Introduction à la programmation – Bloc B

Objectifs

Habiletés à développer:

- Les structures de contrôles
- Les conditions
- Les boucles
- Les switch
- Les fonctions et la pile d’exécution
- KISS et DRY

Les structures de contrôle

Les structures de contrôles permettent de contrôler le *flow* du programme. Sans ces structures il serait impossible de gérer les programmes informatiques.

Liste des structures:

- Conditions (if / else) ;
- Fonctions ;
- Boucles ;
- Switch ;

Les conditions en C#

```
if (x < y)
{
    Console.WriteLine("X est plus petit que Y");
}

if (w > x)
{
    Console.WriteLine("w est plus petit que x");
}
```

```
if (x < y && w > x)
{
    Console.WriteLine("x < y et w > x");
}

if (x > y || w > x)
{
    Console.WriteLine("x > y ou w > x");
}
```

Les boucles

Dans la majorité des langages de programmation il y a trois (3) types de boucles:

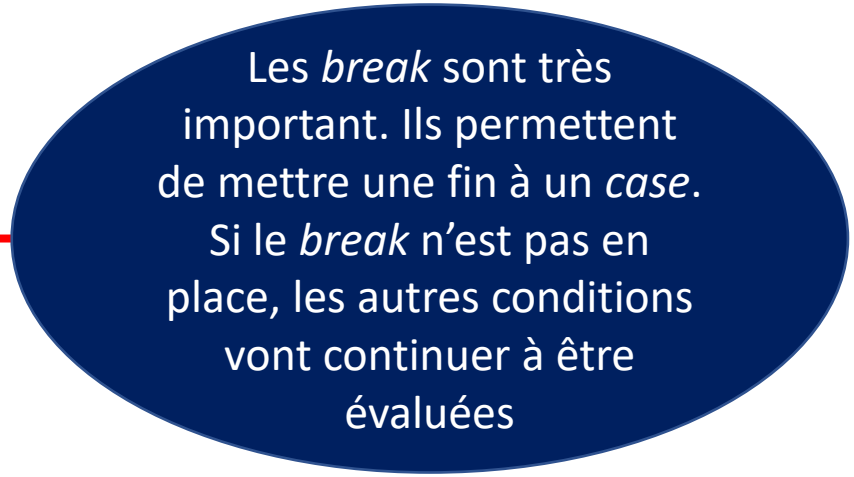
- For
- While
- Do While

```
for (int i=0; i<100; i++)
{
        Console.WriteLine(i);
}

int j = 0;
while (j < 100)
{
        Console.WriteLine(j);
        j = j + 1;
}

int k = 0;
do
{
        Console.WriteLine(k);
        k = k + 1;
} while (k < 100);
```

Les switch

Un switch est comme une série de *if* imbriqué. Il permet d'avoir un code beaucoup plus propre et robuste.

```
char monCar = 'd';
switch (monCar){
        case 'a': // Appel de traitement
        break;
        case 'b': // Appel de traitement
        break;
        case 'c': // Appel de traitement
        break;
        default:
        break;
}
```

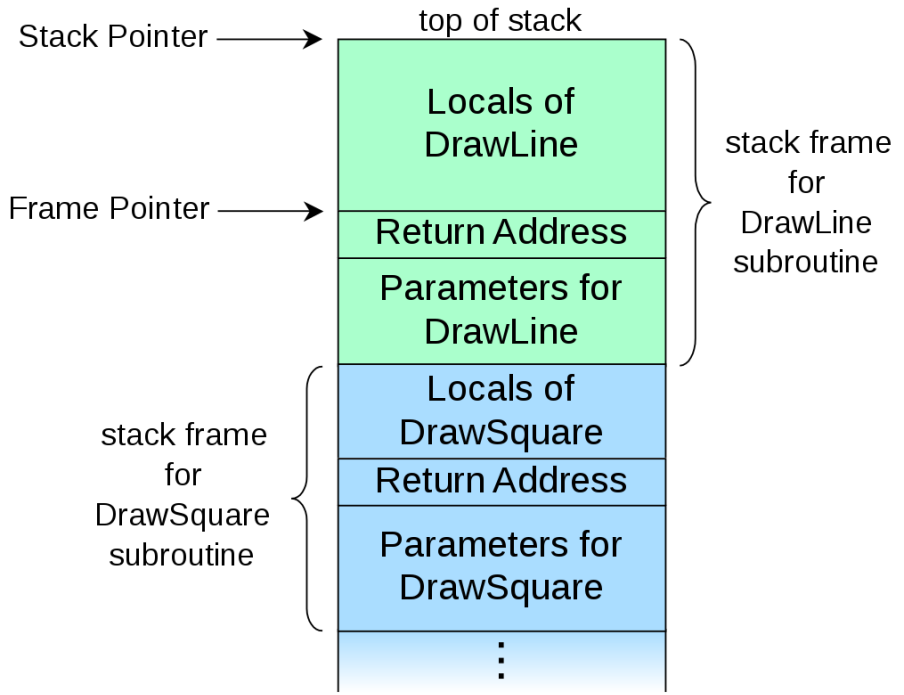


Les fonctions et la pile d'exécution

Wikipedia:

*« En informatique, la **pile d'exécution** (souvent abrégée en la pile ; en anglais, **call stack**) est une structure de données de type pile qui sert à enregistrer des informations au sujet des fonctions actives dans un programme informatique »*

L'utilisation de la pile prend son importance afin de conserver une trace de l'endroit où chaque fonction active doit retourner à la fin de l'exécution. Vous n'avez pas à gérer la pile car tout est géré derrière les rideaux. Le point important c'est de comprendre l'ordre des appels et des retours.



Source: https://fr.wikipedia.org/wiki/Pile_d%27ex%C3%A9cution

Les fonctions

Les fonctions permettent de découper les traitements en petits blocs de traitement.

Au lieu d'avoir un bloc incompréhensible on sépare le tout en petite fonction

```
protected void myFirstFunction()
{
        Console.WriteLine("Ma première fonction");
}
```

Les paramètres de fonctions

Parfois, les fonctions doivent recevoir des paramètres. Les paramètres reçues sont des variables de type primitive ou des objets.

```
// Exemple de variables primitives
protected void myFirstFunction(int x, int y)
{
        Console.WriteLine(x + y);
}

// Exemple d'objet passé en paramètre
protected void myFirstFunction(Animal chien)
{
        Console.WriteLine(chien.getName());
}
```

Les retours de fonctions

Les fonctions ont parfois besoin de retourner de l'information tel que le résultat d'une valeur.

```
// Exemple retour absent
protected void myFirstFunction(int x, int y)
{
        Console.WriteLine(x + y);
}

// Retour d'une chaine de caractère
protected string myFirstFunction()
{
        return "Cookie";
}
```

Void ne retourne rien. Elle est utilisée pour faire afficher des informations à la console

Il est possible de retourner une variable primitive ou un objet. Dans l'exemple, la fonction appelante va recevoir le mot « Cookie »

Le mot clé return

Le mot clé *return* permet de retourner une valeur à la fonction appelante. Par contre, il peut aussi être utilisé dans un autre contexte que le retour d'une variable.

Parfois il faut terminer une function plus rapidement. Pour ce faire il faut utiliser un des deux mots clés suivants:

1- return

2- break;

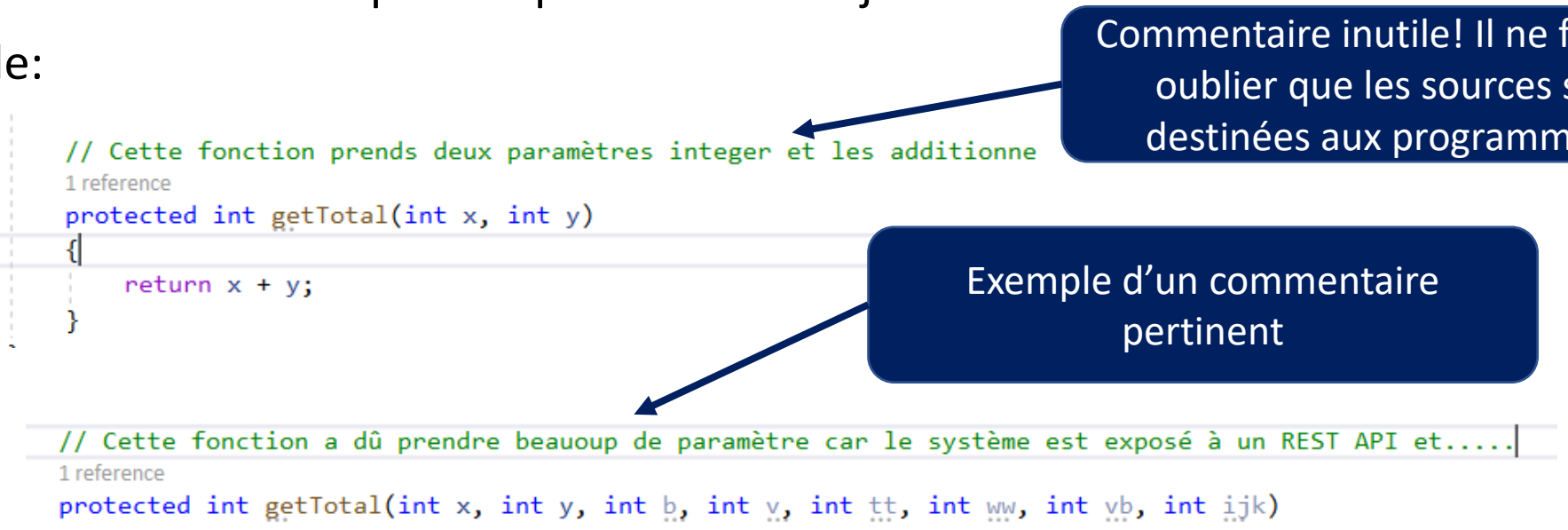
Attention, il ne faut pas les utiliser comme des go to! Les bonnes pratiquent stipulent qu'il ne devrait y avoir **un seul return par function et les break devraient uniquement être utilisés dans les switch.** Cependant, il y a certain type de logiciel dans lesquels il est necessaire de les utiliser plus régulièrement.

Les commentaires dans le code

Les commentaires dans le code sont parfois/souvent nécessaires! En effet, lorsque le code semble un peu ‘confus’ et que l’on sent le besoin de justifier notre code alors il faut ajouter des commentaires.

Il est important de noter qu’il ne faut pas ajouter de commentaires juste pour en ajouter! Choisissez bien la raison pour laquelle vous en ajoutez.

Exemple:



Les bonnes pratiques du KISS et DRY

Il y a beaucoup de normes de programmation! Certaines normes changent selon le langage de programmation mais il y une base commune qui ne change pas! Voici deux approches essentielles à appliquer en programmation:

1- DRY = Don’t Repeat Yourself

2- KISS = Keep It Simple Stupid

Si vous éprouvez des difficultés à comprendre votre code c’est qu’il est trop compliqué! **Un code complexe ne fait pas de vous un meilleur programmeur! Pensez simplicité!**

La méthode du DRY encourage fortement le fait de ne pas répéter du code pour rien! Découper votre programme en fonction afin de réutiliser les instructions communes.

La méthode du KISS est vraiment importante! Plus votre code source sera simple plus il sera facile de le modifier, de l’améliorer, de le comprendre, de corriger les défauts, etc.

Questions ?
info@29a.ca