

# MOOC Init Prog C++

## Corrigés semaine 2

---

Les corrigés proposés correspondent à l'ordre des apprentissages : chaque corrigé correspond à la solution à laquelle vous pourriez aboutir au moyen des connaissances acquises jusqu'à la semaine correspondante.

---

### Exercice 4 : expressions conditionnelles

Cet exercice correspond à l'exercice n°4 (pages 20 et 199) de l'ouvrage [\*C++ par la pratique\* \(3<sup>e</sup> édition, PPUR\)](#).

Il s'agissait dans cet exercice de vérifier l'appartenance d'un nombre à l'intervalle  $I = [-1, 1[$ . En clair, un nombre  $x$  est dans  $I$  s'il est plus grand ou égal à -1, et s'il est strictement plus petit que 1. En C++, la même phrase s'écrit :

```
if ( (x >= -1.0) and (x < 1.0) ) {  
    // x est dans l'intervalle  
} else {  
    // x n'est pas dans l'intervalle  
}
```

#### Programme :

```
#include <iostream>  
using namespace std;  
  
int main() {  
    cout << "Entrez un réel : " ; // demande à l'utilisateur d'entrer un réel  
    double x ; // déclaration de la variable x  
    cin >> x ; // enregistre la réponse dans x  
  
    if ( (x >= -1.0) and (x < 1.0) ) {  
        cout << "x appartient à I" << endl;  
    } else {  
        cout << "x n'appartient pas à I" << endl;  
    }  
  
    return 0;  
}
```

#### Réponses du programme :

```
./intervalle  
Entrez un nombre décimal : -2.5  
x n'appartient pas à I
```

```
./intervalle  
Entrez un nombre décimal : -1  
x appartient à I
```

```
./intervalle  
Entrez un nombre décimal : 0.5  
x appartient à I
```

```
./intervalle  
Entrez un nombre décimal : 1  
x n'appartient pas à I
```

```
./intervalle  
Entrez un nombre décimal : 1.5  
x n'appartient pas à I
```

## Exercice 5 : expressions conditionnelles (2)

Cet exercice correspond à l'exercice n°4 (pages 20 et 199) de l'ouvrage [\*C++ par la pratique\* \(3<sup>e</sup> édition, PPUR\)](#).

### Programme :

```
#include <iostream>
using namespace std;

int main() {
    cout << "Entrez un réel : " ; // demande à l'utilisateur d'entrer un réel
    double x ; // déclaration de la variable x
    cin >> x ; // enregistre la réponse dans x

    if ( (not(x < 2.0) and (x < 3.0)
        or (not(x < 0.0) and not(x == 0.0) and ((x < 1.0) or (x == 1.0)))
        or (not(x < -10.0) and ((x < -2.0) or (x == -2.0)))
    ) {
        cout << "x appartient à I" << endl;
    } else {
        cout << "x n'appartient pas à I" << endl;
    }

    return 0;
}
```

### Réponses du programme :

```
./intervalle
Entrez un nombre décimal : -20
x n'appartient pas à I
```

```
./intervalle
Entrez un nombre décimal : -10
x appartient à I
```

```
./intervalle
Entrez un nombre décimal : -2
x appartient à I
```

```
./intervalle
Entrez un nombre décimal : -1
x n'appartient pas à I
```

```
./intervalle
Entrez un nombre décimal : 0
x n'appartient pas à I
```

```
./intervalle
Entrez un nombre décimal : 1
x appartient à I
```

```
./intervalle
Entrez un nombre décimal : 1.5
x n'appartient pas à I
```

```
./intervalle
Entrez un nombre décimal : 2
x appartient à I
```

```
./intervalle
Entrez un nombre décimal : 3
x n'appartient pas à I
```

```
./intervalle
Entrez un nombre décimal : 4
x n'appartient pas à I
```

## Exercice 6 : expressions arithmétiques

Cet exercice correspond à l'exercice n°5 (pages 20 et 200) de l'ouvrage [\*C++ par la pratique\* \(3<sup>e</sup> édition, PPUR\)](#).

### Programme :

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    double x; // déclaration
    cout << "Entrez un nombre réel : "; // message
    cin >> x; // lecture de x

    double resultat(0.0);

    // Expression 1
    cout << "Expression 1 : ";
    if (exp(x) == 1.0) /* Condition qui rend la division ci-dessous invalide. *
                       * On pourrait bien sûr aussi écrire (x == 0.0), mais *
                       * nous avons ici voulu expliciter la raison de ce test. */
    { cout << "indéfinie" << endl; }
    else {
        resultat = x / (1.0 - exp(x));
        cout << resultat << endl;
    }

    // Expression 2
    cout << "Expression 2 : ";
    if ( (x <= 0.0) // condition qui empêche le calcul du log
        or (x == 1.0) // condition qui empêche la division
    { cout << "indéfinie" << endl; }
    else {
        resultat = x * log(x) * exp( 2.0 / (x-1.0) ) ;
        cout << resultat << endl;
    }

    // Expression 3
    cout << "Expression 3 : ";
    if ((x*x - 8.0*x < 0.0) /* Condition rend la racine ci-dessous invalide. *
                           * On pourrait bien sûr aussi écrire : *
                           * (x > 0.0) and (x < 8.0) *
                           * mais nous avons ici voulu expliciter la raison *
                           * de ce test. */
        or (x == 2.0) /* Condition pour empêche la division. *
                       * On pourrait bien sûr supprimer cette condition qui *
                       * est incluse dans la précédente, mais nous avons à *
                       * nouveau voulu illustrer la démarche générale... *
                       * ...laquelle peut ensuite être simplifiée. */
    { cout << "indéfinie" << endl; }
    else {
        resultat = ( -x - sqrt(x*x - 8.0*x) ) / (2.0-x) ;
        cout << resultat << endl;
    }

    // Expression 4
    cout << "Expression 4 : ";
    if (x == 0.0) // pour pouvoir faire la division par x
    { cout << "indefinie" << endl; }
    else {
        resultat = x*x - 1.0/x; // calcul partiel
        if (resultat <= 0.0) // condition pour pouvoir prendre le log de cette expression
        { cout << "indefinie" << endl; }
        else {
            resultat = (sin(x) - x/20.0) * log(resultat); // calcul partiel
            if (resultat < 0.0) // condition pour prendre la racine de cette expression
            { cout << "indefinie" << endl; }
        }
    }
}
```

```
        else {
            cout << sqrt(resultat) << endl;
        }
    }
}
return 0;
}
```

### Réponses du programme :

```
./formules
Entrez un nombre réel : -1
Expression 1 : -1.58198
Expression 2 : indéfinie
Expression 3 : -0.666667
Expression 4 : indéfinie
```

```
./formules
Entrez un nombre réel : 0
Expression 1 : indéfinie
Expression 2 : indéfinie
Expression 3 : -0
Expression 4 : indéfinie
```

```
./formules
Entrez un nombre réel : 1
Expression 1 : -0.581977
Expression 2 : indéfinie
Expression 3 : indéfinie
Expression 4 : indéfinie
```

```
./formules
Entrez un nombre réel : 2
Expression 1 : -0.313035
Expression 2 : 10.2434
Expression 3 : indéfinie
Expression 4 : 1.00691
```

```
./formules
Entrez un nombre réel : 3
Expression 1 : -0.157187
Expression 2 : 8.95901
Expression 3 : indéfinie
Expression 4 : indéfinie
```

```
./formules
Entrez un nombre réel : 8
Expression 1 : -0.0026846
Expression 2 : 22.1371
Expression 3 : 1.33333
Expression 4 : 1.56522
```

---