Travaux Pratiques N°2
Les expressions et les opérateurs

Objectif :

Apprendre la manipulation des expressions en utilisant les principaux opérateurs offerts par le langage C tels que :

- l'opérateur d'affectation simple = et composé +=, -=, /=, *=, %=…
- les opérateurs arithmétiques +,- , *, / et %
- les opérateurs de comparaison <, <=, >, >=, == et !=
- les opérateurs d'incrémentation ++ et de décrémentation --

I. Les expressions

Une expression est constituée de variables et de constantes reliées par des opérateurs. En C, il existe 3 types d'opérateurs : unaires (un seul opérande), binaire (2opérandes) et ternaire (3 opérandes).Les principales classes d'opérateurs sont : les opérateurs arithmétiques, d'affectation, de comparaison et logiques.

II. Les opérateurs

II.1. Les opérateurs arithmétiques

Le langage C connaît deux opérateurs arithmétiques unaires : + et- et cinq opérateurs binaires : addition (+), soustraction (-), division (/), multiplication (*), et modulo (%) (reste de la division entière)

II.2. Les opérateurs d'affectation

L'affectation simple est effectuée par l'opérateur (=).

Pour tous opérateur arithmétique binaire Δ (-,+,*,/…), le langage C définit un opérateur Δ= (sans espace) d'affectation composée. Ces opérateurs d'affectation composés permettent de simplifier les expressions.

Exemples :

opérateur	utilisation	équivalent
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

II.3 Les opérateurs d'incrémentation et de décrémentation

Les opérateurs (++) et (--) sont des opérateurs unaires permettant respectivement d'ajouter et de retrancher 1 au contenu de leur opérande. Cette opération est effectuée après ou avant l'évaluation de l'expression suivant que l'opérateur suit ou précède son opérande.

Exemples :

```
int i=17 , j=2, k ;
i--; /*équivalent à i= i-1*/
k=5+i++ - j;  /*équivalent à k=5+i-j puis i++*/
k=5+++i - j;  /*équivalent à i++ puis k=5+i-j*/
```

II.3 Les opérateurs de comparaison

Les opérateurs de comparaisons sont les suivants :

Opérateur	Dénomination	Effet	Exemple	Résultat (avec x valant 7)
== A ne pas confondre avec le signe d'affectation (=)!!	opérateur d'égalité	Compare deux valeurs et vérifie leur égalité	x==3	Retourne 1 si X est égal à 3, sinon 0
<	opérateur d'infériorité stricte	Vérifie qu'une variable est strictement inférieure à une valeur	x<3	Retourne 1 si X est inférieur à 3, sinon 0
<=	opérateur d'infériorité	Vérifie qu'une variable est inférieure ou égale à une valeur	x<=3	Retourne 1 si X est inférieur ou égal à 3, sinon 0
>	opérateur de supériorité stricte	Vérifie qu'une variable est strictement supérieure à une valeur	x>3	Retourne 1 si X est supérieur à 3, sinon 0
>=	opérateur de supériorité	Vérifie qu'une variable est supérieure ou égale à une valeur	x>=3	Retourne 1 si X est supérieur ou égal à 3, sinon 0
!=	opérateur de différence	Vérifie qu'une variable est différente d'une valeur	x!=3	Retourne 1 si X est différent de 3, sinon 0

II.4 Les opérateurs logiques (booléens)

Ce type d'opérateur permet de vérifiersi plusieurs conditions sont vraies:

Opérateur	Dénomination	Effet	Syntaxe
||	OU logique	Vérifie qu'une des conditions est réalisée	((condition1)||(condition2))
&&	ET logique	Vérifie que toutes les conditions sont réalisées	((condition1)&&(condition2))
!	NON logique	Inverse l'état d'une variable booléenne (retourne la valeur 1 si la variable vaut 0, 0 si elle vaut 1)	(!condition)

III. Travail demandé

Exercice 1 :
Dans chacune des instructions suivantes, on suppose que A= 27, B= 7, C=5 avant l'exécution. Ecrire un programme permettant d'afficher les valeurs de A, B et C après l'exécution de chaque instruction.

1. **C+A-8-B ;**
2. **C=A%B ;**
3. **C=A%B++ ;**
4. **C=A%++B ;**
5. **C+=++A-B ;**
6. **C+=B ;**
7. **C*=B ;**
8. **C+=--B ;**
9. **C-=C-- ;**
10. **C+=A--+--B ;**

Exercice 2 :
Que fournit ce programme ?

```
#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);
}
```