Le direttive del Preprocessore

Prof. Orazio Mirabella

Direttive

- Un compilatore traduce le istruzioni di un programma sorgente in linguaggio macchina
- Talvolta è conveniente prendere coscienza dell'esistenza del compilatore per impartirgli delle direttive.
- In C è possibile inserire in un codice sorgente tali direttive, dette più propriamente *direttive del preprocessore*.
- Il preprocessore è il modulo del compilatore che scandisce per primo il codice sorgente e interpreta le direttive prima della traduzione in codice macchina.
- Le direttive del preprocessore estendono l'ambiente di programmazione del C includendo il compilatore.

Direttive

#define
#include
#error
#elif
#if
#line
#else
#ifdef
#pragma
#endif
#ifndef
#undef

Lo Standard ANSI definisce 12 direttive del preprocessore C

#define 1/2

- La direttiva #define è usata per associare una sequenza di caratteri a un identificatore. Il preprocessore, nel fare la scansione del codice sorgente, sostituisce a ogni occorrenza dell'identificatore la stringa di caratteri che vi è stata associata con la direttiva #define.

  #define *nome-macro sequenza-caratteri*
- *Nota: le direttive non sono terminate da* ***;***

Es: *#define VERO 1*
#define FALSO 0

- ***Questa direttiva fa sì che il preprocessore sostituisca a ogni occorrenza del nome VERO il numero 1 e a ogni occorrenza di FALSO il numero 0***

printf("%d %d %d", FALSO, VERO, VERO+1);

Al compilatore questa istruzione appare come:

printf("%d %d %d", 0, 1, 1+1);

#define 2/2

- La sostituzione di macro risparmia al programmatore un lavoro ripetitivo e tedioso. Se per esempio si volesse definire un messaggio di errore standard si potrebbe usare una soluzione del tipo:

  #define MYERR "standard error on input\n"

  ...

  printf(MYERR); nota: MYERR non è fra doppi apici ed è maiuscolo.
- Il preprocessore procederebbe fedelmente a sostituire ogni occorrenza di MYERR con la stringa "standard error on input\n".
- Quindi il compilatore vedrebbe una istruzione del tipo:

  printf("standard error on input\n");

Nota: non si ha sostituzione di macro se il nome di macro è usato all'interno di una stringa:

printf("MYERR"); stamperà sul monitor MYERR

#include

- La direttiva **#include**: consente di includere in un file la copia di un file specificato.
- Esistono due forme:
  - ➔#include <nome_file>

    viene utilizzata per includere file di intestazione (*header file*) della libreria standard, che sono memorizzati in directory standard (dipendenti dall'implementazione del compilatore);
  - ➔#include "nome_file"

    cerca il file nella directory corrente, altrimenti in directory standard (perciò è utilizzata per includere *header file* definiti dal programmatore).
- Es:
- #include "stdio.h"
- #include <stdio.h>

#error

- Quando il compilatore incontra la direttiva #error visualizza un messaggio di errore.
- La sintassi di questa direttiva è:

  #error *messaggio-errore*
- Il termine *messaggio-errore* non è racchiuso tra doppi apici.
- La direttiva #error è usata per la correzione degli errori (*debugging*).

Direttive condizionali di compilazione

- alcune porzioni di codice possono essere compilate selettivamente, per esempio, per includere o meno personalizzazioni.
- Questa funzionalità è usata frequentemente quando si devono fornire diverse versioni di uno stesso programma su piattaforme differenti, per esempio Unix, Windows, Macintosh.
- Direttive condizionali: **#if, #else, #elif ed #endif**.
- 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
- Sintassi:

  #if *espressione-costante*

  sequenza istruzioni

  #endif

Nota:L'espressione che segue #if, deve contenere identificatori e costanti precedentemente definiti, ma non variabili

Direttive condizionali di compilazione

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

```
#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
}
```


Direttive condizionali di compilazione

- 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

Direttive condizionali di compilazione

- Altre due direttive per la compilazione condizionale sono #ifdef, che significa “if defined”, e #ifndef, che significa “if not defined”.
- La sintassi di #ifdef 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.
- 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

#undef

- La direttiva #undef rimuove un nome di macro definito da una precedente #define.
- La sintassi è:
  #undef *nome-macro*

- Esempio:

```
#define LEN 100
#define WIDTH 100
char array[LEN][WIDTH];
#undef LEN
#undef WIDTH
/* da questo punto in poi le costanti simboliche
LEN e WIDTH non sono più definite */
```

- La direttiva #undef è usata per confinare la definizione di macro in precise porzioni di codice.

Macro predefinite

- Lo standard ANSI C specifica cinque macro predefinite:
- __LINE__
- __FILE__
- __DATE__
- __TIME__
  __STDC__
- In generale, poi, un compilatore introduce numerose altre macro, specifiche di quella particolare implementazione.
- __LINE__ corrisponde alla riga corrente.
- __FILE__ corrisponde al nome del file corrente.
- __DATE__ è una stringa costruita secondo il formato *mese/giorno/anno* che riporta la data dell'ultima compilazione del codice sorgente.
- L'ora dell'ultima compilazione invece è conservata in __TIME__ , stringa che assume la forma *ora:minuti:secondi*.
- __STDC__ contiene la costante decimale 1. Ciò significa che l'implementazione del compilatore è conforme allo standard. Se invece riporta un qualsiasi altro numero l'implementazione non è standard.