C# : tableaux, collections, énumérations et boucles

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en Programmation par contrainte (IA)
Ingénieur en Génie logiciel

elmouelhi.achref@gmail.com

Plan

Introduction

Les boucles et les ensembles de valeurs

- Les boucles simplifie l'écriture d'un bloc de code qui se répète dans un programme
  - `for`, `foreach` et `.ForEach` que pour les ensembles
  - `while`
  - `do ... while`
- Une variable permet de stocker une seule valeur à la fois. Mais il existe d'autres structures en C# qui permettent de stocker plusieurs valeurs telles que
  - Les tableaux
  - Les collections
  - Les énumérations

Introduction

Dans ce cours

- La famille `for` sera traitée dans les différentes sections sur les ensembles
- `while` et `do ... while` dans la dernière section

Les tableaux

Les tableaux, c'est quoi ?

- une variable
- contenant un ensemble de valeurs
  - du même type
  - et dont le nombre (de valeurs) est fixé à la déclaration

Les tableaux

Déclaration

```
type[] nomTableau = new type[nbrElement];
```

Les tableaux

Déclaration

```
type[] nomTableau = new type[nbrElement];
```

Exemple

```
int[] tab = new int[3];
```

Les tableaux

Déclaration

```
type[] nomTableau = new type[nbrElement];
```

Exemple

```
int[] tab = new int[3];
```

Les tableaux

Déclaration

```
type[] nomTableau = new type[nbrElement];
```

Exemple

```
int[] tab = new int[3];
```

Utilisation

- Tous les éléments du tableau sont initialisés à 0.
- `tab[i]` : permet d'accéder à l'élément d'indice `i` du tableau
- Le premier élément d'un tableau est d'indice 0.
- On ne peut dépasser la taille initiale d'un tableau ni changer le type déclaré.

Les tableaux

Déclaration + initialisation

```
int[] tab = new int[] { 3, 5, 4 };
```

Les tableaux

Déclaration + initialisation

```
int[] tab = new int[] { 3, 5, 4 };
```

On peut aussi utiliser le raccourci suivant

```
int[] tab =  { 3, 5, 4 };
```

Les tableaux

Déclaration + initialisation

```
int[] tab = new int[] { 3, 5, 4 };
```

On peut aussi utiliser le raccourci suivant

```
int[] tab =  { 3, 5, 4 };
```

Cette écriture déclenche un `IndexOutOfRangeException`

```
tab[3] = 2;
```

Les tableaux

Parcourir un tableau avec un `for`

```
for (int i = 0; i < tab.Length; i++)
    Console.WriteLine(tab[i]);
```

Les tableaux

Parcourir un tableau avec un `for`

```
for (int i = 0; i < tab.Length; i++)
    Console.WriteLine(tab[i]);
```

Parcourir un tableau avec un `foreach`

```
foreach (int n in tab)
{
    Console.Write(n);
}
```

Les tableaux

Déclaration d'un tableau à deux dimensions

```
type[,] nomTableau = new type[nbLignes, nrColonnes];
```

Déclaration + initialisation

```
int[,] tab2dim = new int[,]
{
    {1, 2},
    {3, 4}
};
```

Ou

```
int[,] tab2dim =
{
    {1, 2},
    {3, 4}
};
```

Les tableaux

Parcourir un tableau à deux dimensions

```
foreach (int n in tab2dim)
{
    Console.Write(n);
}
```

Les tableaux

Parcourir un tableau à deux dimensions

```
foreach (int n in tab2dim)
{
    Console.Write(n);
}
```

Ou

```
for (int i = 0; i <2; i++)
    for (int j = 0; j < 2; j++)
        Console.WriteLine(tab2dim[i,j]);
```

Les tableaux

Parcourir un tableau à deux dimensions

```
foreach (int n in tab2dim)
{
    Console.Write(n);
}
```

Ou

```
for (int i = 0; i <2; i++)
    for (int j = 0; j < 2; j++)
        Console.WriteLine(tab2dim[i,j]);
```

Ne pas confondre `tab[,]` avec `tab[][]` qui veut dire un tableau de tableaux.

Les tableaux

Trier un tableau (unidimensionnel)

```
Array.sort(tab);
```

Autres opérations sur les tableaux

- `Array.Clear(tab, n, m)` : supprime les `m` valeurs (et non pas les éléments) du tableau en commençant par l'élément d'indice `n`. (il existe aussi `reverse` pour inverser l'ordre,...)
- `Array.IndexOf(tab, n))` : retourne l'indice de la première apparition de la valeur `n` dans le tableau `tab`. (il existe aussi `LastIndex`, `Exists`...)
- `Array.Resize(ref tab, n)` : réduit le nombre d'élément de `tab` au `n` premier élément
- ...

Les collections

Les collections, c'est quoi ?

- sont des objets
- permettent de regrouper et gérer plusieurs objets de taille et/ou type dynamiques
- (des tableaux multi-types extensibles)

Les collections

Quelques collections c#

- `ArrayList` : tableau dynamique
- `BitArray` : tableau statique de booléens
- `Queue` : file appliquant le principe FIFO (First In First Out)
- `Hashtable` : collection de couple clé/valeur
- `SortedList` : collection de couple clé/valeur ordonnée selon la clé
- `Stack` : collection appliquant le principe LIFO (Last In First Out)

Les collections

Pour utiliser une liste, il faut importer le namespace

```
using System.Collections;
```

Les collections

Pour utiliser une liste, il faut importer le namespace

```
using System.Collections;
```

Exemple avec `BitArray`

```
BitArray b = new BitArray(2);
b.Set(0, true);
b.Set(1, false);
b.Set(2, true);
```

Les collections

Pour utiliser une liste, il faut importer le namespace

```
using System.Collections;
```

Exemple avec `BitArray`

```
BitArray b = new BitArray(2);
b.Set(0, true);
b.Set(1, false);
b.Set(2, true);
```

La dernière instruction déclenche une exception car on ne peut dépasser la taille du tableau

Les collections

Exemple avec `Queue`

```
Queue q = new Queue();
q.Enqueue(2);
q.Enqueue(3);
q.Enqueue("bonjour");
q.Enqueue(5);
q.Dequeue();

foreach (var o in q)
    Console.WriteLine(o);
```

Les collections

Exemple avec `Queue`

```
Queue q = new Queue();
q.Enqueue(2);
q.Enqueue(3);
q.Enqueue("bonjour");
q.Enqueue(5);
q.Dequeue();

foreach (var o in q)
    Console.WriteLine(o);
```

Affiche 3 bonjour 5

Les collections

Exemple avec `ArrayList`

```
ArrayList arrayList = new ArrayList();
arrayList.Add(2);
arrayList.Add("Bonjour");
arrayList.Add('c');
arrayList.Add(8);
arrayList.Remove('c');
arrayList.RemoveAt(1);
foreach (var o in arrayList)
    Console.WriteLine(o);
```

Les collections

Exemple avec `ArrayList`

```
ArrayList arrayList = new ArrayList();
arrayList.Add(2);
arrayList.Add("Bonjour");
arrayList.Add('c');
arrayList.Add(8);
arrayList.Remove('c');
arrayList.RemoveAt(1);
foreach (var o in arrayList)
    Console.WriteLine(o);
```

Affiche 2 8

Les collections

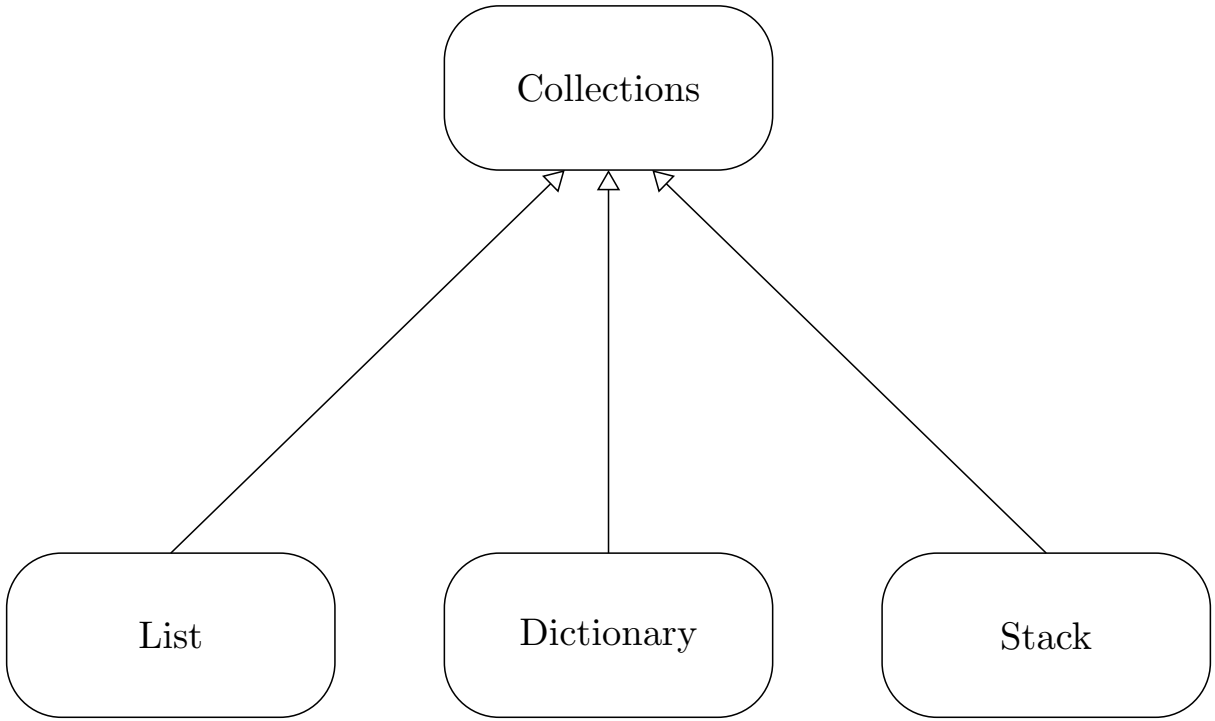
Les collections génériques, c'est quoi ?

- sont des collections
- avec précision de type, mais de taille dynamique
- (des tableaux extensibles)

Les collections

Les collections génériques, c'est quoi ?

- sont des collections
- avec précision de type, mais de taille dynamique
- (des tableaux extensibles)



Les collections

Pourquoi utiliser les collections génériques ?

Pour

- imposer un type pour tous les éléments de la collection
- éviter d'avoir des exceptions si le type attendu ne correspond pas au type d'un élément
- éviter de faire de conversions inutiles
- ...

Les listes

Pour utiliser une liste, il faut importer le namespace

```
using System.Collections.Generic;
```

Déclaration

```
List <type> nomListe = new List<type>();
```

Exemple

```
List <string> voitures = new List<string>();
```

Déclaration + initialisation

```
List <string> voitures = new List<string> {"Citroen"
   , "Ford", "Peugeot", "Mercedes"};
```

Les listes

Ajout d'un élément

```
voitures.Add("Renault");
```

Accès à un élément du tableau en lecture ou en écriture

```
voitures[3] = "Volkswagen"; // correct
Console.WriteLine(voitures[2]); // imprime peugeot
voitures[5] = "Fiat"; // déclenche une exception car
    il n'existe aucun élément d'indice 4
```

Les tableaux

Parcourir un tableau avec un `for`

```
for (int i = 0; i < voitures.Count-1; i++)
    Console.WriteLine(voitures[i]);
// Count retourne la taille exacte du tableau
// Atttention à l'utilisation de Capacity qui retourne le
    plus petit multiple de 4 supérieur au égal à la
   taille réelle de la liste
```

Parcourir un tableau avec un `foreach`

```
foreach (var voiture in voitures)
{
    Console.Write(voiture);
}
```

Parcourir un tableau avec un `.ForEach`

```
voitures.ForEach(elt => Console.Write(elt));
```

Les tableaux

Autres méthodes sur les listes

- `RemoveAt(n)` : supprime d'une liste l'élément d'indice `n` (il existe aussi `Remove` et `RemoveAll`)
- `IndexOf(n)` : retourne l'indice de la première apparition de la valeur `n` dans une liste
- `Contains(n)` : retourne `true` si `n` appartient à la liste
- `Find(elt => condition)` : retourne le premier élément de la liste qui respecte `condition` (`Exists` fonctionne d'une manière similaire mais elle retourne un booléen)
- `Sort()` : trie une liste d'entiers
- `ToArray()` : retourne un tableau statique résultat de la conversion de la liste
- ...

Les tableaux vs les listes

Quoi choisir ?

- Un tableau peut être multidimensionnel mais il est de taille fixe.
- Une liste est unidimensionnel mais elle est de taille variable.
- On ne peut supprimer un élément situé au début ou au milieu d'un tableau.

Les tableaux

Les tableaux

```
object[] obj = new object[2];
obj[0] = "chaine";
obj[1] = 2;
foreach (object elt in obj)
{
    Console.WriteLine(elt );
}
```

Les listes

```
List<object> list = new List<object>();
list.Add("chaine");
list.Add(2);
foreach (object elt in list)
{
    Console.WriteLine(elt );
}
```

Les dictionnaires

Les dictionnaires

- une collection de couple clé/valeur
- une clé est un indice personnalisé (unique)

Déclaration

```
Dictionary<type1, type2> dic = new Dictionary<type1
    , type2>();
```

Exemple

```
Dictionary<int, string> fcb = new Dictionary<int,
   string>();
```

Les dictionnaires

Remplir le dictionnaire

```
fcb.Add(10, "messi");
fcb.Add(23, "umtiti");
fcb.Add(4, "Rakitic");
fcb.Add(9, "Suarez");
```

Vérifier l'appartenance d'une valeur et/ou d'une clé à un dictionnaire

```
Console.WriteLine(fcb.ContainsValue("iniesta"));
Console.WriteLine(fcb.ContainsKey(10));
```

Les dictionnaires

Parcourir le dictionnaire et afficher le couple (clé,valeur)

```
foreach (KeyValuePair<int,string> elt in fcb)
    Console.WriteLine(elt.Key + elt.Value);
```

Les dictionnaires

Parcourir le dictionnaire et afficher le couple (clé,valeur)

```
foreach (KeyValuePair<int,string> elt in fcb)
    Console.WriteLine(elt.Key + elt.Value);
```

Parcourir le dictionnaire pour afficher les valeurs

```
foreach (string elt in fcb.Values)
    Console.WriteLine(elt);
```

Les dictionnaires

Parcourir le dictionnaire et afficher le couple (clé,valeur)

```
foreach (KeyValuePair<int,string> elt in fcb)
    Console.WriteLine(elt.Key + elt.Value);
```

Parcourir le dictionnaire pour afficher les valeurs

```
foreach (string elt in fcb.Values)
    Console.WriteLine(elt);
```

Pour récupérer la liste des clés, il faut déclarer un `KeyCollection`

```
Dictionary<int,string>.KeyCollection clefs = fcb.Keys;
foreach (int elt in clefs)
    Console.WriteLine(elt);
```

Les dictionnaires

Pour récupérer la liste des valeurs, il faut déclarer un `ValueCollection`

```
Dictionary<int,string>.ValueCollection vals = fcb.Values;
foreach (string elt in vals)
    Console.WriteLine(elt);
```

Les dictionnaires

Pour récupérer la liste des valeurs, il faut déclarer un `ValueCollection`

```
Dictionary<int,string>.ValueCollection vals = fcb.Values;
foreach (string elt in vals)
    Console.WriteLine(elt);
```

Pour modifier la valeur d'un élément selon la clé

```
fcb[10] = "Rivaldo";
foreach (int elt in clefs)
    Console.WriteLine(elt);
```

Les piles

Les piles

- une collection appliquant l'algorithme (LIFO, Last In First Out)
- impossible donc de supprimer/modifier/lire un élément au milieu de la liste

Déclaration

```
Stack<type> nomPile = new Stack<type>();
```

Exemple

```
Stack<int> pile = new Stack<int>();
```

Les piles

Empiler : ajouter un élément au sommet de la pile

```
pile.Push(2);
pile.Push(1);
pile.Push(5);
```

Les piles

Empiler : ajouter un élément au sommet de la pile

```
pile.Push(2);
pile.Push(1);
pile.Push(5);
```

Parcourir une pile

```
foreach (int elt in pile)
    Console.WriteLine(elt);
// affiche 5 1 2
```

Les piles

Empiler : ajouter un élément au sommet de la pile

```
pile.Push(2);
pile.Push(1);
pile.Push(5);
```

Parcourir une pile

```
foreach (int elt in pile)
    Console.WriteLine(elt);
// affiche 5 1 2
```

Consulter un élément au sommet de la pile

```
pile.Peek();
```

Les piles

Dépiler : supprimer l'élément au sommet de la pile

```
pile.Pop();
```

Les piles

Dépiler : supprimer l'élément au sommet de la pile

```
pile.Pop();
```

Supprimer tous les éléments de la pile

```
pile.Clear();
```

Les énumérations

Une énumération, c'est quoi ?

- un ensemble de constantes nommées
- ne pouvant pas être déclaré dans le `Main`

Les énumérations

Déclaration d'une énumération

```
enum Sports { Foot, Hand, Hockey, Tennis, Basket };
```

Par défaut, le premier élément a la valeur 0.
L'élément successif a une valeur augmenté de 1.

Exemple

```
Console.WriteLine(Sports.Tennis); // affiche Tennis
```

On peut utiliser une énumération comme un type

```
Sports sport = Sports.Foot;
Console.WriteLine(sport); // affiche Foot
```

Récupérer la valeur

```
Console.WriteLine((int)sport); // affiche 0
Console.WriteLine((int)Sports.Tennis); // affiche 3
```

Les énumérations

Déclaration d'une énumération + modification des constantes par défaut

```
enum Sports { Foot=2, Hand, Hockey=5, Tennis, Basket
    };
```

Exemple

```
Console.WriteLine((int)Sports.Hand); // affiche 3
Console.WriteLine((int)Sports.Hockey); // affiche 5
Console.WriteLine((int)Sports.Tennis); // affiche 6
```

Les boucles

Syntaxe `while`

```
while (condition) // tant que l'expression logique
   est vraie
{
        // ce traitement est fait
}
```

Syntaxe `do while`

```
do // repeter
{
        // ce traitement
}
while (condition); // tant que l'expression logique
   est vraie
```

Attention aux boucles infinies, pensez à une condition d'arrêt

Les boucles

Remarques

Dans ces structures itératives, on peut utiliser :

- `break` : pour quitter la boucle
- `continue` : pour ignorer l'itération courante

Les boucles

Exemple avec `break`

```
int j = 5;
do
{
    Console.WriteLine(j);
    if (j == 3)
        break;
    j--;
}
while (j > 0);
```

Les boucles

Exemple avec `break`

```
int j = 5;
do
{
    Console.WriteLine(j);
    if (j == 3)
        break;
    j--;
}
while (j > 0);
```

Affichage : 5 4 3

Les boucles

Exemple avec `continue`

```
int j = 5;
while (j > 0)
{
    if (j == 3)
    {
        j--;
        continue;
    }
    j--;
    Console.WriteLine(j);
}
```

Les boucles

Exemple avec `continue`

```
int j = 5;
while (j > 0)
{
    if (j == 3)
    {
        j--;
        continue;
    }
    j--;
    Console.WriteLine(j);
}
```

Affichage : 5 4 2 1