scanf

scanf *est une instruction d'entrée formatée , c'est-à-dire qu'elle permet à l'utilisateur de fournir des informations au programme (« entrée ») , en respectant un certain format.*

L'instruction **scanf** permet de fournir au programme des valeurs et de les stocker dans des variables. Elle se présente de la manière suivante :

scanf ("*chaine_de_format*", *adresse_variable_1* , *adresse_variable_2* , …) ;

où ***adresse_variable_1*** , ***adresse_variable_2*** sont les adresses des variables (mentionnées dans l'ordre d'affichage) dans lesquelles on veut stocker les valeurs saisies, et ***chaine_de_format*** décrit le format sous lequel ces valeurs doivent être saisies par l'utilisateur. En particulier, les emplacements dans ***chaine_de_format*** , où l'on doit saisir les contenus des variables, sont matérialisés par des « formats » de la forme **%…** décrivant de quel type est la variable et comment elle doit être saisie.

Deux valeurs successives doivent être saisies, séparées par un caractère d'espacement (blanc, tabulation ou saut de ligne) pour les distinguer l'une de l'autre.

D'autres caractères que les formats **%…** peuvent figurer dans ***chaine_de_format*** :

➔ si ce sont des caractères d'espacement, ils forceront **scanf** à lire l'entrée jusqu'à ce qu'un caractère valide autre qu'un caractère d'espacement soit rencontré.
➔ sinon, ils devront figurer au même emplacement dans la saisie ou alors **scanf** s'arrête.

Attention : ***adresse_variable_1, …*** *sont les adresses des variables à affecter par les valeurs saisies. En particulier, si les variables sont de type simple (entier ou réel), leur nom doit être précédé de l'opérateur unaire d'adressage* ***&***

Exemples :

```
int i ;
printf("Entrez un entier : ") ;
scanf("%d ",&i) ;
```

Après l'affichage du message, le programme attend que l'utilisateur saisisse une valeur suivie d'un **<ENTREE>**. Celle-ci sera considérée comme entière (**%d**) et sera stockée dans la variable **i** .

Attention Les formats **%…** doivent être placés dans la chaîne de format dans le même ordre que les variables qu'ils sont censés remplacer.
➔ *Règle : autant de formats que de variables.*

```
int jour, mois, annee ;
printf("Saisir une date au format JJ/MM/AAAA : ") ;
scanf("%d/%d/%d",&jour,&mois,&annee) ;
```

Les formats **%d** dans la chaîne de format, sont séparés par des / , ce qui implique que l'utilisateur doit saisir ces / à ces mêmes places :

12/11/2001	est une saisie correcte
12-11-2001	est une saisie incorrecte, refusée par le **scanf** de l'exemple ci-dessus.

Si la chaîne de format avait été **"%d-%d-%d"** , alors 12-11-2001 aurait été correcte, et 12/11/2001 non.

La syntaxe de **scanf** implique que les caractères **'** , **"** , **%** et \ soient considérés comme des caractères spéciaux. Pour néanmoins les saisir :

pour saisir	***inclure dans la chaîne de format***
%	**%%**
\	\\
"	\"
'	\'

L'antislash \ *protège* le caractère qui le suit d'une interprétation erronée.

➔ Formats

Format	***Lit la saisie sous la forme d'un(e)***	***Déclaration de la variable réceptacle***
%d	entier décimal	**int**
%i	entier décimal, octal (si précédé de **0**), hexadécimal (si précédé de **0x**)	**int**
%o	entier octal non signé	**int , unsigned int**
%x	entier hexadécimal non signé	
%u	entier décimal non signé	**unsigned int**
%hd	entier décimal court	**short**
%ho	entier octal court non signé	**short** ou **unsigned short**
%hx	entier hexadécimal court	
%hu	entier décimal court non signé	**unsigned short**
%ld	entier décimal long	**long**
%lo	entier octal long non signé	**long** ou **unsigned long**
%lx	entier hexadécimal long non signé	
%lu	entier décimal long non signé	**unsigned long**
%c	caractère	**char**
%f	nombre à virgule flottante	**float**
%e , **%E**	nombre en notation scientifique	
%g, %G	nombre au format **%f** ou aux formats **%e** ou **%E** suivant l'exposant	
%lf	nombre à virgule flottante	**double**
%le , **%lE**	nombre en notation scientifique	
%Lg , %LG	nombre à virgule flottante	**long double**
%s	chaîne de caractères	**char ***
%p	pointeur (adresse d'une variable)	**… ***
%n	nbre de caractères déjà lus	
%%	**%**	

Attention : l'utilisation d'un format **%d , %i , %o , %x , …** pour la saisie d'un **char** lit le code ASCII du caractère et non le caractère lui-même.

L'utilisation de formats ne correspondant pas aux types d'expressions voulus, peut conduire à des saisies tronquées, voire aberrantes.

➔ Sauts, largeurs, précisions et alignements

Pour sauter une valeur lue en entrée, c'est-à-dire la lire mais ne pas la stocker, il faut insérer un ***** entre **%** et la clé de formatage :

```
int i ; char ck ;
scanf("%d%*d%c",&i,&ck) ;
```

lit une entrée du type : **123 45 w**
A l'issue de la saisie, la variable **i** contiendra **123** , la variable **ck** contiendra **'w'** et **45** sera ignorée. (Par contre, ce 2ème champ d'entrée, bien qu'ignoré, doit être présent)

Il est possible de spécifier dans un format, le nombre maximal de caractères ou de chiffres à lire.

Pour ce faire, on intercale entre **%** et la clé de formatage (**d , i , hd , f , c , s , …**) un entier optionnel ***<largeur_maximale>*** : alors ***<largeur_maximale>*** caractères ou chiffres seront lus au maximum, sauf si un caractère d'espacement est rencontré.

```
int i, j ;
scanf("%2d%3d",&i,&j) ;
```

interprétera l'entrée **45781** comme suit :
la variable **i** recevra **45** et la variable **j** recevra **781**
*Dans ce cas, il n'est pas nécessaire de séparer **45** et **781** par un espace*.

Attention

scanf lit les données dans le buffer d'entrée et non directement au clavier. En particulier, le dernier **\n** est laissé dans le buffer, ce qui peut entraîner des dysfonctionnements dans des saisies ultérieures. (*Voir FICHE 3 Entrées-Sorties*).