Exercices: Série 2 - Conditions et fonctions - ICC-C 2025-2026

1. Intervalles

*Cet exercice correspond à l'exercice n°4, page 20, de l'ouvrage **C++ par la Pratique (PPUR)** (que nous résolvons en C).*

Soit $I = [-1, 1[$ dans l'ensemble des nombres réels.

Écrire le programme `intervalles.c` qui :

1. demande à l'utilisateur ou utilisatrice d'entrer un réel ;
2. enregistre la réponse dans une variable `x` (de quel **types** doit être cette variable ?) ;
3. teste si $x \in I$; si c'est le cas, affiche « `x appartient à I` » ; sinon, affiche « `x n'appartient pas à I` ».

Tester le programme en l'exécutant plusieurs fois de suite en donnant successivement à x les valeurs -2.5, -1, 0.5 et 1.5.

Exemple d'exécution du programme :

```
Entrez un réel : 1.5
x n'appartient pas à I
```

Faire de même pour l'ensemble $S = [2, 3[\cup]0, 1] \cup [-10, -2]$.

2. Expressions arithmétiques

*Cet exercice correspond à l'exercice n°5, page 21, de l'ouvrage **C++ par la Pratique (PPUR)** (que nous résolvons en C).*

Soient les expressions mathématiques suivantes :

1. $$\frac{x}{1 - e^x}$$
2. $$x \log(x)\, e^{\frac{2}{x-1}}$$
3. $$\frac{-x - \sqrt{x^2 - 8x}}{2 - x}$$
4. $$\sqrt{\left(\sin(x) - \frac{x}{20}\right) \cdot \log\left(x^2 - \frac{1}{x}\right)}$$

Écrire un programme `formules.c` qui :

1. demande d'entrer un nombre réel ;
2. enregistre la réponse dans une variable `x` ;
3. pour chacune des expressions ci-dessus :
   - teste si elle est définie pour x
   - si elle l'est, calcule et affiche le résultat ;
   - sinon, affiche « `Expression i : indéfinie` » où `i` est le numéro de l'expression.

Le but de cet exercice n'est bien sûr pas de vous faire faire des mathématiques, mais bien de la programmation ! Ne cherchez donc pas *a priori* sur le papier les domaines de définition des expressions ci-dessus. Au lieu de cela, *programmez* les tests nécessaires pour qu'elles soient définies.

Tester le programme avec les valeurs suivantes : -1, 0, 1, 2, 3 et 8.

Exemple d'exécution du programme :

```
Entrez un réel : 2
-0.313035
10.2434
Expression 3 : indéfinie
1.00691
```

Modifiez votre programme pour que chaque expression mathématiques soit définie dans une **fonction C** séparée. La fonction devra prendre `x` en paramètre, et renvoyer le résultat de l'expression si elle est définie. Si elle n'est pas définie pour `x`, renvoyer une valeur « magique », que vous pouvez définir comme :

```
const double FUN_UNDEFINED = 151.189
```

Bien sûr, ce n'est pas idéal ! Qui dit que le résultat de la fonction ne doit pas justement être 151.189 ? Il nous faudra apprendre plus de choses en C pour améliorer cela.

2.1. Indication

Pour utiliser les fonctions mathématiques comme `log`, il faut ajouter au début du programme la ligne

```
#include <math.h>
```

De plus, vous devrez ajouter l'option `-lm` *après* votre fichier source lorsque vous compilez :

```
gcc -Wall formules.c -lm -o formules
```

Vous aurez alors accès aux fonctions `log`, `sqrt`, `exp` et `sin` (entre autres).

Pour déboguer un programme utilisant les fonctions de `<math.h>` dans VS Code, vous aurez des soucis. Ouvrez le fichier `./vscode/tasks.json`, et modifiez la configuration pour `"args"` comme suit :

```
"args": [
  "-fdiagnostics-color=always",
  "-g",
  "-Wall",
  "${file}",
  "-lm",
  "-o",
  "${fileDirname}/${fileBasenameNoExtension}"
],
```

Enregistrez. Vous devriez maintenant pouvoir exécuter et déboguer votre programme avec VS Code.