

Guida al Debugging di programmi C, C++ e Assembler utilizzando il Data Display Debugger (DDD)

A cura dell'Ing. Marco Cococcioni

Cos'è DDD

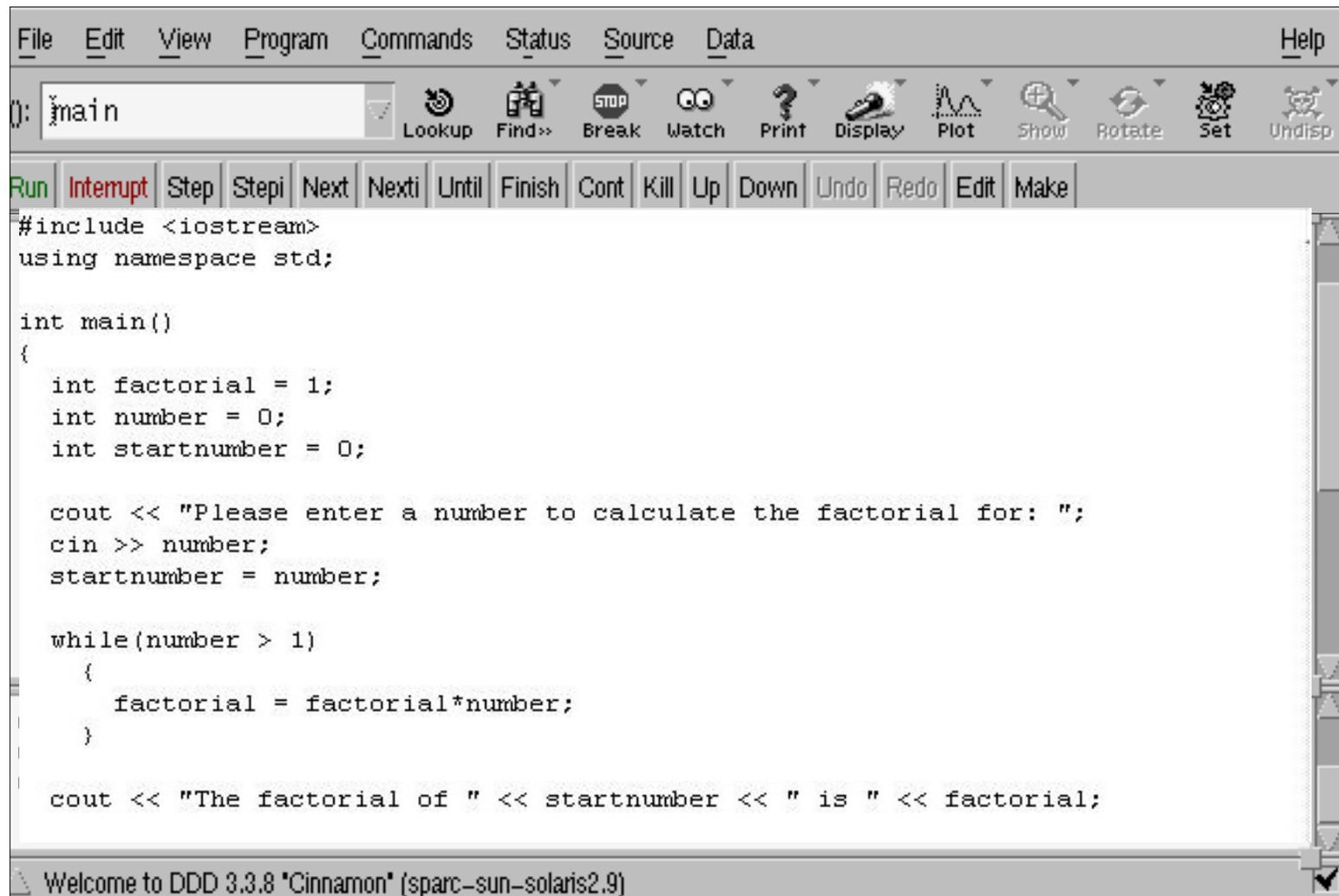
- DDD sta per Data Display Debugger
- DDD è una interfaccia grafica (GUI) per lo GNU debugger, un programma di debug disponibile alla riga di comando.
- Come si avvia DDD alla riga di comando?
- Passo 1: compilare e linkare inserendo le informazioni per il debugger (opzione -g):

```
$ g++ -g es1.cpp -o es1.exe
```

- Passo 2: avviare ddd:

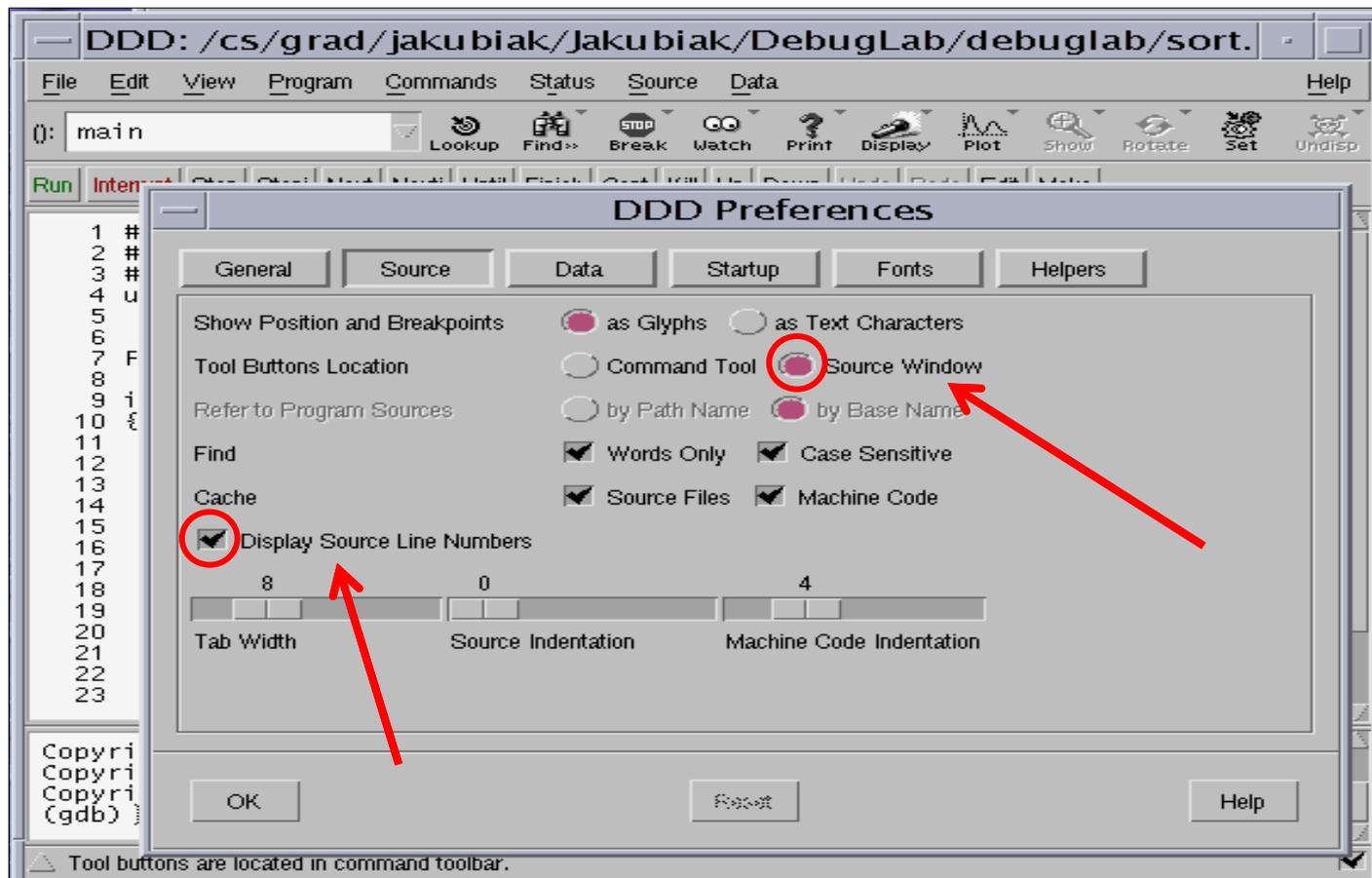
```
$ ddd es1.exe &
```

Come appare DDD



Customizzazione di DDD

- Numero di riga: *Source* → *Display Line Numbers*
- Ancoraggio della barra degli strumenti: *Edit* → *Preferences*



Come impostare i punti di interruzione

- Prima di mettere in esecuzione il programma occorre impostare uno o più **punti di interruzione** (*breakpoints*), altrimenti una volta avviato andrebbe immediatamente alla fine
- Per impostare un breakpoint:
 - **click-pulsante-destro** alla linea in cui si desidera impostare l'interruzione
 - Selezione dell'opzione **Set Breakpoint**
 - (ora dovrebbe comparire un segnale di stop all'inizio di quella linea)

Avvio del programma da debuggare

- In cima allo schermo, selezionare il bottone “Run” per avviare il programma
 - Run può anche essere trovato nel menu ‘program’
- Il programma si arresterà in corrispondenza della riga in cui è posizionato il breakpoint
 - (La linea nera indica la prossima riga che verrà eseguita)

Next, Hover, e Step

- Usare *Next* per muoversi nel programma una riga alla volta
- Dopo ogni *Next* è possibile posizionare il cursore sopra una variabile per vederne il valore
 - Questa operazione è chiamata *hovering*
- *Step* può essere usata in alternativa a *Next*
 - *Step* fa andare il programma una riga avanti, ma nel caso di chiamata di funzioni ci fa eseguire passo passo anche la funzione
 - *Next* fa andare il programma una riga avanti. Nel caso di chiamata di funzione, salta la chiamata e ci porta alla riga successiva alla chiamata stessa.

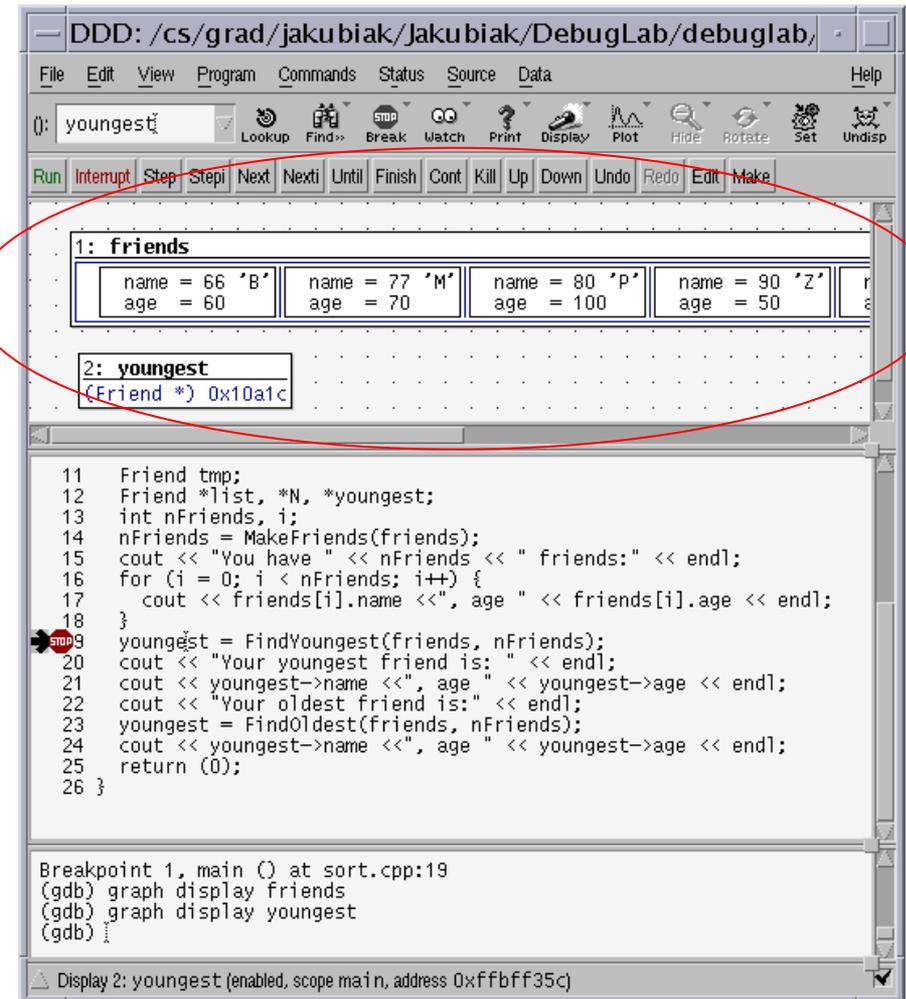
Correzione del programma (*Bug Fixing*)

- Per correggere il programma, una volta rilevato l'errore (baco):
 - chiudere DDD
 - aprire il file sorgente usando l'editor (gedit es1.cpp)
 - correggere il programma
 - ricompilare e rilinkare con opzione -g
 - riaprire l'eseguibile con DDD



Ispezione delle variabili (1/2)

- Fare click sul pulsante destro sopra una variabile e poi scegliere **Display nomeVariabile**
 - questo provoca l'apertura del *display editor*
- Fare ancora click sul pulsante destro sopra al nome della variabile e scegliere **Display *nomeVariabile**
(notare l'asterisco)

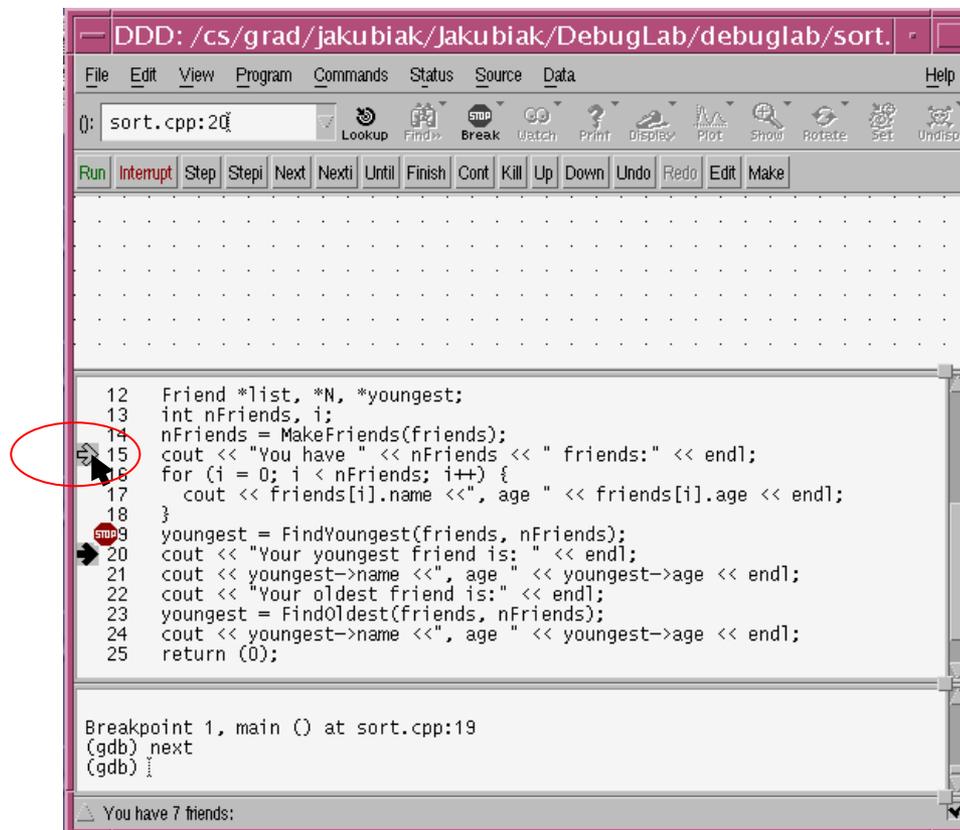


Ispezione delle variabili (2/2)

- Le variabili possono essere visualizzate nell'area di display in diverse rappresentazioni:
 - */t nomeVar* : binario
 - */d nomeVar* : decimale
 - */h nomeVar* : esadecimale
 - */o nomeVar* : ottale
- Nel caso di programmi assembler il contenuto dei registri può essere visualizzato:
 - */t \$eax*: mostra eax in bin
 - */d \$eax*: mostra eax in decimale, ecc...
- Il contenuto dei registri può essere visualizzato anche nella finestra console di dello GNU debugger (GDB), nel seguente modo:
 - *i r eax* (che sta per info register eax)
- NB1: qui non serve \$, né %, prima del nome del registro
- NB2: le porzioni di registro (al, ah, ax, bl, ...) non possono essere visualizzate da sole

Per tornare indietro nell'esecuzione

- Per andare indietro rispetto alla linea di interruzione
 - clickare e trascinare la freccia in alto, alla linea desiderata



```
DDD: /cs/grad/jakubiak/Jakubiak/DebugLab/debuglab/sort.  
File Edit View Program Commands Status Source Data Help  
(): sort.cpp:20  
Lookup Find Break Watch ? Print Display Plot Show Rotate SET Undisp  
Run Interrupt Step Stepi Next Nexti Until Finish Cont Kill Up Down Undo Redo Edit Make  
.....  
.....  
.....  
12 Friend *list, *N, *youngest;  
13 int nFriends, i;  
14 nFriends = MakeFriends(friends);  
15 cout << "You have " << nFriends << " friends:" << endl;  
16 for (i = 0; i < nFriends; i++) {  
17     cout << friends[i].name << ", age " << friends[i].age << endl;  
18 }  
19 youngest = FindYoungest(friends, nFriends);  
20 cout << "Your youngest friend is: " << endl;  
21 cout << youngest->name << ", age " << youngest->age << endl;  
22 cout << "Your oldest friend is:" << endl;  
23 youngest = FindOldest(friends, nFriends);  
24 cout << youngest->name << ", age " << youngest->age << endl;  
25 return (0);  
  
Breakpoint 1, main () at sort.cpp:19  
(gdb) next  
(gdb) [br/>> You have 7 friends:
```