

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 / 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**



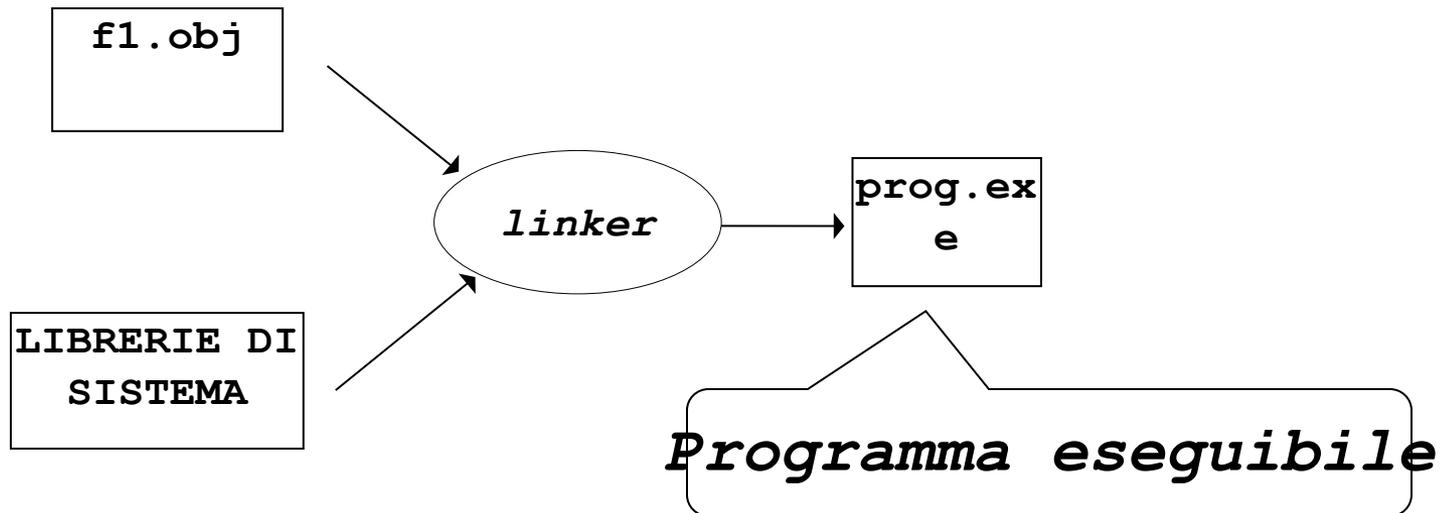
`f1.obj`: Una versione tradotta che però non è autonoma (e, quindi, non è direttamente eseguibile).

# 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

Download dal sito web: <http://downloads.codelite.org/>

## CodeLite 9.1.0 - Stable Release released on Feb 1, 2016

 CodeLite 9.1.0 for Windows **64** bit Installer [Direct Link](#) | [GitHub](#)



Windows

 CodeLite 9.1.0 for Windows **32** bit Installer [Direct Link](#) | [GitHub](#)



 CodeLite 9.1.0 App Bundle for OSX 10.10 [Direct Link](#) | [GitHub](#)



Mac OS X

 [Download CodeLite 9.1.0 tar.gz from GitHub](#)

 [Setup CodeLite apt repository for Ubuntu / Debian and their derivatives e.g. Mint »](#)



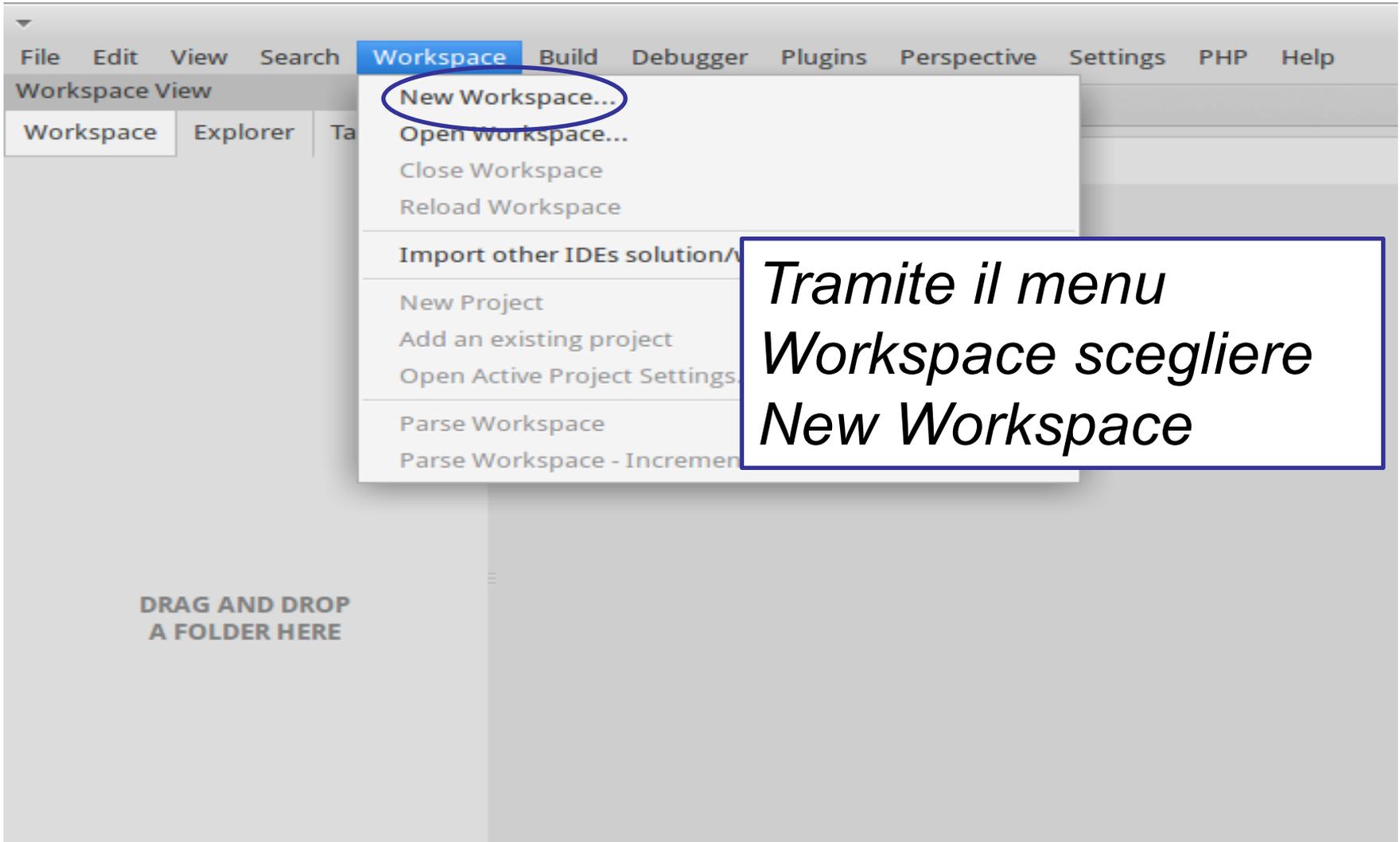
Linux

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

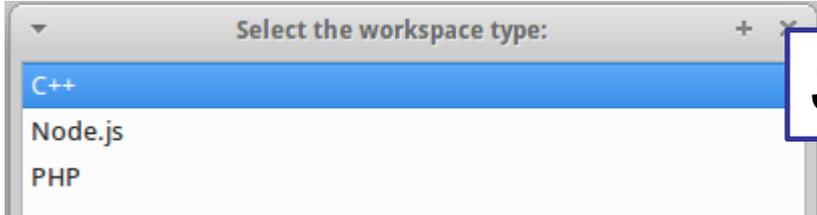
Oppure installare la macchina virtuale fornita:

[Instruzioni per l'installazione](#)

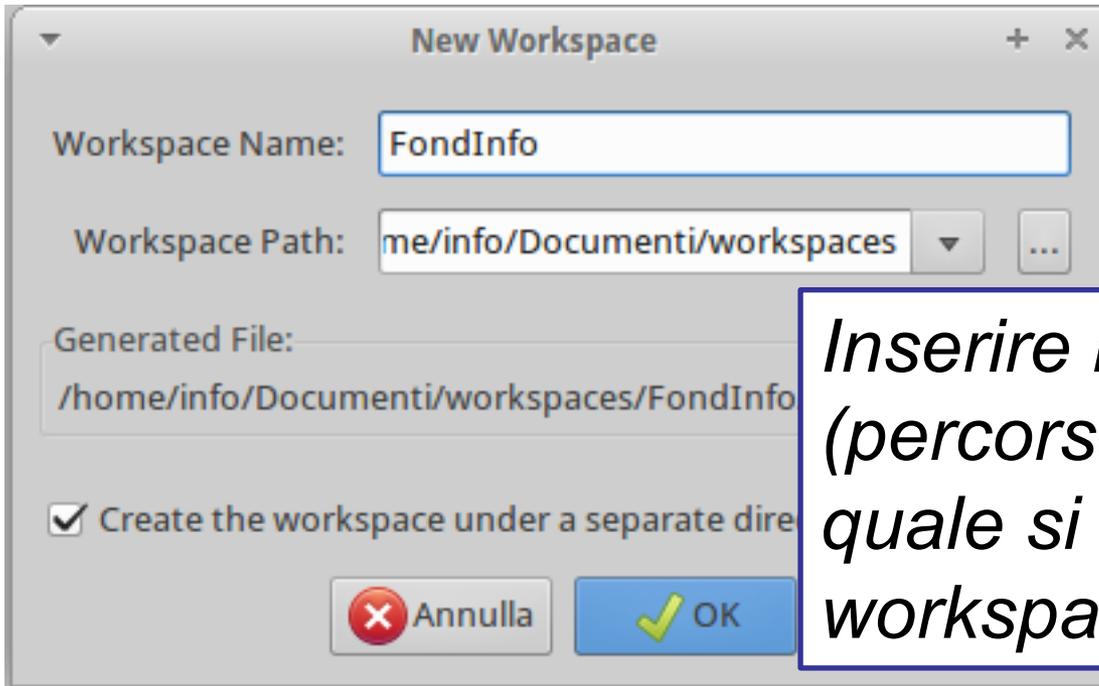
# Progetti in Codelite



# Progetti in Codelite

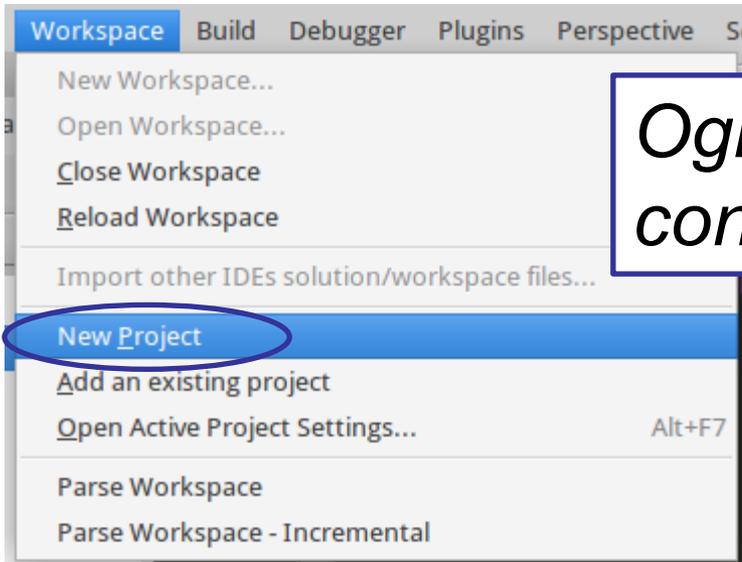


*Selezionare C++*



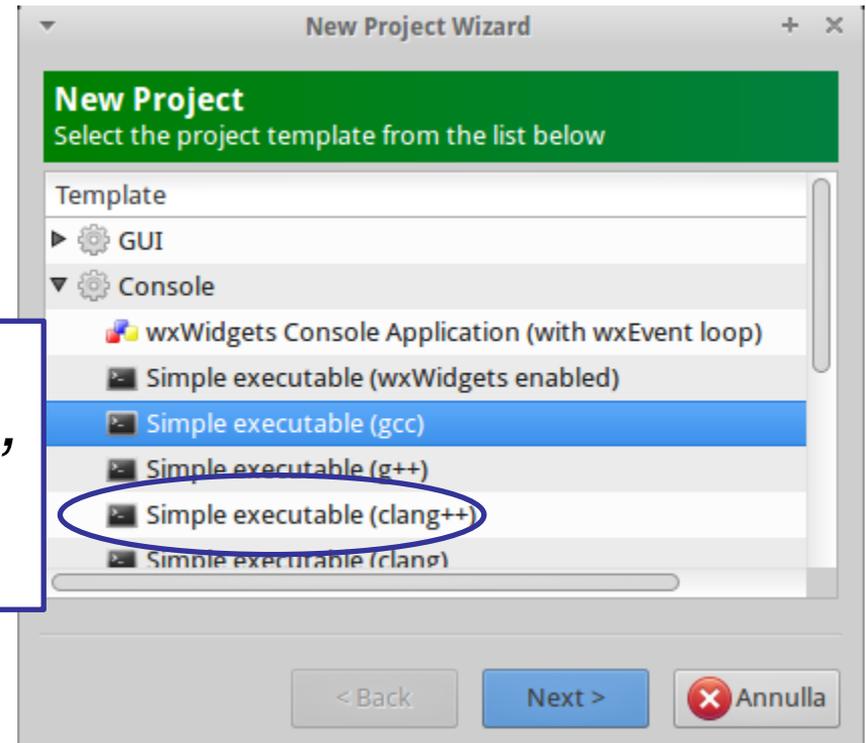
*Inserire il nome del direttorio (percorso) all'interno del quale si vuole creare il workspace*

# Progetti in Codelite

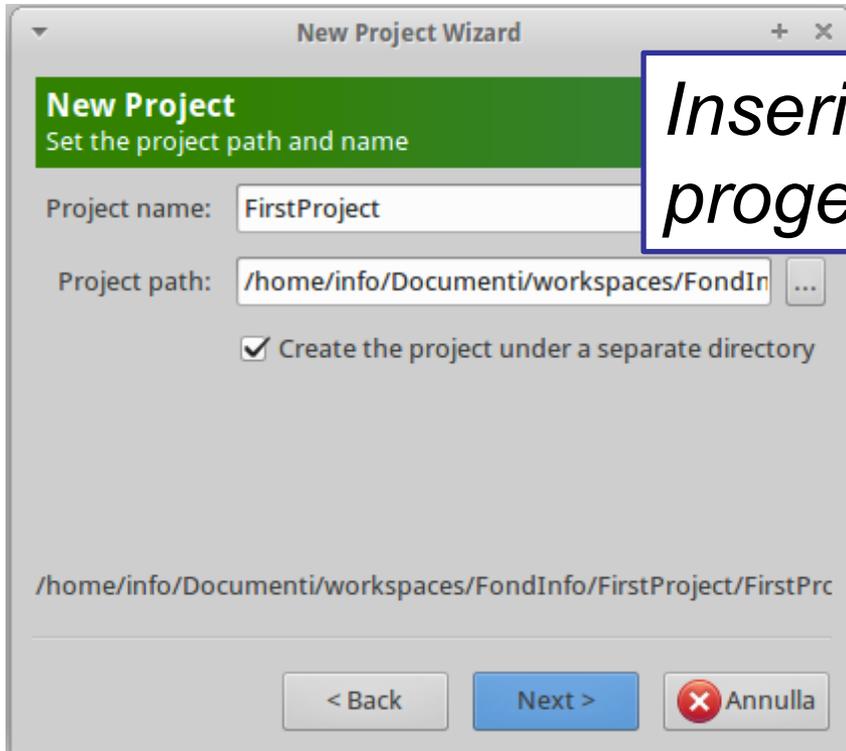


*Ogni workspace può contenere uno o più progetti*

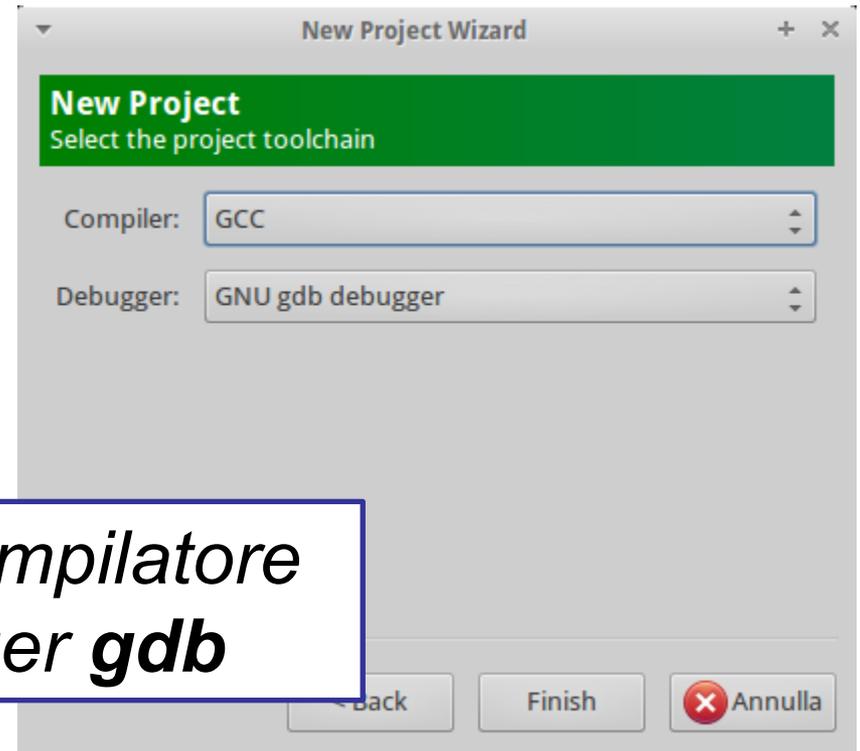
*Selezionare la categoria 'Console' ed il template 'gcc'*



# Progetti in Codelite



*Inserire il nome del progetto*



*Selezionare il compilatore gcc ed il debugger gdb*

# Progetti in Codelite

The screenshot displays the Codelite IDE interface. The top menu bar includes File, Edit, View, Search, Workspace, Build, Debugger, Plugins, Perspective, Settings, PHP, and Help. The Workspace View on the left shows a tree structure with 'FondInfo' as the root, containing 'FirstProject' and 'src' folders, with 'main.c' selected. The EditorView in the center shows the code for 'main.c':

```
1 #include <stdio.h>
2
3 int main(int argc, char **argv)
4 {
5     printf("hello world\n");
6     return 0;
7 }
8
```

The Output View at the bottom shows the following text:

```
Current working directory: /home/info/Documenti/workspaces/FondInfo/FirstProject/Debug
Running program: /usr/bin/codelite_xterm './FirstProject' '/bin/sh -f /usr/bin/codelite_exec ./F
Program exited with return code: 0
```

Blue lines connect the labels to their respective views in the IDE.

Workspace View

EditorView

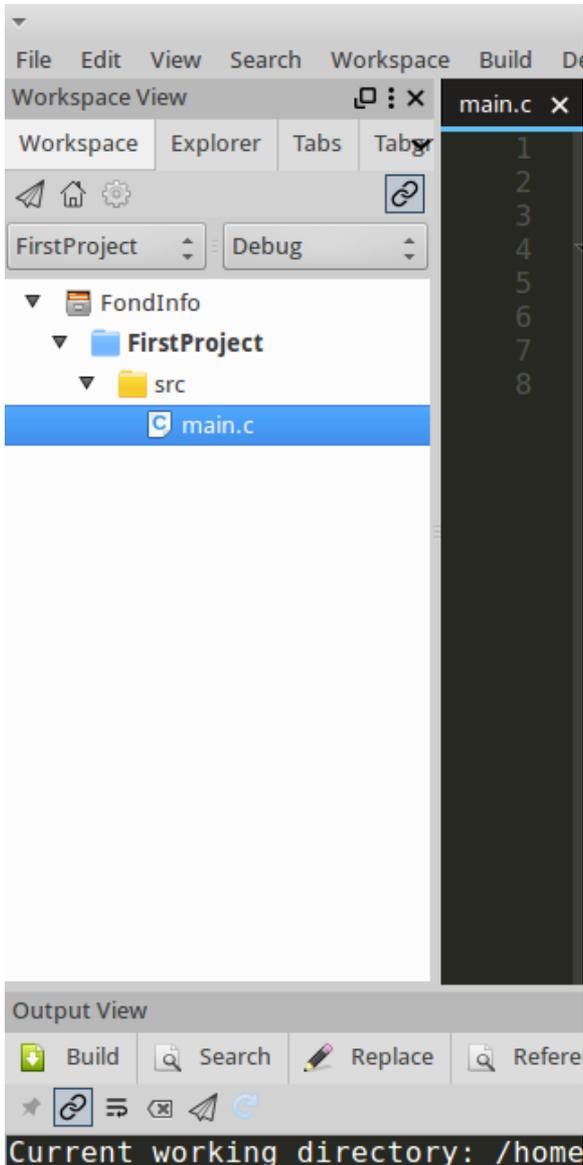
Output View

# Progetti in Codelite

Workspace View

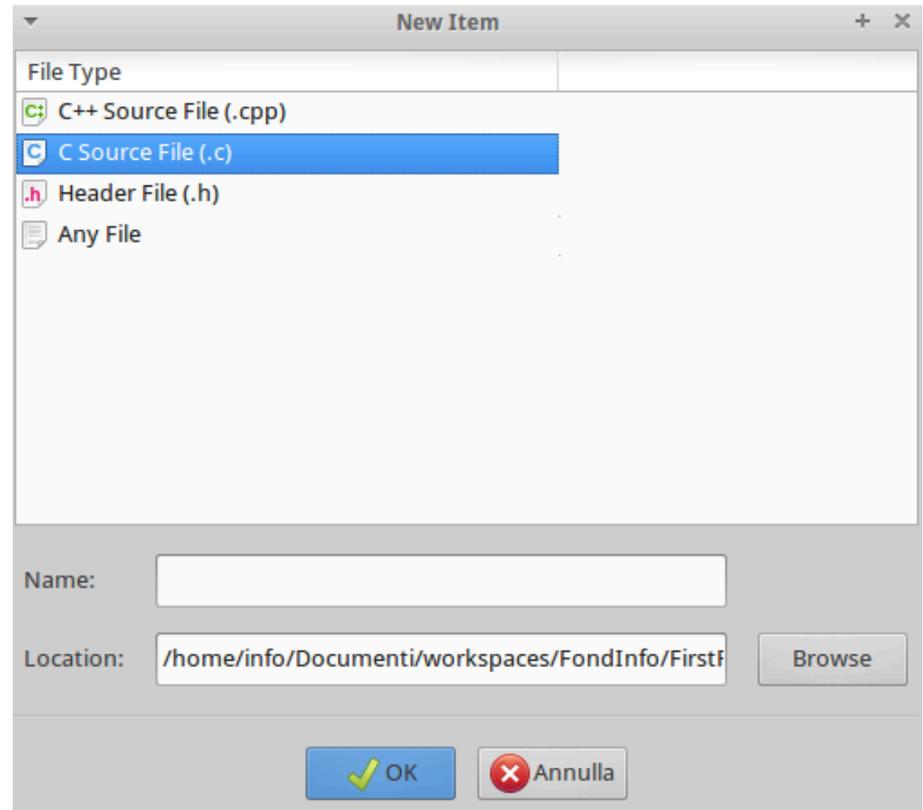
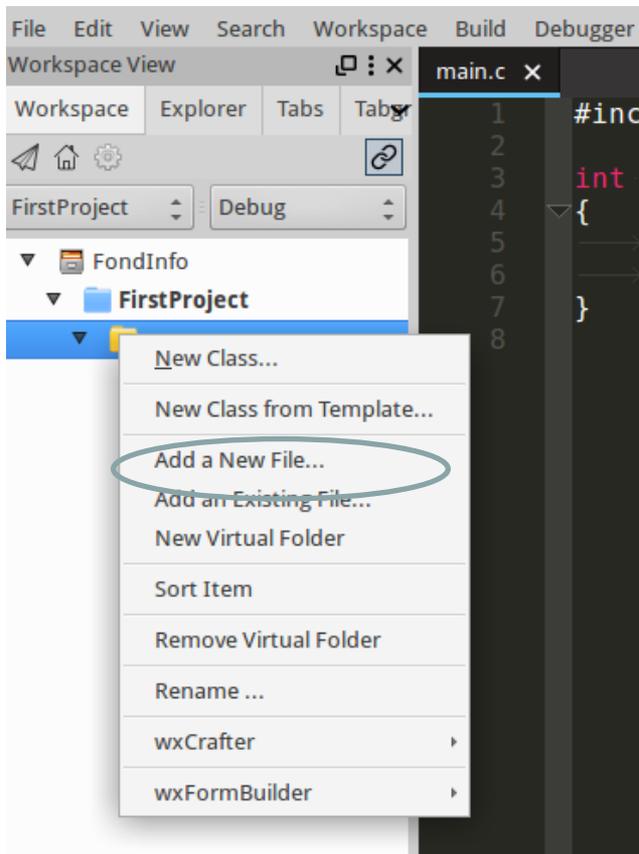
Alla creazione di un progetto, l'IDE **Codelite** crea automaticamente il file principale contenente la funzione main del programma.

*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

The screenshot shows the CodeLite IDE interface. On the left is the 'View' menu with options like 'Word Wrap', 'Toggle Current Fold', 'Toggle All Folds', 'Toggle All Topmost Folds in Selection', 'Toggle Every Fold in Selection', 'Display EOL', 'Show Whitespace', 'Full Screen...', 'Show Welcome Page', 'Load Welcome Page at Startup', 'Output Pane', 'Workspace Pane', 'Navigation Bar', 'Debugger Pane', 'Show Status Bar', and 'Show ToolBar'. The 'Navigation Bar' is highlighted in blue. The main editor area shows a C++ file named 'main.c' with the following code:

```
1 #include <stdio.h>
2
3 int main(int argc, char **argv)
4 {
5     printf("Hello World\n");
6     return 0;
7 }
8
```

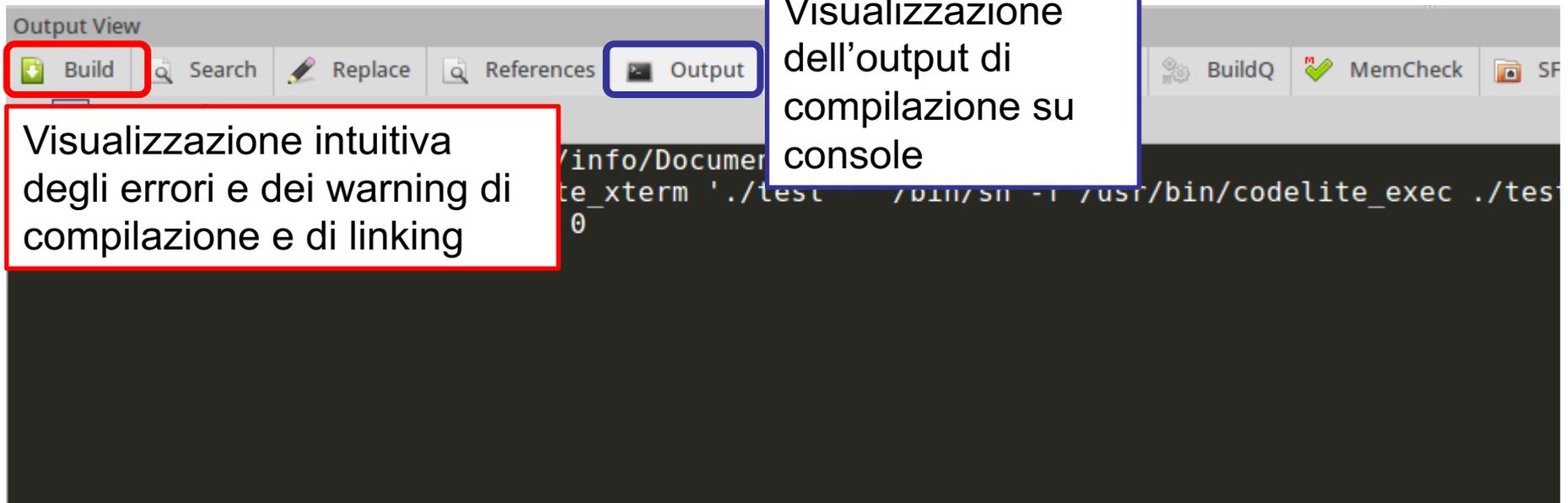
Annotations include:

- A red box around the toolbar icons with the text: "Elenco delle funzioni: per raggiungere velocemente un punto nel codice".
- A red box around the function signature 'main(int argc, char \*\*argv)' in the dropdown menu with the text: "Elenco delle funzioni: per raggiungere velocemente un punto nel codice".
- A blue box around the tab bar with the text: "Barra delle Tab: veloce accesso ai file sorgenti aperti".
- A green box around the line numbers on the left with the text: "Numeri di linea".
- A black box around the text 'EditorView' with the text: "EditorView".
- A black box around the text 'Per attivare l'elenco delle funzioni' with the text: "Per attivare l'elenco delle funzioni".

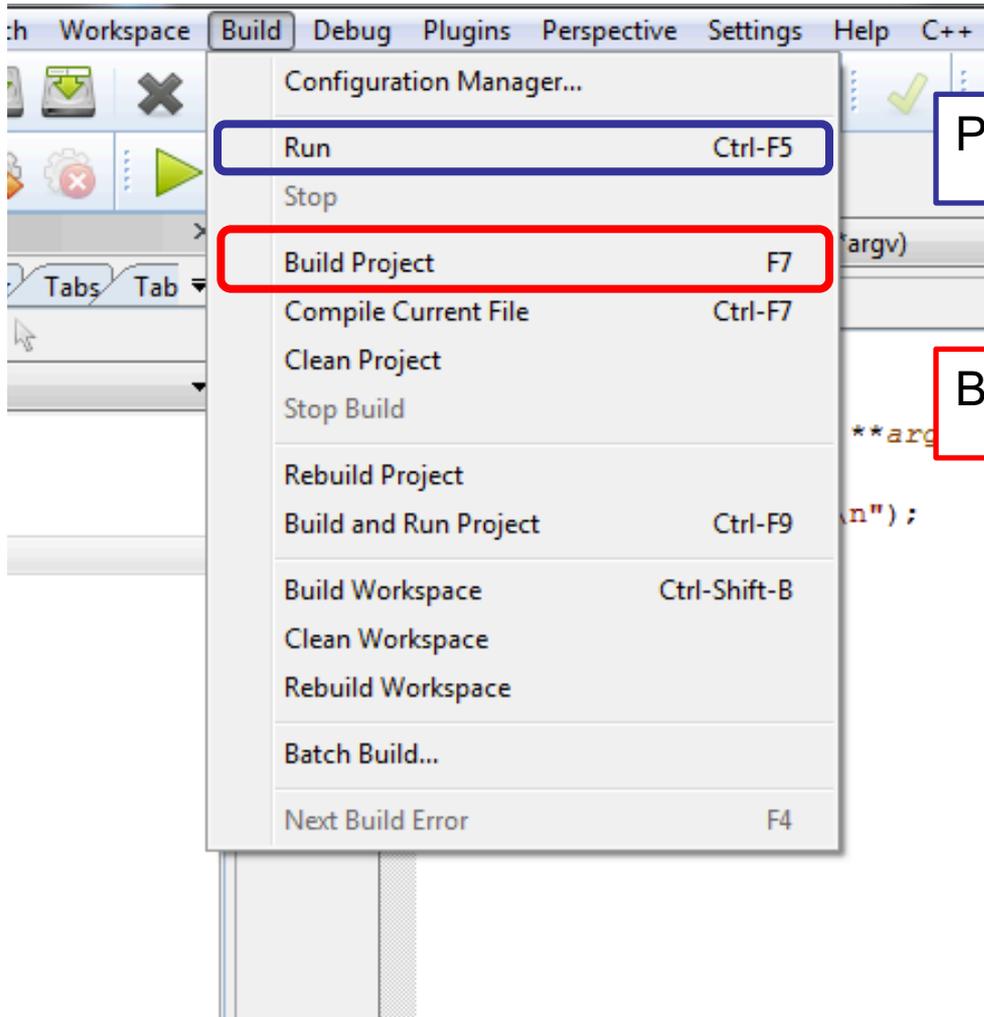
The status bar at the bottom shows: Replace, References, Output, Trace, Tasks, BuildQ, CppCheck, and CScope. The command prompt at the bottom shows: \Users\alessiobonfietti\Desktop\TestWks\Esempio01\Debug ./Esempio01 de: -1073741510

# Progetti in Codelite

Output View



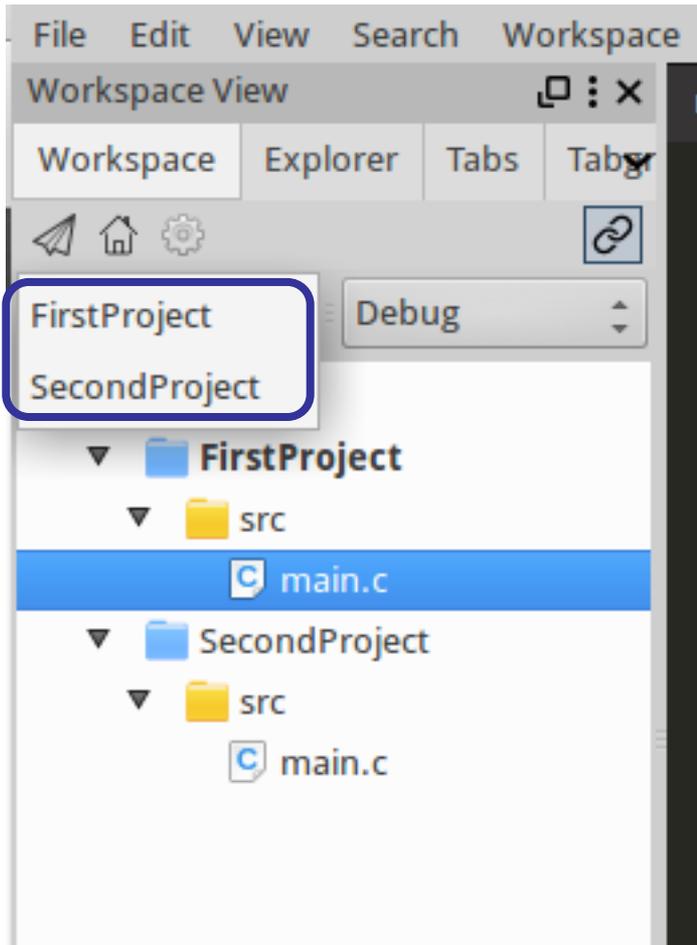
# Progetti in Codelite



Per Eseguire il programma

Build = Compile + Link

# Progetti in Codelite



## Gestione di più progetti

Tramite il menù a tendina di sinistra è possibile selezionare il progetto su cui lavorare (Il progetto selezionato in grassetto sarà quello su cui verranno applicate le operazioni di **Build + Run**)

# Build: Warning

The image shows a screenshot of an IDE with a C program in a file named `main.c`. The code is as follows:

```
1 #include <stdio.h>
2
3 int main(int argc, char **argv)
4 {
5     int i;
6     printf("hello world\n");
7     return 0;
8 }
9
```

A red box highlights the line `int i;` in the code editor. A yellow arrow points to this line. A white text box with a red border contains the text "Indicazione del warning".

The Output View at the bottom shows the following build output:

```
make[1]: Leaving directory `/home/info/Documenti/info/test'
make[1]: Entering directory `/home/info/Documenti/info/test'
/usr/bin/gcc -c "/home/info/Documenti/info/test/main.c" -g -O0 -Wall -o ./Debug/main.c.o -I
/home/info/Documenti/info/test/main.c: In function 'main':
/home/info/Documenti/info/test/main.c:5:9: warning: unused variable 'i' [-Wunused-variable]
    int i;
    ^
/usr/bin/g++ -o ./Debug/test @ test.txt -L.
```

A red box highlights the warning message in the output view.

The status bar at the bottom indicates the current position is Ln 6, Col 15, Pos 77, and the file is C++.

# Build: Errors

The image shows a screenshot of an IDE (likely Visual Studio Code) with a C program in the main editor and its compilation output in the Output View. The code in the editor is:

```
1 int main(int argc, char **argv)
2 {
3     int i = 5;
4     printf("hello world\n");
5     i = 2;
6     return 0;
7 }
```

The Output View shows the following compilation output:

```
make[1]: Leaving directory `/home/info/Documenti/info/test'
make[1]: Entering directory `/home/info/Documenti/info/test'
/usr/bin/gcc -c "/home/info/Documenti/info/test/main.c" -g -O0 -Wall -o ./Debug/main.c.o -I
/home/info/Documenti/info/test/main.c: In function `main':
/home/info/Documenti/info/test/main.c:7:2: error: expected ',' or ';' before 'printf'
    printf("hello world\n");
    ^
/home/info/Documenti/info/test/main.c:5:9: warning: variable 'i' set but not used [-Wunused-bu
    int i = 5;
    ^
```

A blue box highlights the error message in the Output View, and an arrow points from it to the corresponding line in the code editor.

Indicazione degli errori

# ESAMIX

---

<http://esamix.labx>

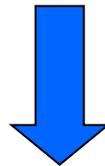
# Il Debugger

---

**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

---

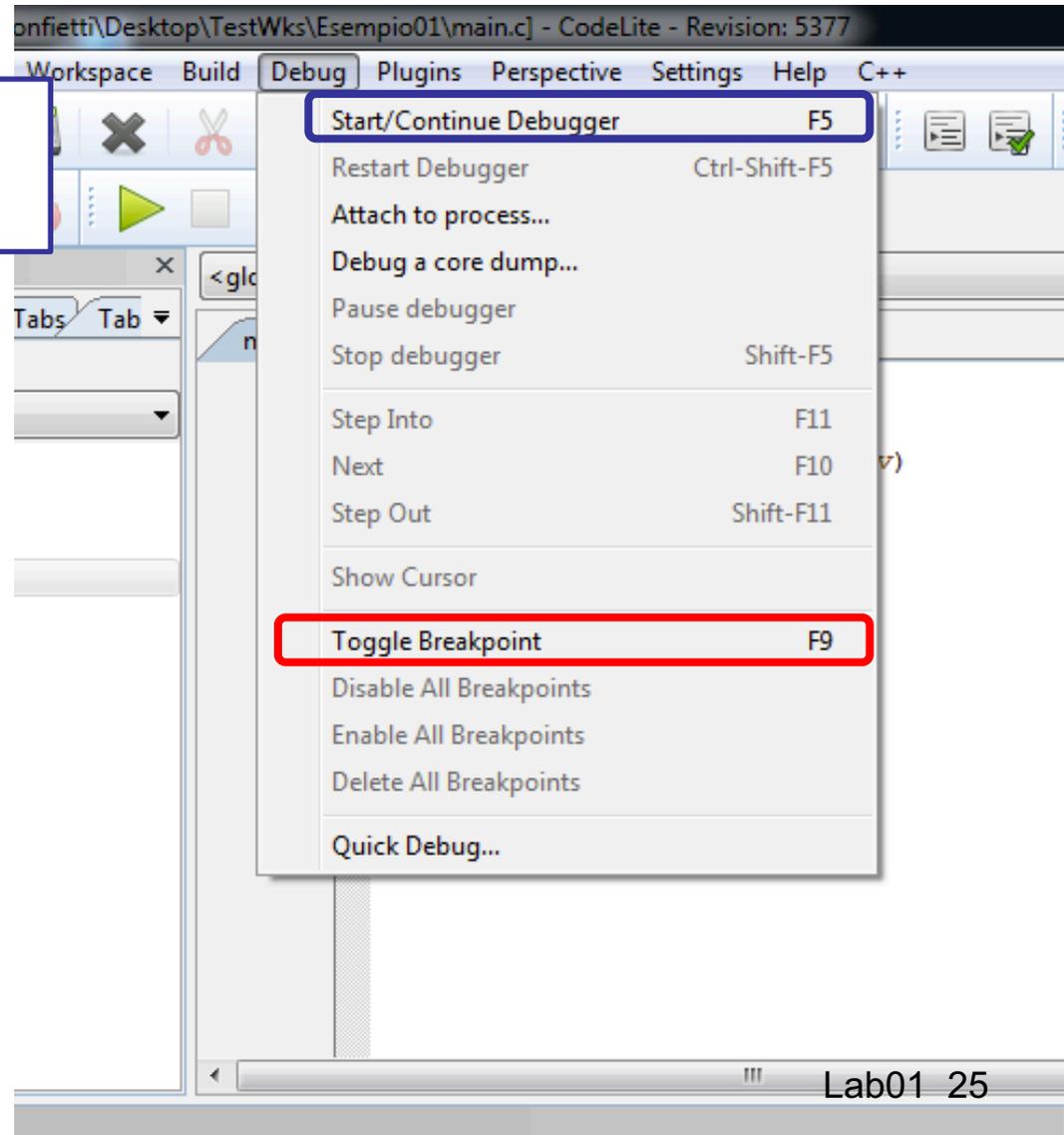
Sia **Codelite** sia altri ambienti di sviluppo incorporano un *debugger* con cui eseguire il programma,

- **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 image shows a screenshot of a debugger interface with several annotations:

- Comandi veloci Debug**: A red box highlights the top toolbar containing various debugging icons like play, stop, pause, and step-through.
- Debug Mode**: A black box highlights the 'Debug' button in the top-left corner of the workspace.
- Indicatore di posizione Debug**: A blue box highlights a small icon on the left margin of the code editor, indicating the current execution position.
- Locals: Vista dello stato corrente di esecuzione Variabili-Valori-Tipo**: A green box highlights the 'Locals' tab and the table below it, which shows the current state of execution variables.

```
int main(int argc, char **argv)
{
    int i = 5;
    printf("hello wor
    i = 2;
    return 0;
}
```

Name	Value	Type
▶ argc	1	int
▶ argv	0x7fffffff5c8	char **
▶ i	5	int

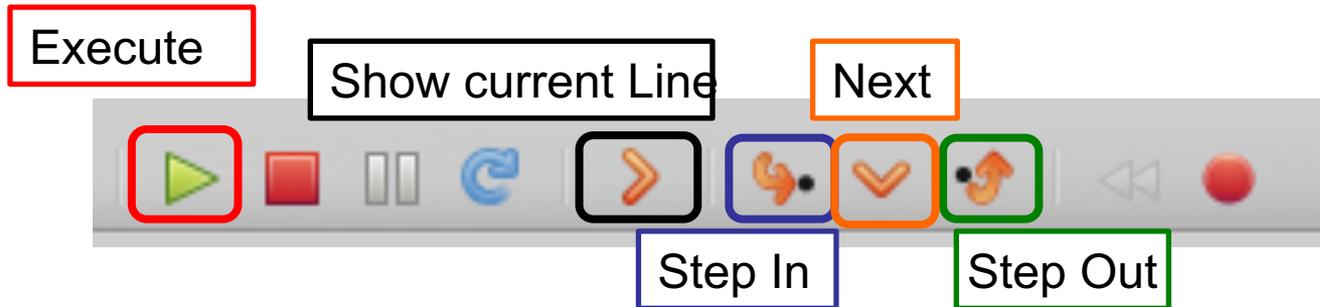
# Debugger: Come Procedere

---

- 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 funziona 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 image shows a debugger interface with a code editor and a memory view. The code editor displays the following C code:

```
int main(int argc, char **argv)
{
    int i = 5;
    char C[4] = {'a', 'b', 'c', '\0'};
    printf("hello world\n");
    i = 2;
    return 0;
}
```

The line `char C[4] = {'a', 'b', 'c', '\0'};` is highlighted with a red box. Below the code editor, the Debugger window shows the memory view for the variable `C`:

Name	Value	Type
argv	0x77777777	char
▼ C	[4]	char [4]
0	97 'a'	char
1	98 'b'	char
2	99 'c'	char
3	0 "	char

The memory view table is also highlighted with a red box. A text box on the right side of the image contains the text: "Rappresentazione Array statici".

# Debugger

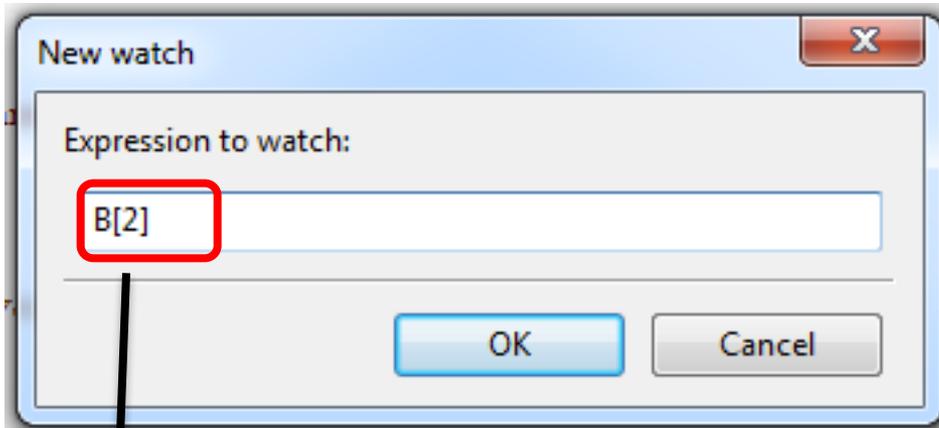
The screenshot shows a debugger interface with a code editor and a watches window. The code editor displays the following C code:

```
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\n");
20
21 stampa(var);
22 stampa(var2);
23 var3 = somma(var, var2);
24 stampa(var3);
25
26 return 0;
27 }
```

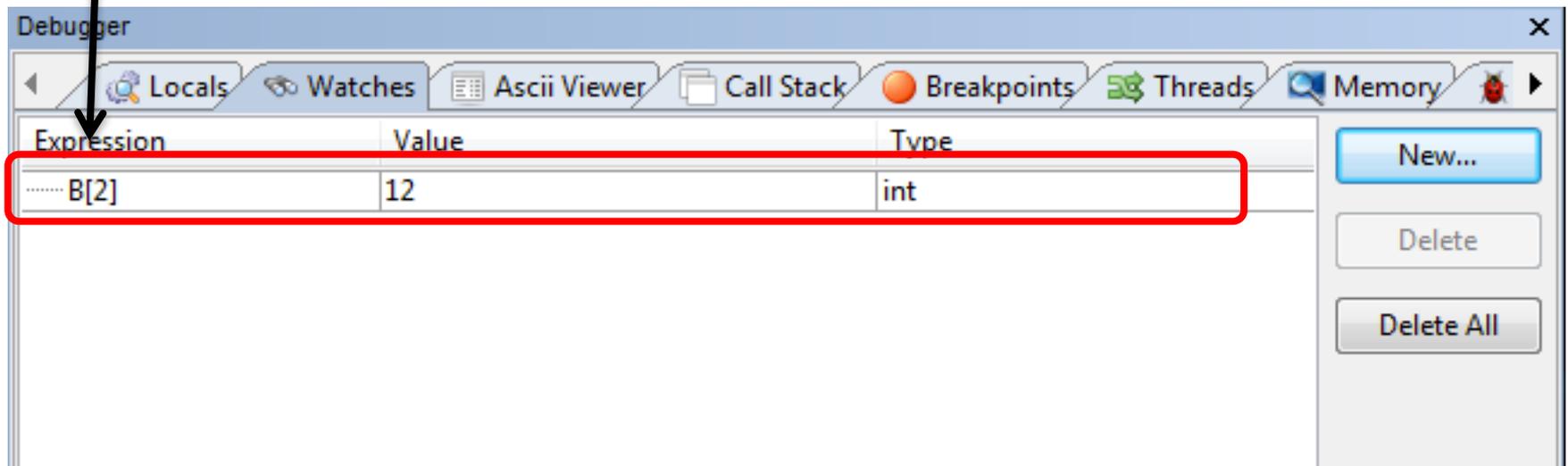
The line `B = (int*)malloc(sizeof(int)*4);` is highlighted with a red box. A callout box points to the watches window with the text "Rappresentazione parti non in stack: Watches". The watches window is also highlighted with a red box and contains the following data:

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



# Mac OS X Notes

---

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
```

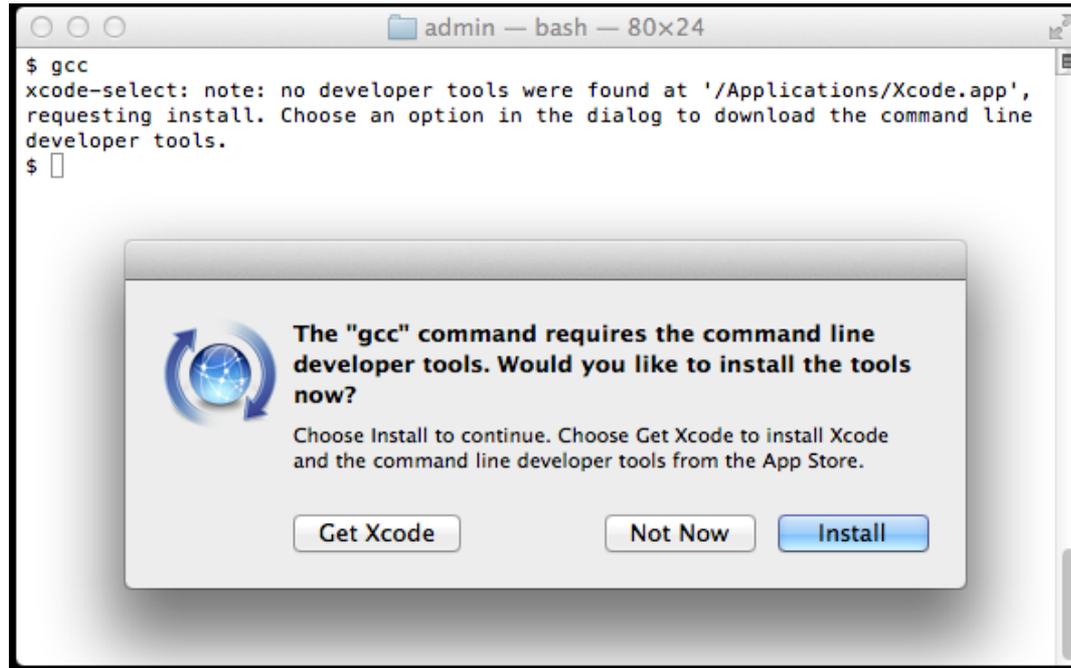
Se vi appare una scritta simile a questa va tutto bene:

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

Significa che il compilatore è già installato

# Mac OS X 10.10 Yosemite Notes

Altrimenti apparirà una finestra di installazione, tipo:

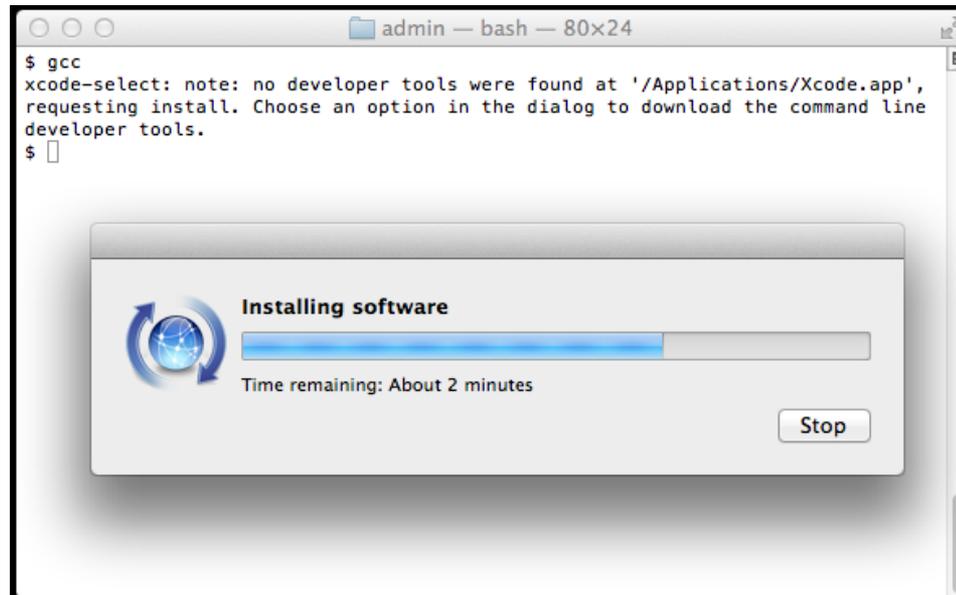


Cliccate **Install** per installare il compilatore (command line tools)

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

# Mac OS X 10.10 Yosemite Notes



Per verificare se è installato correttamente digitare nel terminale:

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

Si dovrebbe leggere una scritta tipo:

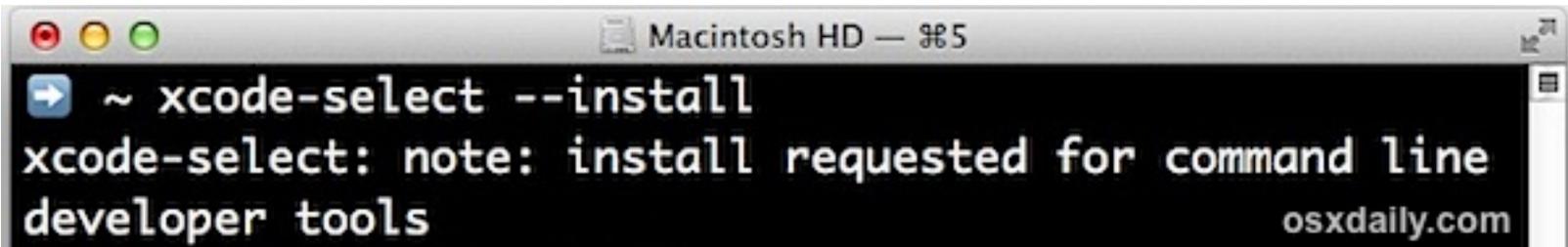
```
/Library/Developer/CommandLineTools
```

# Mac OS X Notes

---

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

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



```
Macintosh HD — 5
~ xcode-select --install
xcode-select: note: install requested for command line
developer tools
osxdaily.com
```

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

# Mac OS X Notes

---

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

# Debian/Ubuntu Notes

---

Potete trovare la guida all'installazione nel sito:

[http://codelite.org/LiteEditor/Repositories#  
toc1](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="
```

# Debian/Ubuntu Notes

---

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
```

# Windows Notes

---

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-toolkit, mingw-base, mingw-gcc-g++, mingw-make (tutti i pacchetti mingw-make)

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

# Windows Notes

---

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.



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

# Windows Notes

---

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

# Windows Notes

---

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;
}
```