

Fondamenti di Informatica e Laboratorio T-AB  
Ingegneria Elettronica e Telecomunicazioni

---

# Lab 01

## Introduzione a Codelite

# Costruzione di un'Applicazione

---

**Per costruire un'applicazione occorre:**

- **compilare il file (o i file se più d'uno) che contengono il testo del programma (file sorgente)**  
**Il risultato sono uno o più file oggetto.**
- **collegare i file oggetto l'uno con l'altro e con le librerie di sistema.**

# Compilazione di un'Applicazione

---

## 1) Compilare il file (o i file se più d'uno) che contengono il testo del programma

- File **sorgente**: estensione **.c**
- File **oggetto**: estensione **.o o .obj**

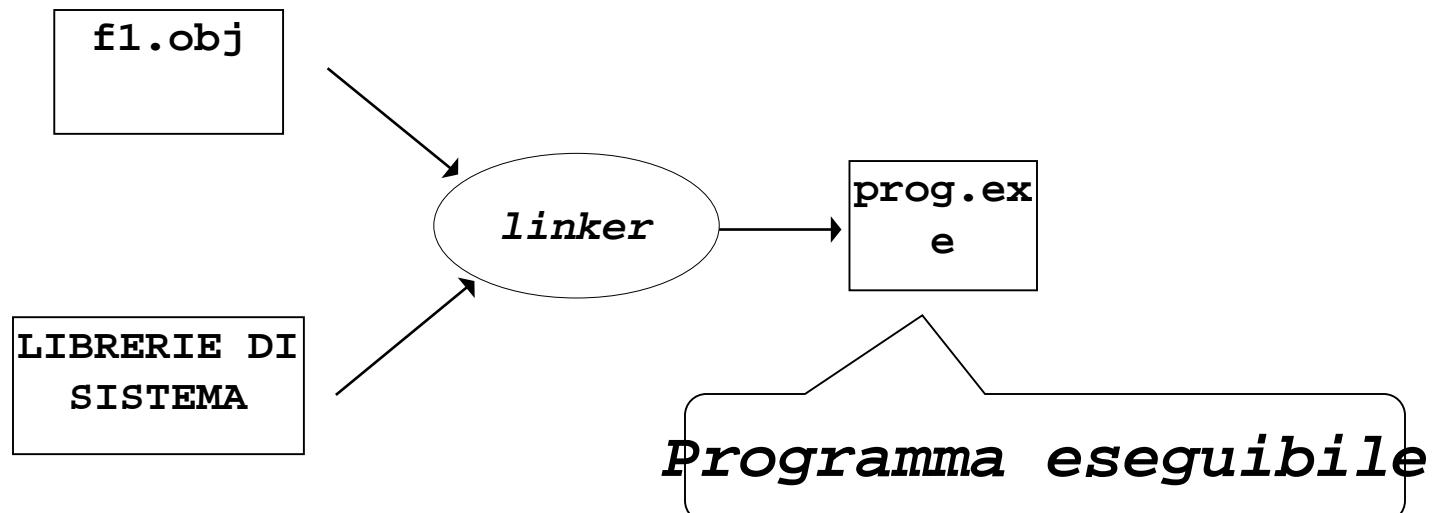


f1.obj: Una versione tradotta che però non è autonoma (e, quindi, non è direttamente eseguibile).

# Collegamento (Linking) di un'Applicazione

## 2) Collegare il file (o i file) oggetto fra loro e con le librerie di sistema

- File **oggetto**: estensione **.o** o **.obj**
- File **eseguibile**: estensione **.exe** o nessuna



# Collegamento (Linking) di un'Applicazione

---

## **LIBRERIE DI SISTEMA:**

**insieme di componenti software che consentono di interfacciarsi col sistema operativo, usare le risorse da esso gestite, e realizzare alcune "istruzioni complesse" del linguaggio**

# Ambienti Integrati

---

Oggi, gli ambienti di lavoro integrati automatizzano la procedura:

- compilano i file sorgente (*se e quando necessario*)
- invocano il linker per costruire l'eseguibile

ma per farlo devono sapere:

- *quali file sorgente costituiscono l'applicazione*
- *il nome dell'eseguibile da produrre.*

# Progetti

---

È da queste esigenze che nasce il concetto di **PROGETTO**

- un **contenitore concettuale (e fisico)**
- che **elenca i file sorgente** in cui l'applicazione è strutturata
- ed eventualmente **altre informazioni utili.**

Oggi, *tutti* gli ambienti di sviluppo integrati, *per qualunque linguaggio*, forniscono questo concetto e lo supportano con idonei strumenti.

# Installare Codelite

**CodeLite 7.0** released on Feb 7, 2015

 CodeLite 7.0 for Windows 64 bit Installer [Direct Link](#) | [SourceForge](#)



Windows

 CodeLite 7.0 for Windows 32 bit Installer [Direct Link](#) | [SourceForge](#)



 CodeLite 7.0 App Bundle for OSX 10.8 [Direct Link](#) | [SourceForge](#)



Mac OS X

 Download CodeLite 7.0 tar.gz from GitHub

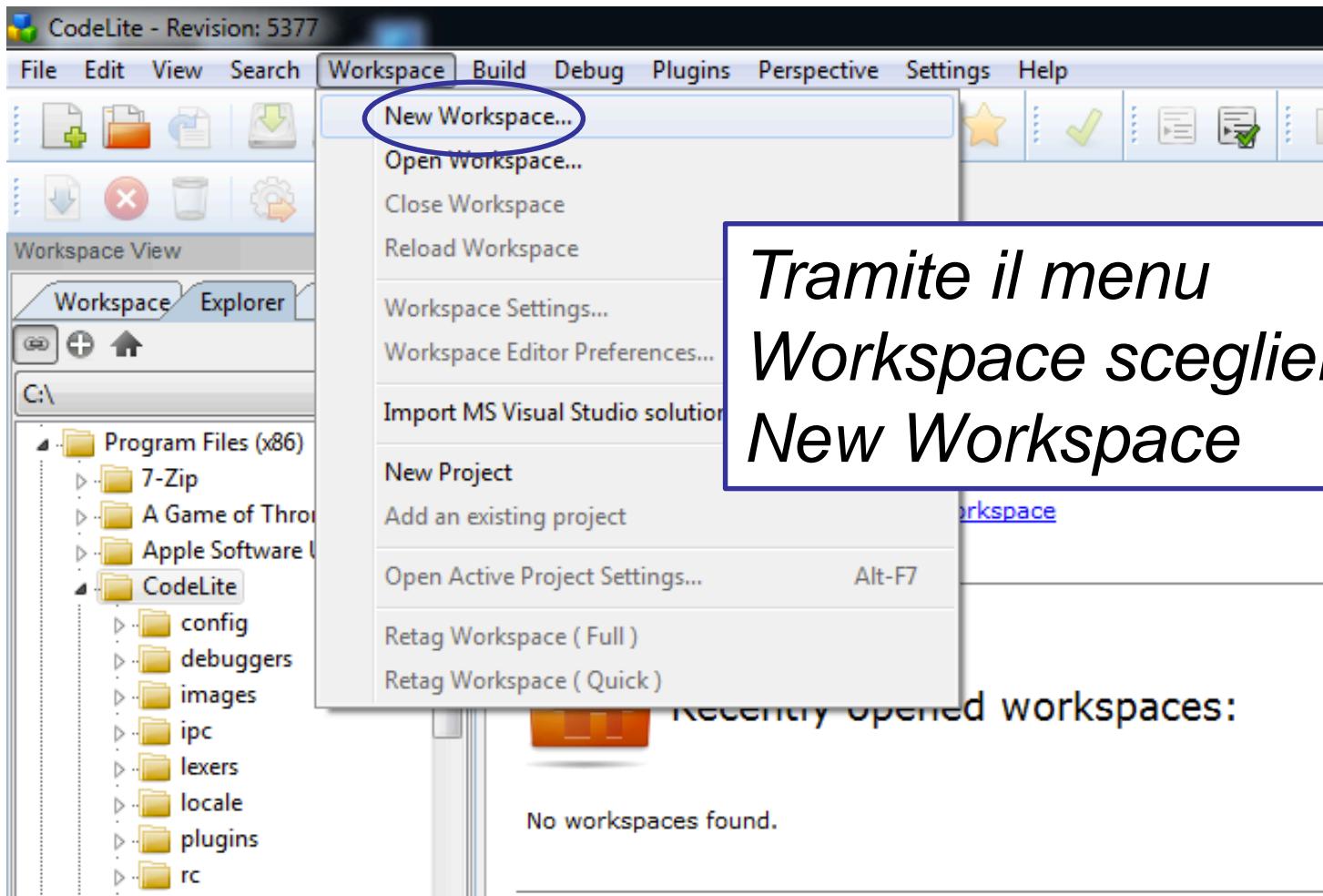
 Setup CodeLite apt repository for Ubuntu / Debian and their derivatives e.g. ....

Linux

 [CodeLite RPMs \(Fedora, openSUSE\) »](#)

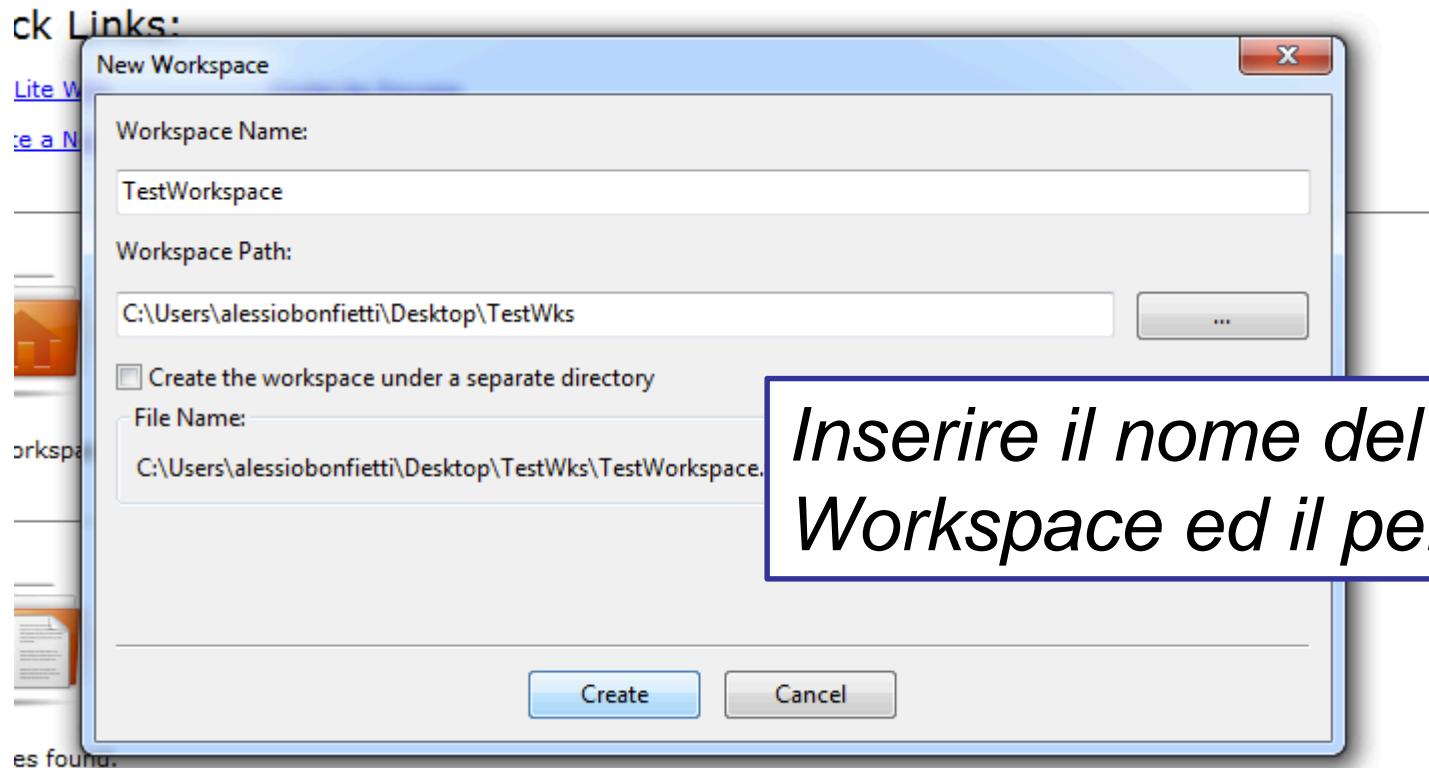
For an older Windows version  
download an installer which includes  
**codelite IDE + MinGW suite (GNU toolchain + WinAPI)**

# Progetti in Codelite



Recent workspace:

# Progetti in Codelite

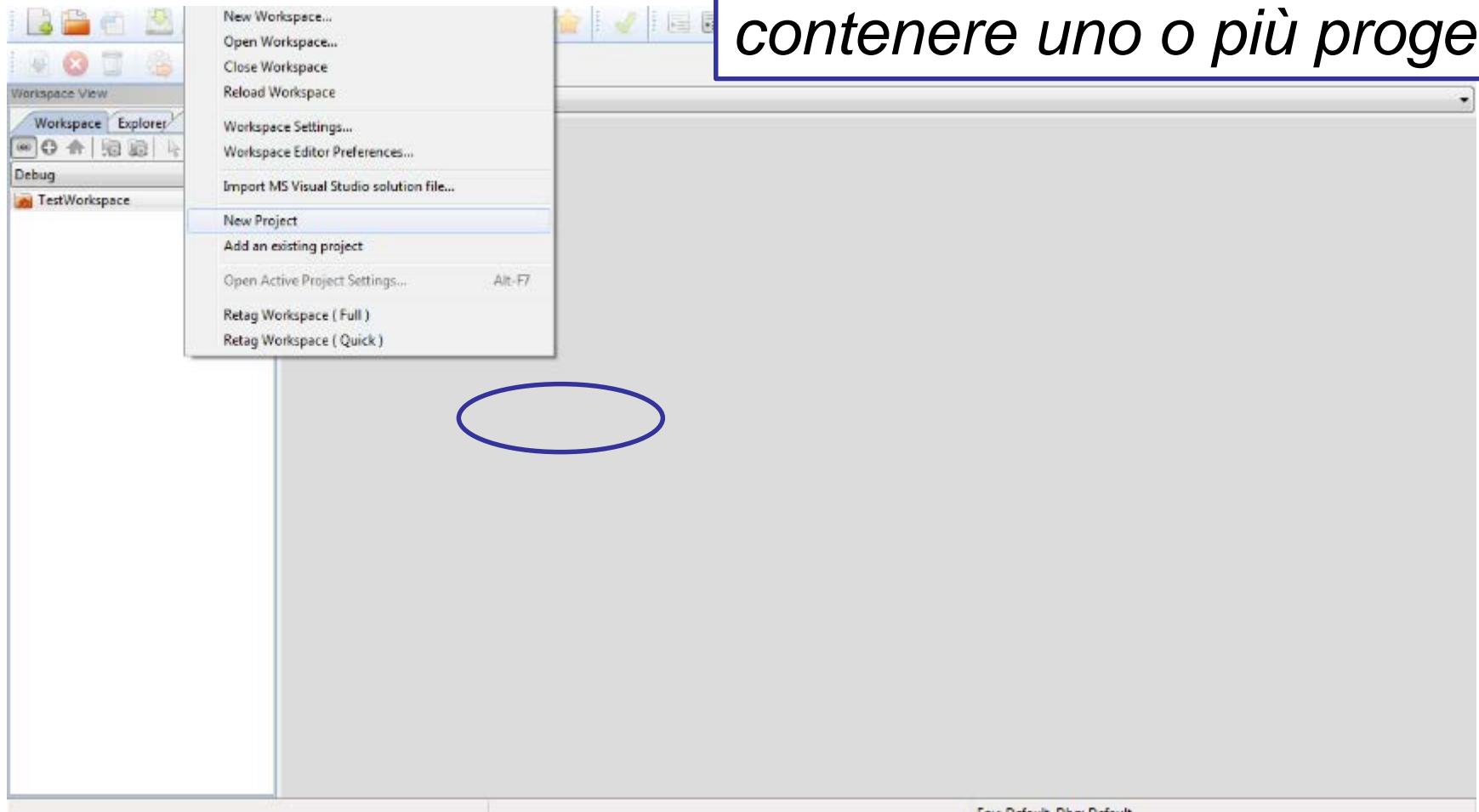


*Inserire il nome del  
Workspace ed il percorso*

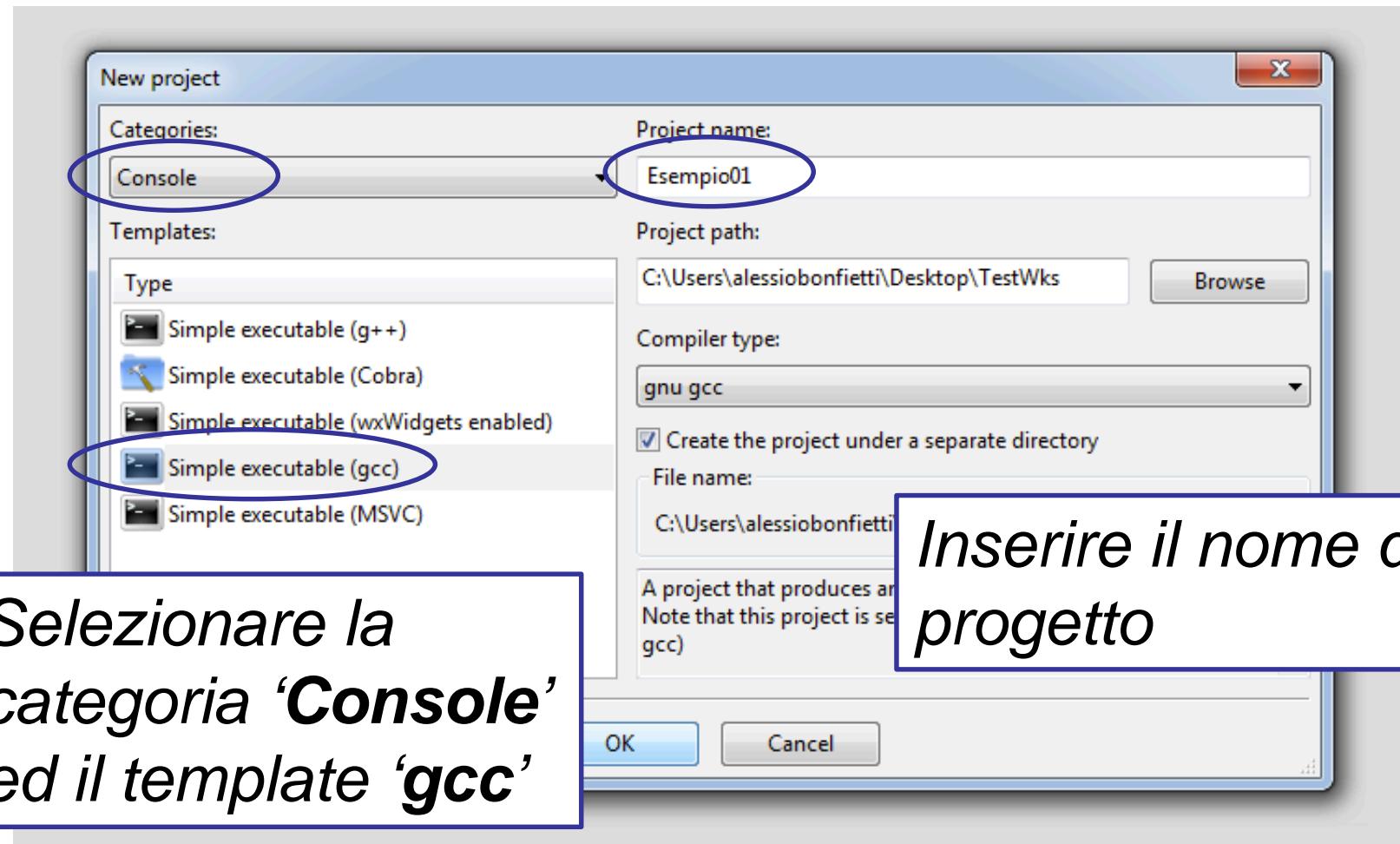
*Si consiglia di lavorare sempre in c:\temp*

# Progetti in Codelite

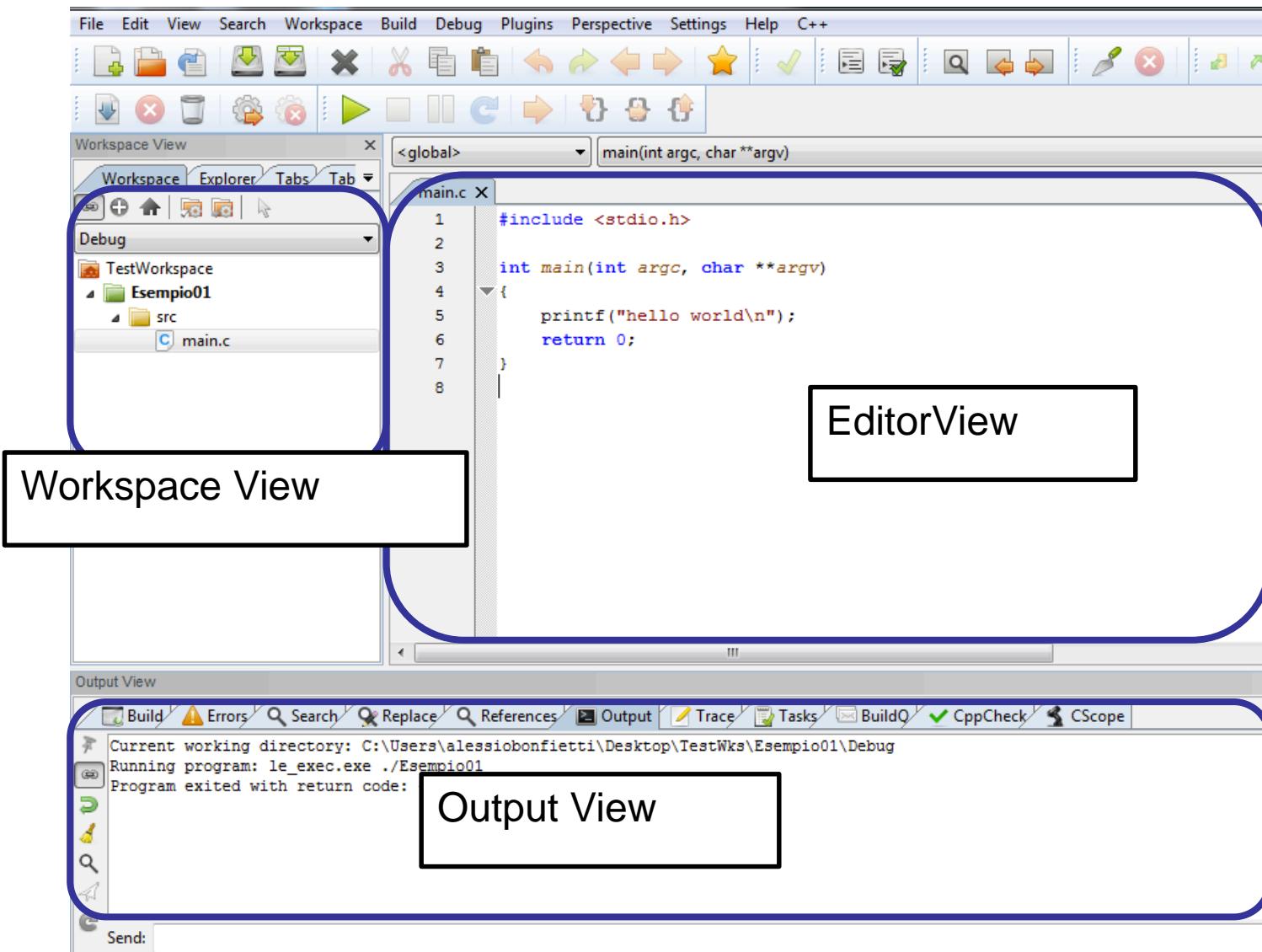
*Ogni workspace può contenere uno o più progetti*



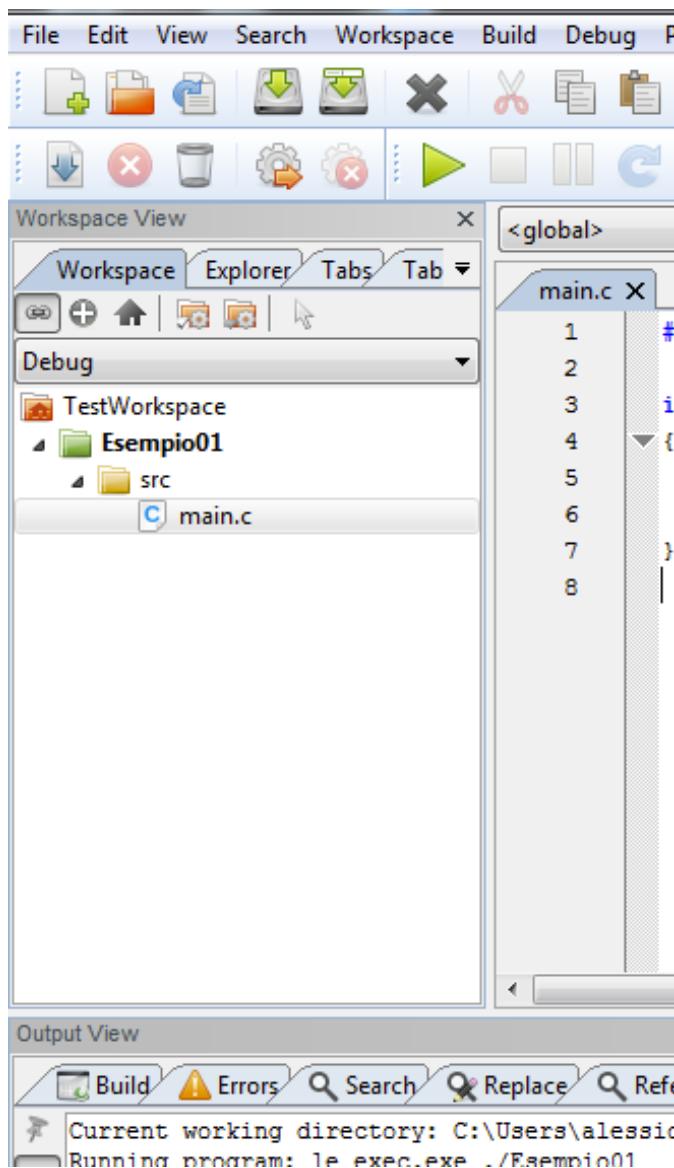
# Progetti in Codelite



# Progetti in Codelite



# Progetti in Codelite



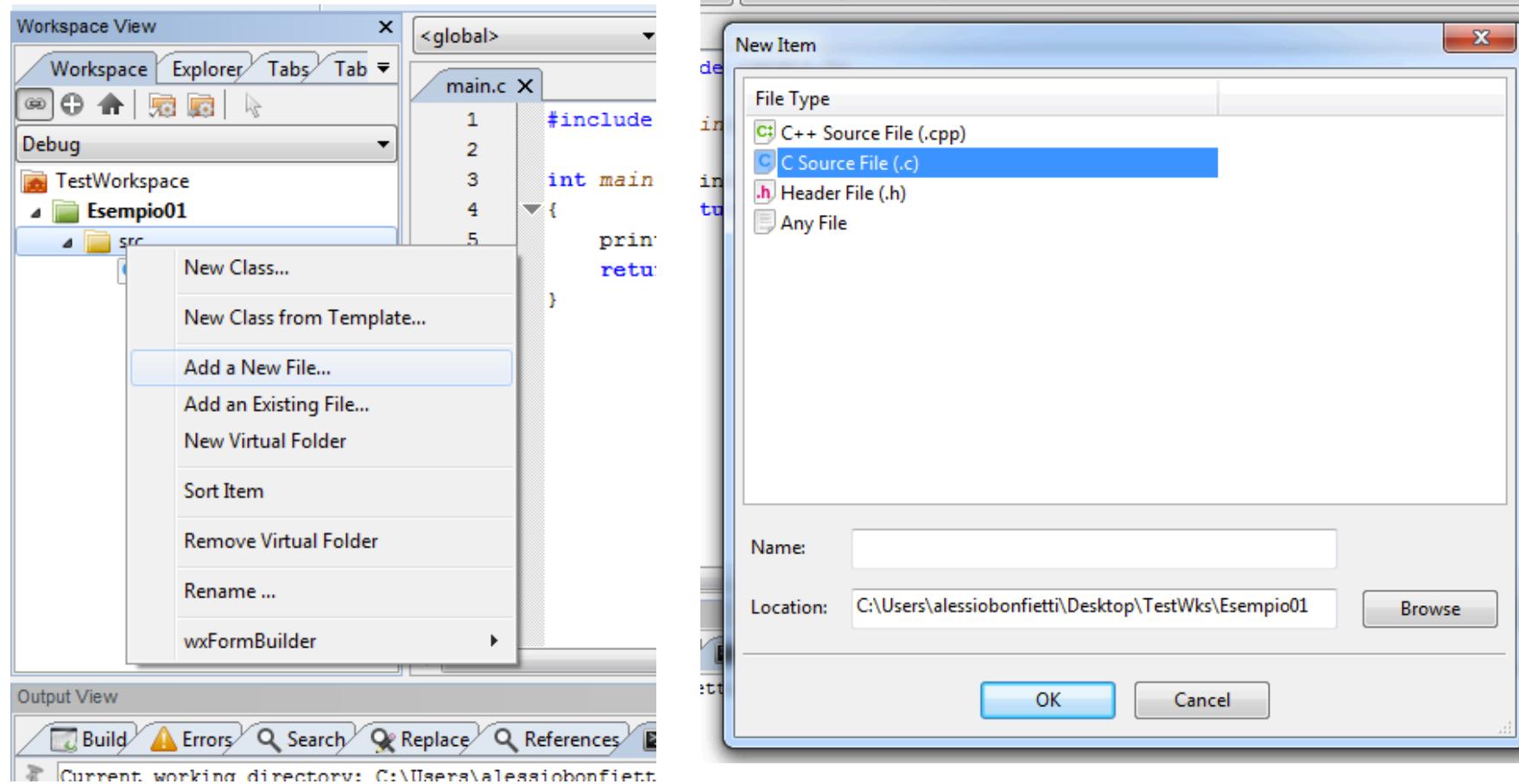
Workspace View

Alla creazione di un progetto, l'IDE **Codelite** crea automaticamente il file principale contenente la funzione main del programma.

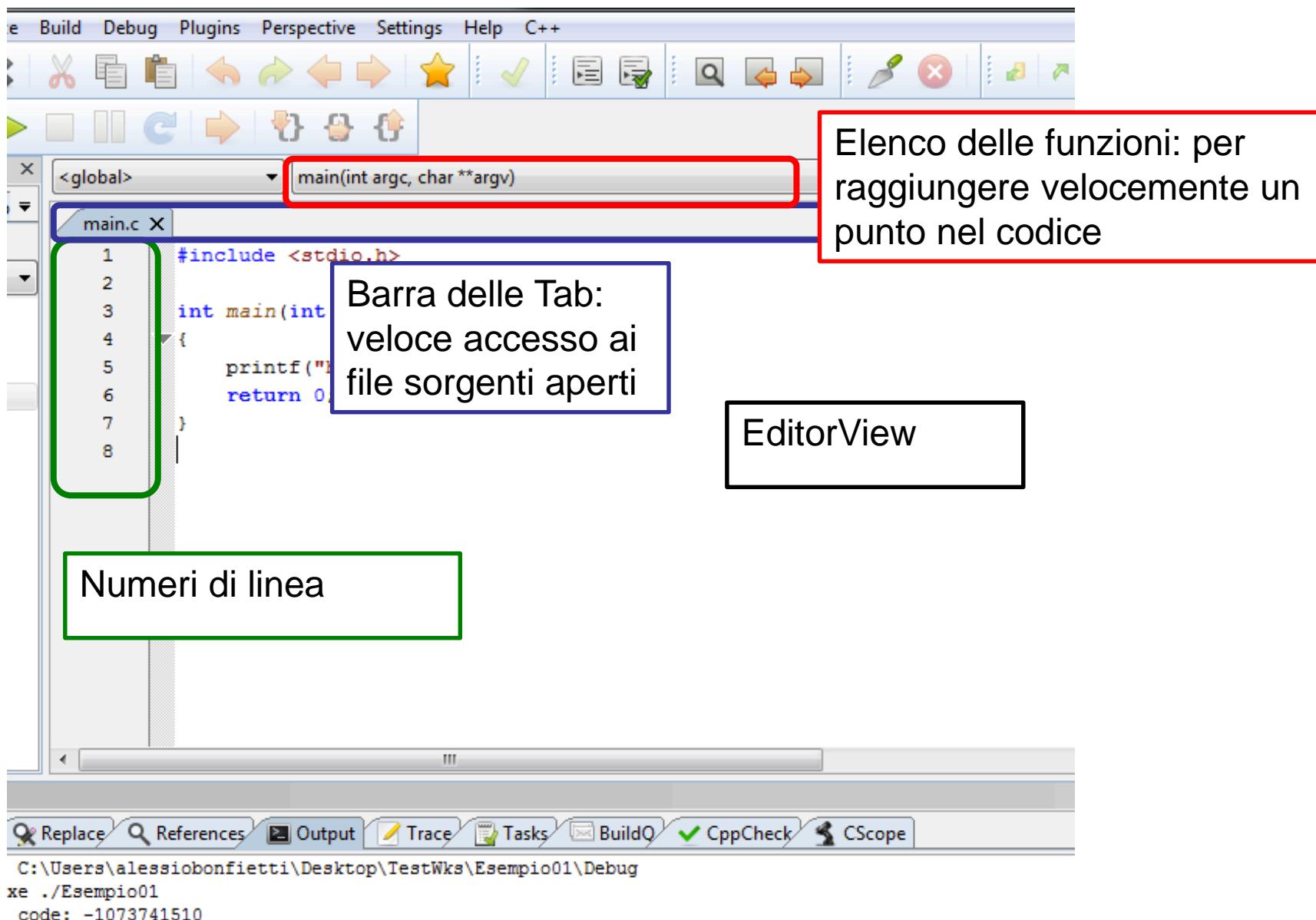
*Da questa interfaccia è dedicata alla gestione dei file sorgente*

# Progetti in Codelite

Click destro sulla directory 'src' per aggiungere un file sorgente

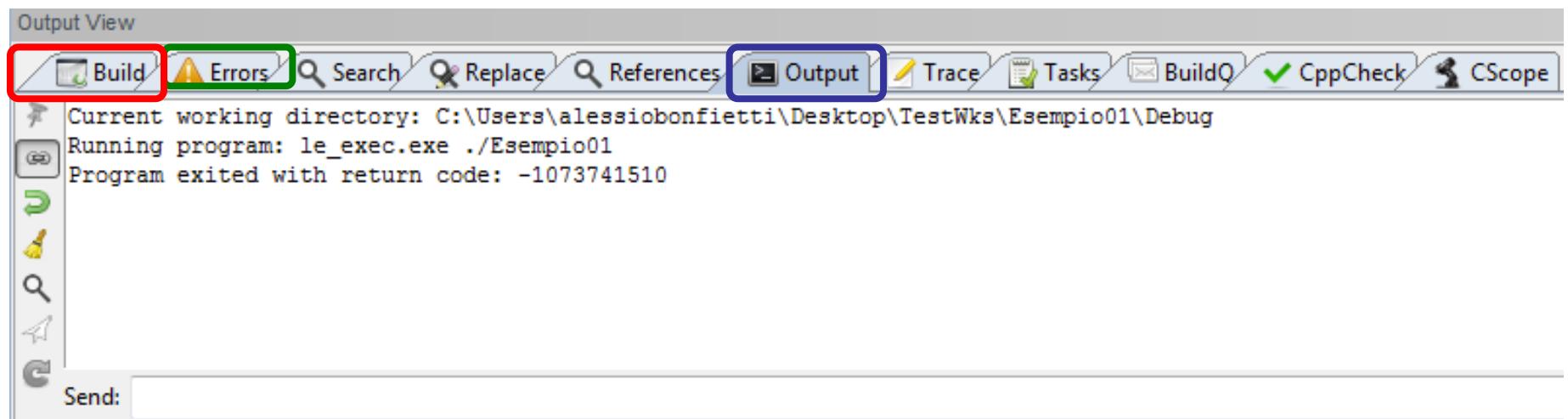


# Progetti in Codelite



# Progetti in Codelite

## Output View

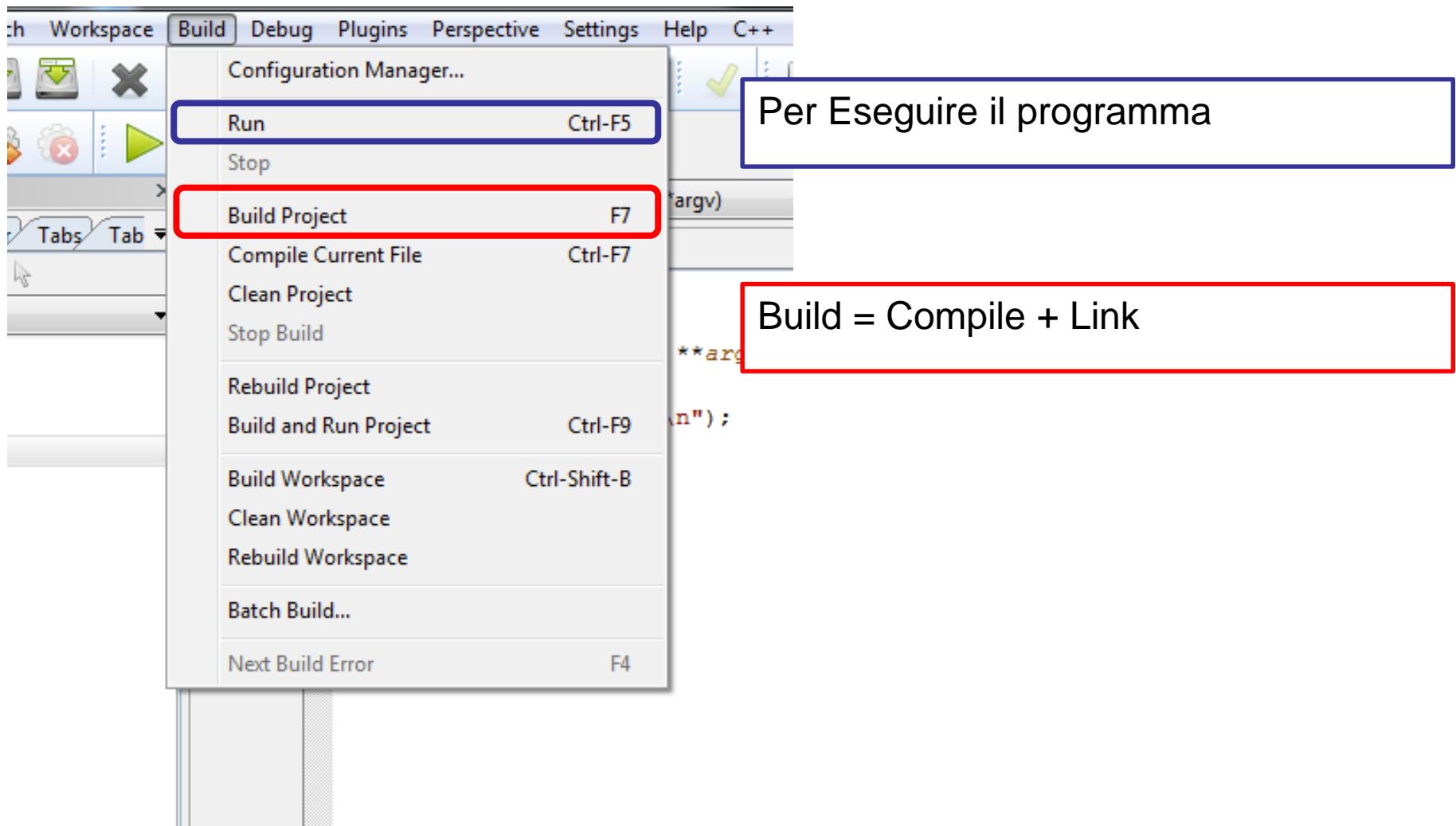


Controllo degli errori con visualizzazione dei comandi di compiling e di linking

Visualizzazione intuitiva degli errori e dei warning di compilazione

Visualizzazione dell'output di compilazione su console

# Progetti in Codelite



# ESAMIX

---

<http://esamix.labx>

# Progetti in Codelite



# Build: Warning

The screenshot shows a C IDE interface with the following details:

- Code Editor:** Displays the file `main.c` with the following content:

```
1 #include <stdio.h>
2
3 void stampa(int num);
4 int somma(int a, int b);
5
6 int main()
7 {
8     int var = 1;
9     int var2 = 5;
10    int var3 = 0;
11    char A[4] = {'a','b','c','\0'};
12
13    printf("Programma di esempio\n");
14
15    stampa(var);
16    stampa(var2);
17    var3 = somma(var, var2);
18 }
```

A red box highlights line 11: `char A[4] = {'a','b','c','\0'};`. A yellow arrow points to this line from the text "Indicazione del warning".
- Message View:** Shows the build results:
  - Yellow bar: **Build ended with 1 warning(s)**
  - List:
    - C:\Users\alessiobonfietti\Desktop\TestWks\Esempio01\main.c**
    - :11:7: warning: unused variable 'A' [-Wunused-variable]**A red box highlights the entire message view area.
- Toolbar:** Includes icons for Build, Errors, Search, Replace, References, Output, Trace, Tasks, BuildO, CppCheck, and CScope.
- Status Bar:** Shows the line number **11**.

Indicazione del  
warning

# Build: Errors

The screenshot shows a C IDE interface. On the left is a code editor with a file named "main.c". The code contains several declarations and a printf statement. A red arrow points from the printf statement to a line number 14 in the message list below. A blue box highlights the printf statement and the line number 14 in the message list. A black arrow points from the code editor to the line number 14 in the message list. On the right, a callout box with a blue border contains the text "Indicazione degli errori".

```
int main()
{
    int var = 1;
    int var2 = 5;
    int var3 = 2;
    char A[4] = {'a','b','c','\0'};
}
printf("Programma di esempio\n");
stampa(var);
stampa(var2);
var3 = somma(var,var2);
```

Indicazione degli errori

View

Build Errors Search Replace References Output Trace Tasks Build CppCheck CScope

Message

Build ended with 1 errors, 1 warnings

C:\Users\alessiobonfietti\Desktop\TestWks\Esempio01\main.c

! :14:2: error: expected ',' or ';' before 'printf' 14

! :11:7: warning: unused variable 'A' [-Wunused-variable] 11

The screenshot shows a terminal window with a blue border around the error output. It displays the command "gcc -c" followed by the path to "main.c". The output shows the errors and warnings from the compilation process.

```
gcc -c "C:/Users/alessiobonfietti/Desktop/TestWks/Esempio01/main.c" -g -O0 -Wall -o ./Debug/main.o -I. -I.
C:/Users/alessiobonfietti/Desktop/TestWks/Esempio01/main.c: In function 'main':
C:/Users/alessiobonfietti/Desktop/TestWks/Esempio01/main.c:14:2: error: expected ',' or ';' before 'printf'
C:/Users/alessiobonfietti/Desktop/TestWks/Esempio01/main.c:11:7: warning: unused variable 'A' [-Wunused-variable]
mingw32-make.exe[1]: *** [Debug/main.o] Error 1
mingw32-make.exe[1]: Leaving directory 'C:/Users/alessiobonfietti/Desktop/TestWks/Esempio01'
mingw32-make.exe: *** [All] Error 2
-----Build Ended-----
1 errors, 1 warnings
```

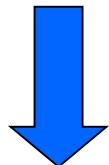
# Il Debugger

---

**Una volta scritto, compilato e collegato il programma (ossia, costruito l'eseguibile)**

**occorre uno strumento che consenta di**

- eseguire il programma passo per passo
- vedendo le variabili e la loro evoluzione
- e seguendo le funzioni via via chiamate.



**Debugger**

# Debugger

---

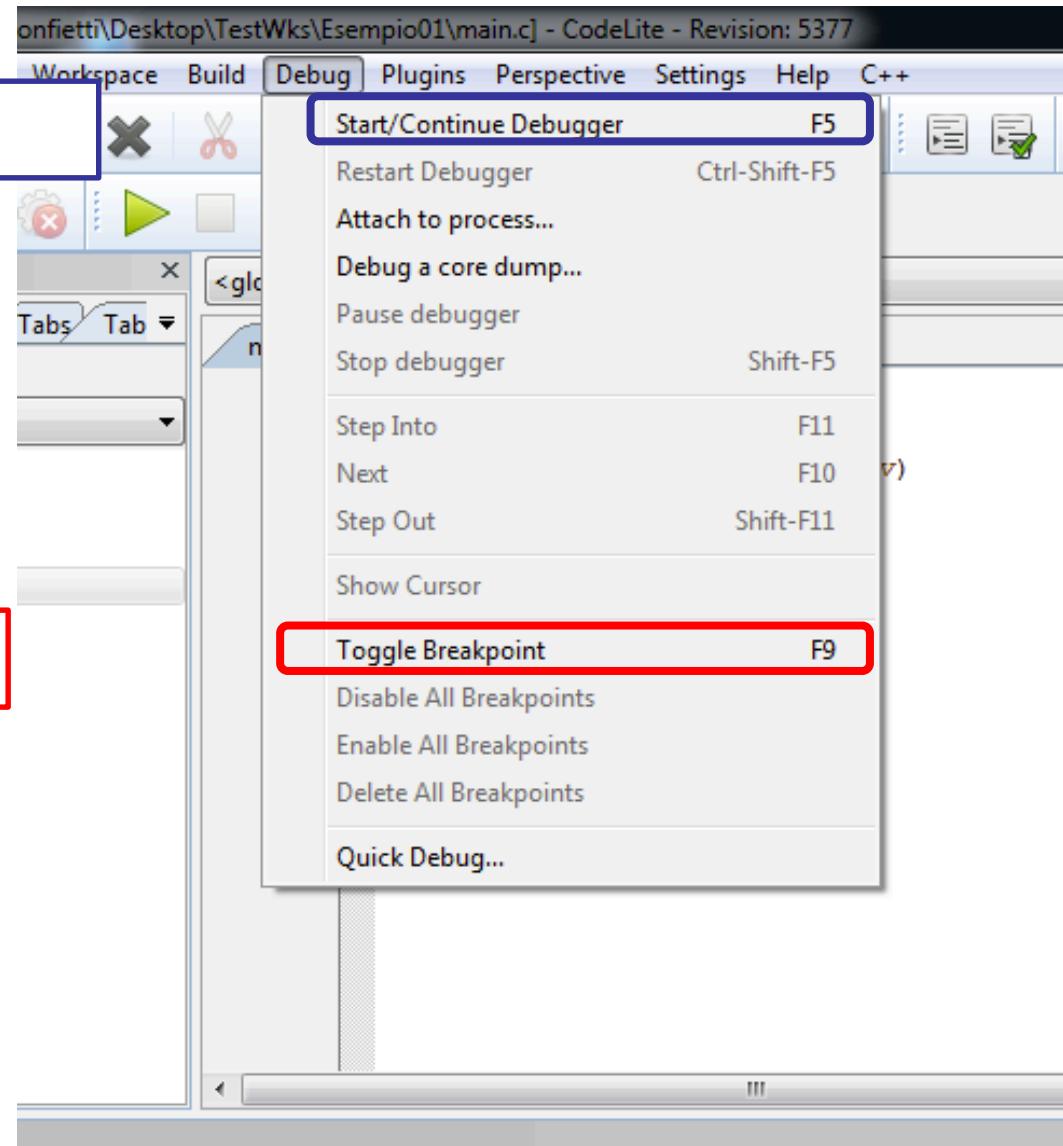
Sia Codelite sia altri ambienti di sviluppo incorporano un *debugger* con cui eseguire il programma,

- *riga per riga*
  - entrando anche dentro alle funzioni chiamate
  - oppure considerando le chiamate di funzione come una singola operazione
- oppure *inserendo breakpoints*

# Progetti in Codelite

Eseguire in modalità  
debug

Inserire un  
Breakpoint



# Fase di Debugging

---

- Prima di iniziare la sessione di debugging e' possibile inserire i cosiddetti *breakpoints*
  - *punti di interruzione nell'esecuzione del programma in cui il debugger fornisce una "fotografia" dello stato delle variabili*
- Per inserire un breakpoint posizionare il cursore nel punto in cui si vuole fermare il debug e (alternative):
  - Utilizzare il comando da Menù
  - Premere F9
  - Singolo click a fianco del numero di riga

# Debugger

The screenshot shows a debugger interface with several key components:

- Toolbar:** A red box highlights the toolbar at the top, which includes icons for file operations (Save, Delete, Open, Save As, Preferences), and various debug commands like Run, Stop, Break, Step Over, Step Into, and Step Out.
- Workspace View:** Shows the workspace with tabs for "Workspace", "Explorer", "Tabs", and "Tab". A dropdown menu shows "Debug" is selected. Below it, "TestWorkspace" is listed.
- Code Editor:** The main editor window displays the code for "main.c".

```
3 void stampa(int num);
4 int somma(int a, int b);
5
6 int main()
7 {
8     int var = 1;
9     int var2 = 5;
10    int var3 = 2;
11
12    printf("Programma di esempio\n");
13
14    stampa(var);
15    stampa(var2);
16    var3 = somma(var, var2);
17    stampa(var3);
18}
```

A green box highlights the line number 12, which contains a breakpoint icon (a red circle with a green arrow).
- Debug Mode:** A black box labeled "Debug Mode" covers the bottom-left area of the interface.
- Indicator di posizione Debug:** A blue box highlights the current execution position at line 12.
- Locals:** A green box highlights the "Locals" tab in the bottom navigation bar, which displays the current state of variables:

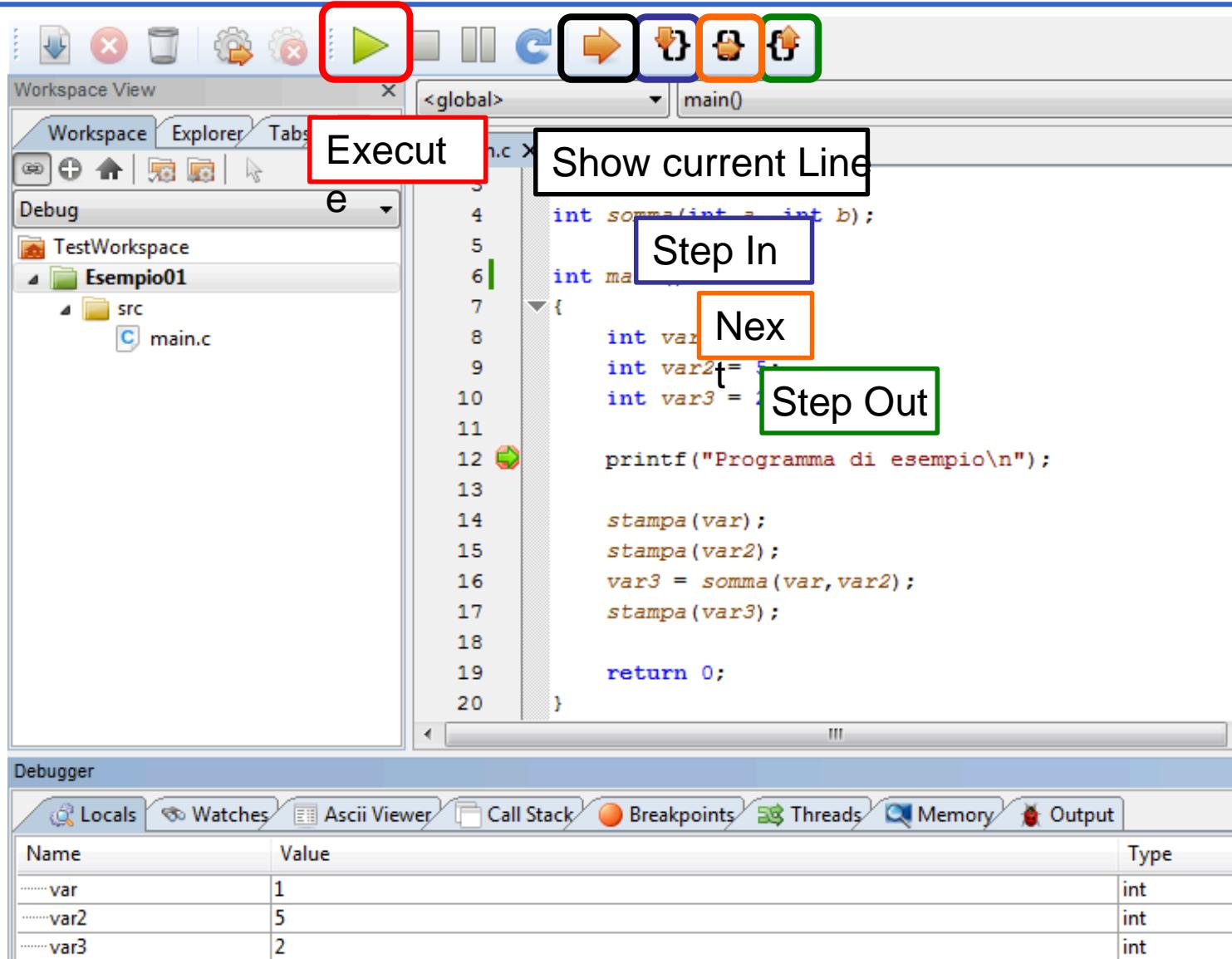
Name	Value	Type
.....var	1	int
.....var2	5	int
.....var3	2	int
- Comandi veloci Debug:** A red box highlights the "Debug" tab in the top right, which contains the text "Comandi veloci" and "Debug".
- Bottom Navigation:** The bottom navigation bar includes tabs for "Locals", "Watches", "Ascii Viewer", "Call Stack", "Breakpoints", "Threads", "Memory", and "Output".

# Debugger: Come Procedere

---

- Nel menu Debug che compare quando il Debugger e' attivo ci sono alcune voci importanti:
  - **Execute**: esegue il programma fino al prossimo Debug
  - **Step in**: esegue passo passo le istruzioni di una funzione
  - **Step Out**: esegue l'istruzione e torna alla funzione chiamante
  - **Next**: esegue l'istruzione corrente
  - **Show current line**: permette di posizionare il cursore in una determinata posizione nel sorgente e esegue tutte le istruzioni fino ad arrestarsi al cursore.

# Debugger



# Debugger

The screenshot shows a debugger interface with the following components:

- Top Bar:** Workspace, Explorer, Tabs, Tab ▾
- Left Sidebar:** Debug, TestWorkspace, Esempio01, src, main.c
- Main Editor:** main.c X

```
4 int somma(int a, int b);
5
6 int main()
7 {
8     int var = 1;
9     int var2 = 5;
10    int var3 = 2;
11    char A[4] = {'a','b','c','\0'};
12
13    printf("Programma di esempio\n");
14
15    stampa(var);
16    stampa(var2);
17    var3 = somma(var, var2);
18    stampa(var3);
19
20    return 0;
21 }
```

A red box highlights the line `char A[4] = {'a','b','c','\0'};`.

- Bottom Bar:** Debugger, Locals, Watches, Ascii Viewer, Call Stack, Breakpoints, Threads, Memory, Output
- Locals Table:** Shows variable values.

Name	Value	Type
var	1	int
var2	5	int
var3	2	int
A	[4]	char [4]
0	97 'a'	char
1	98 'b'	char
2	99 'c'	char
3	0 ''	char

A red box highlights the row for variable `A`.

- Text Box:** Rappresentazione Array statici
- Page Number:** Lab01 30

# Debugger

The screenshot shows a debugger interface with a code editor and a call stack. The code editor displays a C program with several breakpoints set at lines 11, 14, and 19. A red box highlights the assignment statement at line 14: `B = (int*)malloc(sizeof(int)*4);`. The call stack window below shows the current stack frame.

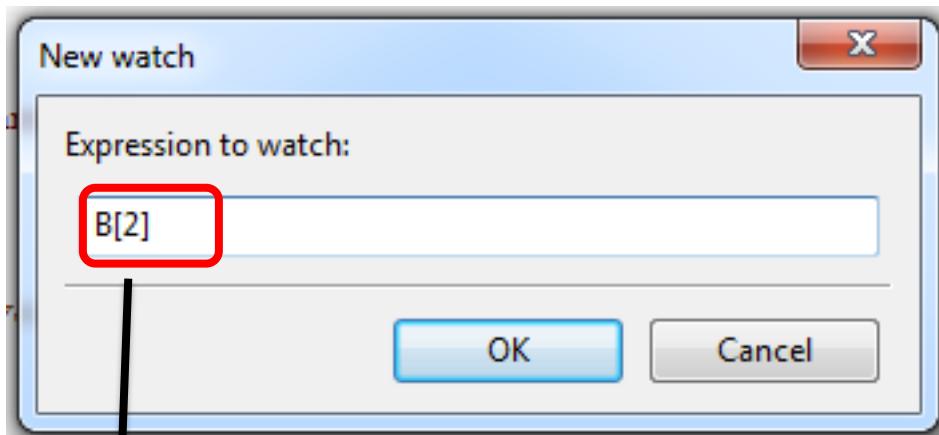
```
10     int var3 = 2;
11     char A[4] = {'a','b','c','\0'};
12     int * B;
13
14     B = (int*)malloc(sizeof(int)*4);
15
16     B[0] = 5;
17     B[2] = 12;
18
19     printf("Programma di esempio");
20
21     stampa(var);
22     stampa(var2);
23     var3 = somma(var, var2);
24     stampa(var3);
25
26
27 }
```

Rappresentazione  
parti non in  
stack: Watches

The screenshot shows the 'Watches' tab in the debugger. A red box highlights the row for variable `B`, which has a value of `0x3d1038` and a type of `int *`. Other variables listed include `*B` (value 5, type int), `var` (value 1, type int), `var2` (value 5, type int), `var3` (value 2, type int), and `A` (value {...}, type char [4]).

Name	Value	Type
B	0x3d1038	int *
*B	5	int
var	1	int
var2	5	int
var3	2	int
A	{...}	char [4]

# Debugger



Rappresentazione  
parti non in  
stack:Watches

A screenshot of the 'Debugger' window. The title bar says 'Debugger'. The tab bar includes Locals, Watches (which is selected), Ascii Viewer, Call Stack, Breakpoints, Threads, Memory, and a search icon. A vertical toolbar on the right contains 'New...', 'Delete', and 'Delete All' buttons. The main area is a table:

Expression	Value	Type
B[2]	12	int

---

# Installazione CodeLite su diversi sistemi operativi

# Installazione Mac OS X

---

Per funzionare, Codelite, necessita del compilatore.

Per verificare se il compilatore è installato, aprire il *terminale* (si trova in /Applicazioni/Utility) e digitare (senza \$):

```
$ gcc
```

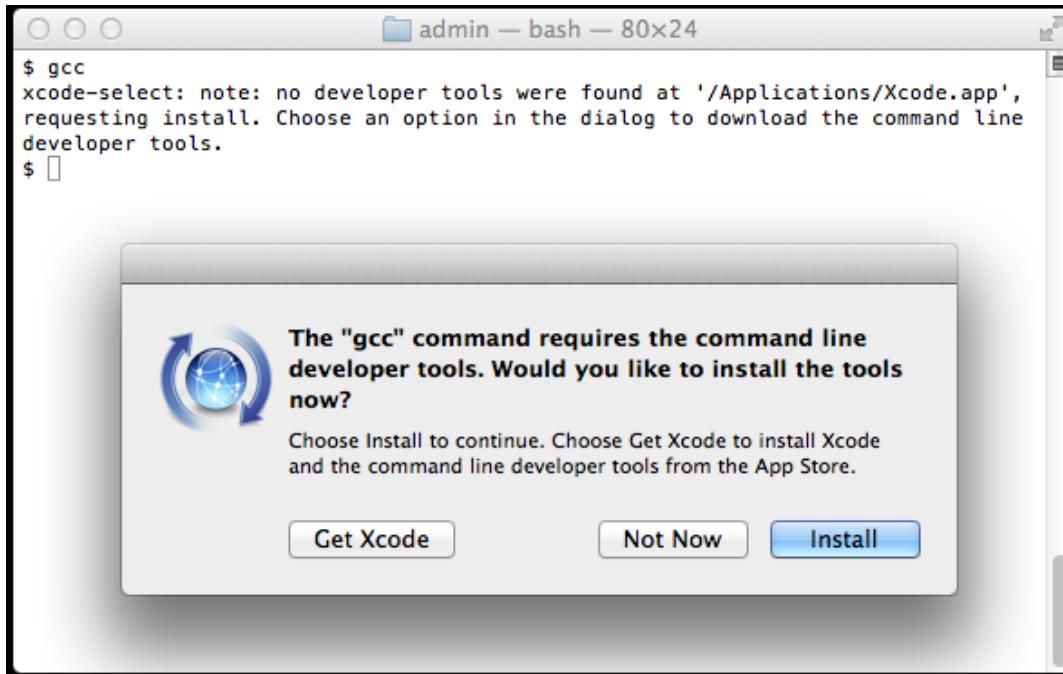
Se vi appare una scritta simile a questa va tutto bene:

```
clang: error: no input files
```

Significa che il compilatore è già installato

# Installazione Mac OS X 10.10 Yosemite

Altrimenti apparirà una finestra di installazione, tipo:

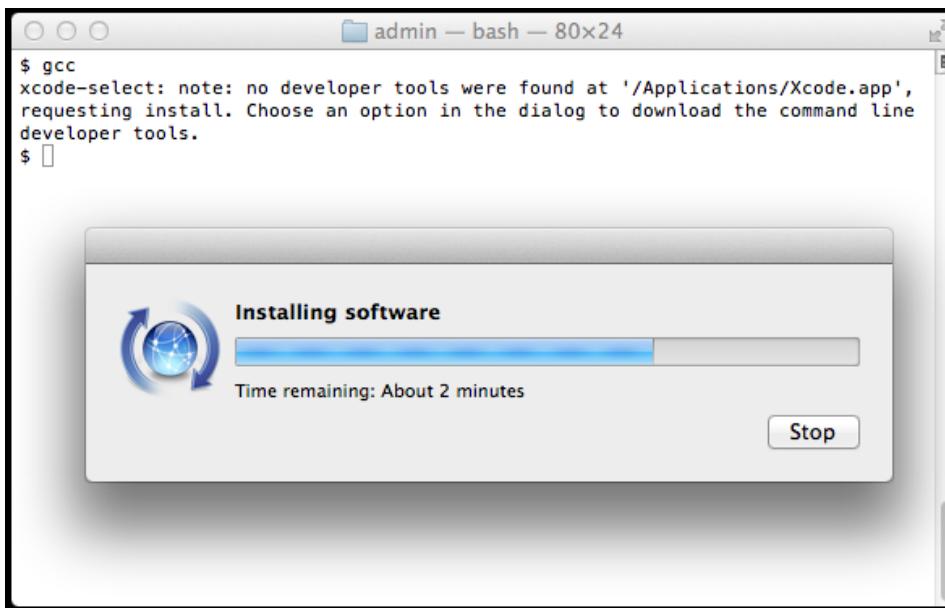


Cliccate **Install** per installare il compilatore (command line tools)

Cliccando *Get Xcode* verrà installato l'intero ambiente di sviluppo Mac Xcode

**NOTA:** Per eseguire Codelite NON è necessario Xcode ma solo il pacchetto command line tools

# Installazione Mac OS X 10.10 Yosemite



Per verificare se è installato correttamente digitare nel terminale:

```
$ xcode-select -p
```

Si dovrebbe leggere una scritta tipo:

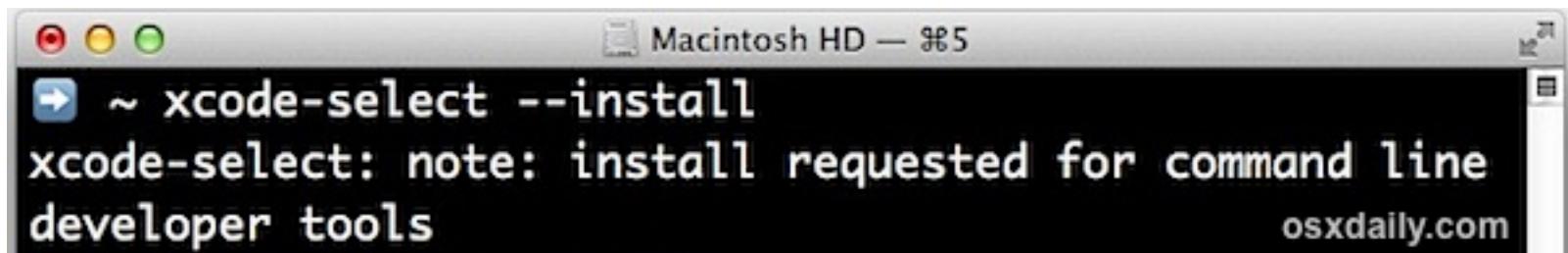
```
/Library/Developer/CommandLineTools
```

# Installazione Mac OS X

---

Se non appare la finestra di installazione provare a digirare nel terminale:

```
$ xcode-select -install
```



NOTA: se ancora non funziona, usare Google per risolvere il problema.

# Installazione Mac OS X

---

Al termine dell'installazione selezionare dal menù "Settings" -> "Build settings", nella scheda "Compiler", cliccare sul menù "Add compilers" e selezionare "Scan computer for installed compilers" -> "OK" -> "OK".

Se il problema persiste, eliminare il workspace e crearne uno nuovo

# Installazione Debian/Ubuntu

---

Potete trovare la guida all'installazione nel sito:

<http://codelite.org/LiteEditor/Repositories#toc1>

Da qualsiasi versione di Debian/Ubuntu, aprire il terminale e eseguire i seguenti comandi:

```
$ sudo apt-get purge codelite codelite-plugins  
$ sudo apt-key adv --fetch-keys
```

<http://repos.codelite.org/CodeLite.asc>

Sempre da terminale ottenere il nome della vostra distribuzione per scegliere la giusta repository:

```
$ cat /etc/*-release | grep "DISTRIB_CODENAME="
```

# Installazione Debian/Ubuntu

---

In base al risultato del comando precedente eseguire:

- **Wheezy**

```
$ sudo apt-add-repository 'deb  
http://repos.codelite.org/ubuntu/ wheezy contrib'
```

- **Jessie**

```
$ sudo apt-add-repository 'deb  
http://repos.codelite.org/ubuntu/ Jessie contrib'
```

- **Trusty**

```
$ sudo apt-add-repository 'deb  
http://repos.codelite.org/ubuntu/ trusty universe'
```

- **Utopic**

```
$ sudo apt-add-repository 'deb  
http://repos.codelite.org/ubuntu/ utopic universe'
```

**In fine eseguire sempre da terminale:**

```
$ sudo apt-get update  
$ sudo apt-get install codelite wxcrafter
```

# Installazione Windows

---

Prima di installare CodeLite è necessario installare diversi pacchetti MinGW:

- Scaricare MinGW dal sito <http://sourceforge.net/projects/mingw/>
- Installare MinGW
- All'interno di MinGW selezionare i pacchetti mingw-developer-tollkit, mingw-base, mingw-gcc-g++, mingw-make (tutti i pacchetti mingw-make)

Quindi inserire nella variabile d'ambiente PATH i percorsi ai binari MinGW e mingw-make (guida nel sito <http://codelite.org/LiteEditor/InstallingMinGW> ):

Pannello di controllo -> Sistema -> Impostazioni avanzate di Sistema -> Variabili d'ambiente, nel primo riquadro (variabili dell'utente per <nome utente>) cercare la variabile PATH e aggiungere i path ai binary MinGW e mingw-make separati da ";" (di default C:\MinGW\bin;C:\MinGW\msys\1.0\bin ), ATTENZIONE: non cancellare il contenuto della variabile PATH ma aggiungere solo i due nuovi percorsi. ATTENZIONE: non modificare la variabile PATH della tabella "variabili di Sistema", potrebbe causare problemi di instabilità del S.O.

Se la variabile PATH non è presente aggiungerla.

Una volta installato MinGW è possibile procedere con l'installazione di codelite

# Installazione Windows

---

Per verificare quale versione installare (32 o 64 bit), da “Pannello di controllo”, selezionare “Sistema” quindi leggere la versione del Sistema operativo:

[Visualizza informazioni di base relative al computer](#)

Edizione Windows

Windows 8.1 Pro

© 2013 Microsoft Corporation. Tutti i diritti riservati.



Windows® 8

[Ancora più funzionalità con una nuova edizione di Windows](#)

Sistema

Processore: Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz 2.50 GHz

Memoria installata (RAM): 8,00 GB (7,60 GB utilizzabile)

Tipo sistema: **Sistema operativo a 64 bit, processore basato su x64**

Penna e tocco: Nessun input penna o tocco disponibile per questo schermo

# Installazione Windows

---

Al termine dell'installazione selezionare dal menù "Settings" -> "Build settings", nella scheda "Compiler", cliccare sul menù "Add compilers" e selezionare "Scan computer for installed compilers" -> "OK" -> "OK".

Se il problema persiste, eliminare il workspace e crearne uno nuovo

# Installazione Windows

---

ATTENZIONE: una volta creato il progetto dovrete inserire nelle opzioni del linker -static-libgcc -static-libstdc++

Tasto destro del mouse sul progetto, “Settings”, “Common settings”, “Linker”, alla voce “Linker Options” inserire -static-libgcc -static-libstdc++

# Windows & Linux Notes

---

**ATTENZIONE:** Su Windows e Linux, CodeLite non controlla la presenza dei diritti di scrittura sulla cartella di salvataggio, controllare preventivamente la presenza dei diritti.

N.B.: In laboratorio le cartelle di Windows in cui CodeLite può salvare sono C:\Temp e Desktop

# Esercizio

Copiare e provare il seguente programma

```
#include <stdio.h>
int main(int argc, char **argv)
{
    printf("Calcolo area rettangolo\n");
    int base,altezza, area;
    printf("Inserire la larghezza del rettangolo:");
    scanf("%d",&base);
    printf("Inserire l'altezza del rettangolo:");
    scanf("%d",&altezza);
    area = base * altezza;
    printf("Il rettangolo ha area uguale a %d\n", area);
    getchar();
    return 0;
}
```