printf

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

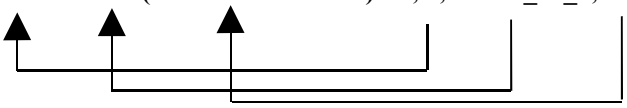
L'instruction **printf** permet d'écrire sur l'écran des messages et/ou le contenu de variables. Elle se présente de la manière suivante :

printf ("*chaine_de_format*", *variable_1_ou expr1* , *variable_2_ou_expr2* , …) ;

où ***variable_1_ou_expr1*** , ***variable_2_ou_expr2*** sont les variables ou expressions (mentionnées dans l'ordre d'affichage) dont on veut afficher le contenu ou le résultat, et ***chaine_de_format*** est le message qui doit être affiché à l'écran. Les emplacements dans ce message, où doivent figurer les contenus des variables, sont matérialisés par des « formats » de la forme **%…** décrivant de quel type est la variable et comment elle doit être affichée. De plus, des séquences spéciales commençant par \ symbolisent des sauts de lignes, de pages, des tabulations, …

Exemple :

```
printf("\tLa racine carree de %d est %.4f (avec %d decimales)\n", i , racine_de_i , 3+1 ) ;
```



Cette instruction affiche après une tabulation horizontale (**\t**) , le message : **La racine carree de** suivi du contenu de la variable **i** considéré comme un entier (**%d**), puis **est** suivi du contenu de la variable **racine_de_i** considéré comme un nombre à virgule flottante (**%f**), mais que l'on arrondira à 4 chiffres après la virgule (**%.4f** au lieu de **%f**) ; puis **s'affiche (avec** suivi du nombre de décimales (résultat de l'expression **3+1**) considéré comme un entier (**%d**) puis **decimales)** enfin un saut de ligne (avec retour en début de ligne suivante) est effectué (**\n**).

Attention Les formats **%…** doivent être placés dans la chaîne de format dans le même ordre que les variables qu'ils sont censés remplacer. Si une même variable doit être affichée plusieurs fois, elle doit également être mentionnée plusieurs fois après la chaîne de format.

➔ *Règle : autant de formats que de variables ou expressions.*

La syntaxe de **printf** implique que les caractères **'** , **"** , **%** et \ soient considérés comme des caractères spéciaux et ne soient pas affichés. Pour néanmoins les afficher :

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

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

➔ Séquences dans la chaîne de format commençant par \

Séquence	**Caractère**
n	saut de ligne avec retour en début de ligne suivante
t	tabulation (horizontale)
b	recul d'un caractère sans effacement
r	retour-chariot (retour en début de ligne courante)
f	saut de page
a	signal sonore
\'	'
\"	"
\\	\
ddd	caractère de code ASCII octal *ddd* (ex : **044** affiche **$**)
x*ddd*	caractère de code ASCII hexadécimal *ddd* (ex : **x024** affiche **$**)

➔ Formats

Format	***Affiche sous la forme d'un(e)***	***Déclaration de la variable affichée***
%d , **%i**	entier décimal	**int** ou **short , char**
%o	entier octal non signé	**int , unsigned int , short** ou **unsigned short , char**
%x , **%X**	entier hexadécimal non signé	
%u	entier décimal non signé	**unsigned int** ou **unsigned short , char**
%hd , **%hi**	entier décimal court	**short , char**
%ho	entier octal court non signé	**short** ou **unsigned short , char**
%hx , **%hX**	entier hexadécimal court	
%hu	entier décimal court non signé	**unsigned short , char**
%ld , **%li**	entier décimal long	**long**
%lo	entier octal long non signé	**long** ou **unsigned long**
%lx , **%lX**	entier hexadécimal long non signé	
%lu	entier décimal long non signé	**unsigned long**
%c	caractère	**char**
%f	nombre à virgule flottante	**float** ou **double**
%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 au format **%f** ou aux formats **%e** ou **%E** suivant l'exposant	
%Lf	nombre à virgule flottante	**long double**
%s	chaîne de caractères	**char ***
%p	pointeur (adresse d'une variable)	**… ***
%n	nbre de caractères déjà affichés	
%%	**%**	

Notation scientifique : 1 chiffre avant la virgule et décimales suivies de **e** (**%e** ou **%g**) ou de E (**%E** ou **%G**) et de l'exposant.
Affichage en hexadécimal : **%x** affiche les lettres A à F en minuscules, et **%X** les affiche en majuscules.

Attention : l'utilisation d'un format **%d , %i , %o , %x , %X , …** pour l'affichage d'un **char** fournit 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 résultats tronqués, voire aberrants.

Ex : **printf("%d est un nombre reel\n" , 3.14) ;**

affiche : `-31457 est un nombre reel`

car on a voulu afficher un **float** avec le format **%d** réservé aux entiers ou caractères.

Exemple :

```
int   i=123 ; float  f1=314.0 , f2=0.0000314 , f3=3.14E6 , f4=314.0e-5 ;
short   s=123 ; long  l=123456  ;
char   c='K' ;

printf("int :\tdec = %d\t\tunsigned = %u\n\t\toct = %o\t\thex = %X\n\n", i , i , i , i) ;
printf("short :\tdec = %hd\t\tunsigned = %hu\n\t\toct = %ho\t\thex = %hx\n\n", s , s , s , s) ;
printf("long :\tdec = %ld\t\tunsigned = %lu\n\t\toct = %lo\t\thex = %lx\n\n", l , l , l , l) ;
printf("float = %f\t%%e = %e\t%%E = %E\n%%g = %g\t%%G = %G\n\n", f1 , f1 , f2 , f1 , f2) ;
printf("float = %f\t%%e = %e\t%%E = %E\n%%g = %g\t%%G = %G\n\n", f3 , f3 , f4 , f3 , f4) ;
printf("char :\tcaract = %c\t\tcode ASCII = %d\n\n", c , c) ;
```

➔ *Résultat*

```
int : dec = 123         unsigned = 123
      oct = 173         hex = 7B

short :     dec = 123         unsigned = 123
            oct = 173         hex = 7b

long :      dec = 123456            unsigned = 123456
            oct = 361100            hex = 1e240

float = 314.000000      %e = 3.140000e+002      %E = 3.140000E-005
%g = 314 %G = 3.14E-005

float = 3140000.000000  %e = 3.140000e+006      %E = 3.140000E-003
%g = 3.14e+006    %G = 0.00314

char :      caract = K        code ASCII = 75
```

➔ Largeurs, précisions et alignements

Il est possible de spécifier dans un format, le nombre minimal de caractères ou de chiffres à afficher, de choisir une précision d'affichage, un alignement, …

Pour ce faire, on intercale entre **%** et la clé de formatage (**d , i , hd , f , c , s , …**) une spécification de la forme :

#+-0<largeur_minimale>.<précision>

où chaque champ (**# , + , - , 0 ,** ***<largeur_minimale>*** **,** ***<précision>***) est optionnel. Si ***<précision>*** est présente, le **.** qui la précède est obligatoire.

Effets sur les formats	**%d , … , %o , %x, …**	**%f , …**	**%s**
<largeur_minimale>	nb minimal de positions à afficher, cadrage à droite, complétion par des espaces sauf si ***<largeur_minimale>*** non spécifiée		
0	complétion par des **0** au lieu d'espaces		
<précision>	nb minimal de chiffres à afficher, complétion par des **0**	nb de décimales à afficher, avec arrondi éventuel ou complétion par des **0** ; si ***<précision>*** est explicitement nulle (**.** présent) , le **.** du nombre n'est pas affiché	nb maximal de caractères à afficher, avec troncature à droite
#	les entiers octaux sont précédés de **0** ; les entiers hexadécimaux sont précédés de **0x**	le **.** du nombre est toujours affiché	
+	le signe (+ ou **-**) est toujours affiché		
-	cadrage à gauche		

Exemples

Instruction	***Affichage***
printf("*%1d*%4d*%04d*%-3d*\n",17,17,17,17) ;	`*17*  17*0017*17 *`
printf("*%4f*%8f*%08f*%-3f*\n",3.141,3.141,3.141,2.8) ;	`*3.141*  3.141*0003.141*2.8 *`
printf("*%3s*%12s*%06s*%-04s*\n","abac","abac","az","az");	`*abac*       abac*0000az*az00*`
printf("*%5.3f*%11.6f*%.2f*\n",3.1415,3.1415,3.1485);	`*3.142*  3.141500*3.15*`
printf("*%5.0f*%5.f*%#5.f*%#5.4o*\n",314.89,314.,314.0,123);	`*  315*  314* 314.* 0173*`
printf("*%5.5d*%8.5d*%+8.5d*%#x*\n",314,314,314,123);	`*00314*  00314*  +00314*0x7b*`
printf("*%6.6s*%6.3s*%.3s*\n","abcdef","abcdef","abcdef");	`*abcdef*  abc*abc*`