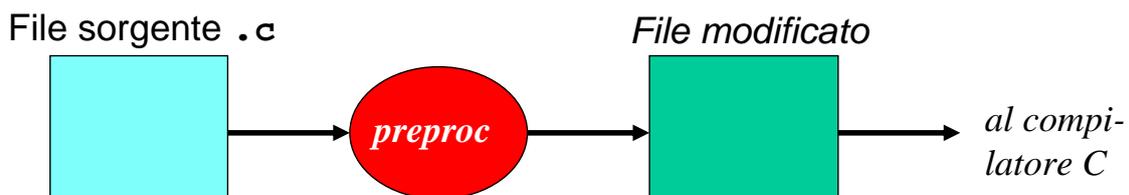


IL PREPROCESSORE C

Il preprocessore **C** opera *modifiche testuali sul programma prima* che esso raggiunga il compilatore vero e proprio



1

IL PREPROCESSORE C

Che 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** specificate

2

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*)

3

ESEMPIO

Prima del pre-processing:

```
#define RADICEDI2 1.4142F
main() {
    float lato = 18;
    float diagonale = lato * RADICEDI2;
}
```

Dopo il pre-processing:

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

4

IL 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*

5

CONTRO-ESEMPIO

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*)