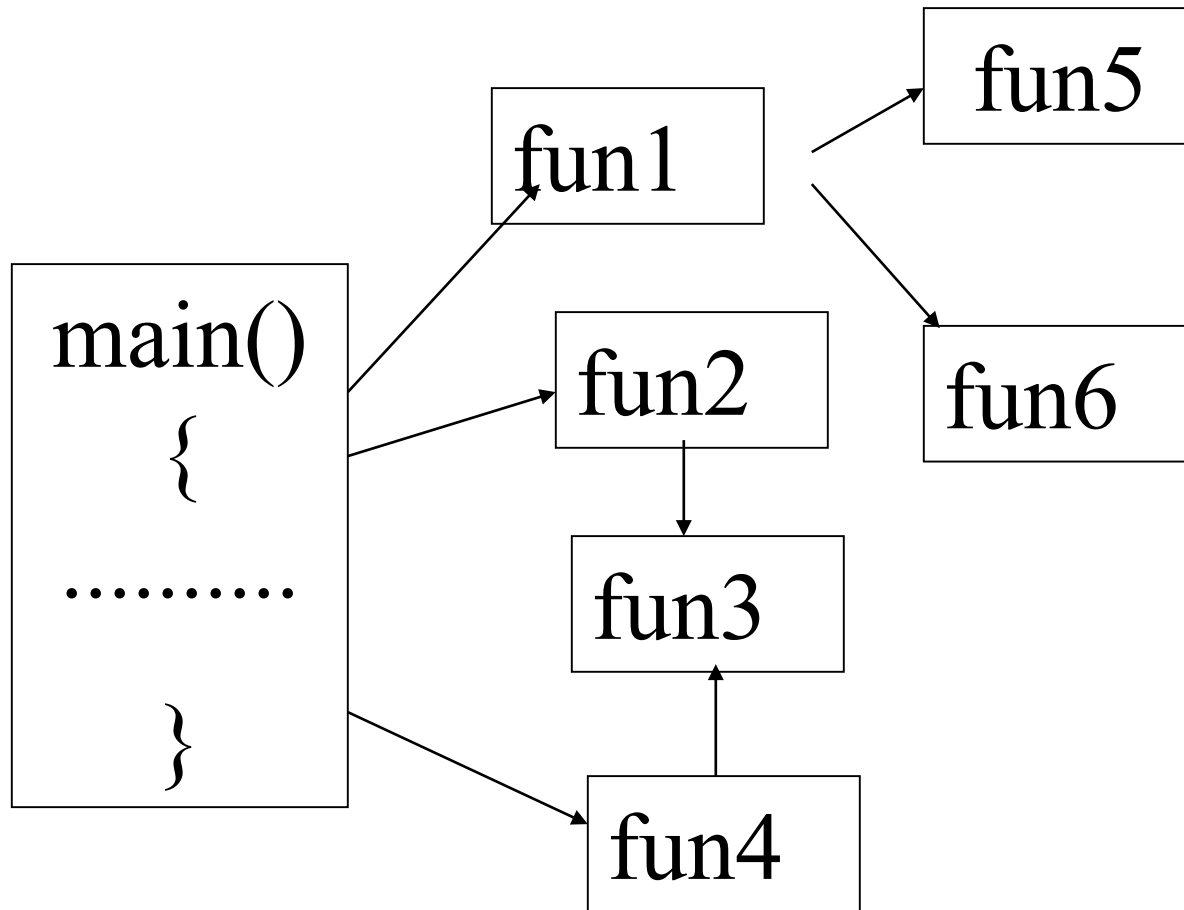


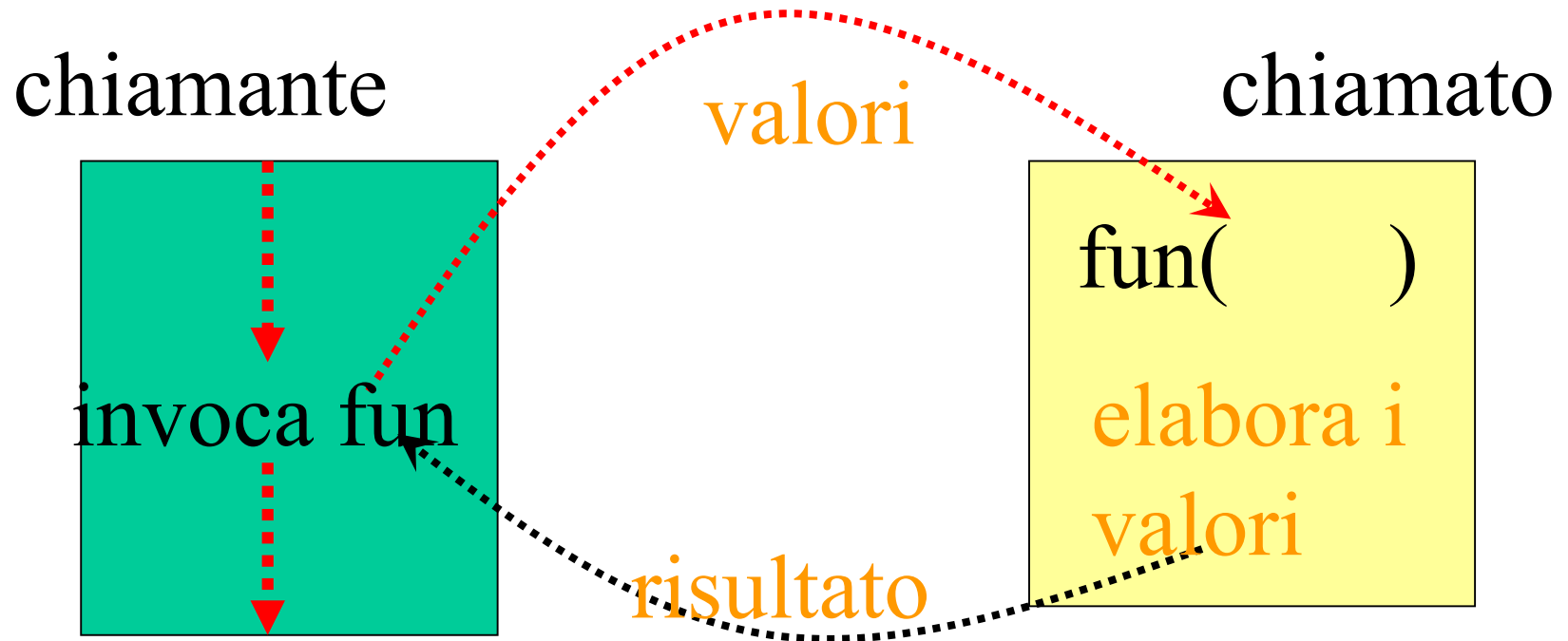
FUNZIONI

cap. 7 del testo

i programmi hanno questa struttura



Una funzione è un pezzo di programma con un nome. Essa viene eseguita tramite l'invocazione del suo nome.



esempio

funzione che calcola il più grande
divisore proprio di un naturale x dato

nome della
funzione

x è parametro
formale

```
int divisore(int x)
{
    return y;
}
```

qui entra il
valore su cui
lavorare

stesso
tipo

valore restituito

```
int divisore(int x) //PRE=(x definito)
{
    int y=x/2;
    while (x % y != 0)
        y=y-1;
    return y;
}
```

Post=(restituisce il max divisore di x)

invocazione della funzione divisore: calcolo del massimo numero primo più piccolo o uguale a z

```
int primo(int z)
```

parametro attuale

```
{int k=z;
```

```
while( k>1 && ! (divisore(k) == 1))
```

```
    k--;
```

```
return k;
```

```
} R=(1<=k<=z)&&(in [k+1..z] non ci  
sono primi)
```

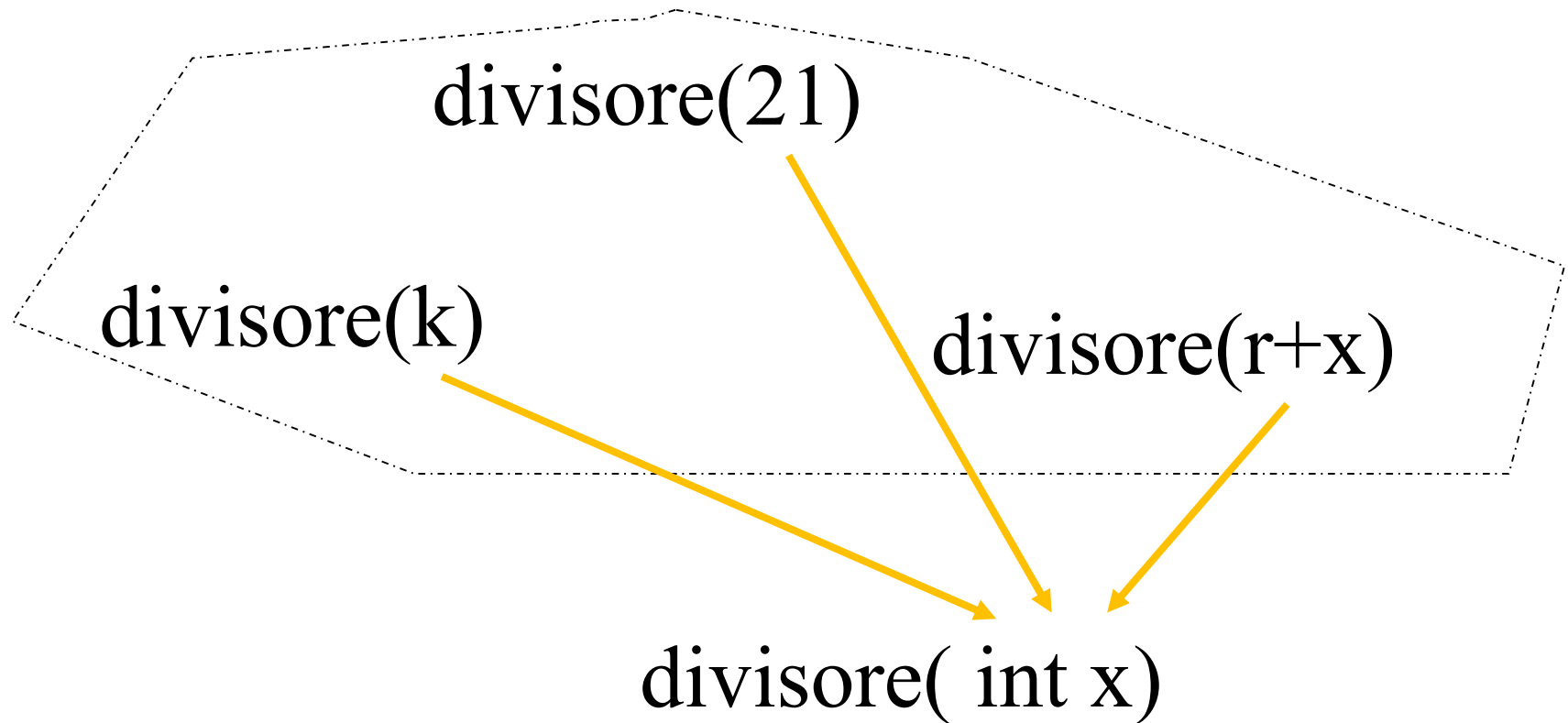
parametro attuale
divisore(k)

parametro formale
int divisore (int z)
{.....}

dovrebbero avere lo
stesso tipo
insomma k dovrebbe
avere valore int

altrimenti, si converte
a int
non c'è scelta

in generale ci sono molte invocazioni di una funzione



il parametro attuale è un valore che diventa
l'R-valore di x

passaggio dei parametri per valore

```
int F(int a, double b) {.....}
```

```
main()
```

```
{
```

```
  int x=10; double y=3,14;
```

```
  int z=F(x, y); //invocazione
```

```
}
```



		x	y		a	b		
RAM		10	3,14		10	3,14		

qualunque cosa succeda ad a e b non ha alcun effetto su x e y

la funzione non produce side-effects

si possono definire puntatori, cioè variabili il cui valore è l'L-valore di altre variabili

```
int x=0, *p=&x;
```

l'R-valore di p è l'L-valore
di x

*p è x

in *p, * dereferenzia p e ottiene l'oggetto
puntato, cioè x

si possono definire anche riferimenti.

```
int y=1;
```

```
int & z = y;
```

z e y hanno lo stesso

L-valore e lo stesso

R-valore

`int x=18, &y=x;` indica che `y` è alias di `x`

```
cout<< x << ' ' <<& x<< endl;
```

```
cout<< y << ' ' <<& y<< endl;
```

mostra che `x` e `y` hanno lo stesso R- ed L-valore. Sono la stessa variabile, e quindi, cambiando uno cambia anche l'altro

```
cout<<x; // stampa 18
```

```
y=y+2;
```

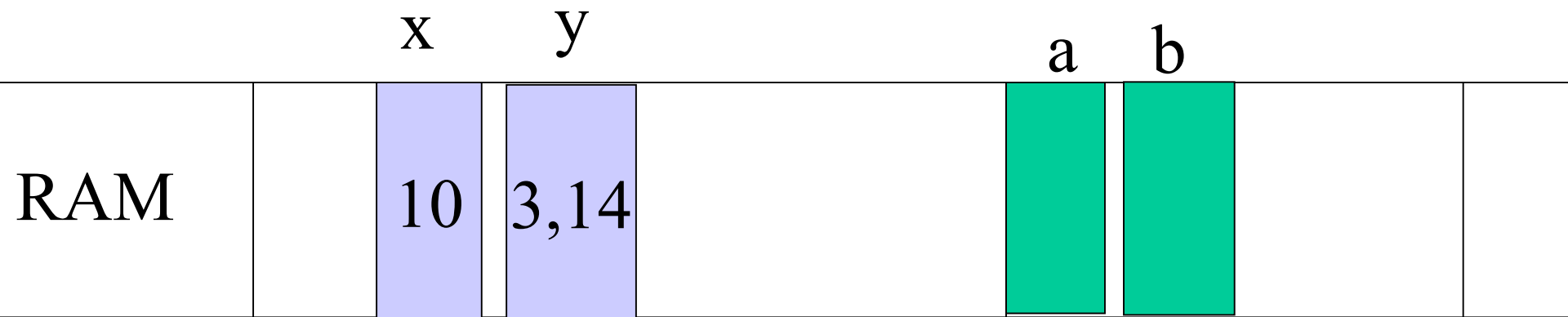
```
cout << x; // stampa 20
```

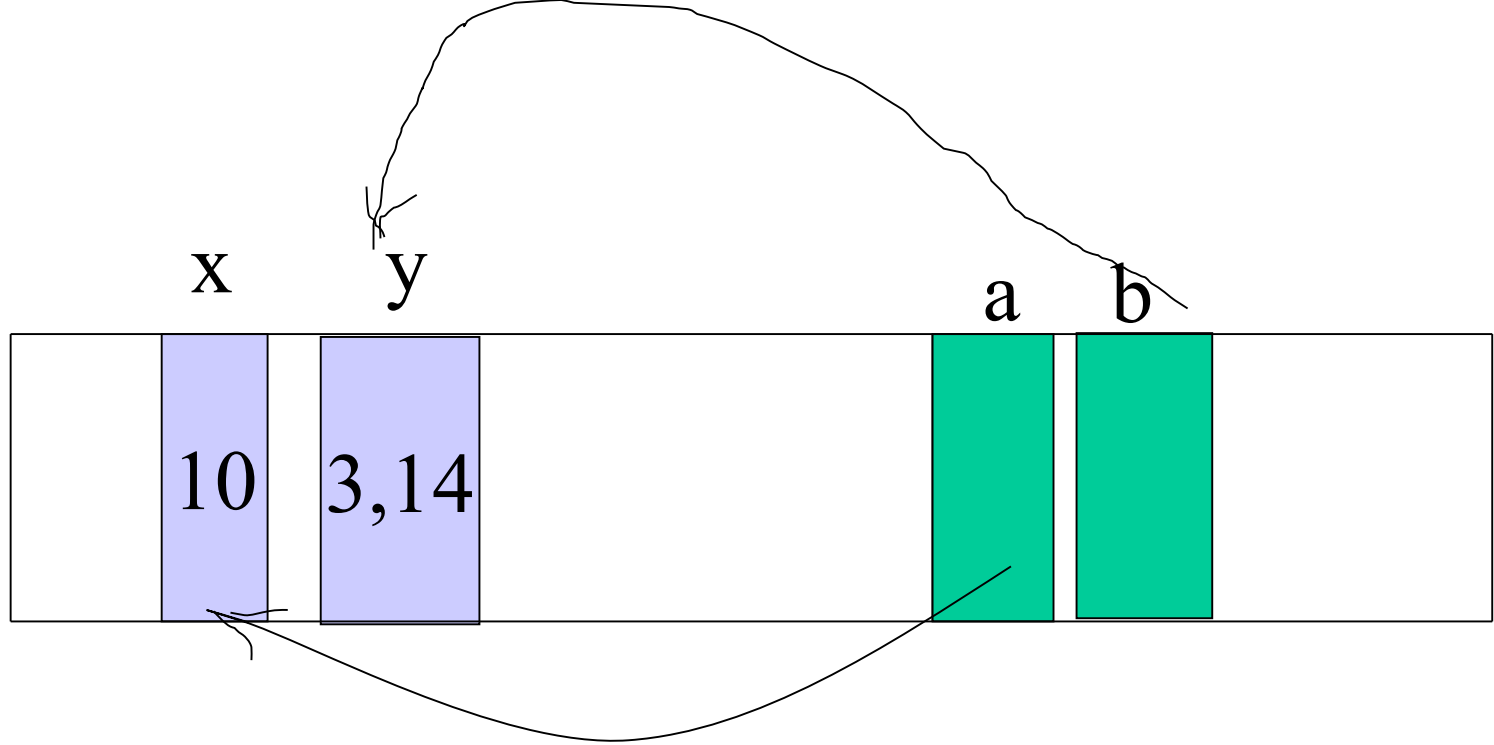
effetti collaterali con puntatori

```
int F(int*a, double *b) {.....}
```

invocazione F(&x, &y)

per valore





$*a \equiv x$

$*b \equiv y$ quindi è facile modificare x e y

effetti collaterali con riferimenti

```
int F(int & a, double & b) {.....}
```

invocazione F(x, y)

x e y sono passate
per riferimento

$x \equiv a$ $y \equiv b$

RAM		10	3,14
-----	--	----	------

è come se F
possedesse x e y, ma
li chiama a e b
sono alias

attenzione: * e & hanno 2 significati :

- nelle dichiarazioni `int * p;` indica che `p` è un puntatore a intero,
- nelle espressioni `*p` dereferenzia `p` e ottiene l'oggetto puntato,
- nei parametri formali `char & x` indica che `x` è passato per riferimento,
- in `&x` è un operatore che calcola l'L-valore di `x`

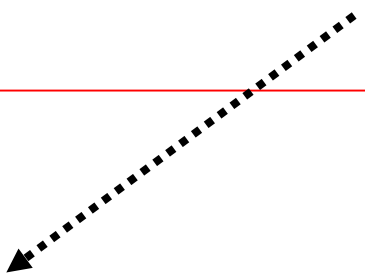
```
void g(int x, int & y)
{ x++; y++; }
main()
{
int a=10;
g(a,a); }

// valore di A ?
```

e con & ?

e se ?

```
void g(int x, int & y)
{ x++; y++;}
```



2 modalità di passaggio dei parametri

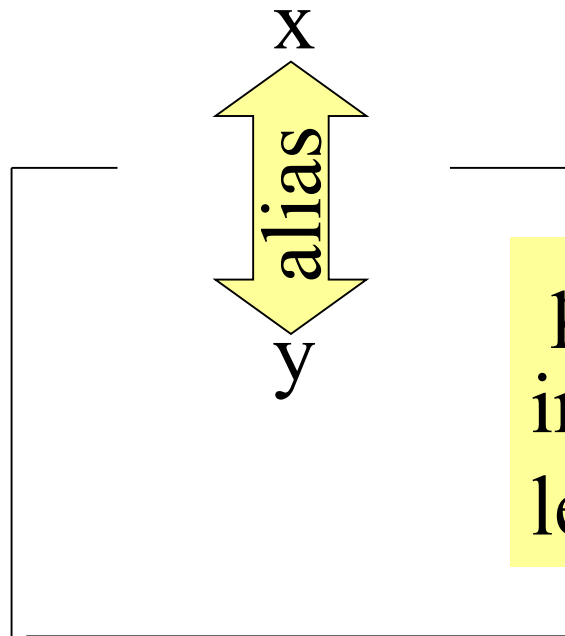
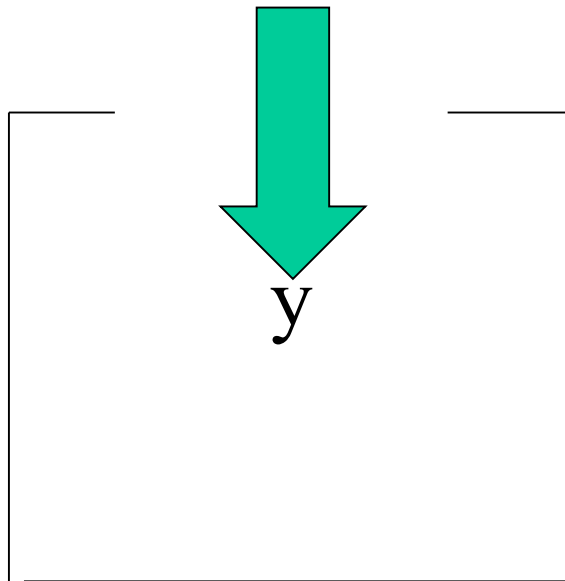
per valore

per riferimento

parametri attuali

espressioni

variabili



può servire
in entrambe
le direzioni

- il passaggio per riferimento è diverso dai puntatori passati per valore ?
- e ha senso passare un puntatore per riferimento?
- e...

gestione in memoria delle variabili delle funzioni

supponiamo che il main invochi $f(z,w)$

```
int f(int x, float y)
```

```
{  ....
```

```
    G(a,b)
```

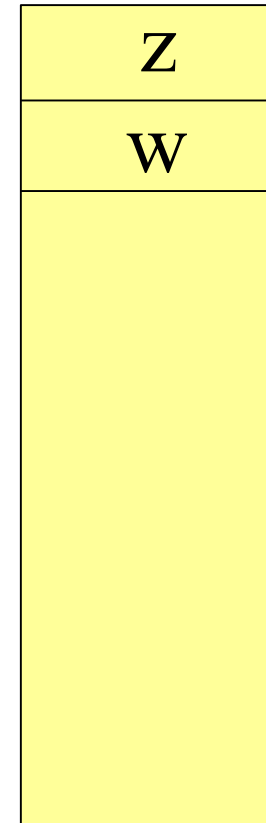
```
}
```

```
double G(char c, double d)
```

```
{
```

```
.....
```

```
}
```



pila dei
Record
di
Attivaz
ione
(RA)

dati automatici

esercizio

```
char x='a', y='b';
```

```
F(x,y);
```

```
// qui vogliamo che x=='b' e y=='a'
```

```
come deve essere ? F( ?, ?){??}
```

```
//PRE=(a e b >0 e a>=b)
```

```
int MCD(int a, int b)
```

```
{
```

```
int r=a % b;
```

```
while(r > 0)
```

```
{ a=b; b=r; r=a%b;}
```

```
return b;
```

```
}
```

```
int mcm(int a, int b)
```

```
return (a*b) / MCD(a,b);
```

2 funzioni
interessanti