



Matlab: Ricorsione e Variabili Funzione

Informatica B

Ricorsione

- ❑ Che cos'è la ricorsione?
 - ▶ Un sottoprogramma P richiama se stesso (ricorsione diretta)
 - ▶ Un sottoprogramma P richiama un'altro sottoprogramma Q che comporta un'altra chiamata a P (ricorsione indiretta)
- ❑ A cosa serve?
 - ▶ È una tecnica di programmazione molto potente
 - ▶ Permette di risolvere in maniera elegante problemi complessi

- ❑ Per risolvere un problema attraverso la programmazione ricorsiva sono necessari alcuni elementi
 - ▶ **Caso base:** caso elementare del problema che può essere risolto immediatamente
 - ▶ **Passo ricorsivo:** chiamata ricorsiva per risolvere uno o più problemi più semplici
 - ▶ **Costruzione della soluzione:** costruzione della soluzione sulla base del risultato delle chiamate ricorsive

Esempio: il fattoriale

- ❑ Definizione:

$$f(n) = n! = n*(n-1)*(n-2)*...*3*2*1$$

- ❑ Passo ricorsivo:

$$f(n) = n*f(n-1)$$

- ❑ Caso base:

$$f(0)=1$$

```
function [f]=factRic(n)
    if (n==0)
        f=1;
    else
        f=n*factRic(n-1);
    end
```

Funzionamento ricorsione

```
function [f]=factRic(n)
    if (n==0)
        f=1;
    else
        f=n*factRic(n-1);
    end
```

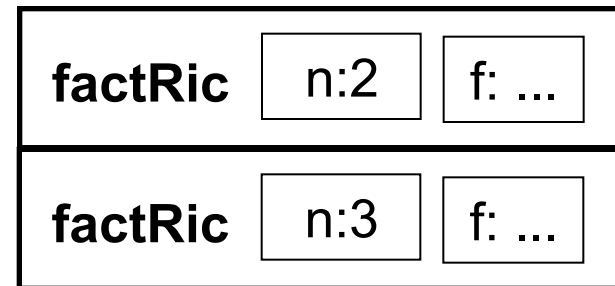
factRic

n:3

f: ...

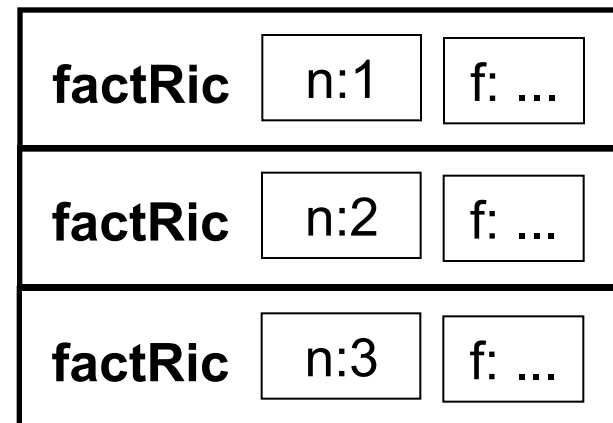
Funzionamento ricorsione

```
function [f]=factRic(n)
    if (n==0)
        f=1;
    else
        f=n*factRic(n-1);
    end
```



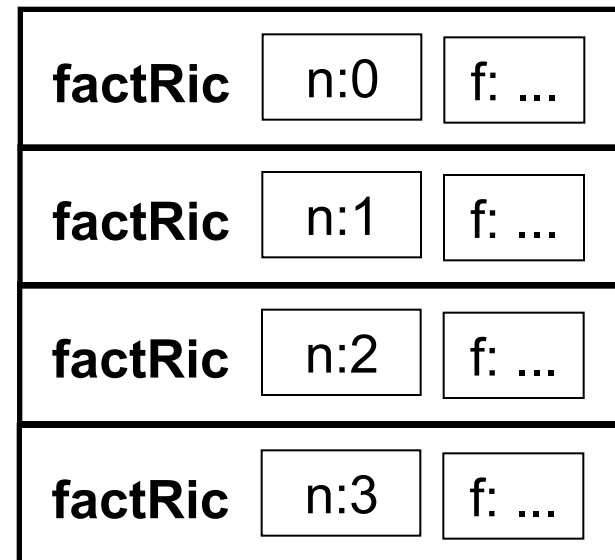
Funzionamento ricorsione

```
function [f]=factRic(n)
    if (n==0)
        f=1;
    else
        f=n*factRic(n-1);
    end
```



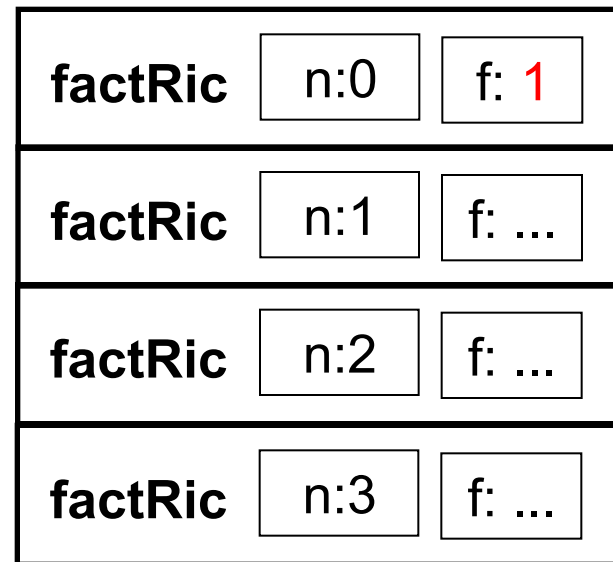
Funzionamento ricorsione

```
function [f]=factRic(n)
    if (n==0)
        f=1;
    else
        f=n*factRic(n-1);
    end
```



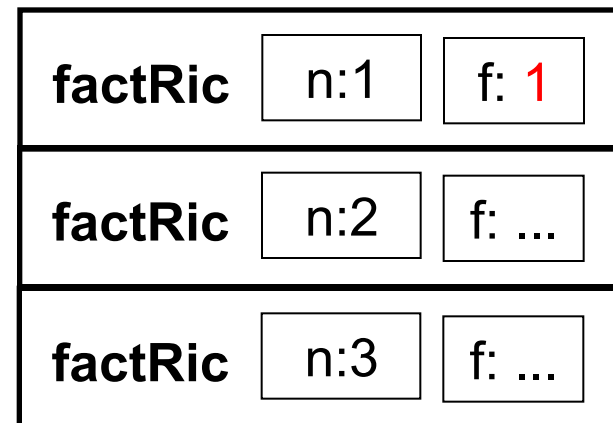
Funzionamento ricorsione

```
function [f]=factRic(n)
    if (n==0)
        f=1;
    else
        f=n*factRic(n-1);
    end
```



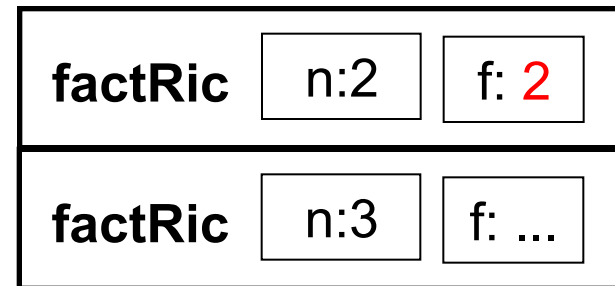
Funzionamento ricorsione

```
function [f]=factRic(n)
    if (n==0)
        f=1;
    else
        f=n*factRic(n-1);
    end
```



Funzionamento ricorsione

```
function [f]=factRic(n)
    if (n==0)
        f=1;
    else
        f=n*factRic(n-1);
    end
```



Funzionamento ricorsione

```
function [f]=factRic(n)
    if (n==0)
        f=1;
    else
        f=n*factRic(n-1);
    end
```

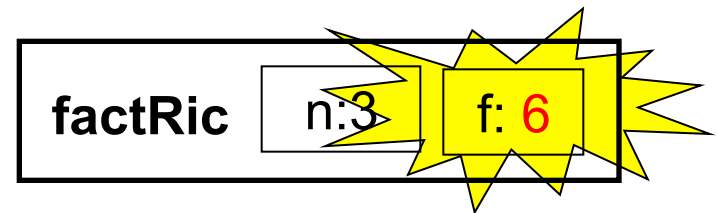
factRic

n:3

f: 6

Funzionamento ricorsione

```
function [f]=factRic(n)
    if (n==0)
        f=1;
    else
        f=n*factRic(n-1);
    end
```



Esempio: Fibonacci

- È una sequenza di numeri interi in cui ogni numero si ottiene sommando i due precedenti nella sequenza. I primi due numeri della sequenza sono per definizione pari ad 1.
 - ▶ $f_1 = 1$ (caso base)
 - ▶ $f_2 = 1$ (caso base)
 - ▶ $f_n = f_{n-1} + f_{n-2}$ (passo ricorsivo)

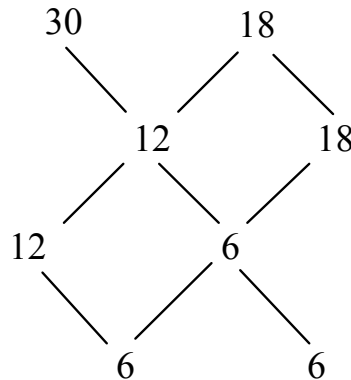
```
function [fib]=FiboRic(n)
    if n==1 | n==2
        fib=1;
    else
        fib=FiboRic(n-2)+FiboRic(n-1);
    end
```

Esempio: MCD

□ Algoritmo di Euclide

- ▶ se $m = n$, $\text{MCD}(m,n) = m$ (caso base)
- ▶ se $m > n$, $\text{MCD}(m,n) = \text{MCD}(m-n,n)$ (caso risorsivo)
- ▶ se $m < n$, $\text{MCD}(m,n) = \text{MCD}(m,n-m)$ (caso risorsivo)

□ Esempio: $\text{MCD}(30,18)$



❑ Algoritmo di Euclide

- ▶ se $m = n$, $\text{MCD}(m,n) = m$ (caso base)
- ▶ se $m > n$, $\text{MCD}(m,n) = \text{MCD}(m-n,n)$ (caso risorsivo)
- ▶ se $m < n$, $\text{MCD}(m,n) = \text{MCD}(m,n-m)$ (caso risorsivo)

❑ Implementazione

```
function [M]=MCDeuclidRic(m,n)
    if m==n
        M=m;
    else
        if m>n
            M = MCDeuclidRic(m-n,n);
        else
            M = MCDeuclidRic(m,n-m);
        end
    end
end
```

- ❑ Terminazione della catena ricorsiva
 - ▶ È presente il caso base?
 - ▶ Viene raggiunto sempre dalla catena di chiamate ricorsive?
 - ▶ Esempi

```
function [f]=factRic(n)
    f=n*factRic(n-1);
```

Catena infinita di chiamate
con argomento decrescente

```
function [f]=factRic(n)
    ...factRic(n);
```

Catena infinita di chiamate
identiche

□ Uso della memoria

- ▶ La programmazione ricorsiva comporta spesso un uso inefficiente della memoria per la gestione degli spazi di lavoro delle chiamate generate
- ▶ In alcuni casi viene comunque preferita ad altri approcci per la sua eleganza e semplicità
- ▶ In altri casi, si può ricorrere ad implementazioni iterative
- ▶ Esempio

```
function [f1]=Fiblist(n)
    f1(1)=1;
    f1(2)=1;
    for k=3:n
        f1(k)=f1(k-2)+f1(k-1);
    end
```

Funzione iterativa che calcola i primi n numeri di fibonacci

Variabili funzione

- ❑ Matlab permette di assegnare a variabili valori di tipo "funzione"
- ❑ Un valore di tipo funzione può essere assegnato a una variabile (quindi passarlo come parametro), detta **handle**
- ❑ L'handle può essere applicato a opportuni argomenti per ottenere una invocazione della funzione

Assegnamento di un valore di tipo funzione

- Handle di una funzione esistente

```
f = @nome_funzione
```

- ▶ Esempio

```
>> seno=@sin  
seno = @sin  
>> seno(pi/2)  
ans = 1
```

- Handle di una funzione definita ex-novo

```
f = @(x,y...)<expr>
```

- ▶ x, y, \dots sono i parametri della funzione
- ▶ $\langle \text{expr} \rangle$ è un'espressione che calcola il valore della funzione
- ▶ Esempio

```
>> sq=@(x) x^2  
sq = @(x) x^2  
>> sq(8)  
ans = 64
```

Funzioni di ordine superiore

- ❑ Se un parametro di una funzione f è un handle (cioè contiene un valore di tipo funzione) allora f è una **funzione di ordine superiore**
- ❑ L'handle passato come parametro consente ad f di invocare la funzione passata
- ❑ Esempio: funzione `map` che applica una funzione f a tutti gli elementi contenuti nel parametro `vin` e ritorna i risultati in `vout`

```
function [vout]=map(f, vin)
    for i=1:length(vin)
        vout(i)=f(vin(i));
    end;
```

handle

```
>> A=[1,2,3,4,5,6];
>> map(sq,A)
ans = 1 4 9 16 25 36
```

Invoca la funzione passata come argomento

Esempio: funzione accumulatore

- ❑ Funzione accumulatore: `[x] = acc(f, a, u)`
 - ▶ applica cumulativamente l'operazione binaria f (con elemento neutro u) a tutti gli elementi di a :

$f(\dots f(f(f(u, a(1)), a(2)), a(3)) \dots, a(\text{length}(a)))$

```
function [x]=acc(f, a, u)
    x=u;
    for i=1:length(a)
        x=f(x, a(i));
    end
```

- ❑ Funzione sommatoria: `function [s]=sommatoria(a)`
 - ▶ calcola la sommatoria degli elementi di a

```
function [s]=sommatoria(a)
    som=@(x,y)x+y;
    s=acc(som, a, 0);
```