

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*)**

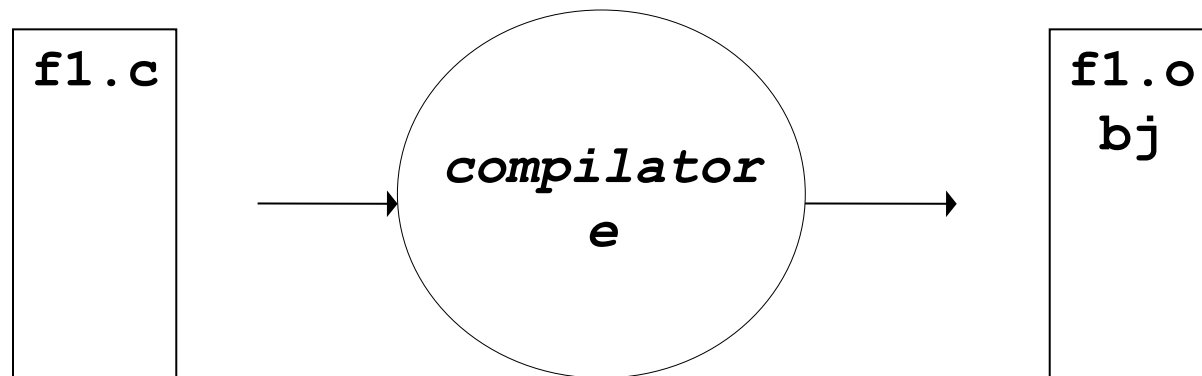
Il risultato sono uno o più file *oggetto*.

- **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**

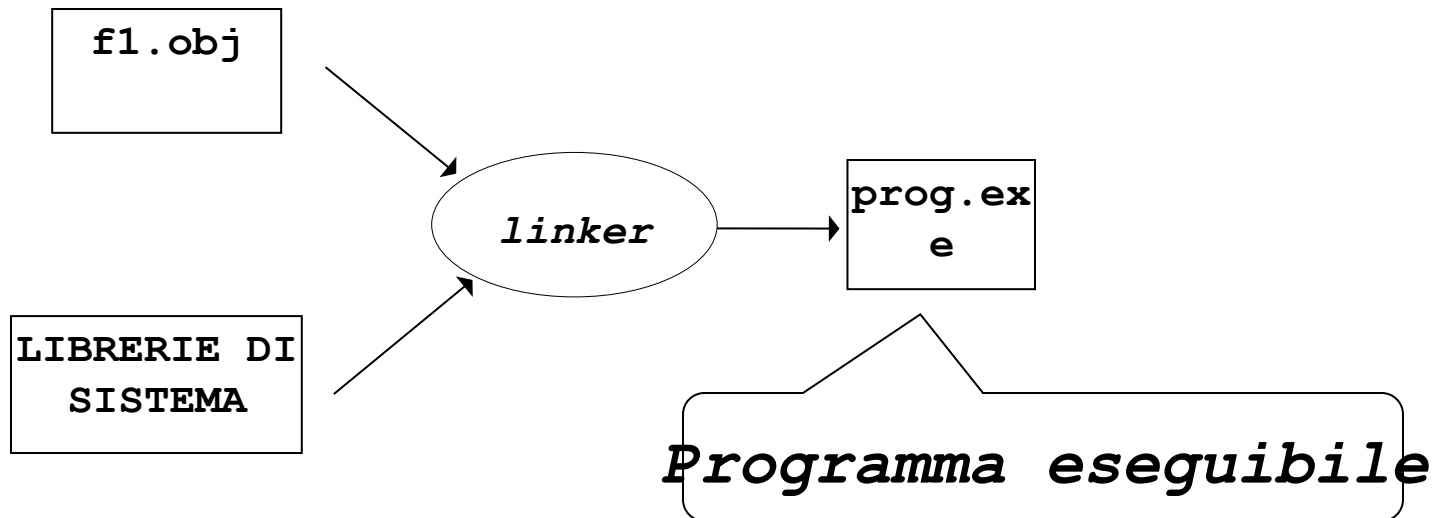


`f1.obj`: Una versione tradotta che però non è autonoma (e, quindi, non è direttamente eseguibile).

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.

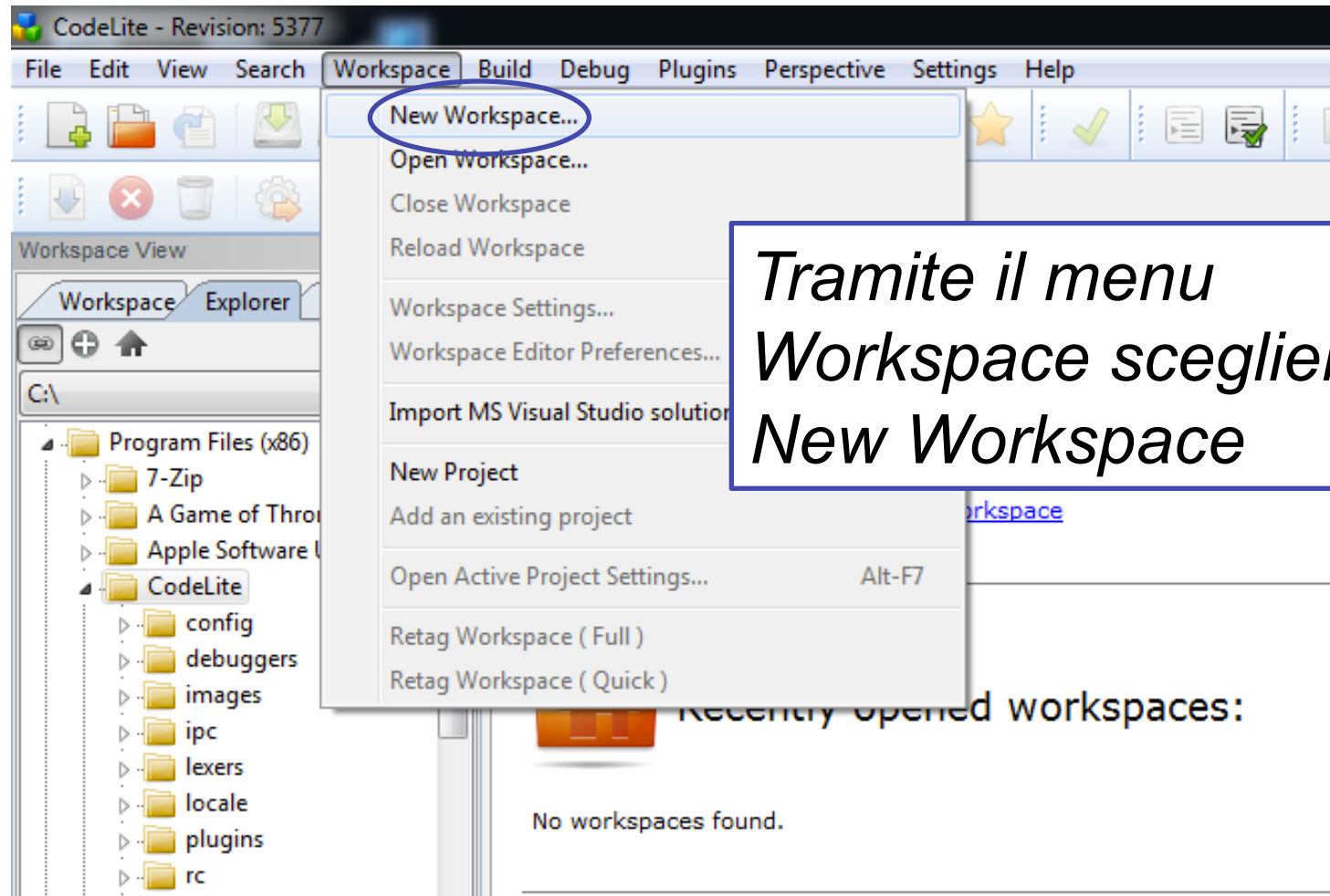
Progetti in Codelite

Download

an installer which includes

codelite IDE + MinGW suite (GNU toolchain + WinAPI)

Progetti in Codelite



Progetti in Codelite

ck Links:

Lite W

e a N

orkspa

as found.

New Workspace

Workspace Name:
TestWorkspace

Workspace Path:
C:\Users\alessiobonfietti\Desktop\TestWks

☐ Create the workspace under a separate directory

File Name:
C:\Users\alessiobonfietti\Desktop\TestWks\TestWorkspace.

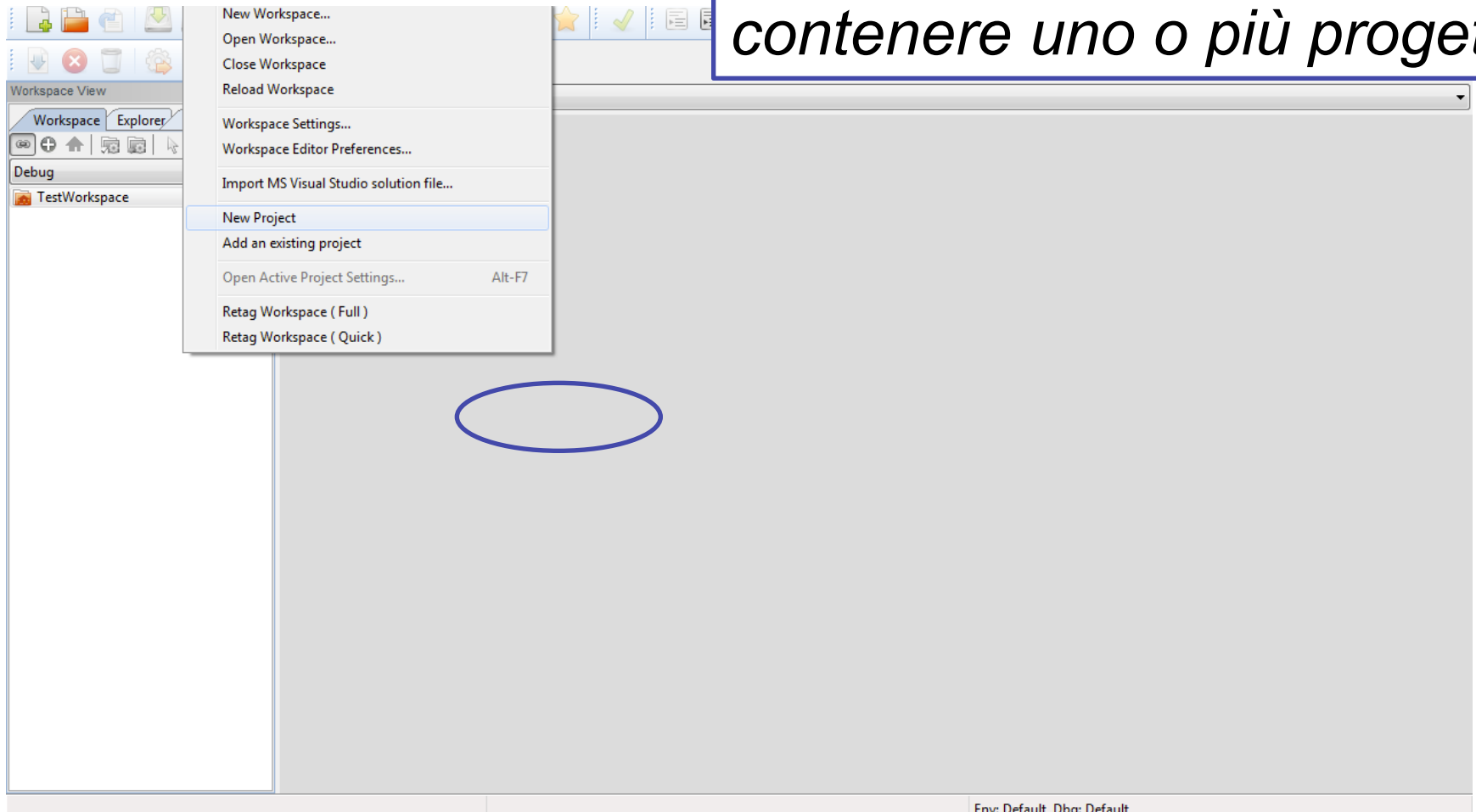
Create Cancel

*Inserire il nome del
Workspace ed il percorso*

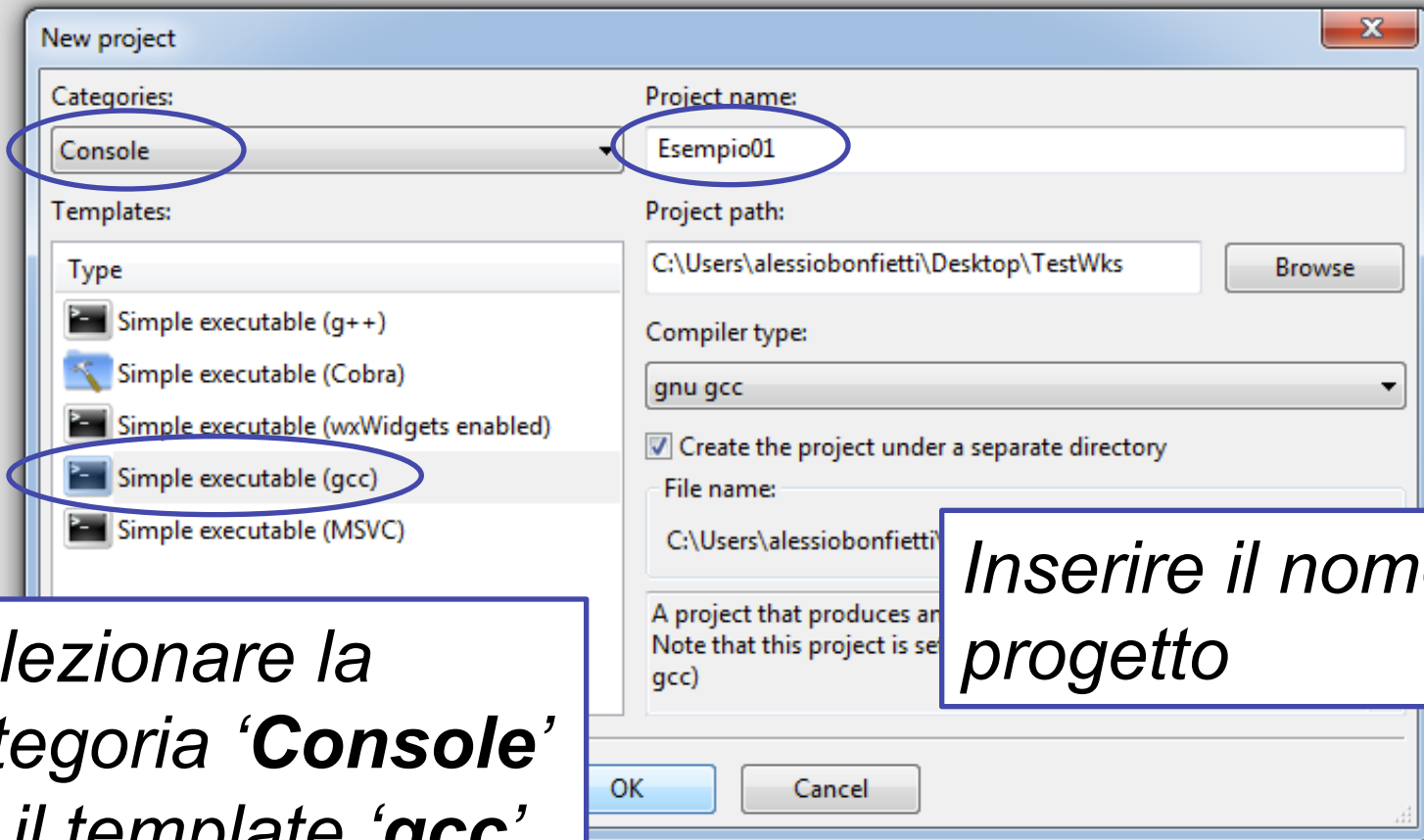
Si consiglia di lavorare sempre in c:\temp

Progetti in Codelite

Ogni workspace può contenere uno o più progetti



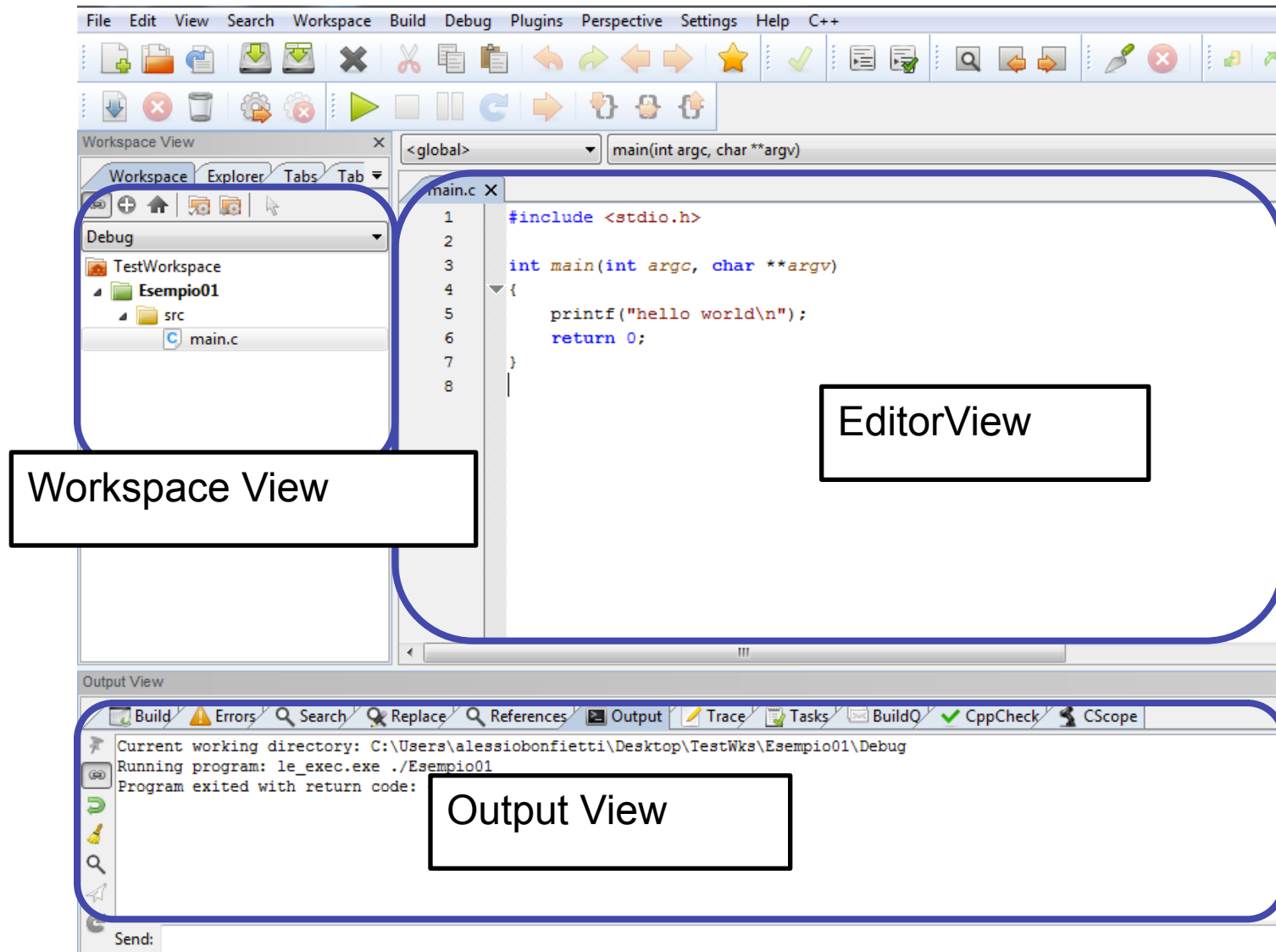
Progetti in Codelite



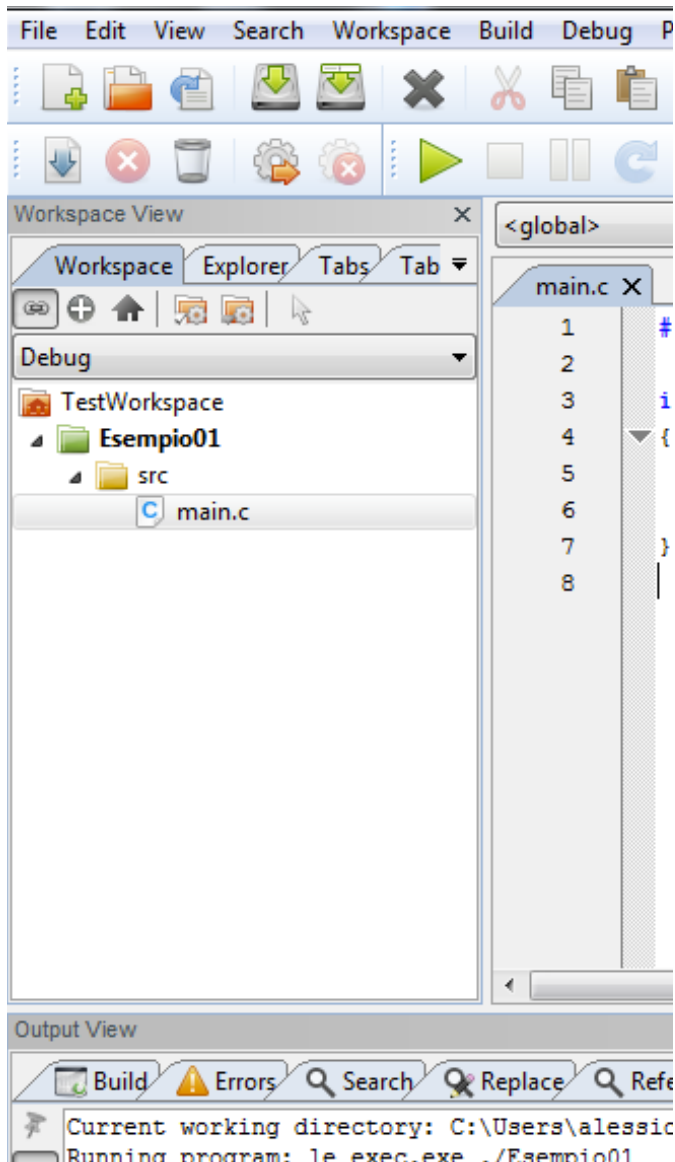
*Selezionare la categoria '**Console**' ed il template '**gcc**'*

Inserire il nome del progetto

Progetti in Codelite



Progetti in Codelite



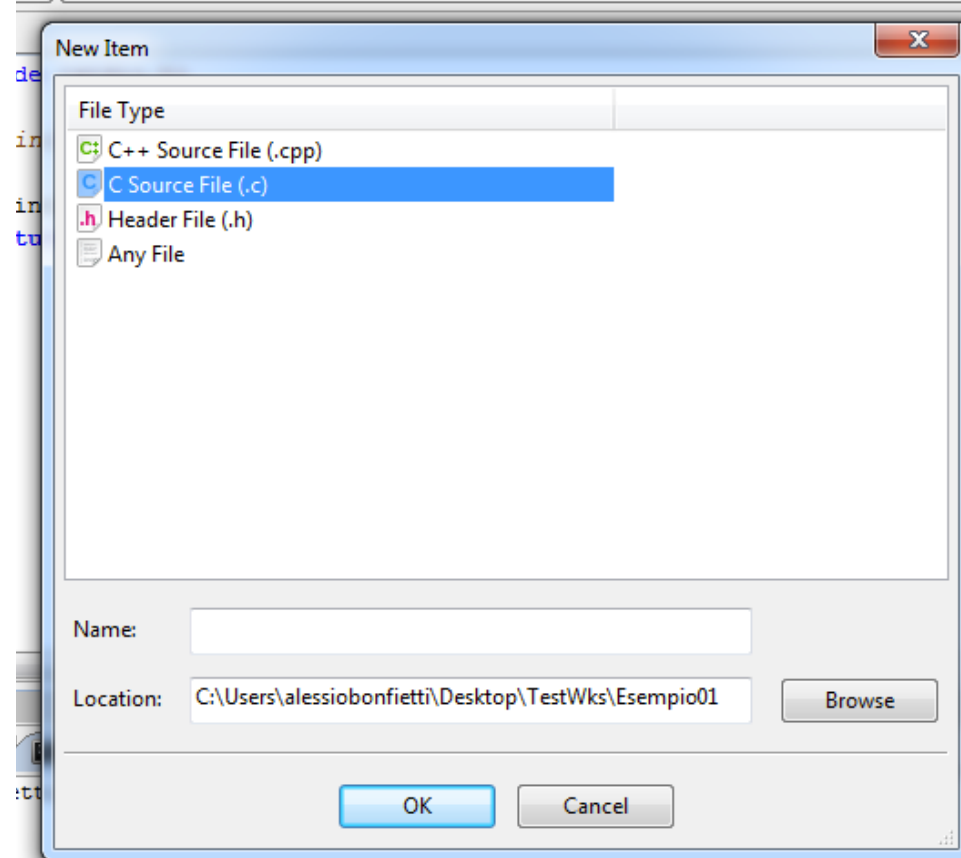
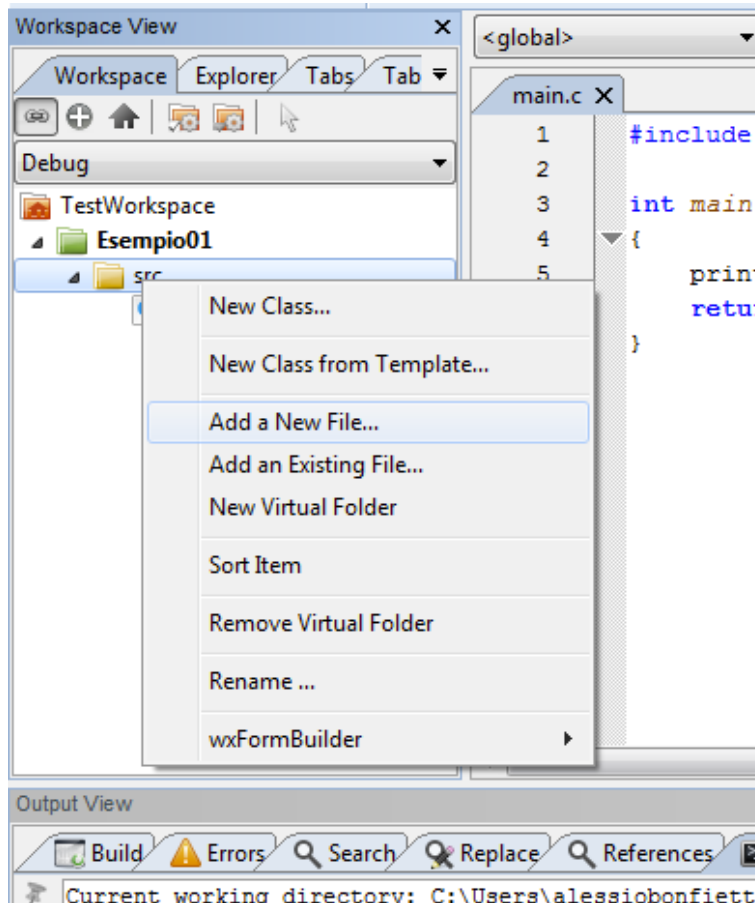
Workspace View

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

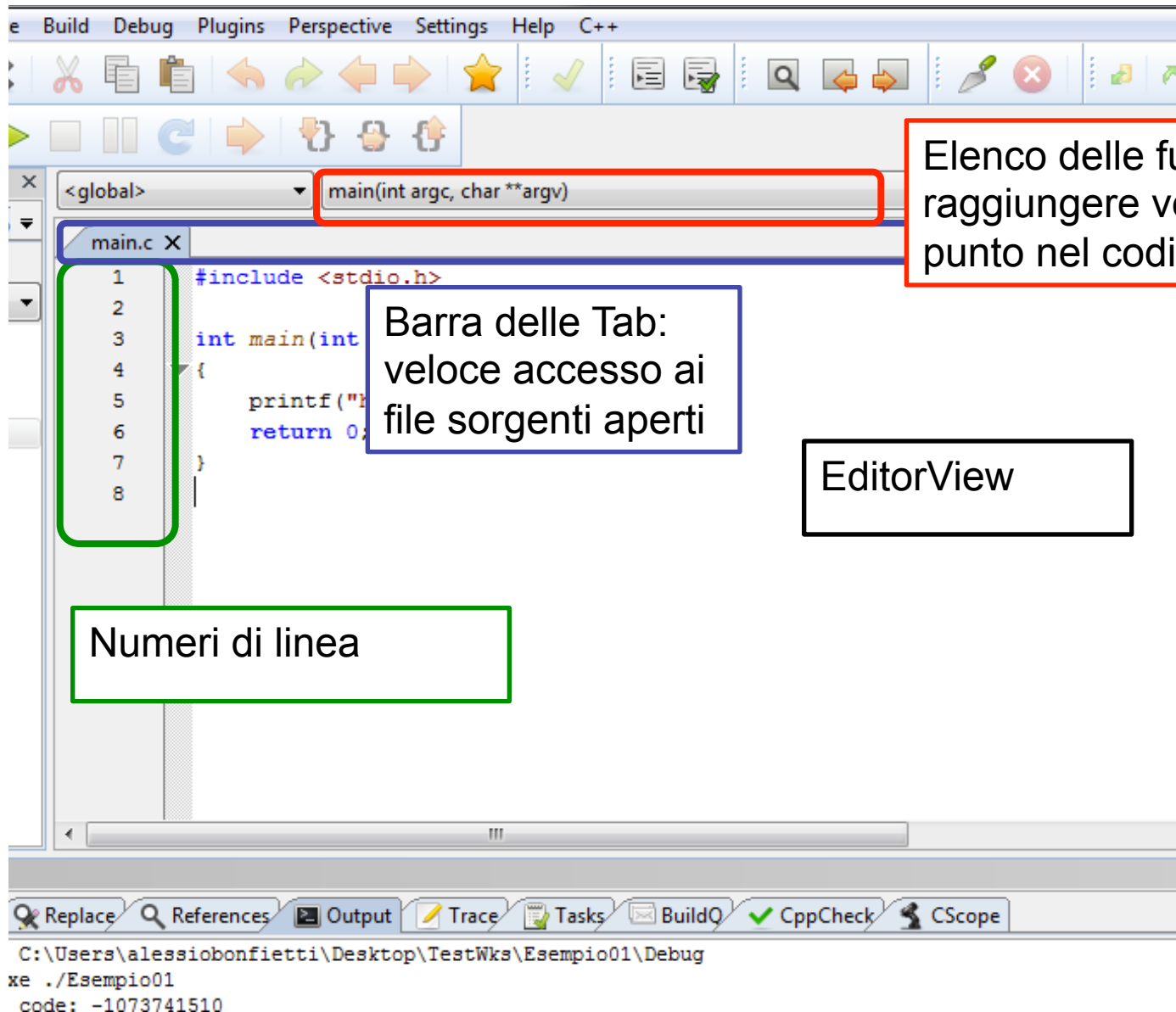
Da questa interfaccia è dedicata alla gestione dei file sorgente

Progetti in Codelite

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

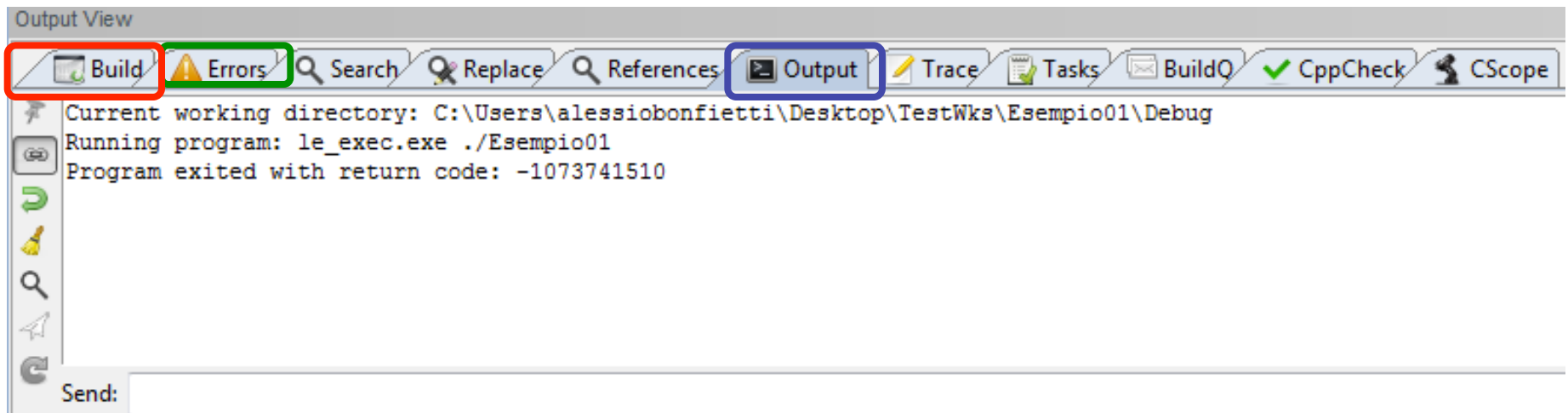


Progetti in CodeLite



Progetti in Codelite

Output View

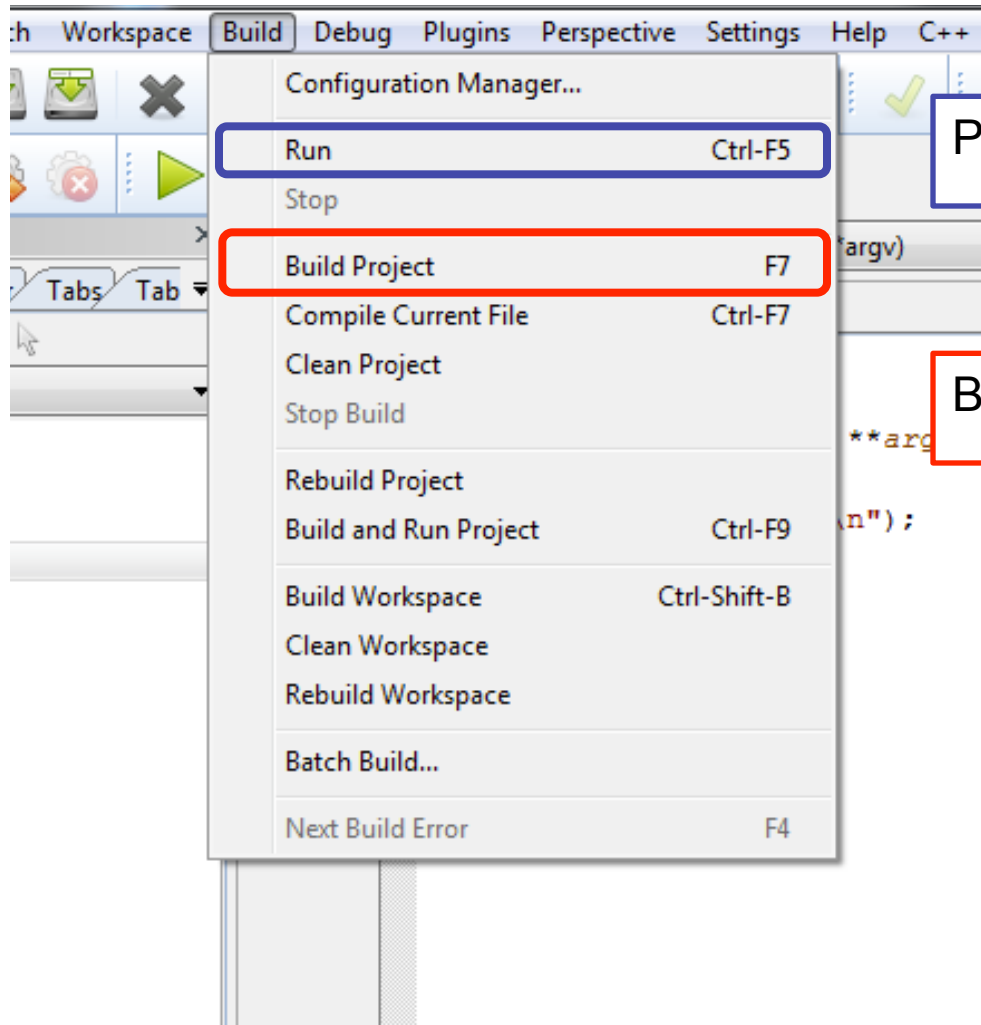


Controllo degli errori con visualizzazione dei comandi di compiling e di linking

Visualizzazione dell'output di compilazione su console

Visualizzazione intuitiva degli errori e dei warning di compilazione

Progetti in Codelite



Per Eseguire il programma

Build = Compile + Link

ESAMIX

`http://esamix.labx`

Progetti in Codelite



Build: Warning

The screenshot shows a C++ IDE with a project named 'Esempio01' and a source file 'main.c'. The code in 'main.c' is as follows:

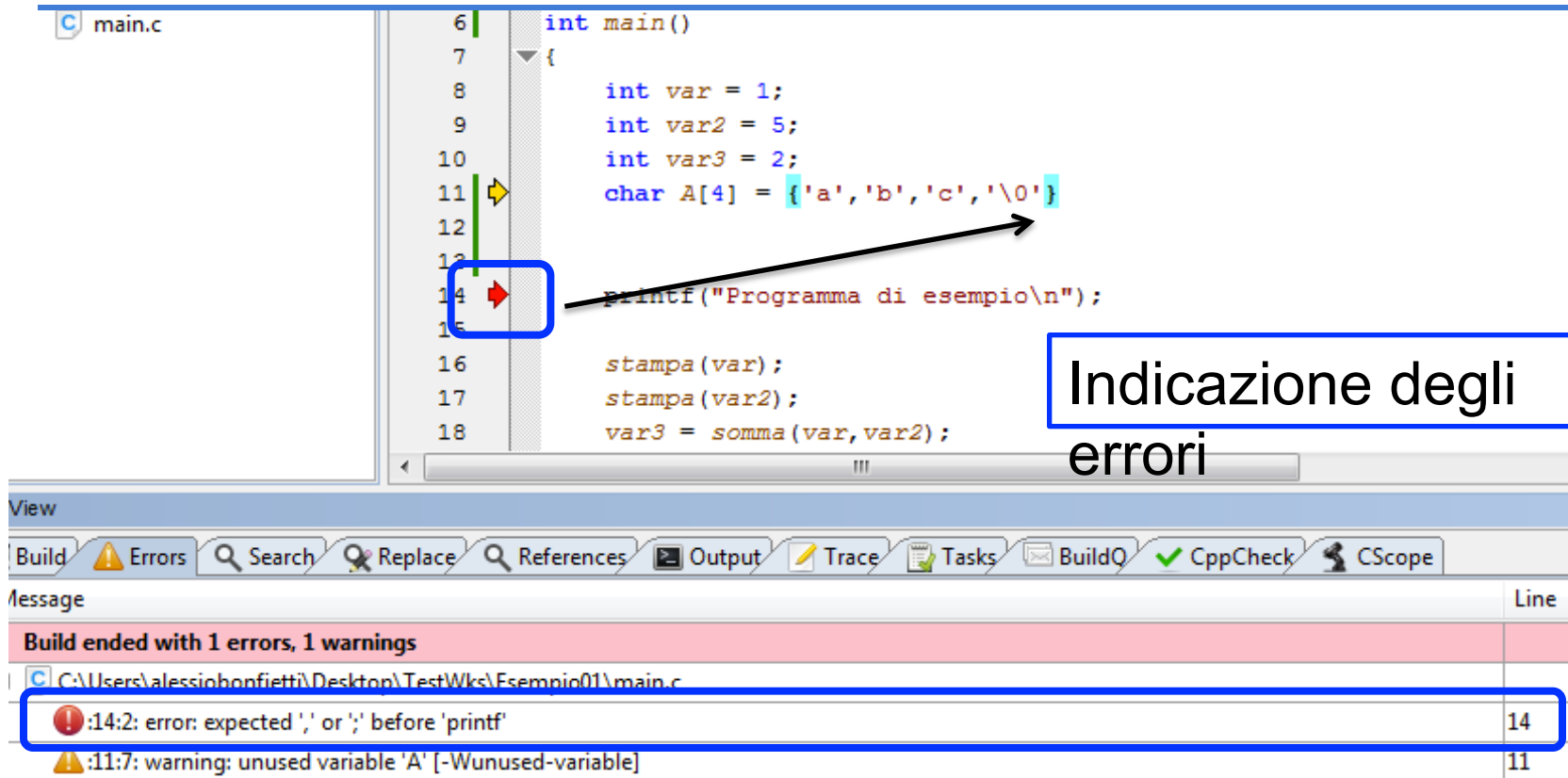
```
1  #include <stdio.h>
2
3  void stampa(int num);
4  int somma(int a, int b);
5
6  int main()
7  {
8      int var = 1;
9      int var2 = 5;
10     int var3 = 2;
11     char A[4] = {'a', 'b', 'c', '\0'};
12
13
14     printf("Programma di esempio\n");
15
16     stampa(var);
17     stampa(var2);
18 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

A red box highlights the line `char A[4] = {'a', 'b', 'c', '\0'};` on line 11, with a yellow arrow pointing to it. Another red box highlights the text 'Indicazione del warning'.

The bottom of the IDE shows the 'Output View' with a 'Message' tab. A red box highlights the following message:

Message	Line
Build ended with 1 warning(s)	
C:\Users\aleSSIobonfietti\Desktop\TestWks\Esempio01\main.c	
:11:7: warning: unused variable 'A' [-Wunused-variable]	11

Build: Errors



The screenshot shows a code editor with a C program. The code is as follows:

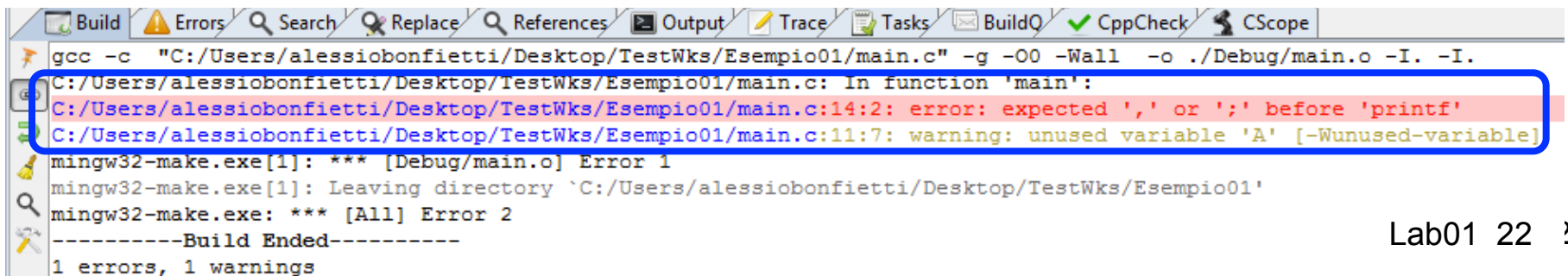
```
6 | int main()  
7 | {  
8 |     int var = 1;  
9 |     int var2 = 5;  
10 |    int var3 = 2;  
11 |    char A[4] = {'a', 'b', 'c', '\0'}  
12 |  
13 |  
14 |    printf("Programma di esempio\n");  
15 |  
16 |    stampa(var);  
17 |    stampa(var2);  
18 |    var3 = somma(var, var2);
```

A red arrow points to the error message in the console:

Indicazione degli errori

Build ended with 1 errors, 1 warnings

Message	Line
C:\Users\allesiobonfietti\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 the following output:

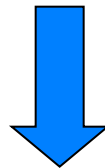
```
gcc -c "C:/Users/allesiobonfietti/Desktop/TestWks/Esempio01/main.c" -g -O0 -Wall -o ./Debug/main.o -I. -I.  
C:/Users/allesiobonfietti/Desktop/TestWks/Esempio01/main.c: In function 'main':  
C:/Users/allesiobonfietti/Desktop/TestWks/Esempio01/main.c:14:2: error: expected ',' or ';' before 'printf'  
C:/Users/allesiobonfietti/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/allesiobonfietti/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

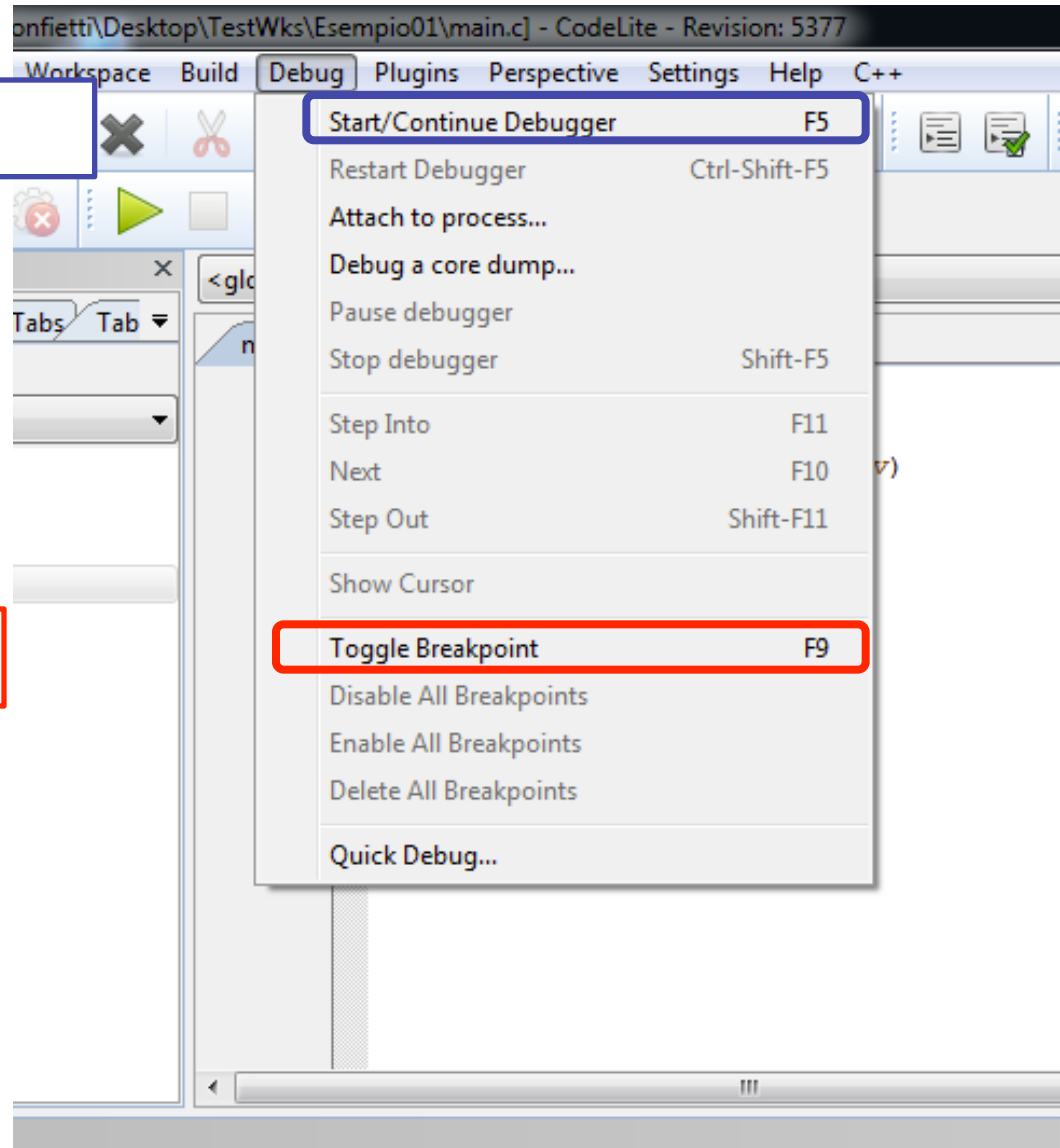
Sia **Codelite** sia altri ambienti di sviluppo incorporano un *debugger* con cui eseguire il programma,

- *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

Comandi veloci
Debug

Debug Mode

Indicatore di
posizione Debug

Locals: Vista dello stato corrente di esecuzione
Variabili-Valori-Tipo

Name	Value	Type
var	1	int
var2	5	int
var3	2	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

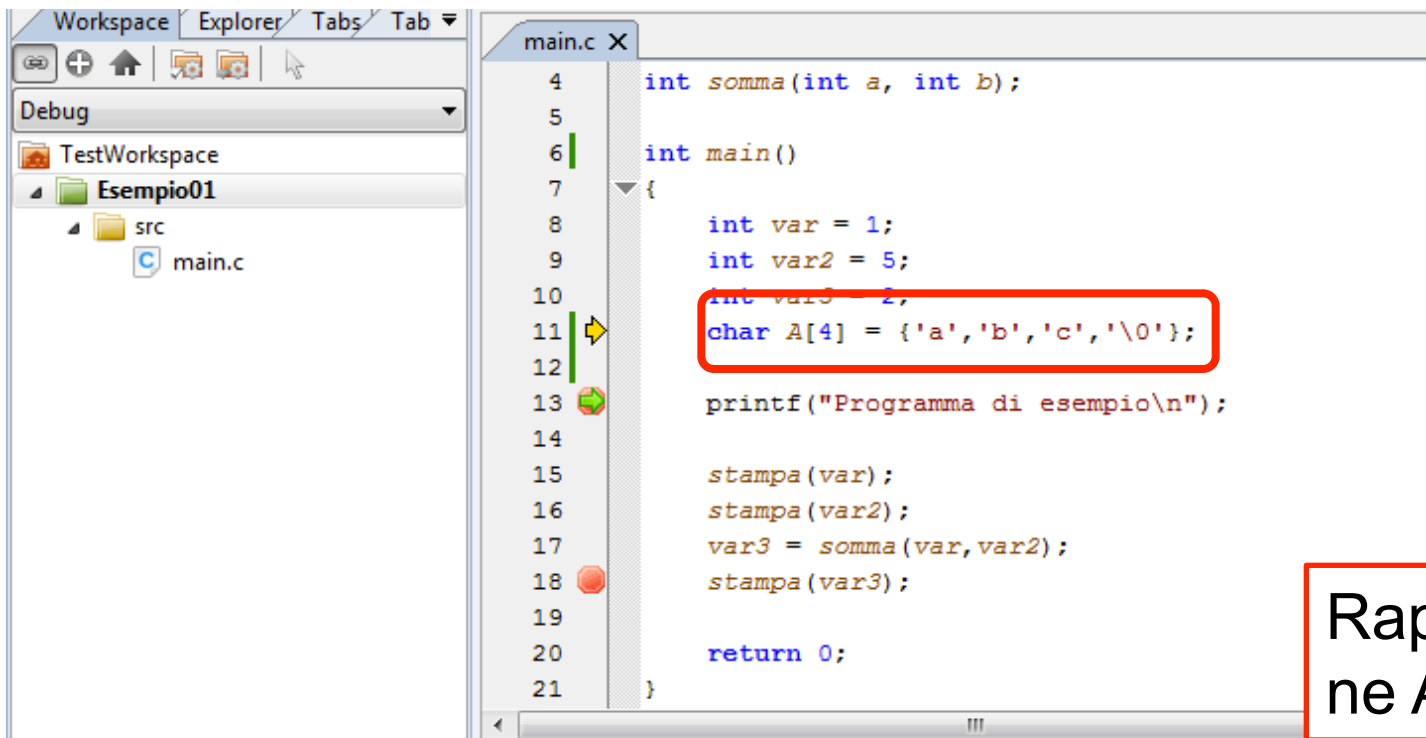
The screenshot shows a C++ IDE with a debugger. The workspace view on the left shows a project named 'Esempio01' with a source file 'main.c'. The code editor displays the following C++ code:

```
4 int somma(int a, int b);  
5  
6 int main()  
7 {  
8     int var1 = 1;  
9     int var2 = 5;  
10    int var3 = 2;  
11  
12    printf("Programma di esempio\n");  
13  
14    stampa(var1);  
15    stampa(var2);  
16    var3 = somma(var1, var2);  
17    stampa(var3);  
18  
19    return 0;  
20 }
```

The debugger toolbar at the top has several icons. The 'Execute' icon (a green play button) is highlighted with a red box and labeled 'Execute'. The 'Step In' icon (a blue arrow pointing right) is highlighted with a blue box and labeled 'Step In'. The 'Next' icon (an orange arrow pointing right) is highlighted with an orange box and labeled 'Nex'. The 'Step Out' icon (a green arrow pointing right) is highlighted with a green box and labeled 'Step Out'. The bottom panel shows the 'Locals' tab with the following variables:

Name	Value	Type
var	1	int
var2	5	int
var3	2	int

Debugger



Rappresentazio
ne Array statici

The screenshot shows the debugger's `Locals` window. The array `A` is expanded, showing its memory representation. The array is of type `char [4]` and contains the characters `'a'`, `'b'`, `'c'`, and `'\\0'`.

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

Debugger

The screenshot shows a C program being debugged. The code is as follows:

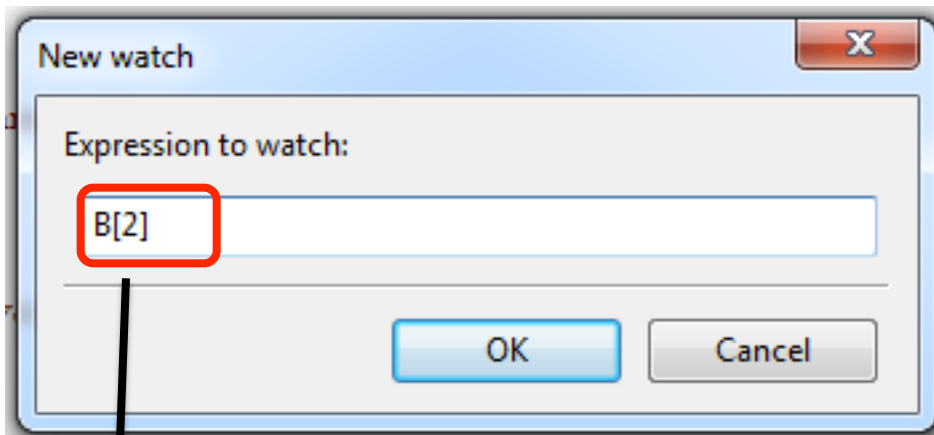
```
10 int var3 = 2;
11 char A[4] = {'a', 'b', 'c', '\0'};
12 int * B;
13
14 B = (int*)malloc(sizeof(int)*4);
15
16 B[0] = 5;
17 B[2] = 12;
18
19 printf("Programma di esempio\n");
20
21 stampa(var);
22 stampa(var2);
23 var3 = somma(var, var2);
24 stampa(var3);
25
26 return 0;
27 }
```

The debugger's 'Watches' window is open, showing the following variables and their values:

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

Rappresentazione
parti non in
stack: **Watches**

Debugger



Rappresentazione
parti non in
stack:Watches

