# TP : Calculs en boucle

Bienvenue dans ce second TP.   
  
Maitriser les boucles, c’est primordial. Il vous faut donc vous entrainer pour être bien sûr d’avoir tout compris. C’est justement l’objectif de ce TP où nous allons manipuler un peu les boucles.  
  
Le but de ce TP va être de créer 3 méthodes.  
  
La première va servir à calculer la sommes d'entiers consécutifs. Si par exemple je veux calculer la somme des entiers de 1 à 10, c'est à dire 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10, je vais appeler cette méthode en lui passant en paramètres 1 et 10, c'est-à-dire les bornes des entiers dont il faut faire la somme.  
  
Quelque chose du genre :  
  
**Code : C# -**[**Sélectionner**](http://www.siteduzero.com/tutoriel-3-545046-tp-calculs-en-boucle.html)

|  |  |
| --- | --- |
| 1  2 | Console.WriteLine(CalculSommeEntiers(1, 10));  Console.WriteLine(CalculSommeEntiers(1, 100)); |

Sachant que le premier résultat de cet exemple vaut 55 (1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55) et le deuxième 5050.  
  
La deuxième méthode acceptera une liste de double en paramètres et devra renvoyer la moyenne des doubles de la liste. Par exemple :  
  
**Code : C# -**[**Sélectionner**](http://www.siteduzero.com/tutoriel-3-545046-tp-calculs-en-boucle.html)

|  |  |
| --- | --- |
| 1  2 | List<double> liste = **new** List<double> { 1.0, 5.5, 9.9, 2.8, 9.6};  Console.WriteLine(CalculMoyenne(liste)); |

Le résultat de cet exemple vaut 5.76.  
  
Enfin, la dernière méthode devra dans un premier temps construire une liste d’entiers de 1 à 100 qui sont des multiples de 3 (3, 6, 9, 12, …). Dans un second temps, construire une autre liste d’entiers de 1 à 100 qui sont des multiples de 5 (5, 10, 15, 20, …). Et dans un dernier temps, il faudra calculer la somme des entiers qui sont communs aux deux listes … vous devez bien sur trouver 315 comme résultat :)   
  
Voilà, c’est à vous de jouer …  
  
Bon, allez, je vais quand même vous donner quelques conseils pour démarrer. Vous n’êtes pas obligé de les lire si vous vous sentez capables de réaliser cet exercice tout seul.  
  
Vous l’aurez évidemment compris, il va falloir utiliser des boucles.  
Je ne donnerai pas de conseils pour la première méthode, c’est juste pour vous échauffer :) .  
  
Pour la deuxième méthode, vous allez avoir besoin de diviser la somme de tous les doubles par la taille de la liste. Vous ne savez sans doute pas comment obtenir cette taille. Le principe est le même que pour la taille d’un tableau et vous l’aurez sans doute trouvé si vous fouillez un peu dans les méthodes de la liste. Toujours est-il que pour obtenir la taille d’une liste, on va utiliser liste.Count, avec par exemple :  
  
**Code : C# -**[**Sélectionner**](http://www.siteduzero.com/tutoriel-3-545046-tp-calculs-en-boucle.html)

|  |  |
| --- | --- |
| 1 | int taille = liste.Count; |

Enfin, pour la dernière méthode, vous allez devoir trouver tous les multiples de 3 et de 5. Le plus simple, à mon avis, pour calculer tous les multiples de 3, est de faire une boucle qui démarre à 3 et d’avancer de 3 en 3 jusqu’à la valeur souhaitée. Et pareil pour les multiples de 5.   
  
Ensuite, il sera nécessaire de faire deux boucles imbriquées afin de déterminer les intersections. C’est-à-dire parcourir la liste de multiple de 3 et à l’intérieur de cette boucle, parcourir la liste des multiples de 5. On compare les deux éléments, s’ils sont égaux, c’est qu’ils sont communs aux deux listes.  
  
Voilà, vous devriez avoir tous les éléments en main pour réussir ce TP, c'est à vous ^^

**Correction**

J’imagine que vous avez réussi ce TP, non pas sans difficultés, mais à force de tâtonnements, vous avez sans doute réussi. Bravo.  
Si vous n’avez même pas essayé, pas bravo :pirate: .  
Quoi qu’il en soit, voici la correction que je propose.

* Première méthode :

**Code : C# -**[**Sélectionner**](http://www.siteduzero.com/tutoriel-3-545046-tp-calculs-en-boucle.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | **static** int CalculSommeEntiers(int borneMin, int borneMax)  {  int resulat = 0;  **for** (int i = borneMin; i <= borneMax ; i++)  {  resulat += i;  }  **return** resulat;  } |

Ce n’était pas très compliqué, il faut dans un premier temps construire une méthode qui renvoie un entier et qui accepte deux entiers en paramètres.  
Ensuite, on boucle grâce à l’instruction **for** de la borne inférieure à la borne supérieure (incluse, donc il faut utiliser l’opérateur de comparaison <= ) en incrémentant un compteur de 1 à chaque itération.  
Nous ajoutons la valeur du compteur à un résultat, que nous retournons en fin de boucle.

* Seconde méthode :

**Code : C# -**[**Sélectionner**](http://www.siteduzero.com/tutoriel-3-545046-tp-calculs-en-boucle.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | **static** double CalculMoyenne(List<double> liste)  {  double somme = 0;  **foreach** (double valeur **in** liste)  {  somme += valeur;  }  **return** somme / liste.Count;  } |

Ici, le principe est grosso modo le même, la différence est que la méthode retourne un double et accepte une liste de double en paramètres. Ici, nous utilisons la boucle**foreach** pour parcourir tous les éléments que nous ajoutons à un résultat. Enfin, nous retournons ce résultat divisé par la taille de la liste.

* Troisième méthode :

**Code : C# -**[**Sélectionner**](http://www.siteduzero.com/tutoriel-3-545046-tp-calculs-en-boucle.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | **static** int CalculSommeIntersection()  {  List<int> multiplesDe3 = **new** List<int>();  List<int> multiplesDe5 = **new** List<int>();  **for** (int i = 3; i <= 100; i += 3)  {  multiplesDe3.Add(i);  }  **for** (int i = 5; i <= 100; i += 5)  {  multiplesDe5.Add(i);  }  int somme = 0;  **foreach** (int m3 **in** multiplesDe3)  {  **foreach** (int m5 **in** multiplesDe5)  {  **if** (m3 == m5)  somme += m3;  }  }  **return** somme;  } |

Peut-être la plus compliquée... :)   
  
On commence par créer nos deux listes de multiples. Comme je vous avais conseillé, j’utilise une boucle **for** qui commence à trois avec un incrément de 3. Comme ça, je suis sûr d’avoir tous les multiples de 3 dans ma liste. C’est le même principe pour les multiples de 5, sachant que dans les deux cas, la condition de sortie est quand l’indice est supérieur à 100.  
  
Ensuite, j’ai mes deux boucles imbriquées où je compare les deux valeurs et si elles sont égales, je rajoute la valeur à la somme globale que je renvoie en fin de méthode.  
Pour bien comprendre ce qu’il se passe dans les boucles imbriquées, il faut comprendre que nous allons parcourir une unique fois la liste multiplesDe3 mais que nous allons parcourir autant de fois la liste multipleDe5 qu’il y a d’éléments dans la liste multipleDe3, c’est-à-dire 33 fois.  
Ce n’est sans doute pas facile de le concevoir dès le début, mais pour vous aider, vous pouvez essayer de vous faire l’algorithme dans la tête :

* On rentre dans la boucle qui parcoure la liste multiplesDe3
* m3 vaut 3
* On rentre dans la boucle qui parcoure la liste multiplesDe5
* m5 vaut 5
* On compare 3 à 5, ils sont différents
* On passe à l’itération suivante de la liste multiplesDe5
* m5 vaut 10
* On compare 3 à 10, ils sont différents
* Etc … jusqu’à ce qu’on ait fini de parcourir la liste des multiplesDe5
* On passe à l’itération suivante de la liste multiplesDe3
* m3 vaut 6
* On rentre dans la boucle qui parcoure la liste multiplesDe5
* m5 vaut 5
* On compare 6 à 5, ils sont différents
* On passe à l’itération suivante de la liste multiplesDe5
* Etc …

**Aller plus loin**

Vous avez remarqué que dans la deuxième méthode, j’utilise une boucle **foreach** pour parcourir la liste :  
  
**Code : C# -**[**Sélectionner**](http://www.siteduzero.com/tutoriel-3-545046-tp-calculs-en-boucle.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | **static** double CalculMoyenne(List<double> liste)  {  double somme = 0;  **foreach** (double valeur **in** liste)  {  somme += valeur;  }  **return** somme / liste.Count;  } |

Il est aussi possible de parcourir les listes avec un **for** et d’accéder aux éléments de la liste avec un indice, comme on le fait pour un tableau.  
Ce qui donnerait :  
  
**Code : C# -**[**Sélectionner**](http://www.siteduzero.com/tutoriel-3-545046-tp-calculs-en-boucle.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | **static** double CalculMoyenne(List<double> liste)  {  double somme = 0;  **for** (int i = 0; i < liste.Count; i++)  {  somme += liste[i];  }  **return** somme / liste.Count;  } |

Notez qu’on se sert de liste.Count pour obtenir la taille de la liste et qu’on accède à l’élément courant avec liste[i].  
Je reconnais que la boucle **foreach** est plus explicite, mais cela peut être utile de connaitre le parcours d’une liste avec la boucle **for**. A vous de voir.  
  
Vous aurez également noté ma technique pour avoir des multiples de 3 et de 5. Il y en a plein d’autres. Je pense à une en particulier et qui fait appel à des notions que nous avons vu auparavant : la division entière et la division de double.  
  
Pour savoir si un nombre est un multiple d’un autre, il suffit de les diviser entre eux et de voir s’il y a un reste à la division. Pour ce faire, on peut le faire de deux façons. Soit en comparant la division entière avec la division « double » et en vérifiant que le résultat est le même. Si le résultat est le même, c’est qu’il n’y a pas de chiffres après la virgule :  
  
**Code : C# -**[**Sélectionner**](http://www.siteduzero.com/tutoriel-3-545046-tp-calculs-en-boucle.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | **for** (int i = 1; i <= 100; i++)  {  **if** (i / 3 == i / 3.0)  multiplesDe3.Add(i);  **if** (i / 5 == i / 5.0)  multiplesDe5.Add(i);  } |

Cette technique fait appel à des notions que nous n’avons pas encore vues : comment cela se fait-il qu’on puisse comparer un entier à un double ? Nous le verrons un peu plus tard.

L'autre solution est d'utiliser l’opérateur modulo que nous avons vu précédemment qui fait justement ça :  
  
**Code : C# -**[**Sélectionner**](http://www.siteduzero.com/tutoriel-3-545046-tp-calculs-en-boucle.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | **for** (int i = 1; i <= 100; i++)  {  **if** (i % 3 == 0)  multiplesDe3.Add(i);  **if** (i % 5 == 0)  multiplesDe5.Add(i);  } |

L’avantage de ces deux techniques est qu’on peut construire les deux listes de multiples en une seule boucle.  
  
Voilà, c’est fini pour ce TP. N’hésitez pas à vous faire la main sur ces boucles car il est fondamental que vous les maitrisiez. Faites-vous plaisir, tentez de les repousser dans leurs limites, essayez de résoudre d’autres problèmes, ... L’important, c’est de pratiquer.