Séance 1 Entrées/Sorties et Variables

Exercice 1a : Programme bonjour

Ecrire le programme qui affiche Bonjour. à l'écran.

Un programme qui exécute des entrés/sorties utilise l'include stdio.h

L'include **stdio.h** contient les fonctions
printf → affichage
scanf → lecture

Notez la syntaxe :

include est une directive
→ la ligne commence par un **#**
→ la ligne ne se termine pas par un **;** (ce n'est pas une instruction)
→ les directives sont définies avant la fonction principale **main**
→ le nom de l'include est fourni encadré par les caractères **< >**

```
#include <stdio.h>
```

Un programme en langage C contient au moins une fonction qui est la fonction principale **main**
Cette fonction est de type **int** car elle retourne l'entier 0 quand l'exécution s'est bien passée
Les parenthèses **()** indiquent que le programme n'a pas d'arguments fournis lors de son appel
Le corps d'une fonction est délimité par des accolades **{ }**

```
int main()
{
```

Les instructions se terminent par le caractère **;**
On utilise ici la fonction **printf** pour afficher une chaîne de caractères constante qui est délimitée par les caractères **" "**
Le caractère **\n** indique un passage à la ligne

```
    printf("Bonjour.\n");
```

Un programme qui se termine normalement retourne le résultat 0

```
    return (0);

}
```

```
#include <stdio.h>
int main()
{
    printf("Bonjour.\n");
    return (0);
}
```

Exercice 1b : Programme coucou

Ecrire le programme qui affiche à l'écran :

```
* * * * * *
* coucou! *
* * * * * *
```

```
#include <stdio.h>
int main()
{
        printf("    * * * * * *\n");
        printf("    * coucou! *\n");
        printf("    * * * * * *\n");
        return (0);
}
```

```
#include <stdio.h>
int main()
{
```

Affichage de trois lignes, chaque printf se termine par l'affichage du caractère \n.
Attention aux espaces entre chaque étoile pour encadrer correctement la chaine coucou!
Cette chaîne de caractères est composée de 7 caractères, pour encadrer cette chaîne avec un espace avant et un espace après la chaîne il faut 2+7+2 caractères soit 11 caractères. Si le cadre est plein il faut 11 étoiles, si le cadre est "aéré" il faut 5 fois le couple de caractères "* " plus une étoile.

```
 printf("    * * * * * *\n");
 printf("    * coucou! *\n");
 printf("    * * * * * *\n");
 return (0);
}
```

Exercice 2 :

1. Ecrire un programme demande2entiers.c qui lit deux nombres au clavier et les affiche à l'écran. On respectera les entrées sorties suivantes :
entier 1 ? **20**
entier 1 = 20
entier 2 ? **30**
entier 2 = 30

```
#include <stdio.h>
int main()
{
```

Déclaration de 2 variables entières **m** et **n**
Cette déclaration réserve en mémoire 2 zones mémoires de 2 octets qui ont une adresse mémoire spécifique.
Les réservations en mémoire sont contigües, si m est à l'adresse $1F01 n sera à l'adresse $1F03 car m occupe 2 octets et chaque octet a son adresse.
Les zones mémoire sont définies mais le contenu de ces zones est **indéterminé**, le contenu d'une variable ou de la zone mémoire qui lui est associée sera définie par une affectation ou par la lecture d'une information qui sera rangée à l'adresse de la variable.

```
    int m, n;
```

On pose une question. On affiche une chaîne de caractères.

```
    printf("entier 1 ? ");
```

On lit la réponse. On utilise pour cela la fonction de lecture scanf.
Cette fonction a 2 paramètres.

- Le premier paramètre est le format de lecture, celui-ci est fourni entre " "
  - Chaque format commence par le caractère **%** et est suivi d'une lettre qui correspond au type de la variable associée
    - %c → type char
    - %d → type int
    - %f → type float
    - %s → type tableau de caractères
  - Attention à ne pas mettre d'espace dans le format de lecture
- Le deuxième paramètre correspond à **l'adresse de la variable** où il faut ranger l'information. L'adresse de la variable **m** est représentée par le caractère **&** suivi du nom de la variable m (Attention → Pas d'espace après le &)

Ici la fonction scanf doit lire un entier et le ranger à l'adresse de **m**

```
    scanf("%d", &m);
```

Ici on utilise la fonction printf pour afficher du texte et le contenu de la variable **m**. Le premier paramètre de la fonction contient le texte à afficher et le format d'entrée/sortie %d réservé aux entiers. Le contenu de la variable m sera afficher en lieu et place du format %d lors de l'exécution de la fonction.

```
    printf("entier 1 = %d\n", m);
    printf("entier 2 ? ");
    scanf("%d", &n);
    printf("entier 2 = %d\n", n);
    return (0);
}
```

```c
#include <stdio.h>
int main()
{
	int m, n;

	printf("entier 1 ? ");
	scanf("%d", &m);
	printf("entier 1 = %d\n", m);

	printf("entier 2 ? ");
	scanf("%d", &n);
	printf("entier 2 = %d\n", n);

	return (0);
}
```

2. Ecrire un programme demande2caracteres.c qui lit deux caractères au clavier et les affiche à l'écran. On respectera les entrées sorties suivantes.

caractère 1 ? **x**
caractère 1 = x (code ascii=120)
caractère 2 ? **z**
caractère 2 = z (code ascii=122)

Version 1

```
#include <stdio.h>
int main()
{
```

Déclaration de 3 variables de type char.
Une variable de type char est codée sur un octet.
Une variable de type char peut représenter un entier non signé compris entre 0 et 255 → 0 et 2^8-1.

Une variable de type char peut représenter un caractère. En réalité la valeur stockée dans une variable de type char est le code ASCII du caractère.
Exemple : A → 65
B → 66
a → 97
b → 98
La valeur binaire stockée est un entier codé sur un octet.

```
    char m, n, bidon;
```

On pose une question. On affiche une chaîne de caractères délimitée par des guillemets.

```
    printf("caractere 1 ? ");
```

On lit la réponse. La fonction scanf lit un caractère en provenance du clavier. Si l'utilisateur répond **x** puis appuie sur la touche entrée il fournit deux caractères. L'exécution de la première fonction scanf affecte le caractère **x** dans la variable **m.** L'exécution de la deuxième fonction scanf affecte le caractère associé à la touche entrée dans la variable **bidon** qui n'est pas utilisée dans la suite du programme, l'objectif est donc ici de se débarrasser de ce caractère afin de pouvoir réaliser plus loin la saisie d'un deuxième caractère. En effet si on ne se débarrasse pas de ce caractère, la lecture du caractère suivant pour définir la variable n utilisera le caractère associé à la touche entrée qui est toujours présent dans la zone mémoire associée au clavier.

```
    scanf("%c", &m);
    scanf("%c", &bidon);
```

On affiche la réponse. La chaîne à afficher contient les formats %c puis %d, la fonction printf devra donc afficher un caractère puis un entier. Dans les deux cas il s'agit du contenu de m, c'est le format d'affichage qui détermine l'interprétation de l'information présente dans m.

```
    printf("caractere 1 = %c (code ascii=%d)\n", m, m);

    printf("caractere 2 ? ");
    scanf("%c", &n);
    scanf("%c", &bidon);
    printf("caractere 2 = %c (code ascii=%d)\n", n, n);

    return (0);
}
```

Version 2

```
#include <stdio.h>
int main()
{
	char m, n;

	printf("caractère 1 ? ");
```

On lit la réponse. La fonction scanf lit deux caractères en provenance du clavier.
Le premier caractère est affecté à l'adresse de la variable m.
Le deuxième caractère lu (ici le caractère correspondant à la touche entrée) n'est pas affecté.

%*c signifie lecture d'un caractère sans affectation, la variable bidon n'est donc plus nécessaire.

```
	scanf("%c%*c", &m);
	printf("caractere 1 = %c (code ascii=%d)\n", m, m);

	printf("caractere 2 ? ");
	scanf("%c%*c", &n);
	printf("caractere 2 = %c (code ascii=%d)\n", n, n);

	return (0);
}
```

Version 2

```
#include <stdio.h>
int main()
{
        char m[2], n[2];
```

m est un tableau de 2 caractères composé de 2 cases indicées de 0 à 1.
m représente l'adresse de la première case du tableau.

```
        printf("caractère 1 ? ");
```

On lit la réponse. La fonction scanf lit une chaine de caractères en provenance du clavier.
Le premier caractère est affecté à l'adresse de la variable m[0].
Le deuxième caractère lu (ici le caractère correspondant à la touche entrée) est affecté dans m[1].

%s signifie lecture d'une chaine de caractères.

```
        scanf("%s", m);
        printf("caractere 1 = %c (code ascii=%d)\n", m[0], m[0]);

        printf("caractere 2 ? ");
        scanf("%s", n);
        printf("caractere 2 = %c (code ascii=%d)\n", n[0], n[0]);

        return (0);
}
```

Exercice 3 :

1) Qu'affiche le programme entierreel.c ?

```
// entierreel.c
#include <stdio.h>
int main() {
int n = 23;
float x = 456.7;
printf("n = %d\n", n);
printf("x = %f\n", x);
return (0);
}
```

n = 23
x = 456.700012

2) Comment modifier le programme pour que sa sortie soit la suivante ?
n = 23
x = 456.7

Format d'affichage numérique :

%d	→	affichage d'un entier
%**3**d	→	affichage d'un entier sur 3 positions (complété à gauche par des espaces si l'entier a moins de 3 chiffres)
%f	→	affichage d'un réel (par défaut 6 chiffres pour la partie décimale) l'approximation des réels explique les décimales affichées
%**.1**f	→	affichage d'un réel avec 1 chiffre après la virgule
%**8.2**f	→	affichage d'un réel avec 2 chiffres après la virgule sur 8 positions
%**5**d	→	si on veut aligner les unités avec le nombre précédent le point décimal occupe une position

```
#include <stdio.h>
int main() {
  int n = 23;
  float x = 456.7;
  printf("n = %3d\n", n);
  printf("x = %.1f\n", x);
  printf("n = %5d\n", n);
  printf("x = %7.1f\n", x);
  return (0);
}
```

Exercice 4:

Ecrire un programme sdpq.c qui demande deux entiers au clavier, et affiche la somme, la différence, le produit et le quotient de ces deux nombres. On respectera les entrées sorties suivantes.
m ? **7**
m = 7
n ? **3**
n = 3
somme = 10, différence = 4, produit = 21, quotient = 2

```c
#include <stdio.h>
int main() {
  int m, n;
  printf("m ? ");
  scanf("%d", &m);
  printf("m = %d\n", m);
  printf("n ? ");
  scanf("%d", &n);
  printf("n = %d\n", n);

  int s=m+n;
  int d=m-n;
  int p=m*n;
  int q=m/n;
  printf("somme = %d,", s);
  printf(" différence = %d,", d);
  printf(" produit = %d,", p);
  printf(" quotient = %d\n", q);
  return (0);
}
```

```c
#include <stdio.h>
int main() {
  int m, n;
  printf("m ? ");
  scanf("%d", &m);
  printf("m = %d\n", m);
  printf("n ? ");
  scanf("%d", &n);
  printf("n = %d\n", n);

  printf("somme = %d,", m+n);
  printf(" difference = %d,", m-n);
  printf(" produit = %d,", m*n);
  printf(" quotient = %d\n", m/n);
  return (0);
}
```

Exercice 5 :

Parmi les chaînes de caractères suivantes, lesquelles sont des identificateurs valides ?
Toto abru ti 1tien 1_tien un_tien
to?to abru_ti programmation main program
casse case cas EstCorrect est_correct

```
int main () {
int Toto;
// int to?to;
int casse;
// int case;
int cas;
//int abru ti;
int abru_ti;
//int 1tien;
//int 1_tien;
int un_tien;
int programme;
int main;
int program;
int EstCorrect;
int est_correct;
return 0;
}
```

Exercice 6 :

Parmi les déclarations de variables suivantes, lesquelles sont correctes ?
float a, a1, a2; float b = 0; real x; int switch;
char ip-v6; char ip_v6; unsigned int _a;
unsigned char c = c; unsigned char d = 'd'; char c = 32;

```
int main () {
float a,a1,a2;
float b=0;
// real x;
// int switch;
// char ip-v6;
unsigned int _a;
// unsigned char c = c;
unsigned char d = 'd';
unsigned char e = d;
char c=32;
return 0;
}
```

Exercice 7 :

Corriger le programme suivant (8 erreurs à trouver).

```
#include <stdio.c>
int main() {
int m, n;
printf("m ? );
scanf("%d", m);
printf("m = %d\n", &m);
printf("n ? ");
scanf("%n", &n);
printf("n = %d\n", n);
printf("%d + %d = %d\n", m+n);
printf("%d - %d = %d\n", m, n, difference);
return (0).
}
```

```
#include <stdio.h>
int main () {
	int m,n;
	printf("m ? ");
	scanf("%d",&m);
	printf("m = %d\n", m);
	printf("n ? ");
	scanf("%d",&n);
	printf("n = %d\n", n);
	printf("%d + %d = %d\n", m, n, m+n);
	printf("%d - %d = %d\n", m, n, m-n);
	return (0);
}
```

Exercice 8 :

Ecrire un programme etudiants.c permettant à l'utilisateur d'entrer un nombre d'étudiants inscrits et un nombre d'étudiants présents et affichant le pourcentage d'étudiants présents. La sortie du programme doit correspondre à l'exécution ci-dessous. On suppose que l'utilisateur entre des valeurs strictement positives. On affichera le pourcentage avec un chiffre après la virgule.

nombre d'étudiants inscrits ? ***400***
nombre d'étudiants inscrits = 400
nombre d'étudiants présents ? ***250***
nombre d'étudiants présents = 250
pourcentage de présences = 62.5 %

```
#include <stdio.h>
int main () {
	int inscrits, presents;
	printf("nombre d'étudiants inscrits ? ");
	scanf("%d",&inscrits);

	printf("nombre d'étudiants inscrits = %d\n",inscrits);

	printf("nombre d'étudiants présents ? ");
	scanf("%d",&presents);

	printf("nombre d'étudiants présents = %d\n",presents);

	printf("pourcentage de présences = %.1f %\n",100.0*presents/inscrits);
	return (0);
}
```