7.5 Visibilità

Prima di passare all'elemento sintattico "`return`", dobbiamo osservare quanto segue.
Una dichiarazione introduce un nome in un determinato ambito di definizione, detto *scope*. In altre parole, ciò significa che un nome può essere usato soltanto – o, come si usa dire, è *visibile* – in una specifica parte del testo del programma.
Per un nome dichiarato all'interno del blocco istruzioni di una funzione (nome *locale*), la visibilità si estende dal punto di dichiarazione alla fine del blocco in cui esso è contenuto.
Per un nome definito al di fuori di una funzione (nome *globale*) la visibilità si estende dal punto di dichiarazione alla fine del file in cui è contenuta la dichiarazione. Così, per esempio, in

```
int x;

f()
{
   int y;

   y = 1;
}
```

la visibilità della variabile `y` si estende dal punto di definizione sino alla fine del blocco di appartenenza.
Anche i parametri formali di una funzione hanno un campo di visibilità che si estende dall'inizio alla fine del blocco istruzioni della funzione; sono quindi considerati a tutti gli effetti variabili locali alla funzione:

```
int x;

g(int y, char z)
{
   int k;
   int l;
   ...
}
```

Le variabili `y` e `z` sono locali alla funzione `g` e hanno una visibilità che si estende dalla parentesi graffa aperta `{` alla corrispondente parentesi graffa chiusa `}`. Quindi la definizione di `y` e `z` precede all'interno del blocco la definizione delle altre variabili locali `k` e `l` aventi anch'esse una visibilità che va dal punto di definizione alla fine del blocco ▬. Per questo motivo la funzione:

```
f(int x)
{
   int x;
}
```

è errata: in essa si tenta di definire due volte la variabile locale `x` nello stesso blocco.
Una dichiarazione di un nome in un blocco può nascondere, o come si dice in gergo, *mascherare*, la dichiarazione dello stesso nome in un blocco più esterno o la dichiarazione dello stesso nome globale. Un nome ridefinito all'interno di un blocco nasconde il significato precedente di quel nome, significato che verrà ripristinato all'uscita del blocco di appartenenza (Listato 7.3) ▬.

```
int x;             /* nome globale */

f()
{
  int x;           /* x locale che nasconde x globale */
  x = 1;           /* assegna 1 a x locale */
  {
    int x;         /* nasconde il primo x locale */
    x = 2;         /* assegna 2 al secondo x locale */
  }
  x = 3;           /* assegna 3 al primo x locale */
}

scanf ("%d", &x); /* inserisce un dato in x globale */
```

Listato 7.3 Esemplificazione di mascheramento dei nomi

Nell'esempio abbiamo tre diverse variabili `x`: la prima, definita al di fuori di qualunque blocco, è la variabile `x` globale, che in generale è visibile in tutto il file, ma viene mascherata dalla `x` locale, definita nel blocco più esterno della funzione `f()`. A sua volta questa `x` locale è nascosta dalla `x` del secondo blocco di `f()` interno al primo. All'uscita del blocco interno `x` denota il primo `x`, locale al blocco esterno, e all'uscita del blocco esterno `x` designa la variabile `x` globale.

✓ NOTA
È inevitabile che in un programma avvenga il mascheramento di nomi e non è infrequente il caso in cui il programmatore non si accorge di aver mascherato un nome all'interno di un blocco. È allora consigliato identificare le variabili globali con dei nomi caratteristici e univoci: usare per variabili globali nomi del tipo `i`, `j` oppure `x` significa rischiare mascheramenti indesiderati.