7.4 Definizione di una funzione

In termini generali una funzione viene definita con la sintassi *prototyping* nel seguente modo:

```
tipo_ritorno nome_funz (tipo_par1 par1, ..., tipo_parN parN)
{
...
}
```

La definizione stabilisce il nome della funzione, i valori in ingresso su cui agisce – detti *parametri formali* –, il blocco di istruzioni che ne costituiscono il contenuto, e l'eventuale valore di ritorno. Per i nomi delle funzioni valgono le consuete regole in uso per gli identificatori. Nelle parentesi tonde che seguono il nome della funzione sono definiti i parametri formali specificandone il tipo e il nome.
Per ogni funzione introdotta nel programma occorre una definizione, ma si ricordi che in C non è ammesso che più funzioni abbiano lo stesso nome. Per esempio, le due definizioni, poste in uno stesso programma:

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

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

darebbero luogo a un errore pur avendo parametri diversi e ritornando valori di tipo diverso.
Nel blocco istruzioni delimitato da parentesi graffe può essere inserita qualunque istruzione, compresa una chiamata di funzione.
Studiamo ora il Listato 7.2.

```
#include <stdio.h>

double quad(float);
double cubo(float);
double quar(float);
double quin(float);
double pote(float, int);

main()
{
```

```
  int    base, esponente;
  double ptnz;

  printf(" Inserire base: " );
  scanf("%d", &base);
  printf(" Inserire esponente (0-5): ");
  scanf("%d", &esponente);

  ptnz = pote( base, esponente);

  if (ptnz == -1)
    printf("Potenza non prevista\n");
  else
    printf("La potenza %d di %d e' %f \n", esponente, base, ptnz);
}

double quad(float c)
{
  return(c*c);
}

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

double quar(float c)
{
  return(c*c*c*c);
}

double quin(float c)
{
  return(c*c*c*c*c);
}

double pote(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);
  }
}
```

Listato 7.2 Dichiarazioni e definizioni di funzioni

La funzione `main` richiama la funzione potenza `pote` passando a essa due parametri attuali: `base` ed `esponente` (Figura 7.1).

```
    ptnz = pote(base, esponente);
```

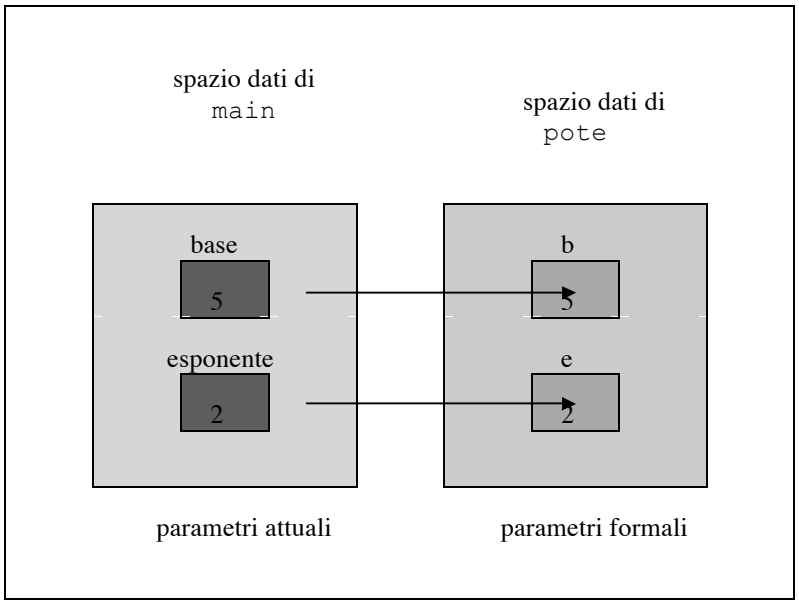

Figura 7.3 Passaggio di parametri tra `main` e `pote`

La funzione `pote`, che riceve in ingresso i valori nei parametri `b` ed `e` corrispondenti rispettivamente a `base` ed `esponente`, valuta il valore dell'esponente; dopo di ciò effettua una delle seguenti azioni: restituisce il valore 1 per esponente 0, la base stessa `b` per esponente 1, invoca la funzione `quad` per esponente 2, `cubo` per esponente 3, `quar` per esponente 4, `quin` per esponente 5 oppure restituisce –1 per segnalare la non disponibilità della potenza richiesta. Se l'esponente è 2, viene dunque invocata la funzione `quad`, cui la funzione `pote` trasmette il parametro attuale `b`:

```
case   2: return quad( b );
```

`quad` lo riceve in ingresso nel parametro formale `c` (Figura 7.2).

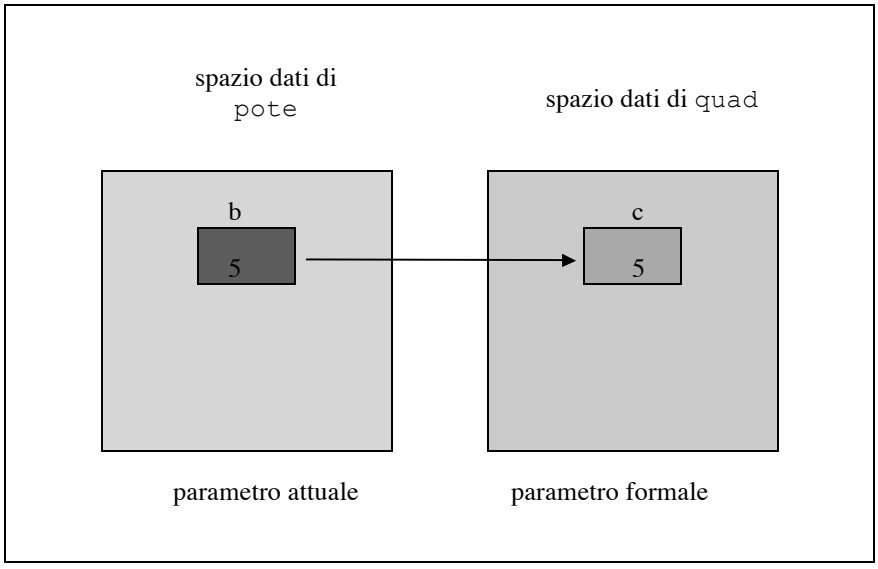

Figura 7.4 Passaggio di parametri tra `pote` e `quad`

La funzione `quad` calcola il quadrato di `c` e restituisce il risultato a `pote`, la quale a sua volta lo restituisce a `main` che l'aveva invocata (Figura 7.3). Il `main` gestisce tramite `if` il ritorno del valore negativo –1, usato per segnalare la non disponibilità della potenza richiesta.

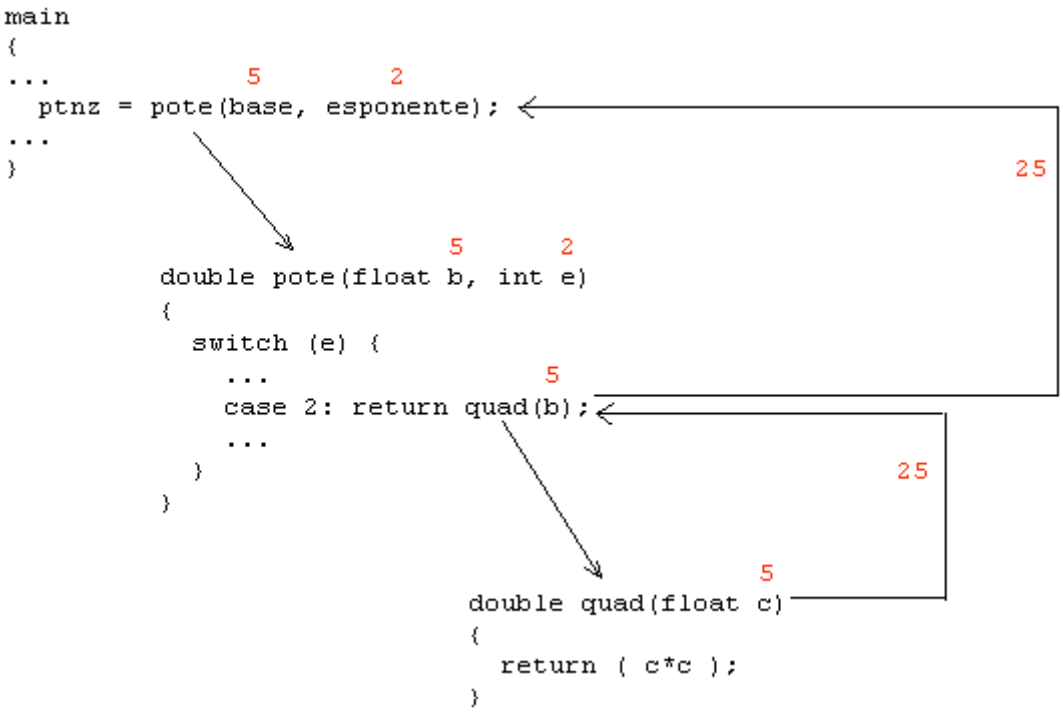

Figura 7.5 Esempio di chiamata e ritorno delle funzioni

✓ **NOTA**

In C, `main` non è una parola chiave del linguaggio ma il nome della funzione principale, cioè la funzione che, fra tutte quelle definite nel programma, viene eseguita per prima. La funzione `main` non è sintatticamente diversa dalle altre funzioni. La sua struttura:

```
main()
{
...
}
```

rispetta la sintassi generale di definizione delle funzioni. Alla funzione `main` possono essere anche passati dei parametri attraverso la linea di comando. Ritorneremo sull'argomento dopo aver trattato i puntatori ■.