COSTRUZIONE DI UN'APPLICAZIONE

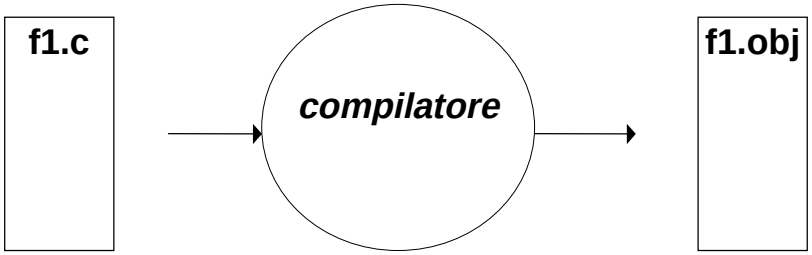
Per costruire un'applicazione occorre:

- **compilare il file (o *i* file se più d'uno) che contengono il testo del programma (file *sorgente*)**
  Il risultato sono uno o più file *oggetto.*
- **collegare i file oggetto l'uno con l'altro *e con le librerie di sistema.***

COMPILAZIONE DI UN'APPLICAZIONE

1) Compilare il file (o *i* file se più d'uno) che contengono il testo del programma

- **File *sorgente*: estensione `.c`**
- **File *oggetto*: estensione `.o` o `.obj`**

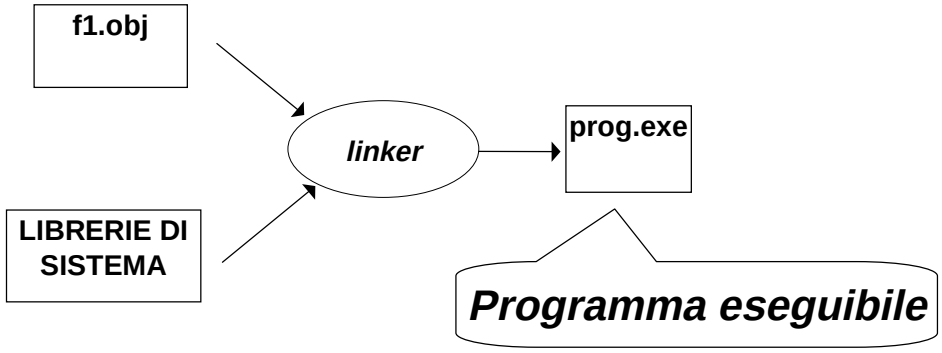


f1.obj: Una versione tradotta che però non è autonoma (e, quindi, non è direttamente eseguibile).

COLLEGAMENTO DI UN'APPLICAZIONE

2) Collegare il file (o *i* file) oggetto fra loro *e con le librerie di sistema*

- **File *oggetto*: estensione `.o` o `.obj`**
- **File *eseguibile*: estensione `.exe` o nessuna**



COLLEGAMENTO DI UN'APPLICAZIONE

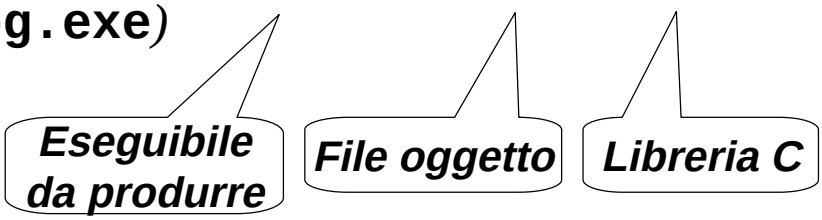
<u>LIBRERIE DI SISTEMA</u>:
insieme di componenti software che consentono di <u>interfacciarsi col sistema operativo</u>, <u>usare le risorse da esso gestite</u>, e realizzare alcune "istruzioni complesse" del linguaggio

COSTRUZIONE “MANUALE”

In passato, la costruzione si faceva “*a mano*”, attivando *compilatore* e *linker* dalla *linea di comando* del sistema operativo (DOS, Unix, ...)

```
C:\PROVA> gcc -c f1.c
```
(genera **`f1.obj`***)*

```
C:\PROVA> ld -o prog.exe f1.obj –lc
```
(genera **`prog.exe`***)*



AMBIENTI INTEGRATI

Oggi, gli *ambienti di lavoro integrati* automatizzano la procedura:

- **compilano i file sorgente (*se e quando necessario)***
- **invocano il linker per costruire l’eseguibile**

ma per farlo devono sapere:

- ***quali file sorgente* costituiscono l’applicazione**
- ***il nome dell’eseguibile* da produrre.**

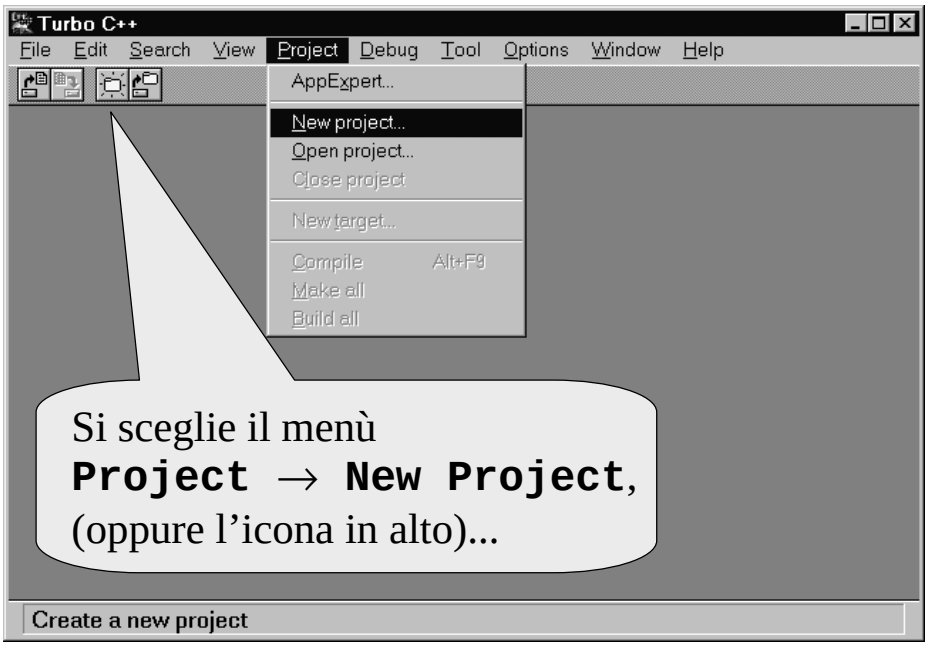
PROGETTI

È da queste esigenze che nasce il concetto di *PROGETTO*

- **un *contenitore concettuale (e fisico)***
- **che *elenca i file sorgente* in cui l'applicazione è strutturata**
- **ed eventualmente altre informazioni utili.**

Oggi, *tutti* gli ambienti di sviluppo integrati, *per qualunque linguaggio*, forniscono questo concetto e lo supportano con idonei strumenti.

PROGETTI IN TURBO C



PROGETTI IN TURBO C

La finestra che appare richiede *varie informazioni*.

1) percorso e *nome del progetto* (per default coincide con il **`Target Name`**, cioè col nome dell'eseguibile da produrre)

PROGETTI IN TURBO C

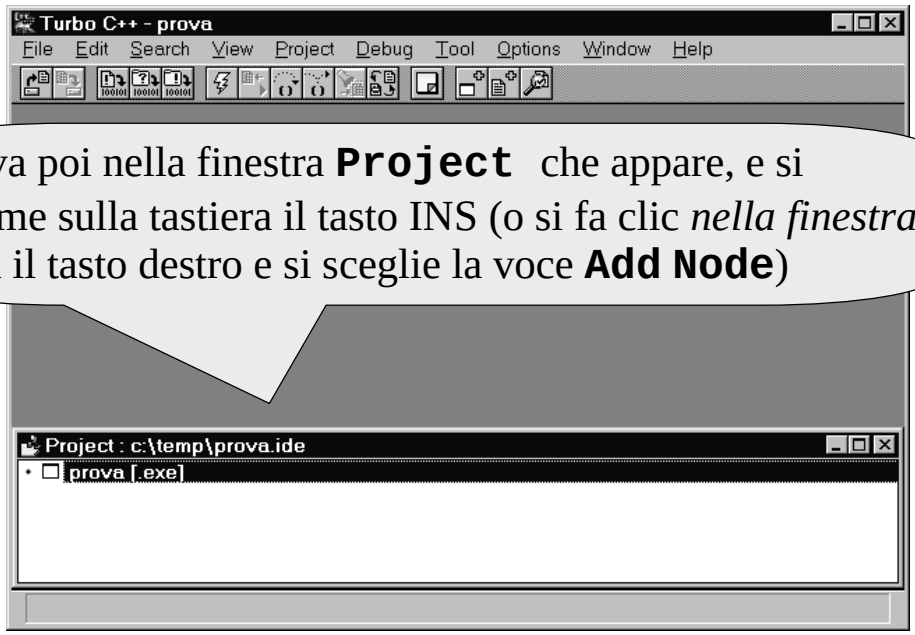
2) il **`Target Type`**, cioè il ***tipo di applicazione*** da generare: **essenziale scegliere `EasyWin`** (***non*** `Application`)

Turbo C è un ambiente ***multi-target***, cioè **può produrre sia *applicazioni per Windows*** (che vanno strutturate in modo particolare), **sia *applicazioni standard*** *da eseguire in una finestra di Windows emulando una console.*

PROGETTI IN TUR

3) premendo il tasto **`Advanced`**:
un *nodo iniziale* `.c`
nessuna risorsa
(né `.rc`, né `.def`)

PROGETTI IN TURBO C



Si va poi nella finestra **`Project`** che appare, e si preme sulla tastiera il tasto INS (o si fa clic *nella finestra* con il tasto destro e si sceglie la voce **`Add Node`**)

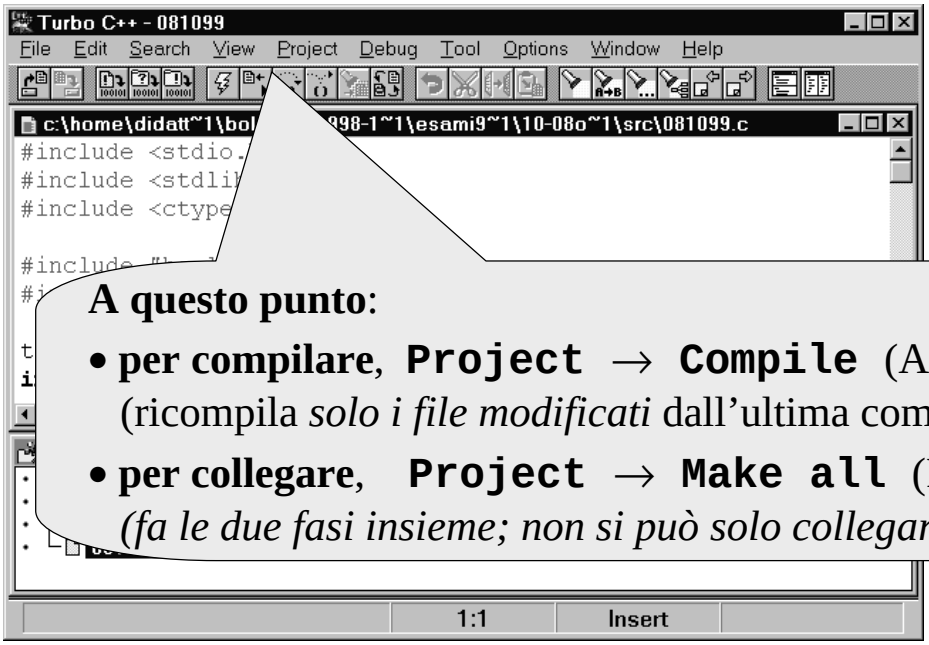
PROGETTI IN TURBO C

Dalla finestra **`Add to Project List`** si selezionano i file da inserire nel progetto.

PROGETTI IN TURBO C

Facendo doppio clic su un nome di file (ad esempio, **`aaa.c`**) si apre l'editor per modificarlo.

PROGETTI IN TURBO C



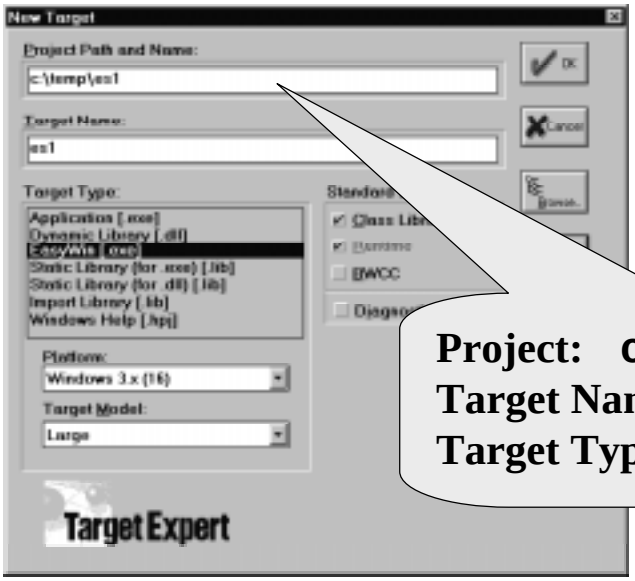
A questo punto:

- **per compilare**, **`Project` → `Compile`** (ALT+F9)
  (ricompila *solo i file modificati* dall'ultima compilazione)
- **per collegare**, **`Project` → `Make all`** (F9)
  (fa le due fasi insieme; non si può solo collegare)

UN PRIMO ESEMPIO

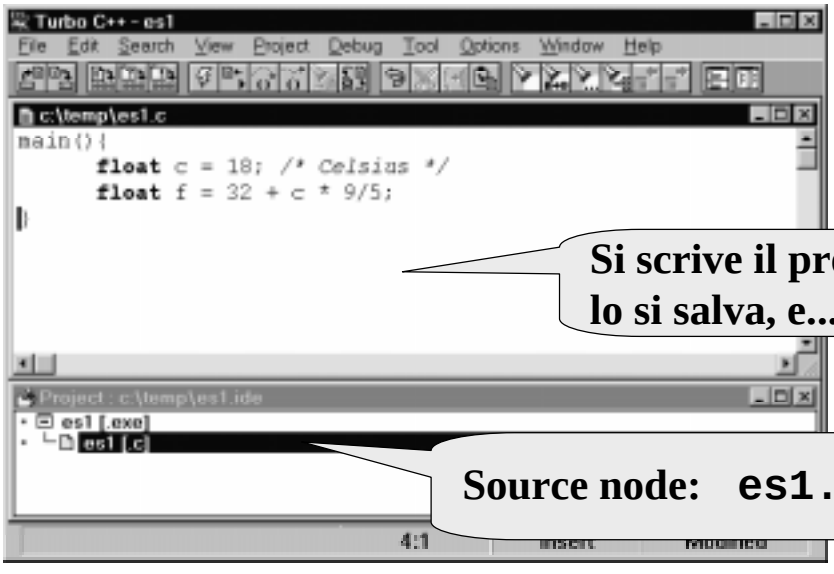
```
main(){
    float c = 18; /* Celsius */
    float f = 32 + c * 9/5;
 }
```

L'ESEMPIO "PASSO PASSO"



Project: `c:\temp\es1`
Target Name: `es1`
Target Type: `EasyWin`

ESEMPIO: Fahrenheit / Celsius



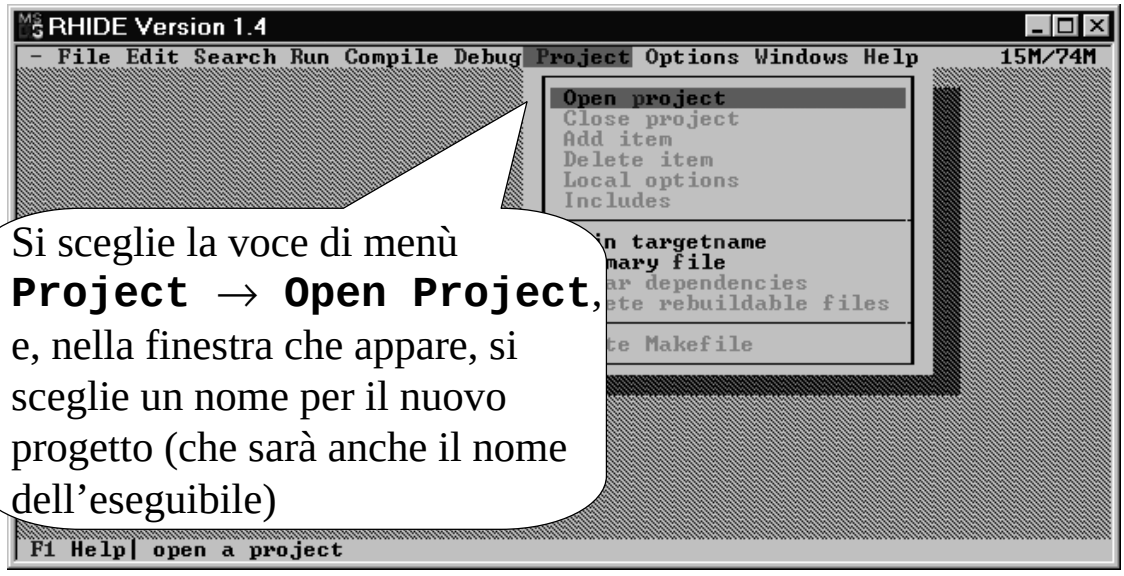
Si scrive il programma, lo si salva, e...

Source node: `es1.c`

ESEMPIO: Fahrenheit / Celsius

… lo si costruisce (F9).
Ora abbiamo un eseguibile

PROGETTI IN DJGPP/RHide

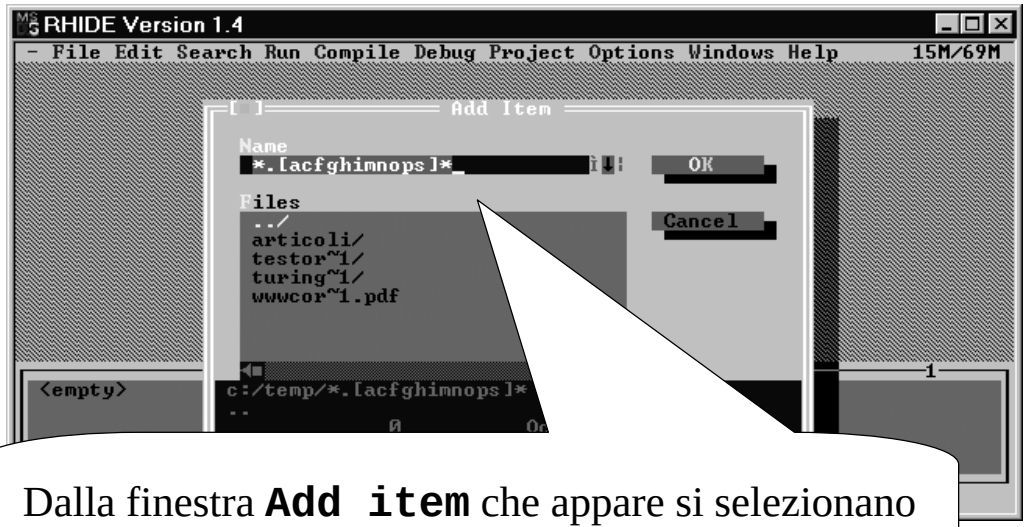


Si sceglie la voce di menù **`Project` → `Open Project`**, e, nella finestra che appare, si sceglie un nome per il nuovo progetto (che sarà anche il nome dell'eseguibile)

PROGETTI IN DJGPP/RHide

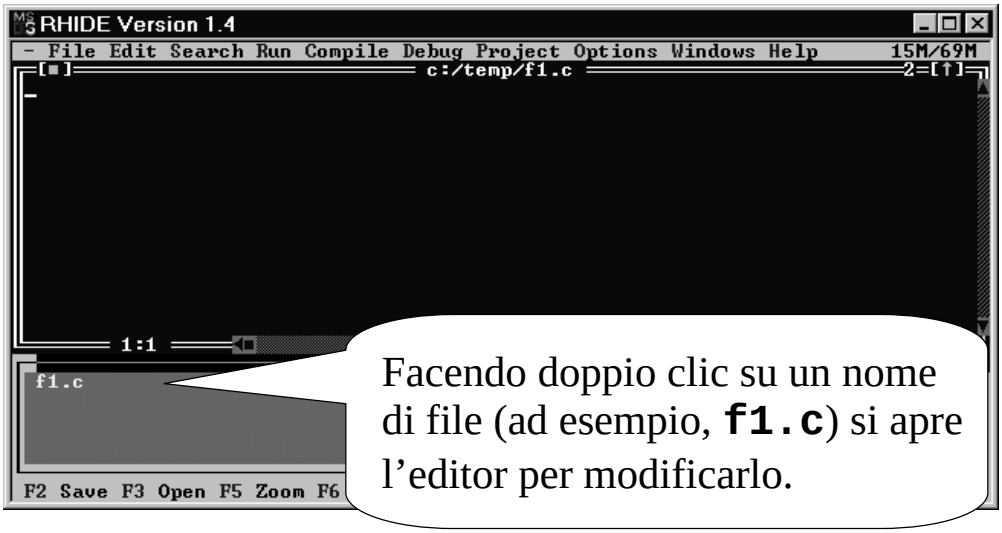
Si va poi nella **`Project Window`** che appare, e si preme sulla tastiera il tasto INS (o si sceglie da menù l'opzione **`Project` → `add item`**)

PROGETTI IN DJGPP/RHide



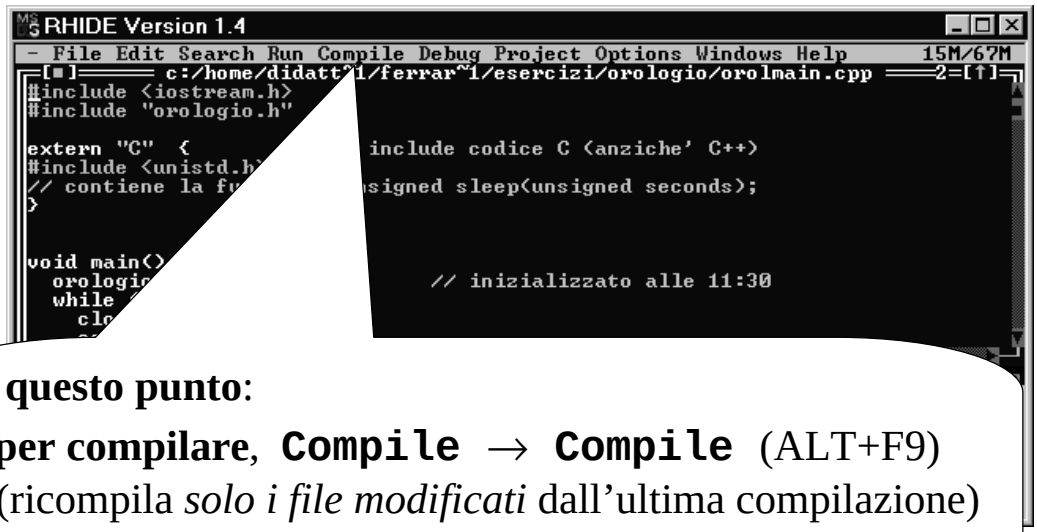
Dalla finestra **`Add item`** che appare si selezionano i file sorgente che si vogliono inserire nel progetto.

PROGETTI IN DJGPP/RHide



Facendo doppio clic su un nome di file (ad esempio, **`f1.c`**) si apre l'editor per modificarlo.

PROGETTI IN DJGPP/RHide



A questo punto:

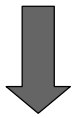
- **per compilare**, **`Compile`** → **`Compile`** (ALT+F9) (ricompila *solo i file modificati* dall'ultima compilazione)
- **per collegare**, **`Compile`** → **`Link`**
- **oppure le due fasi insieme: `Compile`** → **`Make`** **(F9)**

IL DEBUGGER

Una volta scritto, compilato e collegato il programma (ossia, costruito l'eseguibile)

occorre uno strumento che consenta di

- **eseguire il programma *passo per passo***
- ***vedendo le variabili* e la loro evoluzione**
- **e *seguendo le funzioni* via via chiamate.**



Debugger

ESEGUIRE IL PROGRAMMA

Sia Rhide sia TURBO C incorporano un debugger con cui eseguire il programma

- ***riga per riga***
  - **entrando anche dentro alle funzioni chiamate (*Run* → *trace into*; F7) (*Debug* → *trace into*; F7)**
  - **oppure considerando le chiamate di funzione come una singola operazione (*Run* → *step over*; F8) (*Debug* → *step over*; F8)**
- **oppure *fino alla riga desiderata***

  (*Run* → *Go to cursor*; F4) (F4)

ESEGUIRE IL PROGRAMMA

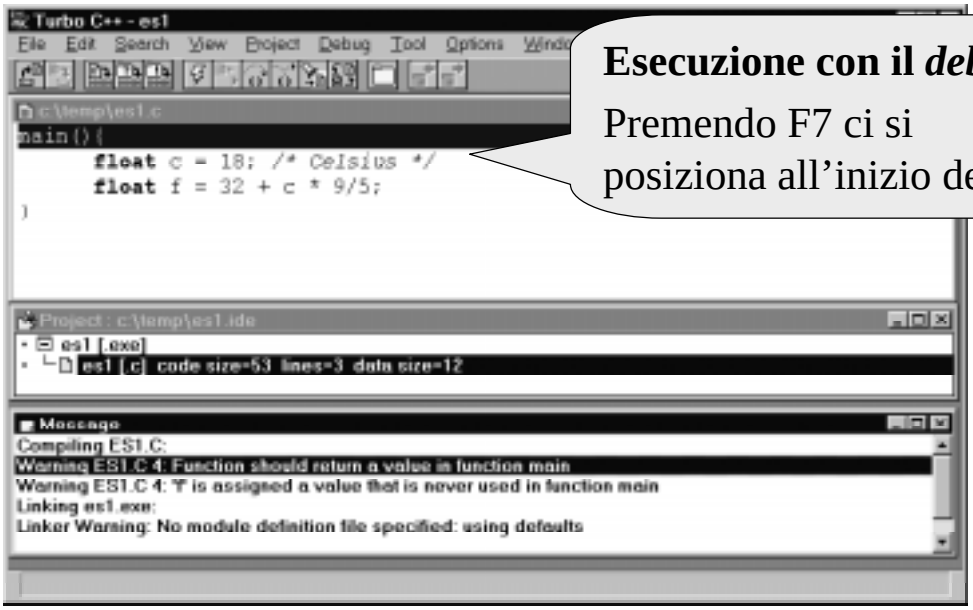
… e con cui inoltre è possibile:

- **controllare *istante per istante* quanto vale una variabile**
  - ***Debug → Watch an expression (CTRL+F7)***
  - ***Debug → Add watch (CTRL+F5)***
- **vedere *istante per istante* le funzioni attive *(record di attivazione nello stack)***
  - ***Debug → Call stack (CTRL+F3)***
  - ***View → Call stack***

In sintesi

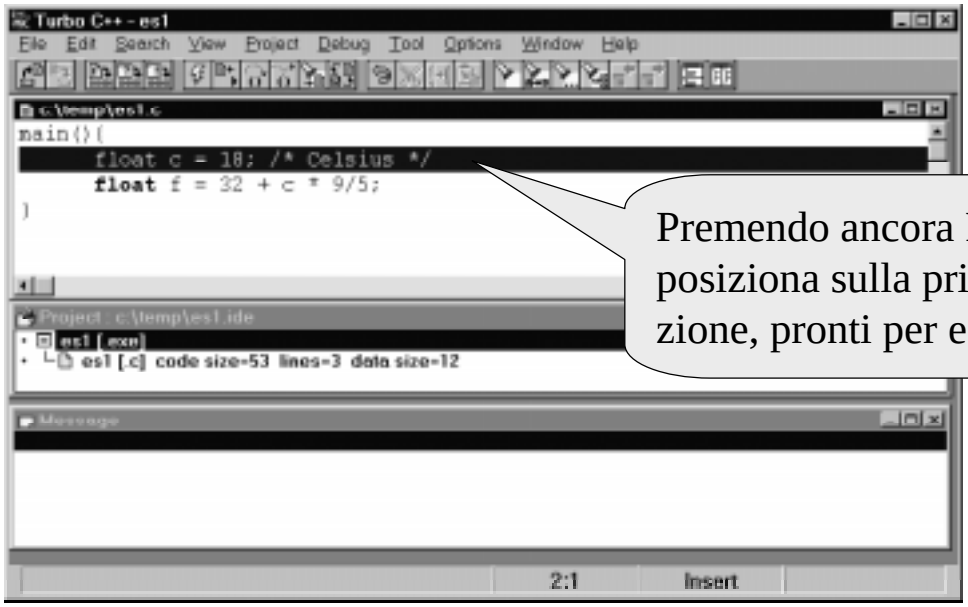
Run	**esegue il programma dalla prima istruzione all'ultima**
Step over	**esegue la funzione come istruzione singola**
Trace into	**esegue passo-passo tutte le istruzioni di una funzione**
Go to the cursor *(Rhide)* ***F4*** *(Turbo C)*	**esegue il programma fino alla posizione del cursore e poi passo-passo**
Program reset *(Rhide)* ***Terminate Program*** *(Turbo C)*	**interrompe la fase di esecuzione (in debugging)**

ESEMPIO: Fahrenheit / Celsius



Esecuzione con il *debugger*.
Premendo F7 ci si posiziona all'inizio del *main*.

ESEMPIO: Fahrenheit / Celsius



Premendo ancora F7 ci si posiziona sulla prima istruzione, pronti per eseguirla.

ESEMPIO: Fahrenheit / Celsius

Per seguire l'evoluzione è utile *mettere sotto osservazione (watch)* le due variabili **c** e **f.**

ESEMPIO: Fahrenheit / Celsius

La *finestra di watch* mostra istante per istante il valore di **c** e **f.**

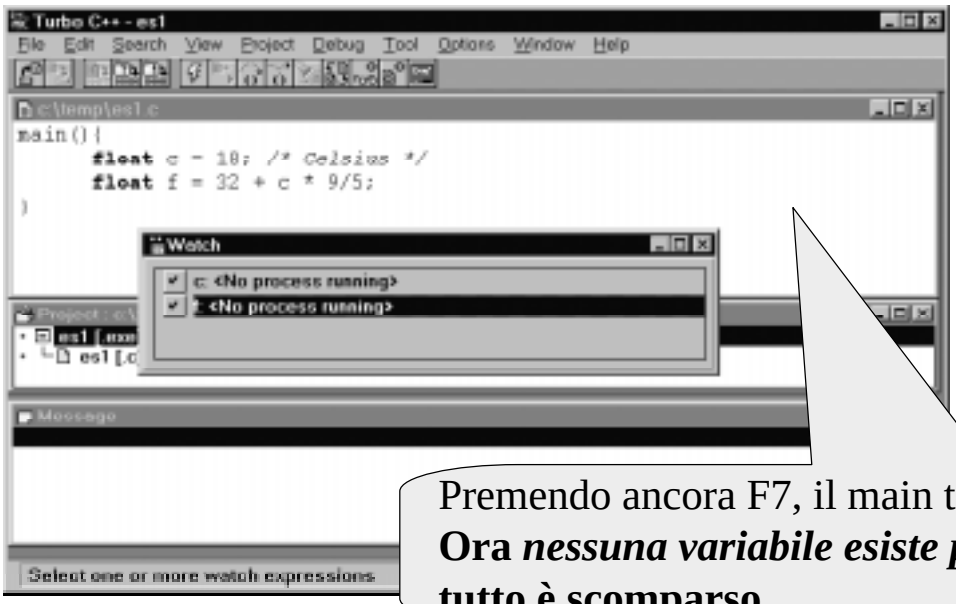
ESEMPIO: Fahrenheit / Celsius

Dopo l'esecuzione della prima riga, **c** vale **18.** Il valore di **f**, invece, è ancora *indefinito* (casuale).

ESEMPIO: Fahrenheit / Celsius

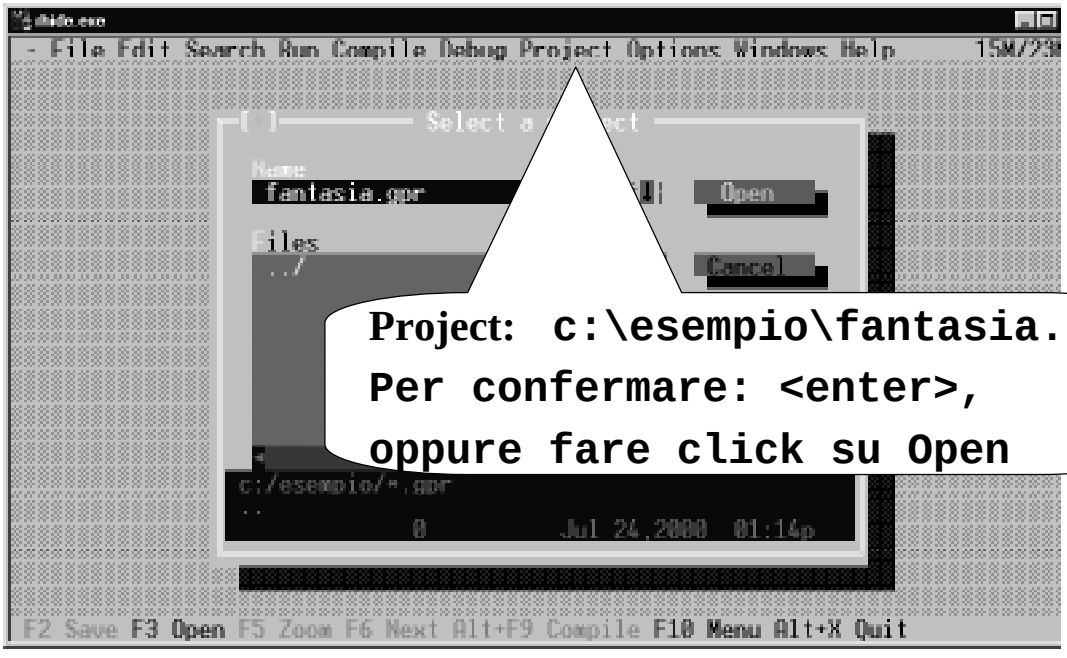
Dopo l'esecuzione della seconda riga, **f** assume il valore **64.4**.

ESEMPIO: Fahrenheit / Celsius



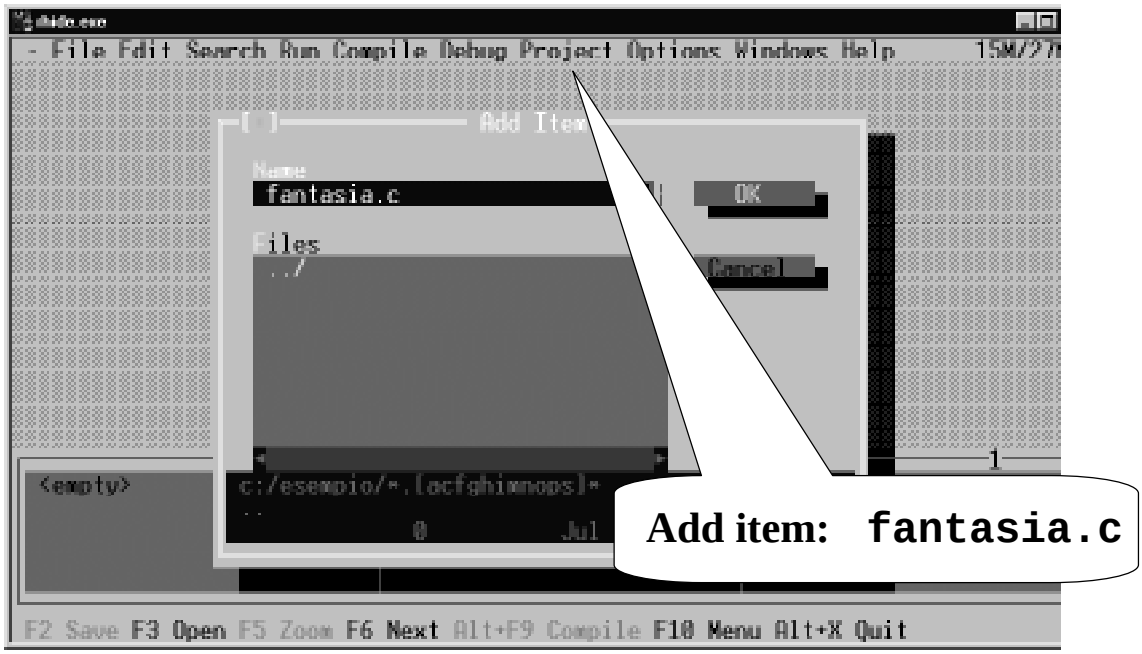
Premendo ancora F7, il main termina.
Ora *nessuna variabile esiste più:*
tutto è scomparso.

UN ESEMPIO COMPLETO in Rhide



Project: c:\esempio\fantasia.gpr
Per confermare: <enter>,
oppure fare click su Open

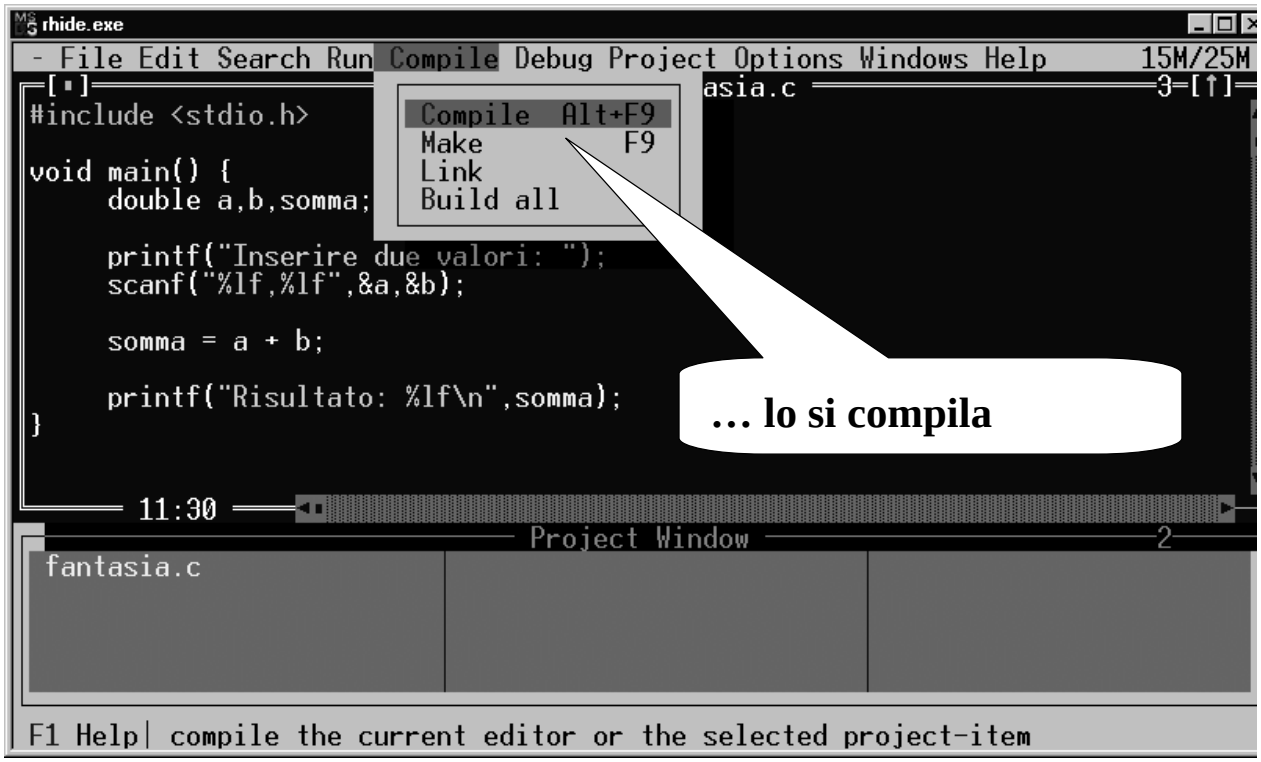
UN ESEMPIO COMPLETO



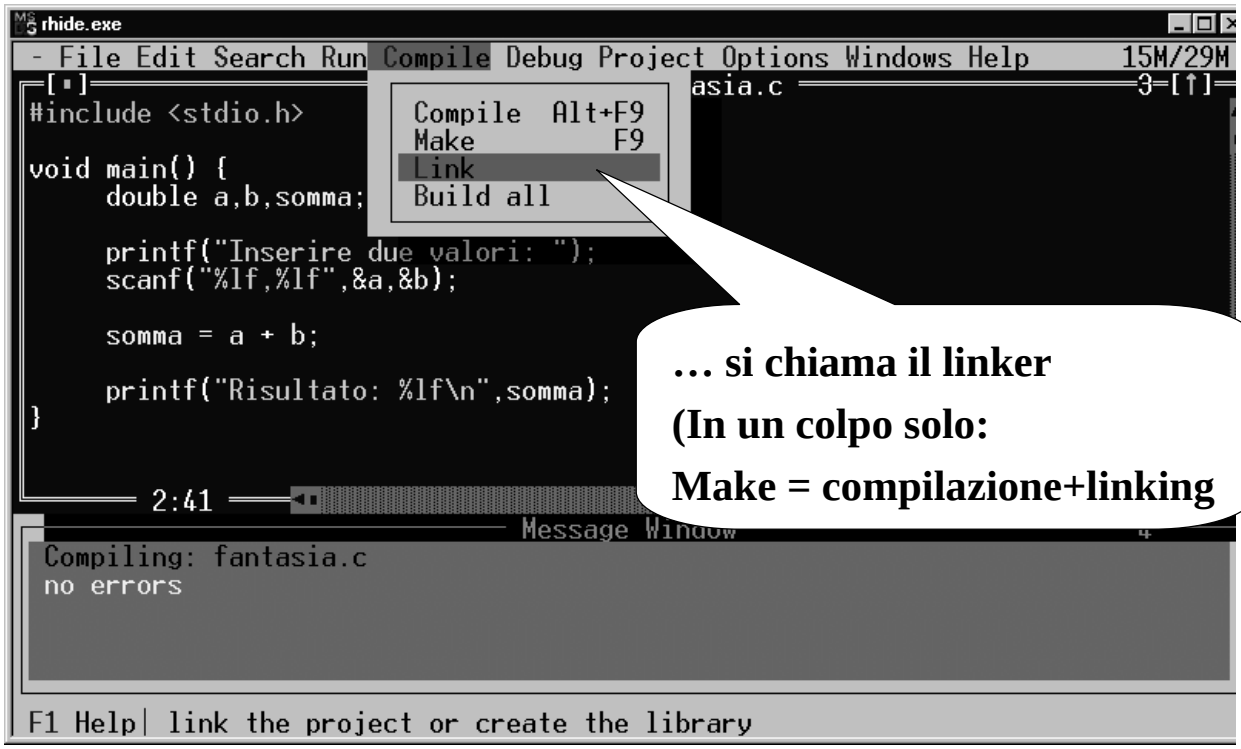
UN ESEMPIO COMPLETO

UN ESEMPIO COMPLETO

UN ESEMPIO COMPLETO



UN ESEMPIO COMPLETO

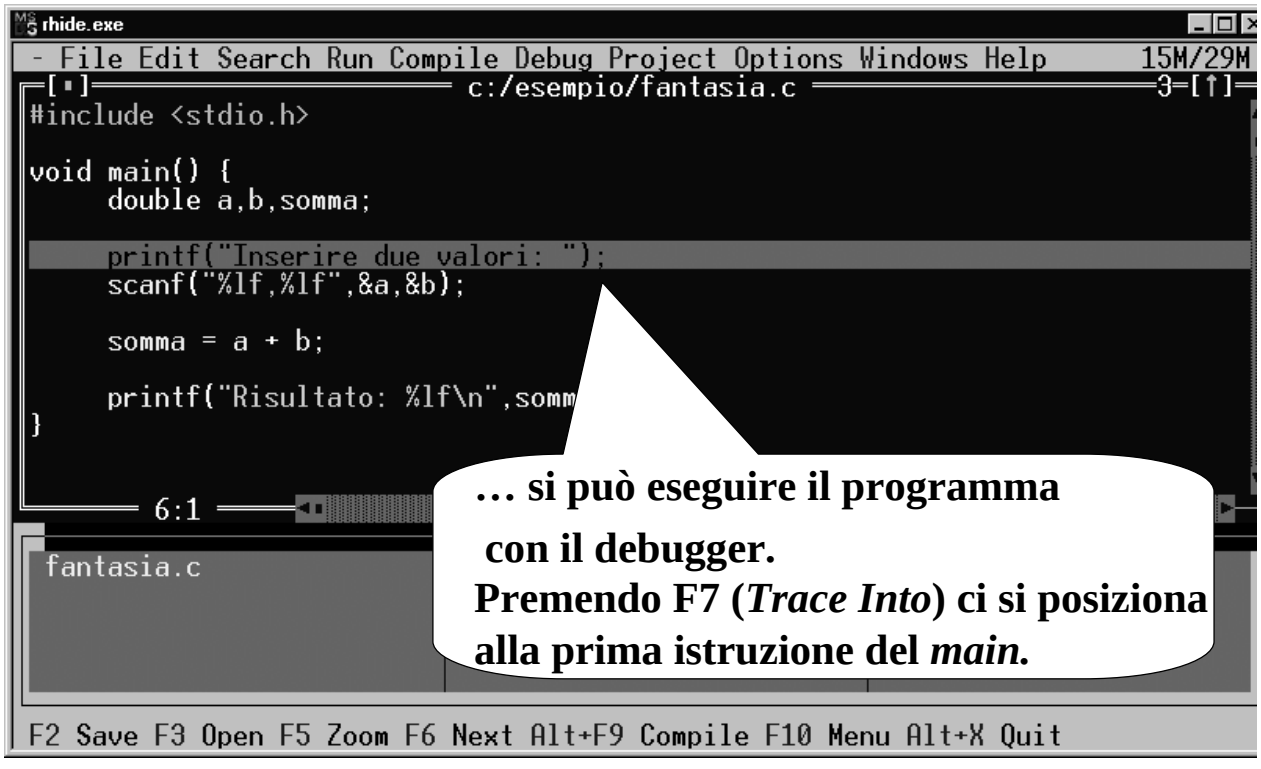


… si chiama il linker

(In un colpo solo:

Make = compilazione+linking

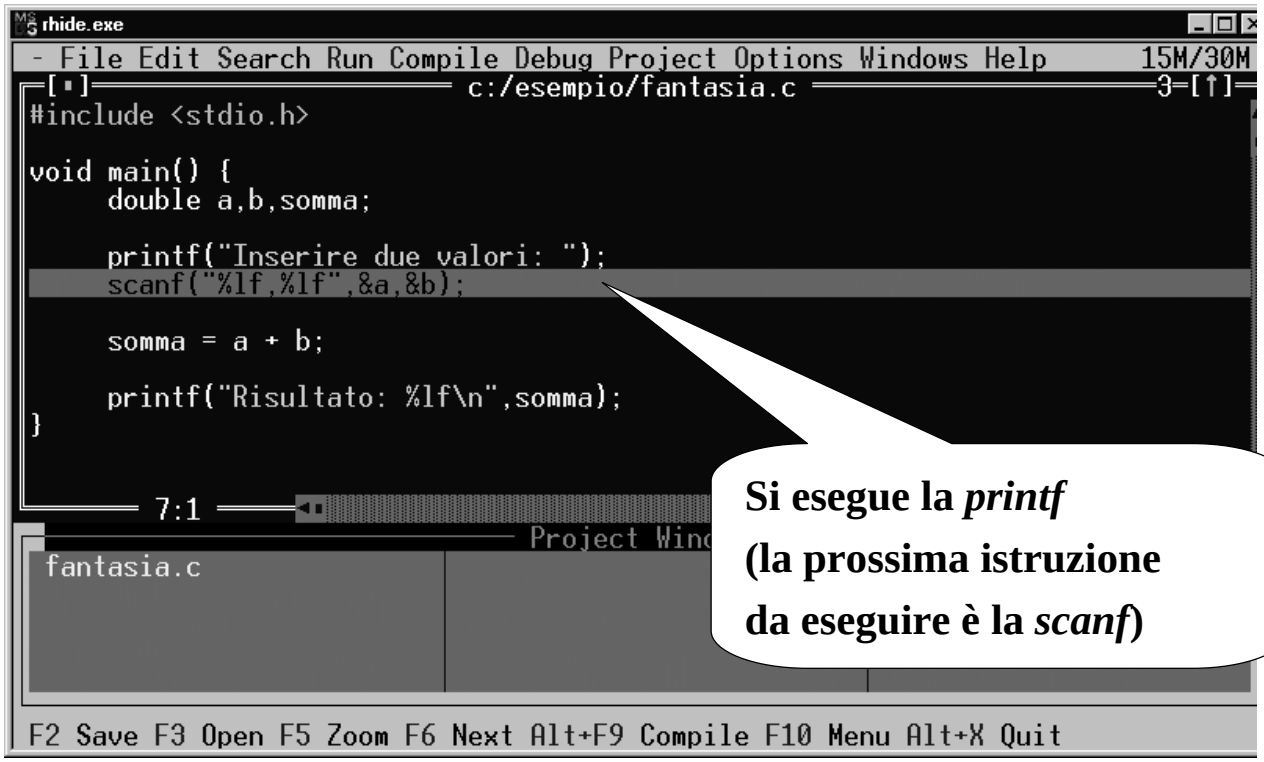
UN ESEMPIO COMPLETO



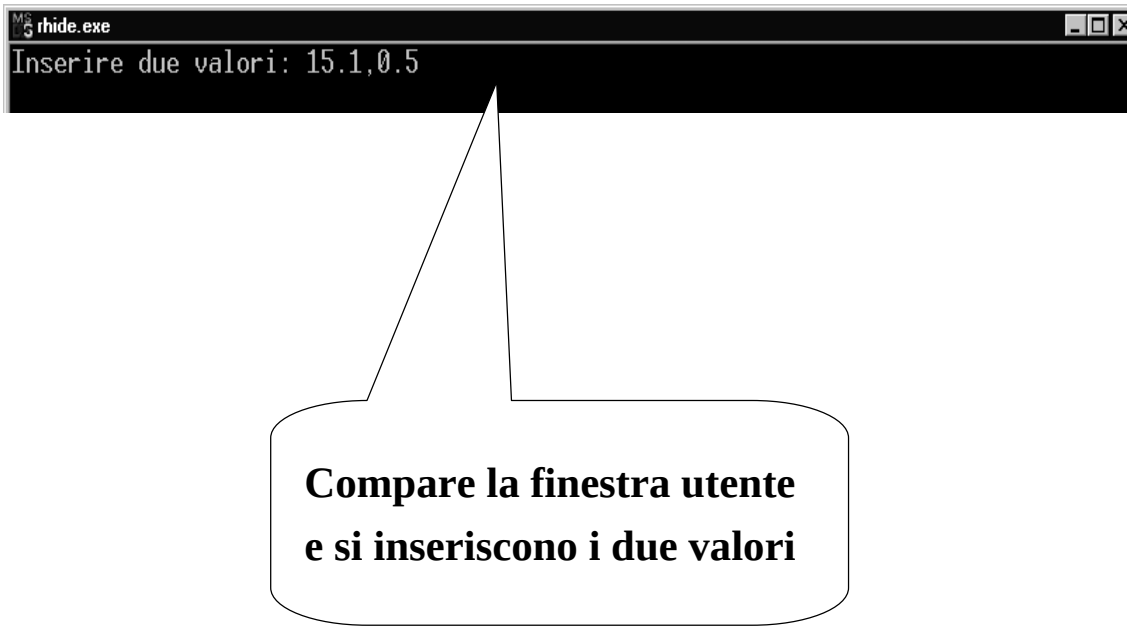
… si può eseguire il programma con il debugger.

Premendo F7 (*Trace Into*) ci si posiziona alla prima istruzione del *main*.

UN ESEMPIO COMPLETO

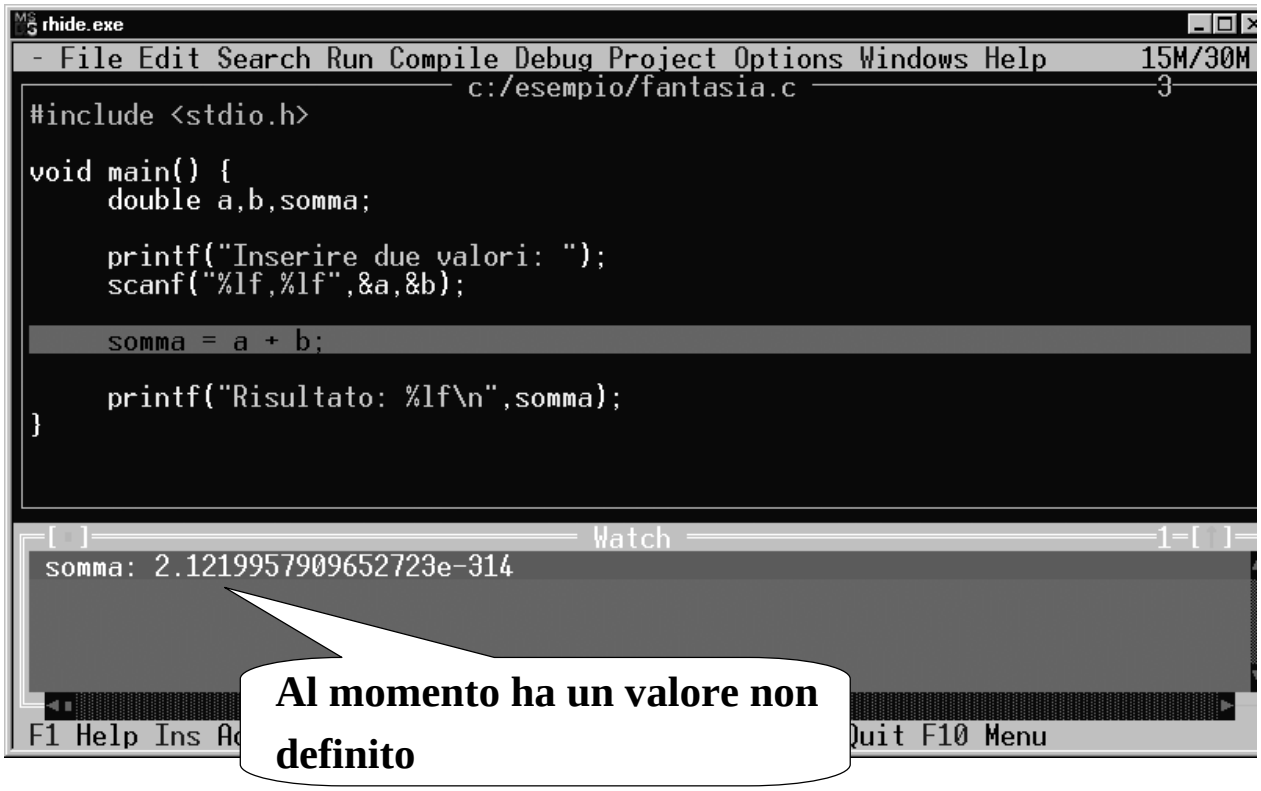


UN ESEMPIO COMPLETO



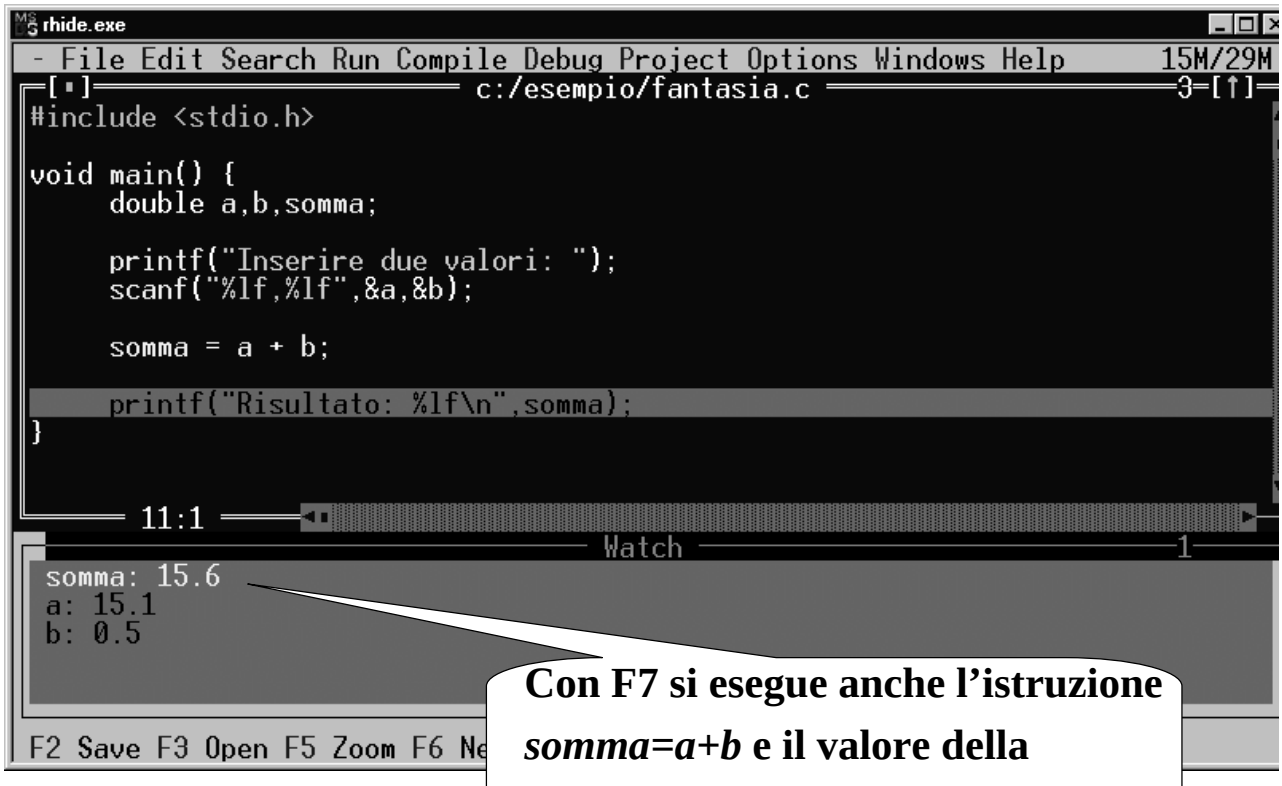
UN ESEMPIO COMPLETO

UN ESEMPIO COMPLETO



UN ESEMPIO COMPLETO

UN ESEMPIO COMPLETO



UN ESEMPIO COMPLETO

UN ESEMPIO COMPLETO

Per vedere l'effetto della *printf* si passa alla finestra utente (ALT+F5)