

## FASE DEBUGGING:

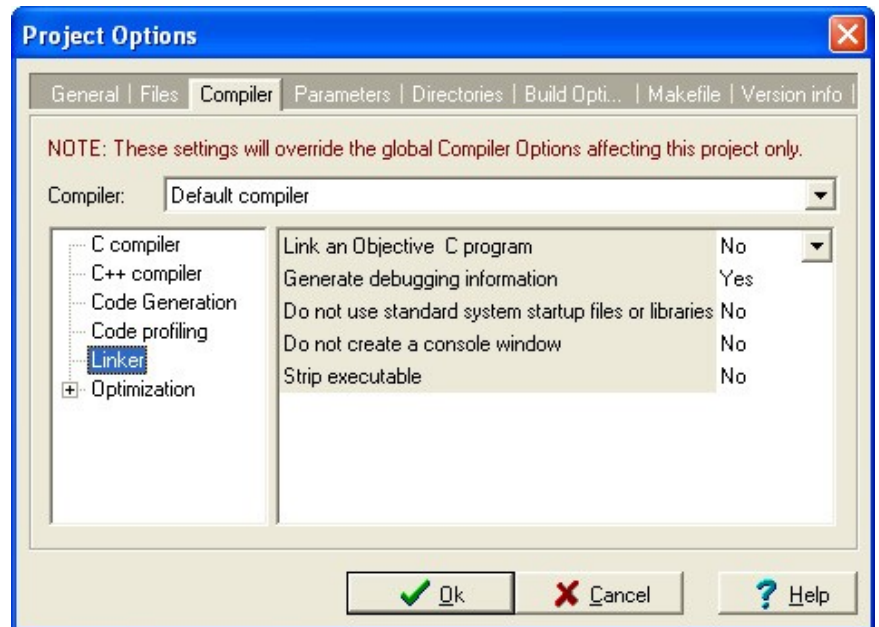
- Prima della compilazione, si devono inserire<sup>1</sup> nel progetto **informazioni per il debug** cioè si devono visualizzare le **opzioni di progetto** seguendo il percorso:


**Compiler → Linker**

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

"*Generate debugging information*"

sia selezionata: **Yes**



- Si attiva il programma *debugger*<sup>2</sup> 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:

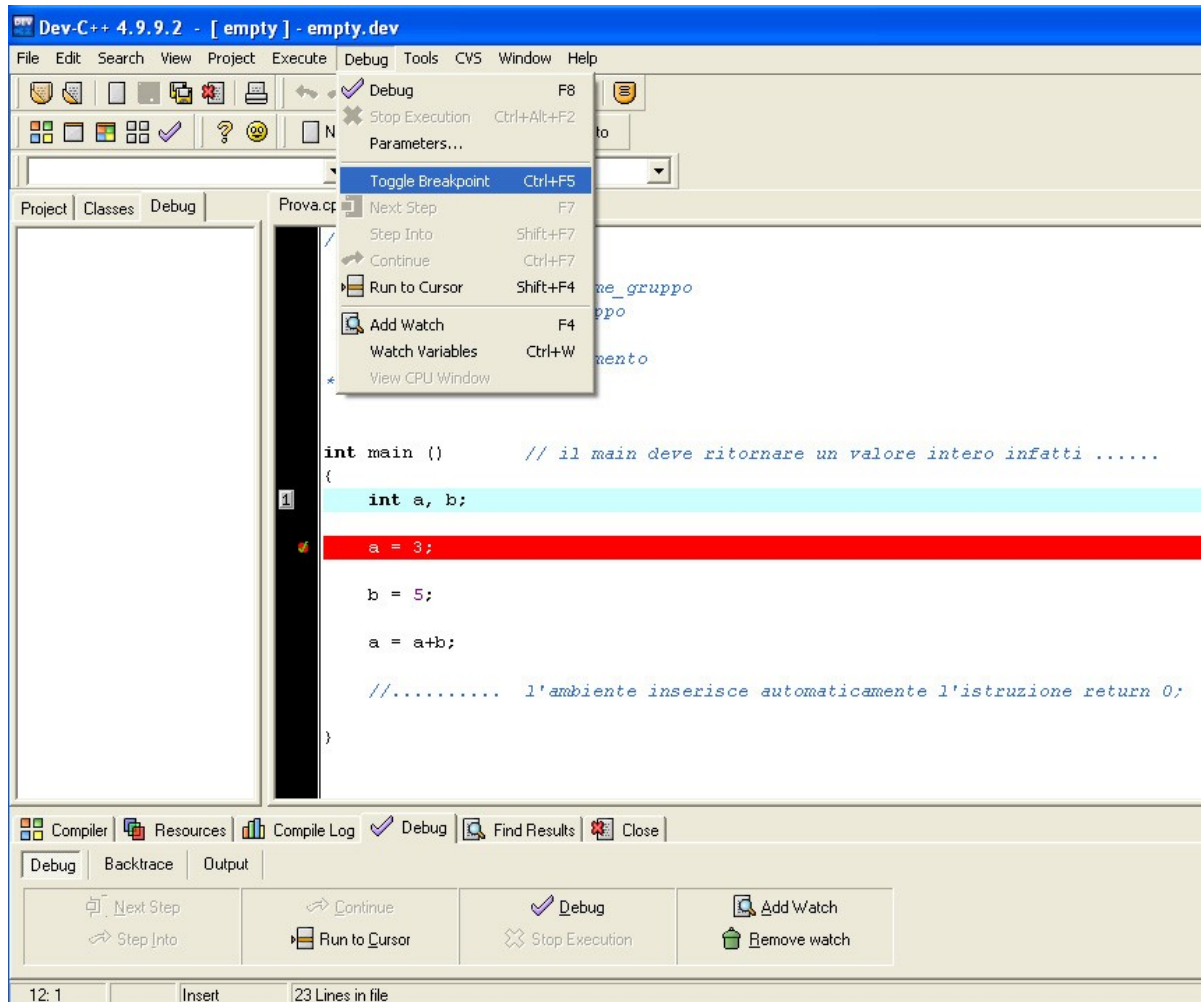
**Debug → Esegui dal Cursore**




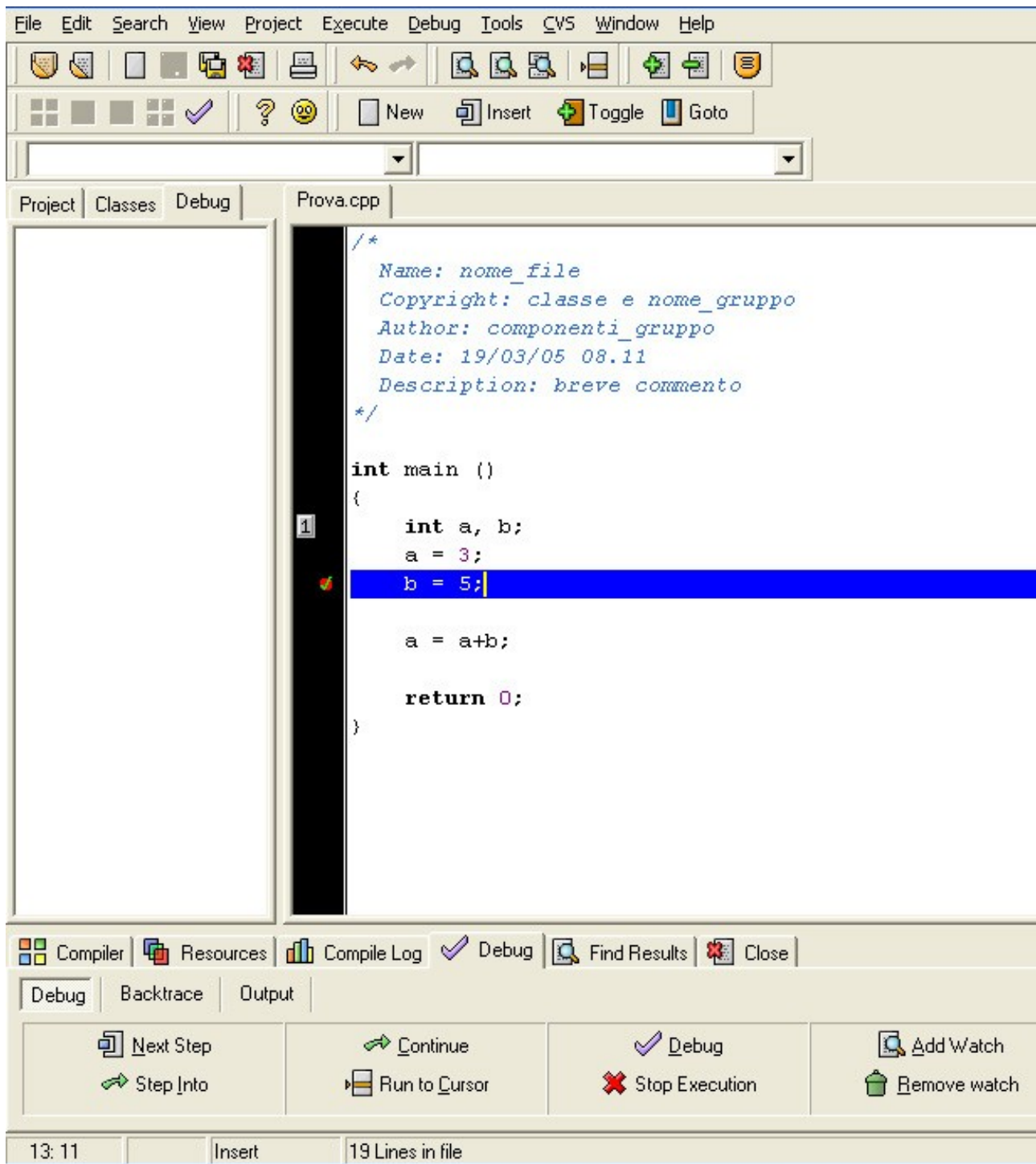
<sup>1</sup> Se non si inseriscono informazioni per il debug, una finestra avverte che bisogna provvedere e ricostruire il progetto.

<sup>2</sup> *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**

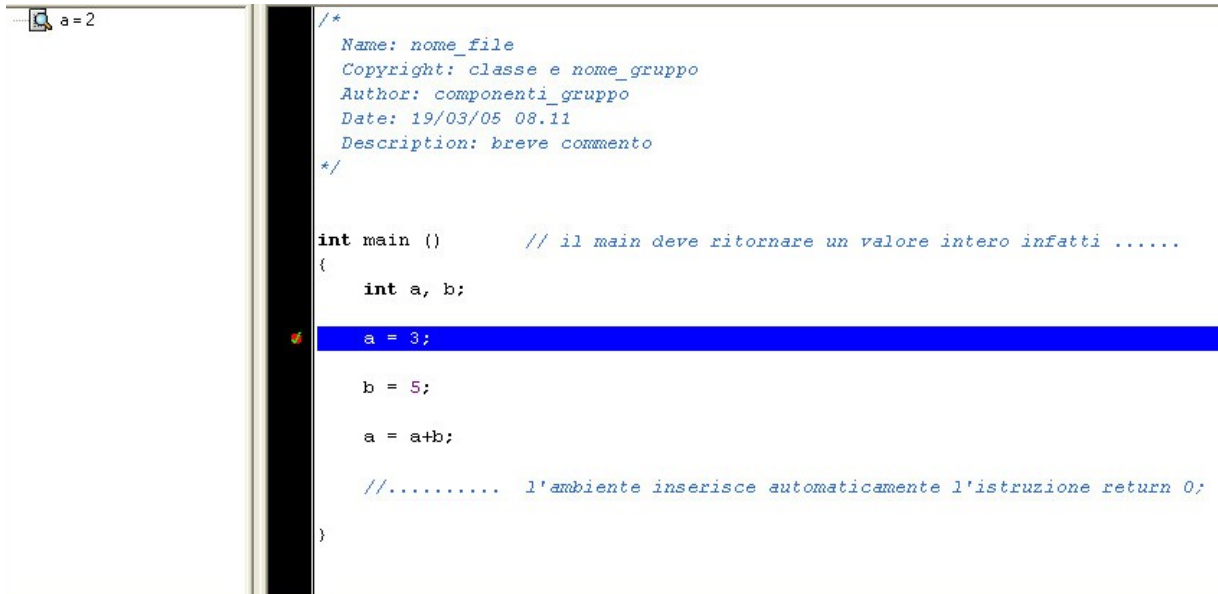


- Per procedere **passo-passo** si esegue, allora, fino al primo punto di interruzione o **breakpoint** premendo, come già detto, **F8** oppure l'icona  **Debug** oppure selezionando, da menu, il comando: **Debug → Debug**



**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)



The screenshot shows a code editor window with a title bar containing a magnifying glass icon and the text 'a=2'. The code is as follows:

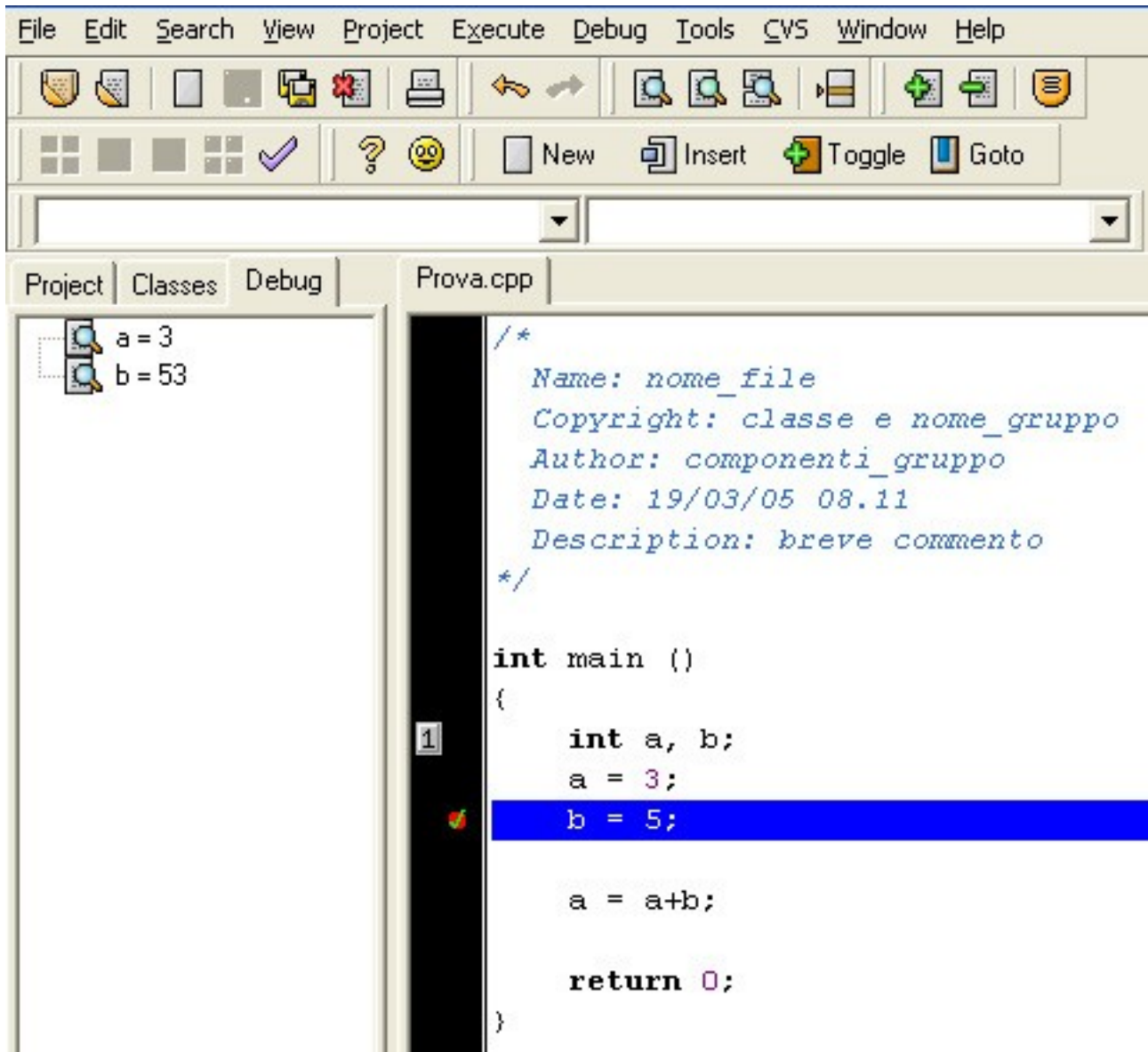
```
/*  
Name: nome_file  
Copyright: classe e nome_gruppo  
Author: componenti_gruppo  
Date: 19/03/05 08.11  
Description: breve commento  
*/  
  
int main ()      // il main deve ritornare un valore intero infatti .....  
{  
    int a, b;  
  
    a = 3;  
  
    b = 5;  
  
    a = a+b;  
  
    //..... l'ambiente inserisce automaticamente l'istruzione return 0;  
}
```


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

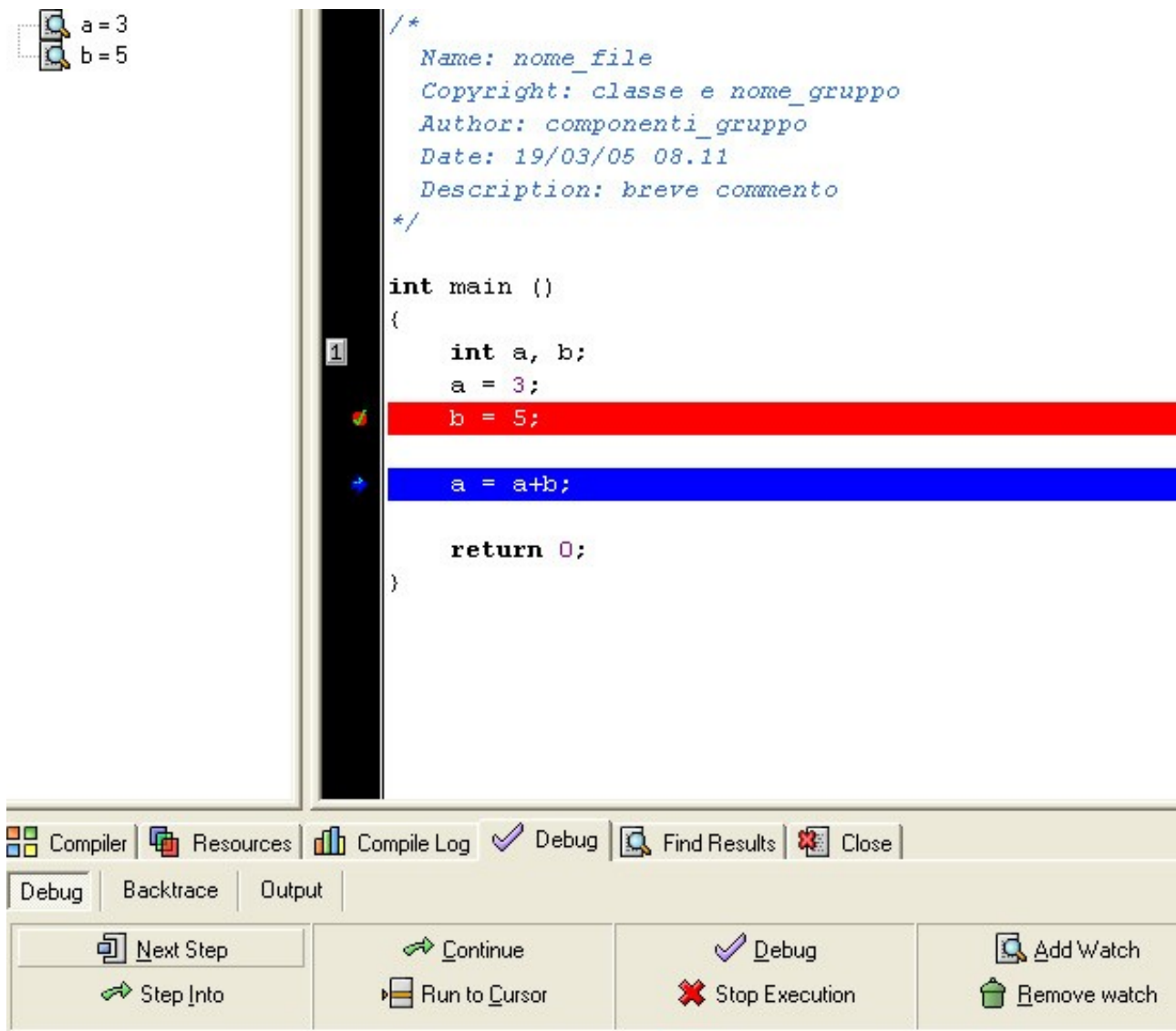
- oppure usando  **Add Watch** ed impostando il nome della variabile



**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  Next Step



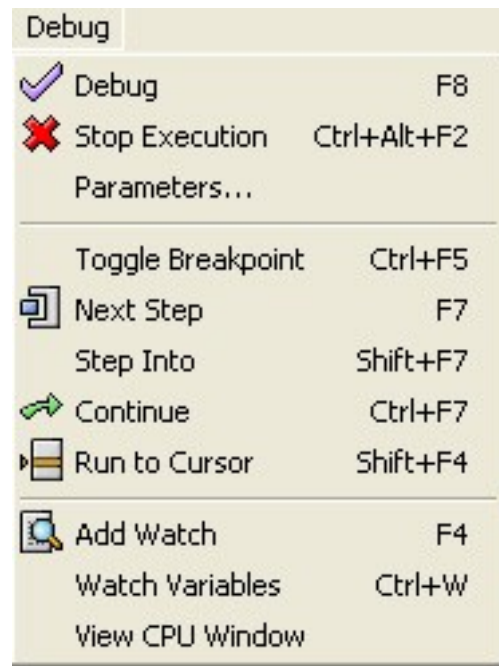
The screenshot shows a debugger window with the following elements:

- Watch Window:** Shows variables `a = 3` and `b = 5`.
- Code Editor:** Contains the following C code:

```
/*  
  Name: nome_file  
  Copyright: classe e nome_gruppo  
  Author: componenti_gruppo  
  Date: 19/03/05 08.11  
  Description: breve commento  
*/  
  
int main ()  
{  
    1  int a, b;  
      a = 3;  
      b = 5;  
      a = a+b;  
      return 0;  
}
```

The line `b = 5;` is highlighted in red, and `a = a+b;` is highlighted in blue. A cursor is positioned at the start of the `a = a+b;` line.
- Debugger Toolbar:** Includes buttons for `Next Step`, `Step Into`, `Continue`, `Run to Cursor`, `Debug`, `Stop Execution`, `Add Watch`, and `Remove watch`.

- Le opzioni di **Debug**



permettono anche di:

- vedere il **codice Assembler** con sintassi Intel oppure AT&T

**Debug → Finestra CPU**

View CPU Window

```

#include <iostream>
using namespace std;

int main ()
{
    int a,b;

    a = 3;
    b = 5;
    a = a + b;
    return 0;
}

```

**Finestra CPU**

Codice Assembler:  
Funzione: main

```

0x401390 <main>:    push    ebp
0x401391 <main+1>:    mov     ebp,esp
0x401393 <main+3>:    sub     esp,0x18
0x401396 <main+6>:    and     esp,0xfffffff0
0x401399 <main+9>:    mov     eax,0x0
0x40139e <main+14>:   add     eax,0xf
0x4013a1 <main+17>:   add     eax,0xf
0x4013a4 <main+20>:   shr     eax,0x4
0x4013a7 <main+23>:   shl     eax,0x4
0x4013aa <main+26>:   mov     DWORD PTR [ebp-12],eax
0x4013ad <main+29>:   mov     eax,DWORD PTR [ebp-12]
0x4013b0 <main+32>:   call   0x40d070 <_alloca>
0x4013b5 <main+37>:   call   0x40ccb0 <_main>
0x4013ba <main+42>:   mov     DWORD PTR [ebp-4],0x3
0x4013c1 <main+49>:   mov     DWORD PTR [ebp-8],0x5
0x4013c8 <main+56>:   mov     edx,DWORD PTR [ebp-8]
0x4013cb <main+59>:   lea    eax,[ebp-4]
0x4013ce <main+62>:   add     DWORD PTR [eax],edx
0x4013d0 <main+64>:   mov     eax,0x0
0x4013d5 <main+69>:   leave
0x4013d6 <main+70>:   ret

```

Sintassi Assembler:  
 AT&T  Intel

Registri:  
EAX: 0x0  
EBX:  
ECX:  
EDX:  
ESI:  
EDI:  
EBP:  
ESP:  
EIP:  
CS:  
DS:  
SS:  
ES:

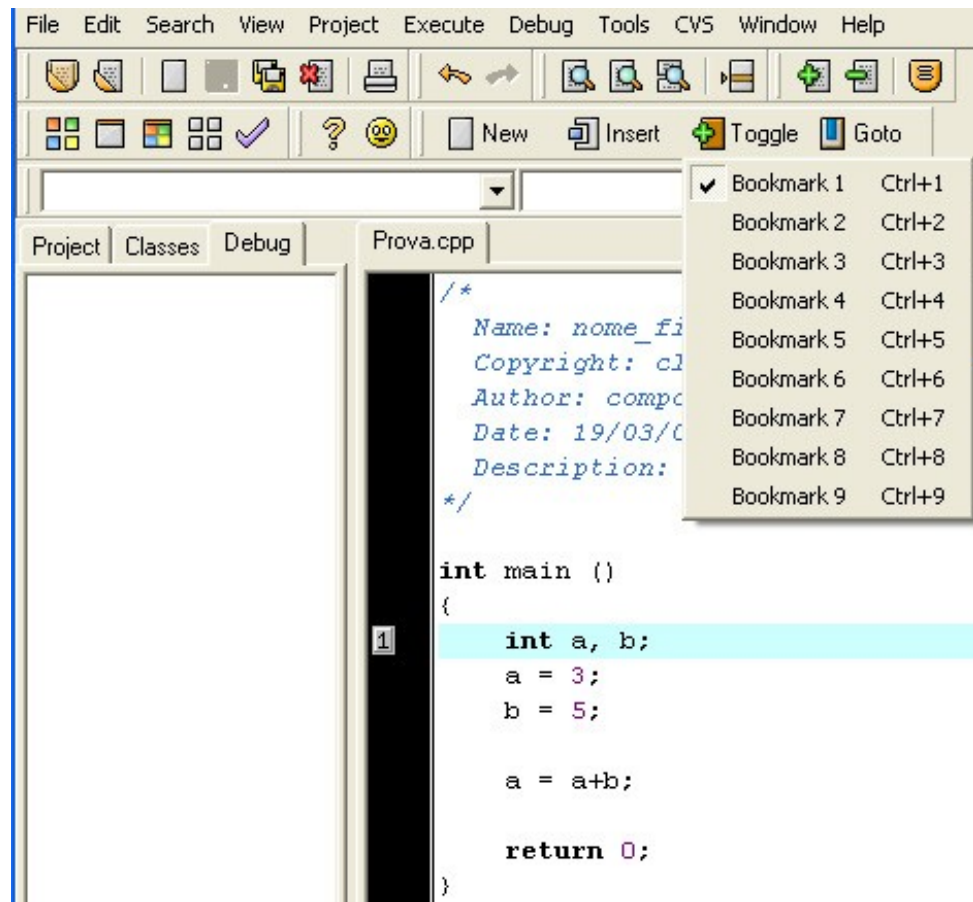
Chiudi

- 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

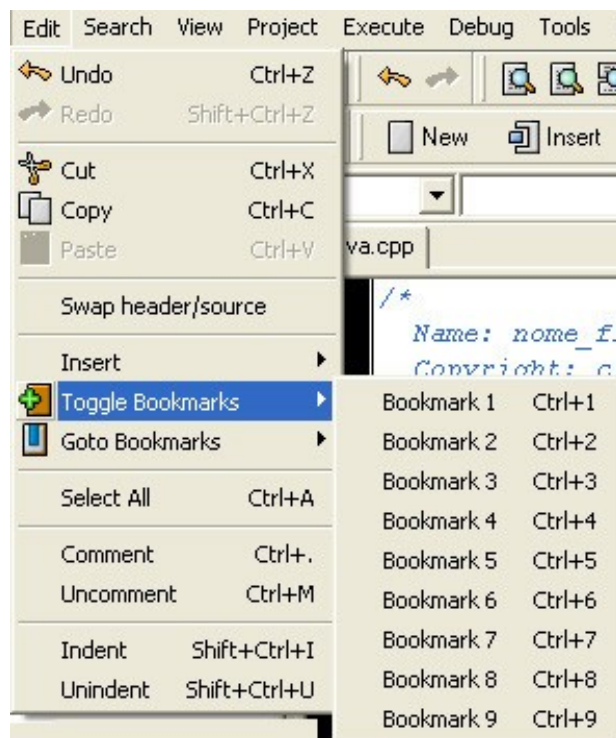



In fase di editing si può inserire<sup>3</sup> almeno un **bookmark**: si posiziona il cursore sulla riga voluta

e si seleziona l'icona **Toggle**



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



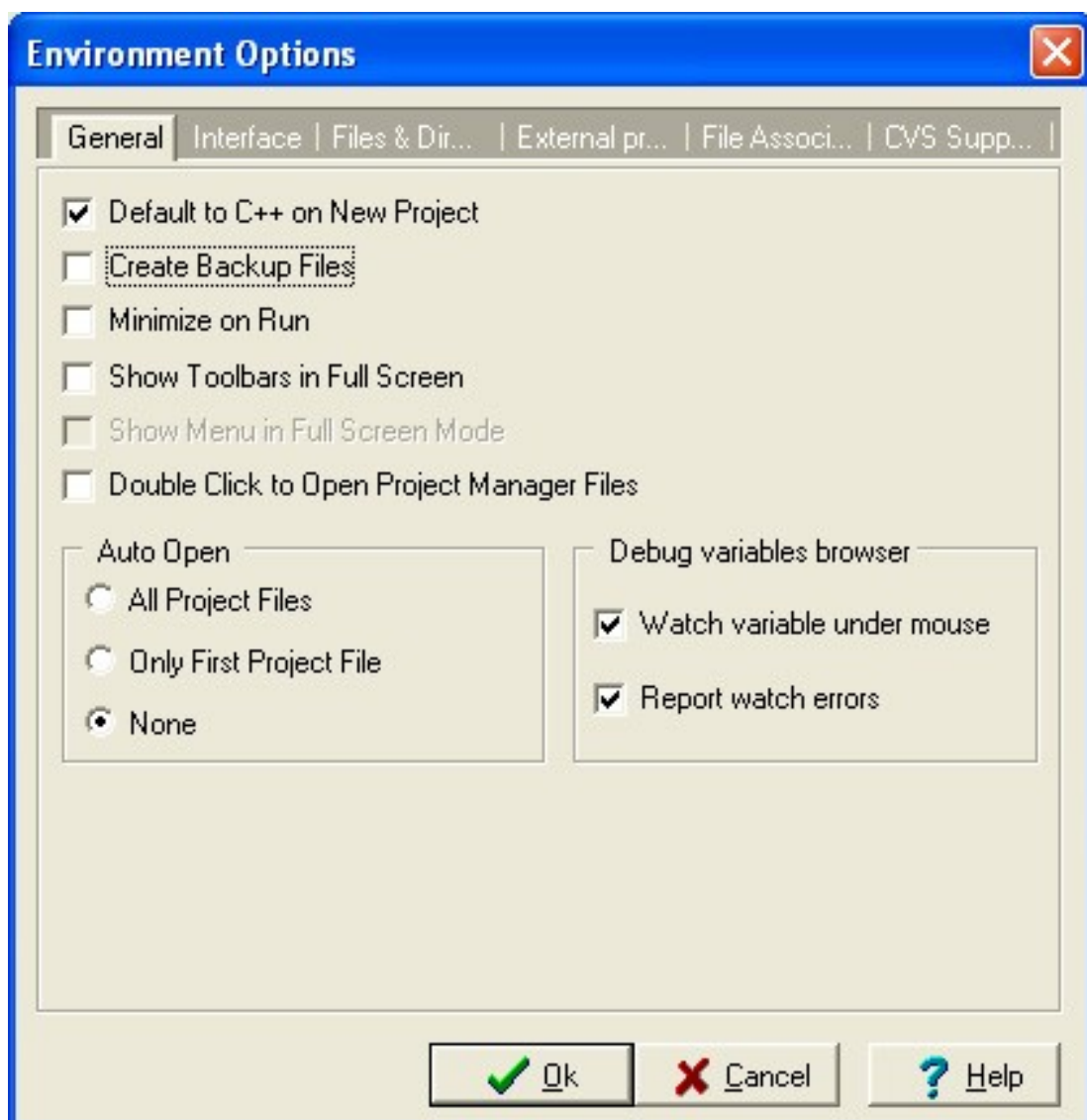
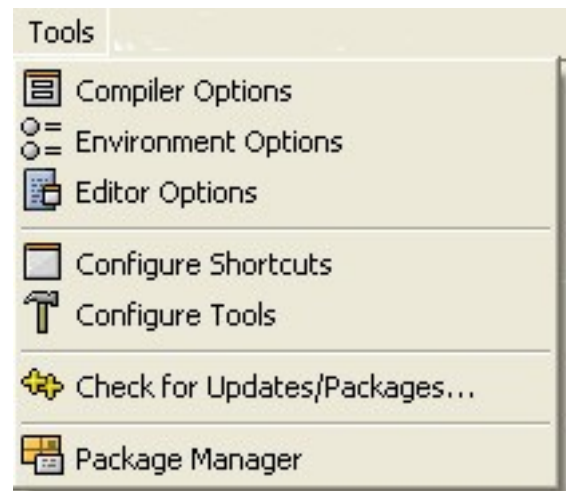
<sup>3</sup> Per velocizzare il posizionamento con *Vai al segnalibro*  quando si attiva l'opzione "Esegui dal Cursore" o "Run to Cursor"



## Opzioni dell'ambiente

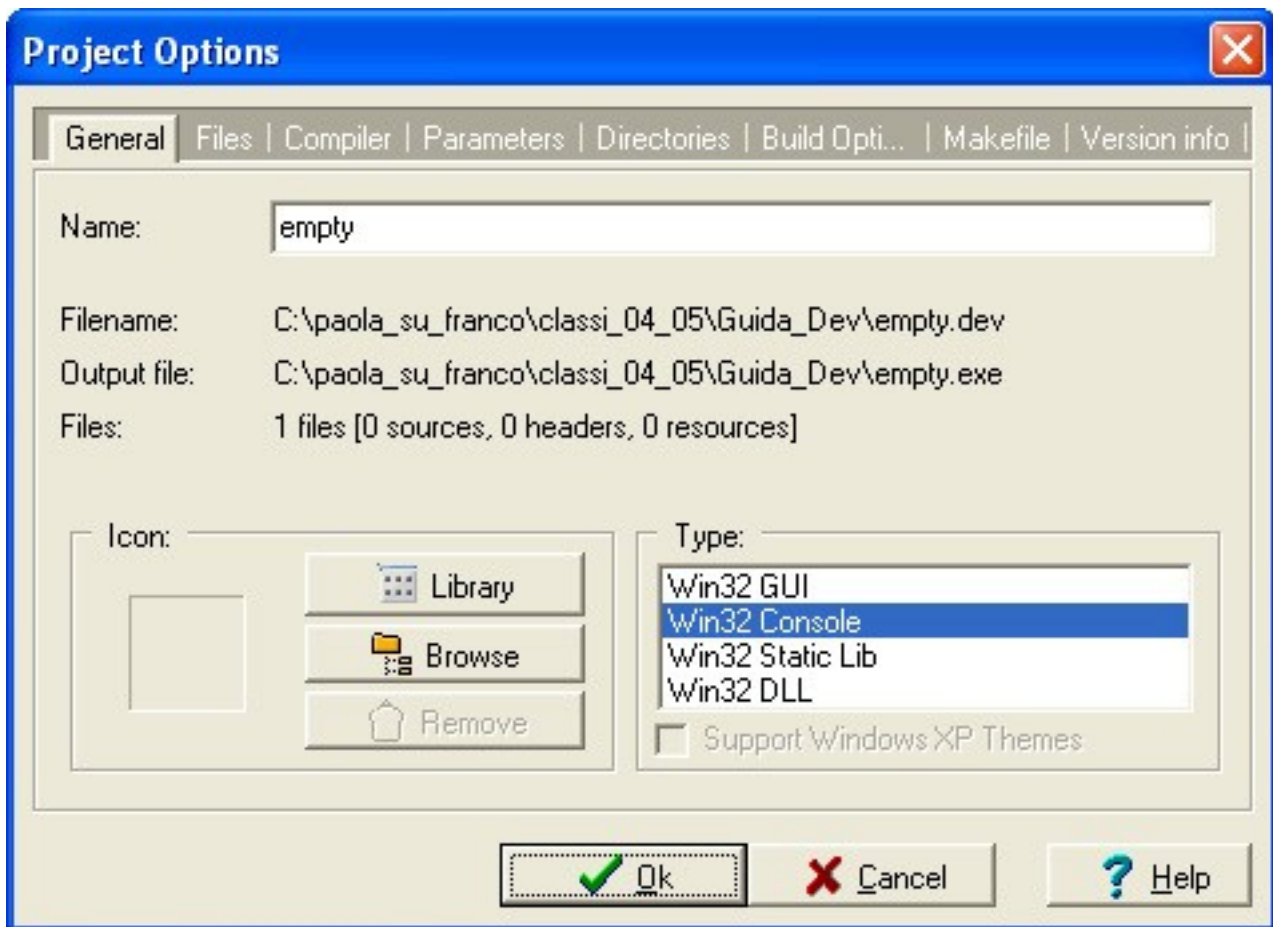
L'installazione scelta: **linguaggio C++**

Verificabile con **Tools → Environment Options**



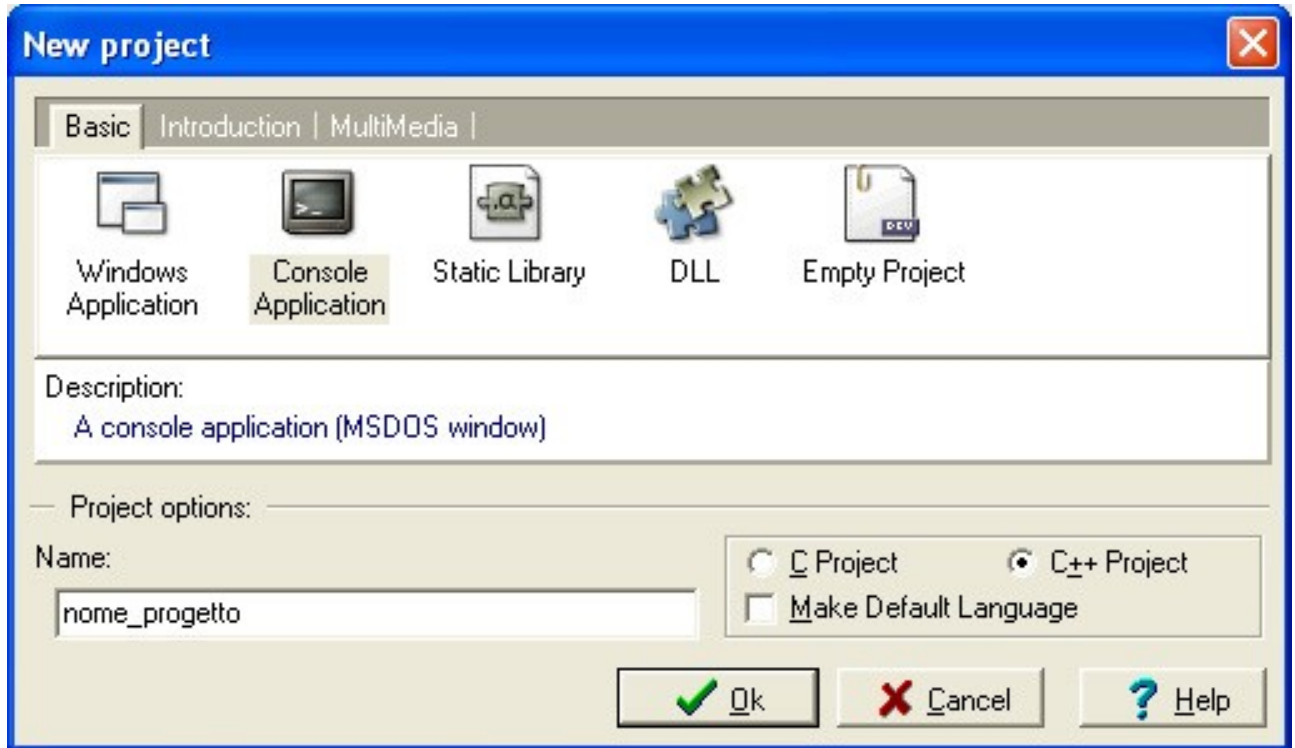
## Opzioni generali di progetto

Tipo di applicazione: **Win32 Console**

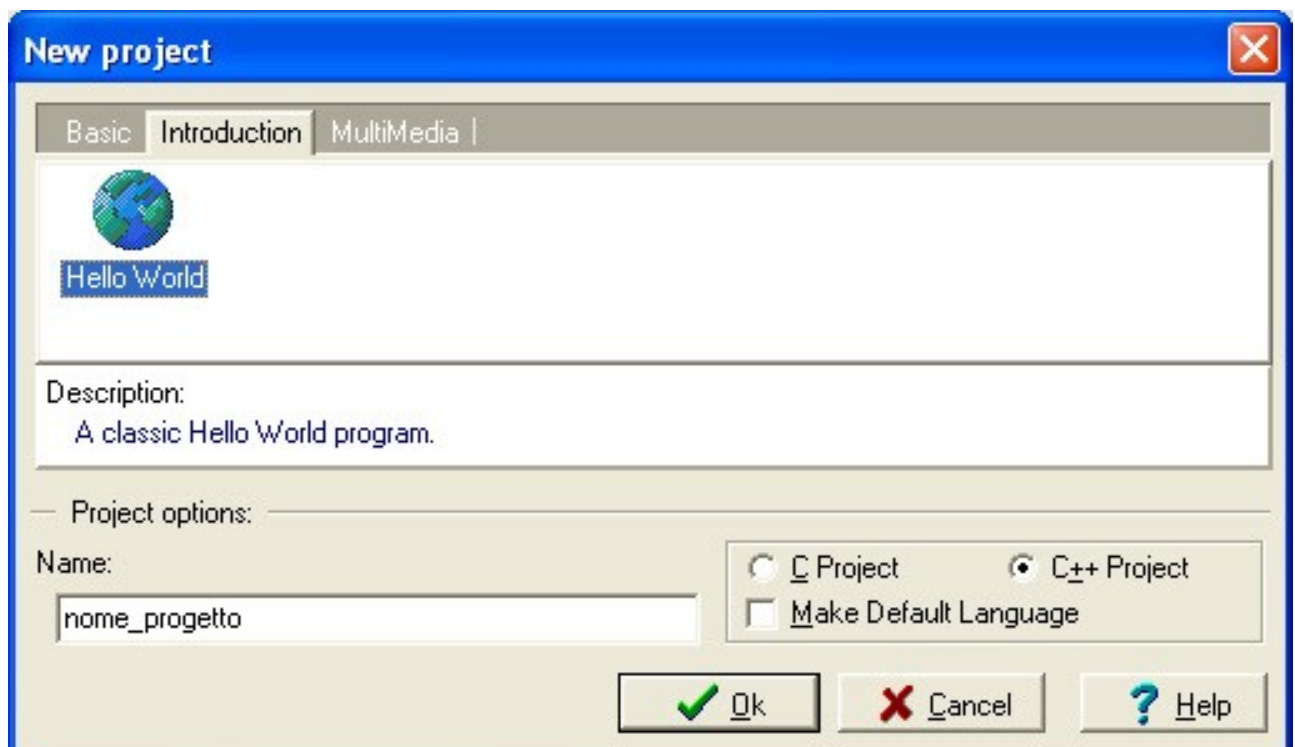


Altre tipologie di progetto dove automaticamente nelle Project Options viene selezionata un'applicazione tipo: **Win32 Console**

- **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<sup>4</sup> 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 **Minimalist GNU for Windows** cioè un compilatore C/C++ per windows<sup>5</sup> e la libreria standard **glibc** (GNU libraries for C) acquisendone implicitamente la compatibilità POSIX<sup>6</sup> (Portable Open System Interface) dove la X è un chiaro riferimento al sistema UNIX.



<sup>4</sup> 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

<sup>5</sup> La home page del sito è <http://www.mingw.org/>

<sup>6</sup> Sul finire degli anni '80 l' Institute of Electrical and Electronics Engineers (**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.