7.8 Passaggio dei parametri

In questo capitolo abbiamo operato una distinzione tra due tipi di parametri: i *parametri formali* e i *parametri attuali*. I parametri formali sono quelli dichiarati per tipo, numero e ordine nella definizione della funzione. I parametri attuali sono invece quelli che vengono passati alla funzione all'atto della chiamata.
In C il passaggio dei parametri avviene sempre e soltanto *per valore*. Ciò significa che all'atto dell'invocazione di una funzione ogni parametro formale è inizializzato con il valore del corrispondente parametro attuale. Ecco perché deve esistere una coerenza di tipo e di numero tra parametri formali e parametri attuali. Occorre comunque chiarire che non è necessaria la *perfetta* corrispondenza. Infatti, nel trasferimento di valore da parametro attuale a parametro formale possono essere effettuate delle conversioni implicite di tipo. Per esempio, nel semplice programma:

```
main()
{
   double c;
```

```
   c = cubo( 2 );
}

double cubo(float c);
{
   return( c*c*c );
}
```

l'istruzione

```
c = cubo(2);
```

è perfettamente valida poiché la costante intera `2` viene convertita nella costante di tipo `double 2.0` (si ricordi che non esistono in C le costanti `float`).
Poiché con il passaggio dei parametri i valori dei parametri attuali sono travasati nelle locazioni di memoria corrispondenti ai parametri formali, si ha che la semantica del passaggio dei parametri è quella delle inizializzazioni di variabile: come per le inizializzazioni sono previste delle conversioni implicite di tipo. Più in dettaglio, si ha che nel passaggio dei parametri possono avvenire le conversioni seguenti.

`float` I parametri attuali `float` sono convertiti in `double` prima di essere passati alla funzione. Di conseguenza tutti i parametri formali `float` sono automaticamente trasformati in `double`.

`char` Tutti i parametri attuali `char` e `short int`, che esamineremo nei capitoli successivi, sono convertiti in `int`. Di conseguenza tutti i parametri formali `char` sono trasformati in `int`.

Occorre poi osservare che non è consentito il passaggio di parametri di tipo array, proprio perché in C il passaggio dei parametri avviene esclusivamente per valore. Infatti, se il compilatore si trovasse nella necessità di passare un array di tipo `int a[1000]`, occorrerebbe una quantità di tempo proporzionale per effettuare il travaso di valori tra due array di 1000 `int`.
Oltre al passaggio *esplicito* di parametri, è possibile anche il passaggio *implicito*. Infatti basta definire una variabile globale sia alla funzione chiamante sia a quella chiamata per ottenere la condivisione della variabile stessa. Si consideri l'esempio del Listato 7.6, in cui la variabile globale

```
char str[] = "Lupus in fabula";
```

è visibile sia dalla funzione `main` sia dalla funzione `lung_string`: quest'ultima fa riferimento a `str` per calcolarne il numero di caratteri, mentre la funzione `main` vi fa riferimento per visualizzarne il contenuto.

```
#include <stdio.h>

char str[] = "Lupus in fabula";

int lung_string(void);

main()
{
  int l;

  l = lung_string();
  printf("La stringa %s ha %d caratteri\n", str, l);
}

int lung_string(void)
{
  int i;
  for (i = 0; str[i] != '\0'; i++);
  return i;
}
```

Listato 7.6 Passaggio di parametri con variabile globale

✓ **NOTA**

Il passaggio implicito di parametri attraverso variabile globale è questione fortemente dibattuta. I dettami più severi della programmazione strutturata vorrebbero che i soli parametri passati a una funzione fossero quelli esplicitamente menzionati tra i parametri formali. Nella pratica non è tuttavia infrequente il caso di violazione di questa regola, soprattutto nelle applicazioni di tempo reale, in cui una variabile globale serve per il passaggio di dati tra due programmi (detti task) eseguiti in parallelo. Il lettore è comunque invitato a non abusare delle variabili globali: laddove è possibile è buona norma evitarle.