

TD 1
Programmation C L2.1
Variables, instructions conditionnelles

Dans cette première séance de travaux dirigés, nous abordons les points suivants :

- les entrées/sorties en C ;
 - les déclarations de variables et de constantes ;
 - les types de base ;
 - les structures conditionnelles *if* et *if ... else*.
-

1. Déclaration de variables, les types en C

Dans tout programme, le programmeur fait appel à des données. Ces données peuvent prendre des valeurs qui peuvent évoluer pendant toute l'exécution du programme. On parle alors de *variables*. Une variable a un *type* qui détermine les valeurs que la variable peut prendre ainsi que les opérations pouvant s'y appliquer. Nous utiliserons les types standards de C suivant : `int`, `float` et `char`. Commentez le programme suivant :

```
#include <stdio.h>
int main(void){
    int i,j ;
    float x,y,z ;

    i=3;
    j=5;
    x=3.0;
    printf(" i= %d j=%d somme = %d \n" , i , j , i+j );
    i=j / 2; y=x / 2; z=j / 2;
    printf(" i= %d \n   y=%f   z =%f \n" , i , y , z );
    return 0;
}
```

2. Lecture au clavier

La lecture au clavier se fait grace à la fonction `scanf`. Il faut lui indiquer exactement le type des données à lire :

`%d` pour `int`, `%f` pour `float`, `%c` pour `char`.

Pour les deux premiers types, les *espaces*, *tabulations* et *passages à la ligne* sont considérés comme des séparateurs.

De plus chaque variable à lire doit être précédée du symbole `&` qui permet d'obtenir l'adresse.

Ainsi `scanf("%d",&i);scanf("%f",&x);scanf("%d",&j);` lit successivement au clavier un entier affecté à la variable `i`, un réel affecté à la variable `x` et un entier affecté à la variable `j` (on peut regrouper en une seule instruction `scanf("%d%f%d",&i,&x,&j)`, mais cela demande plus de précaution). Comme pour `printf`, il doit y avoir concordance entre le type et le nombre des valeurs à lire et les variables fournies à la fonction. Ecrire un programme qui demande un entier à l'utilisateur, le lit au clavier puis l'affiche entre deux lignes vides.

3. Manipulation de nombres

Ecrire un programme lisant quatre variables de type `int`, affiche ensuite chaque valeur sur une ligne, puis leur somme et leur moyenne sur la ligne suivante.

Par exemple, si l'utilisateur entre les valeurs :

12 3 14

5

Le programme affiche :

12

3

14

5

somme = 34, moyenne = 8.5

4. Structures conditionnelles

Ecrire un programme qui lit un entier et l'affiche s'il est positif.

Ecrire un programme qui lit un entier et affiche un message indiquant sa parité.

5. Echange de valeurs

Ecrire une suite d'instructions échangeant le contenu de deux variables `a`, `b` de type `int` entrées par l'utilisateur..

En déduire un programme qui effectue l'échange si `a` est inférieur à `b`, et qui sinon augmente de 10 la valeur de `b`. On affiche les nouvelles valeurs de `a` et de `b`.

6. Étude du nombre de solutions d'une équation du second degré

On veut déterminer le nombre de solutions d'une équation du second degré $ax^2 + bx + c = 0$.

Nous envisagerons les cas suivants :

- si $a = 0$ et $b = 0$, l'équation est dégénérée (0 ou une infinité de solutions);
- si $a = 0$ et $b \neq 0$, il y a une racine;
- sinon, on utilise le discriminant $b^2 - 4ac$ pour déterminer le nombre de solutions
 - si le déterminant est négatif, pas de racines réelles ;
 - si le déterminant est nul, une racine double ;
 - si le déterminant est positif, deux racines ;

Ecrire un programme qui lit les coefficients `a`, `b` et `c`, et qui calcule le nombre de solutions de l'équation du second degré associée.