Programmation Impérative – TP 1
Conditions, boucles et fonctions

Responsable
Philippe SWARTVAGHER
philippe.swartvagher@enseirb-matmeca.fr

Intervenants
Fadwa ABAKARIM
fadwa.abakarim@enseirb-matmeca.fr
Guillaume MERCIER
mercier@enseirb-matmeca.fr

2025 – 2026

Dans la majorité des exercices, les paramètres sont à passer comme arguments du programme (à récupérer dans les paramètres de la fonction `main()`). Pour chaque programme, on gérera le cas où les paramètres demandés sont mal fournis (manquants, invalides, ...).

Une correction sera disponible en ligne une fois que tous les groupes ont réalisé le TP.

1 Divison euclidienne

Écrire un programme qui prend en paramètres deux nombres entiers a et b et affiche le résultat de la division euclidienne de a par b.

Exemples d'exécutions :

```
./division 4 2
4 = 2 * 2
./division 5 2
5 = 2 * 2 + 1
```

Gérer le cas où $b = 0$.

2 Factorielle

Écrire un programme qui affiche la factorielle du nombre passé en paramètre. On fera une version récursive et une version non-recursive.

Exemples d'exécutions :

```
./factorielle 1
1! = 1
./factorielle 4
4! = 24
```

3 Affichages de formes

Écrire un programme qui affiche un triangle d'astérisques de longueur passée en paramètre.

Exemples d'exécutions :

```
./forme 1
*
./forme 4
*
**
***
****
```

Afficher ensuite un triangle « isocèle ». Le nombre passé en paramètre correpondra au nombre d'astérisques à la base du triangle.

Exemples d'exécutions :

```
./isocele 1
*
./isocele 3
 *
***
./isocele 5
  *
 ***
*****
```

Que faire si le nombre est pair ?

4 Années bissextiles

Écrire un programme qui détermine si l'année passée en paramètre est bissextile. Une année est bissextile si elle est divisible par 4 mais pas par 100, ou divisible par 400.

Exemples d'exécutions :

```
./bissextile 2018
2018 n'est pas bissextile
./bissextile 2016
```

```
2016 est bissextile
./bissextile 2000
2000 est bissextile
./bissextile 1900
1900 n'est pas bissextile
./bissextile 2023
2023 n'est pas bissextile
```

5 Nombres premiers

Écrire un programme qui détermine si le nombre passé en paramètre n est premier. On utilisera un algorithme naïf qui teste si un des nombres compris entre 2 et $n-1$ divisent n.

Exemples d'exécutions :

```
./premier 1
1 est un nombre premier
./premier 2
2 est un nombre premier
./premier 3
3 est un nombre premier
./premier 4
4 n'est pas un nombre premier
```

6 Conversions entre degrés Fahrenheit et Celsius

Écrire un programme qui réalise la conversion entre degrés Fahrenheit et Celsius suivant le premier paramètre, la valeur de la température est le deuxième paramètre. Les formules de conversion sont les suivantes :

$$T_C = \frac{5}{9}(T_F - 32) \qquad T_F = \frac{9}{5}T_C + 32$$

Exemples d'exécutions :

```
./temperature f2c 32.3
32.3°F = 0.2°C
./temperature c2f 37
37.0°C = 98.6°F
```

7 Votre version de `seq`

Écrire un programme qui fait la même chose que la commande `seq`.

Exemples d'exécutions :

```
./seq 3
1
2
3
./seq 3 5
3
4
5
./seq 3 2 8
3
5
7
```

8 PGCD

Écrire un programme qui affiche le PGCD de deux nombres entiers a et b passés en paramètre, en utilisant l'algorithme d'Euclide : $\text{pgcd}(a, b) = \text{pgcd}(b, a \mod b)$ et $\text{pgcd}(a, 0) = a$. On fera une version récursive et une version non-recursive.

Exemples d'exécutions :

```
./pgcd 17 1
1
./pgcd 30 36
6
```

9 Tables de multiplications

Écrire un programme qui affiche les tables de multiplications de 1 à 10. On veillera à ce que les nombres soient bien alignés sur la droite. On pourra utiliser le caractère tabulation (`'\t'`) pour aligner les colonnes.

Exemple d'exécution :

```
./table_multiplication
      1   2   3   4   5   6   7   8   9  10

  1   1   2   3   4   5   6   7   8   9  10
  2   2   4   6   8  10  12  14  16  18  20
  3   3   6   9  12  15  18  21  24  27  30
  4   4   8  12  16  20  24  28  32  36  40
  5   5  10  15  20  25  30  35  40  45  50
  6   6  12  18  24  30  36  42  48  54  60
  7   7  14  21  28  35  42  49  56  63  70
  8   8  16  24  32  40  48  56  64  72  80
  9   9  18  27  36  45  54  63  72  81  90
```

```
 10   10   20   30   40   50   60   70   80   90 100
```

Bonus Accepter en paramètre la taille de la table de multiplication (dans l'exemple ci-dessus, ce paramètre vaudrait 10).

10 Plus ou moins

Écrire un programme qui choisit aléatoirement un nombre entre 1 et 100 (inclus) et demande à l'utilisateur de le deviner en répondant à chaque essai si le nombre saisi est supérieur ou inférieur au nombre choisi. L'objectif est de trouver le nombre en un minimum de coups.

Pour choisir un nombre aléatoirement, on pourra utiliser la fonction `rand(3)` [1].

Exemple d'exécution :

```
./plus_ou_moins
Quel nombre ? 50
C'est plus !
Quel nombre ? 75
C'est moins !
Quel nombre ? 60
C'est plus !
Quel nombre ? 70
C'est moins !
Quel nombre ? 65
C'est plus !
Quel nombre ? 67
Bravo ! Vous avez trouvé le nombre en 6 coups !
```

Question Quelle stratégie appliquer pour trouver le nombre en un minimum de coups ?

Bonus Accepter en paramètre la borne supérieure de l'intervalle des nombres possibles.

11 Révision des multiplications

Écrire un programme qui aide à réviser les tables de multiplications.

Exemple d'exécution :

```
./revision_multiplication
4 x 6 = ? 24
Correct !
```

1. Cette notation (fréquemment utilisée) indique qu'il faut regarder la section 3 des pages de manuel : `man 3 rand`.

```
4 x 2 = ? 8
Correct !
8 x 7 = ? 39
Incorrect : 8 x 7 = 56
4 x 6 = ?
Score : 2 réponses correctes sur 3
```

Bonus Accepter en paramètre la table de multiplication que l'on souhaite réviser.

Variante Rester sur la même question tant que la réponse saisie est incorrecte.

12 Racines d'une fonction du second degré

Écrire un programme qui à partir des coefficients a, b et c d'une fonction du second degré $f : x \mapsto ax^2 + bx + c$ affiche les potentielles racines de la fonction (ie, les valeurs de x pour lesquelles $f(x) = 0$).

On peut trouver les racines à l'aide du discriminant $\Delta = b^2 - 4ac$:

- si $\Delta > 0$ alors f possède deux racines $x_0 = \frac{-b-\sqrt{\Delta}}{2a}$ et $x_1 = \frac{-b+\sqrt{\Delta}}{2a}$
- si $\Delta = 0$ alors f possède une racine double $x_0 = \frac{-b}{2a}$
- si $\Delta < 0$ alors f ne possède pas de racine réelle

Exemple d'exécution :

```
./racines 1 0 0
1.0x^2 + 0.0x + 0.0 = 0 a une solution x = -0.0
./racines 1 0 1
1.0x^2 + 0.0x + 1.0 = 0 n'a pas de solution réelle.
./racines 1 0 -0.5
1.0x^2 + 0.0x + -1.0 = 0 a deux solutions x1 = -0.7 et x2 =
   0.7
```

Il faudra peut-être compiler en ajoutant le paramètre `-lm` pour préciser qu'il faut lier l'exécutable produit avec la bibliothèque qui contient la fonction `sqrt`.

Terminé ?

Essayez d'implémenter en C les programmes nécessaires pour résoudre les défis de l'Advent of Code, dont certaines annales se trouvent à l'adresse `https://gitlab.com/phsw/adventofcode`.