

Unità didattica: Programmazione iterativa vs
programmazione ricorsiva

[1-T]

Titolo: Organizzazione dei programmi ricorsivi

Argomenti trattati:

- ✓ Classificazione degli algoritmi ricorsivi
- ✓ Come viene realizzata la ricorsione dal compilatore
- ✓ Esempi di function C ricorsive

Prerequisiti richiesti: function C, struttura dati pila

Richiamo della definizione

Un algoritmo o un programma diconsi ricorsivi se richiamano direttamente o indirettamente sé stessi.

Qualsiasi algoritmo espresso tramite costrutti:

- **sequenza,**
- **di selezione**
- **ripetitivi**

può essere scritto in forma ricorsiva.

Esempio: algoritmo per il Massimo Comun Divisore

(Algoritmo di Euclide)

$$\begin{aligned} \text{MCD}(m, 0) &= m \\ \text{MCD}(m, n) &= \text{MCD}(m-n, n) \end{aligned}$$

definizione ricorsiva

$$\text{MCD}(m, n) = \begin{cases} m & \text{se } n=0 \\ \text{MCD}(n, m \bmod n) & \text{se } n>0 \end{cases}$$

$$\begin{aligned} \text{MCD}(14, 8) &= \text{MCD}(8, 6) && (\text{perchè } n=8 > 0) = \\ &= \text{MCD}(6, 2) && (\text{perchè } n=6 > 0) = \\ &= \text{MCD}(2, 0) && (\text{perchè } n=2 > 0) = \\ &= 2 && (\text{perchè } n = 0) \end{aligned}$$

Attenzione !!!

```
procedure recurs(...)
```

```
begin
```

-

-

-

- **chiamata ricorsiva a recurs(...)**

-

-

-

```
end
```

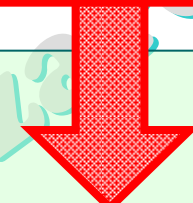
*ricorsione
senza fine*

Una procedura ricorsiva deve sempre contenere una **computazione diretta** (relativa ad un **caso elementare**) per interrompere la nidificazione delle chiamate.

Struttura della Ricorsione

struttura generale

computazione diretta



```
if (caso elementare) then  
    risolvi il problema direttamente e ritorna
```

```
else
```

```
begin
```

```
    decomponi il problema P in sottoproblemi  
    SP1, SP2, ..., SPn
```

```
    chiamate ricorsive a recurs(SP1),  
    recurs(SP2),
```

```
    ricomponi la soluzione
```

```
end
```



ricorsione

Esempio

function ricorsiva C per il fattoriale di n

$$n! = \begin{cases} 1 & \text{se } n = 0 \\ n \cdot (n-1)! & \text{se } n \geq 1 \end{cases}$$

```
float recurs_fact(short n)
{float nfatt;
if (n <= 1) nfatt=1;
else nfatt=n*recurs_fact(n-1);
return nfatt;
}
```

Caso banale: serve per terminare il processo ricorsivo altrimenti senza fine!!!

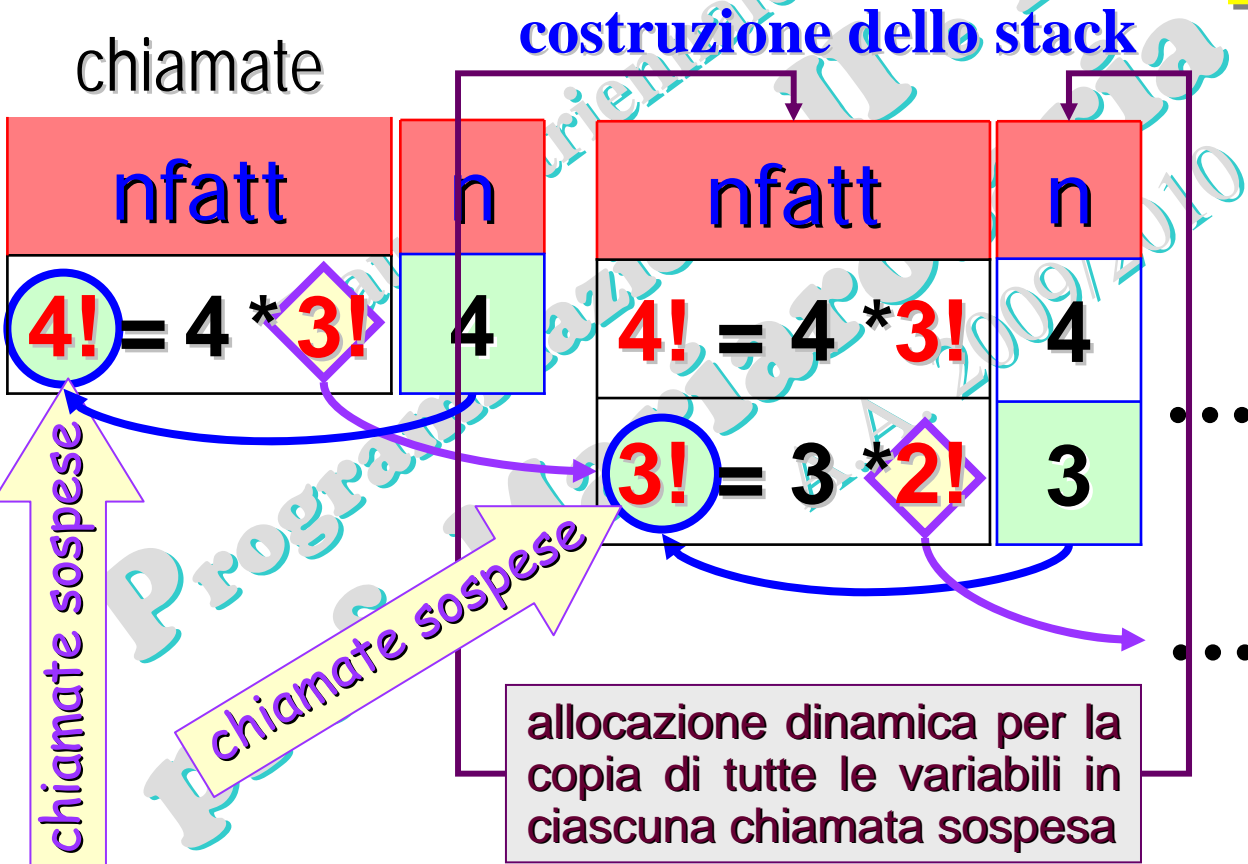
```
float recurs_fact(short n)
{if (n <= 1) return 1;
else return n*recurs_fact(n-1);
}
```

Come viene realizzata dal compilatore la ricorsione?

Attraverso uno stack e la gestione dinamica della memoria

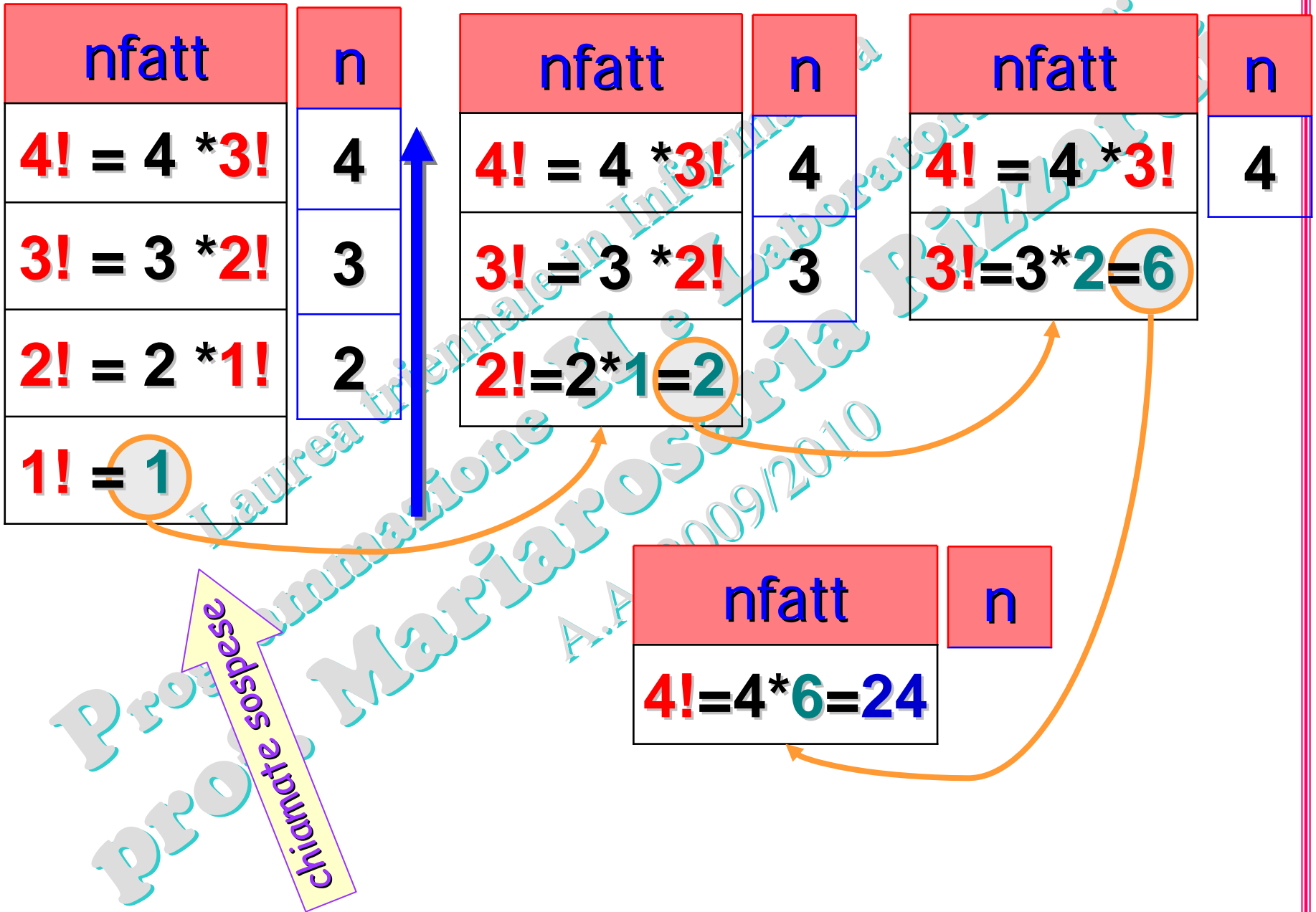
Esempio: fattoriale di $n=4$

```
float recurs_fact(short n){  
    float nfatt;  
    if (n <= 1) nfatt=1;  
    else nfatt=n*recurs_fact(n-1);  
    return nfatt;  
}
```



nfatt	n
$4! = 4 * 3!$	4
$3! = 3 * 2!$	3
$2! = 2 * 1!$	2
$1! = 1$	1

uso stack



function C che simula la ricorsione tramite uno stack

```
float s_rekurs_fact(short n)
{float nfatt; short i,len_stack,*p_stack;
  len_stack=n; nfatt=1;
  if (n > 0)
    {p_stack = calloc(len_stack, sizeof(short));

    for (i=0; i<len_stack; i++)
      {push(p_stack,i,n); n--;}

    for (i=len_stack-1; i>=0; i--)
      {n = pop(p_stack,i); nfatt=nfatt*n;}
    }
  return nfatt;
}

void push(short *p_stack, short i, short n)
{ *(p_stack+i)=n; }

short pop(short *p_stack, short i)
{short n;
  n=*(p_stack+i); free(p_stack+i); return n;}
```

manca controllo
allocazione

simula chiamate sospese

Classificazione degli algoritmi ricorsivi

ricorsione

ricorsione diretta

lineare

1 sola chiamata ricorsiva nella procedura.

binaria

2 chiamate ricorsive nella procedura.

non lineare

solitamente, nella procedura, almeno **1** chiamata ricorsiva dentro un ciclo.

ricorsione indiretta

**mutua
ricorsione**

chiamata ricorsiva indiretta tramite un'altra procedura.

Ricorsione lineare

struttura

```
procedure lin_rec(...)  
begin  
    if (condizione di fine) then  
        restituisci i risultati e ritorna  
    else  
        begin  
            compi delle azioni  
            chiamata ricorsiva a lin_rec(...)  
        end  
    end  
end
```

1 sola chiamata ricorsiva
nella procedura.

Esempi di ricorsione lineare in C

Es. 1: $n!$

```
float recurs_fact(short n)
{if  (n <= 1) return 1;
  else return n*recurs_fact(n-1);
}
```

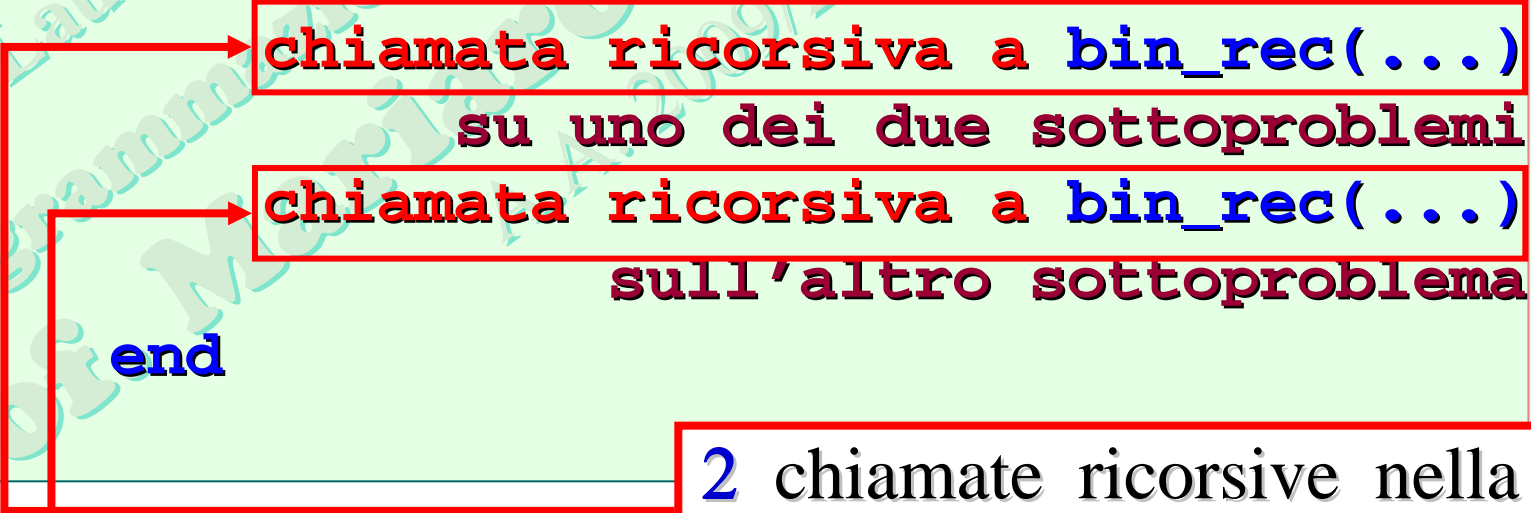
Es. 2: MCD

```
long recurs_MCD(long m, long n)
{if  (n == 0) return m;
  else return recurs_MCD(n, m%n);
}
```

Ricorsione binaria

struttura

```
procedure bin_rec(...)  
begin  
    if (condizione di fine) then  
        restituisci i risultati e ritorna  
    else  
        begin  
            compi delle azioni (suddividi in  
                                due sottoproblemi)  
            chiamata ricorsiva a bin_rec(...)  
            su uno dei due sottoproblemi  
            chiamata ricorsiva a bin_rec(...)  
            sull'altro sottoproblema  
        end  
    end  
end
```



2 chiamate ricorsive nella procedura.

Esempio di ricorsione binaria in C

Es. 1: numeri di Fibonacci

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

Es. 2: massimo in un array

```
char recurs_massimo(char v[],short inizio, short fine)
{short mez; char t1, t2;
 if  (inizio == fine) return v[fine];
 else
 {mez=(inizio+fine)/2;
  t1 = recurs_massimo(v, inizio, mez);
  t2 = recurs_massimo(v, mez+1, fine);
  if (t1<t2)return t2;
  else      return t1;
 }
}
```

Esempio di ricorsione binaria in C

Es. 3: ricerca binaria in un array

```
short ric_bin(char el, char a[], short inizio, short fine)
/* cerca elemento el in array ordinato a[]:
 * valore di ritorno: 1 <=> trovato, 0 <=> non trovato
 */
{short centro;
if (inizio == fine) return (el == a[inizio]);
if (inizio > fine) return 0;
if (inizio < fine)
{centro=(inizio+fine)/2;
if (el == a[centro]) return 1;
else if (el < a[centro])
return ric_bin(el,a,inizio,centro-1);
else
return ric_bin(el,a,centro+1,fine);
}
}
```


Ricorsione non lineare

struttura

```
procedure nonlin_rec(...)  
begin  
    while .not.condizione di fine do  
        begin  
            compi delle azioni  
            ...  
            chiamata ricorsiva a nonlin_rec(...)  
            ...  
            compi delle azioni  
        end  
    end  
end
```

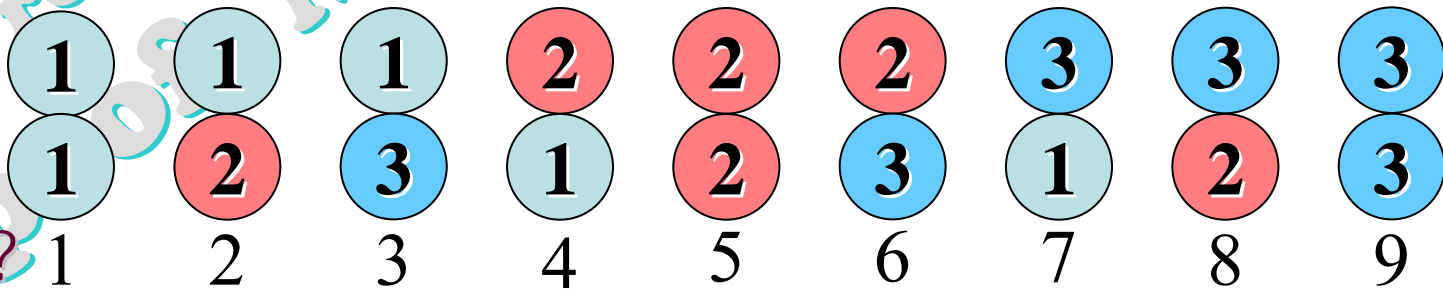
1 chiamata ricorsiva dentro
un ciclo.

Esempio di ricorsione non lineare in C

Problema combinatorio: generare le n^r disposizioni con ripetizioni dei primi n numeri naturali ($n > 0$), presi r ($r > 0$) alla volta.

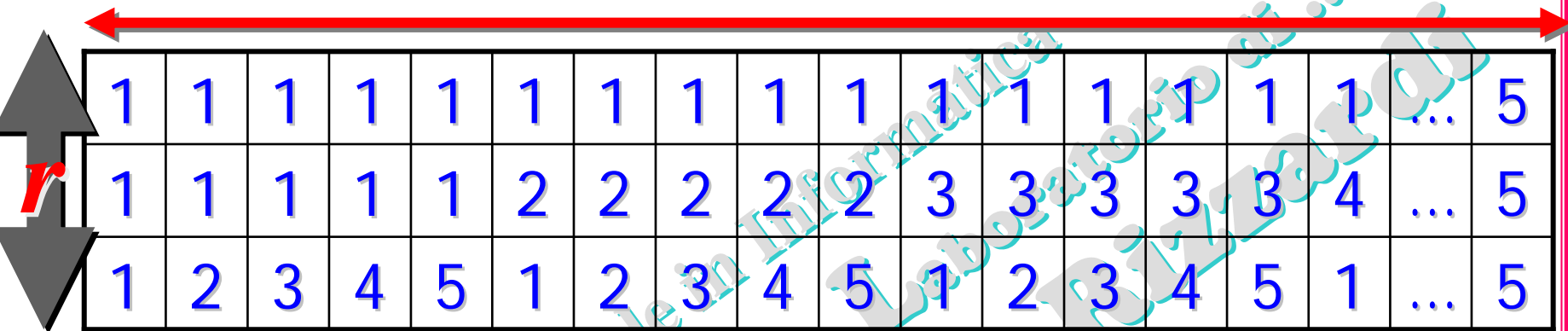
disposizioni con ripetizioni dei primi n numeri naturali ($n > 0$), presi r ($r > 0$) alla volta: tutti i possibili modi di disporre in r -ple ordinate n oggetti, anche ripetuti, solitamente numerati come $1, 2, \dots, n$.

3 oggetti  disposti a **coppie**: 



quante? 1 2 3 4 5 6 7 8 9 $= 3^2$

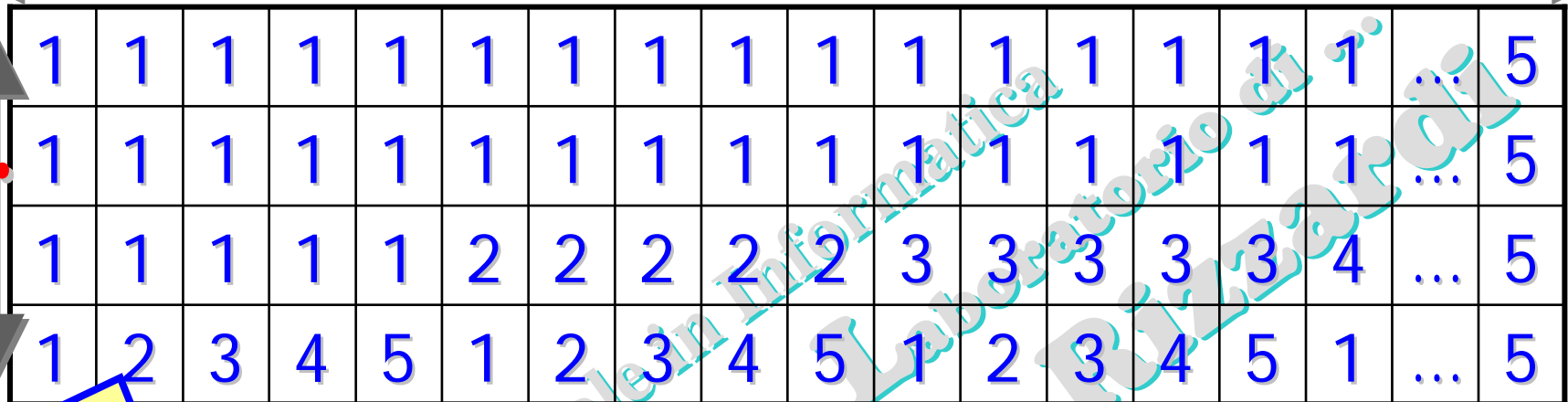
Caso particolare: $n=5$, $r=3$ (soluzioni $n^r=5^3=125$)



1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	...	5	
1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4	...	5
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	...	5

```
void iterative_disposizioni(int n)
/* r fissato r=3 */
{int i,j,h;
  for (i=1; i<=n; i++)
    for (j=1; j<=n; j++)
      for (h=1; h<=n; h++) printf("...",i,j,h);
}
```

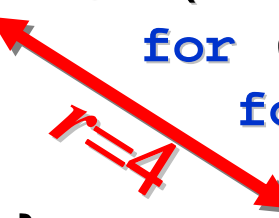
Caso particolare: $n=5$, $r=4$ (soluzioni $n^r=5^4=625$)



1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	...	5
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	...	5
1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4	...	5
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	...	5

soluzioni

```
void iterative_disposizioni(int n)
/* r fissato r=4 */
{int i,j,h,k;
  for (i=1; i<=n; i++)
    for (j=1; j<=n; j++)
      for (h=1; h<=n; h++)
        for (k=1; k<=n; k++) printf("...",i,j,h,k)
}
```



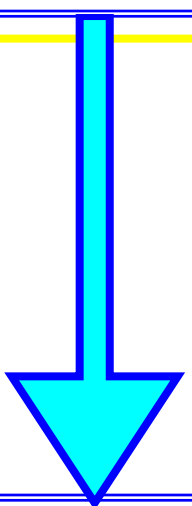
...e per r generico ?

Non esiste linguaggio di programmazione per l'algoritmo iterativo con r cicli nidificati

Il problema può risolversi solo mediante **function ricorsiva**

```
#include <stdio.h>
#include <stdlib.h>
void sample(int , int , int , int []);
void main()
{int n,r,column[20]; /* r ≤ n ≤ 20 */
  n=3;r=2;          /* n oggetti disposti in r-ple */
  sample(n,r,1,column); }

void sample(int n, int r, int k, int column[])
{int j;
  column[k]=0;
  while (column[k]<n)
  {column[k] = column[k]+1;
    if (k<r) sample(n,r,k+1,column);
    else
      {for (j=1; j<=r; j++) printf("\t%d",column[j]);
        puts("");
      }
  }
}
```



ricorsione non lineare

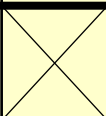
visualizza tutte le n^r disposizioni

```

column[k]=0;
while (column[k]<n)
{column[k] = column[k]+1;
if (k<r) sample(n,r,k+1,column); ...

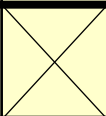
```

Chiamata [n=3, r=2, k=1]

k	column[]														
1		1												...	

if (k<r) then ... 

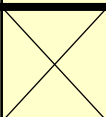
Chiamata [n=3, r=2, k=2]

k	column[]														
2		1	1, 2, 3											...	

if (k<r) else ...

caso banale

Chiamata [n=3, r=2, k=1]

k	column[]														
1		2												...	

then ...

Mutua ricorsione

struttura

```
procedure mutua_rec(...)  
begin  
  :  
  :  
  chiamata a altra(...)  
  :  
  :  
end
```

chiamata ricorsiva indiretta
tramite un'altra procedura.

```
procedure altra(...)  
begin  
  :  
  chiamata a mutua_rec(...)  
  :  
end
```


Esempio di mutua ricorsione in C

```
float recurs_fact(short n)
{float nfatt;
if (n <= 1) nfatt=1;
else nfatt= prod_fact(n);
return nfatt;
}

float prod_fact(short m)
{return m*recurs_fact(m-1);
}
```

Esercizi:

Scrivere 2 *function C* (risp. iterativa e ricorsiva) per ...

Calcolare (con ricorsione sia *lineare* che *binaria*):

- la somma delle componenti di un array;
- la potenza x^n ;