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.***

1

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`**

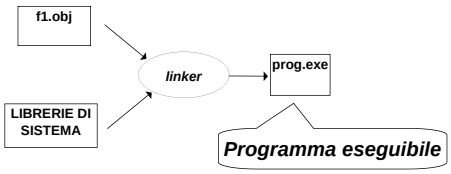
f1.obj: Una versione tradotta che però non è autonoma (e, quindi, non è direttamente eseguibile).

2

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**



3

COLLEGAMENTO DI UN'APPLICAZIONE

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

4

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.**

5

PROGETTI

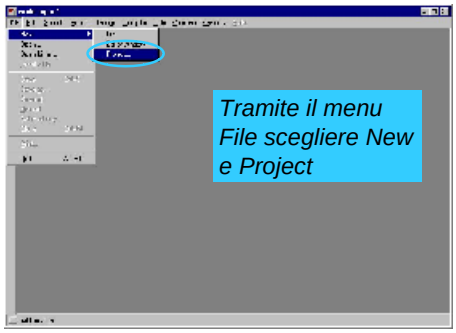
È da queste esigenze che nasce il concetto di *PROGETTO*

- **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.

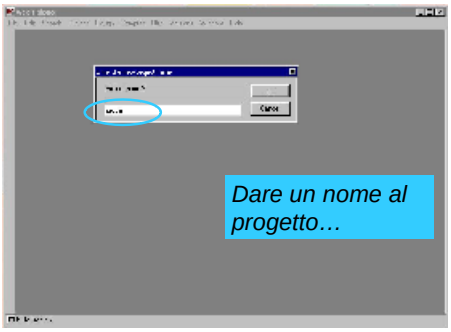
6

PROGETTI IN LCC



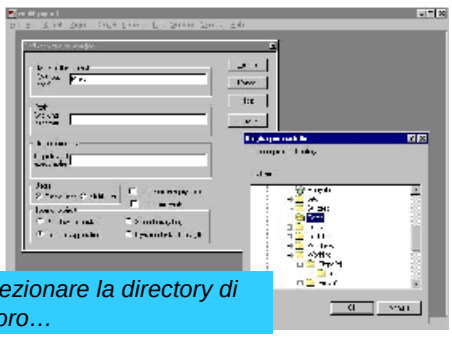
7

PROGETTI IN LCC



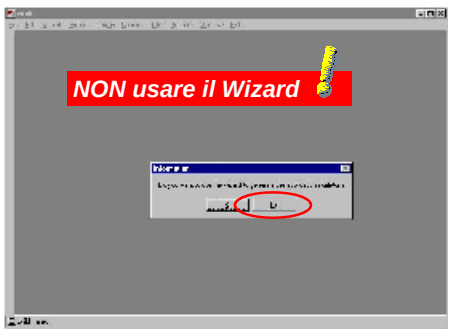
8

PROGETTI IN LCC



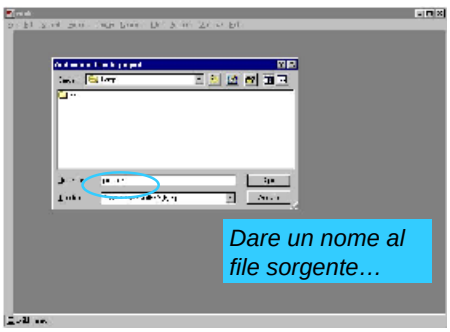
9

PROGETTI IN LCC



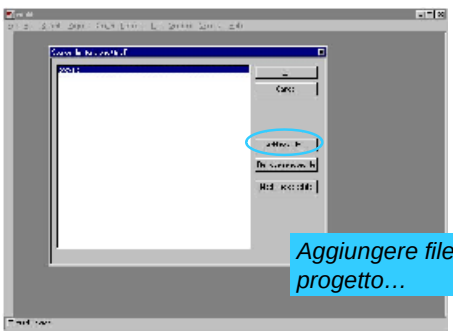
10

PROGETTI IN LCC



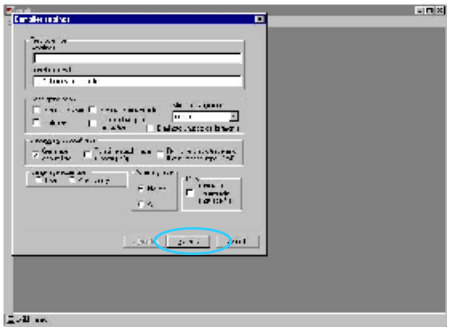
11

PROGETTI IN LCC



12

PROGETTI IN LCC



13

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

14

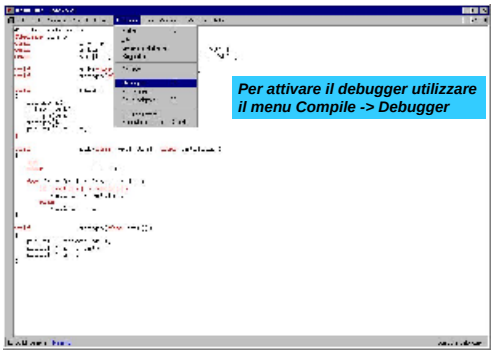
DEBUGGER

Sia LCC sia altri ambienti di sviluppo incorporano un *debugger* con cui eseguire il programma,

- ***riga per riga***
  - **entrando anche dentro alle funzioni chiamate**
  - **oppure considerando le chiamate di funzione come una singola operazione**
- **oppure *inserendo breakpoints***

15

DEBUGGER



Per attivare il debugger utilizzare il menu Compile -> Debugger

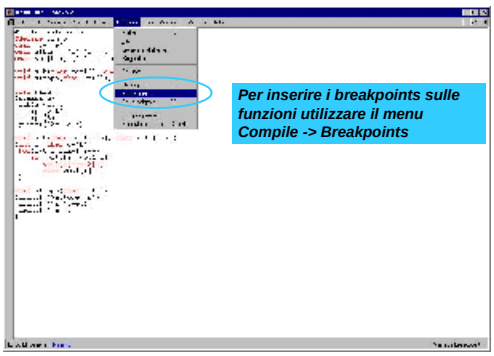
16

FASE DI DEBUGGING

- **Prima di iniziare la sessione di debugging e' possibile inserire i cosiddetti *breakpoints***
  - ***punti di interruzione nell'esecuzione del programma in cui il debugger fornisce una "fotografia" dello stato delle variabili***
- **Due modi per inserirli:**
  - ***sulle funzioni***
  - ***sulle singole istruzioni***

17

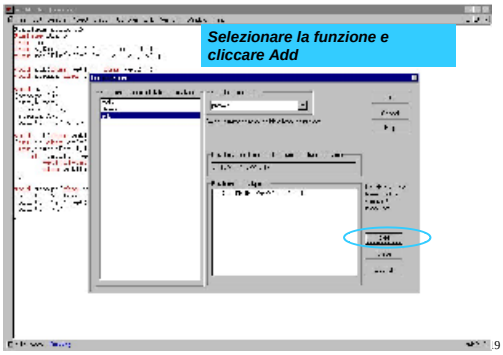
DEBUGGER



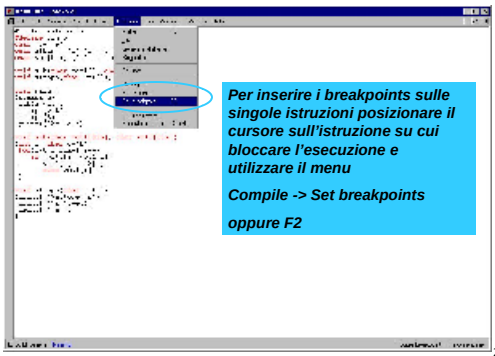
Per inserire i breakpoints sulle funzioni utilizzare il menu Compile -> Breakpoints

18

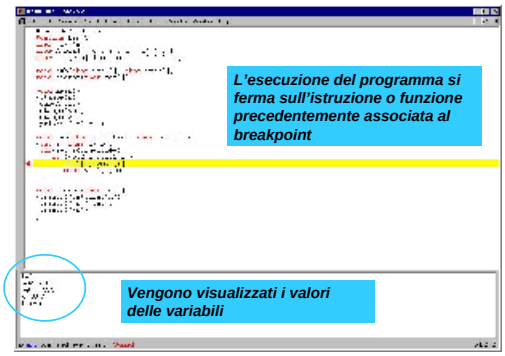
DEBUGGER



DEBUGGER



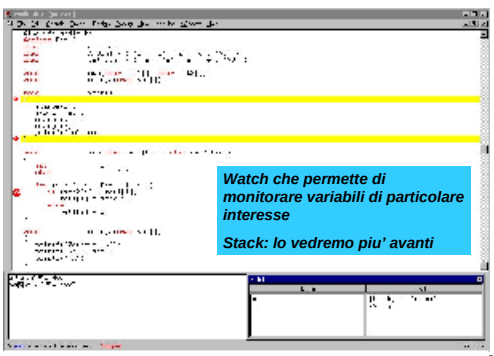
DEBUGGER



DEBUGGER: COME PROCEDERE

- **Nel menu Debug che compare quando il Debugger e' attivo ci sono alcune voci importanti:**
  - **Execute: esegue il programma fino alla fine senza interruzioni**
  - **Step in: esegue passo passo le istruzioni di una funzione**
  - **Same level: esegue la funzione come istruzione singola**
  - **Run to cursor: permette di posizionare il cursore in una determinata posizione nel sorgente e esegue tutte le istruzioni fino ad arrestarsi al cursore.**

DEBUGGER: COME PROCEDERE



This document was created with Win2PDF available at http://www.daneprairie.com.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.