參考：<https://www.runoob.com/csharp/csharp-tutorial.html>

[一、 基礎教程 3](#_Toc28542902)

[1. 程序結構 3](#_Toc28542903)

[2. 基本語法 3](#_Toc28542904)

[3. 數據類型 3](#_Toc28542905)

[4. 類型轉換 4](#_Toc28542906)

[5. 變數、進制、常數 4](#_Toc28542907)

[6. 運算符號 5](#_Toc28542908)

[7. 判斷式 5](#_Toc28542909)

[8. 迴圈 5](#_Toc28542910)

[9. 封裝(存取級別) 5](#_Toc28542911)

[10. 方法 6](#_Toc28542912)

[11. 可空類型 6](#_Toc28542913)

[12. 陣列(Array) 6](#_Toc28542914)

[13. 字串(String) 6](#_Toc28542915)

[14. 結構體(Struct) 6](#_Toc28542916)

[15. 枚舉(Enum) 6](#_Toc28542917)

[16. 類別(Class) 6](#_Toc28542918)

[17. 繼承 6](#_Toc28542919)

[18. 多型 6](#_Toc28542920)

[19. 運算符重載 6](#_Toc28542921)

[20. 接口(Interface) 6](#_Toc28542922)

[21. 命名空間(Namespace) 6](#_Toc28542923)

[22. 預處理器指令 6](#_Toc28542924)

[23. 正規式 6](#_Toc28542925)

[24. 異常處理 6](#_Toc28542926)

[25. 文件輸入輸出 6](#_Toc28542927)

[二、 進階教程 6](#_Toc28542928)

[1. 特性(Attribute) 6](#_Toc28542929)

[2. 反射(Reflection) 6](#_Toc28542930)

[3. 屬性(Property) 6](#_Toc28542931)

[4. 索引器(Indexer) 6](#_Toc28542932)

[5. 委託(Delegate) 6](#_Toc28542933)

[6. 事件(Event) 6](#_Toc28542934)

[7. 集合(Collection) 6](#_Toc28542935)

[8. 泛型(Generic) 6](#_Toc28542936)

[9. 匿名方法 6](#_Toc28542937)

[10. 不安全代碼 6](#_Toc28542938)

[11. 多線程 6](#_Toc28542939)

1. 基礎教程
2. 程序結構
   1. 一個C# 程序主要包括以下部分

using System; // 等同java的import

namespace test1 // 等同java的package,有大括號

{

class test1 // 類別

{

static void Main(string[] args) // 主方法

{

Console.WriteLine("Hello World");

Console.ReadKey();

}

}

}

執行程式方法：按開始(綠箭頭) 或 F5

* 1. 注意事項

C# 是大小寫敏感的。

所有的語句和表達式必須以分號（;）結尾。

程序的執行從Main 方法開始。

與Java 不同的是，文件名可以不同於類的名稱。

* 1. using語句

用法1. 匯入類別,寫在程式開頭,宣告

C#的程序第一行都是using System;

用法2. 釋放區塊,寫在程式裡面,包住某區段

using(T t = new T())

{

.........

} // 此時T.Dispose()會被呼叫

\* using區塊 和try finally 是等價的 (但不包含catch,不會幫你捕捉例外)

\* using區塊 適用釋放單一物件, try finally可釋放多物件

\* using區塊 不需手動 Dispose,Close (Stream.Close 呼叫 Dispose )

\* SqlConnection的Close是關閉連接，可以用Open再打開，Dispose則不行

1. 基本語法

C#的類別、變數、實例化new()，用法跟java一樣

保留字<https://www.runoob.com/csharp/csharp-basic-syntax.html>

1. 數據類型

C#變數有三種類型：

* 1. 值類型（Value types）

C#資料型態跟java差不多，底下只列出差異部分

bool 布林值

sbyte 1byte 有號整數 -127~128

ushort 2byte 無號整數

uint 4byte 無號整數

ulong 8byte 無號整數

想知道資料型態占多少空間可使用，sizeof( type )

* 1. 參考類型（Reference types）

object obj = 100; // 賦值給物件是'裝箱'

int a = (int) obj; // 物件轉為值類型是'拆箱'

dynamic d = 20; // 動態變數,可以是任何資料型態

// 字串特殊字元

Console.WriteLine("\""); // 用 \" 來輸出雙引號

Console.WriteLine(@"C:\"); // 單獨輸出\是不合法的,須加上@

* 1. 指針類型（Pointer types）

跟C++的指標一樣，在後面章節’不安全代碼’會討論

語法：type\* 變數; 範例：int\* iptr;

1. 類型轉換
   1. 強制轉型,可能遺失數值6.6->6

(int) 6.6;

* 1. 用 Convert.Toxxx() <=xxx是資料型態

Convert.ToInt32("1");

Convert.ToInt32(null); // 轉換null,得到 0

* 1. 用 xxx.Parse() <=xxx是資料型態

int.Parse("2");

int.Parse(null); // 轉換null,拋出例外

* 1. 用xxx.TryParse("", out i),成功true、失敗false

int i;

Int32.TryParse("3", out i); // 可用來判斷變數是否為int

\*轉型效率，實測5000次最快最慢只差0.001秒，所以可忽略

1. 變數、進制、常數

變數宣告、賦值和java都一樣，要接收user輸入的話如下

int num = Convert.ToInt32(Console.ReadLine());

前墜0表示為8進制、0x表示為16進制

後墜U和L分别表示unsigned和long

使用const關鍵字來定義常數

const double pi = 3.14;

1. 運算符號
   1. 算數運算

+加 –減 \*乘 /除 %餘 ++加一 --減一

* 1. 關係運算

==等於 !=不等於 >大於 <小於 >=大於等於 <=小於等於

* 1. 邏輯運算

&&且 ||或 !非(not)

* 1. 位運算

& 都1得1

| 有1得1

^ 只有一個1得1

~ 1得0、0得1

<< 二進制左移(乘2)

>> 二進制左移(除2)

* 1. 賦值運算

在6.1和6.4的運算符右邊加上=，先運算在賦值

* 1. 其他運算

其他運算

|  |  |  |
| --- | --- | --- |
| sizeof(type) | 取得資料型態大小 | sizeof(int) |
| typeof(類別) | 取得類別的封包路徑 | typeof(class) <=是類別不是物件 |
| & | 取得變數位址 | &a |
| \* | 變數的指針 | \*a指向變數a |
| ?: | 條件式 | a>0? 1: 0; |
| is | 判斷物件是否為某類型 | If(Ford is Car)… |
| as | 強制轉型,失敗不拋例外 | Reader r = obj as Reader |

1. 判斷式

If、if else、switch、?: <=用法都跟java一樣

1. 迴圈

while、do while、for、foreach、break、continue <=用法都跟java一樣

1. 封裝(存取級別)

存取級別

|  |  |
| --- | --- |
| public | 誰都可以存取 |
| protected | 類別內部或子類可存取 |
| private | 只有類別內部可存取 |
| internal | 相同組件(Assembly)可存取 <=同一dll內 |
| protected internal | 相同組件或子類可存取 |

1. 方法
2. 可空類型
3. 陣列(Array)
4. 字串(String)
5. 結構體(Struct)
6. 枚舉(Enum)
7. 類別(Class)
8. 繼承
9. 多型
10. 運算符重載
11. 接口(Interface)
12. 命名空間(Namespace)
13. 預處理器指令
14. 正規式
15. 異常處理
16. 文件輸入輸出
17. 進階教程
18. 特性(Attribute)
19. 反射(Reflection)
20. 屬性(Property)
21. 索引器(Indexer)
22. 委託(Delegate)
23. 事件(Event)
24. 集合(Collection)
25. 泛型(Generic)
26. 匿名方法
27. 不安全代碼
28. 多線程