Ricorsione e Function Handles

Informatica B AA 17/18

Giacomo Boracchi

giacomo.boracchi@polimi.it

6 Dicembre 2017

Comunicazione di servizio

Vi chiedo gentilmente di compilare il questionario del corso. E' indispensabile per individuare le maggiori criticità e migliorare l'offerta nelle prossime edizioni.

Il questionario integra quello di ateneo, con domande specifiche riguardanti modalità di insegnamento e gli argomenti trattati.

Il link è nella pagina del corso e su:

https://tinyurl.com/y7l9aye9

Grazie!

Funzioni

Richiami

Un esempio

```
function f = fattoriale(n)
f = 1;
for ii = 2 : n
    f = f * ii;
end
```

Un esempio

Parametro formale in uscita

```
function f = fattoriale(n)
f = 1;
for ii = 2 : n
    f = f * ii;
end
```

Parametro formale in ingresso

Definizione di $n!$

Il fattoriale di un intero $n > 0$ è definito come segue:

$$n! = n * (n-1) * (n-2) * \cdots * 2 * 1$$

Un esempio di chiamata

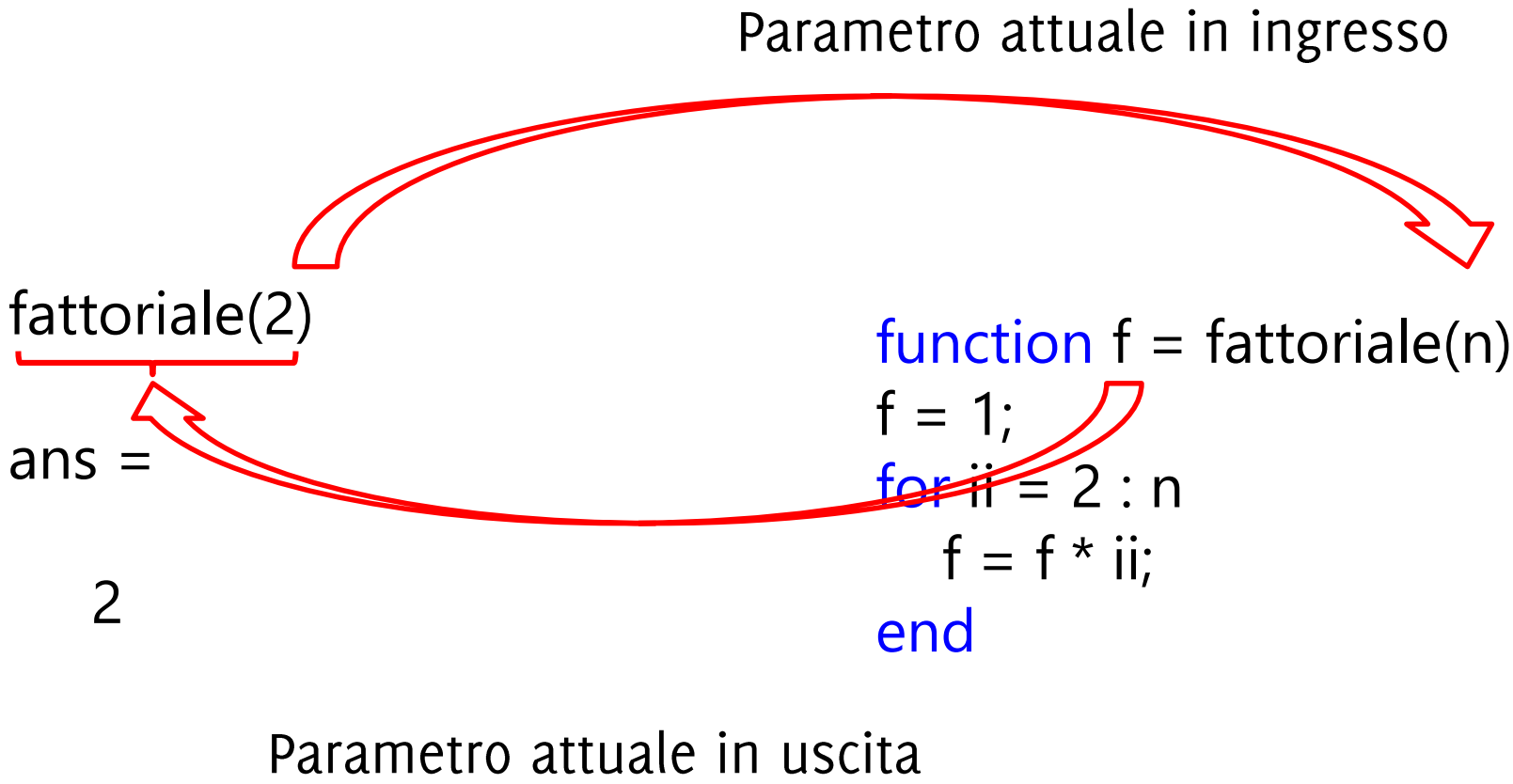
```
fattoriale(2)

ans =

     2
```

```
function f = fattoriale(n)
f = 1;
for ii = 2 : n
    f = f * ii;
end
```

Un esempio di chiamata



Un esempio di chiamata

```
k = 2;
f2 = fattoriale(k);
```

Workspace principale

- Da qui si invoca la funzione
- Contiene le variabili k, f2
- Non ha visibilità di f,ii,n

Un esempio di chiamata

```
k = 2;
f2 = fattoriale(k);
```

Workspace principale

- Da qui si invoca la funzione
- Contiene le variabili k, f2
- Non ha visibilità di f,ii,n

```
function f = fattoriale(n)
f = 1;
for ii = 2 : n
    f = f * ii;
end
```

Workspace locale

- Creato al momento dell'invocazione di fattoriale
- Contiene le variabili n, f, ii
- Non ha visibilità su k ed f2
- Non ha legami con il workspace principale se non per i parametri attuali che vengono copiati (sia in ingresso che in uscita)
- Distrutto terminata l'esecuzione

Un esempio

f2 = fattoriale(2)

f2 =

2

f3 = fattoriale(3)

f3 =

6

f4 = fattoriale(4)

f4 =

24

f5 = fattoriale(5)

f5 =

120

f6 = fattoriale(6)

f6 =

720

Un esempio

- Vale la seguente relazione

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

si dimostra immediatamente dalla definizione di fattoriale

$$n! = n * \underbrace{(n-1) * (n-2) * \cdots * 2 * 1}_{(n-1)!}$$

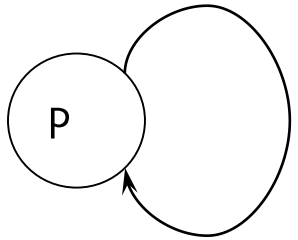
- È possibile usare questa proprietà per definire un'implementazione di fattoriale totalmente diversa?

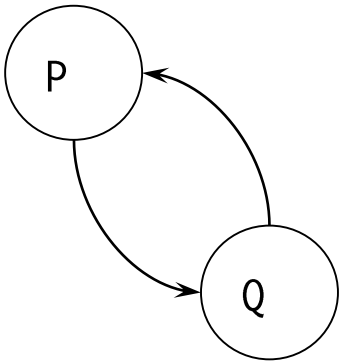
Ricorsione

Programmi che chiamano se stessi

Ricorsione

- Che cos'è la ricorsione?
  - Un sottoprogramma P richiama se stesso (ricorsione diretta)
  - Un sottoprogramma P richiama un sottoprogramma Q che comporta un'altra chiamata a P (ricorsione indiretta)





- È una tecnica di programmazione molto potente
- Permette di risolvere in maniera elegante problemi complessi
- Le funzioni che richiamano se stesse (direttamente o indirettamente) sono dette **funzioni ricorsive**

Esempio: il fattoriale Ricorsivo

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

Esempio: il fattoriale Ricorsivo

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

Questa ci ricorda
$0! = 1$

Esempio: il fattoriale Ricorsivo

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

Questa ci ricorda
$n! = n * (n-1)!$

Una funzione che chiama se stessa

Una Funzione che Chiama se Stessa?

In ogni istante possono essere in corso **diverse attivazioni** dello **stesso** sottoprogramma

- Ovviamente sono **tutte sospese tranne** una, **l'ultima invocata**, all'interno della quale si sta svolgendo il **flusso di esecuzione**.

Una Funzione che Chiama se Stessa?

In ogni istante possono essere in corso **diverse attivazioni** dello **stesso** sottoprogramma

- Ovviamente sono **tutte sospese tranne** una, **l'ultima invocata**, all'interno della quale si sta svolgendo il **flusso di esecuzione**.

Ogni attivazione esegue **lo stesso codice** ma opera su **workspace distinti** (in Matlab, ogni funzione attivata ha un workspace distinto)

- Si hanno quindi **copie distinte** dei **parametri attuali** e delle **variabili locali** nelle varie invocazioni

Una funzione che chiama se stessa?

… se ogni volta la funzione richiama se stessa… *perché la catena di invocazioni non continua* ***all'infinito***?

Una funzione che chiama se stessa?

… se ogni volta la funzione richiama se stessa… *perché la catena di invocazioni non continua* ***all'infinito****?*

La funzione ricorsiva deve prevedere una situazione in cui non richiama se stessa, i.e., il **caso base**

Programmazione Ricorsiva

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:

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

Definire caso base, e passo ricorsivo

Esempio: il fattoriale

Definizione:

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

Passo ricorsivo:

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

Caso base:

$$f(0) = 1$$

Esempio: il fattoriale

Definizione:

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

Passo ricorsivo:

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

Caso base:

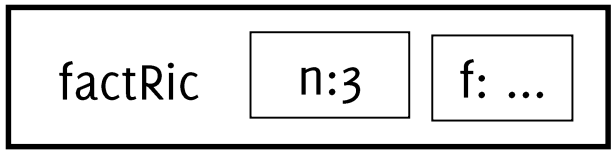
$$f(0) = 1$$

```
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
```

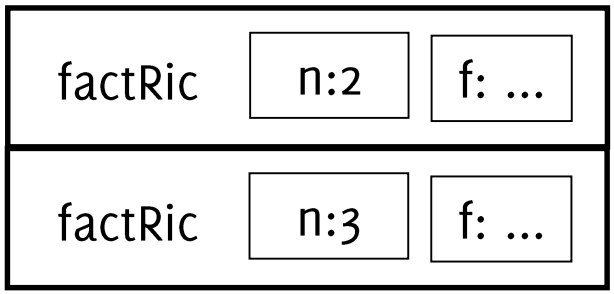
factRic(3)



Funzionamento ricorsione

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

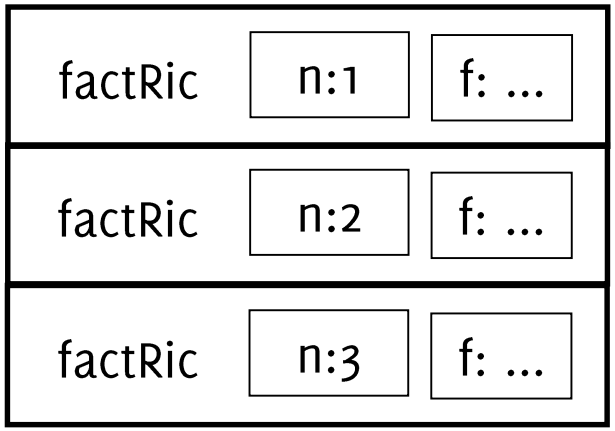
factRic(3)



Funzionamento ricorsione

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

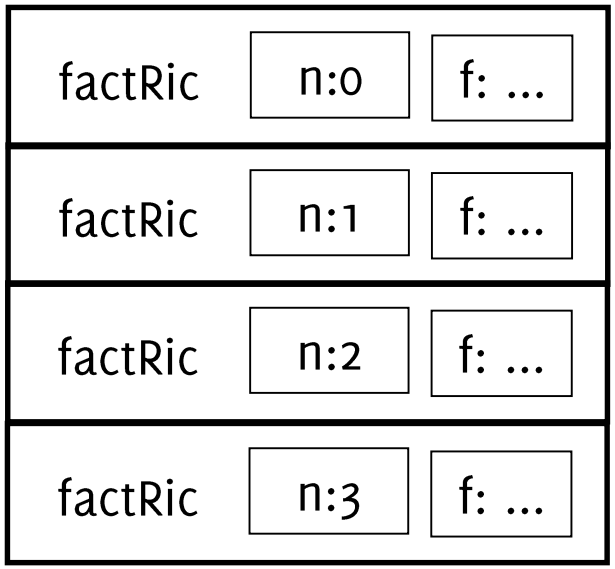
factRic(3)



Funzionamento ricorsione

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

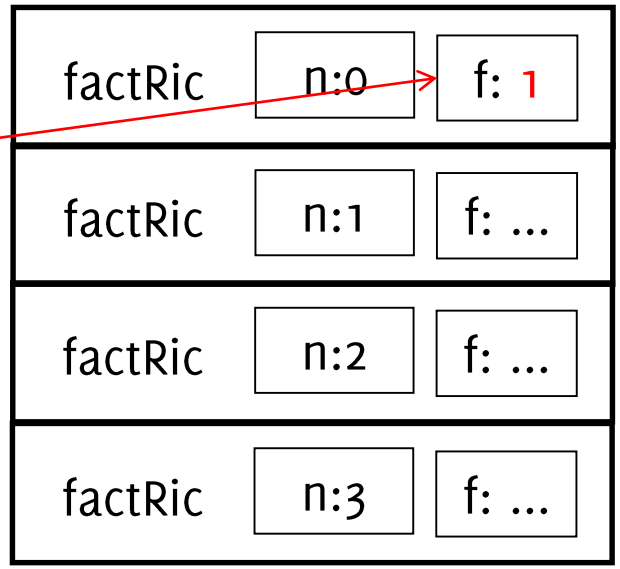
factRic(3)



Funzionamento ricorsione

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

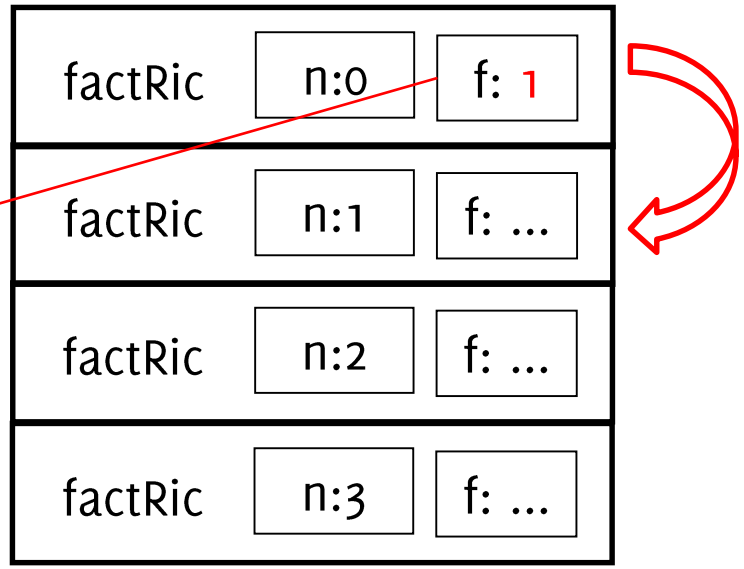
factRic(3)



Funzionamento ricorsione

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

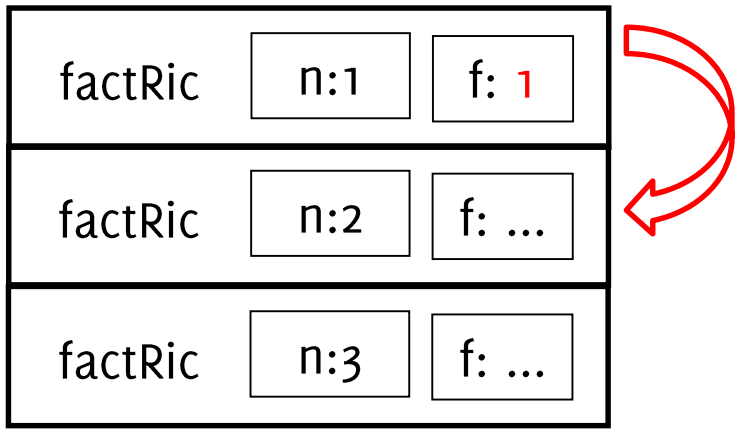
factRic(3)



Funzionamento ricorsione

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

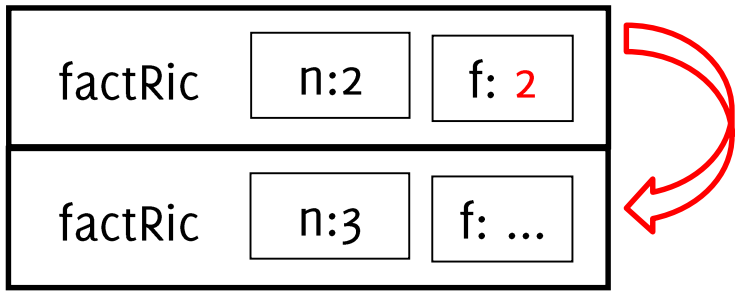
factRic(3)



Funzionamento ricorsione

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

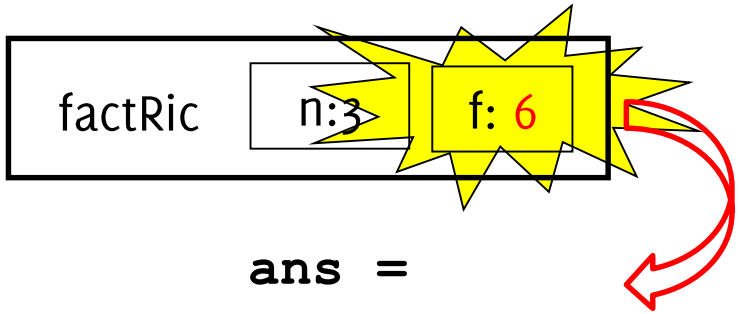
factRic(3)



Funzionamento ricorsione

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

factRic(3)



ans =

6

Ricorsione in Coda

Ricorsione in cui la funzione:

- prevede **una sola chiamata ricorsiva**
- esegue la chiamata ricorsiva come **ultima istruzione**

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

Problemi nell'Uso della Ricorsione

Terminazione della catena ricorsiva

- È presente il caso base?
- Viene raggiunto sempre dalla catena di chiamate ricorsive?

Errori nell'Uso della Ricorsione

Catena infinita di chiamate incondizionate

- **Deve esistere** una **condizione** tale per cui **non** viene eseguita la **chiamata ricorsiva** (**caso base**)

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

Es la chiamata a factRic(7) chiama factRic(7),
che chiama factRic(6),
che chiama factRic(5),
che chiama factRic(4),
che chiama factRic(3),….

che chiama factRic(-1000),….

Errori nell'Uso della Ricorsione

Catena infinita di chiamate incondizionate

- Attenzione che **è necessario che questa condizione venga raggiunta**: anche programmi formalmente corretti potrebbero non funzionare per alcuni valori degli ingressi

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

- Ad es, factRic(-1) da luogo ad una sequenza di chiamate infinite

Errori nell'Uso della Ricorsione

Catena infinita di chiamate identiche:

- La chiamata ricorsiva **non** può avere i **parametri formali uguali a quelli attuali**

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

Es la chiamata a factRic(7) chiama factRic(7),
che chiama factRic(7),
che chiama factRic(7),
che chiama factRic(7),….

Di Default…

Matlab si blocca le chiamate ricorsive dopo un po'

>> factRic(600)

Out of memory. The likely cause is an infinite recursion within the program.

Error in **fatRic** at line XXX

dove XXX indica la riga in cui compare la chiamata ricorsiva

Condizioni Necessarie

.. Per evitare ricorsione infinita:

Occorre che le **chiamate ricorsive** siano **soggette** a una **condizione** che prima o poi assicura che la catena termini

Occorre anche che **l'argomento sia *progressivamente ridotto*** dal passo induttivo, in modo da tendere prima o poi al caso base

- Nella pratica l'argomento si avvicina al valore nel caso base

Esempio

Scrivere una funzione ricorsiva che calcola la somma di tutti gli interi compresi tra i due argomenti

Esempio

Scrivere una funzione ricorsiva che calcola la somma di tutti gli interi compresi tra i due argomenti

```
function somma = sommaNumeriCompresi(a ,b)

if a == b
    somma = a;
    disp(['caso base somma vale ' , num2str(somma)]);

else
    disp(['prima chiamata a = ' ,num2str(a)]);
    somma = a + sommaNumeriCompresi(a+1 , b);
    disp(['dopo chiamata a = ' ,num2str(a)]);
end
```

Esempio

Scrivere una funzione ricorsiva che calcola la somma di tutti gli interi compresi tra i due argomenti

```
function somma = sommaNumeriCompresi(a ,b)

if b< a
    somma = 0;
end
if a == b
    somma = a;
    disp(['caso base somma vale ' , num2str(somma)]);
else
    disp(['prima chiamata a = ' ,num2str(a)]);
    somma = a + sommaNumeriCompresi(a+1 , b);
    disp(['dopo chiamata a = ' ,num2str(a)]);
end
```

Errore! Non è un caso base perchè dopo di questo viene comunque eseguita la chiamata ricorsiva

Esempio

Scrivere una funzione ricorsiva che calcola la somma di tutti gli interi compresi tra i due argomenti

```
function somma = sommaNumeriCompresi(a ,b)

if b< a
    somma = 0;
    return;
end
if a == b
    somma = a;
    disp(['caso base somma vale ' , num2str(somma)]);
else
    disp(['prima chiamata a = ' ,num2str(a)]);
    somma = a + sommaNumeriCompresi(a+1 , b);
    disp(['dopo chiamata a = ' ,num2str(a)]);
end
```

In questo moto l'esecuzione termina e abbiamo un caso base corretto. Si sarebbe ottenuto lo stesso risultato annidando gli if o utilizzando variabili di flag

Esempio

Scrivere una funzione ricorsiva che calcola la somma di tutti gli interi compresi tra i due argomenti

```
function somma=calcolaSommaCompresi2(a_temp,b_temp)
a=min([a_temp,b_temp]);
b=max([a_temp,b_temp]);
if(a==b)
    disp(['caso base 1 a=',num2str(a),'b=',num2str(b)])
    somma=a;
else
    if(b-a==1)
        disp(['caso base 2 a=',num2str(a),'b=',num2str(b)])
        somma=a+b;
    else
        disp(['prima della chiamata ricorsiva a=',num2str(a),'b=',num2str(b)])
        somma=a+calcolaSommaCompresi2(a+1,b-1)+b;
        disp(['dopo la chiamata ricorsiva a=',num2str(a),'b=',num2str(b),'
somma =',num2str(somma)])
    end
end
```

Inverte le variabili per calcolare comunque i numeri compresi anche se l'ordine non è corretto

Esempio

Scrivere una funzione ricorsiva che calcola la somma di tutti gli interi compresi tra i due argomenti

```
function somma = sommaCompresi(a, b)
% calcolare la somma degli interi tra a e b
```

```
if(b < a)
    somma = 0;
    return
end
```

In questo caso restituisce o se $b < a$ e non serve un caso base per il caso $b - a == 1$

```
if (a == b)
    somma = a;
else
    somma = a + sommaCompresi(a + 1, b - 1) + b;
end
```

Le ninfee

- Uno stagno è pieno di ninfee
  - Ogni ninfea in 1 giorno si riproduce, generando un'altra ninfea
  - Dopo 30 gg lo stagno è pieno
- Quanto hanno impiegato le ninfee a riempire a metà stagno?

- Scrivere una funzione ricorsiva per modellare
  - una popolazione iniziale di *n* ninfee,
  - che si riproduce ad un fattore *f* ogni giorno,

  e per dire quanti giorni richiede per raggiungere una popolazione di *N* individui

Soluzione

```
function gg = stagno(n, f, N)
 % caso base: se ho già N ninfee nello stagno
% non devo aspettare alcun giorno
if (n >= N)
    gg = 0;
else
    % non abbiamo ancora N ninfee -> chiamata ricorsiva:
    % il numero di giorni necessari per coprire lo stagno è
    % uguale a: un giorno +
    % il numero di giorni che saranno necessari domani
    % (quando le ninfee saranno diventate n*f).
     gg = 1 + stagno(n * f, f, N);
end
```

```
function [giorni, vettore_n] = ninfee(n, f, N)

% caso base
if (n >= N)
    giorni = 0;
    vettore_n = [];
    fprintf('\n caso base oggi ho %2.2f ninfee', n);
else
    nDomani = round(n * f);
    if nDomani == n
        giorni = nan;
        fprintf('\ncrescita o popolazione iniziale insufficiente');
        return
    end
    % chiamata ricorsiva
    fprintf('\n oggi ho %2.2f ninfee', n);
    % la popolazione aumenta
    [giorni, vettore_n] = ninfee(nDomani, f, N);
    giorni = giorni + 1;
    vettore_n = [nDomani, vettore_n];
end
```

Esempio

Scrivere una funzione ricorsiva per calcolare la lunghezza di una stringa

Esempio

Scrivere una funzione ricorsiva per calcolare la lunghezza di una stringa

```
function s = calcolaLunghezza(str)
if(isempty(str))
    s = 0;
else
    s = 1 + calcolaLunghezza(str(2 : end));
end
```

Esempio:

Scrivere una funzione per stampare una stringa al contrario

Esempio:

Scrivere una funzione per stampare una stringa al contrario

```
function stampaAlContrario(frase)

% caso base
if isempty(frase)
    % return
else
    % chiamata ricorsiva
    disp(frase(end)); % stampa al contrario
    stampaAlContrario(frase(1:end-1))
end
```

Esempio:

Modificare la funzione per stampare la stringa nello stesso ordine in cui viene inserita

Esempio:

Modificare la funzione per stampare la stringa nello stesso ordine in cui viene inserita

```
function stampaAlContrario(frase)

% caso base
if isempty(frase)
    % return
else
    % chiamata ricorsiva
     stampaAlContrario(frase(1:end-1))
     disp(frase(end)); % stampa dritto
end
```

In questo caso la ricorsione non è in coda

Esempio:

Cosa stampa??

```
function stampa(frase)

% caso base
if isempty(frase)
    % return
else
        stampa(frase(2:end))
        disp(frase(1));
end
```

Esempio:

Cosa stampa??

```
function stampa(vettore)
if (length(vettore)==1)
    % caso base
    fprintf('%c',vettore(1));
    fprintf('%c',vettore(1));
else
    fprintf('%c',vettore(1));
    stampa(vettore(2:end));
    fprintf('%c',vettore(1));
end
```

Cosa Stampa?

```
function stampaAlContrario(str)

if(isempty(str))
% return
else
    disp(str(end));
    stampaAlContrario(str(1 : end -1));
    disp(str(end));
end
```

Cosa Stampa?

```
function stampaAlContrario(str)

if(isempty(str))
% return
else
    disp(str(end));
    stampaAlContrario(str(1 : end -1));
    disp(str(end));
end
```

```
>> stampaAlContrario('ciao')
    o
    a
    i
    c
    c
    i
    a
    o
```

Cosa Stampa?

```
function stampaAlContrario(str)

if(isempty(str))
% return
else
    disp(str(end));
    stampaAlContrario(str(2 : end));
    disp(str(end));
end
```

Cosa Stampa?

```
function stampaAlContrario(str)

if(isempty(str))
% return
else
    disp(str(end));
    stampaAlContrario(str(2 : end));
    disp(str(end));
end
```

```
›› stampaAlContrario('ciao')
     o
     o
     o
     o
     o
     o
     o
     o
```

Cosa Stampa?

```
function stampaAlContrario(str)

if(isempty(str))
% return
else
    disp(str(1));
    stampaAlContrario(str(2 : end));
    disp(str(1));
end
```

Cosa Stampa?

```
function stampaAlContrario(str)

if(isempty(str))
% return
else
    disp(str(1));
    stampaAlContrario(str(2 : end));
    disp(str(1));
end
```

```
›› stampaAlContrario('ciao')
    c
    i
    a
    o
    o
    a
    i
    c
```

Cosa Stampa?

```
function stampaAlContrario(str)

if(isempty(str))
% return
else
    disp(str(1));
    stampaAlContrario(str(1 : end -1));
    disp(str(1));
end
```

Cosa Stampa?

```
function stampaAlContrario(str)

if(isempty(str))
% return
else
    disp(str(1));
    stampaAlContrario(str(1 : end -1));
    disp(str(1));
end
```

```
>> stampaAlContrario('ciao')
     c
     c
     c
     c
     c
     c
     c
     c
```

Cosa Fa?

```
function vet = chefa(vet)
% dimmi che fa…

if isempty(vet) || length(vet) == 1
    return
end
if all(vet(1) >= vet(2 : end)) % punto 1) del codice
    return
end
vet(vet == vet(1)) = [];
vet = chefa(vet);

>> chefa([19 18 12 19 12 3])

>> chefa([7 18 13 19 12 3])

>> chefa([ 2 3 18 12 19 12 3])
```

Cosa Fa?

```
>> chefa([ 19 18 12 19 12 3])

ans =

    19    18    12    19    12     3

>> chefa([7 18 13 19 12 3])

ans =

    19    12     3

>> chefa([ 2 3 18 12 19 12 3])

ans =

    19
```

Cosa Fa?

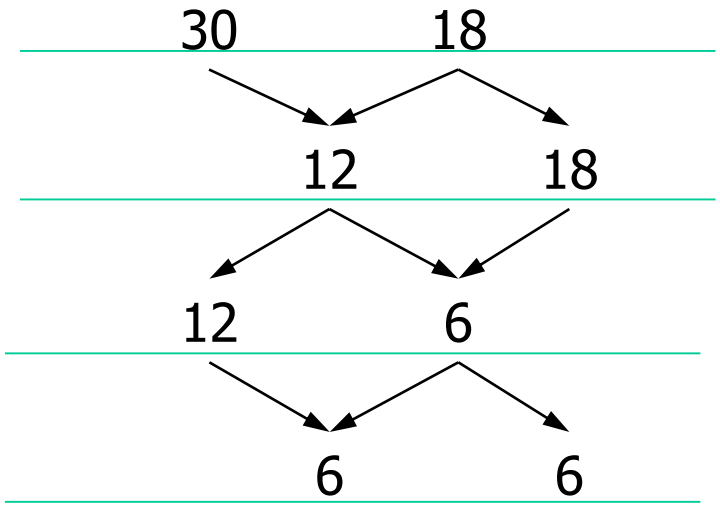
La funzione rimuove restituisce un sottovettore dell'input avente il primo elemento maggiore dei successivi. La funzione ricerca i sottovettori rimuovendo ad ogni invocazione il primo elemento. Inoltre, qualora vi fossero più occorrenze di un elemento diverso dal massimo del vettore, tutte queste vengono rimosse.

Se modificassi la condizione 1) con `all(vet(1) > vet(2 : end))` avremmo che anche il massimo del vettore viene rimosso se questo non è unico.

```
>> chefa([ 19 18 12 19 12 3])
ans =
    18     12     12      3
```

Esempio: MCD

- Algoritmo di Euclide
  - se m = n, MCD(m,n) = m (caso base)
  - se m > n, MCD(m,n) = MCD(m-n,n) (caso risorsivo)
  - se m < n, MCD(m,n) = MCD(m,n-m) (caso risorsivo)
- Esempio: MCD(30,18)



Esempio: MCD

- Algoritmo di Euclide
  - se m = n, MCD(m,n) = m (caso base)
  - se m > n, MCD(m,n) = MCD(m-n,n) (caso risorsivo)
  - se m < n, MCD(m,n) = MCD(m,n-m) (caso risorsivo)

```
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
```

Due possibili chiamate ricorsive, e la chiamata è condizionata

Un altro esempio: la serie di Fibonacci

Fibonacci (1202) partì dallo studio sullo sviluppo di una colonia di conigli in circostanze ideali

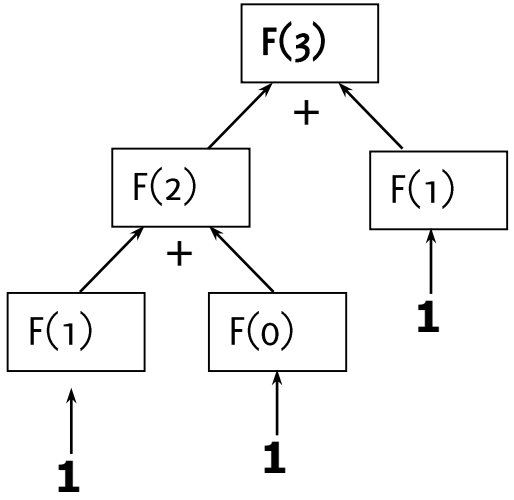
- Partiamo da una coppia di conigli
- I conigli possono riprodursi all'età di un mese
- Supponiamo che dal secondo mese di vita in poi, ogni femmina produca una nuova coppia
- e inoltre che i conigli non muoiano mai…

- Quante coppie ci sono dopo n mesi?

Definizione ricorsiva della serie

I numeri di Fibonacci

- Modello a base di molte dinamiche evolutive delle popolazioni



$F = \{f_0, \ldots, f_n\}$

- $f_0 = 1$
- $f_1 = 1$

} casi base (sono 2!)

- Per $n > 1$, $f_n = f_{n-1} + f_{n-2}$

} 1 passo induttivo

Notazione "funzionale": $F(i) = f_i$

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
```

Problemi nell'uso della ricorsione

Uso della memoria

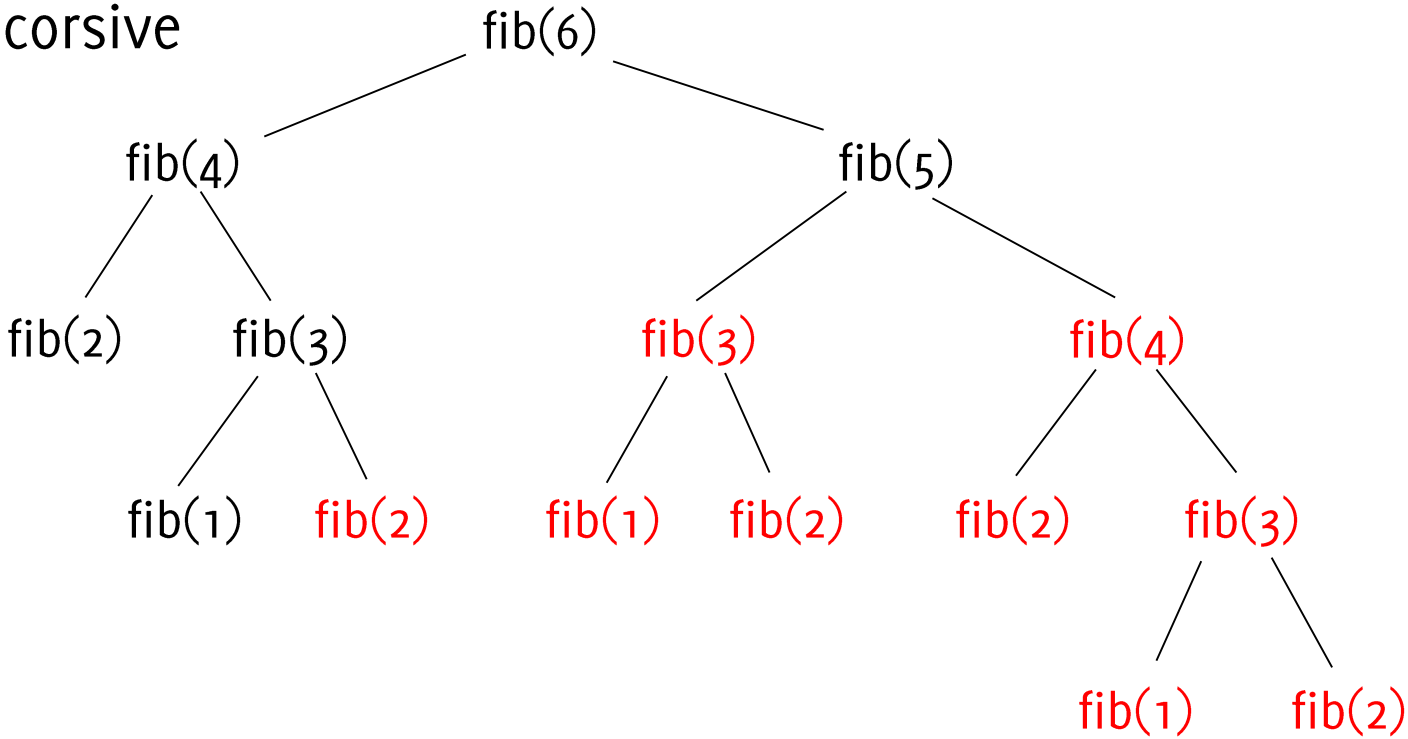
- La programmazione ricorsiva comporta spesso un uso inefficiente della memoria per la gestione degli spazi di lavoro delle chiamate generate
- In alcuni casi viene comunque preferita ad altri approcci per la sua eleganza e semplicità
- In altri casi, si può ricorrere ad implementazioni iterative
- Esempio

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

Funzione iterativa che calcola i primi n numeri di fibonacci

Ricorsione Eccessiva

Soluzione elegante ma dispendiosa: numero esorbitante di chiamate ricorsive



Provate a far calcolare fib(5), poi fib(10), fib(15), fib(20), fib(25), fib(30), ...

Qante volte viene calcolato fab(3)????

Meglio usare una soluzione non ricorsiva...

TODO

Scrivere una funzione ricorsiva in Matlab che stampi a schermo tutti i valori del triangolo di Tartaglia per un certo ordine massimo N. i.e., e che restituisca al chiamante la riga N-sima

1	n = 0
1 1	n = 1
1 2 1	n = 2
1 3 3 1	n = 3
1 4 6 4 1	n = 4
1 5 10 10 5 1	n = 5
1 6 15 20 15 6 1	n = 6

Altri Esempi

Esempio

Scrivere una funzione ricorsiva *palindroma* che controlla se una stringa è palindroma

Ricorsione:

Scrivere una **funzione ricorsiva** che prende in ingresso due numeri n e d che determina se n è una potenza di d

Interpretazione del codice

```
function r=cosafa(array)
k=size(array,2);

if (k == 1)
  r = 1;
elseif (k==2)
  if (array(1)+array(2)==10)
      r = 1;
  else
      r=0;
  end
else
  if (array(1)+array(k)==10)
      r = cosafa(array(2:k-1));
  else
      r=0;
  end
end
```

TDE Febbraio 2012

- Si consideri la seguente funzione in Matlab

```
function [ris] = mistero(v, n)
if (n > 1)
    v2 = v(mod(v, n) ~= 0 | v == n);
    ris = mistero(v2, n-1);
else
    ris = v;
end
```

1. Dire qual è il contenuto di x dopo la seguente chiamata

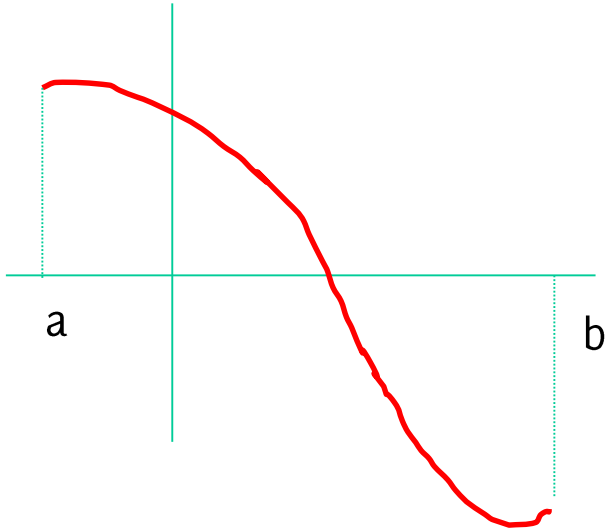
```
x = mistero([2 3 4 5 7 9 11 13 15], 10)
```

Esempio dal teorema di Bolzano

Sia f una funzione reale e continua in $[a, b]$ per cui

$$f(a) * f(b) \leq 0$$

allora estiste almeno un punto $c \in [a, b]$ tale che $f(c) = 0$.



Usare questa teorema per scrivere una funzione haUnoZero che prende in ingresso un function handle ad f, gli estremi dell'intervallo e determina se la funzione ha uno zero nell'intervallo

TODO:

- Utilizzando il metodo di bisezione e il teorema di Bolzano, scrivere una funzione **ricorsiva e di ordine superiore** **`calcolaZeri`** che calcola gli zeri di una funzione analitica f (passata in un function handle) nell' intervallo continuo $[a, b]$ (gli estremi sono passati come parametri)
- Si consideri come criterio di arresto invece di $f(x) == 0$

  $|f(x)| < \epsilon$ dove ϵ viene passato come parametro.