# 8.5 Direttive condizionali di compilazione

Una comoda funzionalità offerta dal preprocessore C è quella delle compilazioni condizionali. Ciò significa che alcune porzioni di codice possono essere selettivamente compilate, per esempio, per includere o meno personalizzazioni dell'applicativo. Questa funzionalità è usata frequentemente quando si devono fornire diverse versioni di uno stesso programma su piattaforme differenti, per esempio Unix, Windows, Macintosh.

Le compilazioni condizionali si ottengono con le direttive `#if`, `#else`, `#elif` ed `#endif`, il cui significato è molto semplice. Se l'espressione costante che segue `#if` è vera, la porzione di codice compresa tra `#if` ed `#endif` viene compilata, altrimenti sarà ignorata dal compilatore. La direttiva `#endif` viene usata per marcare la fine del blocco `#if`. La sua forma sintattica generale è:

#if *espressione-costante*
*sequenza istruzioni*
#endif

Per esempio il semplice programma seguente:

```
#include <stdio.h>

#define MAX 100

main()
{
  #if MAX>99
    printf("compilato per array maggiori di 99\n");
  #endif
}
```

mostrerebbe il messaggio argomento della `printf` poiché la costante simbolica `MAX` è maggiore di 99.

✓ **NOTA**
L'espressione che segue `#if`, se valutata, è valutata a tempo di compilazione. Ne consegue che essa deve contenere identificatori e costanti precedentemente definiti, ma non variabili. Il contenuto di una variabile può essere noto solo a tempo di esecuzione.

La direttiva `#else` ha lo stesso significato di `else` nelle istruzioni condizionali: stabilisce un'alternativa nel caso in cui `#if` sia valutato falso:

```
#include <stdio.h>

#define MAX 10

main()
{
 #if  MAX>99
   printf("compilato per array maggiori di 99\n");
 #else
   printf("compilato per array piccoli\n");
 #endif
}
```

In questo caso `MAX` è inferiore a 99, motivo per cui la porzione di codice che segue `#if` non è compilata, mentre lo è il codice che segue `#else`. In altre parole, sarà mostrato il messaggio `compilato per array piccoli`.
Si osservi come il termine `#else` venga usato sia per marcare la fine del blocco `#if` sia per segnare l'inizio del blocco `#else`. Ciò è necessario perché in una direttiva `#if` può essere presente un solo termine `#endif`.
La direttiva `#elif` significa "`else if`" ed è usata per stabilire una catena di "`if-else-if`" che realizza una compilazione condizionale multipla. Se l'espressione è vera, la sequenza di istruzioni è compilata e nessun altra sequenza `#elif` è valutata; altrimenti si controlla la prossima serie. La forma generale è:

```
#if espressione
     sequenza istruzioni
   #elif espressione 1
     sequenza istruzioni
   #elif espressione 2
     sequenza istruzione
   ...
   #elif espressione N
     sequenza istruzioni
   #endif
```

Per esempio, il seguente frammento usa il valore `COUNTRY` per definire la moneta corrente:

```
#define US 0
   #define ENGLAND 1
   #define ITALY 2

   #define COUNTRY ITALY

   #if COUNTRY==ITALY
     char moneta[]="lit";
   #elif CONTRY==US
     char moneta[]="dollar";
   #else
     char moneta[]="pound";
   #endif
```

Le direttive `#if` ed `#elif` possono essere annidate per un numero di livelli dipendente dal particolare compilatore. Le direttive `#endif`, `#else` ed `#elif` si associano con la direttiva `#if` o `#elif` più prossima. Per esempio:

```
#if MAX>100
     #if SERIAL_VERSION
       int port=198;
     #elif
       int port=200;
     #endif
   #else
```

```
  char out_buffer[100];
#endif
```

Nell'espressione di `#if` o di `#elif` si può usare l'operatore di preprocessore `defined` per determinare l'esistenza di una macro. Nella forma più generale si ha:

```
#if defined nome-macro
    sequenza istruzioni
#endif
```

Se *`nome-macro`* è stato definito, la sequenza di istruzioni viene regolarmente compilata, altrimenti è ignorata. Si veda per esempio:

```
#include <stdio.h>

    #define DEBUG

    main()
    {
     int i = 100;

     #if defined DEBUG
      printf("la variabile i vale: %d\n", i);
     #endif
    }
```

Facendo precedere la parola chiave `define` dal simbolo `!` si ottiene una compilazione condizionale nel caso in cui la macro non sia stata definita.
Altre due direttive per la compilazione condizionale sono `#ifdef`, che significa "`if defined`", e `#ifndef`, che significa "`if not defined`". La sintassi generale è:

```
#ifdef nome-macro
    sequenza istruzioni
#endif
```

Se il *`nome-macro`* è stato precedentemente definito da una `#define`, allora la sequenza di istruzioni verrà compilata.

✓ **NOTA**
Usare `#ifdef` è equivalente a usare `#if` insieme all'operatore `defined`.

La sintassi di `#ifndef` è:

```
#ifndef nome-macro
    sequenza istruzioni
#endif
```

Se il *`nome-macro`* non è stato definito da alcuna `#define`, allora la sequenza istruzioni verrà compilata. Sia `#ifdef` sia `#ifndef` possono far uso della clausola `else`, ma non di quella `#elif`. Per esempio, il programma:

```
#include <stdio.h>

    #define UGO 10

    main()
    {
     #ifdef UGO
      printf("Ciao Ugo\n");
     #else
      printf("Ciao a tutti\n");
     #endif
```

```
  #ifndef ADA
    printf("Ada non definita\n");
  #endif
}
```

una volta eseguito visualizzerà:

```
Ciao Ugo
        Ada non definita
```

Anche `#ifdef` e `#ifndef` possono essere annidati secondo le solite regole della direttiva `#if`.