Il preprocessore

Direttiva define
Direttiva include
Direttiva if

Il preprocessore

- Il preprocessore è un processore di testi che elabora il contenuto di un file sorgente prima della compilazione vera e propria.
- Il preprocessore è un programma che opera sostituzioni tipografiche sul codice sorgente prima che tale codice venga visto dal compilatore vero e proprio.
- Anche se il lavoro del preprocessore è un'operazione formalmente distinta dalla compilazione, il preprocessore fa parte del compilatore e delle specifiche del linguaggio; ogni sorgente C viene sempre preprocessato.

Il preprocessore

- Tutte le righe nel codice sorgente che iniziano con il carattere '#' sono direttive per il preprocessore
- Tali direttive permettono, tra le altre cose di:
  - includere (fisicamente) altri file all'interno del proprio sorgente
  - ridefinire il significato degli identificatori, tramite sostituzione puramente tipografica nel codice sorgente
  - disabilitare condizionalmente parti di codice in fase di compilazione, eliminando fisicamente il testo prima che il compilatore lo veda
- Il preprocessore inoltre elimina i commenti dal codice sorgente

La direttiva define

- Viene usata per definire delle macro. Le macro sono utilizzate per sostituire del testo all'interno del programma prima della compilazione
- Dopo una definizione della forma

```
#define nome testo-da-sostituire
```

- Tutte le successive occorrenze di nome che non sono racchiuse tra doppi apici sono sostituite da testo-da-sostituire. La stringa di testo testo-da-sostituire va dallo spazio dopo nome fino alla fine della linea
  - può continuare su linee successive se l'ultimo carattere della linea è \, il quale fa ignorare il carattere di a capo al precompilatore

La direttiva define

- Per esempio, è possibile codificare con delle macro i codici di errore gestiti da un programma:

```
#define ERR_NOERROR 0
#define ERR_INVALID 1
#define ERR_NODATA 2
#define ERR_PERMISSION 3
```

- Per una convenzione universalmente accettata, le costanti definite tramite preprocessore si scrivono in maiuscolo in modo da essere subito riconoscibili leggendo il testo del programma, per non confonderle con le variabili

La direttiva define con parametri

- Le macro vengono spesso usate anche per realizzare piccole "pseudo-funzioni", sfruttando la possibilità di passare dei parametri alla macro, ad esempio:

```
#define SQUARE(X)       X*X
```

- Purtroppo tale prassi si presta ad errori molto subdoli. Nell'esempio riportato, l'errore simanifesta per esempio in "SQUARE(1+2)" che diventa "1+2*1+2" cioè 5 invece di 9, come ci si attenderebbe
- Si consiglia sempre un abbondante uso di parentesi

```
#define SQUARE(X)       ((X)*(X))
```

- "SQUARE(1+2)" diventa "((1+2)*(1+2))" come ci si attende

La direttiva define con parametri

- Quando un argomento di macro appare più di una volta nell'espansione, la macro non può essere equivalente ad una funzione perché operatori come "++" appaiono ripetuti nel testo effettivo del programma, con effetti in genere non desiderati

  SQUARE(i++) diventa ((i++)*(i++))

La direttiva include

- Il preprocessore sostituisce ogni riga della forma:

```
#include <nome-file>
```

oppure

```
#include "nome-file"
```

con il contenuto del file "nome-file"

  - Se il file incluso è specificato con le parentesi ad angolo viene cercato tra quelli di sistema
  - Se è specificato con le virgolette viene cercato prima nella directory corrente

La direttiva include

- Il file incluso può contenere qualunque porzione di codice C, comprese altre direttive #include.
  - In genere contiene direttive #define e dichiarazioni di variabili e funzioni
  - Funzioni, tipi, macro della libreria del C sono definiti in alcuni header file standard
    - es.: math.h stdio.h stdlib.h string.h
- In generale è buona regola non mettere negli header-file il codice delle funzioni, ma solo la loro dichiarazione

La direttiva if

- Si possono introdurre segmenti di codice in dipendenza da particolari condizioni.
  - Il costrutto seguente valuta una espressione intera costante, il cui valore deve essere noto all'atto della compilazione:

```
#if espressione-costante-intera
/*
 * questo codice viene considerato solo se
 * l'espressione risulta diversa da 0
 */
/*
 * endif termina la sezione condizionale
 */
#endif
```

  - Tutte le righe di codice comprese tra #if e #endif vengono incluse nel file che verrà passato al compilatore solo se l'espressione è diversa da 0

La direttiva if

- È possibile utilizzare la direttiva #if per eliminare porzioni di codice senza cancellarle, per esempio in fase di debugging:

```
#if 0
/* codice da non considerare */
#endif
```

Una volta eliminati i problemi si può velocemente ripristinare il codice sostituendo 0 con 1

```
#if 1
/* codice ripristinato */
#endif
```

Le direttive ifdef / ifndef

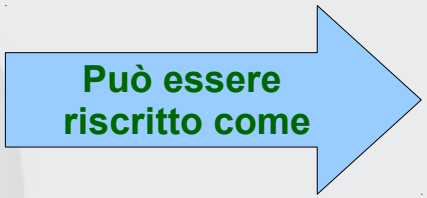
```
#ifdef MACRO
/*
 * questo codice viene considerato solo se
 * "MACRO" è già stata definita
 */
#endif

#ifndef MACRO
/*
 * questo codice viene considerato solo se
 * "MACRO" non è già stata definita
 */
#endif
```

`#ifdef MACRO` è equivalente a `#if defined(MACRO)`

La direttiva elif

```
#if CODICE == 1
/* codice 1 */
#else
if CODICE == 2
/* codice 2 */
#else
if CODICE == 3
/* codice 3 */
#endif
#endif
#endif
```




```
#if CODICE == 1
/* codice 1 */
#elif CODICE == 2
/* codice 2 */
#elif CODICE == 3
/* codice 3 */
#endif
```

Esecuzione del precompilatore

- È possibile definire una (o più) macro al momento della compilazione

```
% gcc -DMACRO file.c
```
  - Ho definito MACRO

```
% gcc -DMACRO=3 file.c
```
  - Ho definito MACRO e inizializzata a 3
- È possibile eseguire la sola precompilazione
  - Il risultato va sullo standard output

```
% gcc -E -DMACRO=3 -DMACRO2 file.c
```

Esempio di uso del preprocessore

Esempio di file header

*my-header.h (~) - gedit

Open Save Undo

*my-header.h

```
/*
 * file my-header.h
 */
#ifndef _MY_HEADER_H
#define _MY_HEADER_H
/* in questo modo il file non verrà mai
 incluso 2 volte */

#include <math.h>
#include <stdio.h>

/* se PI non è già definito recupero la definizione standard di math.h */
#ifndef PI
#define PI M_PI /* M_PI è definita in math.h, è il valore di pi greco */
#endif

#endif
```

pi.c (~) - gedit

Open Save Undo

pi.c

```
/*
 * file pi.c
 */
#include "my-header.h"

int main()
{
    printf("pi=%f\n", PI);
    return 0;
}
```

C Tab Width: 4 Ln 11, Col 1 INS

user@ubuntu: ~

```
user@ubuntu:~$ gcc -Wall pi.c
user@ubuntu:~$ ./a.out
pi=3.141593
user@ubuntu:~$ gcc -Wall -DPI=3.1415 pi.c
user@ubuntu:~$ ./a.out
pi=3.141500
user@ubuntu:~$
```