

# Linguaggio C

## funzioni e procedure

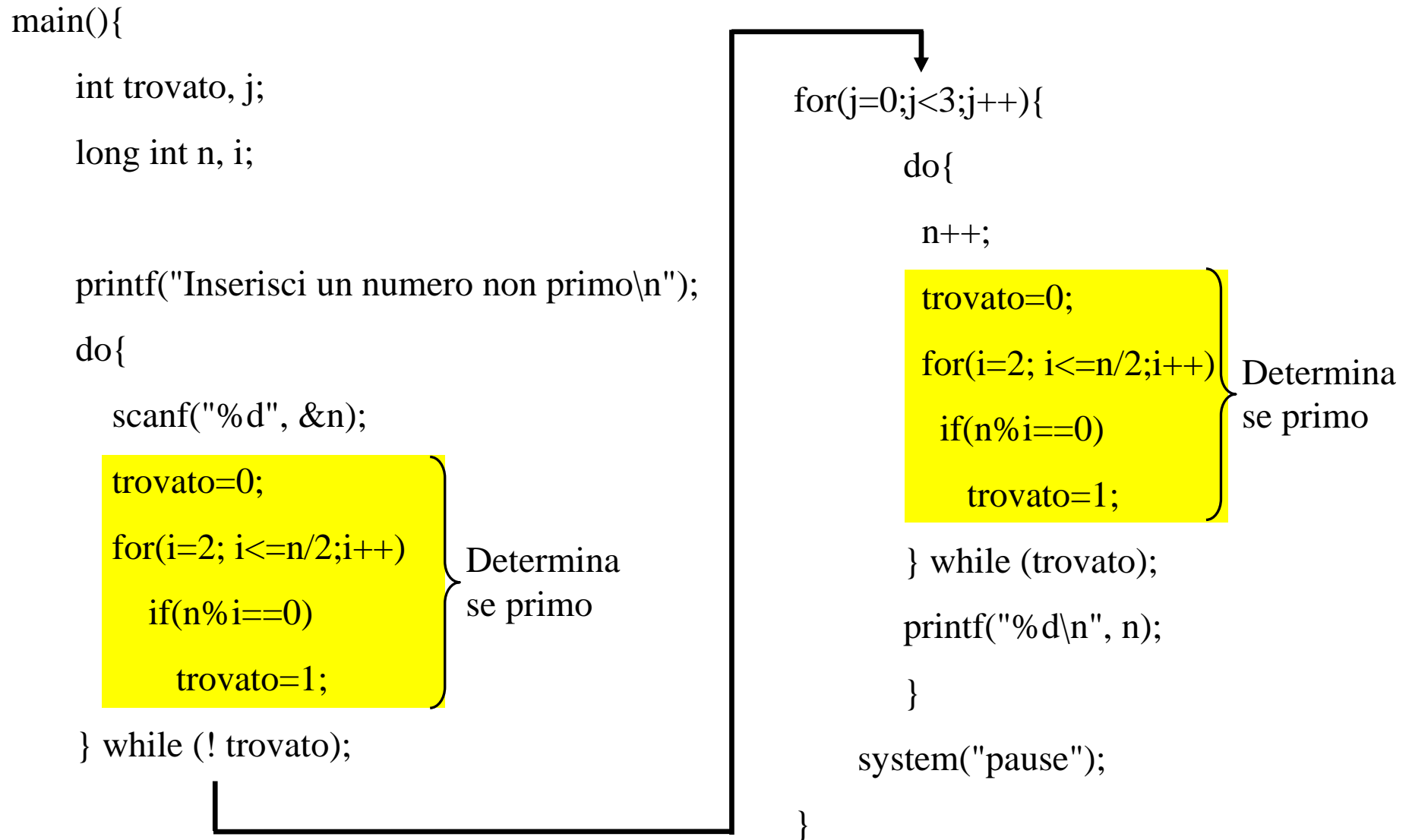
Università degli Studi di Brescia

*Docente: Massimiliano Giacomini*

## Un esempio

Acquisire dall'utente un numero intero  $n$ , ripetendo l'acquisizione se è primo. Successivamente, stampare 5 numeri primi superiori a  $n$ .

*Vediamo come risolveremmo il problema con le strutture note...*



## Una soluzione più elegante: definiamo la funzione “primo”

```
main(){
```

```
    int primo(long int num){  
        long int i;  
        int trovato=0;  
        for(i=2; i<=num/2;i++)  
            if(num%i==0)  
                trovato=1;  
        return(!trovato);  
    }
```

```
    long int n;
```

```
    int i;
```

```
    printf("Inserisci un numero non primo\n");
```

```
    do
```

```
        scanf("%d", &n);
```

```
    while(primo(n));
```

```
    for(i=0;i<3;i++){
```

```
        do
```

```
            n++;
```

```
        while(!primo(n));
```

```
        printf("%d\n", n);
```

```
    }
```

```
    system("pause");
```

```
}
```

# Definizione di funzioni in C

## Esempio

Parametri formali  
(argomenti della funzione)

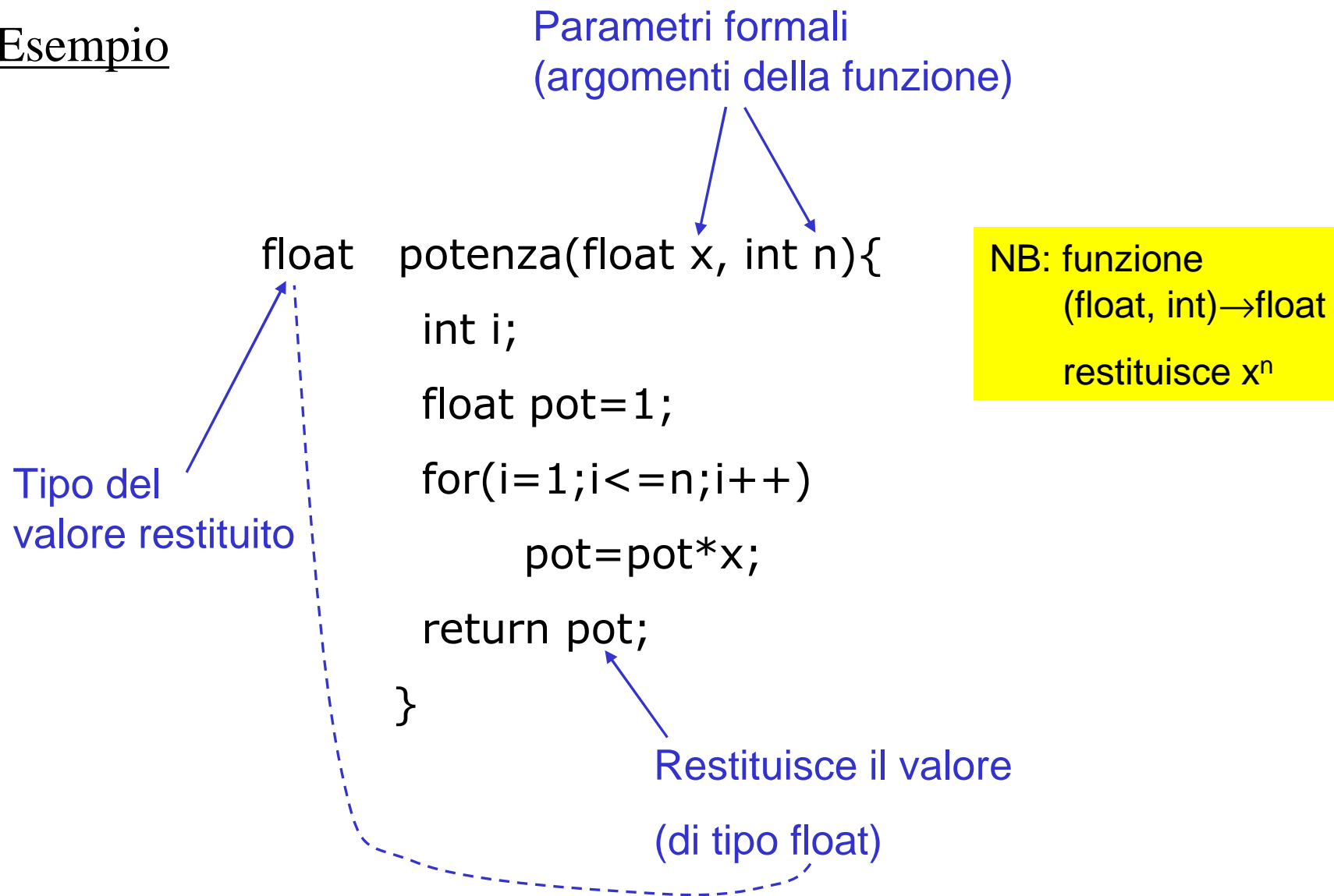
```
float potenza(float x, int n){  
    int i;  
    float pot=1;  
    for(i=1;i<=n;i++){  
        pot=pot*x;  
    }  
    return pot;  
}
```

Tipo del  
valore restituito

NB: funzione  
(float, int)→float  
restituisce  $x^n$

# Definizione di funzioni in C

## Esempio



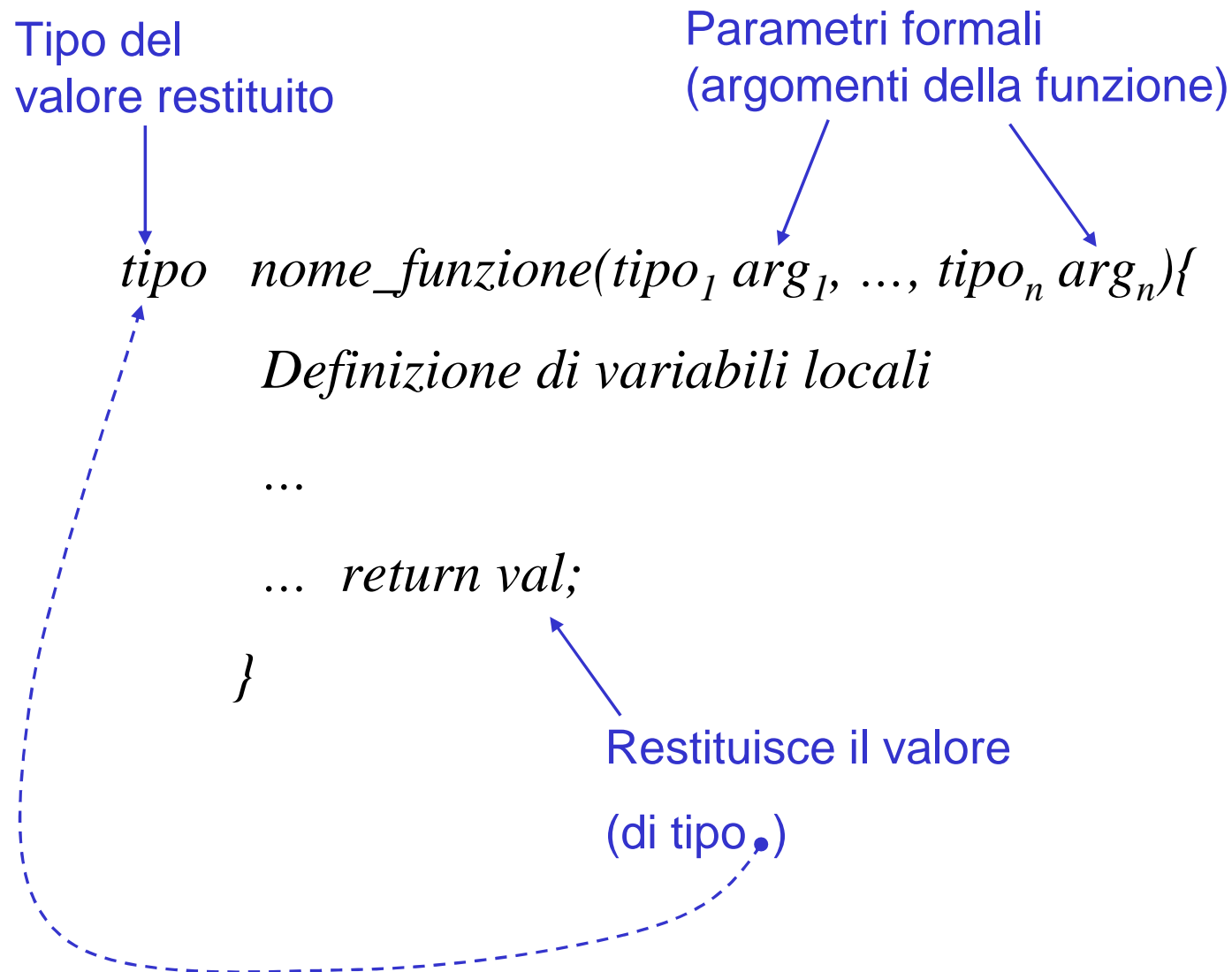
# Definizione di funzioni in C

## Esempio

30.25 ← float     $\begin{matrix} 5.5 \\ \downarrow \\ \text{float } x \end{matrix}$      $\begin{matrix} 2 \\ \downarrow \\ \text{int } n \end{matrix}$     potenza(float x, int n){  
    int i;  
    float pot=1;  
    for(i=1;i<=n;i++)  
        pot=pot\*x;  
    return pot;  
}

NB: funzione  
(float, int)→float  
restituisce  $x^n$

## Definizione di funzioni in C (in generale)





## Chiamata di funzioni in C

Se nel programma vogliamo usare la funzione definita, dobbiamo “chiamarla” passandole i *parametri attuali*

### Esempio precedente

```
...  
b= potenza(5.5, 2);  
...  
float  potenza(float x, int n){  
    ...  
    return pot;  
}
```

### In generale

*nome\_funzione*(*arg\_att*<sub>1</sub>, ..., *arg\_att*<sub>*n*</sub>)

Parametri attuali: sono valori  
i cui tipi devono corrispondere  
a quelli dei parametri formali

# Esecuzione delle funzioni e passaggio dei parametri in C

## Prima della chiamata

### Programma chiamante

b

...

b= potenza(5.5, 2);

...

### Funzione Potenza

x

n

Parametri  
formali

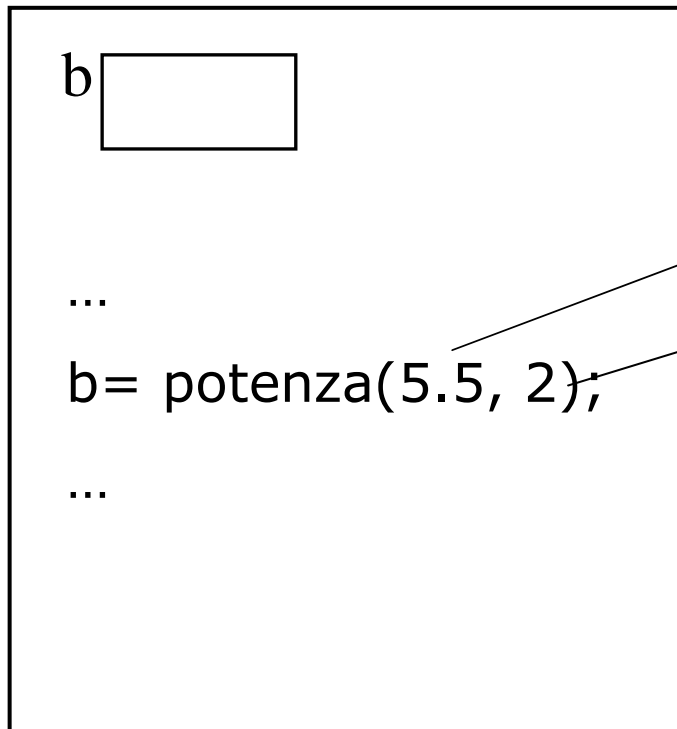
i

pot

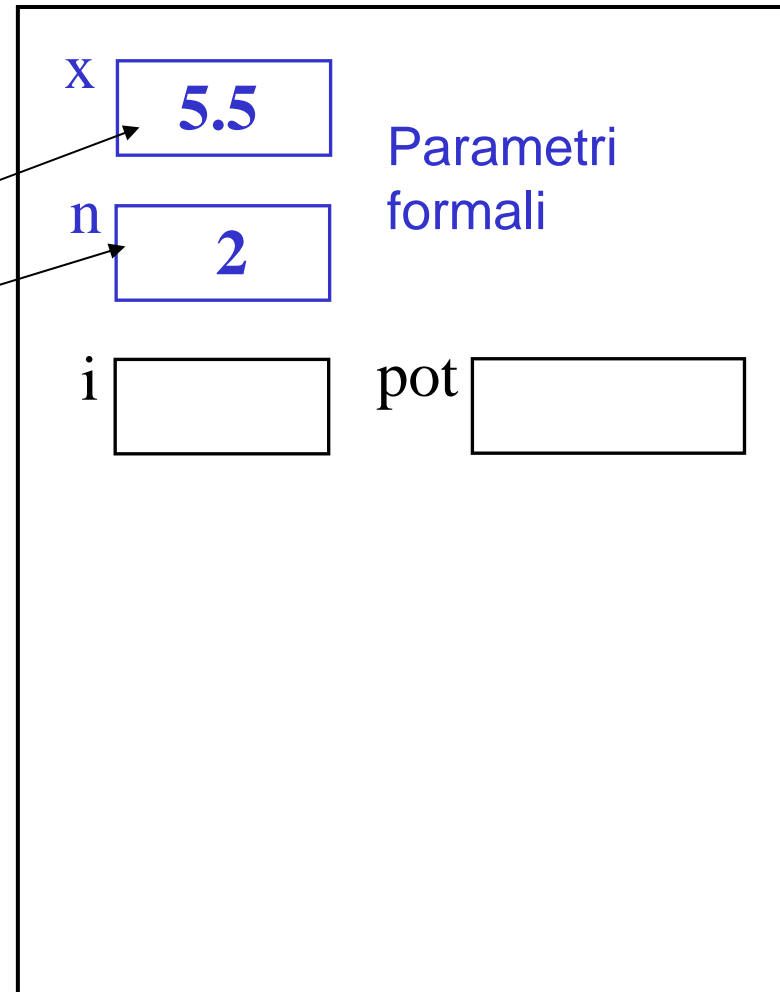
# Esecuzione delle funzioni e passaggio dei parametri in C

Chiamata: passaggio dei parametri

## Programma chiamante



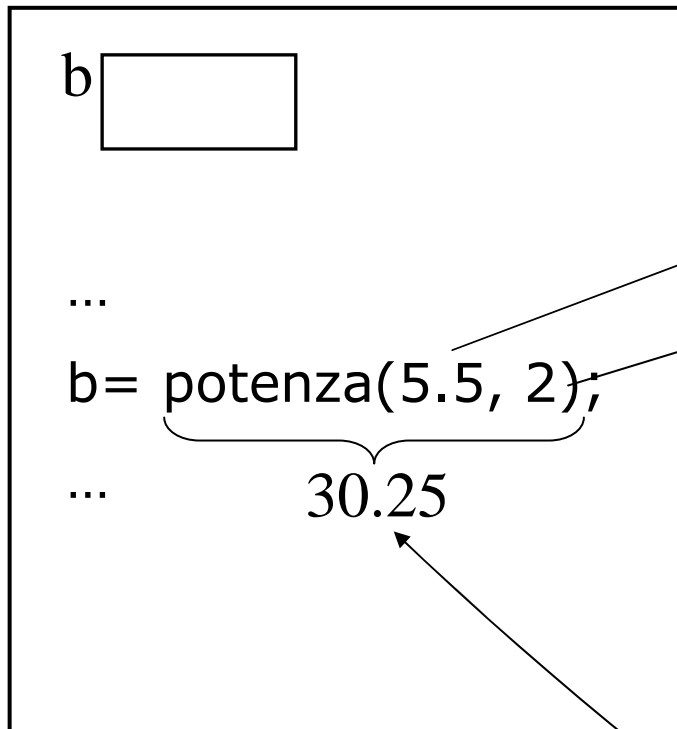
## Funzione Potenza



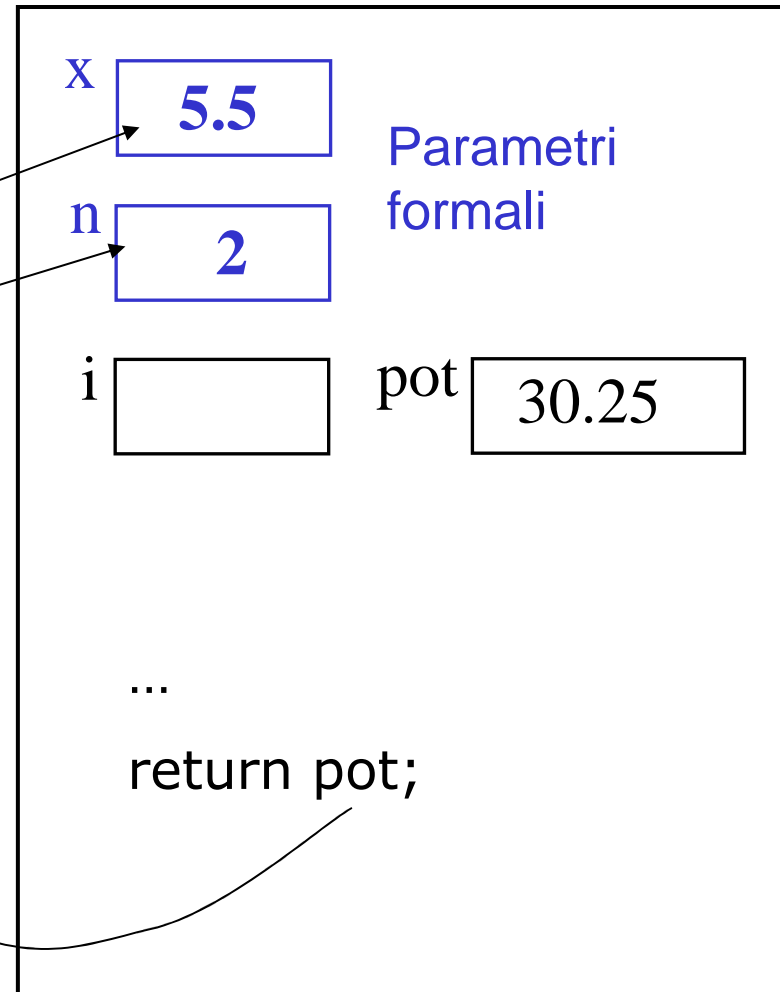
# Esecuzione delle funzioni e passaggio dei parametri in C

## Ritorno

### Programma chiamante



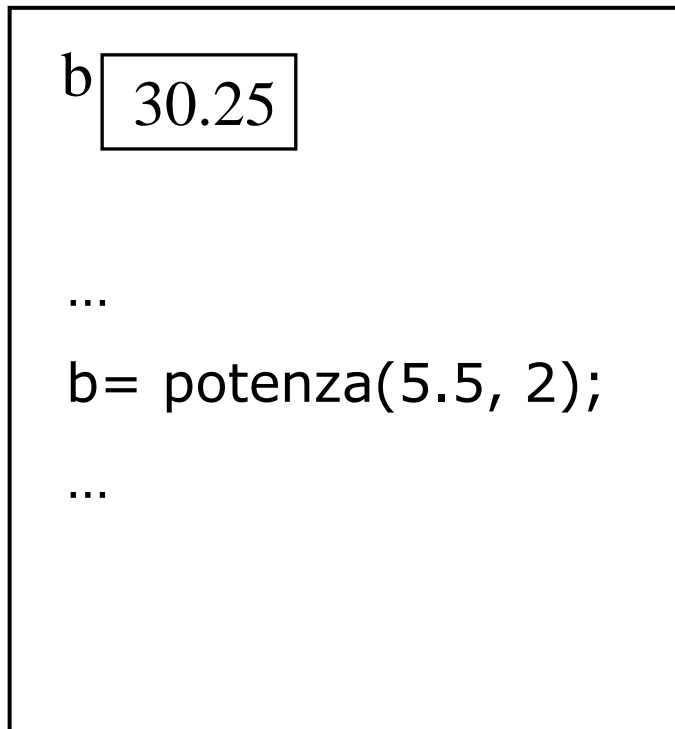
### Funzione Potenza



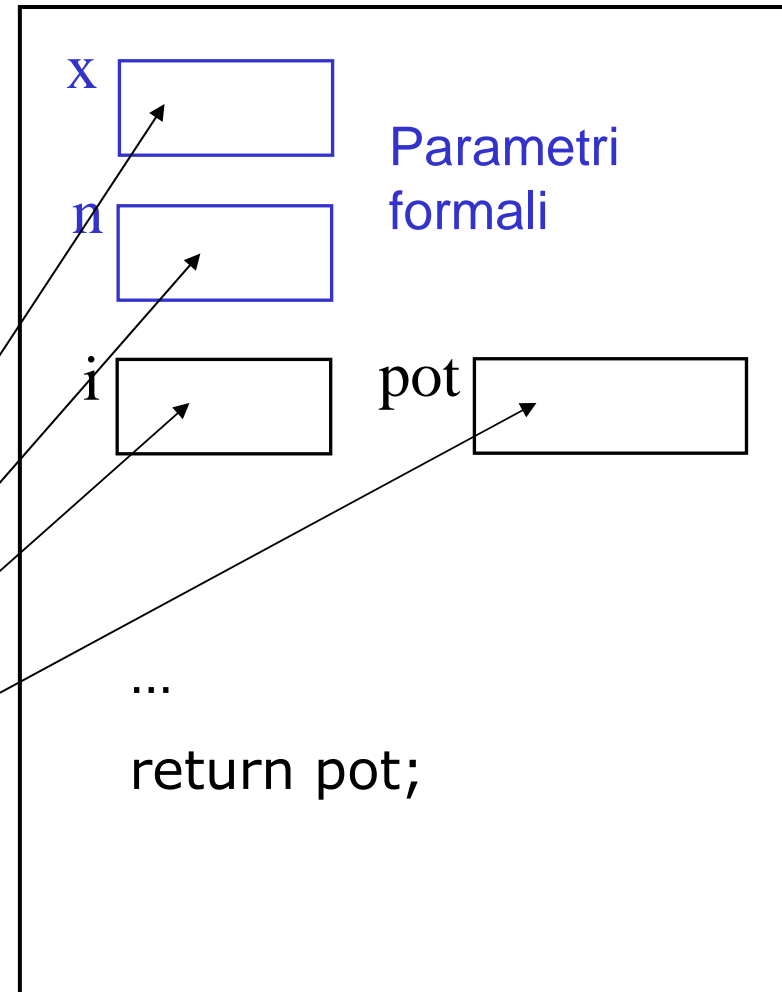
# Esecuzione delle funzioni e passaggio dei parametri in C

Dopo il ritorno

**Programma chiamante**



**Funzione Potenza**



i valori non vengono conservati!

## Le procedure

- In C: funzioni che non ritornano un valore (tipo ritorno void)

### Esempio

```
typedef struct {  
    char nome[20];  
    char cognome[20];  
    int eta;  
} persona;
```

```
void stampadati(persona p){  
    printf("Nome: %s\n", p.nome);  
    printf("Cognome: %s\n", p.cognome);  
    printf("%d anni\n", p.eta);  
}
```

# Visibilità delle variabili (1)

Programma

x

i

Procedura

y

i

...

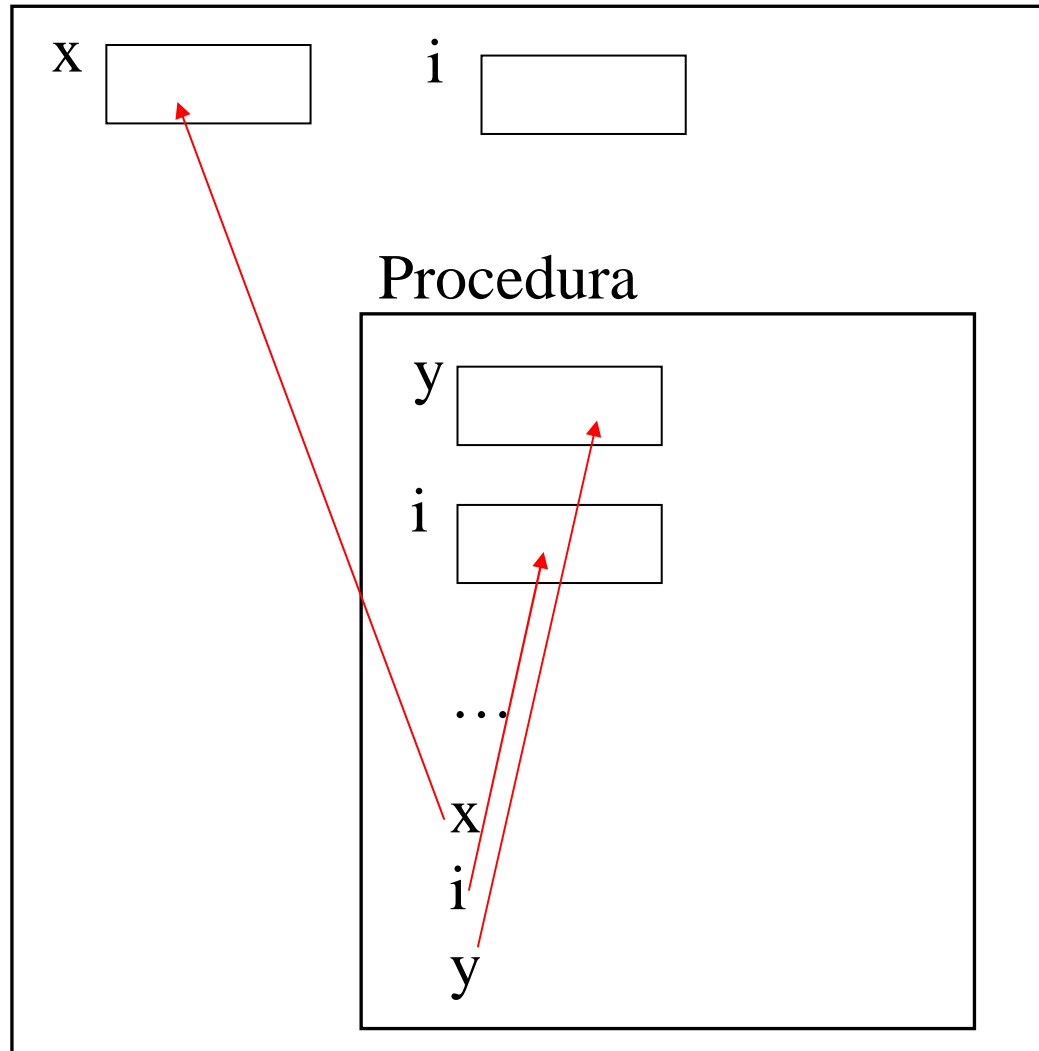
x

i

y

## Visibilità delle variabili (1)

Programma



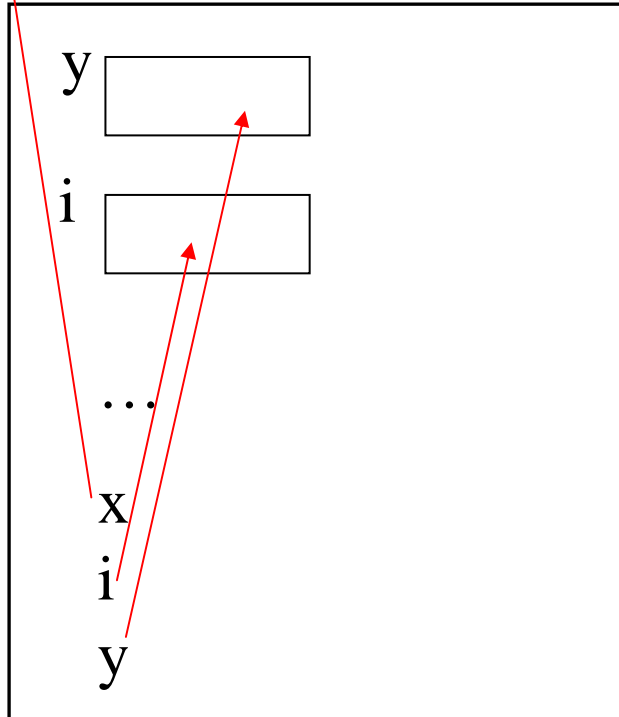
- Le variabili globali sono accessibili dalla procedura
- Le variabili locali (inclusi i parametri formali) “mascherano” le variabili globali con lo stesso nome
- Come noto, le locali non lasciano traccia dopo l’uscita dalla procedura



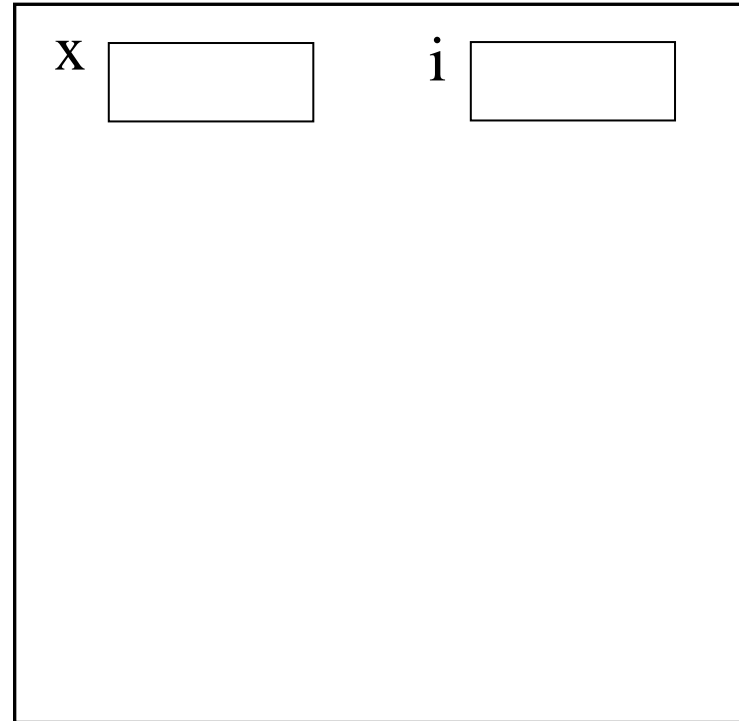
## Visibilità delle variabili (2)

### VARIABILE INDEFINITA

Procedura



Programma



## ESEMPIO 1

```
main(){  
    int a=2, b=5;  
    int i=0, j;  
  
    int mul(int a, int b){  
        int i;  
        i=a*b;  
        return i;  
    }  
  
    j = mul(a, b);  
    system("PAUSE");  
}
```

NB: i rimane 0

## ESEMPIO 2

```
main(){  
    int a=2, b=5;  
    int i=0, j;  
  
    int mul(int a, int b){  
        i=a*b;  
        return i;  
    }  
  
    j = mul(a, b);  
    system("PAUSE")  
}
```

NB: i diventa 10

## ESEMPIO 3

```
int mul(int a, int b){  
    i=a*b;  
    return i;  
}  
  
main(){  
    int a=2, b=5;  
    int i=0, j;  
  
    j = mul(a, b);  
    system("PAUSE");  
}
```

NB: non compila!