7.6 return

A ogni funzione C è associato un tipo, scelto tra quelli fondamentali o derivati, che caratterizza un valore. Questo valore è detto *valore di ritorno* della funzione ed è restituito dalla funzione al programma chiamante per mezzo dell'istruzione `return`. La sintassi da usare è:

```
return (espressione);
```

oppure:

```
return espressione;
```

Quando all'interno di un blocco di una funzione si incontra una istruzione `return`, il controllo viene restituito al programma chiamante, insieme a *espressione*. Per esempio, la funzione `cubo()`

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

restituisce il controllo al programma chiamante e ritorna il cubo di `c` per mezzo dell'istruzione

```
return (c*c*c);
```

All'interno del blocco istruzioni di una funzione si possono avere più istruzioni `return`. Nella funzione `pote`:

```
double pote( b, e)
float b;
int   e;
{
  switch (e) {
    case   0: return 1;
    case   1: return b;
    case   2: return quad(b);
    case   3: return cubo(b);
    case   4: return quar(b);
    case   5: return quin(b);
    default : return -1;
  }
}
```

a ogni scelta del costrutto `switch-case` corrisponde un'uscita e la restituzione di un diverso valore. In questo caso abbiamo usato la forma sintattica del `return` non inserendo l'espressione di ritorno tra parentesi tonde.
L'invocazione di una funzione, detta anche *chiamata di funzione* (Paragrafo 7.8), può stare alla destra dell'operatore di assegnamento, salvo il caso in cui il tipo sia `void`. Nella funzione `cubo` del Listato 7.1, per esempio:

```
b = cubo(a);
```

alla variabile `b` viene assegnato il valore restituito dalla funzione `cubo`. Naturalmente il valore di ritorno può essere utilizzato all'interno di un'espressione:

```
y = a * cubo(x) + b * quad(x) + c * x + d;
```

Nel caso in cui non sia esplicitamente definito un tipo di ritorno il linguaggio C assume che esso sia il tipo fondamentale `int`.