

Exercice 5b

MISE EN ŒUVRE DES METHODES AVEC REF ET OUT

L'objectif de cet exercice est de découvrir l'utilisation des mots clés **out** et **ref** au niveau du passage des paramètres.

L'exercice se compose de trois actions séparées:

- La réalisation de la méthode **Extraire**, permettant d'obtenir les l'octet de poids fort et l'octet de poids faible d'une variable de type short.
- La réalisation de la méthode **Permute**, permettant de permuter les propriétés Text de 2 TextBox.
- La gestion de 5 boutons formant un affichage binaire sur 5 digits, avec l'affichage de la valeur convertie en décimal dans un TextBox.

EXEMPLE DE RESULTAT

VC# Exercice 5b : utilisation des méthodes avec ref & out

Test1

Extraire

Val short Hex: A1F5

Msb: A1

Lsb: F5

Test2

Permuter

Valeur A: BB

Valeur B: AAAAA

Test3

1 0 1 0 1

21

DETAIL DU TEST1

Le bouton doit être nommé btnExtraire et son libellé doit être "Extraire".

Le TextBox nommé txtValHex est accompagnée du label "Val short Hex" et le TextBox nommé txtMsb est accompagnée du label "Msb" et encore. le TextBox nommé txtLsb est accompagnée du label "Lsb"

Il faut créer une méthode **Extraire** ayant 2 paramètres :

- type **ushort**, la valeur obtenue du txtValHex
- type **byte**, avec le mot clé **out**, pour fournir le Msb

La méthode doit retourner le Lsb du type byte.

ACTION DANS LA METHODE EVENEMENTIELLE BTNEXTRAIRE_CLICK

Le contenu du TextBox txtValHex est un string représentant un nombre en hexadécimal, pour obtenir sa valeur numérique il faut utiliser :

```
Convert.ToInt16(txtValHex.Text, 16);
```

Ensuite il faut appeler la méthode Extraire, et afficher en hexadécimal le Msb et le Lsb obtenu.

DETAIL DU TEST2

Le bouton doit être nommé btnPermute et son libellé doit être "Permuter".

Le TextBox nommé txtValA est accompagnée du label "Valeur A" et le TextBox nommé txtValB est accompagnée du label "Valeur B".

Il faut créer une fonction **Permute** dont le prototype est le suivant :

```
private void Permute(ref TextBox ValA, ref TextBox ValB)
```

Cette méthode reçoit en référence des paramètres du type `TextBox` c, ce qui permet de fournir n'importe quel TextBox à la méthode.

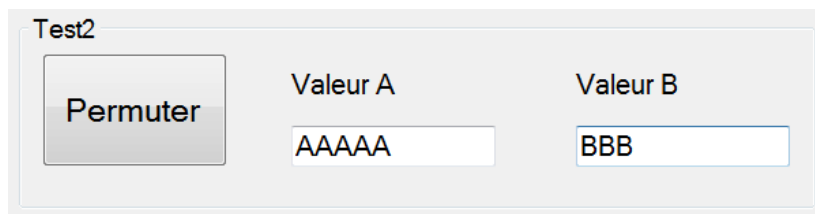
⊗ Le C# ne permet pas de mettre en référence seulement la propriété Text du TextBox !

Dans l'événement du bouton btnPermute, on appelle la méthode comme suit :

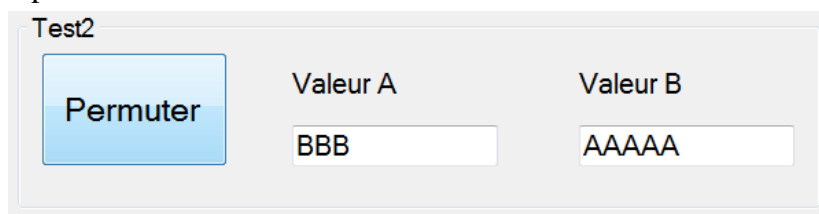
```
private void btnPermute_click(object sender, EventArgs e)
{
    Permute(ref txtValA, ref txtValB);
}
```

Dans la méthode Permute sans utiliser directement TxtValA et TxtValB, réalisez la permutation du contenu de ValA avec celui de ValB.

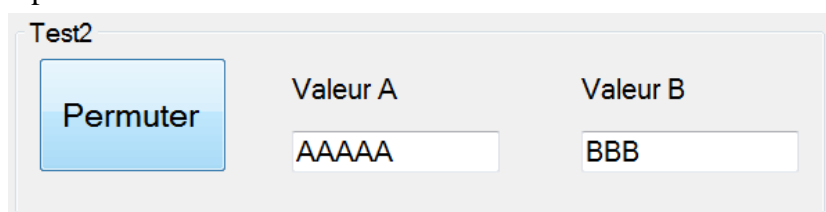
Situation au départ :



Après action "Permuter":



Après encore une action "Permuter":



DETAIL DU TEST3

On dispose de 5 boutons. Au départ les boutons affichent 0. Lorsque l'on clique sur le bouton il affiche 1. Si on clique à nouveau, il affiche 0.

Les 5 boutons forment une valeur binaire. L'action sur les boutons doit mettre à jour un tableau de 5 byte (membre appelé m_tabBin), chaque élément valant 1 ou 0.

Pour éviter d'écrire 5 fois le traitement il faut créer la méthode **GestionBit** qui reçoit en paramètre par référence un des éléments de TabBin pour inverser sa valeur. Le 2^{ème} paramètre par référence du type Button correspond au bouton dont il faut mettre à jour la propriété Text.

Il faut créer une méthode **DispDecimal** qui reçoit en paramètre par référence m_tabBin et affiche directement dans le TextBox nommé TxtDecimal la valeur décimale correspondante. ☞ La méthode **GestionBit** appelle la méthode **DispDecimal**.

Dans chaque méthode événementielle des boutons appelés bit0, bit1, bit2 et bit3 et bit4, il faut appeler **GestionBit** pour obtenir le résultat mis à jour.

Exemple d'utilisation de la méthode **GestionBit**, (cas du bouton bit0)

```
private void bit0_Click(object sender, EventArgs e)
{
    GestionBit(ref m_tabBin[0], ref bit0);
}
```