### PER UTILIZZARE LCC IN LAB

La prima volta che un utente utilizza lcc-win32 deve impostare una chiave nel registro utente.

### Per fare questo occorre semplicemente:

- Aprire con il gestione risorse la cartella x:\lcc
- Fare doppio clic sul file registry.reg

E' sufficiente fare questa operazione la prima volta e le impostazioni verranno mantenute.

### 1

### **COSTRUZIONE DI UN'APPLICAZIONE**

### Per costruire un'applicazione occorre:

- compilare il file (o / 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**

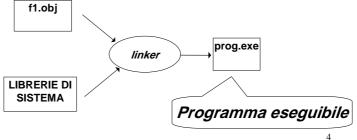
- 1) Compilare il file (o *i* file se più d'uno) che contengono il testo del programma
  - File sorgente: estensione .c
  - File *oggetto*: estensione .o o .obj



f1.obj: Una versione tradotta che però <u>non è autonoma</u> (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**

### **LIBRERIE DI SISTEMA:**

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

-

### AMBIENTI INTEGRATI

## Oggi, gli <u>ambienti di lavoro integrati</u> 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**

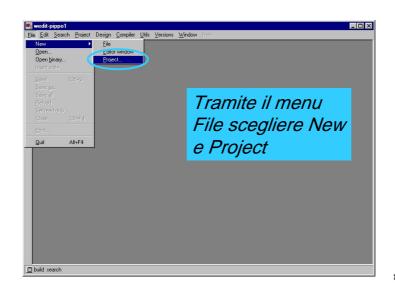
# È da queste esigenze che nasce il concetto di *PROGETTO*

- 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.

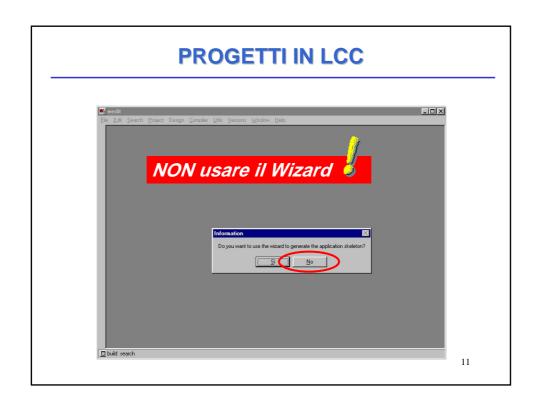
,

### **PROGETTI IN LCC**

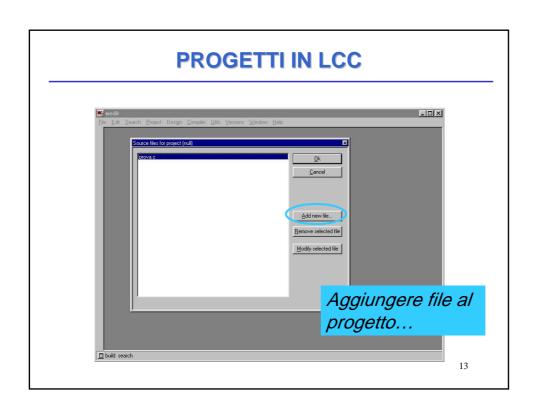


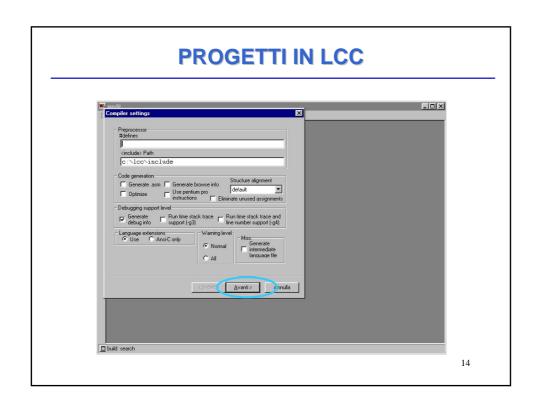








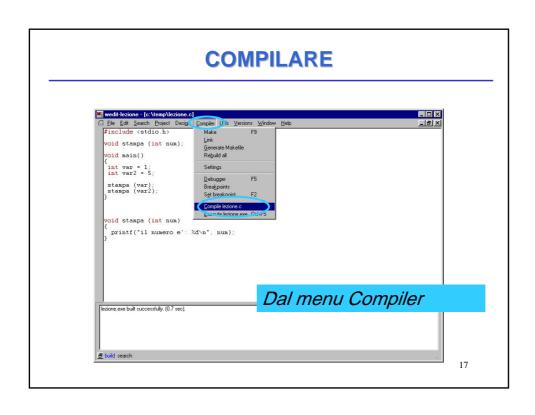


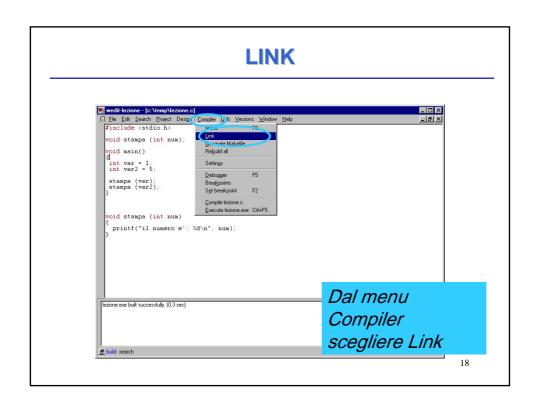


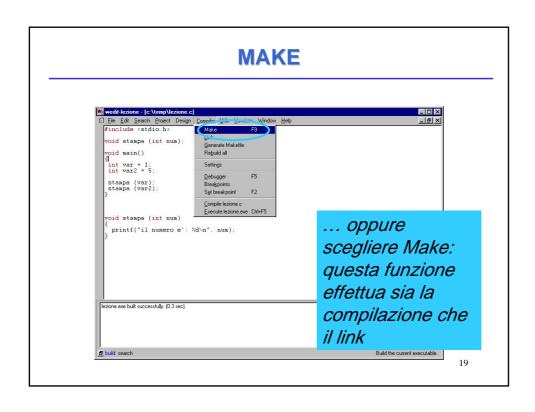
# #include <stdio.h> main() {int x,y; scanf("%d%d",&x,&y); printf("%d",x+y);

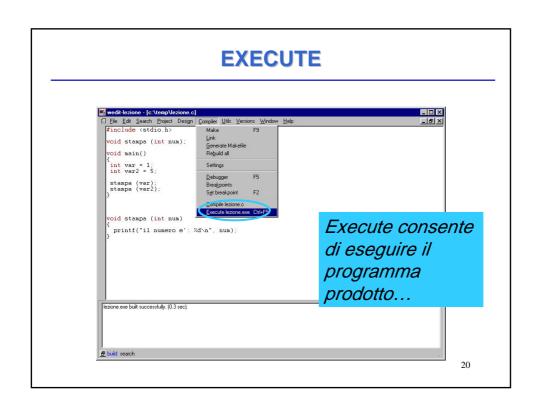
}

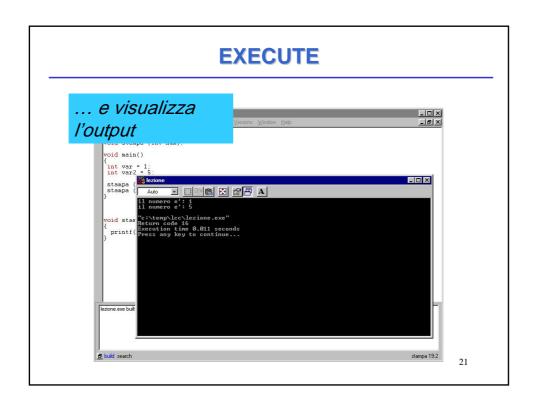












# #include <stdio.h> main() {int x,y; printf("Inserire due numeri separati da virgola: "); scanf("%d,%d", &x, &y); printf("%d + %d = %d", x, y, x+y); }

### **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.



23

### **DEBUGGER**

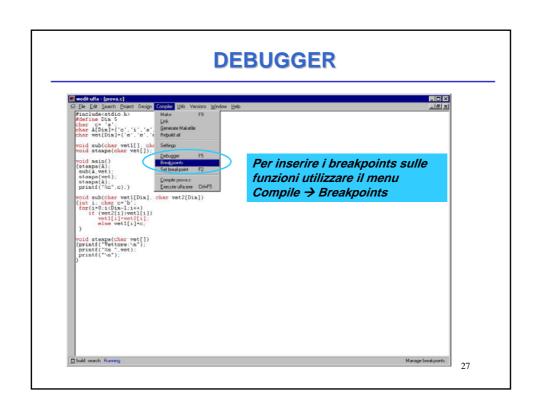
Sia LCC sia altri ambienti di sviluppo incorporano un *debugger* con cui eseguire il programma,

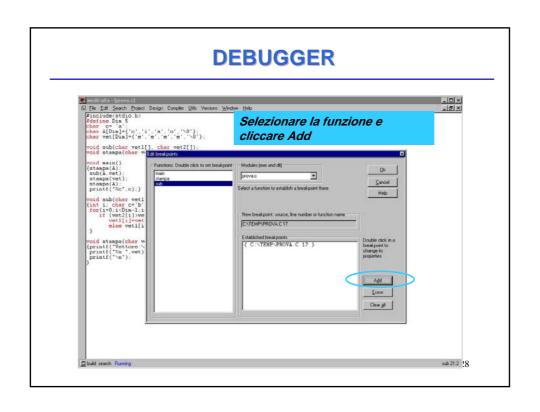
- riga per riga
  - entrando anche dentro alle funzioni chiamate
  - oppure considerando le chiamate di funzione come una singola operazione
- oppure inserendo breakpoints

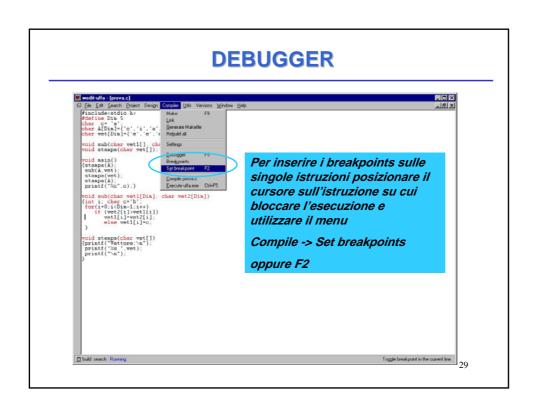
# 

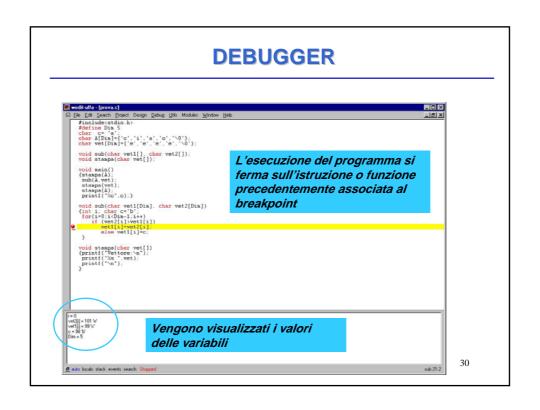
### **FASE DI DEBUGGING**

- 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









### **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.

31

### **DEBUGGER: COME PROCEDERE**

