Institut Supérieur d'Informatique et Mathématiques de Monastir
Département d'Informatique
Section : Licence Informatique,
Niveau : 1^{ere} Année
A.U : 2025-2026

Matière : Atelier de Programmation 1
Travaux Pratiques N°1

1 Priorité des opérateurs

Priorité	Opérateur	Ordre d'évaluation
1 (la plus forte)	()	→(de gauche à droite)
2	!, ++, - -, + unaire, - unaire	←(de droite à gauche)
3	*, /, %	→
4	+, -	→
5	$<, \leq, >, \geq$	→
6	==, !=	→
7	&&	→
8	||	→
9 (la plus faible)	=, +=, -=, *=, /=,	←

2 Fonctions arithmétiques standards

Les fonctions suivantes sont prédéfinies dans la bibliothèque standard <math>. Pour pouvoir les utiliser, le programme doit contenir la ligne : # include <math.h>
Type des données : Les arguments et les résultats des fonctions arithmétiques sont du type double.

Nom	Explication
log(X)	logarithme naturel
log10(X)	logarithme à base 10
exp(X)	fonction exponentielle
pow(X,Y)	X exposant Y
sqrt(X)	racine carrée de X
fabs(X)	valeur absolue de X
floor(X)	arrondir en moins
ceil(X)	arrondir en plus
fmod(X,Y)	reste rationnel de X/Y (même signe que X) pour X différent de 0
sin(X) cos(X) tan(X)	sinus, cosinus, tangente de X
asin(X) acos(X) atan(X)	arcsin(X), arccos(X), arctan(X)

3 Les entrées-sorties

3.1 Écriture formatée de données

a) Syntaxe :

printf("<format>", <expr 1>, <expr 2>,..... , <expr n>)

b) Rôle :
Elle permet d'afficher sur l'écran, du texte, des valeurs de variables ou des résultats d'expressions dans le format choisi.
Le format est une chaîne de caractères qui peut contenir des séquences d'échappement et des codes de format, à raison d'un code par expression. Les codes de format commencent toujours par le symbole % et se terminent par un ou plusieurs caractères qui indiquent le format d'affichage

Symboles de conversion :
Il indique la nature de la conversion de type à effectuer.

Symbole	**Objet de donnée**
d, i	Entier relatif
u	Entier naturel (unsigned)
o	Entier exprimé en octal
x , X	Entier exprimé en hexadécimal
c	caractère
f	Réel en notation décimale
e, E	Réel en notation exponentielle
s	Chaîne de caractères

Les caractères de contrôle

Séquence	**Signification**
$\backslash n$	Génère une nouvelle ligne
$\backslash t$	Pose une tabulation horizontale
$\backslash f$	Provoque un saut de page
$\backslash a$	Déclenche un signal sonore
$\backslash r$	ramène le curseur au début de la ligne courante.

3.2 Lecture formatée de données

a) Syntaxe :

scanf("<format>", <&var1>, <&var 2>, ... , <&var n>)

b) Rôle :
Elle permet de lire des données à partir du clavier, dans le format spécifié. Les valeurs lues par scanf() sont rangées dans les adresses spécifiées. L'adresse (adr) d'une variable est indiquée par le nom de la variable précédé du signe (&).

Symbole de conversion :
Mêmes symboles que ceux de printf().

Exercice 1

Saisir et exécuter le programme suivant :

```
# include <stdio.h>
void main()
{
     printf("mon premier programme C") ;
}
```

Modifier le programme en changeant la structure de la fonction "printf" (d'après le tableau ci-dessous) puis compléter le tableau :

Instruction	**Résultat d'exécution**
printf("mon premier programme C") ;	
printf("mon\npremier\nprogramme C") ;	
printf("mon\tpremier\tprogramme C") ;	
printf("mon premier programme C\b\b\b\b\bm ") ;	
printf("mon\rpremier\rprogramme C") ;	

Exercice 2

Saisir et exécuter le programme suivant :

```
# include <stdio.h>
void main()
{
    int i = 6 ;   /* déclaration et initialisation de la variable i*/
    printf("%d\n", i) ;
}
```

Modifier le programme en changeant la structure de la fonction printf (d'après le tableau ci-dessous) puis compléter le tableau :

Instruction	**Résultat d'exécution**
printf("%d\n", i) ;	
printf("%d\n", 4+2) ;	
printf("la somme 4+2 donne %d\n", 4+2) ;	
printf("%d + %d donne : %d\n", i, 3, i+3) ;	
printf("%d donne en octal %o et en hexadécimal %x ou %X\n", 90, 90, 90, 90) ;	

Exercice 3

Saisir et exécuter le programme suivant :

```
# include <stdio.h>
void main()
{
    char x = 'b', y = 'A' ;
    printf("%c et %c sont des caractères", x, y) ;
}
```

Modifier le programme en changeant la structure de la fonction printf (d'après le tableau ci-dessous) puis compléter le tableau :

Instruction	**Résultat d'exécution**
printf("%c et %c sont des caractères\n", x, y) ;	
printf("%c et %c ont les codes ASCII %d et %d\n", 'b', 'A', 'b', 'A') ;	
printf("%c et %c ont les codes ASCII %d et %d\n", 98, 65, 98, 65)	

Exercice 4

Saisir et exécuter le programme suivant :

```
# include <stdio.h>
void main()
{
    float x ;
    printf("Donner x: ");
    scanf("%f", &x) ;
    printf("%f", x) ;
}
```

1. Tester ce programme avec les valeurs de x données ci-dessous et compléter le tableau :

Valeur de x	**Résultat d'exécution**	**Valeur de x**	**Résultat d'exécution**
3.4567		0.000012345	
12.3456789		1e-10	

2. Modifier l'instruction printf("%f", x) en printf("%e", x) ; et réexécuter le programme avec les valeurs suivantes de x :

Valeur de x	**Résultat d'exécution**	**Valeur de x**	**Résultat d'exécution**
3.4567		123.456789E8	
123.45		-123.456789E8	

Exercice 5

Saisir le programme suivant :

```
# include <stdio.h>
void main()
{
    int x;  float y ;
    printf("Donner x: ");
    scanf("%d", &x) ;
    printf("Donner y: ");
    scanf("%f", &y) ;
    printf("%3d\n", x) ;
    printf("%10f\n", y) ;
}
```

1. Tester ce programme avec les valeurs de x et de y données ci-dessous et compléter le tableau :

	Valeur de x	**Résultat d'exécution**	**Valeur de y**	**Résultat d'exécution**
1	17		1.2345	
2	6		12.345	
3	5432		1.2345E5	
4	-4567		0.000012345	

2. Modifier les instructions printf("%10f\n", y) et printf (%3d\n", x) selon le tableau donné ci-dessous et réexécuter le programme avec les valeurs suivantes de x et de y : y = 100.123 et x = 217 (ou ˜ : désigne un espace)

Instruction	**Résultat d'exécution**	**Instruction**	**Résultat d'exécution**
1-printf("%.2f\n", y)		printf("%˜ 3d\n", x)	
2-printf("%011.6f\n", y)		printf("%5.5d\n", x)	
3-printf("%5.0f\n", y)		printf("%8.5d\n", x)	

Exercice 6

Quels résultats fournit le programme suivant :

```
#include <stdio.h>
void main()
{   int i, j, n;
    i = 0; n=i++;
    printf("A : i=%d  n=%d \n", i, n);
    i = 10; n=++i;
    printf("B : i=%d  n=%d \n", i, n);
    i = 20; j=5; n=i++ *++j;
    printf("C : i=%d  j=%d n=%d \n", i, j, n);
    i = 15; n=i+=3;
```

```
    printf("D : i=%d  n=%d \n", i, n);
    i = 3; j = 5; n=i+=--j;
    printf("E : i=%d  j =%d n=%d \n", i, j, n);
}
```

Exercice 7

Écrire un programme C permettant de (d') :

1. Déclarer 3 variables w, x et y de type double et 3 autres variables a, b et c de type int.
2. Initialiser les six variables à partir du clavier en utilisant les fonctions "printf" et "scanf" .
3. Déclarer une variable z de type double et une autre variable de type int. Ensuite, traduire les opérations mathématiques suivantes en langage C en affichant, à chaque fois, le résultat sur écran.

$d = \|a\|$	$z = \|w\|$
$z = \sqrt{w}$	$z = \sqrt{a}$
$z = e^w$	$z = ln(x)$
$z = ln(y)$	$d = \|a - bc\|$