

SEQUENCE 3 – METHODES
SEANCE4-PASSAGE DES PARAMETRES PAR REFERENCE

OBJECTIFS

- Savoir faire un passage par référence (ref , out)

A retenir !

Les **paramètres effectifs, par défaut** :

- ne sont **pas modifiables** par les méthodes quand ce sont **des variables basiques** : de types primitifs ou structurés. Ex : int, double, structure TimeSpan, DateTime, Rectangle
- sont **modifiables** par les méthodes quand ce sont **des objets**. Ex : TextBox, ...

EXO 1 : MODIFIER UN PARAMETRE EFFECTIF DE TYPE PRIMITIF

1. D'après vous qu'affiche le code suivant ? A quoi servent les instructions ?

```
static void Main(string[] args)
{
    int a = 2, b = 3;
    Console.WriteLine("a = " + a + "      b = " + b);
    int tmp = b;
    b = a;
    a = tmp;
    Console.WriteLine("a = " + a + "      b = " + b);
}
```

2. On décide d'en faire une méthode : Au sein du répertoire « **Sequence_3_Methodes** », créez un projet « **Exo1_Echange** » type console dans une solution, nommée « **Seance4_PassageParReference** ».
3. créez un sous répertoire « TP3-MethodesPassageParReference » puis créez un projet « Exo1 » dans une solution « TP3 », copiez-collez le code ci-dessous. T

```
static void Main(string[] args)
{
    int a = 2, b = 3;
    Console.WriteLine("a = " + a + "      b = " + b);
    Program.Echange(a, b);
    Console.WriteLine("a = " + a + "      b = " + b);
}

public static void Echange ( int a , int b)
{
    int tmp = b;
    b = a;
    a = tmp;
}
```

4. Testez. Cela a-t-il fonctionné ? Mettez un point d'arrêt , en cliquant à gauche dans la colonne grise. Puis lancez l'exécution et puis faites F11, et observez les variables.



5. Modifiez le code en mettant le mot clef **ref** : devant les paramètres effectifs et formels.
Puis testez à nouveau.

```
Program.Echange(ref a, ref b);
public static void Echange (ref int a , ref int b)
```

EXO 2 : RETOUR OU PASSAGE PAR REFERENCE

Ajoutez un projet « **Exo2_Carre** » à votre solution : copiez-collez les instructions ci-dessous :
et définissez les 2 surcharges de la méthode **Carre** :

- la 1ere retourne la valeur spécifiée élevée au carré.
- La 2eme modifie directement la valeur spécifiée passée en référence

Program.cs

```
static void Main(string[] args)
{
    int val = 5;
    Console.WriteLine("Carré de " + val);
    val = Program.Carre(val);
    Console.WriteLine(" => " + val );
    val = 5;
    Program.Carre(ref val);
    Console.WriteLine(" => " + val);
}
```

EXO 3 : PASSAGE PAR REFERENCE POUR RECUPERER PLUSIEURS RESULTATS

Ajoutez un projet « **Exo3_ConversionHeureMin** » à votre solution : copiez-collez les instructions ci-dessous : et définissez les 2 méthodes sans changer le code donné.

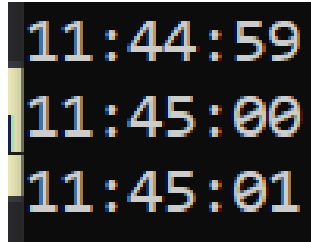
1. 1ere solution : **NbMinEnHeureMin** : la méthode doit convertir un temps spécifié en minutes en heures et minutes or elle ne peut renvoyer qu'un seul résultat : alors on passe par référence avec le mot clef **out** les paramètres devant être modifiés pour accueillir les résultats.
2. 2eme solution : **NbMinEnTimeSpan** : la méthode retourne une seule variable mais une variable complexe (un objet) contenant plusieurs données. Ici : une variable de type **TimeSpan**.

Program.cs

```
static void Main(string[] args)
{
    int h, m, nbMin = 98;
    Program.NbMinEnHeureMin(nbMin, out h, out m);
    Console.WriteLine(nbMin + " <=> " + h + ":" + m);
    TimeSpan temps = Program.NbMinEnTimeSpan(nbMin);
    Console.WriteLine(nbMin + " <=> " + temps);
}
```

Exo 4 : MODIFIER UN PARAMETRE EFFECTIF STRUCTURE

1. Ajoutez un projet « **Exo4** » à votre solution, testez ces lignes. Cela fonctionne-t-il ?
2. Refaites PlusUneSeconde : proposez 2 surcharges afin d'obtenir cet affichage :



```
11:44:59
11:45:00
11:45:01
```

```
static void Main(string[] args)
{
    TimeSpan temps = new TimeSpan(11, 44, 59);
    Console.WriteLine(temps);
    Program.PlusUneSeconde(temps);
    Console.WriteLine(temps);
}

private static void PlusUneSeconde(TimeSpan temps)
{
    temps = temps.Add(new TimeSpan(0, 0, 1));
}
```