**C# : Les bases**

Flux de control / Flux d’exécution : Ordre dans lequel sont exécutées les instructions du programme et dépendant de la réaction de celui-ci à l’entrée qui lui est donnée.

Instructions :

* De déclaration : Déclaration d’une nouvelle variable ou constante. Assignation d’une valeur à celles-ci ;
* D’expression : Expression qui calcule la valeur à stocker dans une variable ;
* De sélection : Permettent de rediriger le flux d’exécution vers de nouvelles parties du programme selon certaines conditions ;
* D’itération : Expression permettant d’exécuter une boucle d’instruction ;
* De saut : Permettent de rediriger le flux d’exécution vers une autre section du programme ;
* De gestion des exceptions : Gère les exceptions qui rend possible leur récupération au cours de l’exécution ;
* De checked et unchecked : Permet de vérifier les opérations autorisées à effectuer un dépassement de capacité ou non lors du stockage d’une valeur dans une variable trop petite ;

Type :

* **Lors de la déclaration de la variable :** Famille de la variable étudiée ;
* **Lors de la déclaration d’une fonction :** Famille des paramètres utilisé dans la fonction et famille de la variable retournée par la fonction.

CLR : Common Langage Runtime.

Class : type de référence. Lors de la création d’un objet, la variable à laquelle l’objet est affecté conserve uniquement une référence à son adresse mémoire. Lorsque l’adresse est affectée à une nouvelle variable, celle-ci fait toujours référence à l’objet d’origine. Les modifications apportées à une des deux variables sont répercutées sur la seconde puisqu’elles font référence au même objet.

Struct : type de valeur. Lors de la création d’un struct, les valeurs réelles de de celui-ci sont assignées à la variable. Si l’on affecte le struct à une nouvelle variable, il est copié. Les deux variables contiennent donc deux copies distinctes de l’objet et peuvent être modifiées indépendamment l’une de l’autre.

Délégué : Type de variable permettant d’instancier une méthode possédant une signature avec les mêmes types de variables de renvoie et arguments d’entrées que la méthode instanciée grâce à elle. Cela crée une sorte de Classe permettant de faire appel au type de fonction décrit précédemment. Cela représente un type d’encapsulation des méthodes. Cela permet également de passer cette méthode en argument d’une autre pour y faire appel simplement. Cette encapsulation fonctionne également pour des méthodes dites anonymes. Les méthodes anonymes ne permettent pas d’utiliser les méthodes de saut d’instructions telles que Goto, Break et Continue.