

Somme de nombres [lp02]

Exercice résolu

Karine Zampieri, Stéphane Rivière

Unisciel 

algoprog 

Version 17 mai 2018

Table des matières

1	Somme de nombres / pgsomme	2
1.1	Variante 1	2
1.2	Variante 2	3
1.3	Variante 3	4
2	Que retenir de cet exercice ?	5
3	Références générales	6

alg - Somme de nombres (Solution)



Utilise Structures répétitives ■
Durée estimée 1 h ■



Objectif

Cet exercice calcule la somme d'une série de nombres (ici des entiers) donnés par l'utilisateur. Pour qu'il soit **profitable**, essayez de le faire par vous même avant de le visualiser et/ou de télécharger les solutions.

1 Somme de nombres / pgsomme

On veut calculer la somme d'une série d'entiers. Il faut d'abord se demander comment l'utilisateur va pouvoir indiquer combien de nombres il faut additionner, ou bien quand est-ce que le dernier nombre à additionner a été entré. Voyons quelques possibilités.

1.1 Variante 1

Au départ l'utilisateur indique le nombre `n` de termes. Ce problème est proche de ce qui a été fait dans @[Afficher les entiers pairs].



Écrivez un algorithme qui saisit un entier dans `n`.
Affichez l'invite :

```
n?
```



On souhaite sommer. Il convient donc de,
Déclarez une variable `somme` (ou `s`) (entier) et de l'initialiser à zéro (car c'est une somme).



Écrivez ensuite une répétitive `Itérer` ou `Pour` qui varie de 1 à `n` (inclus) et dans laquelle, saisissez un entier dans `nombre` (ou `nb`) que l'on ajoute ensuite à la `somme`.



Finalement affichez (où `[x]` désigne le contenu de `x`) :

```
Somme = [somme]
```



Testez. Exemple d'exécution :

```
n? 6
5 -2 3 7 0 12
Somme = 25
```



Validez votre algorithme avec la solution.

Solution alg

@[pgsomme1.alg]

```
Algorithme PGSomme1
Variable j , n : Entier
Variable nombre : Entier
Variable somme : Entier
Début
  | Afficher ( "n? " )
  | Saisir ( n )
  | somme <- 0
  | Pour j <- 1 à n
```

```

|   | Saisir ( nombre )
|   | somme <- somme + nombre
|   FinPour
|   Afficher ( "Somme = " , somme )
Fin

```

1.2 Variante 2

Après chaque nombre, on demande à l'utilisateur s'il y a encore un nombre à additionner. Ici, il faut donc chercher une solution différente car on ne connaît pas au départ le nombre de valeurs à additionner et donc le nombre d'exécution de la boucle. On va devoir passer à une répétitive inconditionnelle ([TantQue](#) ou [Répéter](#)). Si on envisage de demander en fin de boucle s'il reste encore un nombre à additionner, cela donne :



Au choix :

- Écrivez un nouvel algorithme,
- Mettez la boucle [Pour](#) en commentaire et modifiez votre algorithme,

de sorte qu'il effectue la somme d'entiers en utilisant une structure [TantQue](#) ou [Répéter](#). A chaque fois, demandez si l'utilisateur souhaite finir ou continuer.



Testez. Exemple d'exécution :

```

5 Vrai -2 Vrai -3 Vrai 8 Vrai 6 Faux
Somme = 14

```



Validez votre algorithme avec la solution.

Solution alg, Répéter @[pgsomme2a.alg]

```

Algorithme PGSomme2a
Variable encore : Booléen
Variable nombre : Entier
Variable somme : Entier
Début
|   somme <- 0
|   Répéter
|   |   Saisir ( nombre )
|   |   somme <- somme + nombre
|   |   Saisir ( encore )
|   Jusqu'à Non encore
|   Afficher ( "Somme = " , somme )
Fin

```

Solution alg, TantQue

@[pgsomme2b.alg]

```

Algorithme PGSomme2b
Variable encore : Booléen
Variable nombre : Entier
Variable somme : Entier
Début
  | somme <- 0
  | Saisir ( encore )
  | TantQue ( encore ) Faire
  |   | Saisir ( nombre )
  |   | somme <- somme + nombre
  |   | Saisir ( encore )
  | FinTantQue
  | Afficher ( "Somme = " , somme )
Fin

```

1.3 Variante 3

L'utilisateur entre une valeur spéciale pour indiquer la fin. On parle de valeur **sentinelle**. Ceci n'est possible que si cette valeur **sentinelle** ne peut pas être un terme valide de l'addition. Par exemple, si on veut additionner des nombres positifs uniquement, la valeur -1 peut servir de valeur sentinelle. Mais sans limite sur les nombres à additionner (positifs, négatifs ou nuls), il n'est pas possible de choisir une sentinelle.

Comme nous souhaitons additionner des entiers et qu'un zéro ne modifie pas le calcul, nous prendrons la valeur 0 pour la sentinelle.



Au choix :

- Écrivez un nouvel algorithme,
- Modifiez votre algorithme,

de sorte qu'il définit une constante `SENTINELLE=0` puis, `TantQue` l'entier lu n'est pas la `SENTINELLE`, cumule sa valeur dans la `somme` puis saisit l'entier suivant.



Testez. Exemple d'exécution :

```

5 3 -2 7 -1 11 9 9 0
Somme = 41

```



Validez votre algorithme avec la solution.

Solution alg

@[pgsomme3.alg]

```

Algorithme PGSomme3
Constante SENTINELLE <- 0
Variable nombre : Entier
Variable somme : Entier
Début

```

```

| somme <- 0
| Saisir ( nombre )
| TantQue ( nombre <> SENTINELLE ) Faire
|   | somme <- somme + nombre
|   | Saisir ( nombre )
| FinTantQue
| Afficher ( "Somme = " , somme )
Fin

```

Solution commentée

Ici, on se base sur la valeur entrée pour décider si on continue ou pas. Il faut donc **toujours** effectuer un test après une lecture de valeur. C'est pour cela qu'il faut effectuer une lecture avant et une autre à la fin de la boucle.



Quelle valeur sentinelle prendrait-on pour additionner :

1. Une série de notes d'interrogations ?
2. Une série de températures ?

2 Que retenir de cet exercice ?



Initialiser une variable signifie lui donner une première valeur, soit par une affectation, soit par une saisie.



Il y a deux modèles de répétitives qui s'appliquent chacun à une situation différente :

- Nombre de répétitions connu avant d'acquérir les valeurs et les traiter :

```

obtenir le nombre de valeurs (nbval)
Pour j <- 1 à nbval Faire
| obtenir la valeur
| traiter la valeur
FinPour

```

- Nombre de répétitions inconnu mais arrêt de la saisie marqué par une valeur fictive dite *sentinelle* qui ne doit pas être traitée :

```

obtenir la première valeur
TantQue la valeur n'est pas celle de la sentinelle Faire
| traiter la valeur
| obtenir la valeur suivante
FinPour

```



Il faut prendre conscience qu'avec les répétitives, ordre d'écriture et ordre chronologique ne concordent plus. En effet, dans la boucle **TantQue**, une valeur saisie pendant le tour **n** sera traitée au tour **n+1**.

3 Références générales

Comprend [Chaty-PG1 :c3 :ex1] ■