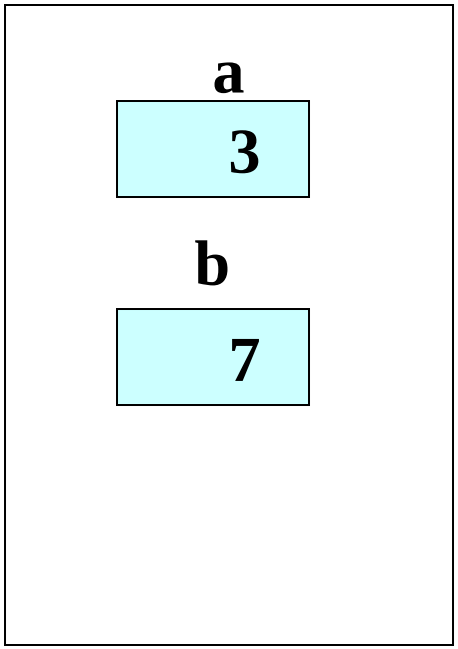
I sottoprogrammi: passaggio per indirizzo

Esempio: scambio di 2 valori interi

```
void swap (int p, int q) {
      int temp;
      temp = p;
      p = q;
      q = temp;
}
```

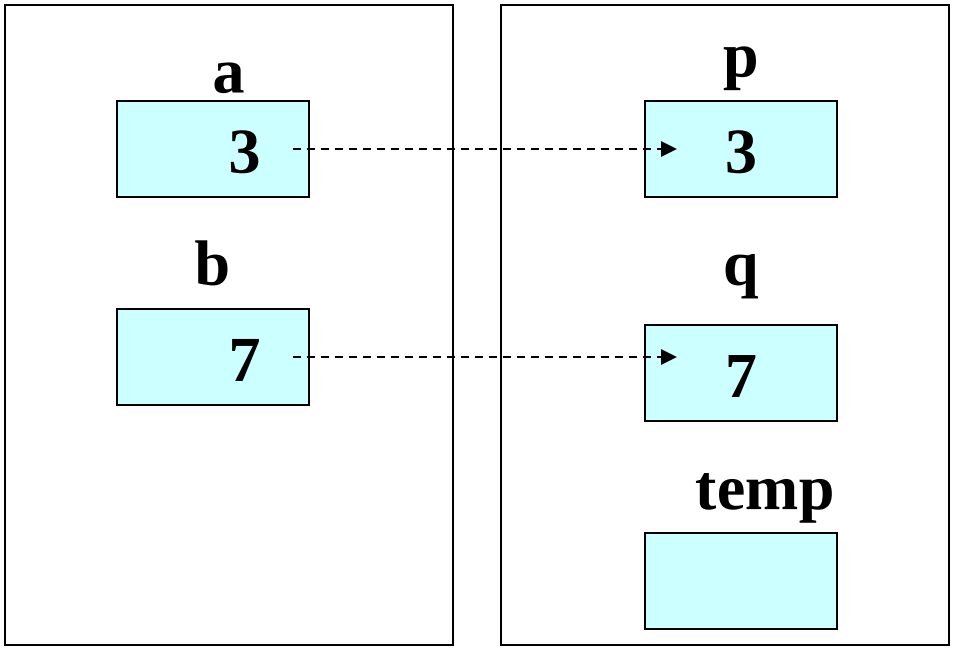
Nel main: `swap(a,b)`



Esempio: scambio di 2 valori interi

```
void swap (int p, int q) {
      int temp;
      temp = p;
      p = q;
      q = temp;
}
```

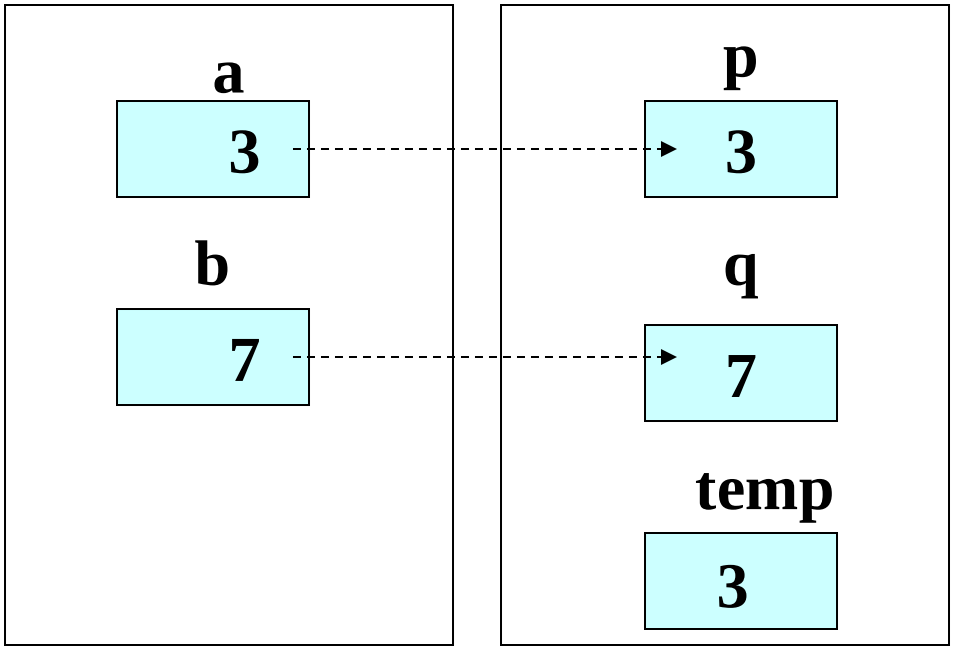
Nel main: `swap(a,b)`



Esempio: scambio di 2 valori interi

```
void swap (int p, int q) {
      int temp;
      temp = p;
      p = q;
      q = temp;
}
```

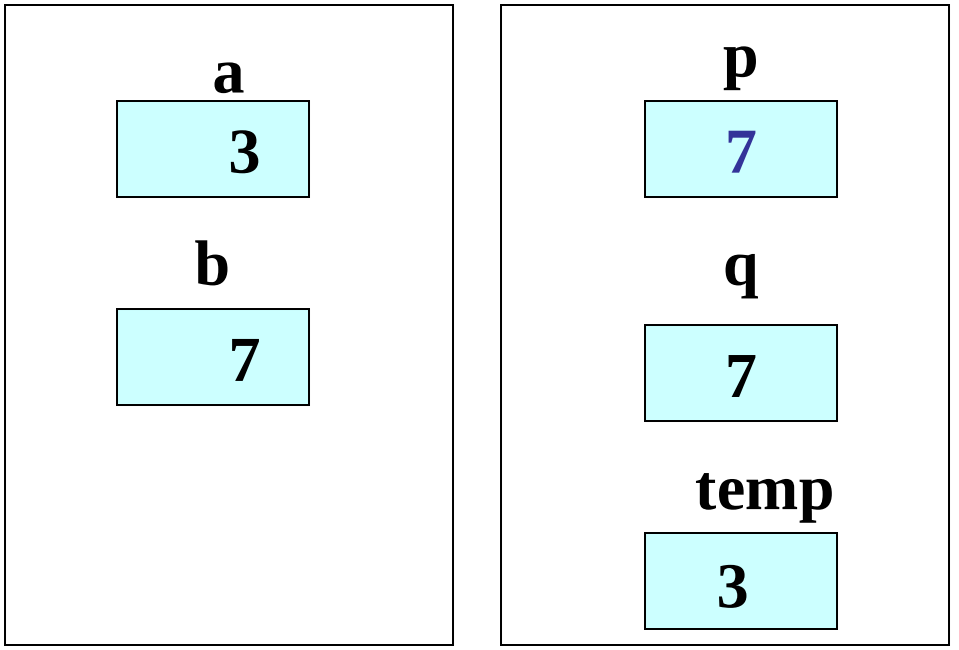
Nel main: `swap(a,b)`



Esempio: scambio di 2 valori interi

```
void swap (int p, int q) {
      int temp;
      temp = p;
      p = q;
      q = temp;
}
```

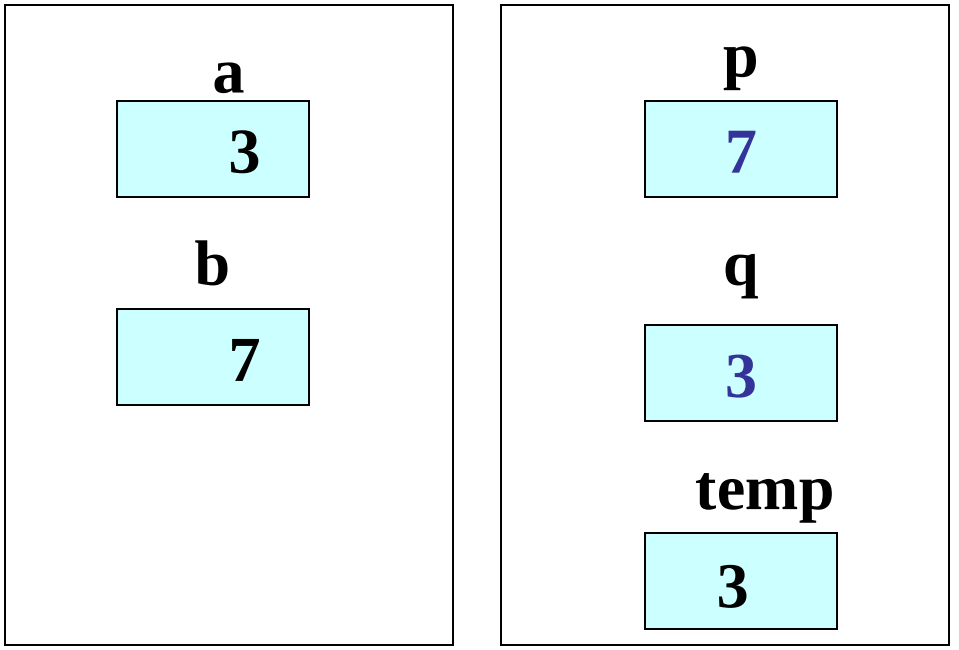
Nel main: `swap(a,b)`



Esempio: scambio di 2 valori interi

```
void swap (int p, int q) {
      int temp;
      temp = p;
      p = q;
      q = temp;
}
```

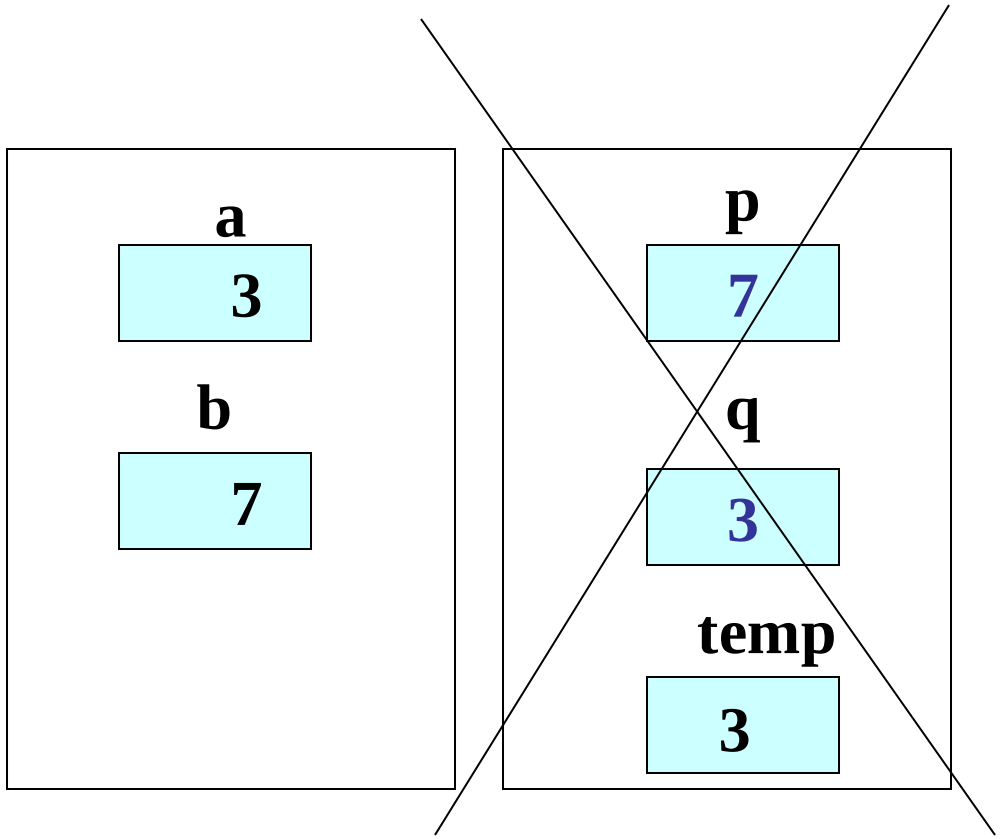
Nel main: `swap(a,b)`



Esempio: scambio di 2 valori interi

```
void swap (int p, int q) {
      int temp;
      temp = p;
      p = q;
      q = temp;
}
```

Nel main: `swap(a,b)`



Al termine dell'esecuzione di `swap` le variabili nel main restano inalterate!

Passaggio per indirizzo

- All'atto della chiamata l'indirizzo dei parametri attuali viene associato ai parametri formali
  - Il parametro attuale e il parametro formale **si riferiscono alla stessa cella di memoria**
- Il sottoprogramma in esecuzione lavora nel suo ambiente sui parametri formali (e di conseguenza anche sui parametri attuali) e ogni modifica sul parametro formale è una modifica del corrispondente parametro attuale
- Gli effetti del sottoprogramma si manifestano nel chiamante con modifiche al suo ambiente locale di esecuzione
- **Meccanismo per implementare uno scambio di informazioni bidirezionale con le funzioni**

Passaggio per indirizzo in C

- Il passaggio per indirizzo è realizzato mediante l'utilizzo dei puntatori
  - Nel prototipo della funzione si specifica il parametro formale di tipo puntatore
  - All'atto della chiamata si passa come parametro attuale l'indirizzo della variabile (con l'operatore &)

- È possibile passare variabili di tutti i tipi di dato già visti con il passaggio per valore
- Il passaggio delle struct è più efficiente poiché si copia solo un indirizzo e non l'intero contenuto della struttura (che può avere dimensioni considerevoli)
- È possibile passare anche array (come vi vedrà nel seguito)

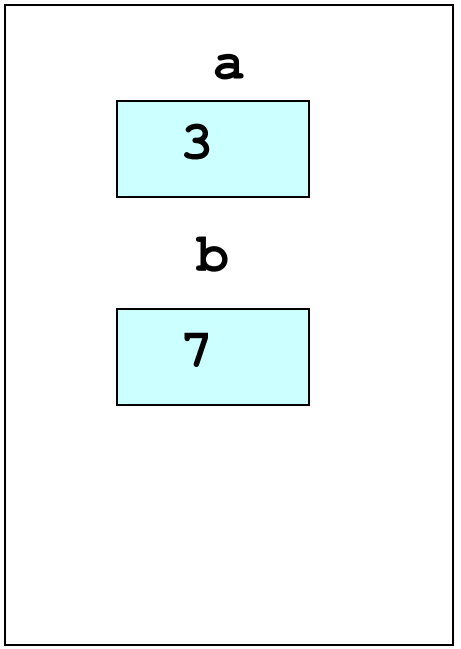
Esempio di passaggio per indirizzo in C

```
...
void swap (int *p, int *q);

void swap (int *p, int *q){
        int temp;
        temp = *p;
        *p = *q;
        *q = temp;
}

void main(){
   int a, b;
  a=3;
  b=7;
  swap(&a,&b);
  ...
}
```

Prima dell'esecuzione di `swap`



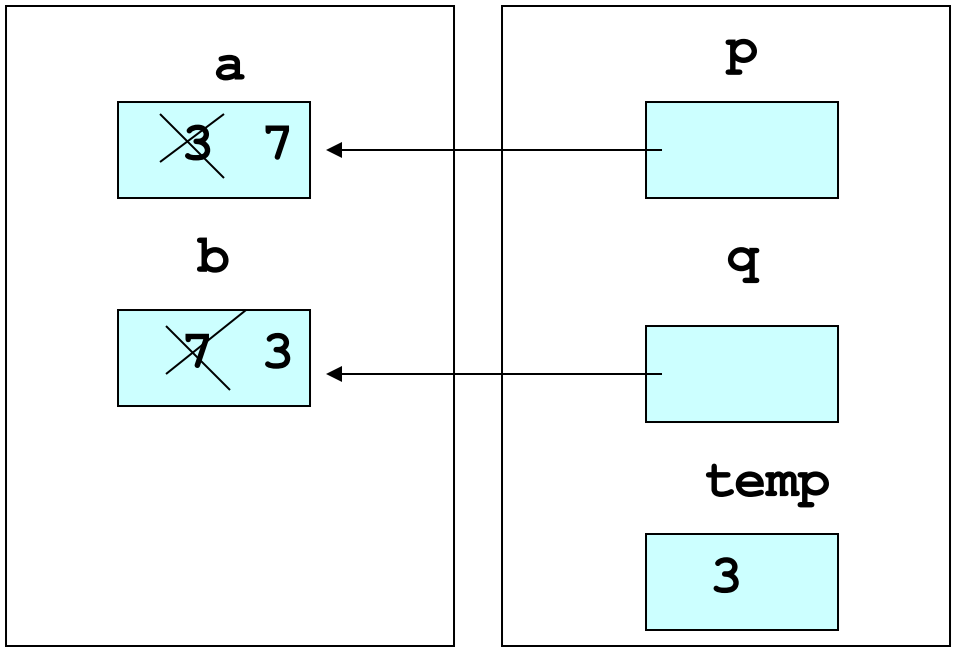
Esempio di passaggio per indirizzo in C

```
...
void swap (int *p, int *q);

void swap (int *p, int *q){
        int temp;
        temp = *p;
        *p = *q;
        *q = temp;
}

void main(){
   int a, b;
  a=3;
  b=7;
  swap(&a,&b);
  ...
}
```

Alla fine dell'esecuzione di `swap`



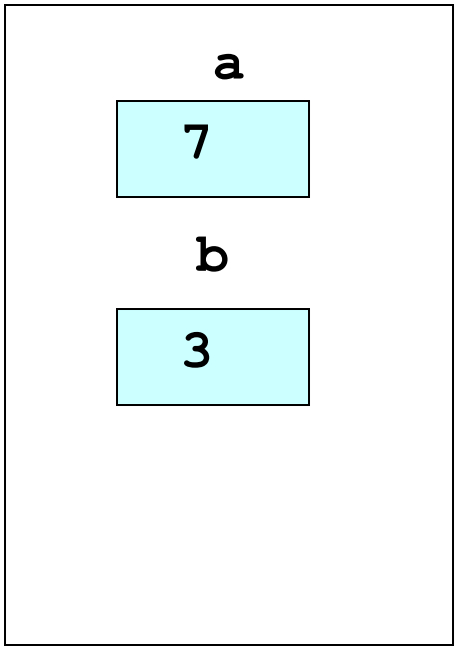
Esempio di passaggio per indirizzo in C

```
...
void swap (int *p, int *q);

void swap (int *p, int *q){
        int temp;
        temp = *p;
        *p = *q;
        *q = temp;
}

void main(){
   int a, b;
  a=3;
  b=7;
  swap(&a,&b);
  ...
}
```

Dopo l'esecuzione di swap



Passaggio di parametri di tipo array C

- Prototipo del sottoprogramma:

```
void invertiArray(int [], int);
```

- Sottoprogramma:

```
void invertiArray(int a[], int dim){
  int tmp;
  for(i=0; i<dim/2; i++){
    tmp=a[i];
    a[i]=a[dim-1-i];
    a[dim-1-i]=tmp;
  }
}
```

- Invocazione:

```
invertiArray(array,dimArray);
```

Passaggio di parametri di tipo array C

- È possibile utilizzare anche un puntatore:

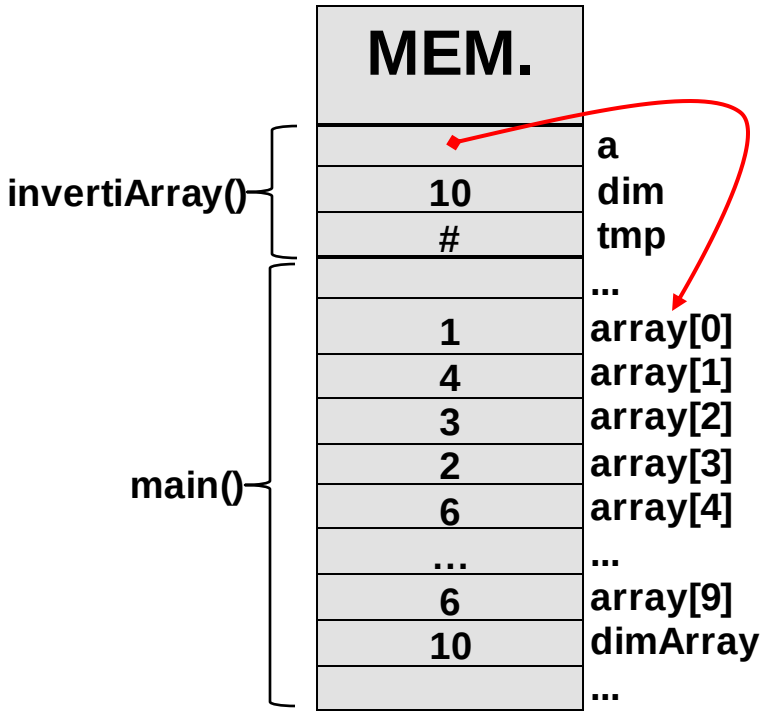
```
void invertiArray(int* , int);
```

- E quindi nel sottoprogramma utilizzare l'aritmetica dei puntatori:

```
void invertiArray(int a*, int dim){
   int tmp;
   for(i=0; i<dim/2; i++){
      tmp=*(a+i);
      *(a+i)=*(a+dim-1-i);
      *(a+dim-1-i)=tmp;
   }
}
```

Passaggio di parametri di tipo array C

- Stato della memoria durante l'invocazione della funzione `invertiArray` eseguita nel `main`



Il parametro formale `a` contiene l'indirizzo del parametro attuale `array`. Ogni modifica viene effettuata direttamente su `array`

Passaggio di parametri di tipo array C

- Per gli array monodimensionali non va specificata la dimensione tra le parentesi sia nel prototipo che nell'intestazione:

```
void invertiArray(int [], int);
```

- Per gli array multidimensionali si deve specificare tutte le dimensioni dalla seconda in poi (per permettere la delinearizzazione della matrice), sia nel prototipo che nell'intestazione:

```
void trasponiMatriceArray(int a[][N], int dim1, int dim2);
```

- È consigliabile passare come parametro la dimensione dell'array
  - Il sottoprogramma in alternativa dovrebbe utilizzare la costante utilizzata nella definizione dell'array
  - È sempre nota quando si realizza una funzione di libreria?

- Non si può specificare un array come valore di `return`