffres à afficher (à ne pas confondre avec la largeur minimale qui indique le nombre de position à afficher. Pour ce qui concerne les chaînes de caractères, la précision détermine le nombre maximal de caractères à afficher.

Exemples:

```
printf("%2.3f\n%11.6f\n\n",3.1415,3.1415);
printf("%5.5f\n%8.5f\n\n",314.0 ,314.0 );
printf("%6.6s\n%6.3s", "string", "string");
```

Donner le résultat de l'affichage:

2°) Saisie formatée au clavier

Fonction : **scanf**
Syntaxe : **scanf** (" Le format des données a lire", [adresse des arguments]);

La fonction **scanf** permet de saisir des données dans le format désigné par la chaîne de caractères. La chaîne de format peut contenir aussi bien des caractères usuels que des spécifications de format. Ces derniers commencent par le caractère %. Les formats sont utilisés par la fonction d'éditions **printf**. Par contre une différence fondamentale entre **printf** et **scanf** réside dans la nature des arguments manipulés par ces fonctions. En effet la fonction **printf** manipule des "copies des arguments" quelle reçoit mais la fonction **scanf** utilise directement les originaux de données. Pour cela lors de l'appel de la fonction reçoit les adresses des arguments.

Exemple:

```
int i;
char c;

printf("entrer la valeur de votre entier ? :"); scanf("%d", &i) ;
printf("l'entier i saisi est %d",i);

printf("entrer un caractère au clavier :"); scanf("%c",&c);
printf(" la caractère lu est : %c",c);
```

Il est tout à fait possible d'inclure, dans la chaîne de format du **scanf**, des caractères qui ne sont pas des spécifications de format ou des espaces de séparation qui doivent figurer aussi dans la saisie à l'emplacement adéquat.