Ricorsione vs Iterazione

Due modalità per l’esecuzione di una routine

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**)

Ricorsione

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:
  (1+2+3+...+(N-1)) + N

  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

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

Ricorsione e Iterazione

	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.

icubed

Ricorsione e Iterazione

L'approccio iterativo

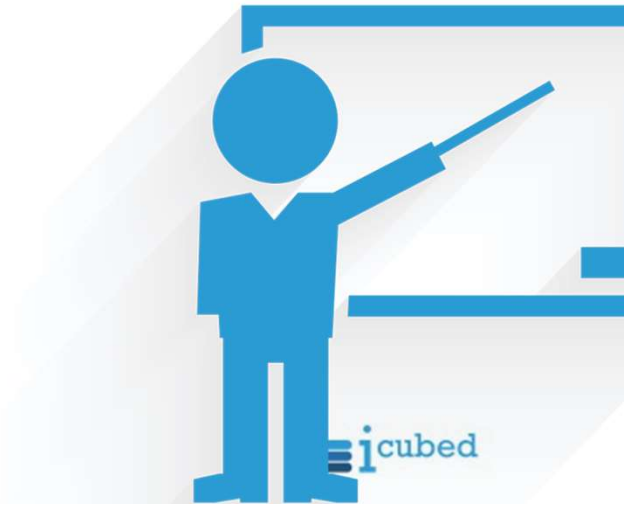
- richiede di vedere la soluzione del problema "tutta insieme" in termini di mosse elementari
- Le soluzioni iterative sono generalmente più efficienti di quelle ricorsive, in termini sia di memoria occupata sia di tempo di esecuzione.

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.

Demo

Fibonacci



icubed

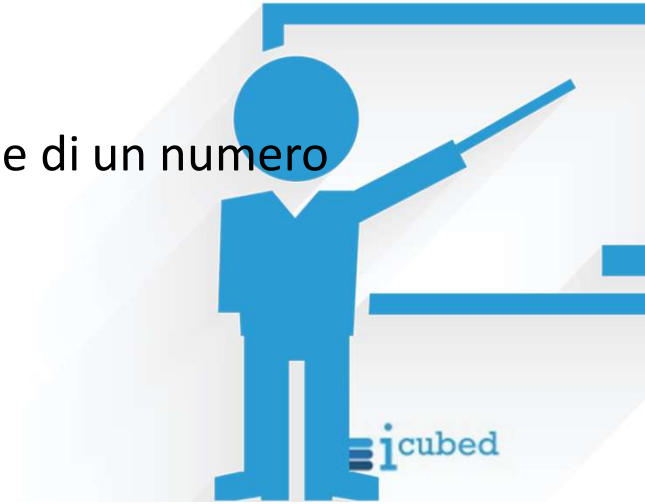
Esercitazione - Fattoriale

Dato un numero intero n >= 0, il suo fattoriale è definito come
n! = n * (n-1) * (n-2) * .... 2 * 1
Il suo valore può essere calcolato sfruttando questa regola

Il fattoriale

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

Realizzare un programma che consenta di calcolare il fattoriale di un numero utilizzando sia una versione



Commenti

- All’interno del codice è possibile definire alcuni commenti utili per la documentazione del codice

```
//Commento su una linea
```

```
/*
 Commento su più linee
 */
```

Documentazione

C# fornisce un meccanismo per documentare il proprio codice utilizzando una speciale sintassi di commento realizzata mediante **XML**

I commenti che usano tale sintassi sono detti ***commenti alla documentazione**

Documentazione

Devono precedere immediatamente un tipo definito dall'utente o un membro.

Lo strumento di generazione XML è denominato *Generatore di documentazione*. Questo generatore può essere, ma non necessariamente, il compilatore C#.

Documentazione

I commenti generatori di documentazione XML sono caratterizzati dall'avere tre barre (///)

```
/// <summary>
/// Commento di documentazione su singola linea
/// </summary>
```

Se la documentazione prende più linee il commento è delimitato da una barra e due asterischi (/**)

```
/**
 * Commento documentazione su più righe
 */
```

icubed

Documentazione - Tag

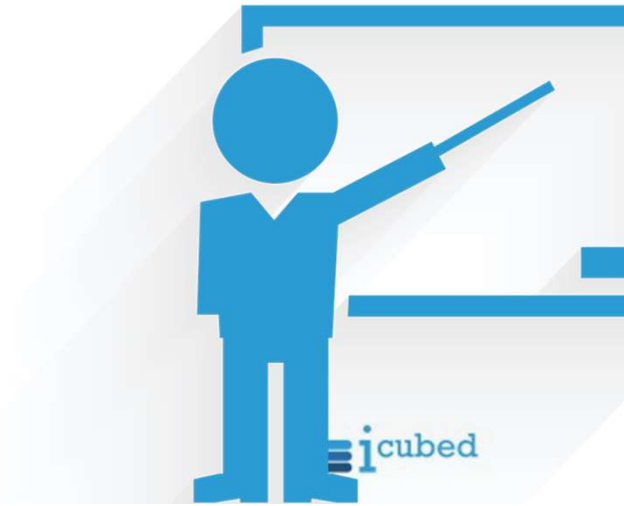
Alcuni tag più utilizzati

<summary> - tag viene usato da IntelliSense all'interno Visual Studio per visualizzare informazioni aggiuntive su un tipo o un membro

<param> - tag viene usato per descrivere i parametri.

<cref> - attributo associato a qualsiasi tag per fare riferimento ad un elemento del codice

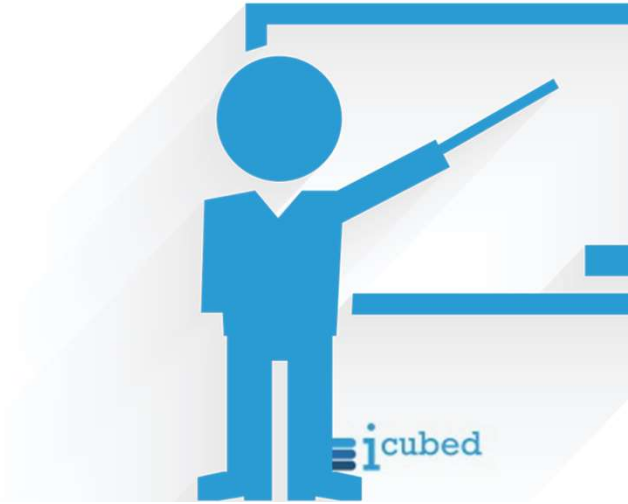
Demo



Esercitazione n. 3

Le camere di un albergo sono rappresentate da un array che contiene il nome del cliente mentre l'indice individua il numero della camera.
Carica l'array fornendo il numero della camera e il nome da inserire.
Dopo aver caricato il vettore implementa le seguenti funzionalità:

- Stampare l'elenco dei clienti presenti con il loro numero di camera
- Stampare l'elenco delle camere libere, con un messaggio se non ve ne sono
- Dato il nome di un cliente stampare la sua camera



Programmazione Procedurale

Organizzazione e suddivisione del codice in funzioni e procedure.

- Un’operazione è corrispondente a una routine, che accetta parametri iniziali e che produce eventualmente un risultato.

- Separazione tra logica applicativa e dati

Object Oriented Programming

OOP – Object Oriented Programming

- È un paradigma di programmazione

- Si basa sulla definizione e uso di diverse entità, collegate e interagenti, caratterizzate da un'insieme di informazioni di stato e di comportamenti
- Tali entità vengono denominate *Oggetti*

Oggetti

Gli oggetti possono contenere:

- Dati
- Funzioni
- Procedure

Funzioni e procedure possono sfruttare lo stato dell'oggetto per ricavare informazioni utili per la rispettiva elaborazione.

Classe

Gli oggetti sono istanze di una classe.

Una classe:
- È un reference type
- È composta da membri

I membri di una classe sono:
- Campi
- Proprietà
- Metodi
- Eventi

```
MyClass c = new MyClass();

public class MyClass {
    //...
}
```

icubed

Tipi, classi e oggetti

- Un **tipo** è una rappresentazione concreta di un concetto. Per esempio, il tipo built-in *float* fornisce una rappresentazione concreta di un numero reale. (*)

- Una **classe** è un tipo definito dall'utente. (*)

- Un **oggetto** è l'istanza di una classe caratterizzato da:
  - un'identità (distinto dagli altri);
  - un comportamento (compie elaborazioni tramite i metodi);
  - uno stato (memorizza dati tramite campi e proprietà).

(*) The C++ Programming Language, Third Edition. Bjarne Stroustrup

icubed

Classi e proprietà

- È il modo migliore per soddisfare uno dei pilastri della programmazione OOP: *incapsulamento*

- Una proprietà può provvedere accessibilità in lettura (**get**) scrittura (**set**) o entrambi.

- Si può usare una proprietà per ritornare valori calcolati o eseguire una validazione.

Classi e proprietà

Proprietà tradizionale

```
public class MyClass
{
    private string _name;

    public string Name
    {
        get { return _name; }
        set { _name = value; }
    }
}

MyClass c = new MyClass();
c.Name = "C#";
```

ReadOnly / WriteOnly

```
public class MyClass
{
    private string _name = "C#";

    public string Name
    {
        get { return _name; }
    }
}

MyClass c = new MyClass();
c.Name = "C#"; // non si può fare
Console.WriteLine(c.Name); // si può fare
```

Metodi

Definisce un comportamento o un’elaborazione relative all’oggetto.

Si definisce come una routine, quindi ha una firma in cui si definiscono eventuali parametri d’ingresso e valori di ritorno.

```
int MyMethod(string str) {
    int a = int.Parse(str);
    return a;
}
```

Istanze delle classi

- La creazione dell'istanza di una classe (ovvero un oggetto) può avvenire utilizzando la *keyword* ***new***

Convenzioni sul codice

- **Notazione ungherese:** al nome dell'identificatore viene aggiunto un prefisso che ne indica il tipo (es. intNumber identifica una variabile intera)

- **Notazione Pascal:** l'inizio di ogni parola che compone il nome dell'identificatore è maiuscola, mentre tutte le altre lettere sono minuscule (es. FullName)

- **Notazione Camel:** come la notazione Pascal, a differenza del fatto che la prima iniziale deve essere minuscola (es. fullName)

Convenzioni sul codice

Elementi	Notazione
Namespace	Notazione Pascal
Classi	Notazione Pascal
Interfacce	Notazione Pascal
Strutture	Notazione Pascal
Enumerazioni	Notazione Pascal
Campi privati	Notazione Camel
Proprietà, metodi e eventi	Notazione Pascal
Parametri di metodi e funzioni	Notazione Camel
Variabili locali	Notazione Camel

Membri e Classi statiche

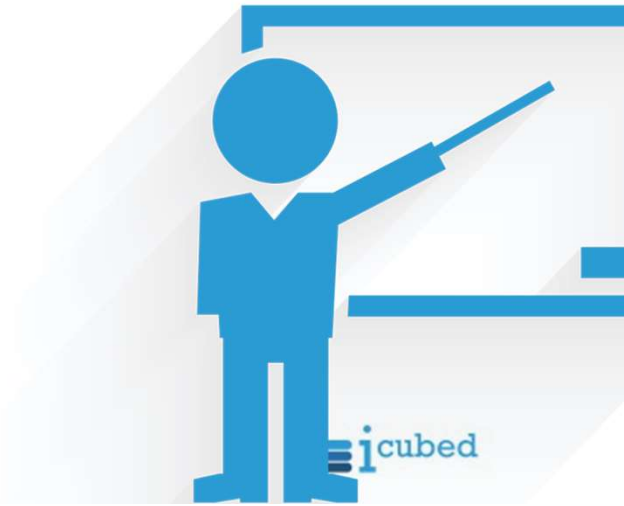
- Gli elementi direttamente associati al tipo e condivisi con tutte le istanze vengono detti membri ***statici.***

- Una classe che contiene unicamente membri statici viene anch'essa indicata come statica.

- Per indicare che una classe o un membro è statico si usa la *keyword* ***static.***

Demo

Classi e Oggetti



Demo

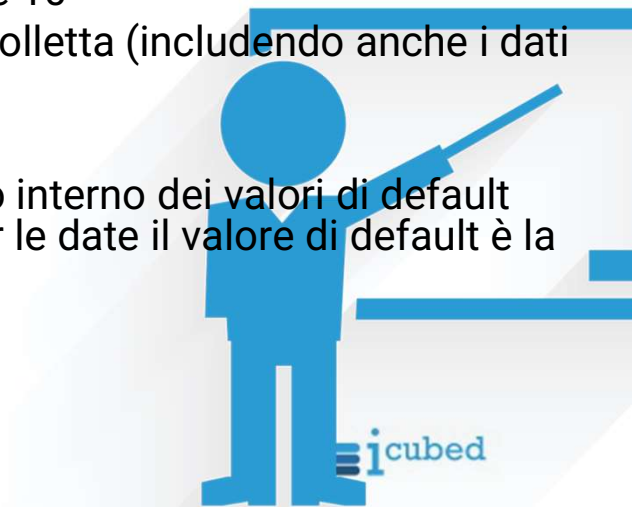
Realizzare un programma per il calcolo delle bollette.

In particolare si richiede di definire:

- la classe Utente con **codice utente** (definito come stringa), **nome**, **cognome** e **data di nascita**;
- la classe Bolletta in cui si tiene traccia di: **unità di misura(kwh, mc, minuti), consumo totale**, **data di scadenza**, **importo, tipologia bolletta (corrente, gas, telefono)** e **utente** a cui si riferisce**.**
- Implementare**:**
  - un metodo che consenta di calcolare l'importo della bolletta: il costo della bolletta che è costituito da una quota fissa di 40€ più il prodotto tra l'unità di misura scelta e 10
  - un metodo che consenta di stampare opportunamente i dati della bolletta (includendo anche i dati dell'utente).

Requisiti tecnici:

- Al momento della creazione sia l'utente che la bolletta hanno al loro interno dei valori di default specifici (per le stringhe il valore di default è una stringa ''xxxxx', per le date il valore di default è la data 01/01/2000);



icubed

Esercitazione n. 4

Scrivere un'applicazione che permetta di gestire il bollo di un veicolo:

per ogni autoveicolo occorre tenere traccia delle seguenti informazioni: (**marca, kilowatt, tipologia euro(es 1, 2, ecc), anno immatricolazione, prezzo di acquisto)**.

L'applicazione deve consentire di svolgere i seguenti casi d'uso:

- Creazione di un nuovo veicolo
- Calcola costo bollo

  Data di un'autovettura, il sistema calcola il costo del bollo.

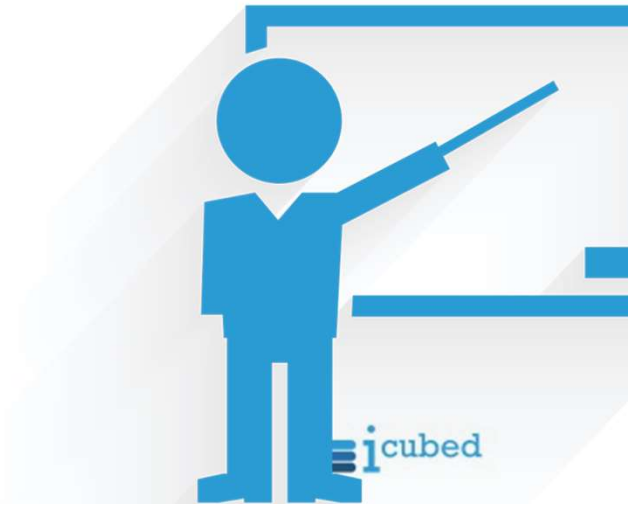
  Il bollo deve essere calcolato nel seguente modo:

  Euro 1 pagherà € 2,90 fino a 100 kW e € 4,35 oltre tale soglia

  Euro 2 pagherà € 2,80 fino a 100 kW e € 4,20 oltre tale soglia

  Euro 3 pagherà € 2,70 fino a 100 kW e € 4,05 oltre tale soglia

  Euro 4 in poi, pagherà € 2,58 fino a 100 kW e € 3,87 oltre tale
- Stampare a video i dettagli del veicolo (compreso il costo del bollo).
- Requisito tecnico: gestire opportunamente l'input dell'utente.



icubed