

1- Ambiente di Sviluppo

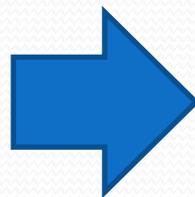
INTRODUZIONE, LINGUAGGIO, HANDS ON

Giuseppe Cirillo
g.cirillo@unina.it

Il linguaggio C

BCPL

1966
Martin Richards (MIT)
Semplificando CPL



B

1970
Ken Thompson

1972-Dennis Ritchie
1978-Definizione
1990-ANSI C

C



- usato per sviluppare unix, ms-dos etc
- Alla base di molti applicativi
- Poche Keyword
- Librerie standard
- Accesso anche a basso livello
- Efficiente gestione indirizzi
- Modularità
-

Linguaggi Compilati vs. Interpretati

```
object matchCase = true;  
object matchWholeWord = true  
object matchWildCards = fals  
object matchSoundsLike = fal  
object nmatchAllWordForms =  
object forward = true;  
object format = false;  
object matchKashida = false;  
object matchDiacritics = fal  
object matchAlefHamza = fals  
object matchControl = false;
```

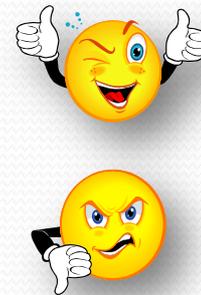
compilatore

compilatore

compilatore



Es. C, C++



Performance e Personalizzazione

Portabilità

```
object matchCase = true;  
object matchWholeWord = true  
object matchWildCards = fals  
object matchSoundsLike = fal  
object nmatchAllWordForms =  
object forward = true;  
object format = false;  
object matchKashida = fa  
object matchDiacritics = fal  
object matchAlefHamza = fals  
object matchControl = false;
```

compilatore



Interprete
(macchina virtuale)



Performance e Personalizzazione

Portabilità

Es. Java

Compilato o interpretato?



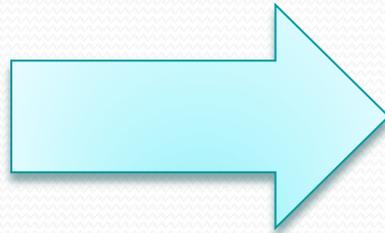
© Solarbotica Ltd. WWW.SOLARBOTICA.COM



Che significa “compilare”?



Sorgente



Oggetto

Wikipedia:

“ un **compilatore** è un programma che traduce una serie di istruzioni scritte in un determinato linguaggio di programmazione (codice sorgente) in istruzioni di un altro linguaggio (codice oggetto). Questo processo di traduzione si chiama **compilazione.**”

Gli “attori” in gioco

1

PRE-PROCESSORE

- Toglie i **commenti**
- Interpreta **direttive** al pre-processor di inclusione, compilazione condizionale, macro... (es. #include, #define ecc)

2

COMPILATORE

Traduce il codice sorgente in codice **Sorgente Assembly**

3

ASSEMBLER

Produce **codice assembler** (unico che la cpu capisce) con i relativi offset e li salva in file oggetto (uno per ogni unità di compilazione)

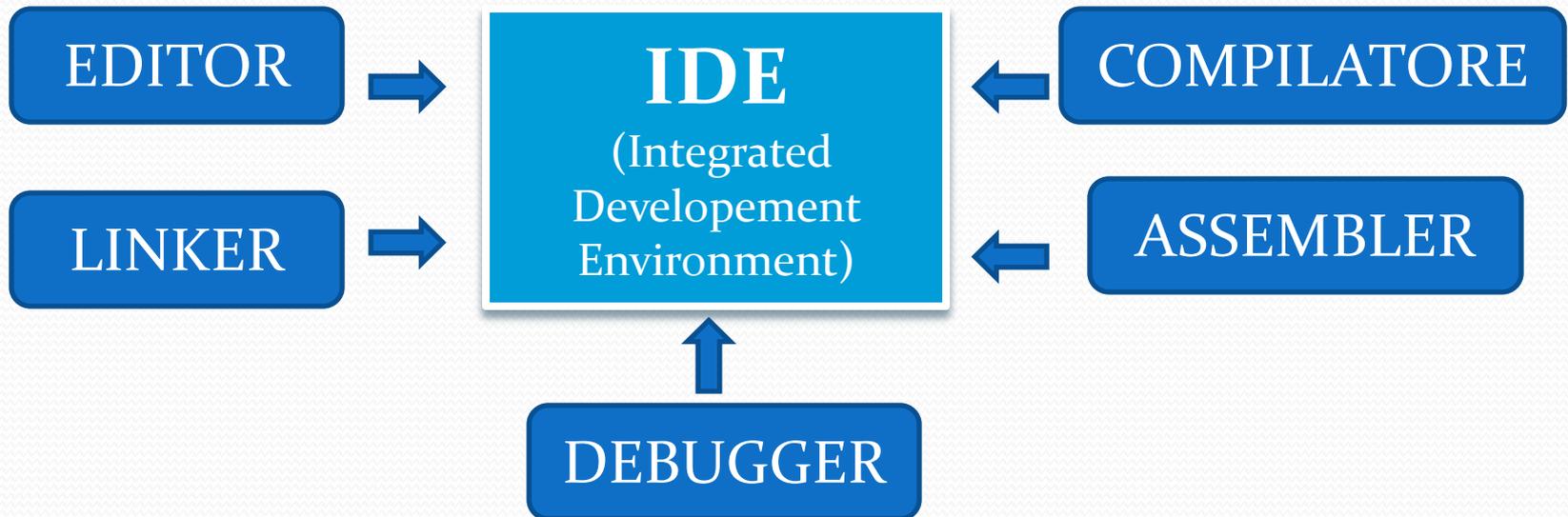
4

LINKER

Prende uno o più file oggetto (custom o di libreria) e genera un singolo **eseguibile**.

Il DEV C++: chi è e cosa fa?

- Sviluppato da bloodshed (www.bloodshed.net) è uno degli IDE gratuiti più utilizzati (insieme ad eclipse con supporto c++)
- Compilatore GCC/Mingw
- Supporto file singoli o progetti
- Etc. etc.



NB. Installazione e files in percorsi “brevis” (<= 8 caratteri) es. c:\esame\... ,

I files coinvolti nel processo

Estensione	Descrizione	Esempio
.H	Header File Contiene le intestazioni per funzioni di libreria (anche definite dall'utente) Non vengono compilati e non devono contenere istruzioni o procedure	Stdio.h
.C / .CPP	File sorgente in linguaggio c che devono essere preprocessati (compilati, assemblati e linkati)	Main.c
.i	File sorgente che non deve essere pre-processato ma solo compilato	Main.i
.s	Codice Assembler che viene passato all'assembler	Main.s
.O	File oggetto generati dall'assembler che devono essere "linkati" dal linker	Main.o
.EXE	File eseguibile generato dal linker, pronto per essere caricato ed eseguito dal sistema operativo	Main.exe

Colori e sintassi

- L'editor del DEV C++ (come molti editor per linguaggi di programmazione) è in grado di riconoscere la sintassi del linguaggio e di evidenziarla:

ESEMPIO	DESCRIZIONE	COLORE
<code>#include</code>	Direttive al pre-processore	verde
<code>//commento</code>	Commenti al codice (ignorati dal compilatore, servono al programmatore)	azzurro
<code>int</code>	Parole riservate (costrutti, tipi ecc..)	Nero bold
<code>"ciao"</code>	Stringhe di caratteri	rosso
<code>34</code>	Numeri	violetto
<code>c = ..</code>	Linea con errore	Marrone ev.

Indentazione

```
if (a==1) {  
printf("ciao ciao \n");  
}else {  
if (b==0) {  
printf("bla bla \n");  
} else {  
printf("boh boh \n");  
}  
}
```



```
if (a==1) {  
printf("ciao ciao \n");  
}else {  
    if (b==0) {  
printf("bla bla \n");  
    } else {  
        printf("boh boh \n");  
    }  
}
```

- Per una migliore leggibilità del codice (da parte dell'uomo) è buona norma indentare correttamente il codice scritto. (l'IDE ci aiuta)

Direttive al Preprocessore

- Le principali direttive al pre-processore sono:

#define

#define nome valore

Esempio

#define MAX 10

#include

#include <libreria>

Esempio

#include <stdlib.h>

#include "mioheader.h"

(in caso di header personali)

- Definisce una MACRO (simbolo) valida per l'intero file.
- Per convenzione si scrive in MAIUSCOLO
- Il pre-processore sostituisce il valore al nome
- E' utile, ad esempio, per parametri collettivi.
- Segnala al pre-processore le librerie da includere per trovare le funzioni utilizzate.
- Si utilizza sia per librerie del linguaggio sia per header scritti dal programmatore

Le librerie

- Le librerie di base da conoscere per scrivere piccoli programmi in C sono:

Stdio.h

Standard Input output
libreria standard di C
(compatibile su tutte le
piattaforme per cui esiste un
compilatore C)

- Stampa a video di stringhe (printf, fprintf)
- Gestione dei files
- Operazioni di input da console (scanf, fscanf..)

Stdlib.h

Standard Library

- Chiamate al sistema operativo (system)
- Conversione di numeri e stringhe
- Matematica basilare(rand, abs, div...)

Il main

- Per poter scrivere un programma in C (o c++, java ecc..) è necessario specificare un punto di ingresso (entry point).
- L'**entry point** corrisponde ad un indirizzo di memoria contenente la porzione di codice di programma destinazione di una chiamata a funzione
- In C la funzione main() indica al compilatore la funzione principale da cui partire per l'esecuzione (ed il linking...)

Es.

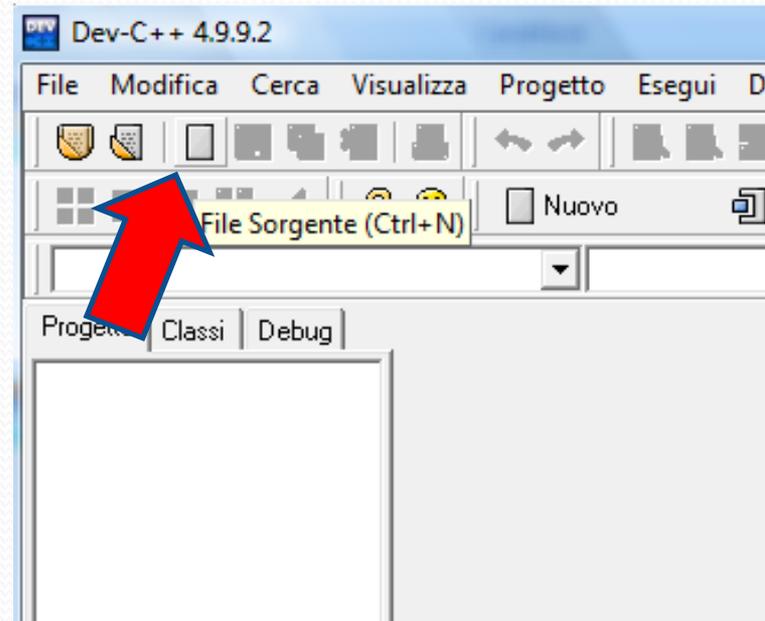
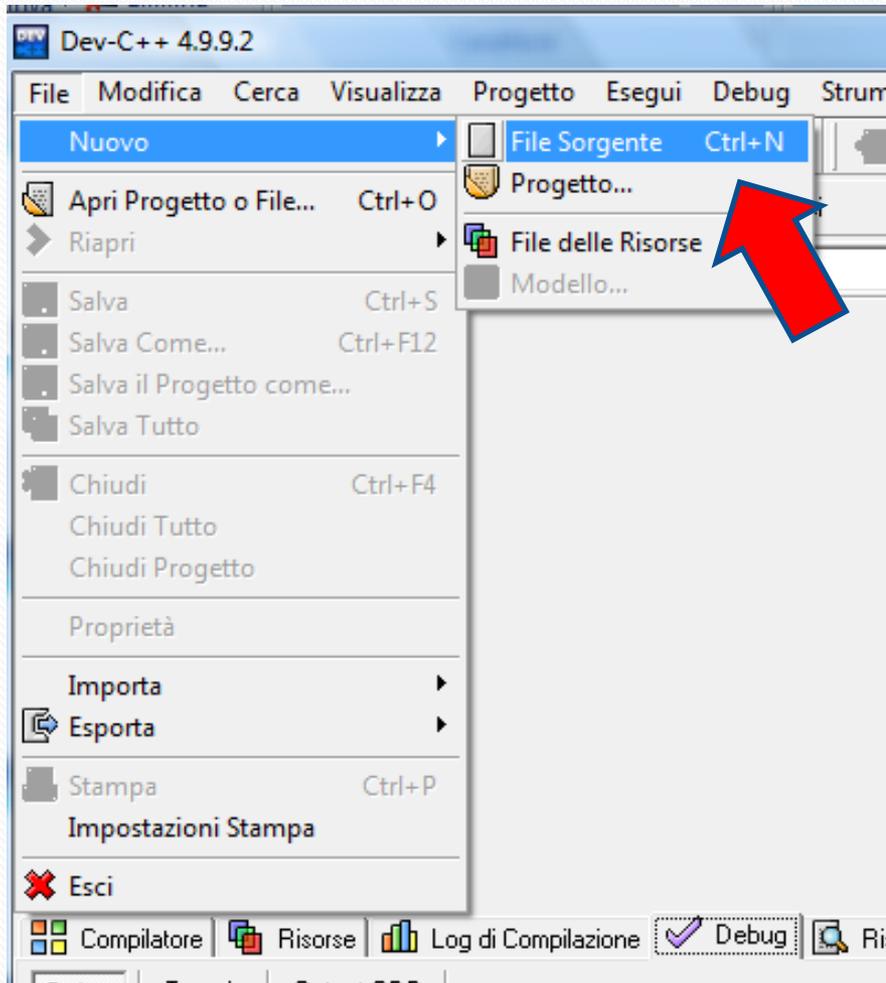
```
➔ int main()
  {
    int a,b=0;
    if (a==b) {
        //sono uguali
    }
    return 0; }

```

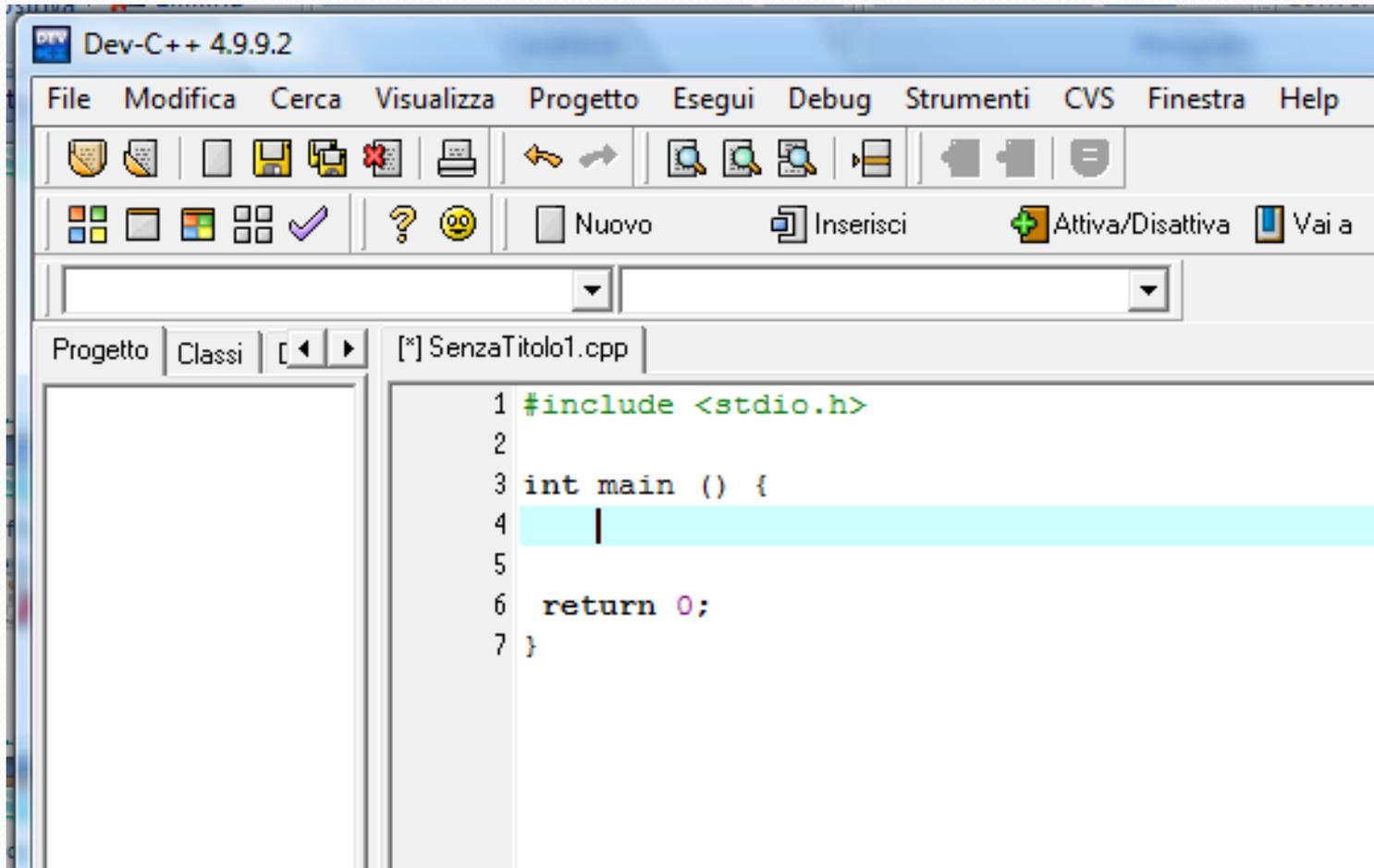


Indirizzo 1	Istruz..
Indirizzo 2	Istruz..
Indirizzo 5	Istruz..
Indirizzo 4	Istruz..
Indirizzo6	Istruz..
Indirizzo 7	Istruz..

Dev c++ - Creazione nuovo file



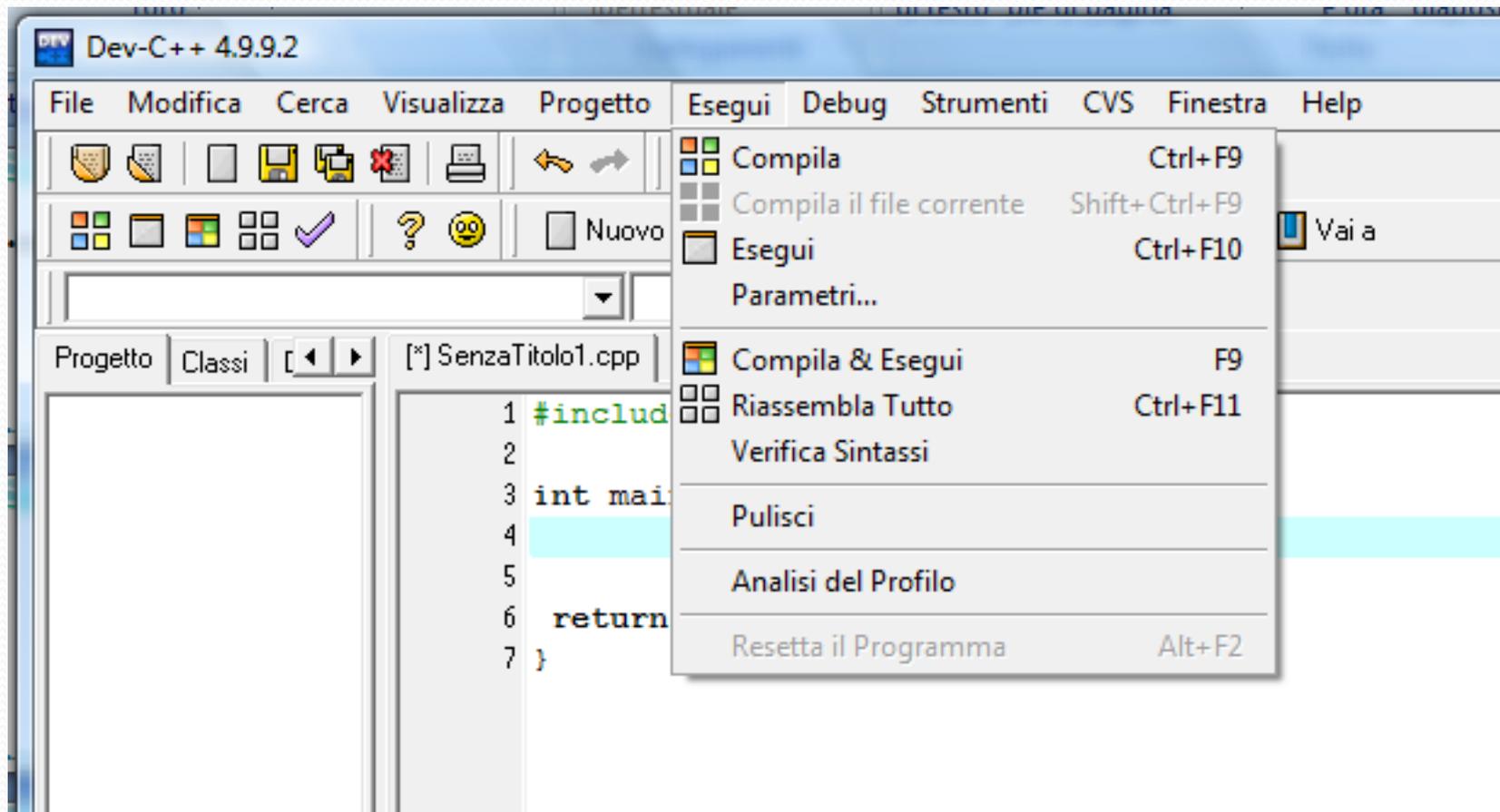
Dev c++ - Scrittura del codice



The image shows a screenshot of the Dev-C++ 4.9.9.2 IDE. The window title is "Dev-C++ 4.9.9.2". The menu bar includes "File", "Modifica", "Cerca", "Visualizza", "Progetto", "Esegui", "Debug", "Strumenti", "CVS", "Finestra", and "Help". The toolbar contains various icons for file operations, editing, and execution. The main editor area shows a C++ code file named "SenzaTitolo1.cpp" with the following code:

```
1 #include <stdio.h>
2
3 int main () {
4     |
5
6     return 0;
7 }
```

Dev c++ - Compila & Esegui

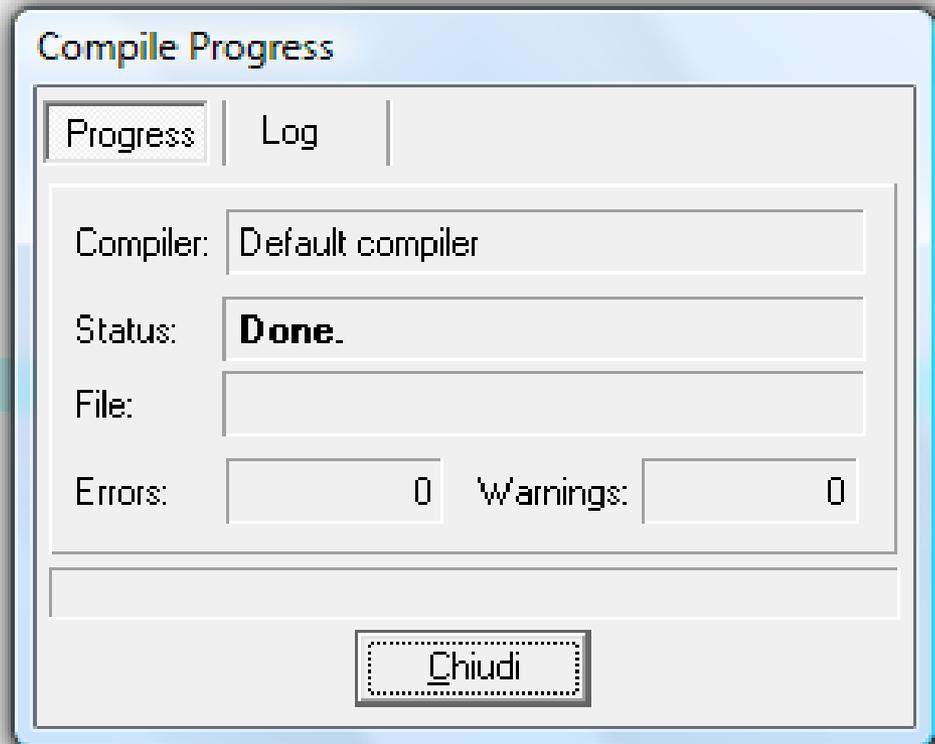


Esempio 1: Somma

esempio1.c

```
1
2
3 int main () {
4     int a;
5     int b;
6
7     a+b;
8
9     return 0;
10 }
11
```

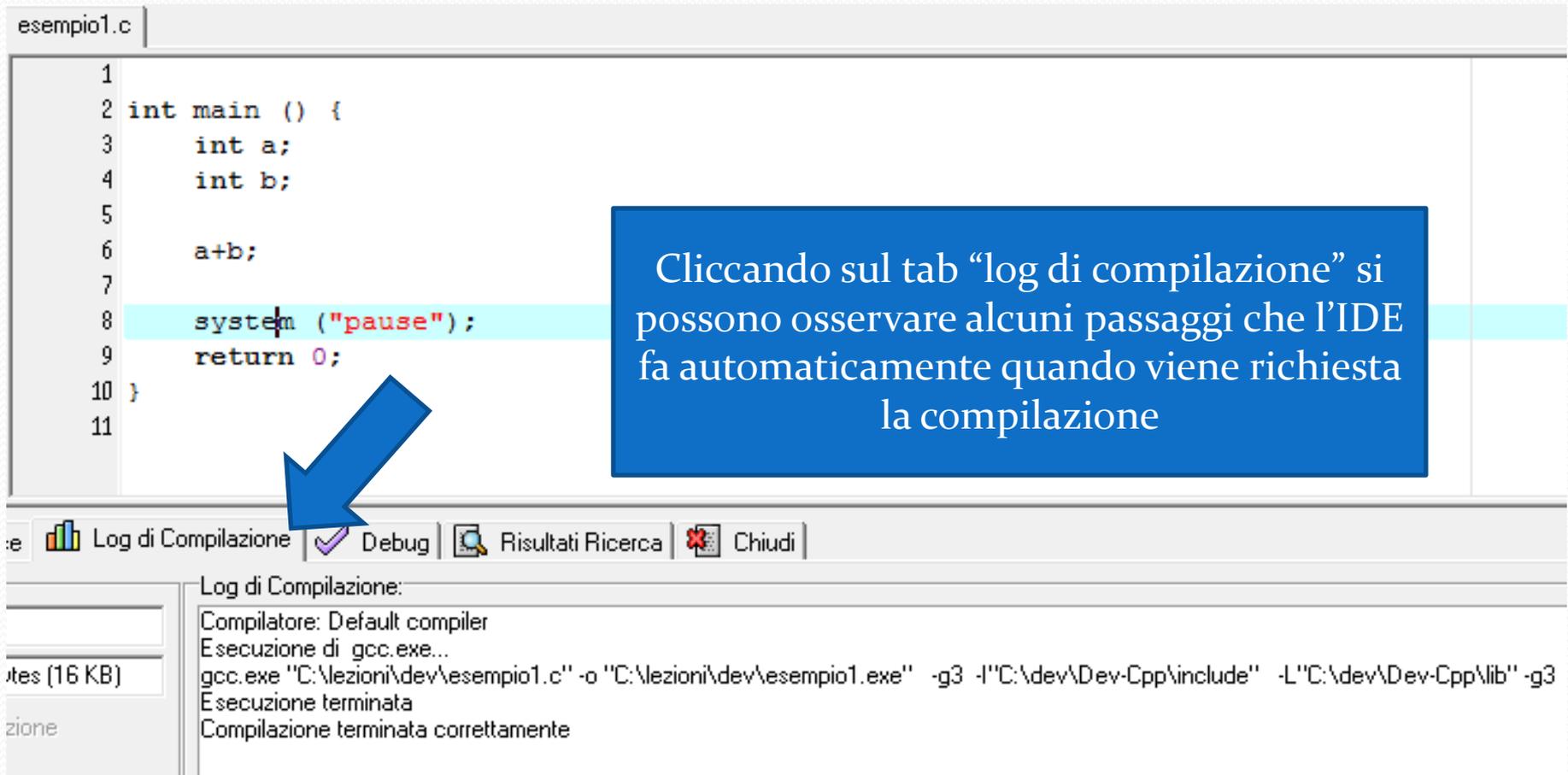
Cliccando sul menu
ESEGUI -> Compila
verrà compilata l'unità



La finestra scompare!?!

- Provando ad eseguire il codice dell'esempio 1 la finestra della console di windows si chiuderà subito dopo l'esecuzione.
- Per impedirgli di chiudere la finestra possiamo utilizzare più “trucchi”:
 - `system("PAUSE");` //fa una chiamata al sistema operativo chiedendogli di eseguire PAUSE
 - `getch();` //attende un carattere in input dall'utente

Il Log di Compilazione



```
1
2 int main () {
3     int a;
4     int b;
5
6     a+b;
7
8     system ("pause");
9     return 0;
10 }
11
```

Cliccando sul tab “log di compilazione” si possono osservare alcuni passaggi che l’IDE fa automaticamente quando viene richiesta la compilazione

Log di Compilazione:
Compilatore: Default compiler
Esecuzione di gcc.exe...
gcc.exe "C:\lezioni\dev\esempio1.c" -o "C:\lezioni\dev\esempio1.exe" -g3 -I"C:\dev\Dev-Cpp\include" -L"C:\dev\Dev-Cpp\lib" -g3
Esecuzione terminata
Compilazione terminata correttamente

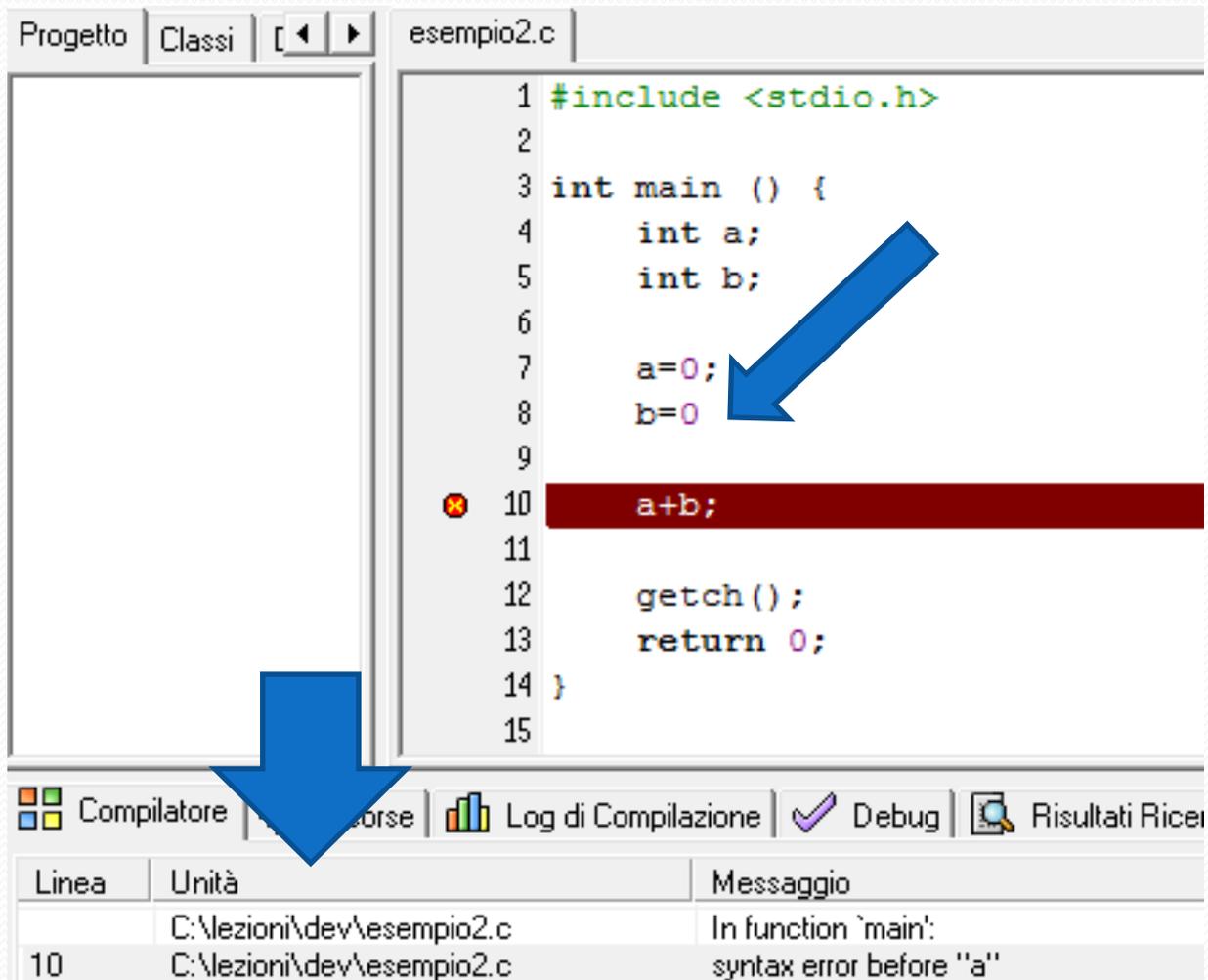
ERRORE: undeclared (first use ...)

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6 if (a==1) {
7 printf("ciao ciao \n");
8 }else {
9     if (b==0) {
10 printf("bla bla \n");
11     } else {
12 printf("boh boh \n");
13 }
```

Linea	Unità	Messaggio
6	C:\lezioni\indentazione.c	In function `main`: `a` undeclared (first use in this function) (Each undeclared identifier is reported only once for each function it appears in.)
9	C:\lezioni\indentazione.c	`b` undeclared (first use in this function)

Il compilatore segnala che la variabile non è stata dichiarata. Occorre quindi anteporre alla riga segnalata una dichiarazione della variabile. (int a;)

ERRORE: la linea "finta"



```
1 #include <stdio.h>
2
3 int main () {
4     int a;
5     int b;
6
7     a=0;
8     b=0;
9
10    a+b;
11
12    getch();
13    return 0;
14 }
15
```

Compiler | Source | Log di Compilazione | Debug | Risultati Ricerca

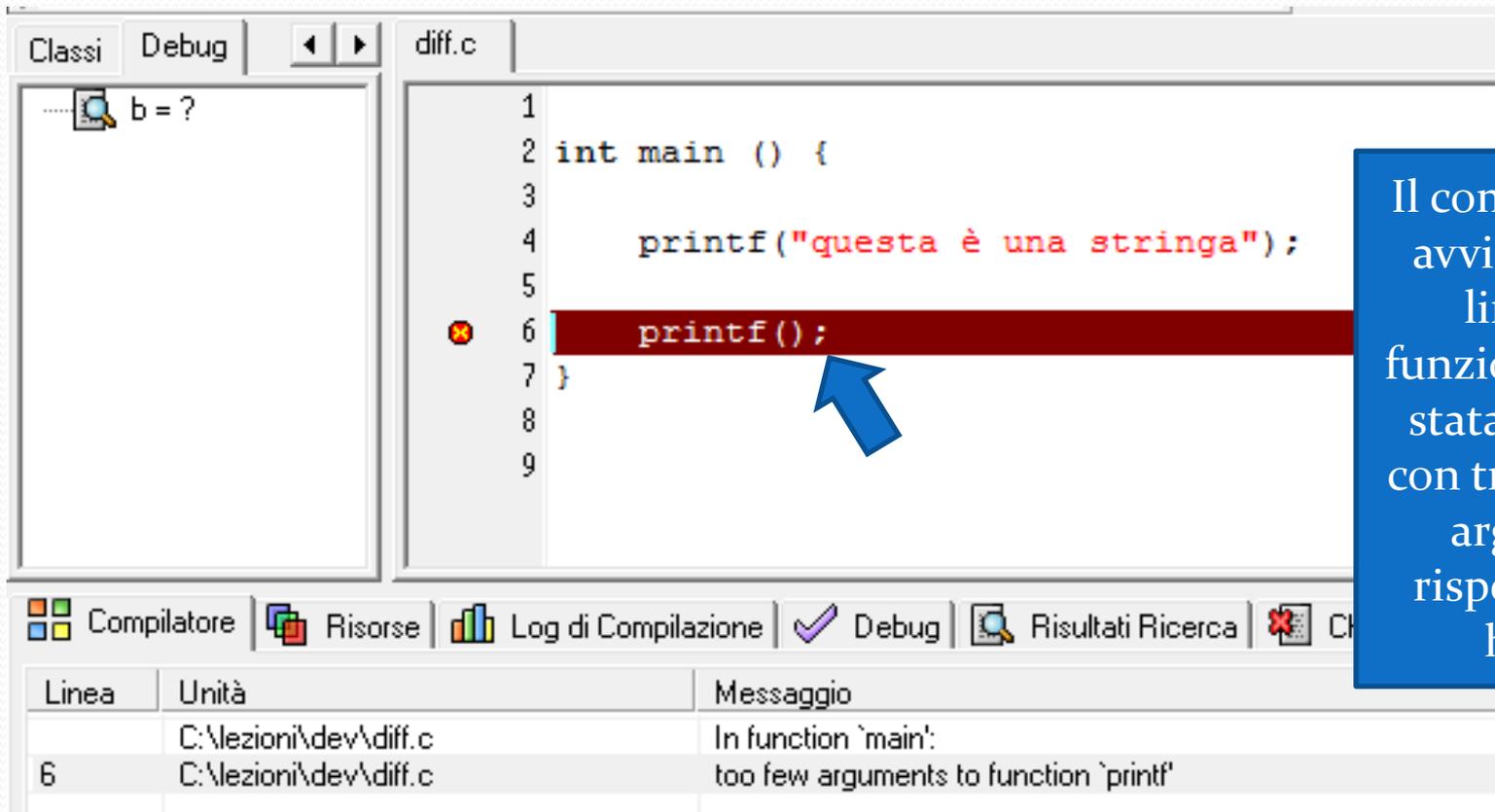
Linea	Unità	Messaggio
10	C:\lezioni\dev\esempio2.c	In function `main': syntax error before "a"

Viene segnalato un errore alla linea 10.

Osservando bene nel tab "Compilatore" leggiamo che c'è un errore di sintassi PRIMA di a

L'errore sarà, molto probabilmente, alla linea precedente.

ERRORE: too few arguments



```
1
2 int main () {
3
4     printf("questa è una stringa");
5
6     printf();
7 }
8
9
```

Il compilatore ci avvisa che alla linea 6 la funzione printf è stata chiamata con troppi pochi argomenti rispetto al suo header

Linea	Unità	Messaggio
6	C:\lezioni\dev\diff.c	In function `main': too few arguments to function `printf'

ERRORE: linker error

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello napoli\n");

    System ("PAUSE");

    return 0;
}
```

Il linker non riesce a capire la reference alla funzione System perché non la trova né nel file attuale né nelle librerie incluse

Log di Compilazione | Debug | Risultati Ricerca | Chiudi

Messaggio

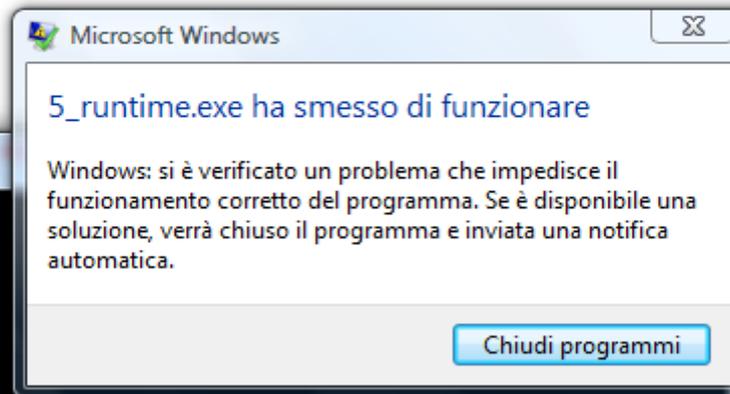
[Linker error] undefined reference to `System'
ld returned 1 exit status

ERRORI A RUNTIME

Anche se la sintassi è corretta e la compilazione va a buon fine, potrebbero verificarsi errori in fase di esecuzione del programma..

```
1 #include <stdio.h>
2
3
4 int main () {
5     int num;
6     int numero;
7     int somma;
8
9     num=2;
10
```

```
C:\lezioni\dev\5_runtime.exe
effettua la somma di due numeri
```



Esercizio del 22-03-2011

Scrivere un programma in linguaggio C che dati in ingresso i coefficienti A B C dell'equazione di secondo grado

$$ax^2+bx+c = 0$$

Calcoli le due radici x e ne stampi a video il valore.