Manipulation de données simples

Pour tous ces exercices, les données seront lues dans les paramètres de lancement de l'application via la ligne de commande – sauf le premier exercice qui permet justement de mettre en lumière la différence entre lecture de paramètres sur la ligne de commande et interaction avec l'utilisateur. Afin de vérifier la bonne compréhension de cette différence, ne pas hésiter à faire d'autre exercice avec les deux modes. Pour tous les programmes, si le nombre de paramètres sur la ligne de commande n'est pas correct, il faudra que le programme affiche un message d'erreur et s'arrête immédiatement.

Dans les exemples qui illustrent le fonctionnement de l'application, la commande `exec` sera remplacée par l'invocation correcte du programme selon le langage utilisé. Par exemple, pour le premier exercice :

- en C, après compilation dans un fichier nommé `calc`, il faudra écrire : `calc 13 8` ou `./calc 13 8`.
- en Java, après compilation du fichier dans lequel la classe aurait été appelée `Calc`, il faudra écrire : `java Calc 13 8`.
- en Python, si le fichier source s'appelle `calc.py`, il faudra écrire : `./calc.py 13 8` ou bien `python calc.py 13 8`.

Attention : les exercices marqués d'une étoile sont plus difficiles et à réaliser après suffisamment d'entraînement.

Exercice 1 : [Calculatrice] Écrire un programme calculant la somme de deux entiers :

```
# exec 13 8
La somme de 13 et de 8 est 21.
#
```

Écrire un programme effectuant la même tâche mais en interagissant avec l'utilisateur pour obtenir les données :

```
# exec
Indiquez le premier nombre : 13
Indiquez le second nombre : 8
La somme de 13 et de 8 est 21.
#
```

Modifier le programme pour qu'il propose de recommencer en saisissant le caractère `O` ou `N` :

```
# exec
Indiquez le premier nombre : 13
Indiquez le second nombre : 8
La somme de 13 et de 8 est 21. Recommencer ? O
Indiquez le premier nombre : 5
Indiquez le second nombre : 7
La somme de 5 et de 7 est 12. Recommencer ? N
#
```

Exercice 2 : [Entiers]

1. Écrire un programme qui dit si un entier est pair ou impair.

```
# exec 13
Le nombre 13 est impair.
# exec 6
Le nombre 6 est pair.
#
```

2. Écrire un programme qui affiche le plus grand nombre parmi deux.

```
# exec 13 8
Le nombre 13 est le plus grand.
#
```

3. Écrire un programme qui affiche le plus grand nombre parmi trois.

```
# exec 13 17 5
Le nombre 17 est le plus grand.
#
```

4. Écrire un programme qui affiche dans l'ordre trois nombre.

```
# exec 13 17 5
5, 13, 17.
#
```

5. Écrire un programme qui affiche la moyenne de 4 nombres, d'abord la moyenne entière puis la moyenne décimale.

```
# exec 13 17 5 6
10 , 10.25
#
```

Exercice 3 : [Calculatrice - suite] Modifier le programme de telle sorte que l'utilisateur choisisse l'opération à effectuer en l'indiquant après les entiers avec le code suivant : `+` pour addition, `x` pour multiplication, `-` pour soustraction, `/` pour division (entière). Attention au cas de la division par 0.

```
# exec 13 8 +
La somme de 13 et de 8 est 21.
# exec 3 8 x
Le produit de 3 et de 8 est 24.
# exec 15 2 -
La différence de 15 et de 2 est 13.
# exec 11 4 /
Le quotient de 11 par 4 est 2.
# exec 11 0 /
Erreur : le quotient de 11 par 0 n'est pas calculable.
#
```

Exercice 4 : [Suites d'entiers]

1. Écrire un programme qui affiche successivement les entiers de 1 à 10, un par ligne.

```
# exec
1
2
3
4
5
6
7
8
9
10
#
```

2. Écrire un programme qui affiche successivement les entiers de 1 à 10 sur une seule ligne, séparés par des virgules (attention au dernier caractère affiché).

```
# exec
1,2,3,4,5,6,7,8,9,10
#
```

3. Écrire un programme qui affiche successivement les entiers de 1 à N sur une seule ligne, séparés par des virgules.

```
# exec 5
1,2,3,4,5
#
```

4. Écrire un programme qui affiche successivement les entiers de N à 1 sur une seule ligne, séparés par des virgules.

```
# exec 12
12,11,10,9,8,7,6,5,4,3,2,1
#
```

5. Écrire un programme qui affiche successivement verticalement les suites croissantes et décroissantes de 1 à N.

```
# exec 7
1 // 7
2 // 6
3 // 5
4 // 4
5 // 3
6 // 2
7 // 1
#
```

6. Reprendre les questions 1 à 5 en affichant uniquement les entiers pairs.
7. Reprendre les question 1 à 5 en affichant uniquement les entiers impairs.

Exercice 5 : [Tables de calcul mental]

1. Écrire un programme qui affiche la table de multiplication par N. Le comportement attendu est le suivant :

```
# exec 7
Voici la table de multiplication de 7 :
7 x 0 = 0
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
#
```

2. Modifier le programme de telle sorte qu'il affiche la table de l'opération choisie (utiliser la lettre `x` pour la multiplication) :

```
# exec 7 +
Voici la table d'addition de 7 :
7 + 0 = 7
7 + 1 = 8
7 + 2 = 9
7 + 3 = 10
7 + 4 = 11
7 + 5 = 12
7 + 6 = 13
7 + 7 = 14
7 + 8 = 15
7 + 9 = 16
7 + 10 = 17
#
```

3. Modifier le programme afin qu'il affiche une table jusqu'à une valeur quelconque donnée.

```
# exec 7 - 12
Voici la table d'addition de 7 de 0 à 9 :
7 - 0 = 7
7 - 1 = 6
7 - 2 = 5
7 - 3 = 4
7 - 4 = 3
7 - 5 = 2
7 - 6 = 1
7 - 7 = 0
7 - 8 = -1
7 - 9 = -2
#
```

Exercice 6 : [Entiers]

1. Écrire un programme qui calcule puis affiche la factorielle d'un entier.

```
# exec 5
La factorielle de 5 est 120.
#
```

2. Écrire un programme qui calcule la somme des nombres impairs de 1 à N.

```
# exec 11
La somme des entiers impairs de 1 à 11 est 36.
#
```

3. Écrire un programme qui affiche successivement les diviseurs d'un entier.

```
# exec 12
2 | 3 | 4 | 6 | 12 |
# exec 17
17 |
#
```

4. Écrire un programme qui calcule la somme de deux entiers en n'utilisant que l'incrément de 1 comme opération (c'est-à-dire, pour une variable `x`, `x = x + 1`). Pour observer le comportement de l'application, on pourra afficher sans saut de ligne un "." à chaque tour de boucle.
5. Écrire un programme qui calcule la multiplication de deux entiers en n'utilisant que l'addition comme opération.
6. En déduire un programme qui calcule la multiplication de deux entiers en n'utilisant que l'incrément de 1 comme opération.
7. En s'inspirant du programme du point 3, écrire un programme qui affiche tous les facteurs premiers d'un entier.

```
# exec 12
2 | 2 | 3 |
# exec 3920
2 | 2 | 2 | 2 | 5 | 7 | 7 |
#
```

8. Écrire un programme qui dit si un entier est un nombre premier.

```
# exec 12
Le nombre 12 n'est pas premier.
# exec 17
Le nombre 17 est premier.
#
```

Exercice 7 : [Rationnels] Écrire des programmes qui manipule des nombres rationnels, en affichant les données en entrée et les résultats en sortie.

1. Affichage d'un rationnel sous forme de fraction et de nombre décimal (attention à la différence entre les opérations `/` et `//` sur les entiers !) :

```
# exec 5 6
5 / 6 ; 0.8333333333333334
#
```

2. Inversion d'un rationnel :

```
# exec 5 6
Inversion de 5 / 6 = 6 / 5
#
```

3. Multiplication de deux rationnels :

```
# exec 5 6 2 3
Multiplication de 5 / 6 par 2 / 3 = 10 / 18
#
```

4. Addition de deux rationnels :

```
# exec 5 6 2 3
Addition de 5 / 6 et de 2 / 3 = 27 / 18
#
```

5. Programmer d'autres opérations (addition et multiplication par un entier, simplification de la fraction, etc.)

Exercice 8 : [Jeu du "plus ou moins"] Écrire un programme pour jouer au jeu du "plus ou moins" dans la situation où c'est l'ordinateur qui fait chercher le nombre. Le nombre à trouver sera indiqué sur la ligne de commande.

```
# exec 57
Jeu du plus ou moins : vous devez trouver un nombre compris entre 1 et 100.
Votre essai : 35
Votre nombre est trop petit...
Votre essai : 78
Votre nombre est trop grand...
Votre essai : 57
Bravo ! Vous l'avez trouvé.
#
```

Modifier le programme afin qu'il affiche à la fin le nombre de tentatives effectuées. Dans l'exemple ci-dessus, l'affichage final serait alors :

```
Bravo ! Vous l'avez trouvé en 3 tentatives.
#
```

Pour aller plus loin avec ce jeu, voir le projet associé.

Exercice 9 : $\star$ [Calculatrice - pour aller plus loin] La notation dite "polonaise inverse" est une notation mathématique des expressions arithmétiques permettant de lever toute ambiguïté sur l'ordre des opérations. Elle permet en outre un traitement simple d'expressions arbitrairement longues.

Dans cette notation, les symboles des opérateurs sont placés avant les opérandes. Par exemple :

- $5+3$ s'écrit $5\ 3\ +$
- $(2+4)+3$ s'écrit $2\ 4\ +\ 3\ +$, tandis que $2+(4+3)$ s'écrit $2\ 4\ 3\ +\ +$.
- $(3\times4)+(5-(6\times2))$ s'écrit $3\ 4\ \times\ 5\ 6\ 2\ \times\ -\ +$.

Pour calculer une telle expression, il suffit de garder intelligemment dans quelques variables les valeurs et de faire les calculs au fur et à mesure du parcours des données.

```
# exec 1 2 + 4 x 3 + 3 4 + 3 - x 5 +
65
#
```