BAT 4 – Polytech'Nice

Le langage C#: Concepts fondamentaux

Le langage C#: Plan

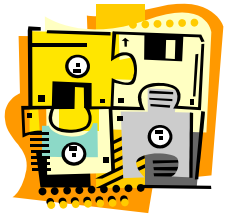
- Plan de séance
  - Comprendre les éléments fondamentaux
    - Types primitifs
    - Variables
    - Constantes
    - Expressions

C# et la plateforme .NET

- L'objectif recherché de C# est d'offrir un langage performant pour le développement .NET en plus d'être simple, facile et efficace
- C# est un nouveau langage (2000) mais il hérite des leçons apprises depuis les 30 dernières années
  - De la même façon qu'on peut dire qu'un enfant hérite de ressemblance et de caractéristiques de ses parents et grand-parents, C# est influencé de Java, C++, VB et +

Étapes de compilation d'un programme C#

Conception



Fichiers Textes
(Code source)

***.cs**

Compilation

csc *.cs

MSIL code
(127 32 0 120 45)

***.exe, dll**

JIT

Code Machine (0011 0010 1101…)

Éléments fondamentaux

- Structure d'un programme
  - Exécution d'un programme débute au Main()
  - Le mot réservé **`using`** fait référence à l'utilisation d'une classe provenant du FCL
  - Les instructions sont des commandes exécutants des actions
    - Un programme comprend plusieurs instructions séparées
    - Ceux-ci sont séparées par des ( )
    - Les {..} servent à délimiter les instructions

```
using System;
class HelloWorld {
    static void Main() {
        Console.WriteLine ("Hello, World");
    }
}
```

Éléments fondamentaux

- Comment formater le code en C#
  - Faite bonne usage de l'indentation
  - C# est sensible à la case
  - Les espaces blancs sont ignorés
  - L'utilisation de // indique une seule ligne de commentaire
  - Pour indiquer plusieurs ligne: /*…*/

```
using System;

class HelloWorld {
  static void Main() {
      Console.WriteLine ("Hello, World");
      //writes Hello, World
  }
}
```

```
/*
 * Multiple line comment
 * This example code shows how to format
 * multiple line comments in C#
 */

/* alternative use of this comment style */
```

Types prédéfinis dans le langage C#

Types dans le langage C#

- Quels sont les types prédéfinis dans C#
- Comment déclarer et initialiser une variable
- Comment déclarer et initialiser un string

Types prédéfinis dans C#

- Les types servent à la déclaration des variables
- Les variables conservent toutes sortes d'information
  - Idée: Laissé l'information déterminer le type de variable à employer
- Les types prédéfinis concernes ceux offert par C# et le Framework .NET
  - i.e. `int`, `byte`, `char`, `string`, `object`, …
  - Vous pouvez également définir les votres!
- Une variable doit toutefois être déclarée avant de pouvoir être utilisée

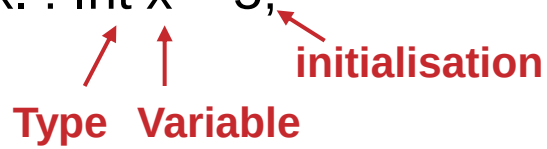
Types prédéfinis dans C#

Predefined type	Definition	# Bytes
byte	Integer between 0 and 255	1
sbyte	Integer between -128 and 127	1
short	Integer between -32768 and 32767	2
ushort	Integer between 0 and 65535	2
int	Integer between -2147483648 and 2147483647	4
uint	Integer between 0 and 4294967295	4
long	Integer between -9223372036854775808 and 9223372036854775807	8
ulong	Integer between 0 and 18446744073709551615	8
bool	Boolean value: true or false	1
float	Single-precision floating point value (non-whole number)	4
double	Double-precision floating point value	8
decimal	Precise decimal value to 28 significant digits	12
object	Base type of all other types	N/A
char	Single Unicode character between 0 and 65535	2
string	An unlimited sequence of Unicode characters	N/A

Les variables C#

Les variables C#

- Variable
  - Logiquement: permet de mémoriser une donnée à laquelle on peut référer par un nom;
  - Physiquement: une case de mémoire;
  - Nom: Un nom significatif qui réfère au contenu
  - Une variable a un type
- Une variable doit être initialisée avant d'être lue
  - Explicitement ou automatiquement
  - Ex. : Int x = 5;



- On peut aussi déclarer plusieurs variables en même temps
  - Ex. : Int compte, temps, resultat;

Comment Déclarer et initialiser une variable

- Déclarer
  - Assigner un type
  - Assigner un nom
  - Terminer avec ;
- Initialiser
  - Utiliser l'opérateur =
  - Assigner une valeur
  - Terminer avec ;

```
int numberOfVisitors;
```

```
string bear;
```

```
string bear = "Grizzly";
```

```
decimal deposit = 100M;
```

Les variables C#

- En C#, on doit respecter certaines règles pour nommer tout objets, variables et constantes.
- Identificateurs
  - Noms pour les types, les méthodes, les champs, etc.
  - Un seul mot sans espace
  - Caractère Unicode
  - Le premier caractère est soit une lettre soit ‘_’
  - Ne doit pas être un mot clé
    - Sauf si préfixé par ‘@’ (class, main, static…)

Les variables C#

- Conventions
  - Les noms de classe commencent par une majuscule.
  - La première lettre des méthodes et des variables doivent débuter en minuscule.
    - Utilisez le style (*Camel writing)*
      - *Classes = NomDeMaClasse*
      - *Méthodes = nomDeMaMéthode*
      - *Variables = nomDeMaVariable*

Les caractères d'échappements

- Du fait que les (“”) sont utilisés en C# pour délimiter le début et la fin d'une chaîne de caractères, il faut utiliser un caractère spécial. La barre oblique!

Escape sequence	Character name
\'	Single quotation mark
\"	Double quotation mark
\\	Backslash
\0	Null
\a	Alert
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab

Comment déclarer et initialiser String

- Exemple

```
string s = "Hello World"; // Hello World
```

```
string s = "\"Hello\""; // "Hello"
```

```
string s = "Hello\nWorld"; // a new line is added
```

Constantes

- La déclaration d'une constante ce fait à l'aide du mot réservé **`const`** en plus du type
- Ne varie pas/quantité fixe (ex. tps tvq)
- Vous devez obligatoirement assigner une valeur au moment de la déclaration

```
const int earthRadius = 6378;//km

const long meanDistanceToSun = 149600000;//km

const double meanOrbitalVelocity = 29.79D;//km sec
```

Autres types

- Enumeration
- Struct

- Exercice
  - Déclarer et initialiser des variables et voir leurs résultats avec le débogueur VS.NET

Expressions

Les opérateurs logiques

- Les opérateurs &&(AND), ||(OR) et !(NOT) nous permettent de construire des expressions logiques.
- Dans le cas des opérateurs && et ||
  - Aussitôt qu'on peut déterminer le résultat final de l'expression, on arrête d'évaluer l'expression
  - On arrête à la première valeur fausse pour un &&
  - On arrête à la première valeur vraie pour un ||

a	b	a && b	a || b
vrai	vrai	vrai	vrai
vrai	faux	faux	vrai
faux	vrai	faux	vrai
faux	faux	faux	faux

Opérateurs d'incrémentation et de décrémentation

- C# définit un opérateur d'incrémentation ++
  - La valeur de l'opérande est incrémentée de 1 (valeur + 1)

    Ex. count ++; ou *count = count +1;*
  - Lorsque placée avant l'opérande, la décrémentation se fait avant d'utiliser le contenu de l'opérande

    Ex. ++count;
- C# définit un opérateur de décrémentation --
  - La valeur de l'opérande est décrémentée de 1 (valeur – 1)

    Ex. count --; ou *count = count -1;*
  - Lorsque placée avant l'opérande, la décrémentation se fait avant d'utiliser le contenu de l'opérande

    Ex. -- count;

Opérateurs d'incrémentation et de décrémentation

- Si le compte contient actuellement 45, puis on effectue une incrémentation

  total = compte++;

  …assigne 45 au total et 46 à compte

- Par contre, si le compte contient actuellement 45, puis on effectue une post-incrémentation

  total = ++count;

  …assigne la valeur 46 au total et au compte

Opérateur d'assignation

- Souvent nous effectuons une opération sur une variable, puis nous « stockons » le résultat de nouveau dans cette variable.
- C# fournit des opérateurs pour simplifier ce processus

ex. `num += count;`

est équivalent à

`num = num + count;`

Opérateur d'assignation

Opérateur	Exemple	Équivalent à
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

Expressions

Common Operators	Example
• Increment / decrement	++ --
• Arithmetic	* / % + -
• Relational	< > <= >=
• Equality	== !=
• Conditional	&& || ?:
• Assignment	= *= /= %= += -= <<= >>= &= ^= |=

Quelques informations sur le TD 2

BAT4 – Polytech'Nice – 2012 - 2013

La documentation en ligne

- http://msdn.microsoft.com/fr-fr/library/
- Exemple : system.console
- Ses méthodes : http://msdn.microsoft.com/fr-fr/library/system.console_methods(v=vs.80).aspx
- Exemple : math
- http://msdn.microsoft.com/fr-fr/library/system.math_methods(v=vs.80).aspx

Ma première Classe … celle du Main()

- Squelette de la Classe principale du Main

```
class ApplicationParfaits {
    static void Main(string[ ] args) {
        ……..
    }
}
```

Et si je veux ajouter une méthode …

- Syntaxe d'un méthode de la classe principale

```
static int pgcd (int a , int b) {
…
}
```