# Fondamenti di Programmazione



#### **Arianna Bolzoni**

.Net Developer Arianna.Bolzoni@icubed.it cubed

## Codice Elegante

"I bravi programmatori sanno cosa scrivere.

I migliori sanno cosa riscrivere."

**ERIC STEVEN RAYMOND** 



#### 1. Usare nomi descrittivi

Ad esempio: Le variabili sono i soggetti I metodi sono le azioni

### 2. Dare a ogni classe uno scopo

Divisione in blocchi



### 3. Non ha bisogno di commenti per essere capito

Spiegare il come e il perché Non il cosa

### 4. Deve essere leggibile

Codice pulito != Codice intelligente

Anzi

Codice pulito > Codice intelligente



#### 5. E' riutilizzabile

Eliminare parti duplicate

#### 6. E' correttamente indentato

In termini prettamente visivi Visual studio aiuta



### 7. Scegliere l'architettura giusta

Dividere il codice in «file/classi» per concetti.

Consigli

Spesso è utile leggere e debuggare il codice "dei maestri"



# Demo

Codice Elegante
Carte di credito





#### Iterazione

 Blocco di codice continua ad essere eseguita finchè una condizione non viene soddisfatta

#### Ricorsione

 Funzione richiamata all'interno della stessa funzione ovvero espressa in termini di se stessa.



## Ricorsione

In generale una funzione di un linguaggio di programmazione è un metodo che effettua un'operazione.

Può essere richiamata in più punti e quindi anche in altre funzioni.

Come caso particolare può essere richiamata da se stessa (funzione ricorsiva)

#### Una funzione ricorsiva:

- sa come risolvere i casi base
- per risolvere il caso complesso
  - suddivide il problema in termini dello stesso problema applicato a casi più semplici
  - combina le soluzioni di ogni sottoproblema
- per ogni sottoproblema
  - conosce la soluzione oppure
  - richiama se stessa



## Esempio

Calcolare la somma dei primi N numeri positivi

#### SPECIFICA ITERATIVA

- Inizializza sum=0
- Ripeti per N volte l'operazione elementare sum = sum + i

#### SPECIFICA RICORSIVA

- considera la somma dei primi N numeri positivi (1+2+3+...+(N-1)+N)
- · come la somma di due termini:

somma dei primi N-1 numeri positivi

è facile identificare un caso base:
 la somma del primo numero positivo (N=1) vale 1.



## Esempio

Calcolare il valore del N-esimo numero della serie di Fibonacci.

La serie di Fibonacci è:

1, 1, 2, 3, 5, 8, 13, 21.....

I primi 2 elementi sono uguali a 1, il successivo è dato sempre dalla somma dei precedenti.

Es: i primi 6 numeri della serie-> 1, 1, 2, 3, 5, 8

### La funzione di Fibonacci

$$fib (n) = \begin{cases} n & se n=0 o n=1 \\ fib(n-1) + fib(n-2) & altrimenti \end{cases}$$



## Esercizio: Fattoriale

Dato un numero intero  $n \ge 0$ , calcolarne il suo fattoriale.

II fattoriale

$$f(n) = \begin{cases} 1se n=0 \\ n*f(n-1)se n>0 \end{cases}$$



	Ricorsione	Iterazione
Memoria	Consumo alto	Consumo basso
Velocità	Lenta	Veloce
Pila	Stack	Non utilizzata
Leggibilità	Più facile	Più difficile

Quindi entrambi risolvono problemi di programmazione MA

L'iterazione è da preferire rispetto alla ricorsione.



#### L'approccio iterativo

- richiede di vedere la soluzione del problema "tutta insieme" in termini di mosse elementari
- Le soluzioni iterative sono generalmente più efficienti di quelle ricorsive, in termini sia di memoria occupata sia di tempo di esecuzione.

#### L'approccio ricorsivo

- richiede invece solo di esprimere il problema in termini dello stesso problema in casi più semplici, più qualche elaborazione elementare.
- Le soluzioni ricorsive sono quindi più espressive e molto più compatte di soluzioni iterative.



Quindi entrambi risolvono problemi di programmazione

MA

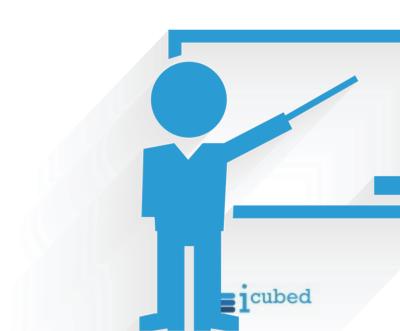
L'iterazione è da preferire rispetto alla ricorsione!



# Demo

Codice Elegante Fibonacci, fattoriale, interessi





## Memory Leak

Consumo della memoria causato dalla mancata deallocazione di variabili/risorse non più utilizzati dai processi.

Un programma rimanendo attivo, continua ad allocare memoria finchè la memoria del sistema non viene completamente consumata.

- Rallentamento delle funzionalità
- Problemi di memoria su altri programmi
- Riavvio del sistema



## Garbage Collector

Il Garbage Collector è un componente il cui ruolo è di liberare la memoria dagli oggetti non più utilizzati.

Gestisce gli oggetti allocate nel managed heap.

Agisce quando si ha necessità di avere maggiori risorse a disposizione:un oggetto può essere rimosso in una fase successive rispetto al suo inutilizzo.



## Garbage Collector

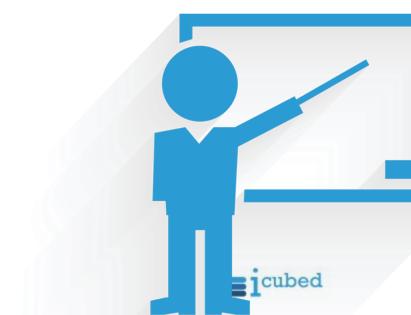
Il funzionamento del Garbage Collector è il seguente:

- Segna tutta la memoria allocata (heap) come "garbage"
- 2. Cerca i blocchi di memoria in uso e li marca come validi
- 3. Dealloca le celle non utilizzate
- 4. Compatta il managed heap



# Demo

Fibonacci, fattoriale, interessi





## File system

• Namespace : System.IO

Permette la lettura su file

• Permette la scrittura su file



## IDisposable e keyword using

• L'interfaccia **IDisposable** dichiara il metodo **Dispose** e definisce il contratto per quelle classi che devono rilasciare risorse.

```
using (StreamWriter writer = new StreamWriter("log.txt"))
{
    //...
}
Qui viene chiamato Dispose()
```

