Introduction à la programmation – Bloc E

Structures de contrôle

Vous allez réutiliser les structures de contrôles pendant toute votre carrière de programmeur!

- Fonctions
- Boucles
- Conditions
- Switch
- Sortie de programme (`Environment.Exit(0);`)

Traitement de chaine

C# et la majorité des langages de programmation possèdent des fonctions de traitement de chaine.

Exemple :

```
string a = "Ma chaine";

a.Substring(0, 5);

Console.WriteLine(a);
```

Les fonctions retournent une chaine! Vous devez prendre la valeur de retour pour avoir la valeur modifié!

La variable « a » demeure inchangé!

Traitement de chaine

Pour que la valeur soit changé, il faut prendre le retour de la fonction et l'ajouter à la même variable ou à une autre variable.

Exemple:

```
string a = "Ma chaine";

a = a.Substring(0, 5);

Console.WriteLine(a);
```

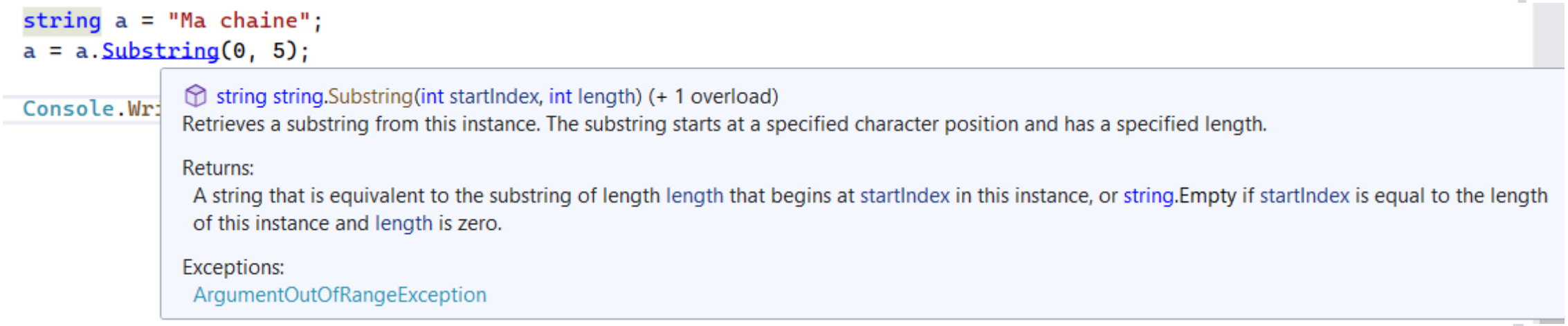
```
Ma ch
```

Pour avoir la variable modifiée, il faut donc prendre le retour de la fonction

Les fonctionnalités internes

Pour voir le détail des fonctionnalités d'une fonction du *framework*.

- Appuyer et tenez enfoncé la touche CTRL
- Cliquez sur le bouton de gauche de la souris



Les fonctionnalités internes

Le système vous informe sur les paramètres et le type de retour.

```
...public bool StartsWith(string value)...
...public bool StartsWith(string value, bool ignoreCase, CultureI
...public bool StartsWith(string value, StringComparison comparis
...public string Substring(int startIndex)...
...public string Substring(int startIndex, int length)...
IEnumerator<char> IEnumerable<char>.GetEnumerator()...
...IEnumerator IEnumerable.GetEnumerator()...
...bool IConvertible.ToBoolean(IFormatProvider provider)...
```

String interpolation 1

L'interpolation permet d'afficher les variables à l'aide d'accolades plutôt qu'à l'aide du signe +.

Exemple:

```
var name = "Cookie";

var name2 = "Boris";

var outputText = string.Format("Hello {0} {1}", name, name2);

Console.WriteLine(outputText);
```

```
Hello Cookie Boris
```

String interpolation 2

```
var name = "Cookie";

var name2 = "Boris";

var outputText = string.Format($"Hello {name} {name2}");

Console.WriteLine(outputText);
```

Même résultat mais en utilisation les variables plutôt que les paramètres de variables

```
Hello Cookie Boris
```

Amélioration

Comment pourrait-on améliorer le code suivant ?

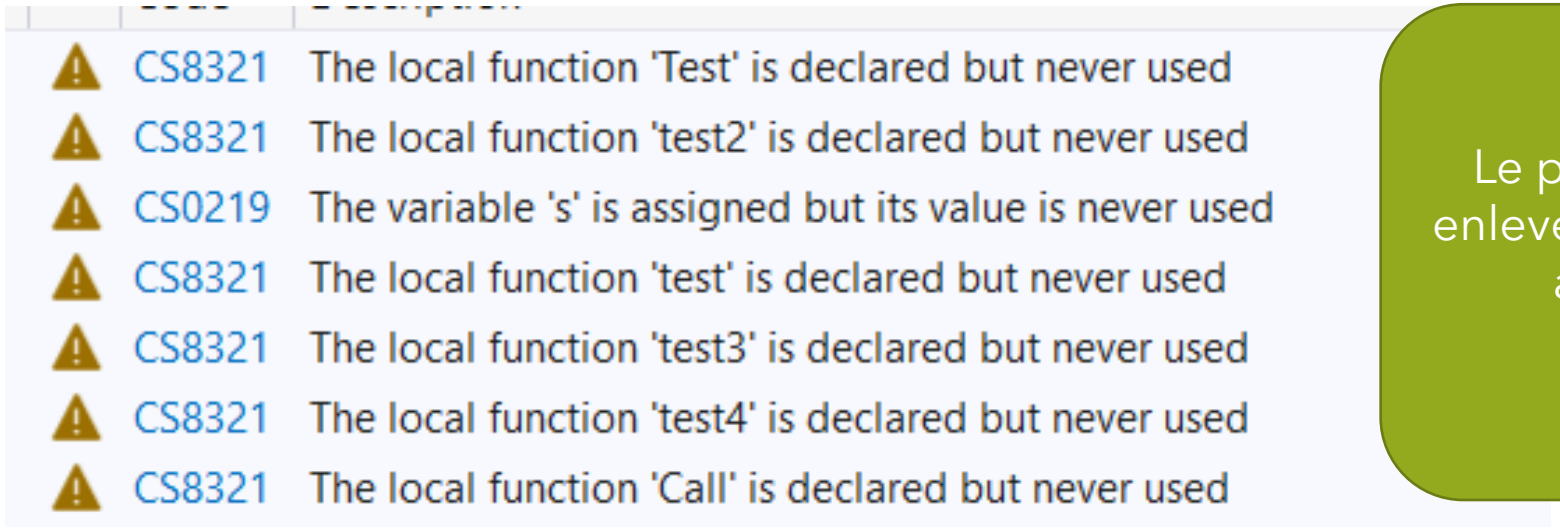
```
class Program
{
    static void Main(string[] args)
    {
        //We get some user input with the ReadLine() method
        string input = Console.ReadLine();

        //Let's see if this is empty or not
        if (input == "")
        {
            Console.WriteLine("The input is empty!");
        }
        else
        {
            //A string is a char array and has a Length property
            //It also can be accessed like a char array, giving
            //us single chars.
            for (int i = 0; i < input.Length; i++)
            {
                switch (input[i])
                {
                    case 'a':
                        Console.Write("An a - and not ... ");
                        //This is always required, we cannot just fall through.
                        goto case 'z';
                    case 'z':
                        Console.WriteLine("A z!");
                        break;
                    default:
                        Console.WriteLine("Whatever ...");
                        //This is required even in the last block
                        break;
                }
            }
        }
    }
}
```


Source: https://www.codeproject.com/Articles/1094079/An-advanced-introduction-to-Csharp-Lecture-Notes-P#basic-concepts

Warnings

Les warnings n'empêchent pas le fonctionnement mais informent le programmeurs qu'une action doit être prise.



Le programmeur doit enlever tous les *warnings* avant de livrer l'application.

Éliminer le deadcode

Une application ne doit pas posséder de « deadcode »

Si certaines lignes de code ne servent pas alors elles doivent disparaître

Limitez les variables globales

Les variables globales sont difficiles à suivre et le programmeur ne sait pas qu'elle(s) fonctions a modifiée une variable globale. Limitez leurs utilisation

Refactoring

Le *refactoring* est le processus dans lequel le programmeur améliore son code. Si le code est plus ou moins propre alors il faut refactoriser! Exemple:

- Renommer des classes
- Diminuer le code
- Changer le nom des variables
- Améliorer un traitement
- Sortir des « nested » traitement
- Etc.

Le programmeur doit effectuer du *refactoring* pendant le développement. Il permet d'améliorer le code et la qualité du produit!

Des questions ?

info@29a.ca