TRAVAUX DIRIGÉS 1

Ce TD porte sur le chapitre « Les Bases du C++ », section « Les bases du langage ».

Exercice 1:

Soient les déclarations suivantes :

```
int     i(5), p(10);
double  x{1.05e1};
float   y = 2;
```

Donner le type et la valeur des expressions suivantes. On suppose que chaque ligne est exécutée seule, indépendemment des autres instructions.

```
i + p
i + x
i / p
x / y
i / y
x < y
!(x < i)
(p >= x/2 )
i * (y=5)
p * (x == 2)
```

Exercice 2:

Simplifier les différentes expressions suivantes en :

(a) retirant les parenthèses superflues ;

```
s = ( 10 + 5 )
s = ( ( 2*a ) + b)
s = ( i++ * (5 + b) )
s = ( ( 2*x ) + (z=5) )
```

(b) donnant directement l'expression finale de l'algorithme.

```
s = ( (a < b) || (a >= b) )
s = (a > b ? a : b) + (a <= b ? a : b)
```

Exercice 3:

À partir des déclarations

```
char c = '\x2e'; // Le symbole affichable '.'
int n{5};
```

Indiquer les types et valeurs des expressions suivantes :

```
(int) c
c + n
(char) n + c
char(n) + c
char{n} + c
```

```
char(n + c)
char{n + c}
(long) c + n
long(c) + n
long{c} + n
float{c} +n
```

Exercice 4:

Donner les résultats du programme suivant :

```
#include <iostream>

using namespace std;

int main ()
{
  int n, p , q;

  n = 5; p = 2;
  q = n++ + 2*p;
  cout << "q1 : \t n = " << n << "\tp = " << p << "\tq = " << q << endl;

  n = 5; p = 2;
  q = ++n + 2*p;
  cout << "q2 : \t n = " << n << "\tp = " << p << "\tq = " << q << endl;

  n = 5; p = 2;
  q = n++ + n++ ;
  cout << "q3 : \t n = " << n << "\tp = " << p << "\tq = " << q << endl;

  n = 5; p = 2;
  q = ++n + ++n ;
  cout << "q4 : \t n = " << n << "\tp = " << p << "\tq = " << q << endl;

  n = 5; p = 2;
  q = n++ < p || p++ != 3 ;
  cout << "q5 : \t n = " << n << "\tp = " << p << "\tq = " << q << endl;

  n = 5; p = 2;
  q = n++ > p || p++ != 3 ;
  cout << "q6 : \t n = " << n << "\tp = " << p << "\tq = " << q << endl;

  n = 5; p = 2;
  q = ++n == 3 && ++p == 3 ;
  cout << "q7 : \t n = " << n << "\tp = " << p << "\tq = " << q << endl;

  n = 5; p = 2;
  q = ++n == 6 && ++p == 3 ;
  cout << "q8 : \t n = " << n << "\tp = " << p << "\tq = " << q << endl;

  return 0;
}
```

Exercice 5:

Corriger les erreurs dans les codes suivants et indiquer la raison de l'erreur.

(a) Corriger les erreurs

```
if(i>0) cout << "positif" << endl
else cout << "negatif ou nul" << endl
```

(b) Indiquer les erreurs et leur cause, proposer une correction conservant le sens de l'algorithme

```
int u, v, w;
cout << "Solution 1 puis solution 2 : ";
cin >> v >> w;

cout << "Proposition :";
cin >> u;

switch(u)
{
  case 1: case 3: case 5: case 7: case 9: cout << "impair" << endl;
  case 2: case 4: case 6: case 8: cout << "pair" << endl;
  case v: cout << "BINGO" << endl;
  case w: cout << "BINGO 2" << endl;
  default: cout << "meh !" <<endl;
}
```

Exercice 6:

Écrire un code de conversion de monnaie :

(a) Demander un montant suivi d'une monnaie (un seul caractère suffit : 'e' pour euro, 'd' pour dollar)

(b) À l'aide d'une boucle if ... else produire la conversion de l'un vers l'autre

(c) On ne demande maintenant que des euros en entrée mais on demande de choisir entre trois monnaies de sortie

(d) Dans ce deuxième cas, remplacer la boucle if ... else par un switch

Exercice 7:

Écrire un code qui calcule une multiplication entre deux entiers sans utiliser l'opérateur $*$ ou $*=$

Exercice 8:

Soit un code qui, à partir d'une valeur entière définie à l'avance, demande à l'utilisateur d'entrer un entier et indique si la proposition est supérieure ou inférieure au résultat. Le programme s'arrête quand la bonne valeur est entrée.

(a) Écrire ce code avec une boucle while

(b) Transformer le code pour utiliser une boucle do ... while

Exercice 9:

Nous allons maintenant effectuer le contraire, l'ordinateur devine. Écrire un programme qui :

— Demande un nombre mystère entier $\in [1, 100]$

— Propose jusqu'à 10 fois un nombre

— L'utilisateur répondra par + ou $-$

— L'ordinateur adapte sa proposition

Il est possible de coder des algorithmes plus ou moins efficaces, commencez par construire un algorithme basique qui ne cherche pas forcément la bonne réponse, améliorez ensuite.