Séance 4 : Opérateurs de comparaison, d'égalité, opérateurs logiques et instruction « if »

Opérateurs de comparaison, opérateurs d'égalité

Les opérateurs de comparaison sont :

```
<     <=    >     >=
```

Ce sont des opérateurs binaires, ils renvoient une expression de type « int » qui vaut 1 si la comparaison est vraie et 0 sinon. De manière générale en C, une expression de type « int » est considérée comme **fausse si elle vaut 0 et vraie sinon**. Ces opérateurs ont tous la même priorité (voir la fiche 2).

Les opérateurs d'égalité sont :

```
==    !=
```

De même que pour les opérateurs de comparaison, ils renvoient une expression de type « int » qui **vaut 0 si la comparaison est fausse et 1 sinon**. Les opérateurs d'égalité ont une priorité immédiatement inférieure aux opérateurs de comparaison.

Opérateurs logiques

L'opérateur «`&&`» est l'opérateur « et » logique. L'opérateur «`||`» est l'opérateur « ou » logique. Ils se combinent avec des opérateurs de comparaison et d'égalité afin de permettre de réaliser des conditions évoluées. Ainsi avec :

```
int i = 1,j = 1;
float a = 0.5;
float b = 2.0;
```

On peut évaluer les expressions suivantes :

Expressions	*Résultats*
`i < j`	faux
`( a < b) && ( i == j)`	
`( a <= i) || ( i < a)`	
`( i <= j) && ( a >= b) || ( j < b)`	
`( i != j) || ( a >= b)`	

Pour pouvoir compléter le tableau précédent, ils vous manquent cependant quelques informations. L'opérateur `&&` est prioritaire sur l'opérateur `||`.

Il existe un dernier opérateur logique, l'opérateur unaire « ! » de négation. Il retourne 1 (vrai) si l'expression qui le suit est fausse et 0 sinon. Ainsi les expressions :

```
!( i == j)
i != j
! i
! (a < b)
```

sont toutes fausses (donc valent 0). Les deux premières expressions de la liste sont d'ailleurs totalement équivalentes. Il y a souvent plusieurs manières d'écrire ce type d'expressions, on privilégiera à chaque fois la lisibilité du code.

Instruction « if »

L'instruction « if » permet de conditionner l'exécution d'une instruction ou d'un bloc d'instructions à une expression de type entière. Si l'expression est vraie (donc différente de 0) l'instruction est exécutée.

```
a = 10;
if( a > 0)
     printf( "a est positif\n");

if( a < 0)
     printf( "a est negatif\n");
```

Si l'on veut exécuter plusieurs instruction il faut créer un bloc, ainsi après :

```
a = 10;
if( a > 0)
{
     printf( "a est positif\n");
     a = -a;
}
```

a vaudra -10 et après :

```
a = 10;
if( a > 0)
     printf( "a est positif\n");
     a = -a;
```

a vaudra toujours 10 car seul l'instruction suivant directement l'instruction « if » est exécutée conditionnellement à l'expression, un bloc étant considéré comme une seule instruction.
Il est fortement conseillé de <u>toujours utiliser des accolades</u>, même pour une seule instruction.

Exercice 1

Ecrire une programme qui demande à l'utilisateur de rentrer une date (3 entiers). Vérifier que l'utilisateur a bien rentré une date (jour entre 1 et 31, mois entre 1 et 12, année postive) et dire à l'utilisateur s' il s'est trompé.

Clause « else »

La clause « else » est utilisée en complément de l'instruction « if ». Elle permet d'exécuter des instructions complémentaires si l'expression est fausse :

```
a = 10;
if( a >= 0)
     printf( "a est positif\n");
else
     printf( "a est négatif\n");
```

La clause « else » permet aussi d'enchainer les conditions (qui s'excluent mutuellement) :

```
if( )
     ...
else if( )
     ...
else if( )
     ...
else
     ...
```

Exercice 2

Ecrire une programme qui demande à l'utilisateur de saisir une température et écrire sur la sortie standard la phase de l'eau à cette température (solide, liquide ou gazeuse).

Exercice 3

Ecrire une programme qui demande à l'utilisateur de rentrer un jour et un mois et donner le nom du jour (lundi, mardi, mercredi...) à cette date pour l'année en cours. Pour cela on calculera dans un premier temps à partir de la date saisie par l'utilisateur le nombre de jour écoulé depuis le premier janvier.

Exercice 4

Ecrire le même programme que l'exercice 3 qui soit de plus fonction de l'année (on demande que le programme fonctionne à partir de l'année 1970).

Test ternaire

Le test ternaire permet d'obtenir en une seule instruction une valeur variant avec le résultat d'une condition. Il utilise l'opérateur ternaire « ? : ».

Sa forme est : condition ? résul1 : résul 2

Cette expression a comme valeur « résul1 » si la condition est vraie et « résul2 » sinon.

Exemple :

```
int a,b=2,c=2;

a=((b==c)?5+c:9);
```

A l'issue du test ternaire, a contiendra 7 (= 5+c = 5+2).

Ce test ternaire équivaut à :

```
if (b == c)
{
     a = 5+c;
}
else
{
     a = 9;
}
```