12.8 Tipi derivati composti tramite funzione

Delle funzioni abbiamo parlato nei Capitoli 7 e 9. Abbiamo trattato il caso di funzioni che accettano i puntatori (a qualsiasi cosa essi puntino) e il caso di funzioni che ritornano un puntatore, come `cer_per` dell'esempio precedente. In sintesi ricordiamo che:

- gli array e le strutture debbono essere passati a una funzione tramite puntatore;
- passando un puntatore a una funzione si effettua un passaggio di parametri per indirizzo e così si risolve il caso di funzioni che devono restituire più di un valore.

Una funzione, poi, ritorna un puntatore tipicamente in seguito a operazioni di ricerca, come speriamo di aver dimostrato con gli esempi. Esiste però un ultimo caso molto importante, che finora non è mai stato trattato: il *puntatore a funzione*.

In C è possibile definire anche un tipo derivato composto "puntatore a funzione":

```
int (*pf)(float);
```

Applicando rigorosamente le regole di interpretazione di una dichiarazione si ha che l'identificatore `pf` è un puntatore a una funzione che ritorna un `int` e ha un solo parametro di tipo `float`. In generale la sintassi di dichiarazione di un puntatore a funzione è

`tipo_funzione (*nome_puntatore_funzione)(tipo_parametri);`

dove *`tipo_funzione`* è il tipo della funzione che agisce sui parametri il cui tipo è *`tipo_parametri`* ed è puntata da *`nome_puntatore_funzione`*.

A che cosa serve un puntatore di funzione? Prima di tentare una risposta si consideri che accedere al contenuto di un puntatore funzione tramite il consueto operatore `*` equivale a invocare la funzione:

```
double risultato;
     float base = 3.0;
     double (*pf)(float);
     double cubo(float);

     pf = cubo; /* inizializzazione del puntatore */
     risultato = (*pf)(base);
     printf("Il cubo di %f è %f", base, risultato)
```

L'istruzione

```
risultato = (*pf)(base);
```

è perfettamente equivalente a

```
risultato = cubo(base);
```

Vediamo ora con il Listato 12.3 un esempio completo volto a chiarire l'importanza dei puntatori di funzione.

```
#include <stdio.h>

void dummy(void), fun1(void), fun2(void), fun3(void);

struct voce_menu {
  char *msg;           /* prompt di voce di menu */
  void (*fun)(void);   /* funzione da innescare */
};

/* inizializza in vettore di strutture menu
assegnando il messaggio della voce di menu e
la relativa funzione */

struct voce_menu menu[] = {
  "1. Funzione fun1\n", fun1,
  "2. Funzione fun2\n", fun2,
  "3. Funzione fun3\n", fun3,
  "0. Fine\n", NULL, NULL , NULL
```

```
};

void main()
{
int scelta;
struct voce_menu *p;
int loop = 0;

while(loop==0) {
  for(p=menu; p->msg!=NULL; p++) /* presentazione del menu */
    printf("%s", p->msg);

  printf("\n\nScegliere l'opzione desiderata: ");
  scanf("%d", &scelta);

  if(scelta==0)      /*uscita programma */
    loop = 1;
  else
    /* esecuzione della funzione associata alla scelta */
    (*menu[scelta-1].fun)();
}
}

void fun1(void)
{printf("\n\n Sto eseguendo fun1\n\n\n");}

void fun2(void)
{printf("\n\n Sto eseguendo fun2\n\n\n");}

void fun3(void)
{printf("\n\n Sto eseguendo fun3\n\n\n");}
```

Listato 12.3 Puntatori di funzione per la creazione di un menu

Questo semplice programma presenta il seguente menu:

```
1. Funzione fun1
     2. Funzione fun2
     3. Funzione fun3
     0. Fine

     Scegliere l'opzione desiderata:
```

Scegliendo una qualsiasi delle funzioni da 1 a 3 per mezzo dell'istruzione

```
(*menu[scelta -1].fun)();
```

si lancia il corrispondente programma. Per esempio, con

```
scelta = 1;
```

si raggiunge l'elemento

```
menu[0].fun
```

il cui contenuto è `fun1`. In conseguenza sul video si ottiene il messaggio:

```
Sto eseguendo fun1
```

Il programma del Listato 12.3, oltre a esemplificare l'uso dei puntatori a funzione, è anche un esempio di come attraverso le strutture sia possibile rappresentare qualsiasi oggetto, compreso un menu.