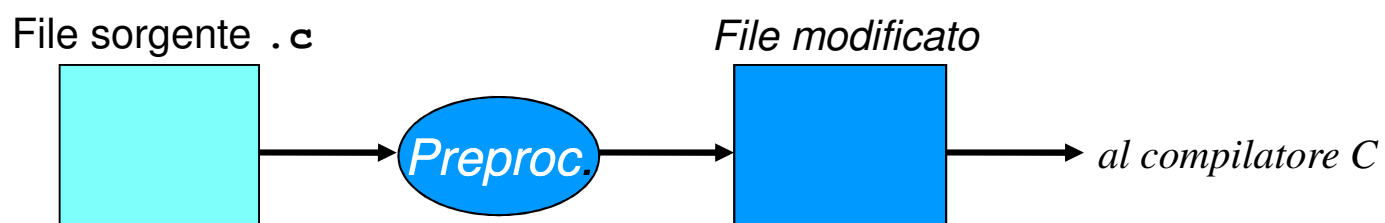


## Preprocessore C

---

- Automa che agisce prima del compilatore
- Segue delle direttive contenute nel file sorgente che sta processando
- Il compilatore riceve una versione “riveduta e corretta” del file sorgente



1

## Preprocessore C – cos'è?

---

- Il preprocessore non è un compilatore:
  - Non conosce il linguaggio C
    - Non può interpretarne le istruzioni
    - Non può verificare la correttezza del programma
- Agisce sul testo del programma
  - Potrebbe manipolare qualunque testo...
  - ...anche sorgenti di altri linguaggi!

2

## Preprocessore C – cosa può fare?

---

- includere altre porzioni di testo, prese da altri file
- effettuare *ricerche e sostituzioni* (più o meno sofisticate) sul testo
- *inserire o sopprimere parti del testo* a seconda del verificarsi di certe condizioni da noi specificate

3

## Preprocessore C – come lo fa?

---

- Il tutto è comandato da direttive contenute nel testo stesso!
- Le direttive non sono istruzioni C e non ne hanno la sintassi
  - Le direttive vengono soppresse una volta elaborate (il compilatore non le vede!)

4

## Preprocessore C - Direttive

---

- includere altre porzioni di testo  
`#include nomefile`
- effettuare *ricerche e sostituzioni*  
`#define testo1 testo2`
- *inserire o sopprimere parti del testo*  

<code>#ifdef cond</code>	<code>#ifndef cond</code>
<code>...testo...</code>	<code>...testo...</code>
<code>#endif</code>	<code>#endif</code>

5

## La direttiva `#define`

---

### Sintassi:

`#define    testo1    testo2`

### Effetto:

definisce una *regola di ricerca e sostituzione*:  
ogni occorrenza di *testo1* verrà sostituita da  
*testo2*

### Scopo:

**definire costanti simboliche** (per convenzione,  
*testo1* è maiuscolo)

6

## La direttiva #define

---

**Prima** del pre-processing:

```
#define RADICED12 1.4142F
main() {
    float lato = 18;
    float diagonale = lato * RADICED12;
}
```

**Dopo** il pre-processing:

```
main() {
    float lato = 18;
    float diagonale = lato * 1.4142F;
}
```

7

## Preprocessore C

---

Attenzione:

- nell'effettuare ricerche e sostituzioni, il preprocessore **si limita a sostituire testo con altro testo**
- **non effettua controlli di nessun tipo**, né può farli: non è un compilatore, e dunque *non conosce la sintassi del C*
- Quindi, regole sbagliate possono produrre *risultati privi di senso*

8

## La direttiva `#define` – (controesempio)

---

**Prima** del pre-processing:

```
#define RADICEDI2 1.414paperino
main() {
    float lato = 18;
    float diag = lato * RADICEDI2;
}
```

**Dopo** il pre-processing (errore sintattico):

```
main() {
    float lato = 18;
    float diag = lato * 1.414paperino;
}
```

## Le macro

---

La regola di ricerca e sostituzione introdotta dalla direttiva `#define` si chiama **macro**

Regole semplici, come le precedenti:

```
#define MAX 10
#define RADICEDIDUE 1.4142F
```

definiscono **macro semplici**

La direttiva `#define` permette però anche di definire regole più complesse, che vanno sotto il nome di **macro parametriche** (che non vedremo in questo corso)

## Preprocessore C – #include

---

### **Sintassi:**

```
#include <libreria.h>  
#include "miofile.h"
```

### **Effetto:**

include il contenuto del file specificato  
*esattamente nella posizione* in cui si trova la  
direttiva stessa.

(La differenza tra le due scritture sopra verrà  
discussa più avanti)