

Academy .NET

Istruzioni Condizionali e Cicli

Sommario

- Confrontare i numeri
- Operatori di Comparazione
- Istruzioni Condizionali if-else
- Operatori Logici
- Ciclo For
- Ciclo Do While
- Break
- Gestione degli errori con Try-Catch
- Switch
- Operatore nameof

Confrontare i numeri

I valori possono essere confrontati in C# come in qualsiasi altro linguaggio:

```
var a = 5;  
var b = 10;  
Console.WriteLine(a < b);           // True  
Console.WriteLine(a > 0);           // True  
Console.WriteLine(a > 100);         // False  
Console.WriteLine(a < a);           // False  
Console.WriteLine(a <= 5);          // True  
Console.WriteLine(b == 2 * a);      // True
```

Operatore < (minore di)

Operatore > (maggiore di)

Operatore <= (minore o uguale)

Operatore == (uguale a)

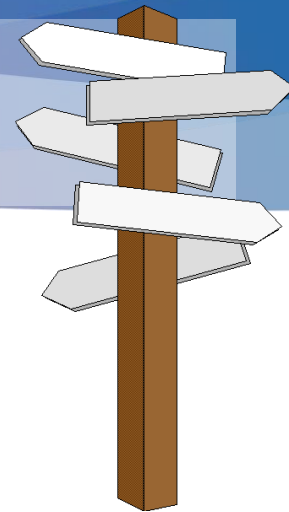
Operatori di Comparazione

Operatore	Notazione in C#	Applicabile per
Uguale a	==	stringhe / numeri / date / oggetti
Diverso da	!=	
Maggiore di	>	numeri / date / oggetti comparabili
Maggiore o uguale di	>=	
Minore di	<	
Minore o uguale di	<=	

Esempio:

```
bool risultato = (5 <= 6);  
Console.WriteLine(risultato); // True
```

Istruzioni Condizionali if-else



L'istruzione **if**

- La dichiarazione condizionale più semplice
- Verifica una condizione

Esempio: prendere come input un voto e controllare se lo studente ha superato l'esame (voto \geq 3,00):

```
var voto = double.Parse(Console.ReadLine());  
if (voto >= 3.00)  
{  
    Console.WriteLine("Superato!");  
}
```

In C# la parentesi graffa si trova in una nuova linea

Istruzioni Condizionali if-else

Le istruzioni **if** ed **else**

- Esegue un ramo se la condizione è vera e un altro se è falsa
- Esempio: aggiornare l'ultimo esempio, in modo che venga stampato "Failed!" se il segno è inferiore a 3.00:

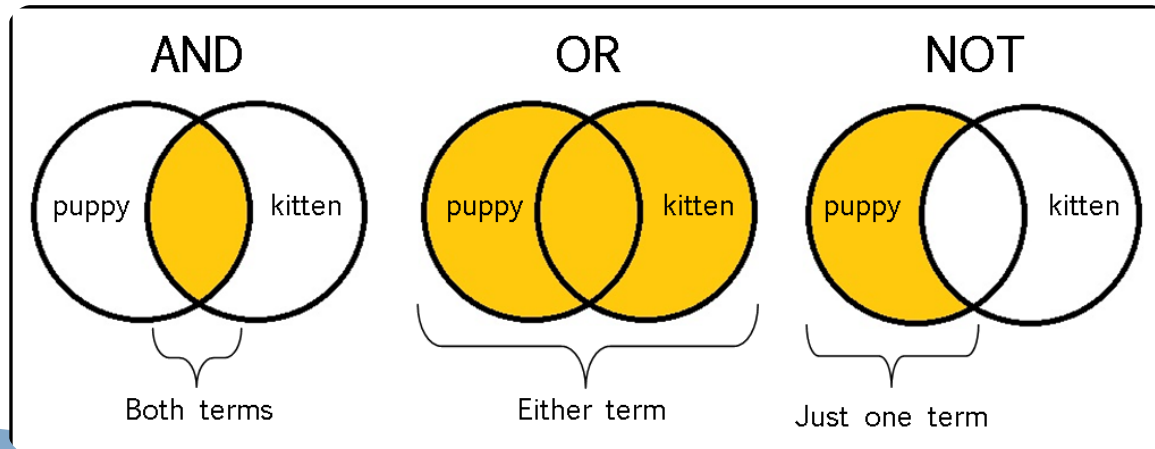
```
if (voto >= 3.00)
{
    Console.WriteLine("Passed!");
}
else
{
    // TODO: Stampa il messaggio
}
```

La parola chiave **else** si trova in una nuova linea

Operatori Logici

Gli operatori logici danno la possibilità di scrivere più condizioni in un'istruzione **if**.

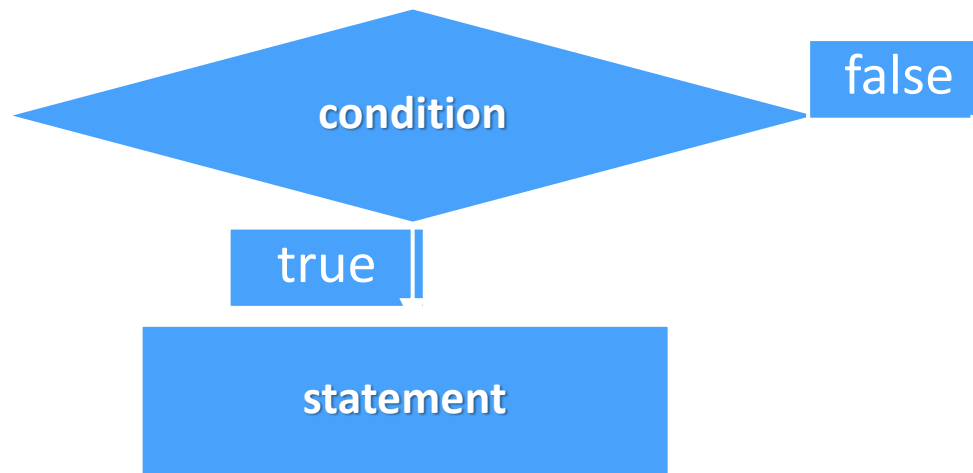
Restituiscono un valore booleano e confrontano i valori booleani .



Operatori Logici

Operatore	Notazione in C#	esempio
NOT logico	!	!false -> true
AND logico	&&	true && false -> false
OR logico		true false -> true

Ciclo While



Ciclo While

Ripete un'istruzione finchè la condizione è **true**

```
while (condition)
{
    statements;
}
```

La condizione:

Restituisce un risultato booleano **true** o **false**

Chiamata anche **loop condition**

Ciclo While

```
int counter = 0;
while (counter < 10)
{
    Console.WriteLine("Number : {0}", counter);
    counter++;
}
```

```
Number : 0
Number : 1
Number : 2
Number : 3
Number : 4
Number : 5
Number : 6
Number : 7
Number : 8
Number : 9
Press any key to continue_
```

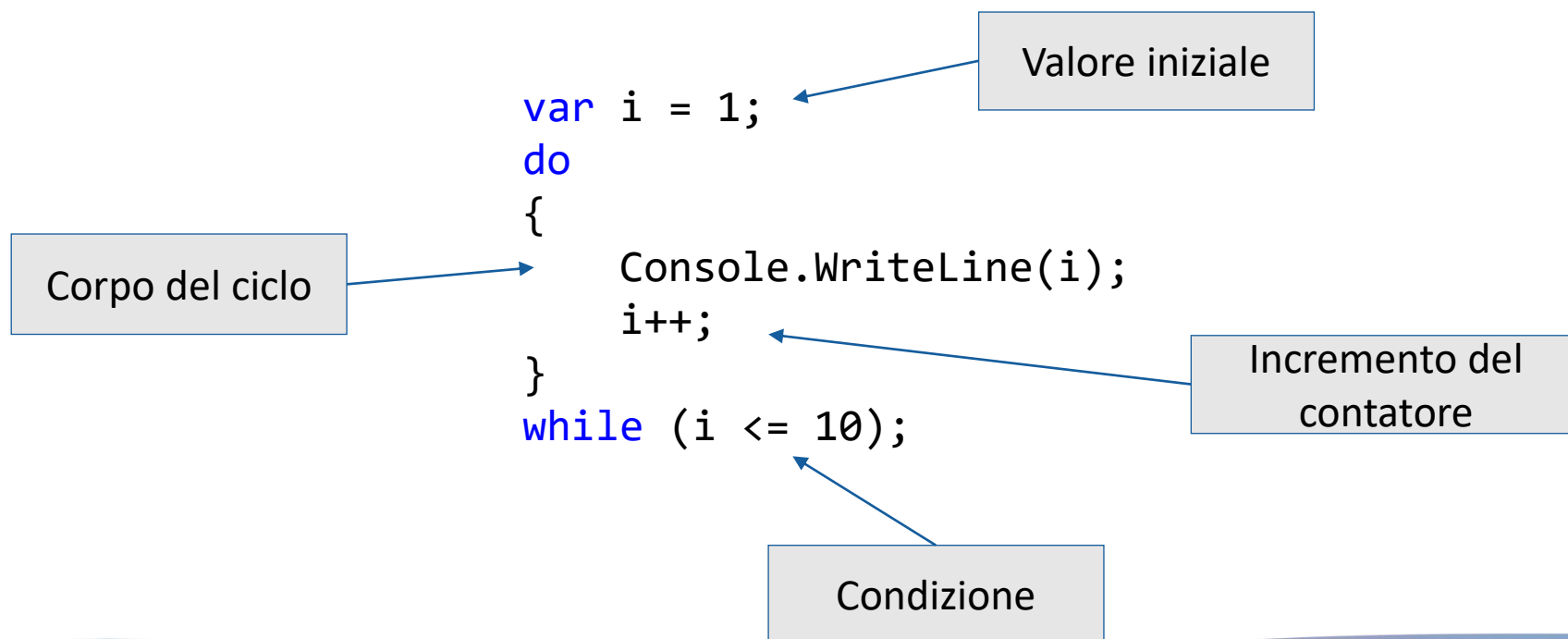
Ciclo While

Calcola e stampa la somma dei primi **n** numeri naturali

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());
int number = 1;
int sum = 1;
Console.Write("The sum 1");
while (number < n)
{
    number++;
    sum += number ;
    Console.Write("+{0}", number);
}
Console.WriteLine(" = {0}", sum);
```

Ciclo Do While

Simile al ciclo while, ma il blocco **do** viene sempre eseguito la prima volta:




La parola chiave break

La parola chiave **break** ferma il ciclo.

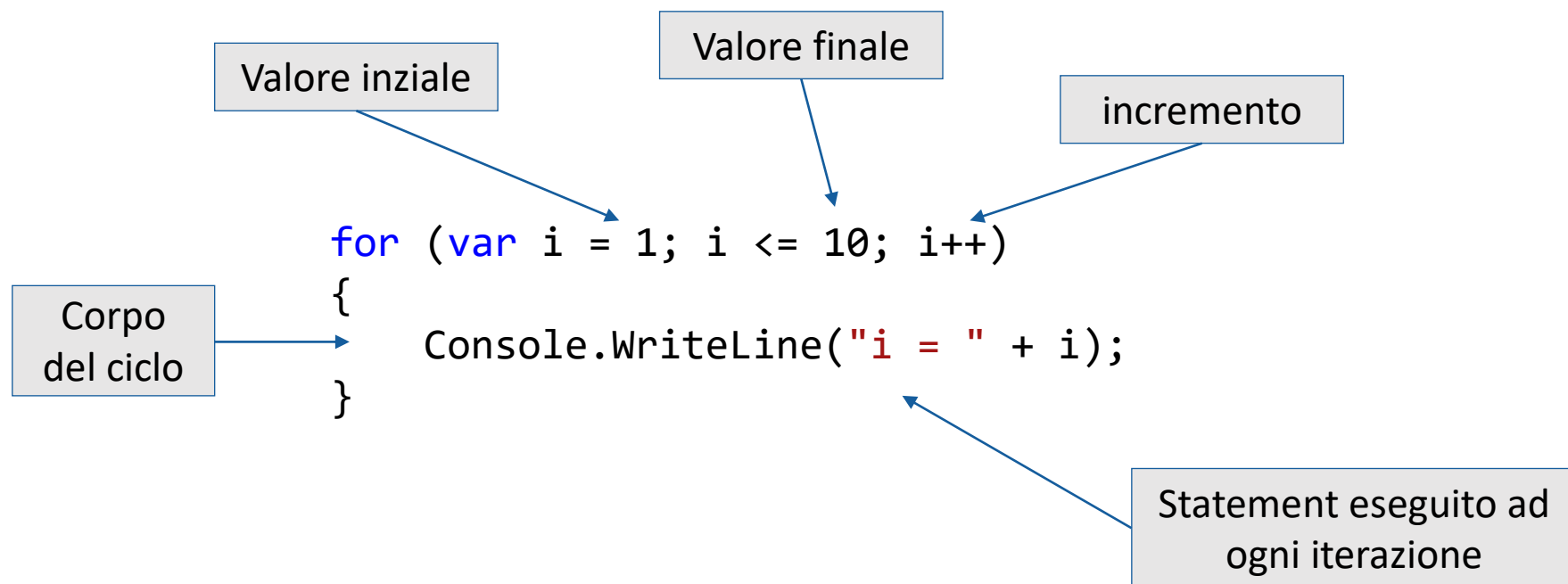
Esempio:

- Prende come input un numero e stampa il suo valore assoluto:
Se il numero è dispari passa all'istruzione Console.WriteLine(...)

```
var num = Math.Abs(int.Parse(Console.ReadLine()));  
while (true)  
{  
    if (num % 2 != 0)  
        break;   
    Console.WriteLine("The number is: {0}", num);  
}
```


Ciclo For

Il ciclo **for** esegue le istruzioni un numero fisso di volte:

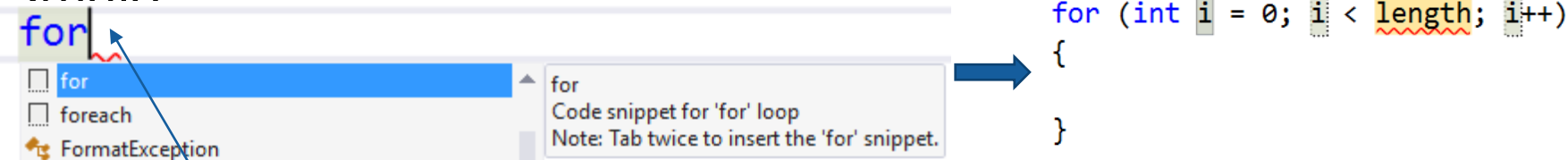


Esempio: Divisibile per 3

Stampa i numeri da 1 a 100 che sono divisibili per 3

```
for (var i = 3; i <= 100; i += 3)
{
    Console.WriteLine(i);
}
```

È possibile utilizzare lo snippet di codice "ciclo for" in Visual Studio



Ciclo Foreach

- La sintassi tipica del ciclo foreach è:

```
foreach (Type element in collection)
{
    statements;
}
```

- Ripete tutti gli elementi di una **collezione**.
- La variabile del ciclo accetta sequenzialmente tutti i valori della collezione
- La raccolta può essere **list**, **array** o altra collezione dello stesso tipo.

Ciclo Foreach: Esempio

```
string[] days = {  
    "Monday", "Tuesday", "Wednesday", "Thursday",  
    "Friday", "Saturday", "Sunday" };  
foreach (string day in days)  
{  
    Console.WriteLine(day);  
}
```

Il ciclo itera l'array di giorni.

La variabile **day** assume tutti i suoi valori

Nel ciclo foreach non possiamo impostare il valore dell'elemento corrente

Cicli Inneitati

Una composizione di cicli è chiamata **ciclo annidato**:

Un ciclo dentro un altro ciclo

Esempio:

```
int outerLoop = 0, innerLoop = 0;
    for (int i=1; i<=5; i++)
    {
        outerLoop++;
        for (int j=1; j<=5; j++)
        {
            innerLoop++;
        }
    }
    Console.WriteLine("Outer Loop runs {0} times", outerLoop);
    Console.WriteLine("Inner Loop runs {0} times", innerLoop);
```

Outer Loop runs 5 times
Inner Loop runs 25 times

Problema: Number Checker

In C# è possibile rilevare gli errori e gestirli utilizzando la logica personalizzata.

- Scrivere un programma per leggere l'input dalla console e stampare il suo tipo
Stampa "è un numero", se è un numero, altrimenti stampa "input invalido!"



Soluzione: Number Checker

```
try
{
    var n = int.Parse(Console.ReadLine());
    Console.WriteLine("è un numero");
}
catch (NumberFormatException)
{
    Console.WriteLine("input invalido!");
}
```

Tipo Eccezione



Switch Statement

Seleziona per l'esecuzione un'istruzione da un elenco a seconda del valore dell'espressione switch

```
switch (day)
{
    case 1: Console.WriteLine("Monday"); break;
    case 2: Console.WriteLine("Tuesday"); break;
    case 3: Console.WriteLine("Wednesday"); break;
    case 4: Console.WriteLine("Thursday"); break;
    case 5: Console.WriteLine("Friday"); break;
    case 6: Console.WriteLine("Saturday"); break;
    case 7: Console.WriteLine("Sunday"); break;
    default: Console.WriteLine("Error!"); break;
}
```


Switch Statement

L'espressione viene valutata:

Quando una delle costanti specificate in una **case label** è uguale all'espressione, l'istruzione che corrisponde in quel caso viene eseguita

- Se nessun caso corrisponde all'espressione:
Viene eseguito **default** se presente
In caso contrario, il controllo di flusso viene restituito al contesto esterno

Uso di switch

- Tipi di variabili come **string**, **enum** e **tipi numerici** possono essere utilizzati per l'espressione switch
- Il valore **null** è consentito come costante nelle istruzioni case
- La parola chiave **break** esce dall'istruzione switch
- Regola "no fall through": obbligo ad utilizzare break dopo ogni istruzione case
- Sono consentite più etichette che corrispondono alla stessa istruzione
- È possibile utilizzare più etichette per eseguire la stessa istruzione in più di un caso

Uso di switch - Esempio

```
switch (animal)
{
    case "dog" :
        Console.WriteLine("MAMMAL");
        break;
    case "crocodile" :
    case "tortoise" :
    case "snake" :
        Console.WriteLine("REPTILE");
        break;
    default :
        Console.WriteLine("There is no such animal!");
        break;
}
```

Operatore nameof

L'operatore **nameof** permette di ottenere in maniera semplice il nome di un qualsiasi elemento di codice, per esempio variabili, tipi, membri di un tipo e così via.

Esempio:

```
void Main()
{
    Console.WriteLine($"esecuzione di {nameof(Main)}");
    string str="hello";
    string name=nameof(str);
}
```

Operatore nameof

Nell'esempio precedente, grazie a **nameof**, viene stampato una sorta di log del metodo in esecuzione. Inoltre si ricava il nome di una variabile.

nameof evita di dover scrivere manualmente tali nomi, evita di commettere errori di battitura, e di correggerli manualmente.

Inoltre in strumenti come VisualStudio, rinominando un membro, possono essere aggiornati automaticamente anche gli operandi di nameof, cosa che non sarebbe possibile utilizzando stringhe.

Domande & approfondimenti

Academy .NET