Exercices chapitre 1 - Séquences et arithmétique

[Exercice 0 - Premiers pas dans Visual Studio 2](#__RefHeading___Toc1438_2839744299)

[Exercice 1.0 - L'affectation 6](#__RefHeading___Toc1440_2839744299)

[Exercice 1.1 – Un calcul simple 8](#__RefHeading___Toc1442_2839744299)

[Exercice 1.2 - Calculer une moyenne 8](#__RefHeading___Toc1446_2839744299)

[Exercice 1.3 – Échange de variable 9](#__RefHeading___Toc1448_2839744299)

[Exercice 1.4 - Georges sur la route 9](#__RefHeading___Toc1450_2839744299)

[Exercice 1.5 - Le salaire net d’un employé 10](#__RefHeading___Toc1452_2839744299)

[Exercice 1.6 - Augmentation chez Vandelay Industries 10](#__RefHeading___Toc1454_2839744299)

[Exercice 1.7 - Le tank 11](#__RefHeading___Toc1456_2839744299)

# 

# 

## 

## Exercice 0 - Premiers pas dans Visual Studio

Le but de cet exercice est de vous familiariser avec l'interface de Visual Studio et de « jouer » un peu avec l'application, même si vous ne comprenez pas encore nécessairement le C#. À ce stade-ci, ce n'est pas trop grave si vous n'êtes pas certain de savoir exactement ce que signifie chacune des lignes de code -- ça viendra bien rapidement par la suite.

Lorsqu'on vous demande de noter une réponse, vous devez le faire sur papier ou dans un document électronique quelconque (Word ou un bloc-notes, par exemple).

1. Démarrez Visual Studio et créez un nouveau projet. Attention de bien choisir le bon type de projet. Notez bien le dossier où vous choisissez de l'enregistrer, peu importe où c'est.

* Quel type de projet avez-vous choisi?

2. Une fois le nouveau projet créé, remarquez la fenêtre "Explorateur de solutions" à gauche.

* Que se passe-t-il si vous double-cliquez sur l'élément "Program.cs" de cette fenêtre?
* À quoi cet élément correspond-il, selon vous?

3. Effacez tout le code qui est déjà présent dans le projet et remplacez-le par celui-ci, que vous pouvez simplement copier/coller :

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace test\_ex0

{

class Program

{

static void Main(string[] args)

{

}

double rayon; //Le rayon de la sphère

double volume; //Le volume de la sphère

const double Pi = 3.1415;

Console.WriteLine("Entrez le rayon de la sphère: ");

rayon = double.Parse(Console.ReadLine());

volume = 4.0 / 3.0 \* Pi\* rayon \* rayon\* rayon

Console.Write("Le volume est : ");

Console.WriteLine(volume);

}

}

Vous noterez rapidement que la majorité du code est souligné en rouge, signifiant que quelque chose ne va pas.

* Placez votre pointeur de souris sur un des mots soulignés en rouge (n'importe lequel). Quel est le message d'erreur qui apparaît alors?
* Pouvez-vous corriger ce qui ne va pas? Il n'est pas nécessaire de faire quelque chose pour chaque mot souligné, tout peut se régler d'un seul coup si on observe bien. Qu'avez-vous eu à faire au juste?

4. Une fois le code réorganisé correctement, vous noterez qu'il reste encore une erreur. Vous pourriez placer votre pointeur de souris sur la ligne rouge qui reste pour obtenir plus de détails sur l'erreur, mais prétendons pour l'instant que vous ne l'avez pas remarquée.

* Générez la solution. Comment avez-vous fait?
* La fenêtre "Sortie" devrait maintenant contenir la mention ========== Génération : 0 a réussi, 1 a échoué, 0 mis à jour, 0 a été ignoré ==========, indiquant que la génération n'a pas fonctionné.
* Afin de savoir ce qui s'est passé, ouvrez le menu Affichage et sélectionnez "Liste d'erreurs"
* Quelle est l'erreur qui apparaît dans la (courte) liste?
* Doublez-cliquez sur l'erreur. Où votre curseur est-il envoyé?
* Corrigez l'erreur.

5. Maintenant que l'erreur est corrigée, vous noterez que la ligne commençant par "Console.Write" n'est pas alignée avec les autres.

* Appuyez sur CTRL-A, K, F (maintenez CTRL enfoncé et appuyez sur A, puis sur K, puis sur F). Que se passe-t-il?

6. Générez de nouveau la solution.

* Que se passe-t-il maintenant?

7. Portez maintenant votre attention sur les lignes qui commencent par "using" au début du code.

* Effacez-les toutes.
* Générez de nouveau la solution.
* Regardez la liste d'erreurs comme vous avez fait au point 4. Qu'y est-il indiqué?

8. Cliquez n'importe où dans la fenêtre de code et faites CTRL-Z.

* Que se passe-t-il?
* Faites encore CTRL-Z, quatre fois. Que constatez-vous?
* Faites CTRL-Y quatre fois. Que se passe-t-il?

9. Si tout va bien, vous devriez avoir récupéré votre code fonctionnel. Observez de nouveau les lignes qui commencent par "using".

* Y a-t-il une différence entre la première (using System;) et les autres? Laquelle?
* Effacez toutes les lignes qui commencent par "using" sauf la première.
* Générez de nouveau la solution. Que se passe-t-il maintenant?

10. Démarrez votre programme en allant dans le menu Déboguer, puis en choisissant "Démarrer le débogage". Entrez 2 comme valeur de Rayon.

* Que constatez-vous?
* Démarrez de nouveau votre programme, mais en choisissant cette fois "Exécuter sans débogage". Quelle est la différence?

11. Enregistrez votre projet et fermez Visual Studio. Ouvrez ensuite une fenêtre d'explorateur Windows et retrouvez le dossier où vous avez sauvegardé votre projet au point 1.

* Cliquez sur l'onglet Affichage et cochez "Extensions de noms de fichiers"
* Entrez dans le dossier qui porte le nom de votre projet.
* Entrez maintenant dans le sous-dossier qui porte le même nom
* Double-cliquez sur Program.cs. Que voyez-vous?
* Essayez de générer votre programme. Que constatez-vous?

12. Fermez de nouveau Visual Studio et retrouvez votre fenêtre d'explorateur Windows.

* Revenez en arrière pour retrouver le premier dossier qui porte le nom de votre projet.
* Doublez-cliquez sur le fichier portant l'extension .sln
* Essayez de générer votre programme. Que constatez-vous?
* Si vous vouliez transférer votre projet sur une clé USB que croyez-vous que vous devriez faire?

## 

## Exercice 1.0 - L'affectation

Directives

1. Déterminez la valeur de chacune des variables après l'exécution de chacun des programmes. N'utilisez pas l'ordinateur! Utilisez seulement un papier et un crayon.
2. Ensuite, transcrivez chacun des programmes dans Visual Studio, puis vérifiez la valeur de chacune des variables après l'exécution de chaque programme.

Est-ce la théorie correspond à la pratique?

Montrez vos résultats à votre gentil enseignant.

|  |  |  |
| --- | --- | --- |
|  | Théorie (sans Visual Studio) | Pratique (avec Visual Studio |
| Programme 1  int gros = 10;  int petit = 20;  gros = petit; | gros vaut 20  petit vaut 20 | gros vaut ? 20  petit vaut ? 20 |
| Programme 2  int a = 10;  int b = 20;  a = b;  b = a; | a vaut ?  b vaut ? | a vaut ? 20  b vaut ? 20 |
| Programme 3  int a = 10;  int b = 20;  b = a;  a = b; | a vaut ?  b vaut ? | a vaut ?  b vaut ? |
| Programme 4  int a = 10;  int b = 20;  int c = 30;  a = b;  b = c; | a vaut ?  b vaut ?  c vaut ? | a vaut ?  b vaut ?  c vaut ? |
| Programme 5  int a = 5;  int b = 3;  int c = 7;  a = c;  b = a;  c = b; | a vaut ?  b vaut ?  c vaut ? | a vaut ?  b vaut ?  c vaut ? |

## 

## 

## Exercice 1.1 – Un calcul simple

Créer un programme qui demande 2 valeurs au clavier. Effectuez ensuite le calcul suivant: la première valeur \* 10 + la deuxième valeur / 10 et faites afficher le résultat.

Utilisez le jeu de tests suivant pour tester votre programme:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Intrants | | Extrant |
|  | valeur1 | valeur2 | resultat |
| Test 1 | 20 | 35 | 203,5 |
| Votre test |  |  |  |

## Exercice 1.2 - Calculer une moyenne

Écrivez un programme qui permet de calculer l’âge moyen de deux personnes. Votre programme doit demander l’âge des deux personnes, calculer la moyenne et afficher le résultat. Utilisez le test suivant pour tester votre programme et **inventez un deuxième test:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Intrants | | Extrant |
|  | Âge 1 | Âge 2 | Âge moyen |
| Test 1 | 20 | 29 | 24,5 |
| Votre test |  |  |  |

## Exercice 1.3 – Échange de variable

Ceci est un problème classique en algorithmique: créez un programme qui permet d'échanger le contenu de deux variables, disons x et y. On supposera que ces variables contiendront uniquement des nombres entiers. Votre programme devra demander à l'usager d'entrer les nombres, les stocker dans les variables x et y, échanger le contenu des variables, puis afficher le contenu final des variables à l'écran. Vous devez réellement échanger le contenu des variables, le fait d'écrire à l'écran "voici le contenu de x" puis d'afficher le contenu de y n'est pas considéré comme une bonne solution!

Utilisez le jeu de tests suivant pour tester votre programme:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Intrants | | Extrants | |
|  | x | y | x | y |
| Test 1 | 3 | 4 | 4 | 3 |
| Test 2 | -6 | 8 | 8 | -6 |

## Exercice 1.4 - Georges sur la route

Georges est un représentant sur la route. Comme il voyage beaucoup et qu'il change régulièrement de voiture, il aimerait avoir un programme qui lui permettrait de calculer le coût de déplacement pour un voyage. Il aimerait que le programme lui permette d'entrer le prix d'un litre d'essence, la distance à parcourir et la consommation du véhicule en litres/100 km, puis le programme doit calculer le coût du voyage et l’afficher..

Utilisez le test suivant pour tester votre programme et **inventez un deuxième test:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Intrants | | | Extrant |
|  | Prix de l’essence ($/L) | Distance à parcourir (km) | Consommation du véhicule (L/100km) | Coût du voyage ($) |
| Test 1 | 1,50 | 50 | 10 | 7,50 |
| Votre test |  |  |  |  |

## Exercice 1.5 - Le salaire net d’un employé

Écrire un programme permettant de calculer et d'afficher le salaire net d'un employé. Pour cela, il vous faudra lire les informations suivantes :

* le nombre d'heures travaillées ;
* le salaire horaire.

Il vous faudra tenir compte que l'employé devra payer :

* 15% d'impôt fédéral ;
* 20% d'impôt provincial.

Utilisez le test suivant pour tester votre programme et inventez un deuxième test:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Intrants | | Extrant |
|  | Nombre heures travaillées (h) | Salaire horaire ($/h) | Salaire net ($) |
| Test 1 | 35 | 15 | 341,25 |
| Votre test |  |  |  |

## Exercice 1.6 - Augmentation chez Vandelay Industries

Les employés de Vandelay Industries viennent enfin d'obtenir une augmentation de 3% suite aux négociations serrées de leur syndicat. La convention collective ayant échu il y a 9 mois, cette augmentation sera donc rétroactive à cette période. Créez-leur un programme qui permettra à un employé d'entrer son ancien salaire annuel pour obtenir:

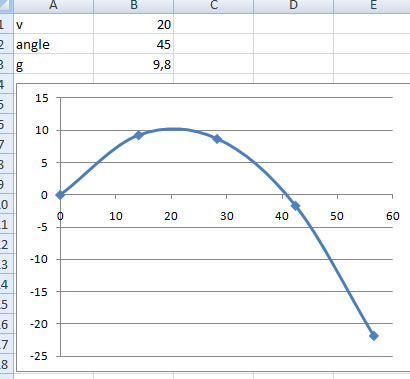
* Le nouveau salaire annuel
* Le nouveau salaire mensuel
* Le montant rétroactif auquel il aura droit sur sa première paye

Utilisez le test suivant pour tester votre programme et inventez un deuxième test:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Intrant | Extrants | | |
|  | Ancien salaire annuel ($) | Nouveau salaire annuel ($) | Nouveau salaire mensuel ($) | Montant rétroactif ($) |
| Test 1 | 30 000 | 30 900 | 2 575 | 675 |
| Votre test |  |  |  |  |

## Exercice 1.7 - Le tank

Un jeu vidéo simule un tank qui tire des projectiles. La distance maximale que peut atteindre le projectile est donnée par la formule **2\*v2\*cos(theta)\*sin(theta)/g**. Le temps que prendra le projectile à atteindre le sol est donnée par la formule **2\*v\*sin(theta)/g**. Écrivez un programme qui demande la vitesse initiale du projectile v (en m/s) et l'angle de tir theta (en degrés) et qui calcule la distance maximale de tir et le temps où le projectile touchera le sol. La valeur de g est de 9,8. N'oubliez pas que les fonctions trigonométriques de la bibliothèque *Math* utilisent des angles en radians!



Utilisez le test suivant pour tester votre programme et inventez un deuxième test:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Intrants | | Extrants | |
|  | Angle de tir (degré) | Vitesse initiale (m/s) | Distance (m) | Temps (s) |
| Test 1 | 45 | 20 | 40,82 | 2,89 |
| Votre test |  |  |  |  |