INTRODUCTION À LA PROGRAMMATION – BLOC C

Compilateurs

Le langage C# est un langage compilé. Donc, pour l'exécuter il doit être compilé!

Il est important de recompiler complètement le projet de temps à autre pour s'assurer que touts les fichiers sont à jour.

Convention de nommage

Retour sur les conventions de nom:

Une liste/tableau/array :	string[] listeNoms = new string[10];
Un total :	int totalAnimaux = 0;
Le poids total :	int totalPoidsAnimaux = 0;
Le nom du propriétaire :	string nomPropriétaire = « »;
Le nom de l'animal :	string nomAnimal = « »;

Soyez précis et significatifs dans le nommage des variables

Scope des variables

Les variables de fonctions sont existantes UNIQUEMENT dans les fonctions. En dehors des fonctions, elles n'existent plus. Pour avoir une variable qui est utilisée sur toute la « classe », elle doit être déclarée au début de la page (dépend des versions du Visual Studio).

```
class Program
{
    int total = 125;
    static void Main() {
       Program p = new Program();
       p.show();
    }
    public void show(){
        Console.WriteLine(total);
    }
}
```

Scope des variables

La variable « total » dans la fonction show_2 n'est pas accessible.

```
class Program
{
    static void Main() {
       Program p = new Program();
       p.show();
    }
    public void show(){
                    int total = 125;
        Console.WriteLine(total);
    }
    public void show_2(){
        Console.WriteLine(total);
    }
}
```

Le compilateur va générer une erreur

Switch/case

Les switch/case doivent contenir des fonctions et non des instructions de traitements.

```
switch (accountStatus)
    {
      case AccountStatus.NotRegistered:
                priceAfterDiscount = price;
                break;
      case AccountStatus.SimpleCustomer:
                priceAfterDiscount = (price - (0.1m * price));
                priceAfterDiscount = priceAfterDiscount -
                                (discountForLoyaltyInPercentage * priceAfterDiscount);
                 break;
      case AccountStatus.ValuableCustomer:
                priceAfterDiscount = (0.7m * price);
                priceAfterDiscount = priceAfterDiscount -
                                (discountForLoyaltyInPercentage * priceAfterDiscount);
        break;
      case AccountStatus.MostValuableCustomer:
                 priceAfterDiscount = (price - (0.5m * price));
                 priceAfterDiscount = priceAfterDiscount -
                          (discountForLoyaltyInPercentage * priceAfterDiscount);
        break;
    }
    return priceAfterDiscount;
```

À proscrire

Switch/case

Les switch/case doivent contenir des fonctions et non des instructions de traitements.

```
switch (accountStatus)
    {
      case AccountStatus.NotRegistered:
            AppelFonction_1();
            break;
      case AccountStatus.SimpleCustomer:
            AppelFonction_2();
            break;
      case AccountStatus.ValuableCustomer:
             AppelFonction_3();
             break;
      case AccountStatus.MostValuableCustomer:
             AppelFonction_4();
             break;
    }
```

Bonne approche

Les constantes

Les constantes sont des « variables » qui ne changent JAMAIS. On s'en sert pour stocker une valeur qui ne changera pas. Par exemple, un pourcentage de taxes serait un bon cas!

Comment on l'écrit :

```
const int X = 0;
public const double GRAVITATIONAL = 6.673e-11;
private const string PRODUCTNAME = "Visual C#";
```

Ces variables ne changeront jamais durant l'exécution de l'application

Les constantes sont en général en majuscule (convention)

Const et readonly

Quelle est la différence entre *const* et *readonly ?*

Const doit être initialisé au début alors que *readonly* doit être initialisé à la compilation.

Comparateur terniaire

Algorithme :

is this condition true ? yes : no

Revient à la même chose que :

```
If (x > 250){
    X = 50;
}
Else{
    x = 250;
}
```

```
class Program
{
    static void Main() {
        Random rnd = new Random();
        int x = rnd.Next(500);
        int z = x > 250 ? 50 : 250;

        Console.Write("Sum of x + y = "+ z);
    }
}
```

Excellent pour faire des conditions sur une seule ligne

Mot clé *continue*

Le mot clé continue permet de continuer d'effectuer un retour sur la boucle.

Dans notre exemple, le chiffre quatre ne sera jamais affiché à la console.

```
for (int i = 0; i < 10; i++)
{
  if (i == 4)
  {
    continue;
  }
  Console.WriteLine(i);
}
```

Le système retourne à l'entête de la boucle et continue le traitement. Attention, ceci est comme un *goto* alors il faut l'utiliser avec parcimonie

Casting

Le casting est une façon de transformer une variable en un autre type de variable.

- **Implicit Casting** (automatically) - converting a smaller type to a larger type size
  `char` -> `int` -> `long` -> `float` -> `double`

- **Explicit Casting** (manually) - converting a larger type to a smaller size type
  `double` -> `float` -> `long` -> `int` -> `char`

Casting implicite : On convertit naturellement du plus petit au plus grand. Exemple :

```
char a = 'b';
int b = a;
float c = b;
```

Casting explicite : On convertit naturellement du plus grand au plus petit. Exemple :

```
double myDouble = 9.78;
int myInt = (int) myDouble;
```

Null et « » (vide)

Null et « » ne sont pas identiques.

Lorsque vous devez effectuer des comparaisons, assurez-vous que l'application compare les bons éléments.

```
string[] liste = new string[10];

 if (liste[0] == null){

 }
 if (liste[0] == ""){

 }
```

Les comparaisons sont différentes?

Conditions if

```
if (time < 10)
{
  Console.WriteLine("Good morning.");
}
else if (time < 20)
{
  Console.WriteLine("Good day.");
}
else
{
  Console.WriteLine("Good evening.");
}
```

Votre préférence ?
Pourquoi ?

```
if (time < 10)
{
  Console.WriteLine("Good morning.");
}
 if (time < 20)
{
  Console.WriteLine("Good day.");
}
else
{
  Console.WriteLine("Good evening.");
}
```

Foreach

La boucle *foreach* l'écriture d'une boucle.

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
foreach (string i in cars)
{
  Console.WriteLine(i);
}
```

Comment obtenir l'index de i ?

Avec *IndexOf()*

Différence entre string et String

Quelle est la différence entre string et String ?

Aucune différence, c'est seulement l'alias qui change

QUESTIONS ?

Écrivez-moi à : info@29a.ca