Le Funzioni

Maurizio Palesi
Salvatore Serrano

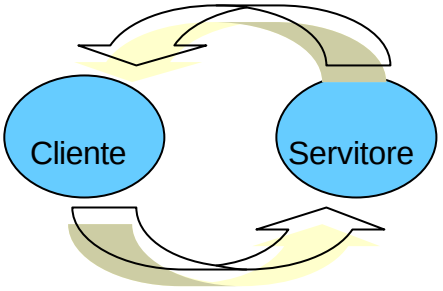
Funzioni

- Spesso può essere utile avere la possibilità di costruire nuove istruzioni che risolvono parti specifiche di un problema.
- Una *funzione* permette di
  - ➔Dare un nome ad un'espressione
  - ➔Rendendola parametrica
- Esempi (pseudo-C)
  - ➔`float f() { 2 + 3 * sin(0.75); }`
  - ➔`float f1(int x) { 2 + x * sin(0.75); }`

Funzioni come Componenti Software

- Una *funzione* è come un *componente software* che cattura l'idea matematica di funzione
  - ➔Molti possibili ingressi (che non vengono modificati)
  - ➔Una sola uscita (il risultato)
- Una funzione
  - ➔Riceve dati in ingresso in corrispondenza ai **parametri**
  - ➔Ha come corpo una espressione la cui valutazione **fornisce un risultato**
  - ➔Denota un valore in corrispondenza al suo **nome**

Modello Cliente/Servente



- Servitore
  - ➔Ente computazionale capace di nascondere la propria organizzazione interna
  - ➔Presenta ai clienti una precisa interfaccia per lo scambio di informazioni
- Cliente
  - ➔Qualunque ente in grado di invocare uno o più servitori per svolgere il proprio compito

Interfaccia di una Funzione

- L'interfaccia (o prototipo) di una funzione comprende
  - ➔Nome della funzione
  - ➔Lista dei parametri
  - ➔Tipo del valore da essa restituito
- Cliente e servitore comunicano mediante
  - ➔I parametri: trasmessi dal cliente al servitore all'atto della chiamata (direzione: cliente → servitore)
  - ➔Il valore restituito dal servitore al cliente (direzione: servitore → cliente)

Esempio

```
int max(int x, int y)
{
   if (x > y)
      return x;
   else
      return y;
}
```

- Il simbolo **max** denota il nome della funzione
- Le variabili **x** e **y** sono i parametri della funzione
- Il valore restituito è un intero **int**

Parametri Formali e Parametri Attuali

- Parametri formali
  - ➔Sono specificati nella dichiarazione del *servitore*
  - ➔Indicano cosa il servitore si aspetta dal *cliente*
- Parametri attuali
  - ➔Sono trasmessi dal *cliente* all'atto della chiamata
  - ➔Devono corrispondere ai parametri formali in
    - ✓Numero
    - ✓Posizione
    - ✓Tipo

Esempio

servitore

```
int max(int x, int y)
{
   if (x > y)
      return x;
   else
      return y;
}
```

Parametri formali

cliente

```
main()
{
   int z = 8;
   int m;
   m = max(z, 4);
   printf("il massimo tra %d e %d è %d\n",
          z, 4, m);
}
```

Parametri formali

Esempio

servitore

```
int max(int x, int y)
{
   if (x > y)
      return x;
   else
      return y;
}
```

Parametri formali

cliente

```
main()
{
   int z = 8;
   int m1, m2;
   m1 = max(z, 4);
   m2 = max(7, z);
}
```

All'atto di questa chiamata si effettua un legame tra:
- z e x
- 4 e y

All'atto di questa chiamata si effettua un legame tra:
- 7 e x
- z e y

Information Hiding

- La struttura interna di una funzione (*corpo della funzione*) è completamente inaccessibile dall'esterno
- Così facendo si garantisce la protezione dell'informazione (*information hiding*)
- Una funzione è accessibile **solo** attraverso la sua interfaccia

Definizione di Funzione

```
<tipo valore><nome>(<parametri formali>)
{
    <corpo>
}
```

- **<parametri formali>**
  - ➔O una lista vuota (void)
  - ➔O una lista di variabili separati da ,
- **<tipo valore>**
  - ➔Deve coincidere con il tipo del valore risultato della funzione

Definizione di Funzione

```
<tipo valore><nome>(<parametri formali>)
{
    <corpo>
}
```

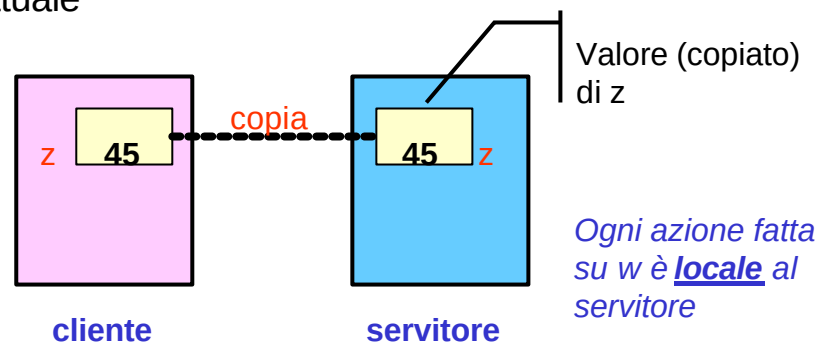
- Nella parte corpo possono essere presenti definizioni e/o dichiarazioni locali (parte dichiarazioni) e un insieme di istruzioni (parte istruzioni)
- All'interno del corpo i parametri formali vengono trattati come variabili

Passaggio dei Parametri

- In generale, un parametro può essere trasferito dal cliente al servitore
  - ➔ **Per valore o copia**
    - ✓ Si trasferisce *il valore* del parametro attuale
  - ➔ **Per riferimento**
    - ✓ Si trasferisce *un riferimento* al parametro attuale

Passaggio per Valore

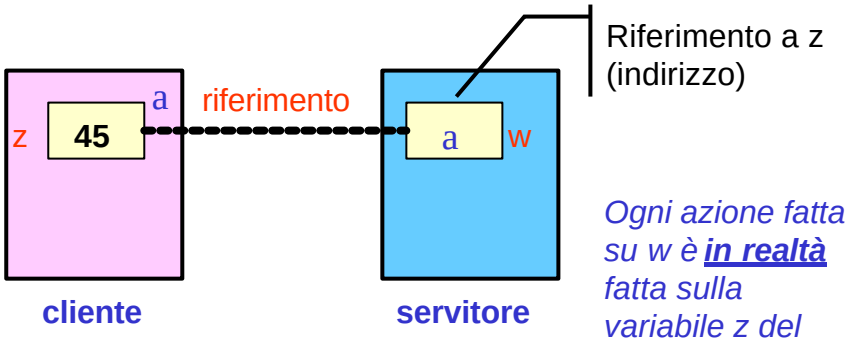
- Si trasferisce **una copia del valore** del parametro attuale



*Ogni azione fatta su w è **locale** al servitore*

Passaggio per Riferimento

- Si trasferisce **un riferimento** al parametro attuale



*Ogni azione fatta su w è **in realtà** fatta sulla variabile z del cliente!*

Passaggio dei Parametri in C

- In C, i parametri sono trasferiti sempre e solo per valore
  - ➔Si trasferisce *una copia* del parametro attuale, non l'originale!
  - ➔Tale copia è strettamente privata e locale a quel servitore
  - ➔Il servitore potrebbe quindi alterare il valore ricevuto senza che ciò abbia alcun impatto sul cliente

Passaggio dei Parametri in C

- In C, i parametri sono trasferiti sempre e solo per valore

  Conseguenza

  ➔E' impossibile usare un parametro per *trasferire informazioni verso il cliente*

  ➔Per trasferire un'informazione al cliente si sfrutta il *valore di ritorno* della funzione

Esempio: Valore Assoluto

- Definizione formale:

  ½x½:Z® N

  ½x½ vale x se x ³ 0

  ½x½ vale -x se x < 0

- Codifica sotto forma di funzione C:

```
int ValAss(int x)
{
   if (x < 0)
      return -x;
   else
      return x;
}
```

Esempio: Valore Assoluto

- **Servitore:**

```
int ValAss(int x)
{
   if (x < 0)
      x = -x;
   return x;
}
```

- **Cliente:**

```
main()
{
   int absz, z = -87;
   absz = ValAss(z);
   printf("%d", absz);
}
```

*Quando ValAss(z) viene invocata, il valore attuale di z viene **copiato** e passato a ValAss*

Esempio: Valore Assoluto

- **Servitore:**

```
int ValAss(int x)
{
   if (x < 0)
      x = -x;
   return x;
}
```

- **Cliente:**

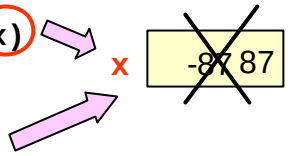
```
main()
{
   int absz, z = -87;
   absz = ValAss(z);
   printf("%d", absz);
}
```

ValAss riceve quindi una copia del valore -87 e la lega al simbolo x. Poi si valuta l'istruzione condizionale e si restituisce il valore 87.

Esempio: Valore Assoluto

- **Servitore:**

```
int ValAss(int x)
{
   if (x < 0)
      x = -x;
   return x;
}
```



- **Cliente:**

```
main()
{
   int absz, z = -87;
   absz = ValAss(z);
   printf("%d", absz);
}
```

ValAss restituisce il valore 87 che viene assegnato a absz.

NOTA: Il valore di z non viene modificato