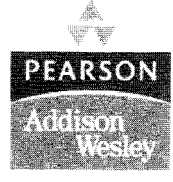
Al Kelley
Ira Pohl

C
Didattica e programmazione

Quarta edizione



Modalità di chiamata di funzione

1. Viene valutata ogni espressione nella lista dei parametri.
2. Se necessario, il valore dell'espressione viene convertito al tipo del parametro formale; tale valore viene assegnato al parametro formale corrispondente all'inizio del corpo della funzione.
3. Viene eseguito il corpo della funzione.
4. Se viene eseguita un'istruzione `return`, il controllo ritorna all'ambiente chiamante.
5. Se l'istruzione `return` contiene un'espressione, il valore di tale espressione viene convertito, se necessario, al tipo dato dallo specificatore di tipo della funzione, e il suo valore è restituito all'ambiente chiamante.
6. Se l'istruzione `return` non contiene un'espressione, all'ambiente chiamante non viene restituito alcun valore utile.
7. Se non è presente alcuna istruzione `return`, quando viene raggiunta la fine del corpo della funzione il controllo ritorna all'ambiente chiamante. Non viene restituito alcun valore utile.
8. Tutti gli argomenti vengono passati "per valore".

5.8 Costruzione di programmi di grandi dimensioni

Solitamente, un programma di grandi dimensioni viene scritto in una directory apposita sotto forma di un insieme di file *.h* e *.c*, dove ogni file *.c* contiene una o più definizioni di funzione.

Ogni file *.c* può essere ricompilato solo se è necessario, risparmiando tempo sia al programmatore che alla macchina. Questo aspetto viene approfondito nel Paragrafo 11.17, trattando le librerie e l'utilizzo di *make*.

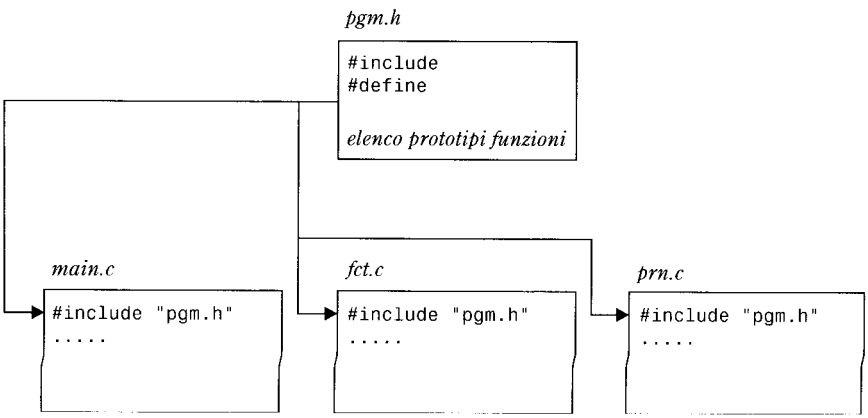
Si supponga di sviluppare un programma di grandi dimensioni di nome *pgm*. All'inizio di ogni file *.c* viene posta la riga:

```
#include "pgm.h"
```

Incontrando questa direttiva, il preprocessore cerca nella directory corrente il file *pgm.h*. Se tale file viene trovato, esso viene incluso, altrimenti la ricerca prosegue in altri punti dipendenti dal sistema. Se il file *pgm.h* non viene trovato, allora il preprocessore stampa un messaggio di errore e provoca l'interruzione della compilazione.

Il file d'intestazione *pgm.h* può contenere `#include`, `#define`, schemi di tipi enumerativi, schemi di strutture e unioni, altri costrutti di programmazione e infine un elenco di prototipi di funzione. Dunque *pgm.h* contiene elementi che risultano utili in tutto il programma. Poiché il file d'intestazione *pgm.h* si trova all'inizio di ogni file *.c*, esso agisce come "collante" che lega tra loro le varie parti del programma.

Creazione di un file .h da includere in tutti i file .c



Ora viene mostrato un esempio del funzionamento di questa strategia. In un'apposita directory viene scritto un programma formato da un file *.h* e da tre file *.c*. Di solito si utilizza lo stesso nome sia per la directory che per il programma. Ecco il programma:

Nel file pgm.h:

```
#include <stdio.h>
#include <stdlib.h>

#define   N   3

void   fct1(int k);
void   fct2(void);
void   wrt_info(char *);
```

Nel file main.c:

```
#include "pgm.h"

int main(void)
{
   char   ans;
   int    i, n = N;

   printf("%s",
      "This program does not do very much.\n"
      "Do you want more information?  ");
   scanf(" %c", &ans);
```

```
    if (ans == 'y' || ans == 'Y')
        wrt_info("pgm");
    for (i = 0; i < n; ++i)
        fct1(i);
    printf("Bye!\n");
    return 0;
}
```

Nel file fct.c:

```
#include "pgm.h"

void fct1(int n)
{
    int   i;

    printf("Hello from fct1()\n");
    for (i = 0; i < n; ++i)
        fct2();
}

void fct2(void)
{
    printf("Hello from fct2()\n");
}
```

Nel file wrt.c:

```
#include "pgm.h"

void wrt_info(char *pgm_name) {
    printf("Usage:  %s\n\n", pgm_name);
    printf("%s\n",
        "This program illustrates how one can write a program\n"
        "in more than one file.  In this example, we have a\n"
        "single .h file that gets included at the top of our\n"
        "three .c files.  Thus the .h file acts as the \"glue\"\n"
        "that binds the program together.\n"
        "\n"
        "Note that the functions fct1() and fct2() when called\n"
        "only say \"hello.\"  When writing a serious program,\n"
        "the programmer sometimes does this in a first working\n"
        "version of the code.\n"); }
```

Si noti l'impiego nel programma del tipo `char *` (puntatore a `char`; i puntatori vengono discussi nel Paragrafo 6.2).

Il programma viene compilato con il seguente comando:

```
cc -o pgm main.c fct.c prn.c
```

Il compilatore crea un file eseguibile di nome *pgm* con tre file *.o* corrispondenti ai file *.c* (in MS-DOS l'estensione è *.obj*). I file *.o* sono chiamati file *oggetto* (per un'ulteriore discussione sui file oggetto e sul loro utilizzo da parte del compilatore vedi Paragrafo 11.13).

Per un esempio più interessante di programma scritto in file separati, si veda il Paragrafo 7.6. Altri esempi sono disponibili all'indirizzo:

```
www.awprofessional.com/title/0201183994
```

Ottenuta la connessione si può scaricare il codice sorgente facendo clic su `Source Code` e, successivamente, su `abc 4e code.tar`. È possibile stampare tutto ciò che interessa e, con l'aiuto di un debugger (vedi Paragrafo 11.19), muoversi nel programma.

Da cosa è costituito un programma grande

Per un individuo un programma grande è formato da poche centinaia di righe di codice. Quali righe vanno contate? Di solito tutte quelle nei file *READ_ME* (dovrebbe essercene almeno uno), nei file *.h*, nei file *.c* e nel *makefile* (discusso nel Paragrafo 11.17). In UNIX, a tale scopo può essere utilizzato il comando *wc* (word count):

```
wc READ_ME *.h *.c makefile
```

Nell'industria i programmi vengono scritti di solito da gruppi di programmatori; viene considerato grande un programma che superi le centomila righe.

Lo stile di scrittura di un programma in una directory apposita come collezione di file *.h* e *.c* si addice a ogni programma serio, sia grande che piccolo, e tutti i programmatori esperti utilizzano tale stile. Per diventare abile in questo stile, il programmatore deve conoscere l'utilizzo di *make* o di strumenti simili (vedi Paragrafo 11.17).