FASE DEBUGGING:

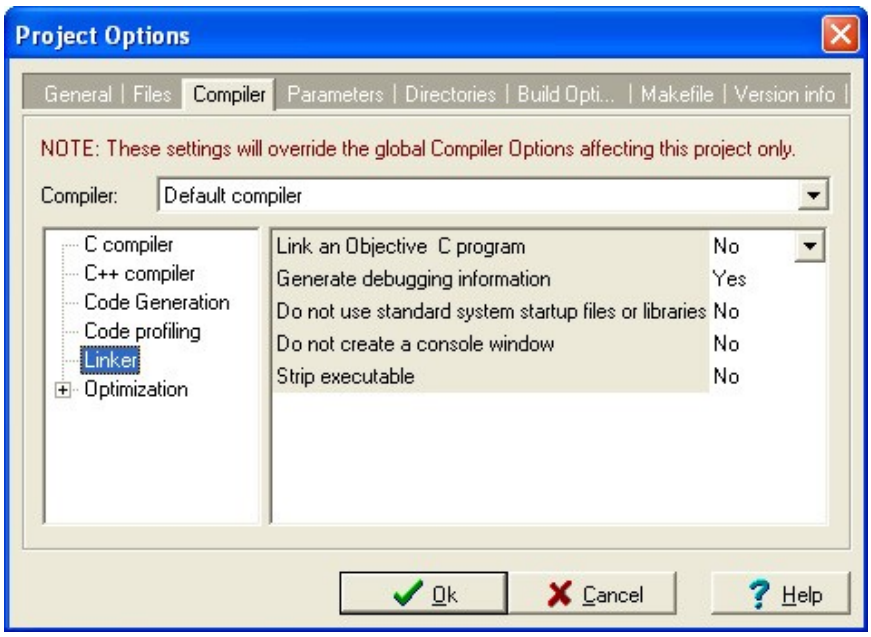
- Prima della compilazione, si devono inserire[1] nel progetto **informazioni per il debug** cioè si devono visualizzare le **opzioni di progetto** seguendo il percorso:

  Compiler → Linker

  controllando che la voce **Genera le informazioni per il debug** cioè

  "*Generate debugging information*"

  sia selezionata: **Yes**



- Si attiva il programma ***debugger***[2] con **icona Debug** 

oppure da menu comando **Debug → Debug** o con tasto **F8** che consente di interrompere e riprendere l'esecuzione del programma utente compilato e linkato.

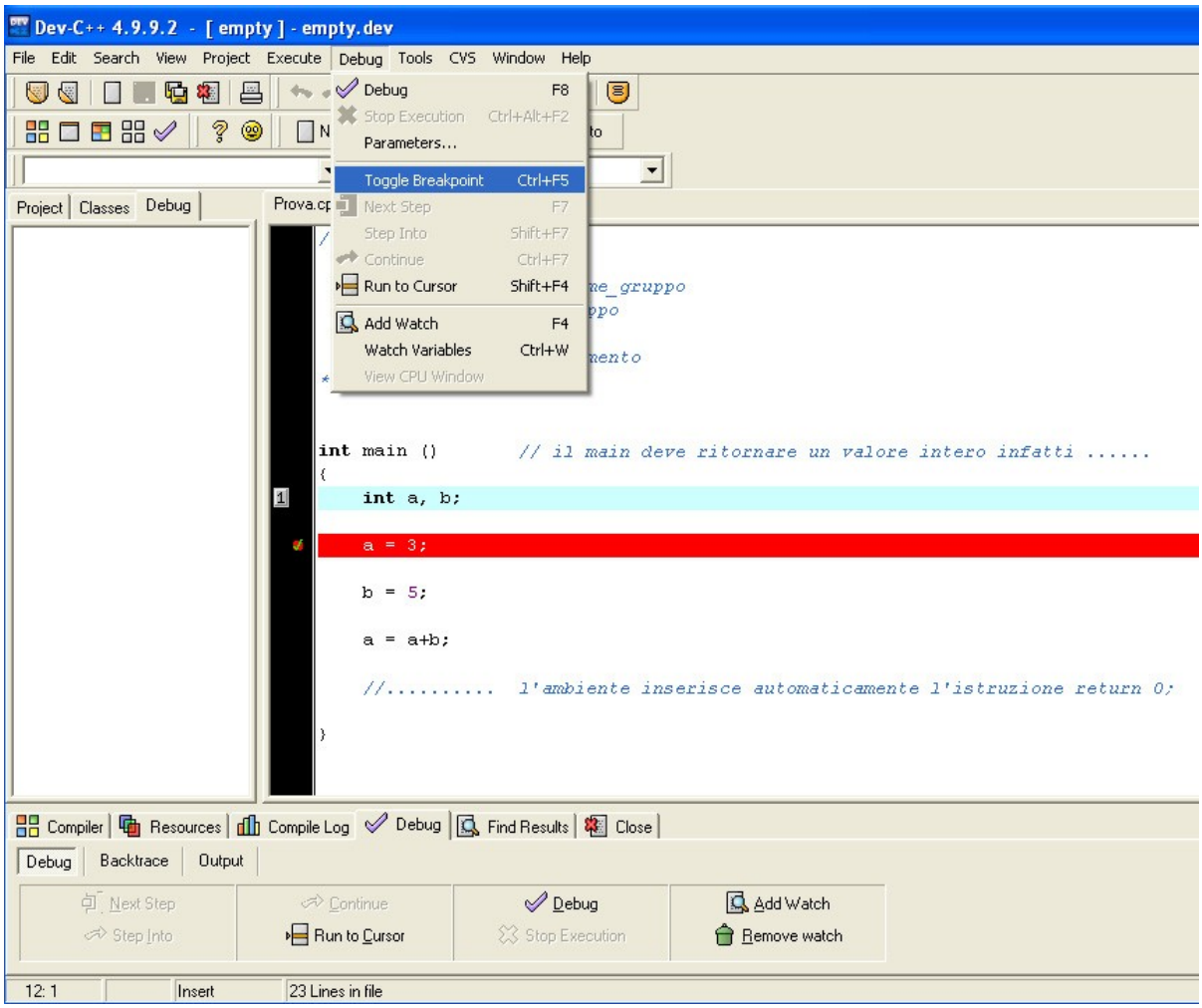
  E' pertanto necessario introdurre almeno un punto di interruzione o ***breakpoint*** ed il programma ***Insight Debugger*** lo inserisce automaticamente nella posizione del cursore se si usa il comando "**Esegui dal Cursore**" (versione in Italiano) o "**Run to Cursor**" (versione in Inglese) . Tra le opzioni di **Debug** è infatti prevista la possibilità di:

  - attivare l'esecuzione fino alla linea dove è posizionato il cursore:

    Debug → Esegui dal Cursore 

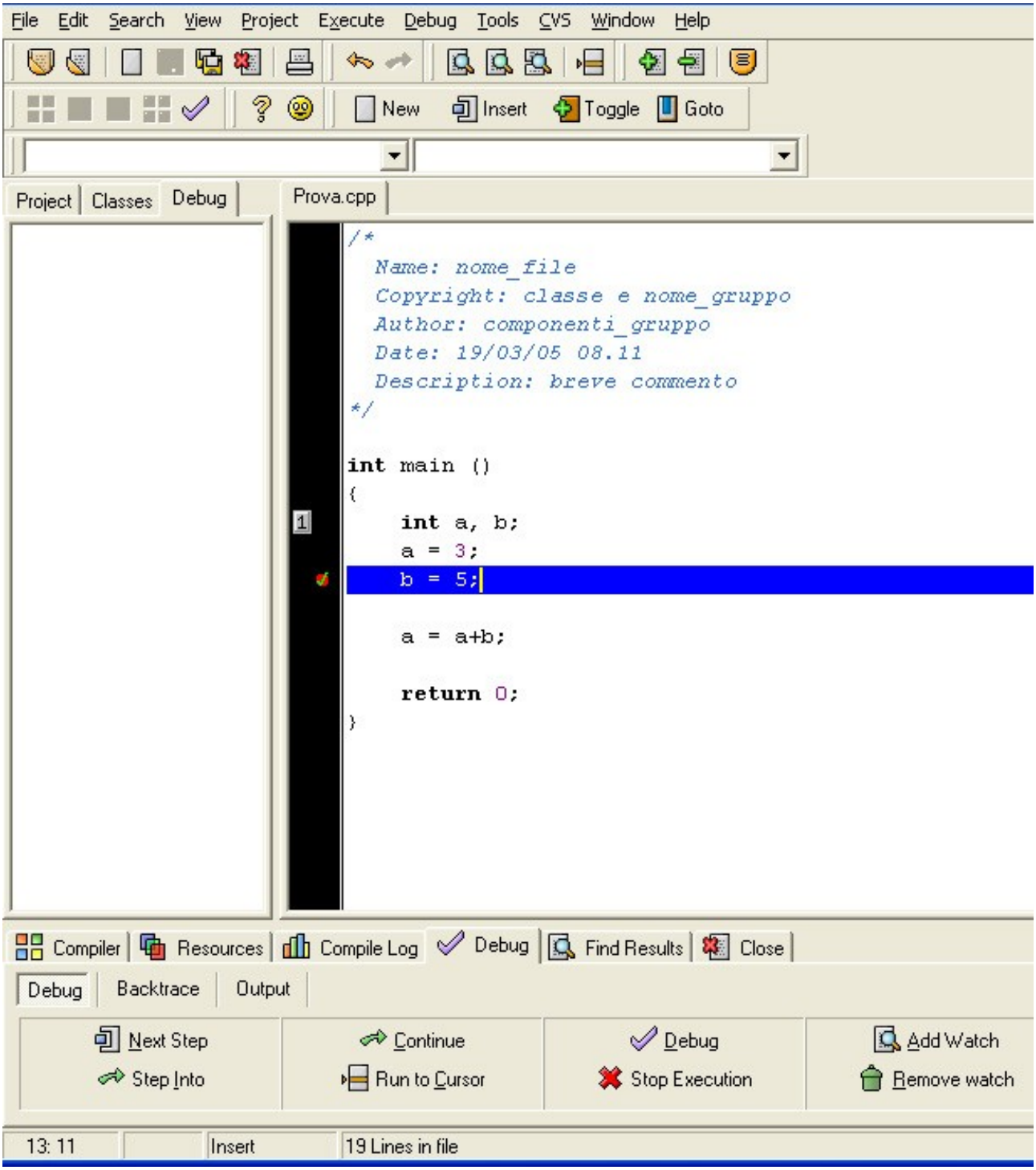
[1] Se non si inseriscono informazioni per il debug, una finestra avverte che bisogna provvedere e ricostruire il progetto.
[2] ***Insight*** cioè il programma **gdb** con interfaccia grafica (Insight Debugger)

- Per **inserire** almeno un ***breakpoint***:
  - basta un click nella zona con sfondo nero, a fianco della linea voluta
  - oppure si usa il comando **Debug → Toggle Breakpoint**



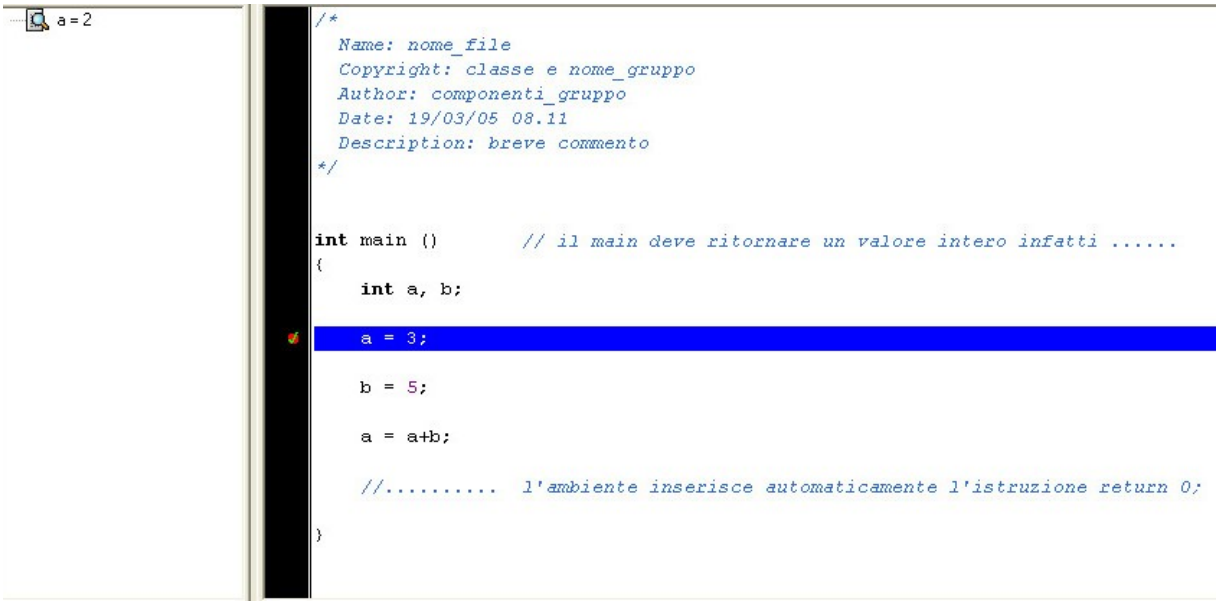
- Per procedere **passo-passo** si esegue, allora, fino al primo punto di interruzione o ***breakpoint*** premendo, come già detto, **F8** oppure l'icona 

oppure selezionando, da menu, il comando: **Debug → Debug**



Potendo, ad esempio, **visualizzare il valore delle variabili prima e dopo l'elaborazione**

- Posizionando il cursore sulla variabile di cui si vuole conoscere il valore (**a** nell'esempio)

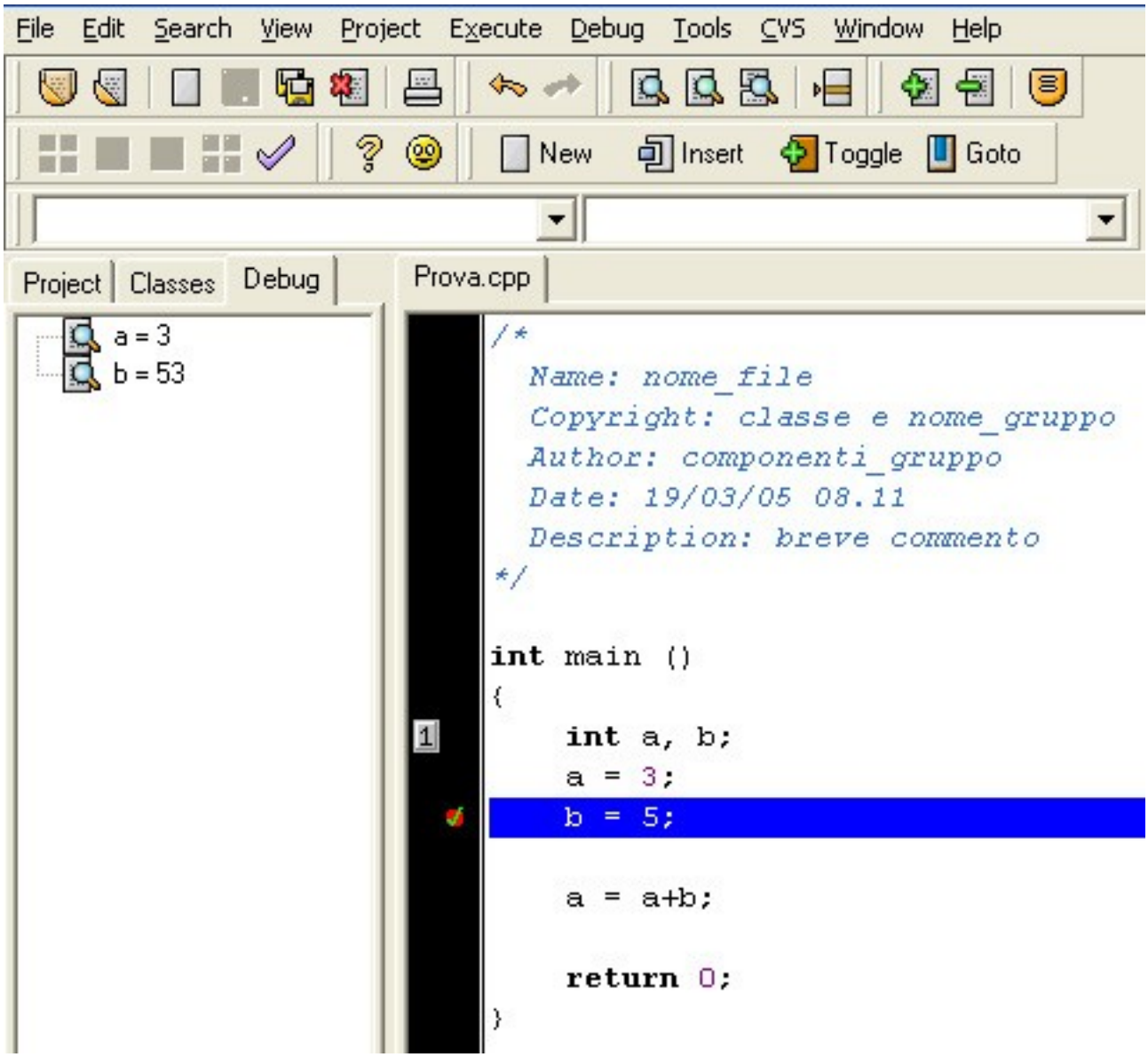


Si noti come, per default, il valore di una variabile intera sia un intero casuale (**a = 2** nell'esempio)

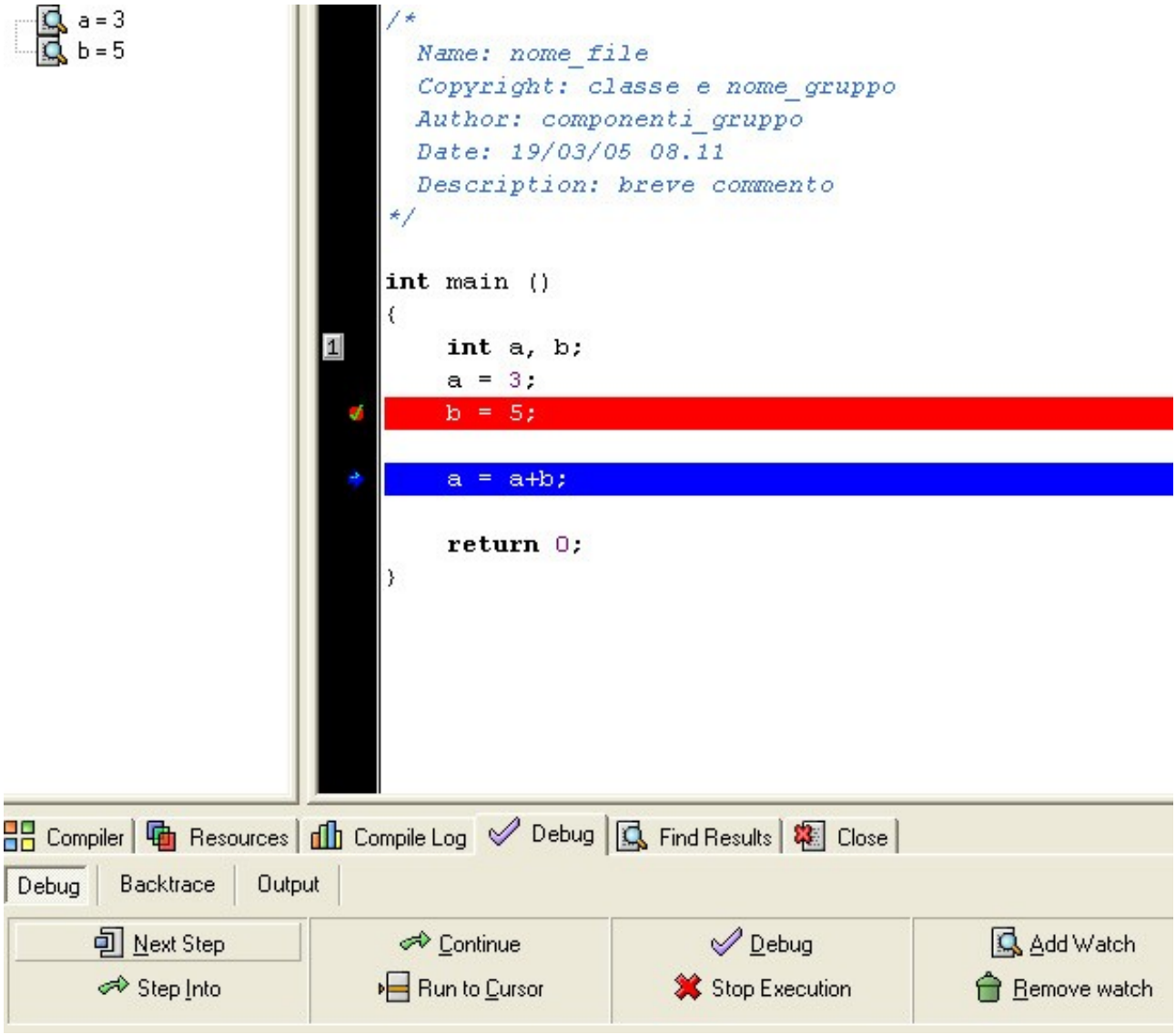
- oppure usando 

ed impostando il nome della variabile

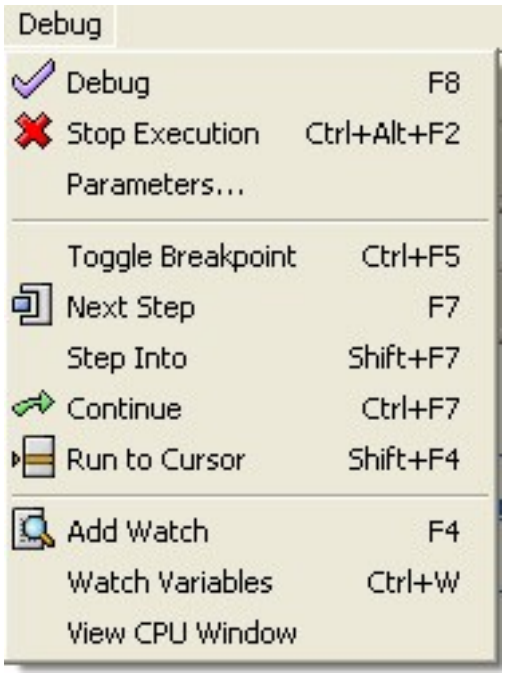
NB: Si noti come, il valore di a **dopo l'inizializzazione** sia quello voluto, mentre per default, il valore di b **prima dell'inizializzazione** sia un intero casuale (**b = 53** nell'esempio)



- Si prosegue con 

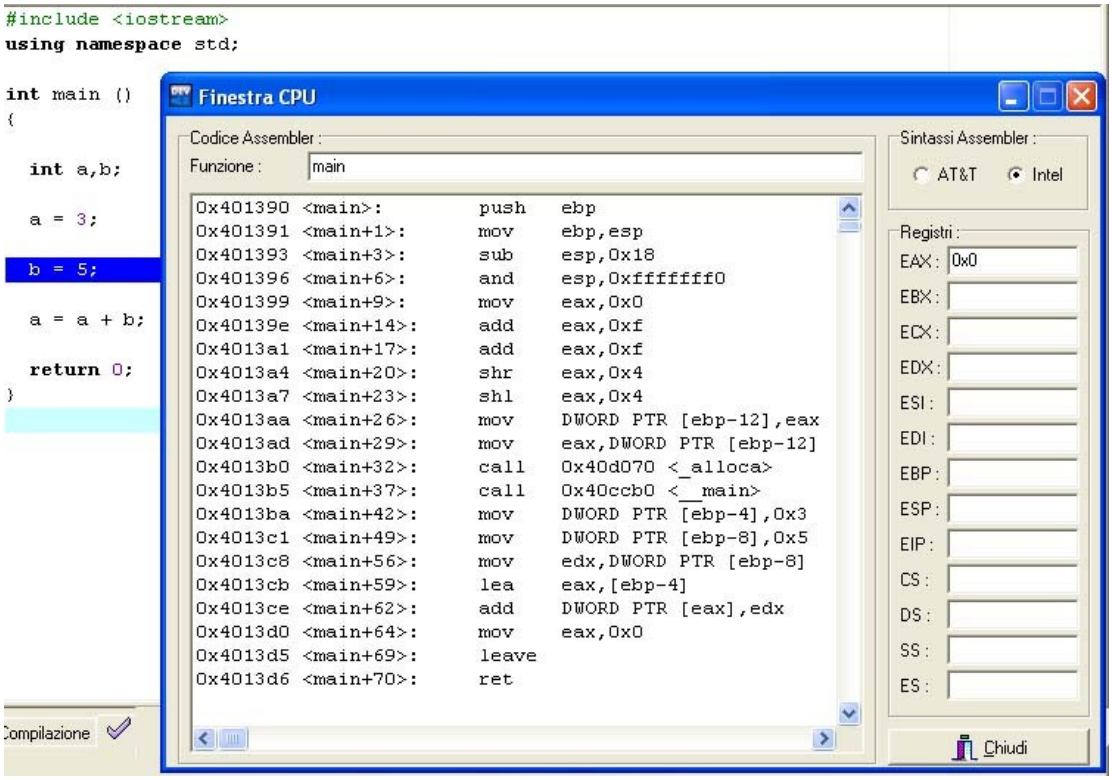


- Le opzioni di **Debug**



permettono anche di:

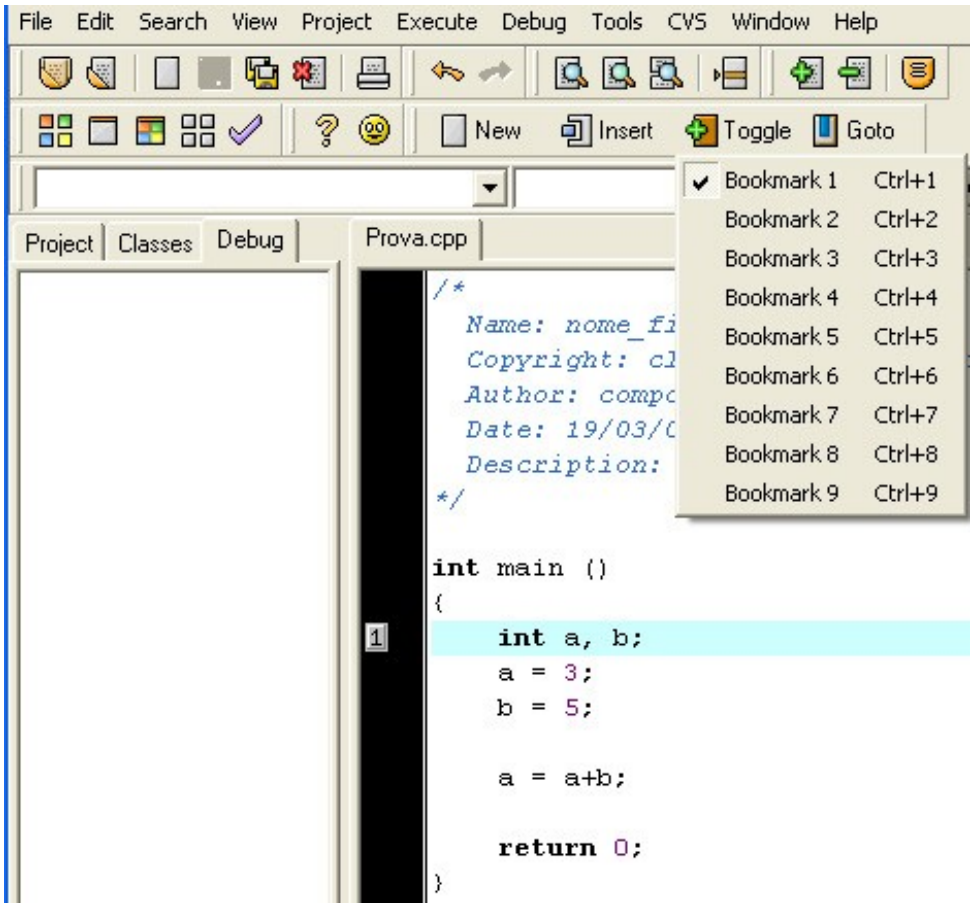
- vedere il **codice Assembler** con sintassi Intel oppure AT&T **Debug → Finestra CPU**



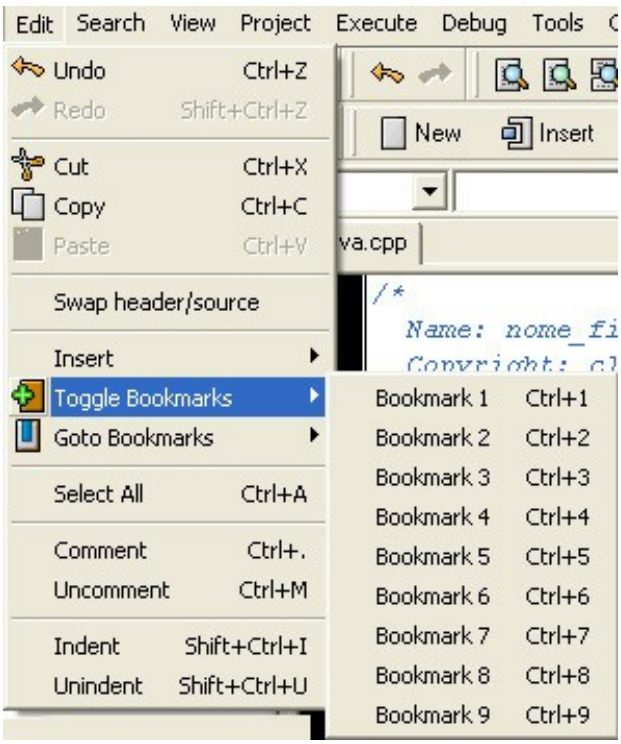
  - Con sintassi **Intel,** l'istruzione **b=5;** corrisponde a salvare il numero esadecimale **mov DWORD PTR [ebp-8], 0x5** che sarà copiato in registro interno alla CPU **mov edx, DWORD PTR [ebp-8]** prima di poterlo sommare

In fase di editing si può inserire[3] almeno un ***bookmark***: si posiziona il cursore sulla riga voluta

e si seleziona l'**icona Toggle**



oppure si seleziona da menu **Edit → Toggle Bookmarks**

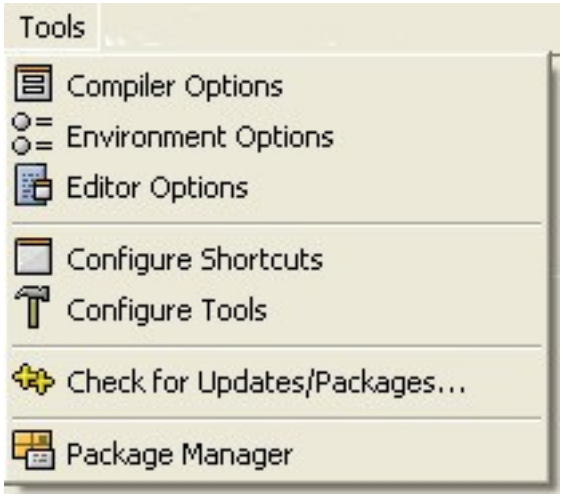


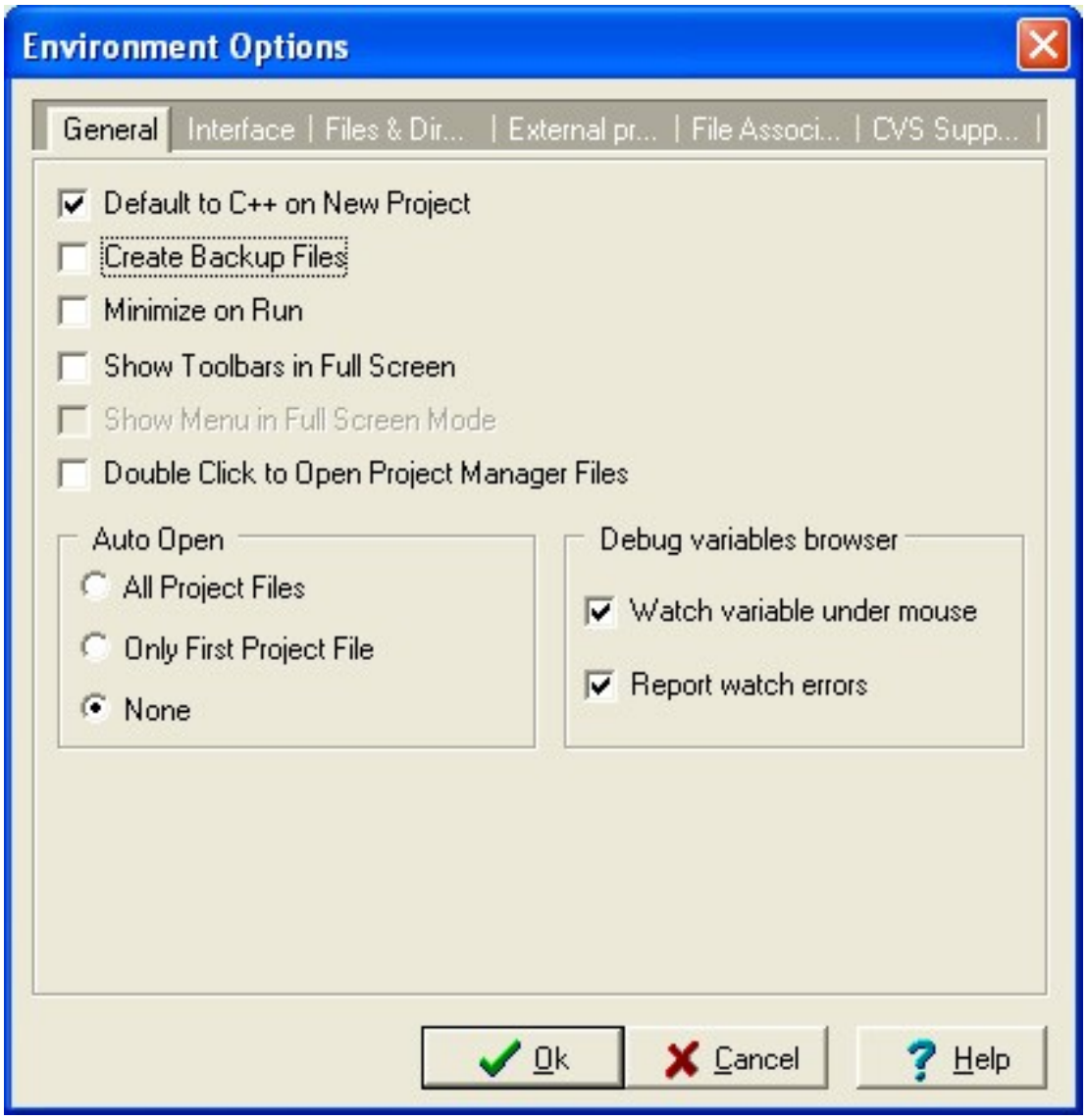
[3] Per velocizzare il posizionamento con *Vai al segnalibro* quando si attiva l'opzione "**Esegui dal Cursore**" o "**Run to Cursor**"

Opzioni dell'ambiente

L'installazione scelta: **linguaggio C++**

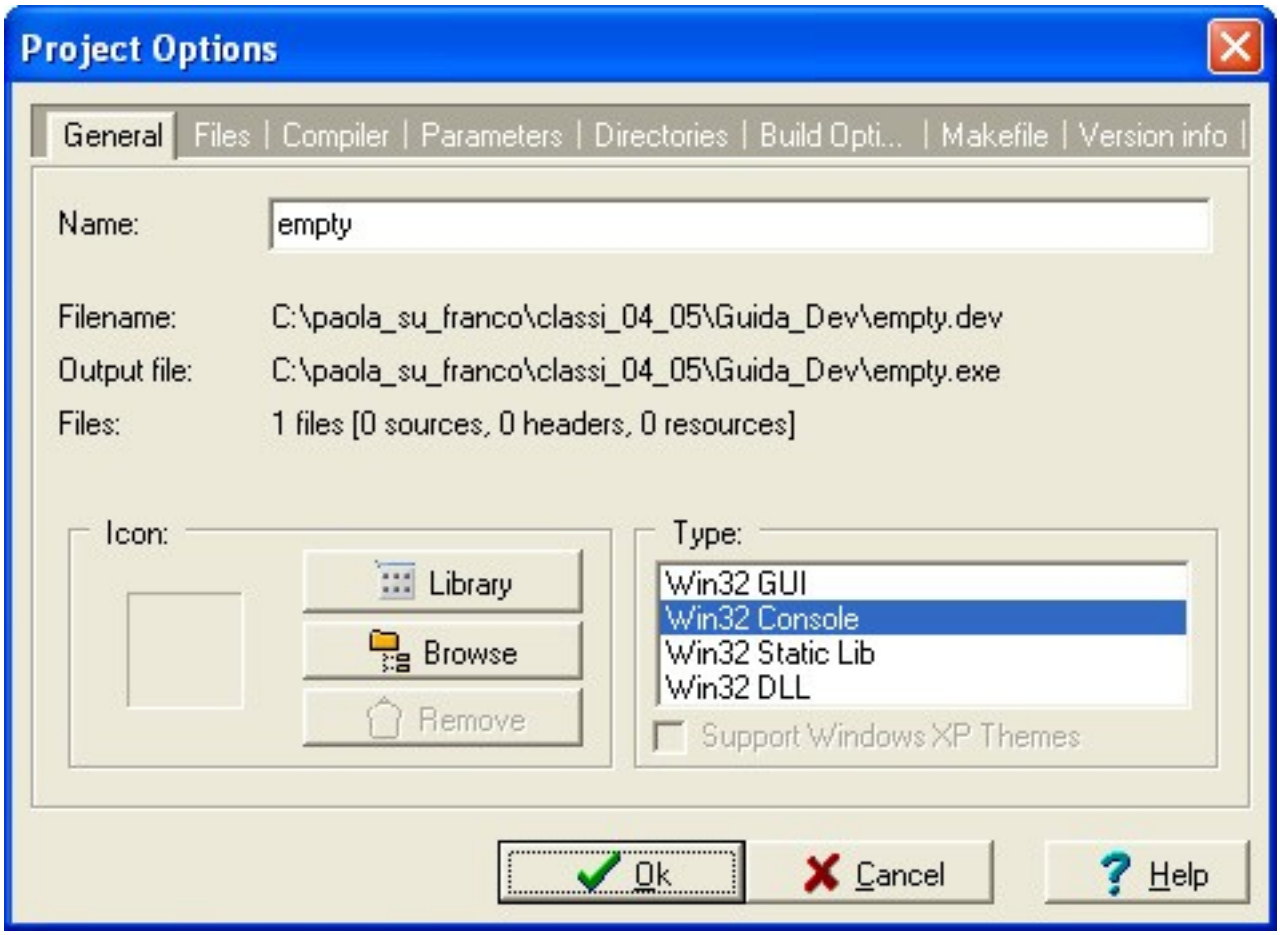
Verificabile con **Tools → Environment Options**





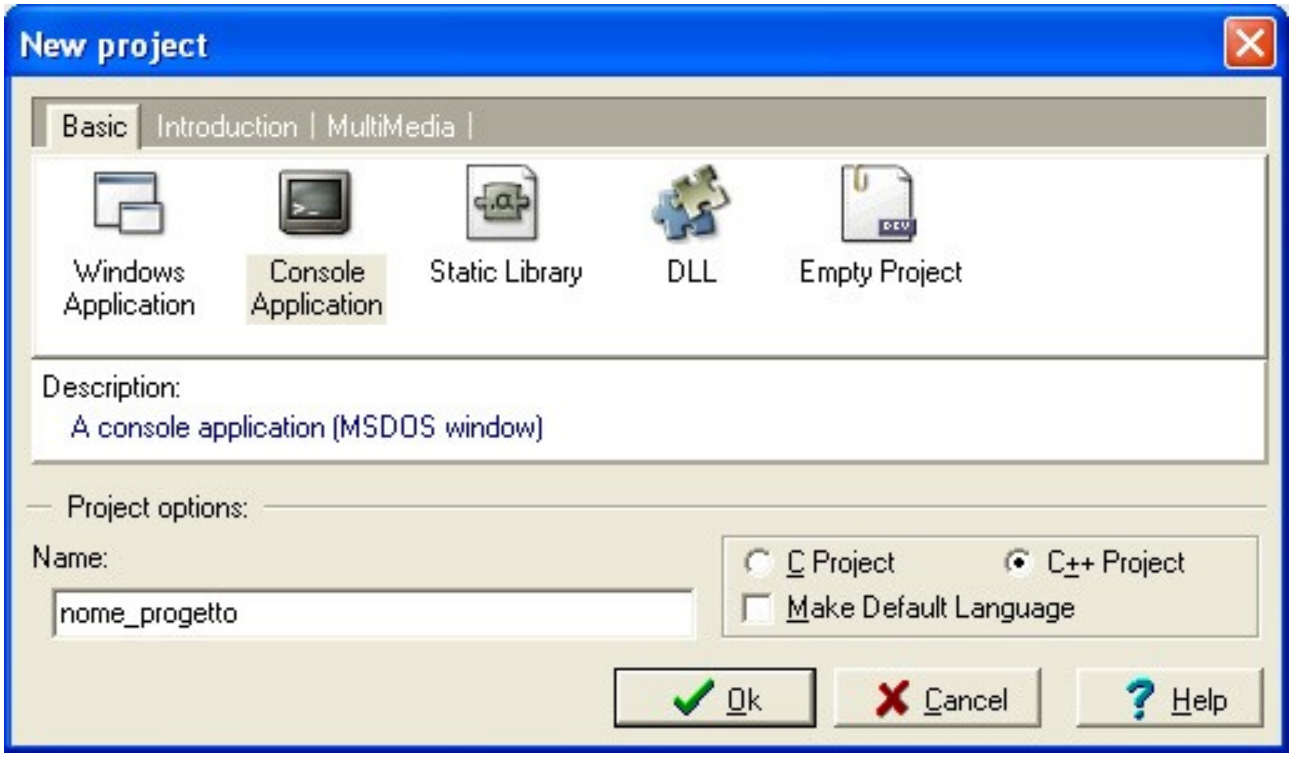
Opzioni generali di progetto

Tipo di applicazione: **Win32 Console**

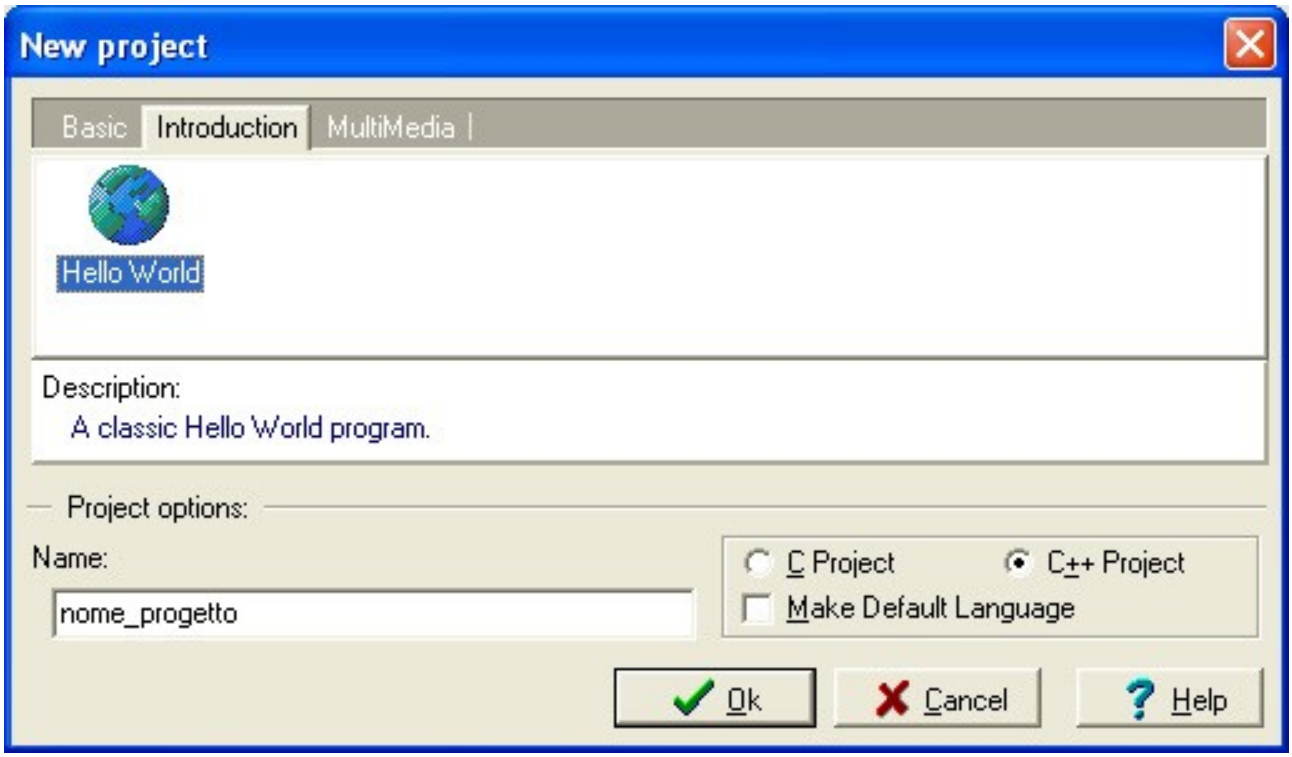


Altre tipologie di progetto dove automaticamente nelle Project Options viene selezionata un'applicazione tipo**: Win32 Console**

- **Basic → Console Application** (con uso **system("PAUSE"); return EXIT_SUCCESS;** definite in **cstdlib**)



- **Introduction → Hello word** (con possibile passaggio di parametri alla funzione main da riga di comando impostabili da menu **Esegui → Parametri …**)



Appendice

Nella versione tradizionale del linguaggio C++ le librerie standard[4] erano contenute in file di intestazione (**header file**)

Il C++ evoluto permette di dividere lo spazio globale di definizione delle variabili (e delle funzioni) in diverse parti dette **namespace** ad ognuna delle quali è associato un nome univoco predefinito nel linguaggio o definito dal programmatore. Tale raggruppamento dei nomi in contenitori detti "***spazio dei nomi***" serve ad evitare ambiguità sull'uso di identificatori in programmi complessi.

Un **namespace** è dunque un insieme di nomi che può essere usato in alternativa ad altri insiemi analoghi e gli identificatori standard del linguaggio C++ sono contenuti nel namespace **std**.

La possibilità di scegliere sintassi diverse, più evolute, è legata alle funzioni predefinite usate e l'uso obbligatorio o alternativo delle versioni più evolute dipende dalle scelte commerciali del produttore del compilatore. Nel caso dell' IDE Dev-C++ si deve usare la sintassi più evoluta per le direttive di precompilazione:

```
#include <fstream>      // nomi delle librerie e non dei file comprensivi di estensione
#include <iostream>

using namespace std;    // clausola per evitare di ripetere il namespace
                        // ad esempio invece di std :: cout  si può scrivere solo cout
```

NB: uso del compilatore GNU

Nell' IDE Dev-C++ si utilizza (come nel sistema Linux) il compilatore GNU (GNU C Compiler o **gcc**) nella versione **Min**imalist **G**NU for **W**indows cioè un compilatore C/C++ per windows[5] e la libreria standard **glibc** (GNU libraries for C) acquisendone implicitamente la compatibilità POSIX[6] (Portable Open System Interface) dove la X è un chiaro riferimento al sistema UNIX.

[4] Per un elenco dei nomi di tali librerie e funzioni contenute puoi digitare **Example Programs Database** usando l'opzione Trova con percorso Help → Help di Dev-C++ 5

[5] La home page del sito è http://www.**mingw**.org/

[6] Sul finire degli anni '80 l' Institute of Electrical and Electronics Engeneers (**IEEE**) iniziò la stesura di una serie di standard pensati per rendere portabili le applicazioni tra architetture diverse: tale famiglia prese nome **POSIX**, sistema parzialmente implementato anche su architetture diametralmente diverse da UNIX ad esempio Windows NT da cui derivano Windows 2000 e Windows XP. Infatti i sottosistemi POSIX definiti nell'architettura Windows NT possono dialogare con il sottosistema Win32 come usare chiamate di sistema opportunamente mediate da sottosistemi di riferimento.