Problema delle doppie inclusioni

`foo.h`

```
struct foo {
  int bar;
};
```

`baz.h`

```
#include “foo.h”
```

`main.c`

```
#include “foo.h”
#include “baz.h”
```

Questa sequenza di include genera un errore di compilazione:

- Il compilatore non accetta la duplicazione della definizione di simboli.
- Quando incontra un `#include` il preprocessore copia il contenuto del file da includere nel file sorgete.
- In `main.c` ci troviamo quindi con una doppia definizione della struttura `foo`, che causa l’errore di compilazione.
- Risolvere a mano il grafo delle dipendenze per evitare questo problema può risultare molto complesso, soprattutto per programmi non banali.
- Fortunatamente alcune direttive per preprocessore vengono in nostro aiuto.

Direttive per il preprocessore

Le direttive per il preprocessore sono linee di codice precedute dal simbolo cancelletto (#)

- Sono utilizzate per comandare azioni al preprocessore, che processa i file sorgente prima che questo venga esaminato dal compilatore.
- Queste direttive si compongono di una sola linea di codice, e non prevedono l'uso del punto e virgola finale (;). Possono essere spezzate su più linee utilizzando backslash (\).
- Abbiamo già incontrato le direttive `#include` e `#define`.
- Altre direttive permettono di evitare il problema delle doppie inclusioni.

Include Guards

Direttive per evitare gli errori di doppia inclusione:

```
#ifndef <token>
#define <token>
// header file normale
#endif
```

- Questa sintassi prende il nome di *include guard*. <token> è un nome univoco che il programmatore deve dare ad ogni header file.
- <token> è una sequenza di caratteri senza spazi.
- In pratica questa direttiva dice al preprocessore «se <token> non è definito definiscilo e procedi normalmente fino ad `#endif`».
- Se il preprocessore incontra nuovamente `#ifndef <token>` salta tutto questo blocco in quando <token> è già stato definito.
- Il programmatore deve avere cura di create <token> univoci per ogni suo header file, per evitare di escludere erroneamente header files differenti.

Include Guards

foo.h

```
#ifndef FOO_H
#define FOO_H
struct foo {
  int bar;
};
#endif
```

baz.h

```
#include “foo.h”
```

main.c

```
#include “foo.h”
#include “baz.h”
```

Riprendiamo l’esempio utilizzando #ifndef

- Definiamo una include guard in `foo.h` utilizzando come token `FOO_H`.
- Quando il preprocessore processa `main.c`, esegue la prima include di `foo.h`. Non essendo definita `FOO_H` le istruzioni successive sono eseguite, `FOO_H` è definito ed il resto dell’header è incluso in `main.c`.
- Al secondo include il preprocessore legge il file `baz.h`, esegue l’include di `foo.h`, ma questa volta l’istruzione `#ifndef` non è verificata in quando il token `FOO_H` è già stato definito precedentemente.
- Il preprocessore salta quindi tutte le operazioni fino all’istruzione `#endif` di chiusura dell’`#ifndef`.