C# Synthèse

Bases

Du C au C#

- Type natifs

Type C#	Type .NET	Donnée représentée	Suffixe des valeurs littérales	Codage	Domaine de valeurs
`char`	Char (S)	caractère		2 octets	caractère Unicode (UTF-16)
`string`	String (C)	chaîne de caractères			référence sur une séquence de caractères Unicode
`int`	Int32 (S)	nombre entier		4 octets	$[-2^{31}, 2^{31}-1]$ [–2147483648, 2147483647]
`uint`	UInt32 (S)	..	U	4 octets	$[0, 2^{32}-1]$ [0, 4294967295]
`long`	Int64 (S)	..	L	8 octets	$[-2^{63}, 2^{63}-1]$ [–9223372036854775808, 9223372036854775807]
`ulong`	UInt64 (S)	..	UL	8 octets	$[0, 2^{64}-1]$ [0, 18446744073709551615]
`sbyte`		..		1 octet	$[-2^{7}, 2^{7}-1]$ [-128,+127]
`byte`	Byte (S)	..		1 octet	$[0, 2^{8}-1]$ [0,255]
`short`	Int16 (S)	..		2 octets	$[-2^{15}, 2^{15}-1]$ [-32768, 32767]
`ushort`	UInt16 (S)	..		2 octets	$[0, 2^{16}-1]$ [0,65535]
`float`	Single (S)	nombre réel	F	4 octets	$[1.5\ 10^{-45}, 3.4\ 10^{+38}]$ en valeur absolue
`double`	Double (S)	..	D	8 octets	$[-1.7\ 10^{+308}, 1.7\ 10^{+308}]$ en valeur absolue
`decimal`	Decimal (S)	nombre décimal	M	16 octets	$[1.0\ 10^{-28}, 7.9\ 10^{+28}]$ en valeur absolue avec 28 chiffres significatifs
`bool`	Boolean (S)	..		1 octet	true, false
`object`	Object (C)	référence d'objet			référence d'objet

Du C au C#

- Structures algorithmiques

STRUCTURE DE CHOIX

SIMPLE

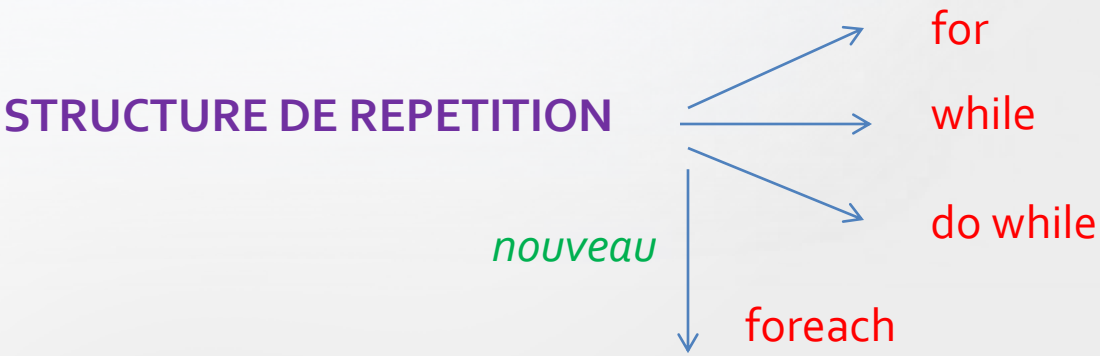
```
syntaxe : if (condition)
 {
actions_condition_vraie;
}
else
{
actions_condition_fausse;
}
```

CAS MULTIPLES

```
switch(expression) {
case v1:   actions1;
break;
case v2:   actions2;
break;
default:   actions_sinon;
```

Du C au C#

- Structures algorithmiques



PARCOURS DE LISTE

syntaxe :

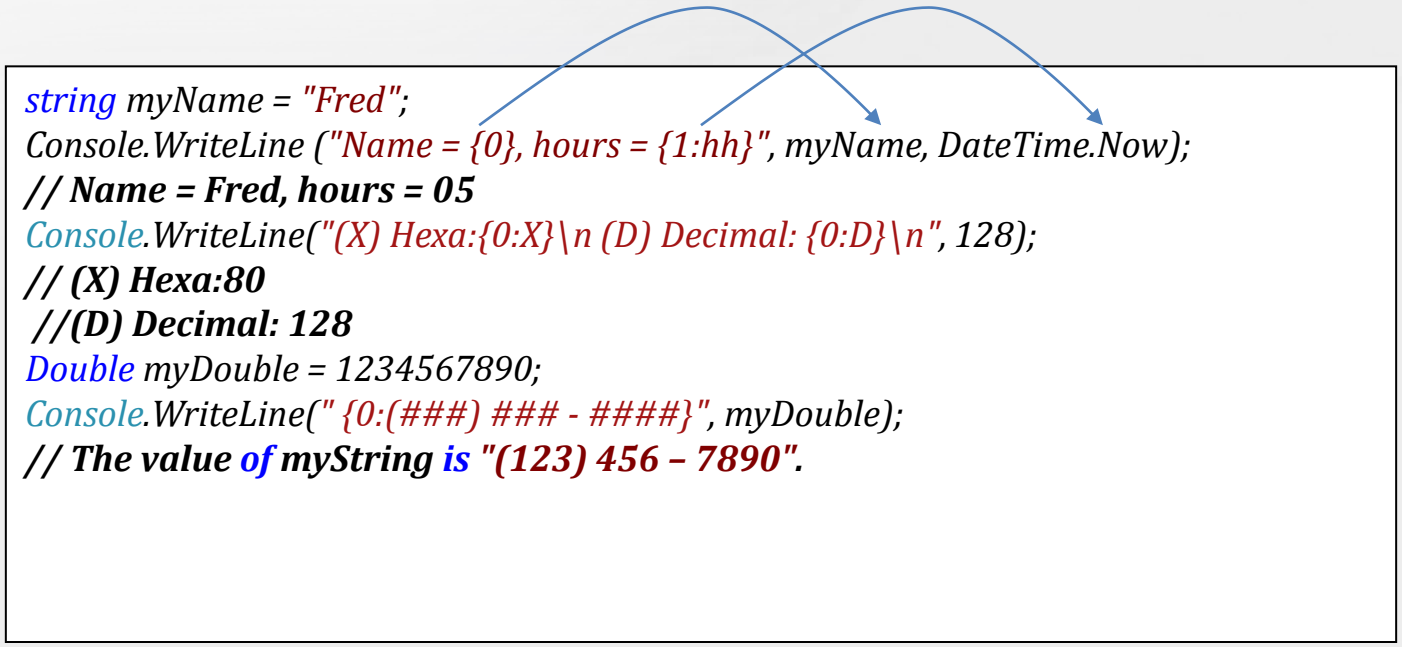
foreach *(Type variable in collection)*
{
instructions;
}

```
string[] amis = { "paul", "hélène", "jacques",
"sylvie" };
foreach (string nom in amis) {
    Console.WriteLine(nom);
 }
```

Entrées Sorties

- Sortie Ecran
  - Classe Console

Console.Write (String, Object, Object) où String est une chaîne type formateur



```
string myName = "Fred";
Console.WriteLine ("Name = {0}, hours = {1:hh}", myName, DateTime.Now);
// Name = Fred, hours = 05
Console.WriteLine("(X) Hexa:{0:X}\n (D) Decimal: {0:D}\n", 128);
// (X) Hexa:80
 //(D) Decimal: 128
Double myDouble = 1234567890;
Console.WriteLine(" {0:(###) ### - ####}", myDouble);
// The value of myString is "(123) 456 – 7890".
```

Entrées Sorties

- Entrée clavier
  - Méthode ReadLine de la classe Console Console.ReadLine ()
  - Résultat de la saisie toujours en String
    - Nécessite une opération de conversion de type si stockage autre que String

Conversion string vers float

```
static void Main(string[] args)
        {
            float a, b, c; // coeff equation
            Console.Write("Entrer a=");
            a = float.Parse(Console.ReadLine());
            Console.Write("Entrer b=");
            b = float.Parse(Console.ReadLine());
            Console.WriteLine("Entrer c=");
            c = float.Parse(Console.ReadLine());
        }
```

Manipulation des chaînes

- Classe String
  - OUBLIEZ les tableaux de caractères
    - char maChaine[10]
  - Facilité de manipulation par le biais des méthodes offertes

Methods

	Name	Description
	Clone	Returns a reference to this instance of String.
s	Compare	Overloaded. Compares two specified String objects and returns an integer that indicates their relationship to one another in the sort order.
s	CompareOrdinal	Overloaded. Compares two String objects by evaluating the numeric values of the corresponding Char objects in each string.
	CompareTo	Overloaded. Compares this instance with a specified object or String and returns an integer that indicates whether this instance precedes, follows, or appears in the same position in the sort order as the specified object or String.
s	Concat	Overloaded. Concatenates one or more instances of String, or the String representations of the values of one or more instances of Object.
	Contains	Returns a value indicating whether the specified String object occurs within this string.
s	Copy	Creates a new instance of String with the same value as a specified String.
	CopyTo	Copies a specified number of characters from a specified position in this instance to a specified position in an array of Unicode characters.
	EndsWith	Overloaded. Determines whether the end of an instance of String matches a specified string.
s	Equals	Overloaded. Determines whether two String objects have the same value.

Manipulation des chaînes

- Exemples

```
string ch;
String ch = "abcdef" ;
string s2,s1="abc" ;
s2 = s1+"def";
//-- tests d'égalité avec l'opérateur = =
if( s2 == "abcdef" )
        System.Console.WriteLine("s2==abcdef");
else
        System.Console.WriteLine ("s2<>abcdef");
if( s2 == ch )
        System.Console.WriteLine("s2==ch");
else
        System.Console.WriteLine ("s2<>ch");
```

1

```
string pass1 = "PasswoRd";
string pass2 = "PaSSwoRD";
string pass3 = "password";

Console.WriteLine(pass1.ToUpper() == "PASSWORD");
Console.WriteLine(pass2.ToUpper() == "PASSWORD");
Console.WriteLine(pass3.ToUpper() == "PASSWORD");
```

2

```
string book = "Introduction to C# book";
int index = book.IndexOf("C#");

Console.WriteLine(index);
// index = 16
```

3

```
string path = "C:\\Pics\\CoolPic.jpg";
string fileName = path.Substring(8, 7);
// fileName = "CoolPic"
```

4

Manipulation des chaînes

- Formatage et conversions
  - numérique vers string
    - Méthode ToString

```
DateTime currentDate = DateTime.Now;
Console.WriteLine(currentDate.ToString());
```

  - string vers numérique
    - Méthode Parse

```
string text = "53";
int intValue = int.Parse(text);
// intValue = 53
```

  - *Composite :*
    - *numérique /date/ string -> string*
      - Méthode Format

```
string myName = "Fred";
string result =String.Format("Name = {0}, hours = {1:hh}, minutes =
{1:mm}", myName, DateTime.Now);
```

    - *string-> numérique /date/ string*
      - Pas d'équivalent au sscanf du C!
      - Utilisez lesméthodes offertes par la classe String
        - » Split, .Parse .Substring

Notion de classe

- Une classe
  - Permet au concepteur de définir un nouveau type d'objet
  - Mot clé: class
  - Elle est constituée schématiquement de:
    - Données membres -> état de l'objet
    - Fonctions membres -> traitement des données

```
class Etudiant
  {
     public float note;
     public string nomEtudiant;
  }
```

Mauvaise pratique!

```
public class Chat
{
   private uint  age;

   public void getAge (void)
  {return (age);}

   public void setAge (uint x)
   {age=x;}

}
```

Notion de classe

- Instance d'une classe
  - Appelée aussi objet (ou instance de la classe)
  - Est une réalisation particulière d'un type donnée
    - Ex: Un 'objet' Labrador est une instance de la classe d'objet Chien
  - Chaque objet possède son propre état indépendamment des autres
    - Possibilité cependant de 'partager ' une donnée grâce à *static*

```
class Etudiant
  {
     public float note;
     public string nomEtudiant;
  }
```

```
Etudiant unEtud;
unEtud= new Etudiant();
unEtud.note=10;
unEtud.nomEtudiant =“arthur”;
```

```
public class Chat
{
   private uint  age;

   public void getAge (void)
  {return (age);}

   public void setAge (uint x)
   {age=x;}
}
```

```
Chat unMatou= new Chat();
unMatou.age=10;//INTERDIT
unMatou.SetAge(10); //methode OK
```

Manipulation d'objets

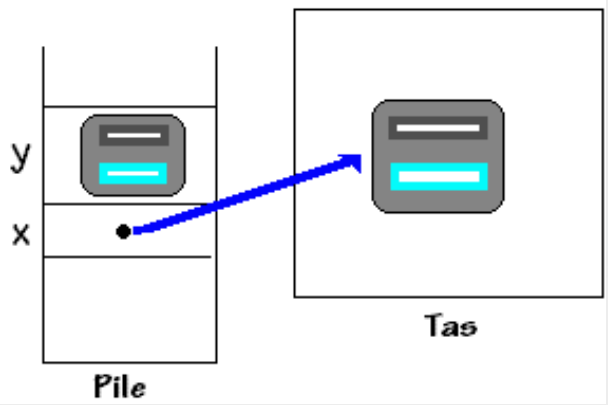
- Stockage des objets en RAM
  - 2 modes de gestion
    - Stack
    - Heap

- Variable de type valeur
  - *Le contenu de la variable reflète la valeur de la donnée manipulée*
  - *Stocké dans la pile*

- Variable de type référence
  - *Le contenu de la variable contient **"un lien"** vers la donnée située **ailleurs en mémoire**.*
  - *La variable référence est stockée dans la pile MAIS la donnée est stocké dans le TAS (heap)*

Manipulation d'objets

- Exemple 1




```
namespace Reference_et_Valeur
{
    class UnInt
    {
        public int maVariable = 10;
    }
    class Program
    {
     static void Main(string[] args)
      {
          int y = 20;
          UnInt x;
          x = new UnInt();
          Console.WriteLine(y);
          Console.WriteLine(x.maVariable);
          Console.ReadKey(); // attente appui touche
      }
    }
}
```

Automatique

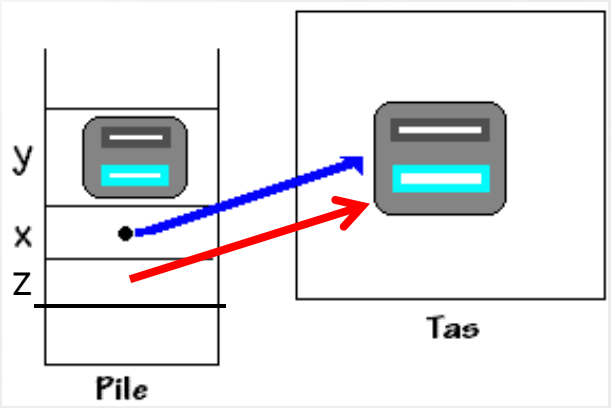
Nom	Valeur
x	{Reference_et_Valeur.UnInt}
maVariable	10
y	20

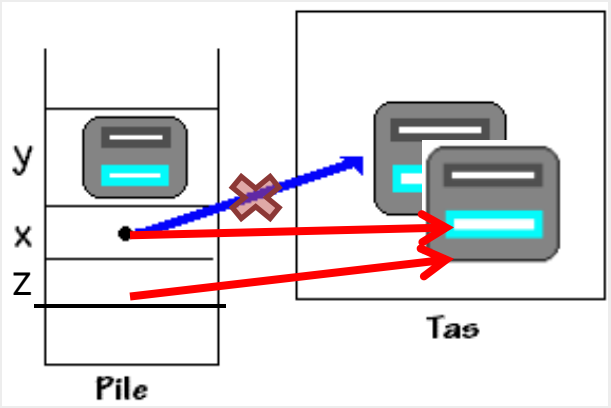
Manipulation d'objets

- Exemple 2

```
int y = 20;
UnInt x=new UnInt();

UnInt z;
z= x;
```



- Exemple 3

```
int y = 20;
UnInt x=new UnInt();

x=new UnInt();
UnInt z;
z= x;
```