

Moyenne d'entiers dans [0,20] [it04] - Exercice

Karine Zampieri, Stéphane Rivière, Béatrice Amerein-Soltner

Unisciel

sciel

algotrog

UNIVERSITÉ
HAUTE-ALSACE

Version 17 mai 2018

Table des matières

1	Moyenne d'entiers dans [0,20] / pgmoynotes	2
1.1	Stratégie de résolution	2
1.2	Moyenne d'entiers, structure tantque	3
1.3	Moyenne d'entiers, structure répéter	5
1.4	Moyenne dans [0..20], structure tantque	6
1.5	Moyenne dans [0..20], structure répéter	8
2	Références générales	9

Java - Moyenne d'entiers dans [0,20] (Solution)



Mots-Clés Schéma itératif ■

Requis Structures de base, Structures conditionnelles, Algorithmes paramétrés, Structures répétitives ■

Difficulté ●●○ (45 min) ■



Objectif

Cet exercice calcule la moyenne d'une suite d'entiers puis la moyenne d'une suite comprise dans l'intervalle [0..20]. Le programme détecte les entiers non valides et utilise la notion de sentinelle.

1 Moyenne d'entiers dans [0,20] / pgmoynotes

1.1 Stratégie de résolution

Afin d'éviter à l'utilisateur de compter le nombre d'entiers qu'il souhaite entrer, une valeur spéciale nommée **sentinelle** permet de stopper la saisie.

Voici un exemple du résultat attendu :

```
Un entier [-1==fin]? 4
Un entier [-1==fin]? 7
Un entier [-1==fin]? -5
Un entier [-1==fin]? 3
Un entier [-1==fin]? -1
==> Nombre d'entiers consideres = 4
==> Moyenne calculee = 2.25
```



Les entiers saisis seront à terme dans l'intervalle [0..20]. Outre la valeur proposée par l'exemple d'exécution, quelle(s) valeur(s) peu(ven)t identifier la sentinelle ?

Solution simple

Convient toute valeur non comprise dans l'intervalle.



En considérant l'exemple, proposez une stratégie de résolution en précisant les profils de vos procédures et/ou fonctions.

Solution simple

S'agissant d'une moyenne, il suffit d'en faire la somme s et de la diviser par le nombre n de termes. Par conséquent, deux procédures sont utiles :

- Une procédure `traiterSuite(somme, ntermes, sentinelle)` qui calcule la somme et le nombre de termes d'une suite d'entiers terminée par une sentinelle.
- Une procédure `afficherResultats(somme, ntermes)` qui affiche les résultats.



Écrivez une procédure `afficherResultats(s, nt)` qui calcule et affiche (où $[x]$ désigne le contenu de x) :

```
==> Nombre d'entiers considérés = [nt]
==> Moyenne calculée = [s / nt]
```

Le paramètre `s` (entier) désigne la somme des entiers et `nt` (entier) celui du nombre de termes considérés. **Attention**, la moyenne ne se calcule que si le nombre de termes est non nul ; dans le cas où il est nul, affichez le message suivant :

```
==> OUPS, aucune entree
```



Validez votre procédure avec la solution.

Solution Java @[pgmoynotes.java]

```
/**
 * Affiche les résultats
 * @param[in] s - somme de la suite
 * @param[in] nt - nombre de termes de la suite
 */
public static void afficherResultats(int s, int nt)
{
    if (nt != 0)
    {
        System.out.println("==> Nombre d'entiers consideres = " + nt);
        System.out.println("==> Moyenne = " + ((double)s / nt));
    }
    else
    {
        System.out.println("==> OUPS, aucune entree");
    }
}
```

1.2 Moyenne d’entiers, structure tantque



Écrivez une procédure `traiterSuite(s,nt,sentinelle)` qui traite une suite d’entiers terminée par une sentinelle `sentinelle` (entier). La procédure restitue la somme dans `s` (entier) et le nombre de termes considérés dans `nt` (entier). Affichez l’invite (où `[x]` désigne le contenu de `x`) :

Un entier [[sentinelle]==fin]?

Solution simple

On effectue les deux opérations suivantes :

- Initialisez la somme et le nombre de termes à zéro.
- Demandez un entier dans `nombre` (à déclarer), puis **tant que** l’entier saisi n’est pas la sentinelle, summez l’entier à la somme actuelle et incrémentez le nombre de termes de 1. Enfin demandez l’entier suivant afin de pouvoir sortir de la boucle.



Validez votre procédure avec la solution.

Solution Java @[pgmoynotes.java]

```
/**
 * Somme et nombre de termes d'une suite d'entiers terminée par sentinelle
 * @param[out] s - somme des termes
 * @param[out] nt - nombre de termes dans la somme
 * @param[in] sentinelle - valeur de la sentinelle
 */
```

```

public static void traiterSuite(int[] s, int[] nt, int sentinelle)
{
    Scanner input = new Scanner(System.in);
    s[0] = 0;
    nt[0] = 0;
    int nombre;
    System.out.print("Un entier [" + sentinelle + "]==fin? ");
    nombre = input.nextInt();
    while (nombre != sentinelle)
    {
        s[0] += nombre;
        ++nt[0];
        System.out.print("Un entier [" + sentinelle + "]==fin? ");
        nombre = input.nextInt();
    }
}

```



Écrivez un programme qui teste votre opération `traiterSuite` avec la sentinelle `-1` pour calculer `s` et `n` puis qui affiche les résultats.



Testez. Exemples d'exécution :

```

Un entier [-1==fin]? 4
Un entier [-1==fin]? 7
Un entier [-1==fin]? -5
Un entier [-1==fin]? 3
Un entier [-1==fin]? -1
==> Nombre d'entiers consideres = 4
==> Moyenne calculee = 2.25

```

```

Un entier [-1==fin]? -1
==> OUPS, aucune entree

```



Validez votre programme avec la solution.

Solution Java @[pgmoynotes.java]

```

/**
 * @test
 */
public static void test1()
{
    int[] somme = new int[1], ntermes = new int[1];
    traiterSuite(somme, ntermes, -1);
    afficherResultats(somme[0], ntermes[0]);
}

```

1.3 Moyenne d'entiers, structure répéter



Copiez/collez la procédure `traiterSuite` en `traiterSuite2(...)` (même en-tête), puis modifiez la procédure afin d'employer une structure **Répéter** (son équivalent en programmation) à la place de la structure `TantQue`.

Orientation

On pourra utiliser la stratégie de transformation directe (c.-à-d. en utilisant le **non** sur la condition) ou encore simplifier la condition en utilisant les lois de DE MORGAN, cf. @[Structures répétitives, Synthèse sur les boucles].



Validez votre procédure avec la solution.

Solution Java

@[pgmoynotes.java]

```
/**
 * Idem traiterSuite, emploi d'une structure Répéter
 * @param[out] s - somme des termes
 * @param[out] nt - nombre de termes dans la somme
 * @param[in] sentinelle - valeur de la sentinelle
 */
public static void traiterSuite2(int[] s, int[] nt, int sentinelle)
{
    Scanner input = new Scanner(System.in);
    s[0] = 0;
    nt[0] = 0;
    int nombre;
    do {
        System.out.print("Un entier [" + sentinelle + "]==fin]? ");
        nombre = input.nextInt();
        if (nombre != sentinelle)
        {
            s[0] += nombre;
            ++nt[0];
        }
    } while (!(nombre == sentinelle));
}
```



Testez en appelant la procédure `traiterSuite2` en place de la procédure `traiterSuite` dans votre programme. Exemples d'exécution :

```
Un entier [-1==fin]? 4
Un entier [-1==fin]? 7
Un entier [-1==fin]? -5
Un entier [-1==fin]? 3
Un entier [-1==fin]? -1
==> Nombre d'entiers consideres = 4
==> Moyenne calculee = 2.25
```

```
Un entier [-1==fin]? -1
==> OUPS, aucune entree
```



Si besoin, validez votre programme avec la solution.

Solution Java @[pgmoynotes.java]

```
/**
 * @test
 */
public static void test2()
{
    int[] somme = new int[1], ntermes = new int[1];
    traiterSuite2(somme, ntermes, -1);
    afficherResultats(somme[0], ntermes[0]);
}
```

1.4 Moyenne dans $[0..20]$, structure tantque

Le calcul de la moyenne ne devant nullement tenir compte d'entiers non valides, ce problème détecte les entiers non compris dans l'intervalle $[0..20]$ et affiche un message d'erreur si tel est le cas. Exemple d'exécution :

```
Moyenne d'entiers dans [0..20]
Un entier [-1==fin]? 11
Un entier [-1==fin]? 12
Un entier [-1==fin]? -4
==> OUPS, erreur!
Un entier [-1==fin]? 15
Un entier [-1==fin]? -1
==> Nombre d'entiers consideres = 3
==> Moyenne calculee = 12.666666667
```



Copiez/collez la procédure `traiterSuite` en `traiterSuite3(..., nmin, nmax)`, où les paramètres supplémentaires définissent l'intervalle d'entiers $[nmin..nmax]$.



Complétez la boucle `TantQue` de sorte que la procédure restitue la somme et le nombre de termes de la suite d'entiers compris dans l'intervalle défini par `nmin` et `nmax` pour être valide. Dans le cas contraire, affichez le message :

```
==> OUPS, erreur
```



Validez votre procédure avec la solution.

Solution Java @[pgmoynotes.java]

```

/**
 * Somme et nombre de termes d'une suite d'entiers dans un intervalle
 * @param[out] s - somme des termes
 * @param[out] nt - nombre de termes dans la somme
 * @param[in] sentinelle - valeur de la sentinelle
 * @param[in] nmin - valeur minimale
 * @param[in] nmax - valeur maximale
 * @pre nmin <= nmax
 */

public static void traiterSuite3(int[] s, int[] nt, int sentinelle, int nmin, int nmax)
{
    Scanner input = new Scanner(System.in);
    System.out.println("Somme d'entiers dans [" + nmin + ".." + nmax + "]");
    s[0] = 0;
    nt[0] = 0;
    int nombre;
    System.out.print("Un entier [" + sentinelle + "==fin]? ");
    nombre = input.nextInt();
    while (nombre != sentinelle)
    {
        if (nmin <= nombre && nombre <= nmax)
        {
            s[0] += nombre;
            ++nt[0];
        }
        else
        {
            System.out.println("==> OUPS, erreur!");
        }
        System.out.print("Un entier [" + sentinelle + "==fin]? ");
        nombre = input.nextInt();
    }
}

```



Testez en appelant la procédure `traiterSuite3` en place de la procédure `traiterSuite2` dans votre programme. Exemples d'exécution :

```

Moyenne d'entiers dans [0..20]
Un entier [-1==fin]? 11
Un entier [-1==fin]? 12
Un entier [-1==fin]? -4
==> OUPS, erreur!
Un entier [-1==fin]? 15
Un entier [-1==fin]? -1
==> Nombre d'entiers consideres = 3
==> Moyenne calculee = 12.6666666667

```

```

Moyenne d'entiers dans [0..20]
Un entier [-1==fin]? -5
==> OUPS, erreur!
Un entier [-1==fin]? 22
==> OUPS, erreur!

```

Un entier [-1==fin]? -1
 ==> OUPS, aucune entree valide



Si besoin, validez votre programme avec la solution.

Solution Java @[pgmoynotes.java]

```
/**
 * @test
 */
public static void test3()
{
    int[] somme = new int[1], ntermes = new int[1];
    traiterSuite3(somme, ntermes, -1, 0, 20);
    afficherResultats(somme[0], ntermes[0]);
}
```

1.5 Moyenne dans $[0..20]$, structure répéter



De même, copiez/collez la procédure `traiterSuite2` en `traiterSuite4(..., nmin, nmax)`, où les paramètres supplémentaires définissent l'intervalle d'entiers $[nmin..nmax]$.



Complétez la boucle **Répéter** de sorte que la procédure restitue la somme et le nombre de termes de la suite d'entiers compris dans l'intervalle défini par `nmin` et `nmax` pour être valide. Dans le cas contraire, affichez le message :

==> OUPS, erreur



Validez votre procédure avec la solution.

Solution Java @[pgmoynotes.java]

```
/**
 * Idem traiterSuite3, emploi d'une structure Repeter
 * @param[out] s - somme des termes
 * @param[out] nt - nombre de termes dans la somme
 * @param[in] sentinelle - valeur de la sentinelle
 * @param[in] nmin - valeur minimale
 * @param[in] nmax - valeur maximale
 * @pre nmin <= nmax
 */
public static void traiterSuite4(int[] s, int[] nt, int sentinelle, int nmin, int nmax)
{
    Scanner input = new Scanner(System.in);
    System.out.println("Somme d'entiers dans [" + nmin + ".." + nmax + "]");
    s[0] = 0;
    nt[0] = 0;
```



```

int nombre;
do {
    System.out.print("Un entier [" + sentinelle + "]==fin]? ");
    nombre = input.nextInt();
    if (nmin <= nombre && nombre <= nmax)
    {
        s[0] += nombre;
        ++nt[0];
    }
    else if (nombre != sentinelle)
    {
        System.out.println("==> OUPS, erreur!");
    }
} while (!(nombre == sentinelle));
}

```



Testez en appelant la procédure `traiterSuite4` en place de la procédure `traiterSuite3` dans votre programme. Exemple d'exécution :

```

Moyenne d'entiers dans [0..20]
Un entier [-1==fin]? 11
Un entier [-1==fin]? 12
Un entier [-1==fin]? -4
==> OUPS, erreur!
Un entier [-1==fin]? 15
Un entier [-1==fin]? -1
==> Nombre d'entiers consideres = 3
==> Moyenne calculee = 12.6666666667

```



Si besoin, validez votre programme avec la solution.

Solution Java @[pgmoynotes.java]

```

/**
 * @test
 */
public static void test4()
{
    int[] somme = new int[1], ntermes = new int[1];
    traiterSuite4(somme, ntermes, -1, 0, 20);
    afficherResultats(somme[0], ntermes[0]);
}

```

2 Références générales

Comprend [Rohaut-JV1 :c4 :xm], [Tartier-AL1 :c5 :ex13] ■