Université Hassan II
Faculté des Sciences Ain Chock – Casablanca
Département Mathématique et Informatique

Série de TD N° 1 Notions de base / Variables / Constantes Fonctions d'entrée-sortie	Informatique INFAPP S1

Exercice 1

Remplissez le tableau suivant :

Déclaration		Type	Taille (octet)	Format (%)	Exemples de valeurs
char		Caractère	1	%c	'A' , ' ?', -90, $[-2^7, 2^7-1]$
unsigned char		Caractère	1	%c	$[0, 2^8-1]$
short		Entier	2	%hd	$[-2^{15}, 2^{15}-1]$
unsigned short		Entier	2	%hu	$[0, 2^{16}-1]$
int	CPU 32 bits	Entier	2	%d , %i %x , %o	$[-2^{15}, 2^{15}-1]$
	CPU 64 bits	Entier	4		$[-2^{31}, 2^{31}-1]$
unsigned int	32 bits	Entier	2	%u	$[0, 2^{16}-1]$
	64 bits	Entier	4		$[0, 2^{32}-1]$
long		Entier	4	%li, %ld	$[-2^{31}, 2^{31}-1]$
unsigned long		Entier	4	%lu	$[0, 2^{32}-1]$
long long		Entier	8	%lld	$[-2^{63}, 2^{63}-1]$
unsigned long long		Entier	8	%llu	$[0, 2^{64}-1]$
float		Float	4	%f %g %e %a	3.4028235e+38
double		Double	8	%lf %lg %e %a	1.7976931157e+308
long double		Double Long	>=8	%Lf %Lg %e %a	1.1897314953e+4932

Exercice 2

Ecrire un programme qui permet de lire un entier au format décimal et de l'afficher en **Octal** et **Hexadécimal**, ainsi que sa taille dans la mémoire et son adresse.

Solution:

```c
#include <stdio.h>

void main() {

    int n;
    printf("Entrez un entier au format décimal : ");
    scanf("%d", &n);

    printf("En octal : %o\n", n);
    printf("En hexadécimal : %x\n", n);

    printf("Taille en mémoire : %d octets\n", sizeof(n));
    printf("Adresse : %d\n", &n);

}
```

Exercice 3

Ecrire un programme qui permet de lire un caractère et d'afficher son code ASCII, ainsi que sa taille dans la mémoire et son adresse. Utilisez les deux fonctions **getchar** et **scanf**.

Solution:

```c
#include <stdio.h>

void main() {
    char c;

    printf("Entrez un caractère : ");
    //scanf(" %c", &c);
    c = getchar();

    printf("Code ASCII de '%c' : %d\n", c, c);
    printf("Taille en mémoire : %d octets\n", sizeof(c));
    printf("Adresse : %p\n", &c);

}
```

Exercice 4

Écrire un programme qui permet de déclarer 4 constantes avec l'instruction **define**, et d'afficher les valeurs correspondantes, ainsi que sa taille en mémoire :

- Un entier « **E** » avec la valeur **100**
- Un réel « **R** » avec la valeur **3.14**
- Un caractère « **C** » avec la valeur **'M'**
- Une chaîne de caractères « **CH** » avec la valeur "**Bonjour**"

Solution:

```c
#include <stdio.h>

#define E 100
#define R 3.14
#define C 'M'
#define CH "Bonjour"

void main() {
    printf("E = %d, Taille : %d octets\n", E, sizeof(E));
    printf("R = %f, Taille : %d octets\n", R, sizeof(R));
    printf("C = %c, Taille : %d octets\n", C, sizeof(C));
    printf("CH = %s, Taille : %d octets\n", CH, sizeof(CH));
}
```

Exercice 5

Écrire un programme qui permet de déclarer deux constantes avec l'instruction **const**, et d'afficher les valeurs correspondantes aux formats octal, hexadécimal et décimal :

- H avec la valeur **0xf**
- O avec la valeur **010**

Solution:

```c
#include <stdio.h>

int main() {
    // Déclaration des constantes
    const int H = 0xf;  // Valeur hexadécimale
    const int O = 010;  // Valeur octale

    // Affichage des valeurs
    printf("Valeur de H :\n");
    printf("  Décimal : %d\n", H);
```

```
    printf("  Octal : %o\n", H);
    printf("  Hexadécimal : %x\n", H);

    printf("Valeur de O :\n");
    printf("  Décimal : %d\n", O);
    printf("  Octal : %o\n", O);
    printf("  Hexadécimal : %x\n", O);

    return 0;
}
```

Exercice 6

Donnez les résultats du programme ainsi que le type de la conversion utilisée :

```
#include <stdio.h>

void main() {
   int x = 22, y = 7;
   int pi_e = x / y;
   float pi_f = (float)x / y;
   int z=pi_f;
   printf("La valeur de pi_e : %d \n", pi_e);
   printf("La valeur de pi_f : %.5f \n", pi_f);
   printf("La valeur de z : %d \n", z);
}
```

Solution:

```
La valeur de pi_e : 3
La valeur de pi_f : 3.14286
La valeur de z : 3
```

Explication :

`int pi_e = x / y;` :

Cette ligne effectue une division entière. Le résultat de `22 / 7` est `3`, car les divisions entières en C tronquent la partie décimale. Ainsi, `pi_e` reçoit la valeur `3`.

`float pi_f = (float)x / y;` :

Ici, `x` est d'abord converti en `float`, ce qui permet de faire une division flottante. Le résultat de `22.0 / 7` est environ `3.14286`. Donc, `pi_f` reçoit la valeur `3.14286`.

`int z = pi_f;` :

Lorsque `pi_f` (qui est `3.14286`) est converti en `int`, la partie décimale est tronquée. Par conséquent, `z` prend la valeur `3`.