Introduzione

Il corso contenuto in questa parte riguarda i concetti elementari della programmazione, al livello minimo di astrazione possibile, utilizzando il linguaggio C per la messa in pratica degli algoritmi. Il corso è «basilare», ma gli argomenti trattati non sono così semplici come il termine potrebbe fare supporre.

Gli argomenti del corso sono già trattati in altri capitoli dell'opera, ma qui, in più, si inseriscono degli esercizi corretti.[1]

Per svolgere il corso correttamente è indispensabile fare tutti gli esercizi, verificando le soluzioni. Se il corso è guidato da un tutore, è bene presentarsi sempre alle lezioni avendo già studiato gli argomenti che devono essere trattati e avendo fatto gli esercizi indicati.

Programma didattico

Il corso, se assistito da un tutore, prevede l'impiego di circa 45 ore, di cui, almeno otto da dedicare alle verifiche (due ore di verifica per modulo, più due ore aggiuntive per una verifica di recupero complessiva).

Modulo 1

- **sistemi di numerazione**
  - decimale
  - binario
  - ottale
  - esadecimale
  - conversioni numeriche intere

  - conversioni numeriche intere tra binario, ottale e esadecimale
- **operazioni aritmetiche elementari in binario**
  - complemento a uno
  - complemento a due
  - somma binaria
  - sottrazione binaria
  - rappresentazione dei numeri interi con segno
- **operazioni elementari all'interno della CPU**
  - aumento e riduzione delle cifre binarie di un numero intero senza segno
  - aumento e riduzione delle cifre binarie di un numero intero con segno
  - somme con i numeri interi con segno
  - somme e sottrazioni con i numeri interi senza segno
  - scorrimento logico (senza segno)
  - scorrimento aritmetico (con segno)
  - rotazione
  - AND
  - OR
  - XOR
  - NOT
- **organizzazione della memoria**
  - pila dei dati o *stack* (cenni)
  - chiamata di funzioni e passaggio degli argomenti attraverso la pila (cenni)
  - variabili scalari

  - array
  - stringhe
  - puntatori
  - ordine dei byte

Modulo 2

- **primo approccio al linguaggio C**
  - commenti, istruzioni, raggruppamenti
  - compilazione
  - emissione di messaggi testuali
  - sospensione dell'esecuzione del programma in attesa della pressione di [*Invio*]
  - costruzione del primo programma che emette un messaggio e attende la pressione di [*Invio*] per terminare
- **tipi principali del linguaggio C**
  - tipi scalari primitivi: char, short int, int, long int, float, double
  - tipi scalari primitivi: distinzione tra presenza e assenza del segno
  - costanti letterali
  - dichiarazione di variabili scalari
  - il tipo void
- **operatori ed espressioni del linguaggio C**
  - operatori aritmetici
  - operatori di confronto
  - operatori logici
  - operatori binari

  - cast (conversione di tipo)
  - espressioni multiple
- **strutture di controllo di flusso del linguaggio C**
  - if
  - switch
  - while
  - for
- **funzioni del linguaggio C**
  - funzione '`main (void)`'
  - prototipo
  - descrizione della funzione
  - valore restituito dalla funzione
  - valore restituito dal programma

Modulo 3

- **puntatori in C**
  - espressioni a cui si assegnano dei valori (*lvalue*)
  - dichiarazione di una variabile puntatore
  - dereferenziazione di un puntatore
  - *big endian*, *little endian* e puntatori
  - puntatori come parametri di una funzione
- **array in C**
  - dichiarazione di un array
  - selezione di un elemento all'interno di un array all'interno delle espressioni
  - array a più dimensioni
  - uso del ciclo '`for`' per la scansione di un array

  - relazione tra array e puntatori
  - dereferenziazione di un puntatore come se fosse un array
  - array come parametri di una funzione
  - aritmetica dei puntatori
  - stringhe
- **puntatori di puntatori**
  - dichiarazione e dereferenziazione
  - puntatori a più dimensioni, ovvero: array di puntatori
  - parametri della funzione ***main()***

Strumenti per la compilazione

«

Per potersi esercitare nell'uso del linguaggio C, è possibile avvalersi di un servizio *pastebin* completo, come *http://codepad.org* e *http://ideone.com* . A questi servizi ci si deve iscrivere, in modo da poter salvare i propri esercizi.

Se si dispone di un elaboratore completo, si può utilizzare un compilatore vero e proprio. I sistemi GNU e derivati, dispongono di norma del compilatore GNU C, ma in generale ogni sistema Unix dovrebbe consentire di compilare un programma utilizzando semplicemente il comando '`cc`', a cui si fa riferimento inizialmente nel capitolo del corso che introduce alla compilazione stessa.

Per compilare un programma C in un sistema operativo come MS-Windows, occorre uno strumento apposito. Nel caso di MS-Windows si suggerisce l'uso di Dev-C++ che è molto facile da installare e da usare, pur non offrendo il classico '`cc`' da riga di comando. Nelle figure successive viene mostrato, intuitivamente, il procedimento per creare un file, compilarlo ed eseguirlo.

Figura u5.1. Aspetto di Dev-C++ dopo l’avvio.

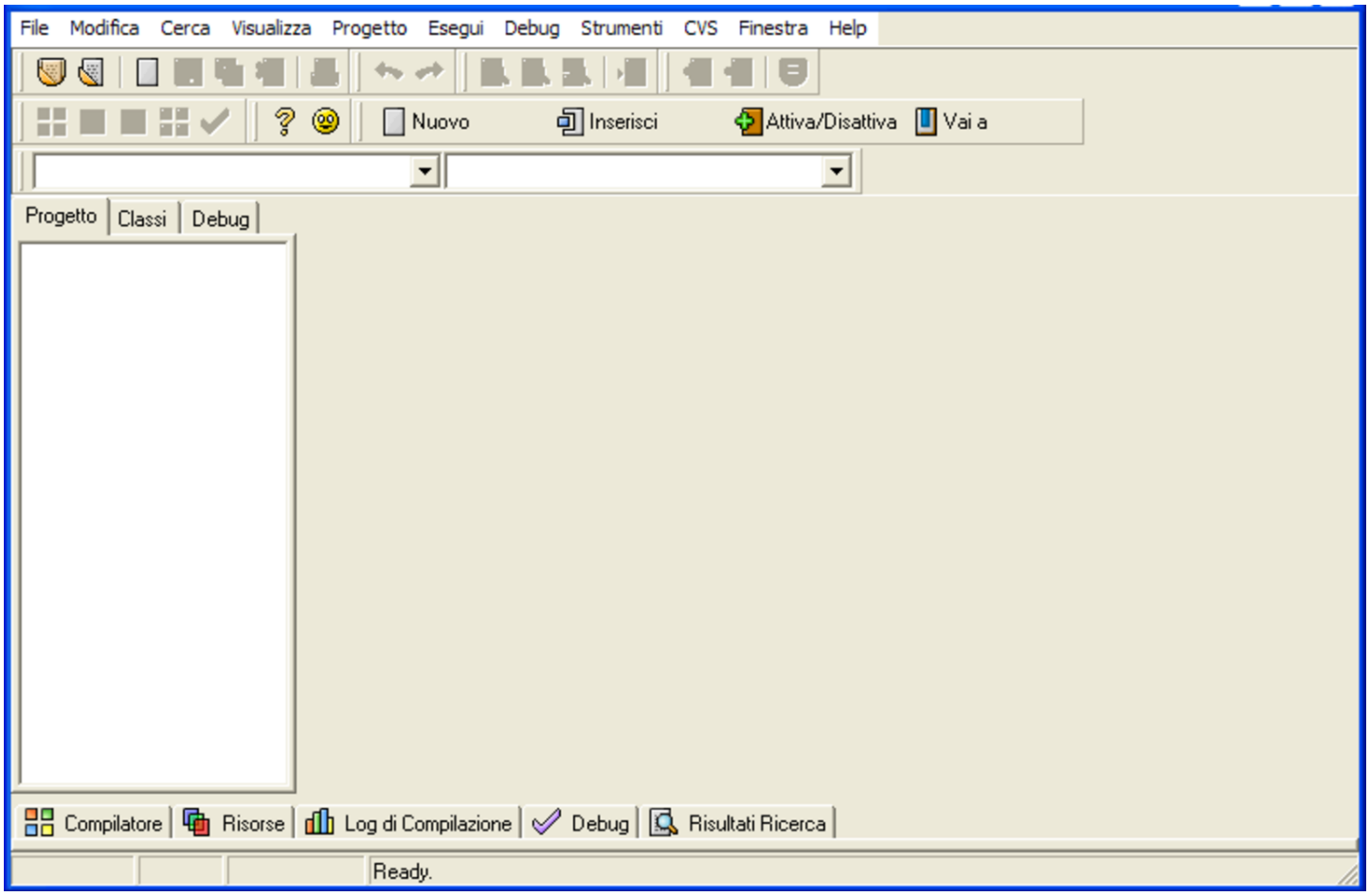

Figura u5.2. Creazione di un file sorgente nuovo.

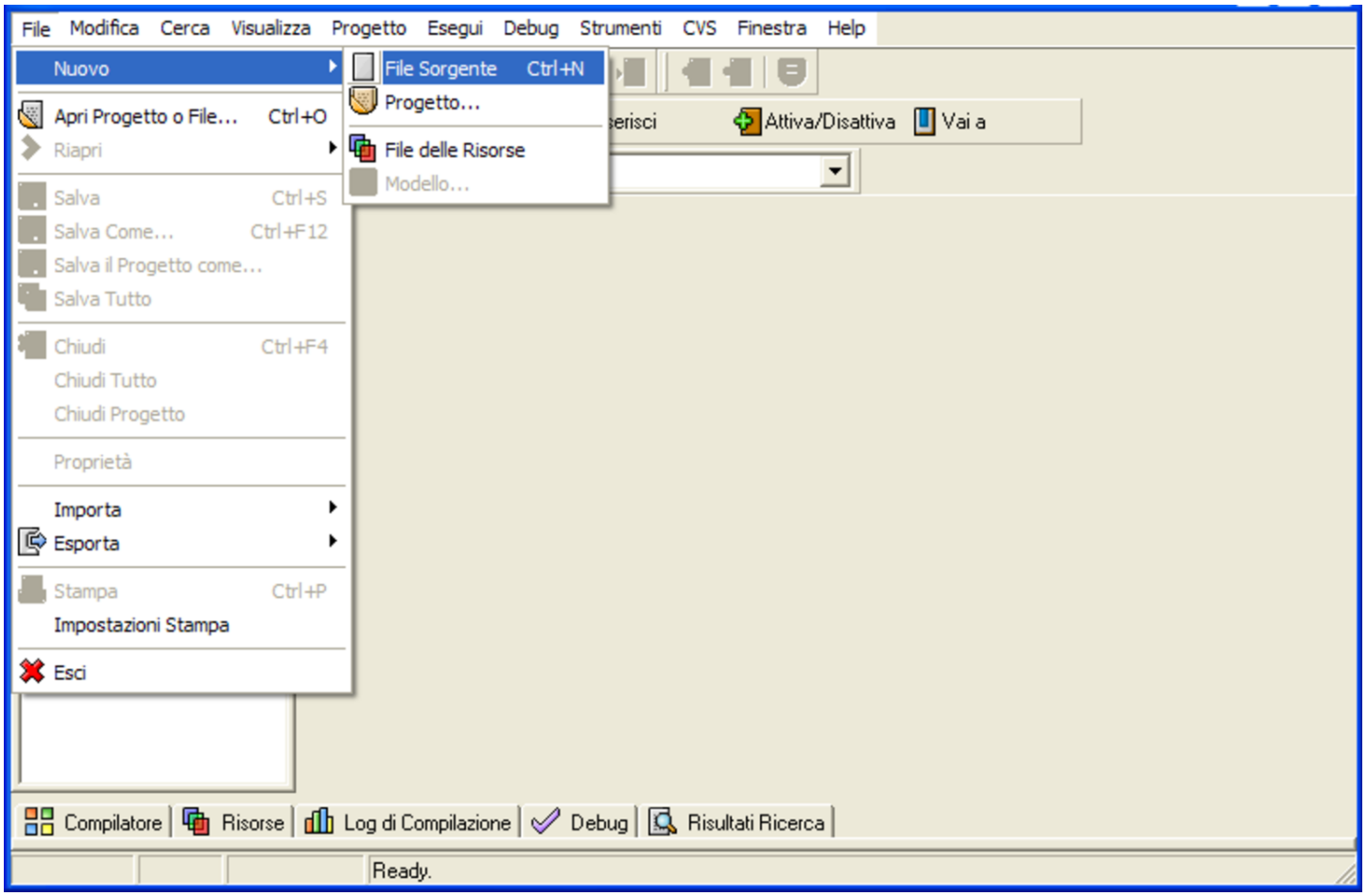

Figura u5.3. Un file che mostra un messaggio, attende la pressione di [*Invio*] e termina di funzionare.

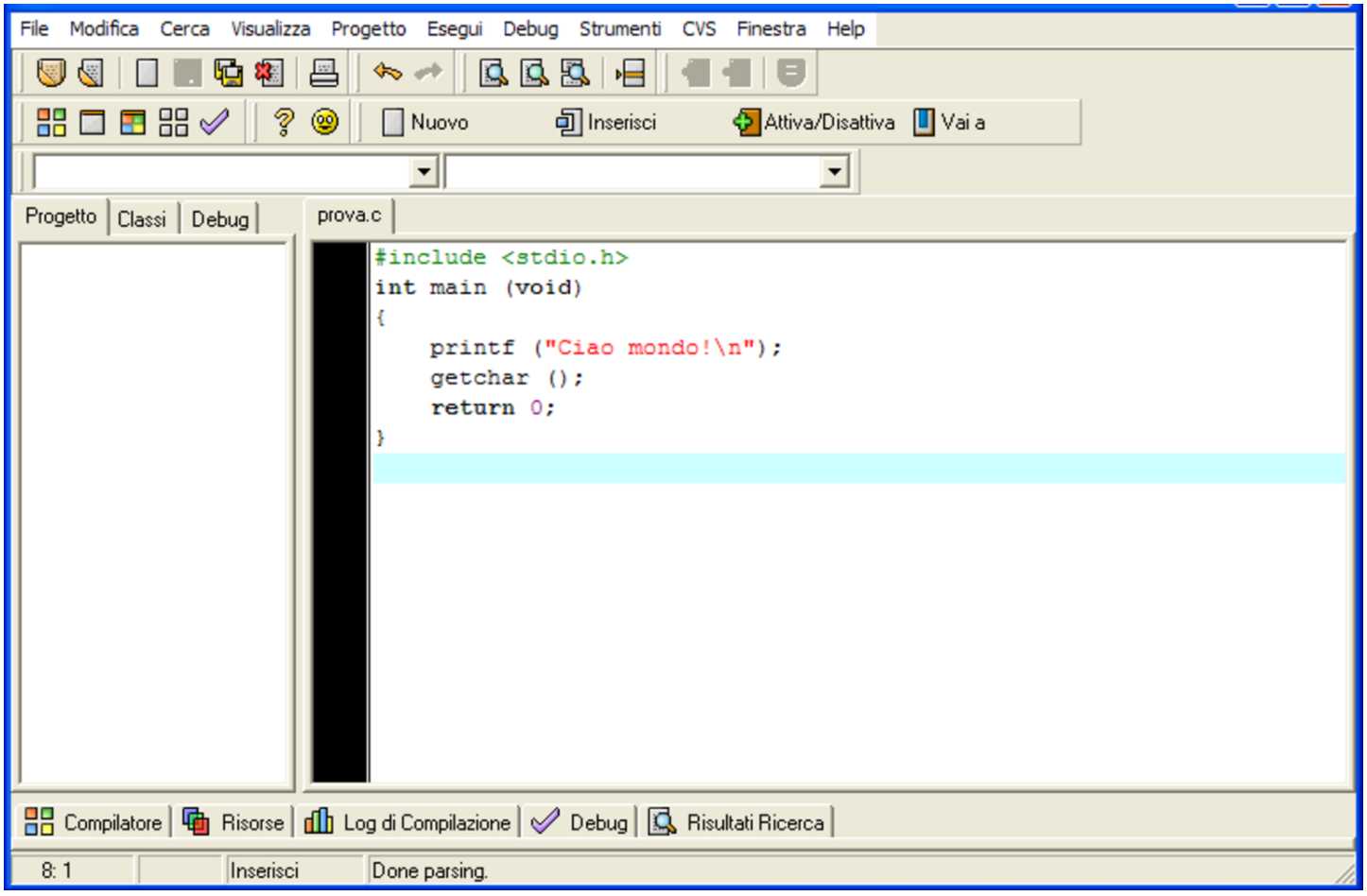

Figura u5.4. Compilazione.

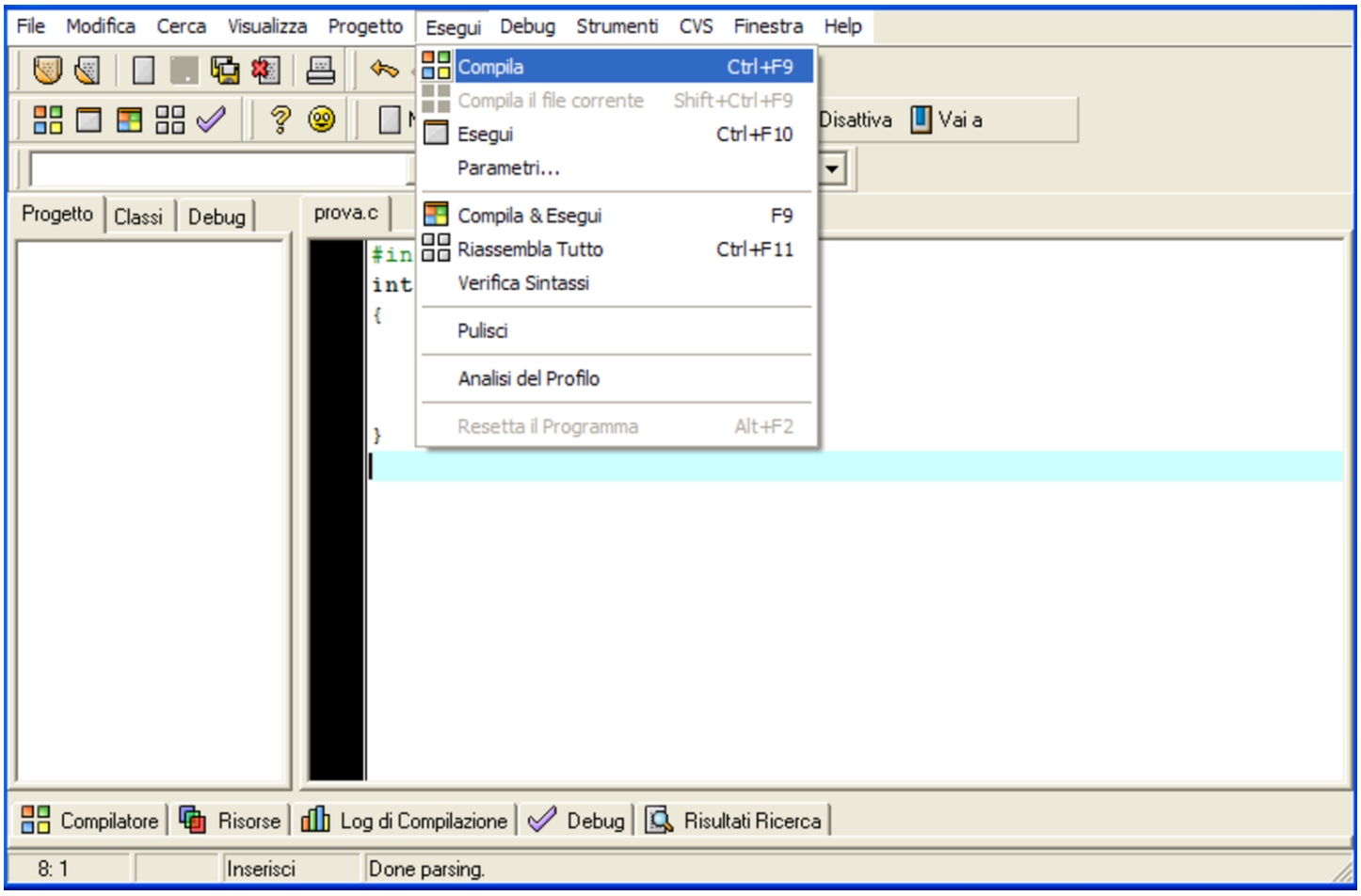

Figura u5.5. Esecuzione.

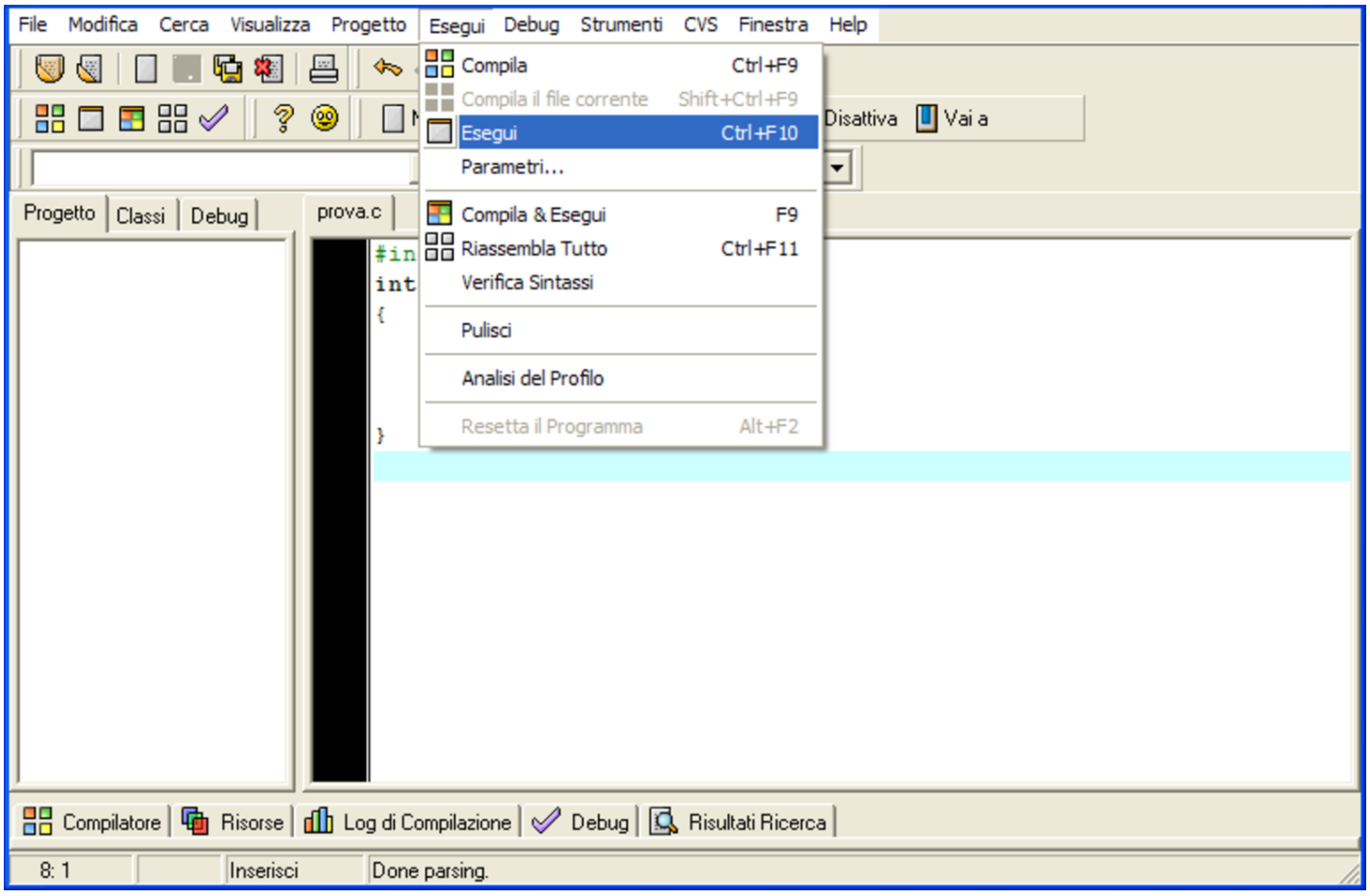

Figura u5.6. Finestra testuale da dove si vede l'emissione del messaggio del programma. Basta premere [*Invio*] per fare terminare il funzionamento del programma e lasciare così che la finestra si chiuda.

Riferimenti:

- *Codepad*, *http://codepad.org*
- *Ideone.com*, *http://ideone.com*
- BloodshedSoftware, *Dev-C++*, *http://www.bloodshed.net/devcpp.html* , *http://www.bloodshed.net/dev/* , *http://sourceforge.net/projects/dev-cpp/*

[1] Va tenuta sempre in considerazione la possibilità che alcune soluzioni o correzioni non siano esatte, pertanto, in caso di dubbio, va consultato un docente o comunque una persona competente.