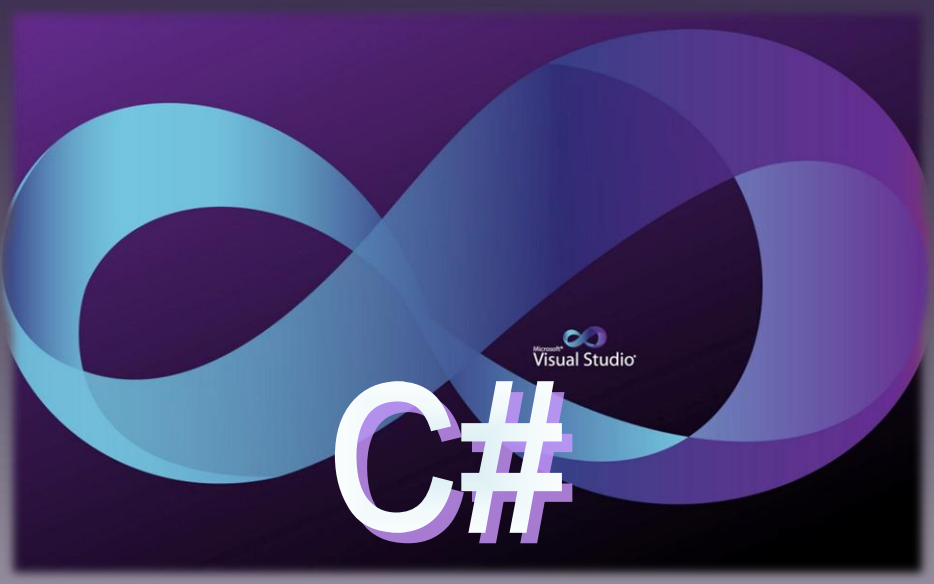
ISTITUTO TECNICO INDUSTRIALE
G. M. ANGIOY
SASSARI



CORSO DI PROGRAMMAZIONE

RICORSIONE E METODI RICORSIVI

DISPENSA 07.03

07-03_Ricorsione_[ver_16]

Versione del: 21/09/2016
Revisione numero: 15

Prof. Andrea Zoccheddu
Dipartimento di Informatica

DIPARTIMENTO
INFORMATICA E TELECOMUNICAZIONI

LA RICORSIONE E I METODI RICORSIVI

I METODI RICORSIVI

FUNZIONI E RICORSIONE

COS'È LA RICORSIONE

La ricorsione è una metodologia con cui la definizione fa riferimento a sé stessa.

Per esempio una definizione ricorsiva di un acronimo è la seguente:

```
ACIDA      =     acronimo che indica definizione acida
```

dove la parola acida compare sia a destra che a sinistra della definizione.

In matematica una definizione ricorsiva di una successione consente di definire il valore di un elemento generico in base a altri elementi della successione.

Un esempio classico di definizione ricorsiva è il fattoriale di un numero: il fattoriale di zero è 1; il fattoriale di un numero positivo K si ottiene moltiplicando K per il fattoriale di K-1.

```
F(0) = 1;
F(K) = K * F(K-1);
```

per esempio l'elemento F(1) si calcola moltiplicando 1 x F(0) = 1 x 1 = 1.

```
F(1) = 1 x F(0) = 1 x 1 = 1
```

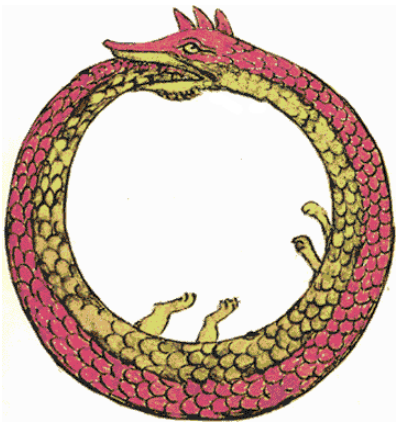
e l'elemento F(2) si calcola così:

```
F(2) = 2 x F(1) = 2 x 1 = 2
```

e l'elemento F(3) si calcola così:

```
F(3) = 3 x F(2) = 3 x 2 = 6
```

In informatica un algoritmo ricorsivo è un algoritmo definito in termini di se stesso, ovvero la cui esecuzione su un insieme di dati comporta l'applicazione dello stesso algoritmo agli insiemi di dati semplificati su un insieme di dati semplificato o ridotto.
L'algoritmo richiama se stesso causando una sequenza di invocazioni che ha termine quando si verifica una condizione particolare che viene chiamata **condizione di terminazione.**
La condizione di terminazione è un caso particolare di dato (o di insieme di dati) a cui la funzione restituisce un valore senza ricorso ad altre invocazioni di se stessa.



Da un punto di vista di efficienza computazionale una funzione ricorsiva non è sempre la migliore soluzione, poiché un algoritmo iterativo è spesso eseguito con un costo di elaborazione inferiore di un equivalente algoritmo ricorsivo. La ricorsione richiede una creazione di ambienti di funzione consecutivi con richieste esigenti in termini di memoria e di calcolo.

D'altra parte in alcuni casi un algoritmo ricorsivo è la soluzione più elegante ad un problema e alcuni problemi sono intrinsecamente ricorsivi.

FUNZIONI RICORSIVE

In molti linguaggi di programmazione è possibile definire funzioni ricorsive. Una funzione ricorsiva prevede una chiamata alla funzione stessa all'interno del suo corpo.

In Visual C# è possibile dichiarare una funzione ricorsiva nel seguente modo:

```
public Tipo MiaFunzione ( . . . )
{
      . . .
      MiaFunzione ( … );
}
```

Per poter essere definita in modo corretto è necessario che all'interno del corpo sia previsto almeno un caso in cui la funzione possa restituire un valore senza fare ricorso all'invocazione di sé stessa: questo caso è detto condizione di terminazione.

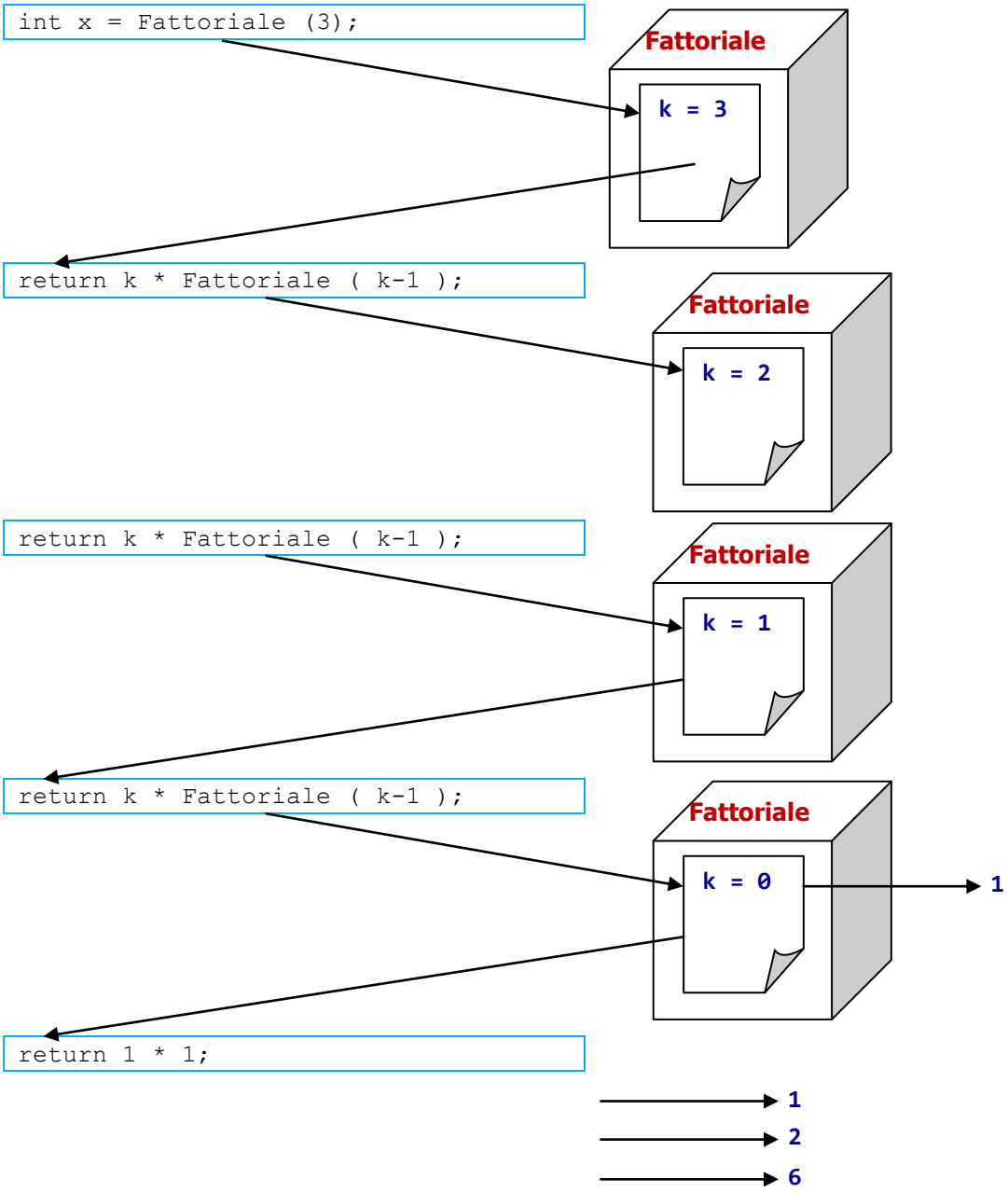
Proviamo a dichiarare una funzione in Visual C# che implementi il calcolo ricorsivo del fattoriale:

```
public int Fattoriale ( int k )
{
   if ( k == 0 )
     return 1;
   else
     return k * Fattoriale ( k-1 );
}
```

Nella funzione appena definita si può osservare che nel caso in cui l'argomento passato alla funzione sia zero, questo rientra nella condizione di terminazione e la funzione restituisce subito il risultato 1. Se invece il valore dell'argomento passato alla funzione è maggiore di zero allora per calcolare il risultato occorre invocare la funzione medesima ma sul predecessore dell'argomento e il risultato andrà moltiplicato per l'argomento stesso. In sintesi il calcolo ripercorre l'esempio già visto per la successione matematica precedente.

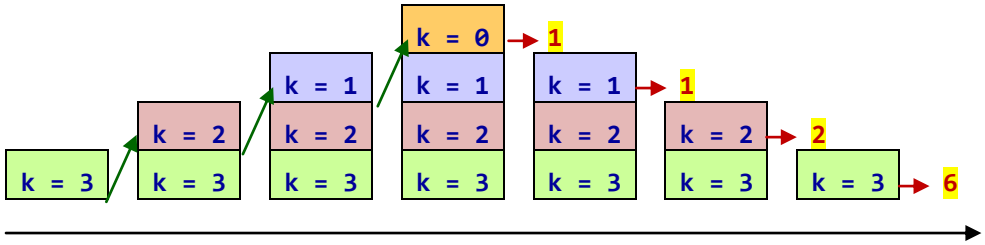
IMPLEMENTAZIONE DELLE FUNZIONI RICORSIVE

Per comprendere più approfonditamente il meccanismo di elaborazione di una funzione ricorsiva, dobbiamo riflettere sul concetto di ambiente legato all'invocazione di una funzione.



Quando una funzione (metodo) è invocato il sistema deve allocare una porzione di memoria per poter ospitare l'ambiente di quella chiamata di funzione. Nel caso di una funzione ricorsiva è normale che la stessa funzione sia richiamata più volte e conseguentemente si debba allocare memoria per ogni chiamata. Tradizionalmente queste chiamate sono allocate in una porzione di memoria denominata **STACK** (o **PILA**).

La configurazione quindi assume il seguente aspetto:



Inizialmente lo **stack** ospita un solo ambiente (invocato col argomento 3), che inizia a eseguire il corpo della funzione (o del metodo); all'interno del corpo la funzione (o il metodo) invoca se stessa causando la creazione di un nuovo ambiente (invocato col argomento 2); analogamente si esegue una chiamata ricorsiva e la creazione di un nuovo ambiente (invocato col argomento 1); infine l'ultima invocazione ricorsiva crea un nuovo ambiente (invocato col argomento 0). Le invocazioni ricorsive e la conseguente creazione di ambienti è indicata dalle frecce verdi.

Il corpo dell'ultimo ambiente si trova nella condizione di terminazione ed è quindi in grado di restituire un valore senza ulteriori chiamate ricorsive; il metodo (con k=0) rende 1 e l'ambiente viene eliminato (e la memoria recuperata). Il valore reso è evidenziato come il numero rosso al fianco dell'ambiente in cima alla pila corrente.

Il corpo del penultimo ambiente riceve il valore (1) dall'ambiente appena distrutto e può elaborare il prodotto nel return (1 x 1 = 1).

Il corpo del terzultimo ambiente riceve il valore (1) dall'ambiente appena distrutto e può elaborare il prodotto nel return (2 x 1 = 2).

Il corpo del terzultimo ambiente riceve il valore (2) dall'ambiente appena distrutto e può elaborare il prodotto nel return (3 x 2 = 6). Questo è il valore reso alla chiamata esterna al metodo (la prima originale chiamata).

ESEMPI

Una funzione ricorsiva deve innanzitutto porsi la questione della condizione di terminazione. Successivamente deve interrogarsi sul modo corretto di applicare una invocazione ricorsiva.

RICERCA DEL MASSIMO COMUN DIVISORE

La funzione MCD può essere sia iterativa che ricorsiva.

La condizione di terminazione può essere il caso in cui si cerca il MCD di due numeri uguali; in questo caso il MCD è il numero stesso.

Nel caso in cui invece i due numeri sono diversi, allora occorre riflettere che uno dei due è sicuramente maggiore dell'altro. Sia M il più grande e N il più piccolo; allora il MCD è lo stesso dei numeri M ed (M-N). Il motivo è una semplice dimostrazione matematica, lasciata per esercizio.

L'algoritmo potrebbe essere quindi implementato nel seguente modo:

```
public int MCD(int a, int b)
{
      if (a == b)
            return a;
      else if (a > b)
            return MCD(b, a - b);
      else
            return MCD(a, b - a);
}
```

CONVERSIONE DI UN NUMERO INTERO POSITIVO IN STRINGA BINARIA

La funzione riceve un numero intero positivo.

I casi di terminazione sono i numeri composti da una sola cifra. 0 rende "0"; invece 1 rende "1".

Il caso ricorsivo deve invocare il metodo sul quoziente e concatenare il resto della divisione del numero per due.

L'algoritmo potrebbe essere quindi implementato nel seguente modo:

```
public string IntToBin(int n)
{
      if (n == 0)
            return "0";
      else if (n == 1)
            return "1";
      else
            return IntToBin(n / 2) + Convert.ToString(n % 2);
}
```

POTENZA

Scrivere una funzione C# ricorsiva che calcoli, dati due numeri interi M ed N, la potenza M^N.

L'algoritmo potrebbe essere implementato nel seguente modo:

```
public int Potenza(int m, int n)
{
     if (n > 0)
          return m * Potenza(m, n - 1);
     else return 1;
}
```

FIBONACCI

Scrivere una funzione C# ricorsiva che calcoli il valore dell'n-simo termine della successione di Fibonacci.

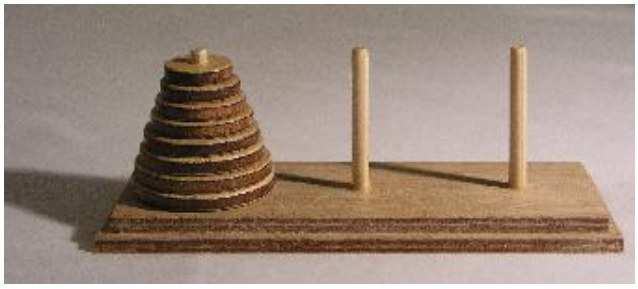
La serie di Fibonacci è una sequenza di numeri interi, i cui primi due sono 1 e 1. Ogni successivo elemento della sequenza si ottiene sommando i precedenti due elementi. Per esempio, il terzo numero della sequenza è la somma del primo e del secondo, ossia vale 2. Quindi, i primi tre elementi della sequenza sono 1 1 2. Il quarto si ottiene sommando gli ultimi due, cioè 1 e 2, e quindi vale 3. Dato che ora abbiamo 1 1 2 3, il quinto si ottiene sommando gli ultimi due, e quindi vale 2+3=5.

L'algoritmo potrebbe essere implementato nel seguente modo:

```
public int Fibonacci(int n)
{
    if (n < 2)
        return 1;
    else
        return Fibonacci(n-1) + Fibonacci(n-2);
}
```

LA TORRE DI HANOI

http://it.wikipedia.org/wiki/Torre_di_Hanoi



La Torre di Hanoi è un rompicapo matematico composto da tre paletti e un certo numero di dischi di grandezza decrescente, che possono essere infilati in uno qualsiasi dei paletti.

Il gioco inizia con tutti i dischi incolonnati su un paletto in ordine decrescente, in modo da formare un cono. Lo scopo del gioco è portare tutti i dischi sull'ultimo paletto, potendo spostare solo un disco alla volta e potendo mettere un disco solo su un altro disco più grande, mai su uno più piccolo.

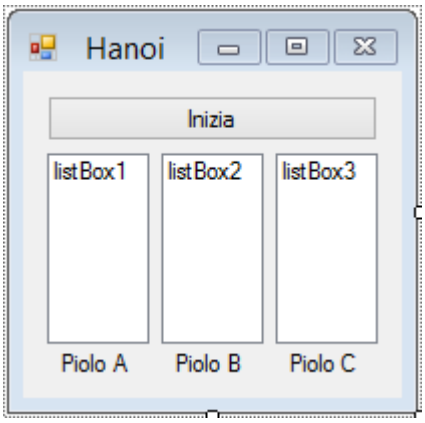
La soluzione base del gioco della torre di Hanoi si formula in modo ricorsivo.

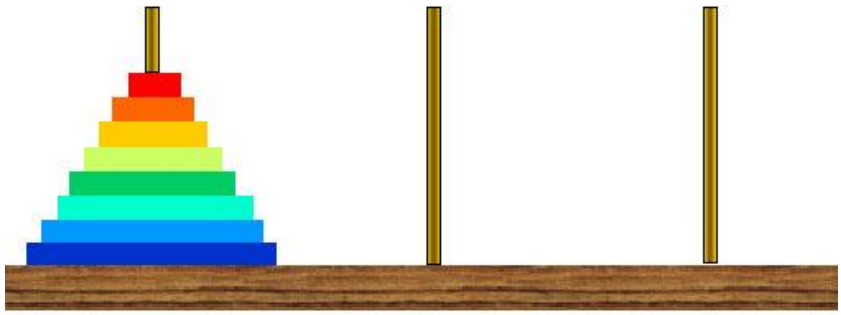
Siano i paletti etichettati con A, B e C, e i dischi numerati da 1 (il più piccolo) a n (il più grande). L'algoritmo si esprime come segue:

- Sposta i primi n-1 dischi da A a B. (Questo lascia il disco n da solo sul paletto A)
- Sposta il disco n da A a C
- Sposta n-1 dischi da B a C

Per spostare n dischi si richiede di compiere un'operazione elementare (spostamento di un singolo disco) ed una complessa, ossia lo spostamento di n-1 dischi. Tuttavia anche questa operazione si risolve nello stesso modo, richiedendo come operazione complessa lo spostamento di n-2 dischi. Iterando questo ragionamento si riduce il processo complesso ad uno elementare, ovvero lo spostamento di n-(n-1)=1 disco.

Questo è un algoritmo ricorsivo di complessità esponenziale.





La *Torre di Hanoi* con otto dischi e tre colonnine.

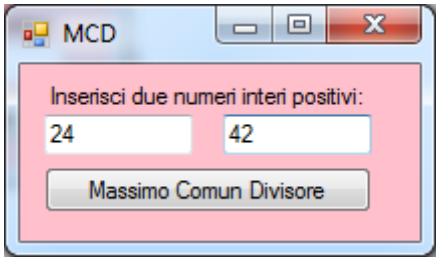
ESERCIZI

ESERCIZI SU: FUNZIONI

ESERCIZIO 1. MCD

Definire un metodo ricorsivo che calcola il massimo comun divisore tra due numeri interi positivi.

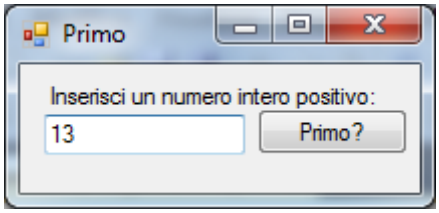
L'utente scrive nelle due textBox dei numeri; il pulsante invoca la funzione e poi visualizza un messaggio con l'esito restituito.



ESERCIZIO 2. SUCCESSIONE MATEMATICA

Definire un metodo ricorsivo che restituisce un valore intero così definito:

- S(0) = 1;
- S(n) = 1 + S(n / 2)

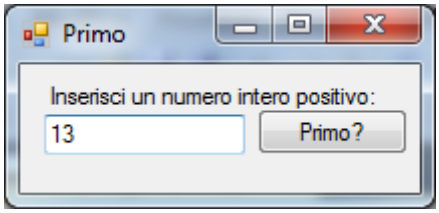


L'utente scrive un numero nella textBox; il pulsante invoca la funzione e poi visualizza un messaggio con l'esito restituito.

ESERCIZIO 3. SUCCESSIONE MATEMATICA

Definire un metodo ricorsivo che restituisce un valore intero così definito:

- S(1) = 1;
- Se n è pari, S(n) = S(n / 2)
- Se n è dispari, S(n) = 3 + S(n -1)

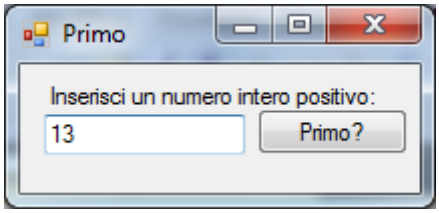


L'utente scrive un numero nella textBox; il pulsante invoca la funzione e poi visualizza un messaggio con l'esito restituito.

ESERCIZIO 4. SUCCESSIONE MATEMATICA

Definire un metodo ricorsivo che restituisce un valore intero così definito:

- S(0) = 0;
- Se n è positivo, S(n) = 1 + S(–n + 1)
- Se n è negativo, S(n) = 1 + S(–n)



L'utente scrive un numero nella textBox; il pulsante invoca la funzione e poi visualizza un messaggio con l'esito restituito.

SOMMARIO