

TP tableaux et collections

1 Introduction

Le but de ce TP est de vous familiariser avec la gestion des tableaux (et revoir la gestion des boucles, et en particulier les boucle for et foreach).

Pour rappel, la boucle for permet de répéter un nombre de fois précis un ensemble d'instructions. La boucle foreach permet de parcourir un tableau.

Les tableaux permettent, au sein d'une seule variable, de stocker plusieurs informations.

Les documents ressources ci-dessous vous aideront pour prendre bien en main la syntaxe :

- « P2022_1OLEN_DOC_Ressource_Les_Iterations.pdf »
- « P2022_1OLEN_DOC_Ressource_Les_Collections.pdf »

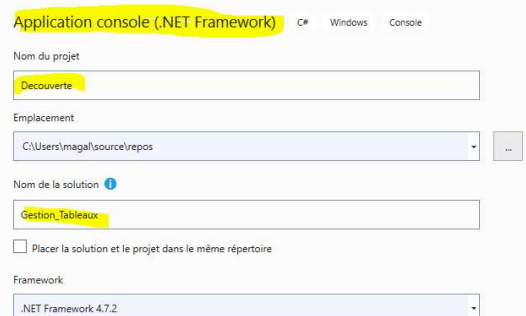
2 Application : Découverte

Le premier projet n'a pas de réel but fonctionnel, il est là pour vous aider dans la prise en main des tableaux.

- ✓ Créer une nouvelle solution de type application console (.Net Framework) nommé « Gestion_Tableaux » avec un projet nommé « Decouverte »
- ✓ Renommer le program.cs en decouverte.cs

Dans cette application, vous allez utiliser le hashage. Chaque composant à une « identité » propre et unique. Pour pouvoir l'afficher et la connaître, il faut utiliser la méthode GetHashCode() qui permet de renvoyer « l'identifiant » de votre composant. Grâce à ce code, vous pourrez plus facilement identifier vos tableaux et les distinguer.

Configurer votre nouveau projet



1. Création d'un tableau d'entiers vide

- ✓ Dans le Main(), déclarer un entier nommé taille et un tableau d'entiers nommé tab
- ✓ La variable taille est déterminée par une saisie au clavier
- ✓ Ajouter une fonction qui crée et retourne un tableau d'entiers vide. Elle répond au prototype suivant :

```
static int[] creerTableauVide(int ptaille)
```

La paramètre ptaille correspond à la variable taille du Main() initialisée par la saisie utilisateur

Votre fonction :

- Crée un tableau d'entiers, nommé tab1, dont la taille est celle passée en paramètre
- Affiche dans la console le hashCode du tableau `Console.WriteLine(tab1.GetHashCode());`
- Renvoie le tableau dans le tableau tab créé dans le Main()

- Dans votre Main()
 - Appeler votre fonction
 - Afficher le hashCode de tab
 - Tester

Voici ce que vous devez obtenir :

```
Question 2.1
quelle taille de tableau voulez-vous
5
Voici mon hashCode dans la fonction :46104728
Voici mon hashCode dans le main :46104728
```

2. Affichage du contenu d'un tableau

- ✓ Ajouter une procédure AfficherTableau qui répond au prototype suivant :
`static void AfficherTableau(int[] ptab)`
- ✓ Dans votre procédure, ajouter une boucle foreach qui permet de balayer le contenu de votre tableau et de l'afficher.
- ✓ Dans votre Main() :
 - Appeler votre procédure, en lui passant en paramètre tab
 - Tester

Voici ce que vous devez obtenir :

```
Question 2.2
Voici le contenu de mon tableau
0
0
0
0
0
```

3. Remplir votre tableau

- ✓ Ajouter une procédure qui permet de remplir un tableau d'entiers à partir de valeurs saisies au clavier par l'utilisateur. Elle répond au prototype suivant : `static void RemplirTableau(int[] ptab)`
- ✓ Dans votre Main() :
 - Appeler votre nouvelle procédure (doit se faire **IMPERATIVEMENT** après l'instanciation, c'est-à-dire l'appel de la fonction CréerTableauVide).
 - Appeler de nouveau votre procédure AfficherTableau pour voir le résultat

Voici ce que vous devez obtenir :

```

Question 2.1
quelle taille de tableau voulez-vous
5
Voici mon hashCode dans la fonction :46104728
Voici mon hashCode dans le main :46104728
Question 2.2
Voici le contenu de mon tableau
0
0
0
0
0
0
Question 2.3
saisir votre valeur
13
saisir votre valeur
7
saisir votre valeur
11
saisir votre valeur
8
saisir votre valeur
30
Voici, maintenant, le contenu de mon tableau
13
7
11
8
30

```

4. Remplir aléatoirement votre tableau

- ✓ Ajouter une procédure RemplirAleatoire qui permet de remplir, de manière aléatoire (Random) avec des entiers entre 1 et 100, le tableau passé en paramètre. Elle répond au prototype suivant :
`static void RemplirAleatoire(int[] ptab)`
- ✓ Dans votre Main() :
 - Déclarer un deuxième tableau d'entiers appelé tabDeux
 - Saisissez sa taille au clavier (même manipulation que pour tab 1)
 - Instancier votre tableau avec la procédure CréerTableauVide
 - Remplissez votre tableau de manière aléatoire en utilisant RemplirAleatoire
 - Afficher tabDeux en utilisant AfficherTableau
 - Tester
 - Trier votre tableau (il existe une méthode toute prête 😊 avec le type Array, il faut un peu chercher)
 - Afficher tabDeux en utilisant AfficherTableau

Voici ce que vous devez obtenir :

```
Question 2.4
quelle taille de tableau voulez-vous pour le second tableau ?
4
Voici mon hashCode dans la fonction :12289376
Voici le contenu de mon deuxième tableau
98
7
11
87
Voici le contenu de mon deuxième tableau trié
7
11
87
98
```

5. Copie de tableau (optionnel)

- ✓ Déclarer, créer et afficher un tableau d'entiers de taille 5 nommé tabTrois
- ✓ Effectuer l'affectation suivante : tabTrois = tab;
- ✓ Afficher le HashCode de tab et de tabTrois
- ✓ Afficher le contenu de tabTrois avec la méthode AfficherTableau

Que constatez-vous ?

- ✓ Modifier la valeur du premier et du dernier élément de tabTrois avec la valeur 999
- ✓ Afficher Tab et TabTrois

Que Constatez-vous ?

- ✓ Ajouter l'instruction suivante : tabDeux.CopyTo(tabTrois, 0);

Que constatez-vous ?

- ✓ Modifier la valeur du premier et du dernier élément de tabTrois avec la valeur 666
- ✓ Afficher tab, tabDeux et tabTrois

Que constatez-vous ?

3 Application Loto

Le but de cette application est la réalisation d'un tirage au sort de 5 numéros.

Le projet va se faire en 2 étapes :

- Tirage au sort des 5 numéros
- Tirage au sort des 5 numéros, mais des numéros forcément différents !!

1.1. Création projet

- ✓ Ajouter à votre solution, un projet Console .net framework nommé MonLoto
- ✓ Renommer le fichier Program.cs en Loto.cs

1.2. 1^{er} sous-programme

Le but de cette procédure est de tirer 5 numéros et de les afficher au fur et à mesure.

- ✓ Dans votre main :
 - Créer une variable de type random nommée hasard
 - Appeler votre procédure (que vous allez créer ci-dessous) dans votre main
- ✓ Créer une procédure LotoSimple qui répond au prototype suivant :


```
public static void LotoSimple(Random hasard)
```
- ✓ Coder votre procédure pour qu'elle :
 - Déclare une variable de type entier nommé numero
 - Affecte un nombre entre 1 et 49 à votre variable numero
 - Affiche la valeur de votre variable dans la console

Maintenant il va falloir répéter ces instructions 5 fois

- ✓ Dans votre procédure :
 - Ajouter une boucle for pour permettre le tirage et l'affichage de 5 numéros.
 - Ajouter un texte avant chaque numéro pour dire « Boule numéro x : »
 - Tester et vérifier que votre programme est cohérent par rapport au visuel ci-dessous

```
Boule numéro 1 : 3
Boule numéro 2 : 43
Boule numéro 3 : 24
Boule numéro 4 : 47
Boule numéro 5 : 35
```

Vous avez réussi à faire une première version d'un loto, cependant sachez que si votre programme reste tel que, il n'est pas juste. En effet vous pouvez tirer 2 fois le même numéro !!! Pour confirmer cette erreur, modifier votre programme pour que votre nombre aléatoire soit entre 1 et 5. Comme dans la capture ci-dessous, vous allez avoir des numéros qui apparaissent plusieurs fois :

```
Boule numéro 1 : 3
Boule numéro 2 : 2
Boule numéro 3 : 4
Boule numéro 4 : 1
Boule numéro 5 : 4
```

Comme vous le constatez, la boule 4 est sorti en 3^{ème} et 5^{ème} position. Or, au loto, un numéro ne peut sortir qu'une et une seule fois !!! Pour corriger cette erreur, vous allez créer le programme ci-dessous.

1.3. 2^{ème} sous-programme

Pour vérifier qu'un numéro ne puisse pas sortir deux fois, vous allez devoir utiliser les tableaux.

Le principe est le suivant :


- 1) Je tire un numéro
 - 2) Je compare le numéro aux numéros sortis précédemment (les numéros sortis précédemment sont stockés dans un tableau)
 - 3) Si le numéro est déjà sorti, j'en tire un nouveau
 - 4) Si le numéro n'est pas sorti, je le stocke dans mon tableau.
- ✓ Ajouter une procédure LotoComplet qui répond au prototype suivant :
`public static void LotoComplet(Random hasard)`
 - ✓ Reprendre le code de la procédure LotoSimple qui permet :
 - De tirer un numéro
 - De répéter le tirage 5 fois
 - D'afficher le résultat
 - ✓ Ajouter à votre code (pour cela nous vous conseillons fortement d'écrire l'algorithme avant) :
 - Un tableau de type Array nommé tabLoto de taille 5
 - Vérifier que le numéro sorti n'est pas présent dans la tableau tabLoto
 - S'il est présent, je relance
 - S'il n'est pas présent, je le stocke
 - Je m'arrête quand mon tableau est plein

1.4. LotoComplet

- ✓ Ajouter une fonction LotoDynamique, qui fait la même chose que LotoComplet mais qui renvoie un tableau dynamique. Cette fonction vous permet de gérer le type ArrayList
`public static ArrayList LotoDynamique(Random hasard)`
- Cette fonction est assez similaire à la procédure LotoComplet mais la taille du tableau contenant les chiffres du loto s'agrandit au fur et à mesure. De plus, l'affichage du résultat se fait en dehors du sous-programme et directement dans le Main()

6. Loto WinForm

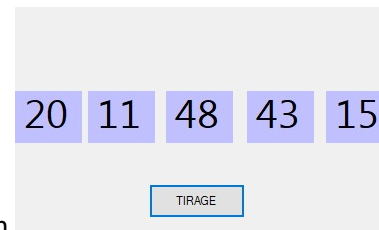
- ✓ Ajouter un nouveau projet à votre solution en Windows Form appelé « LotoForm » qui permet d'afficher les différentes boules dans un label (comme pour le 421).
 - Renommer le Form1.cs en FormLoto.cs
 - Renommer le program.cs en Loto.cs
 - Le formulaire contient 5 labels (boule1, boule2, boule3, boule4, boule5)
 - Un bouton Tirage
 - Ajouter une fonction Tirage qui reprend le code de la fonction LotoDynamique
 - Sur l'événement click de votre bouton, appeler votre fonction Tirage et affecter les valeurs de votre tableau aux différents labels.



0 0 0 0 0

TIRAGE

Après avoir cliqué sur le bouton



20 11 48 43 15

TIRAGE