Serie 1 :

17 - 09 - 2014

Programmation I

expressions, opérateurs, priorité et associativité

exercice 1 :

12332	constante entière sous forme décimale, de type int (par défaut)
345UL	constante entière sous forme décimale de type unsigned long (type forcé)
23.4	constante réel sous forme decimale de type double (par défaut)
34.5L	constante reel sous forme décimale de type long double (type forcé)
-1.0	constante réel sous forme décimale de type double (par défaut)
0xeba	constante entière sous forme hexadecimale type int (par défaut) valeur décimale = $10 \times 16^0 + 11 \times 16 + 14 \times 16^2$ $= (3770)_{10}$
0123L	constante entière sous forme octale type long (type forcé) valeur décimale = $3 \times 8^0 + 2 \times 8 + 1 \times 8^2$ $= 83$
'\n'	const caractère type int (caractère de retour à la ligne) v.d : son code ASCII
1.23ul	incorrect (il n'y a pas de suffixe ul avec les réels)
-1.0e-1	constante réel sous forme exponentielle de type double valeur décimal = -0.1
'0'	constante caractère, valeur décimal = valeur ASCII

1r	incorrect
1e1f	constante réelle sous forme exponentielle type float (forcé) valeur décimal = $1 \times 10^1 = 10$
40000U	constante entière sous forme décimale type unsigned int (forcé)
0XEUL	constante entière sous forme hexadecimale type unsigned long (forcé) valeur décimale = 14

exercice 2 :

Remarques :

- `X++` : post incrémentation
  exemple : $y = X++ \iff \begin{cases} y = x \\ x = x + 1 \end{cases}$
- `++X` : près incrémentation
  exemple : $Z = ++X \iff \begin{cases} x = x + 1 \\ Z = x \end{cases}$
- valeur d'expression : ce qui est à gauche du signe d'affectation

1- $A += (X + 5) \iff A += 17 \iff A = A + 17 = 37$
=> valeur d'expression : 37

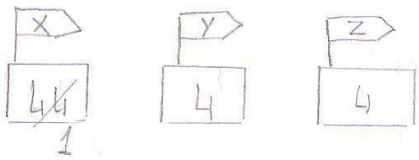
2- $A == (B = 5)$ faux
=> valeur d'expression : 0

3- $A! = (C^* = (-D)) \iff \begin{cases} C^* = (-D) \iff C = 20 \\ A! = C \to 1 \text{ (vrai)} \end{cases}$

4- $A\% = D++ \iff \begin{cases} A = A\% D \Rightarrow A = 1 \\ D = D + 1 \Rightarrow D = 3 \end{cases}$
=> valeur d'expression = 1 → valeur de A
l value = left value

5- $A = X^* (B < C) + y^* ! (B < C) \iff A = x$

Programmation TD1



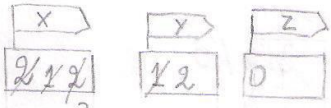
exercice 3 : 1.10.2014

```
main ()
{ int x, y, B ;
x = - 3 + 4* 5 - 6 ; printf ("%d\n", x);
x = 3 + 24 % 5 *2 ; printf ("% d\n", x );
x* = y = z = 4 ; printf ("% d\n", x);
x = y = = z ; printf ("%d\n", x);
x = = (y = z); printf ("% d\n", x);
}
```

console :

11 ✓
11 ✓
44 ✓
1 ✓
1 ✓

Rq : toute valeur non nul est considéré comme vrai (en C)



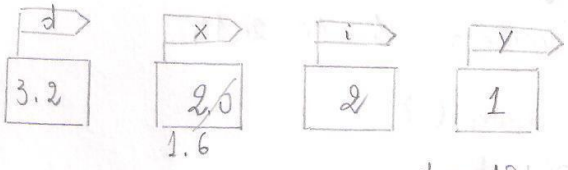
exercice 4 :

```
main ()
{ int x = 2 , y = 1 , z = 0 ;
x = x && y || z ; printf ("% d \n", x);
z + = x ++ - 1 ;
printf ("% d \n", x ); printf ("% d \n", z)
z = - x ++ + ++ y ;
printf ("% d \n", x); printf ("% d \n", z)
}
```

1 ✓
2 ✓
0 ✓
3 ✓
0 ✓

✓ exercice 5 :

```
main ()
{ double d = 3.2, x ;
  int i = 2, y ;
  x = (y = d / i) * 2 ; printf ("x = %f \t ", x); printf ("y = %f \t ", y);
  y = (x = d / i) * 2 ; printf ("x = %f \t", x ); printf ("y = %f \t", y );
  y = d * (x = 2.5 / d) ; printf ("y = %f \t", y);
  x = d * (y = ((int) 2.9 + 1.1) / d ) ; printf ("x = %f \t", x); printf ("y = %f \t", y);
}
```



⚠ converting to 'int' from 'double' ça ne compile pas !? %d ui

$x = (y = d/i) * 2$

$\Rightarrow x = (y = 3.2/2) * 2$

$\rightarrow x = (y = 1.6) * 2$

$\downarrow$ 1 partie entière

$y = d * (x = 2.5/d)$

$\Rightarrow y = 3.2 * (x = 2.5/3.2)$

$\Rightarrow y = 3.2 * (x = 0.78125)$

$\Rightarrow y = 2$

$y = (x = d/i) * 2$

$\Rightarrow y = (x = 3.2/2) * 2$

$\Rightarrow y = (x = 1.6) * 2$

$\Rightarrow y = 3$

$x = d * (y = ((int)2.9 + 1.1)/d)$

$\Rightarrow x = d * (y = (2 + 1.1)/d)$

$\Rightarrow x = 3.2 \ (y = 0)$

$\rightarrow x = 0.000000$

console :

x = 2.000000 y = 1 x = 1.600000 y = 3 y = 2

x = 0.000000 y = 0

exercice 6 :

15 - 10 - 2014

x	y	z
~~03~~ 1 8	~~02~~ ~~23~~ 5	01

```
main ()
{ int x = 03, y = 02 , z = 01;
  printf ("x ^ y & ~ z = %d \n", x ^ y & ~ z);
  x = 1 ; y = 23;
  x <<= 3 ; printf ("x = %d \n", x );
  y >>= 2 ; printf (" y = %d \n", y);
}
```

console :

```
x ^ y & ~ z = 1
x = 8
y = 5
```

$z = 0001$

$\sim z = 1110$

$y = 0010$

$y \,\&\, \sim z = 0010$

$x \,\hat{}\, y \,\&\, \sim z = 0001$

$x \,\hat{}\, y \,\&\, \sim z = 01$

$\Leftrightarrow \begin{cases} \sim z = 00 \\ y \,\&\, 00 = 00 \\ x \,\hat{}\, 00 = 01 \end{cases}$

$x = 1 = (0000\ 0001)_2$

$x << 3 = (0000\ 1000)_2$ (SHL)

$= 2^3 = 8$

$y = 23 = (0001\ 0111)_2$

$y >> 2 = (0000\ 0101)_2$ (SHR)

$= 5$

opérateurs Bit à Bit

not - and - OR - XOR (ou exclusif) - SHR (décalage à droite)
SHL (décalage à gauche)

Les opérandes sont de type entier.
Les opérations s'effectuent bit à bit suivant la logique binaire.

not ~ négation
and & Et logique
OR | } ou logique
XOR ^ } la diff : XOR : $A \hat{} B = 1 \iff \begin{cases} A = 1 \text{ et } B = 0 \\ \text{ou} \quad A = 0 \text{ et } B = 1 \end{cases}$
ou exclusif

SHR >> } Rq : ce n'est pas une rotation
SHL << }

SHR a pour opérande de gauche la valeur initiale et pour opérande de droite le nombre de bits à décaler à droite. Les bits de poids faibles sont perdus les bits de poids forts entrés (à gauche) sont à 0.

SHL a pour opérande de gauche la valeur initial et pour opérande de droite le nombre de bits à décaler à gauche. Les bits de poids fort sont perdus et les bits de poids faibles entrés (à droite) sont à 0

Rq : un décalage à gauche de K bits correspond (sauf débordement) à la multiplication par 2^k

→ un décalage à droite ($n >> k$) correspond à la division entière par 2^k si n est non signé.

SHR : on remplace les bits perdu par des 1 si n est négatif