Fondamenti di Informatica T-1
CdS Ingegneria Informatica

Introduzione agli ambienti di sviluppo

MS Visual Studio 2005/2008/2010 ... 2017
CodeLite 6.1.1

Outline

- Solution/Workspace e Project
- IDE e linguaggio C
- Schermata principale
- Aggiungere file a un progetto
- Compilare ed eseguire un programma
- Debug di un programma

- Appendice A: “Che cosa fare se…”
- Appendice B: “Creare un progetto per il C”
- Appendice C: (VS2017 only) scanf deprecated...

Solutions/Workspace e Projects

- Nei moderni IDE ogni programma si sviluppa come un “progetto”…
- Un progetto contiene tutte le informazioni utili/necessarie per realizzare il programma
  - Elenco dei file sorgenti che compongono quel programma
  - Opzioni particolari relative allo specifico progetto

Solutions/Workspace e Projects

- A volte un programma “usa” funzionalità offerte da un altro programma
- In tal caso, è utile avere due progetti separati (uno per ogni programma)…
- … ma è utile anche raggruppare i programmi
  - secondo criteri di utilizzo (il programma A usa il programma B)
  - secondo criteri di affinità funzionali (i programmi A e B svolgono compiti molto simili)
  - secondo criteri di composizione (i programmi A e B condividono lo stesso componente)
  - …

Solutions/Workspace e Projects

- Una Solution (termine Microsoft) o un Workspace (termine CodeLite ed Eclipse) è un insieme di progetti, raggruppati secondo qualche criterio o esigenza
- Una Solution/Workspace è composta da:
  - uno o più progetti
  - opzioni particolari relative alla specifica solution
- Vantaggi:
  - Riusabilità dei singoli progetti
  - Modularità nella realizzazione di sistemi complessi

Solutions/Workspace e Projects

- In Visual Studio/CodeLite, ogni progetto è parte di almeno una solution/workspace. Quindi, nell'ambito di questo corso…

- Ogni programma sarà un progetto diverso
- Per ogni progetto, una Solution/Workspace distinta, contenente solo quel progetto

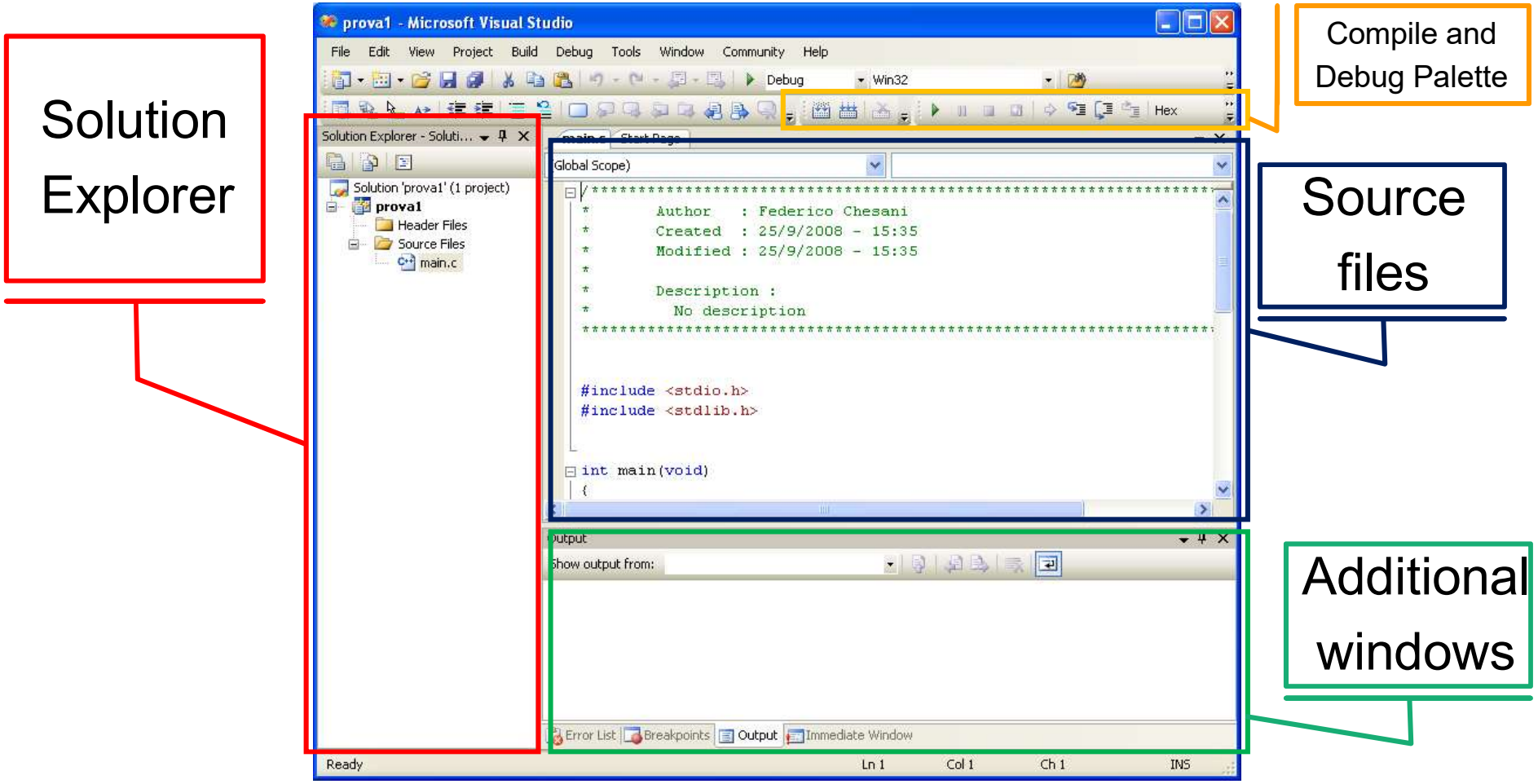
IDEs e il linguaggio C

- Visual Studio/CodeLite “a default” supportano il linguaggio C++, non direttamente il linguaggio C

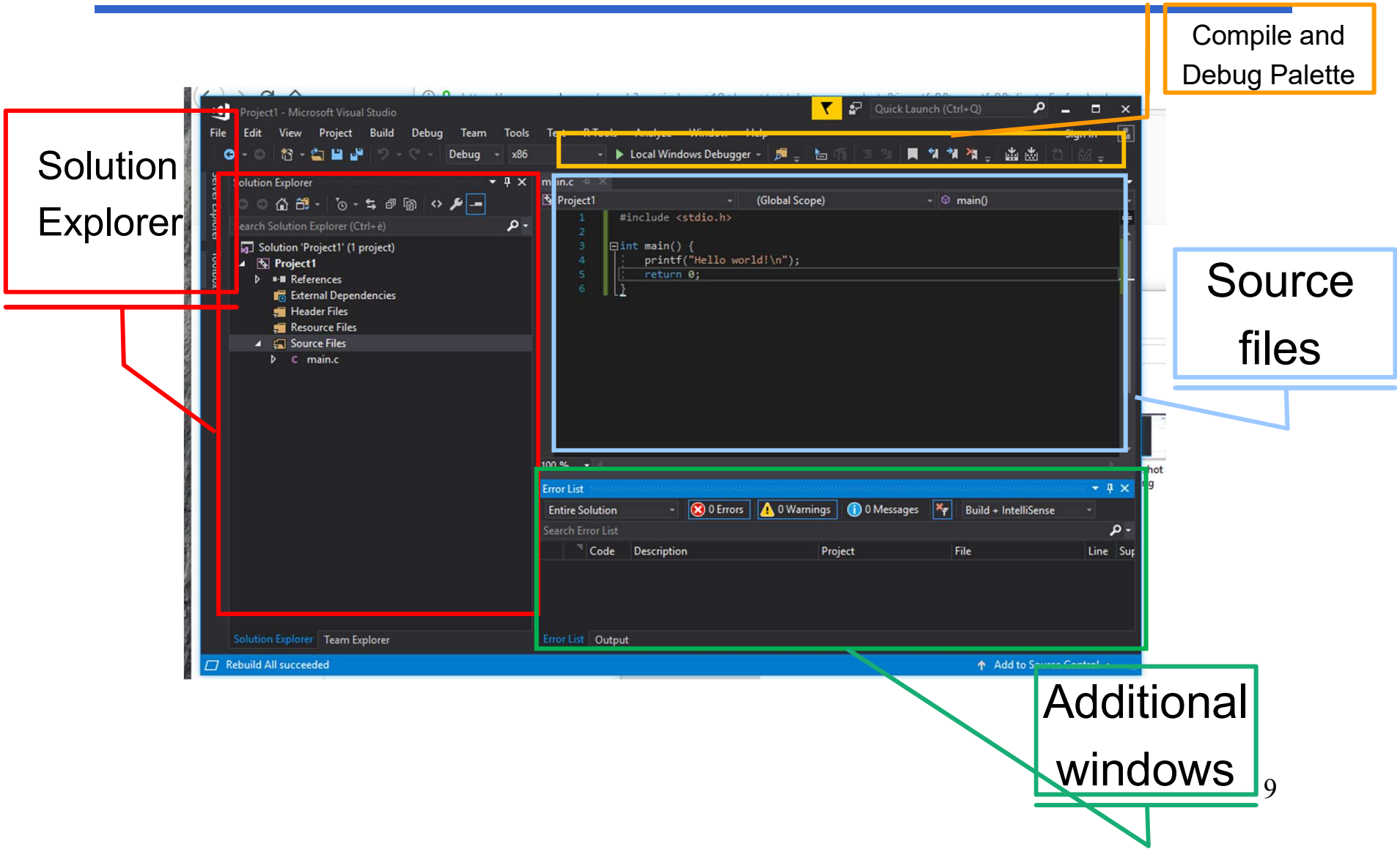
- C++ è un’estensione del C…
  - È possibile usare Visual Studio/CodeLite per realizzare programmi in C

- … può essere vantaggioso specificare nelle opzioni di progetto che si sta scrivendo un programma in linguaggio C!!! Altrimenti:
  - Il compilatore non rileva alcuni errori (ma in questo corso non accadrà...)
  - Può segnalare errori inconsistenti con le regole del linguaggio C

Visual Studio2010 schermata principale



Visual Studio 2017 schermata principale



Visual Studio: schermata principale

- **Solution Explorer**

  Mostra l'elenco dei progetti e dei file appartenenti ad ogni progetto. Per aprire un file, basta fare "doppio click" su di esso…

- **Source files**

  Mostra i file aperti, ogni file in un tab separato

- **Additional Windows**

  Mostrano alcune finstre ausiliarie molto importanti, quali:

  – "Output": mostra i messaggi di errore o di successo forniti dal framework

  – "Error List": elenco degli errori e dei warning rilevati in fase di compilazione. Cliccando su un errore, viene aperto il file corrispondente, e il cursore si posiziona nel luogo dove il compilatore presume ci sia l'errore…

- **Compile e Debug Palette**

  Contengono i pulsanti per compilare e per fare il debugging di un programma

Visual Studio: Error List window

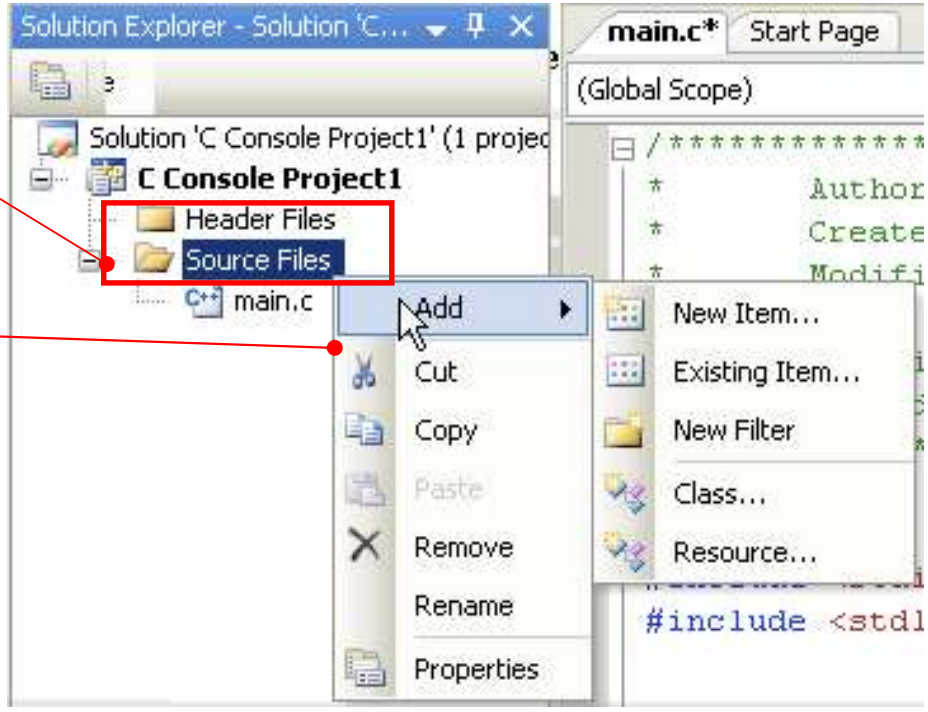
- Contiene la ***lista degli errori e dei warning*** rilevati dal compilatore
- Gli **errori** sono situazioni gravi, che rendono impossibile compilare il programma
- I **warning** sono situazioni in cui qualcosa di strano è stato rilevato dal compilatore, che però riesce a compilare comunque… ma spesso sono sintomi di errori non trascurabili

- **Un programma ben fatto, al momento della compilazione:**
  - **Non contiene errori**
  - **Non genera warning**

VS2010
Aggiungere files ad un progetto

1. Selezionare la cartella relativa a un file header o un file sorgente

2. Tasto dx ➔ Add ➔ New Item…
(oppure Menu File ➔New)

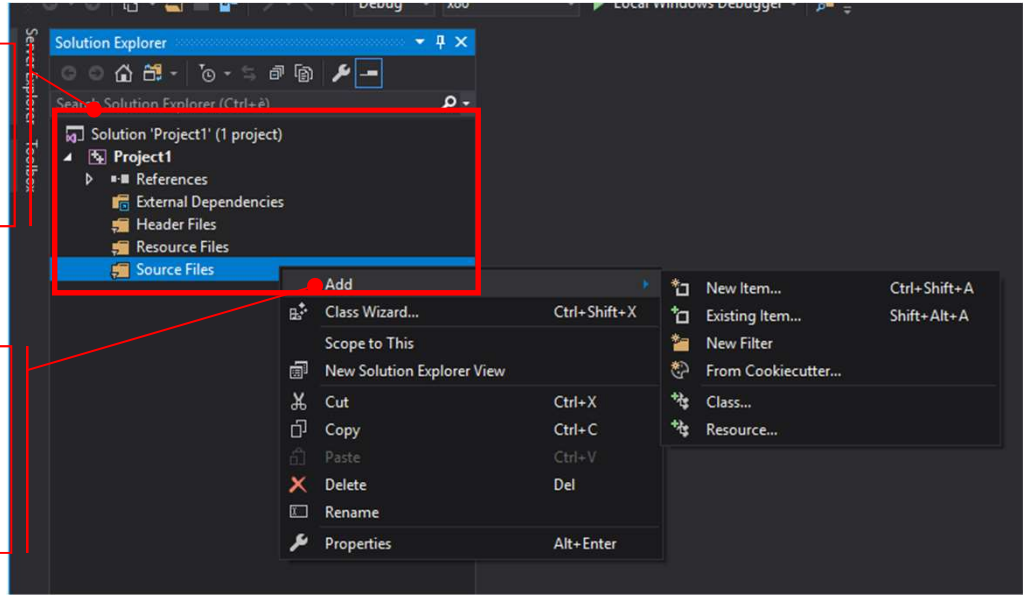


3. Si apre la finestra di creazione file
   - Selezionare Visual C++ ➔ Code
   - Poi scegliere la creazione di un header file (.h) o di un file sorgente (.cpp)
     - NOTA: nel secondo caso, specificare esplicitamente oltre al nome anche l'estensione .c!

VS2017
Aggiungere files ad un progetto

1.
Selezionare la cartella relativa a un file header o un file sorgente

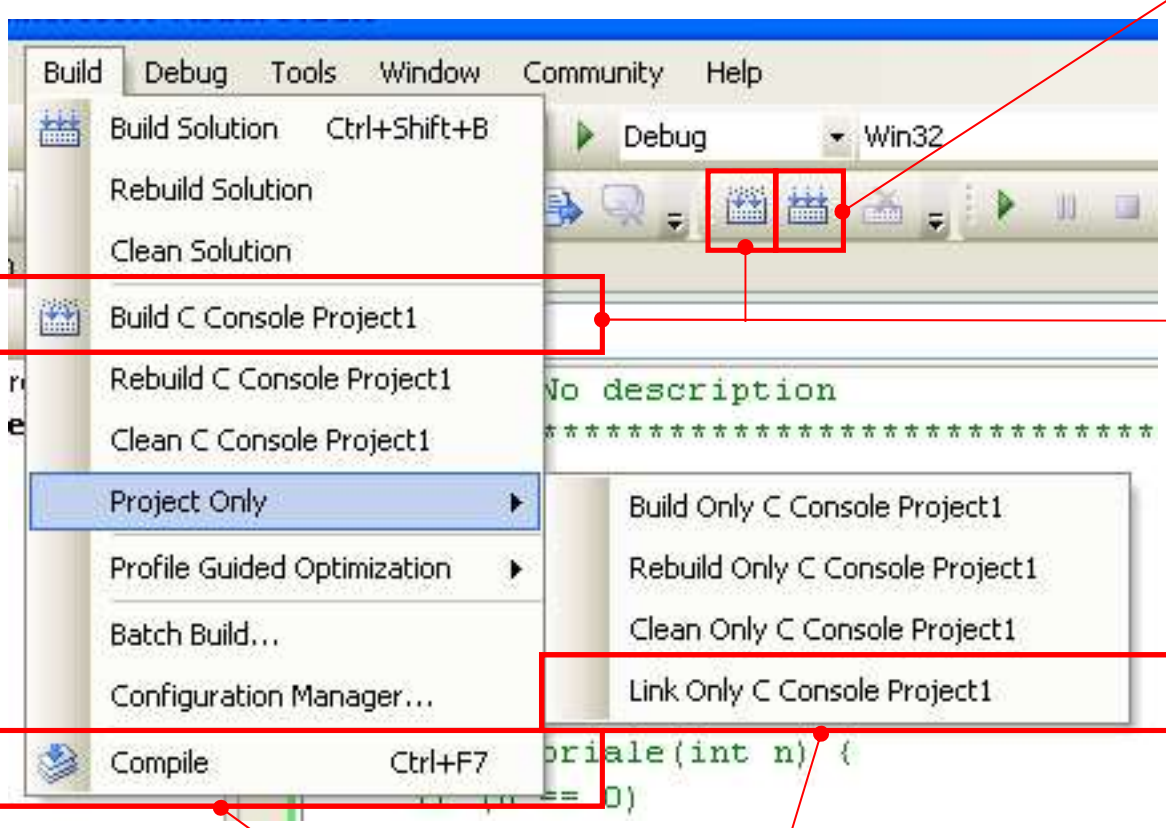
2.
Tasto dx ➔ Add ➔ New Item…
(oppure Menu File ➔New)



3. Si apre la finestra di creazione file
   - Selezionare Visual C++ ➔ Code
   - Poi scegliere la creazione di un header file (.h) o di un file sorgente (.cpp)
     - NOTA: nel secondo caso, specificare esplicitamente oltre al nome anche l'estensione .c!

Compilazione/Linking

- Menu **Build**



Build All
= compilazione
+ linking di tutti progetti

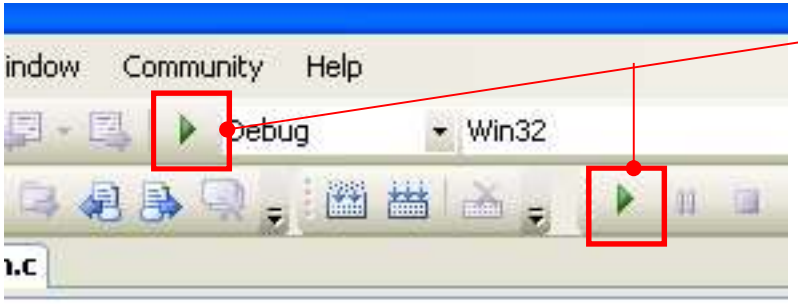
build
= compilazione
+linking

compilazione

linking

Per verificare i warning…
selezionare “Rebuild Project”

Esecuzione di un programma



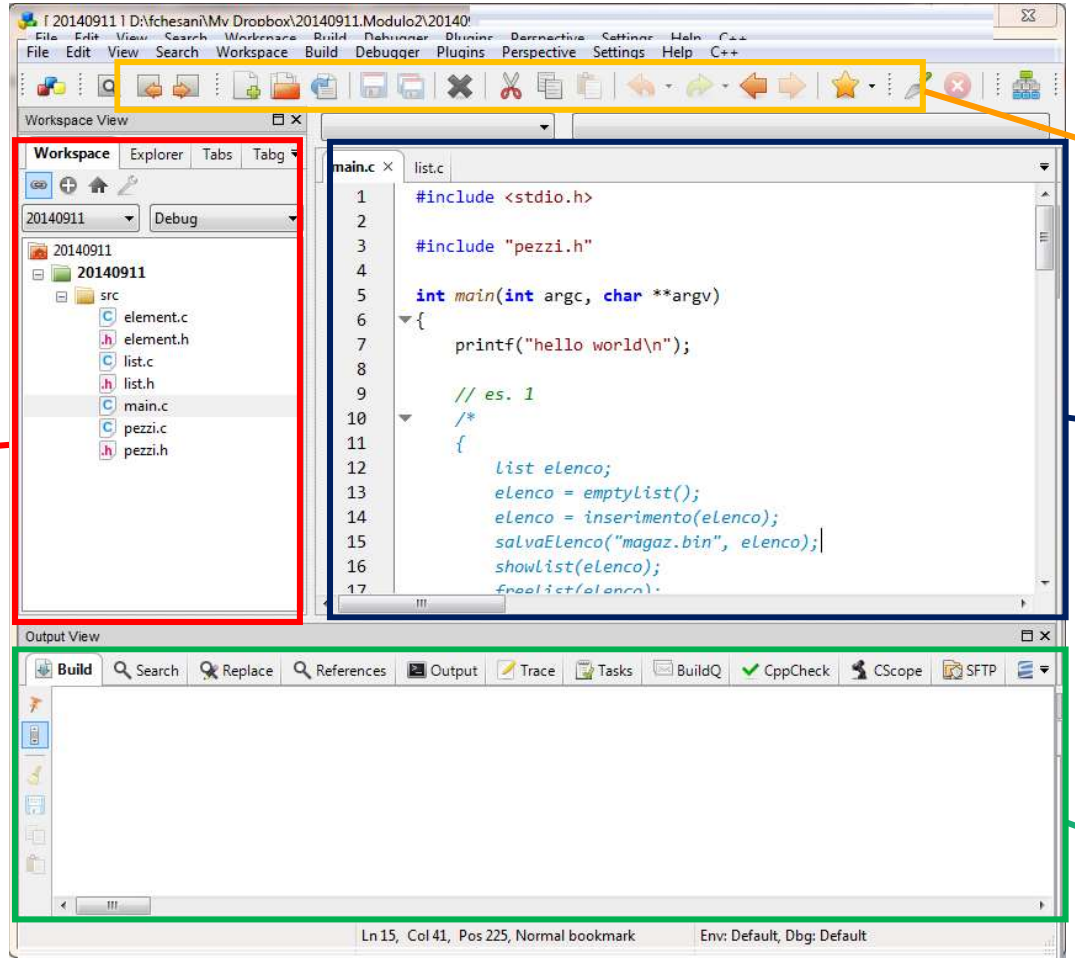
Esegue un programma in modalità "Debug"

Si ferma solo in presenza di un breakpoint

Come possiamo capire se un programma è corretto?

- Innanzitutto deve poter essere compilato senza errori e senza messaggi di warning…
- Prevedere l'output di un programma, e controllare eseguendo che tale output corrisponda

CodeLite: schermata principale



CodeLite: schermata principale

- **WorkspaceView**

  Mostra l'elenco dei progetti e dei file appartenenti ad ogni progetto. Per aprire un file, basta fare "doppio click" su di esso…

- **Source files**

  Mostra i file aperti, ogni file in un tab separato

- **Additional Windows**

  Mostrano alcune finstre ausiliarie molto importanti, quali:

  – "Build": mostra i messaggi di errore o di successo forniti dal framework durante la compilazione

  – "Output": mostra i messaggi di output quando viene eseguito un programma

- **Compile e Debug Palette**

  Contengono i pulsanti per compilare e per fare il debugging di un programma

CodeLite: error list nella finestra Output

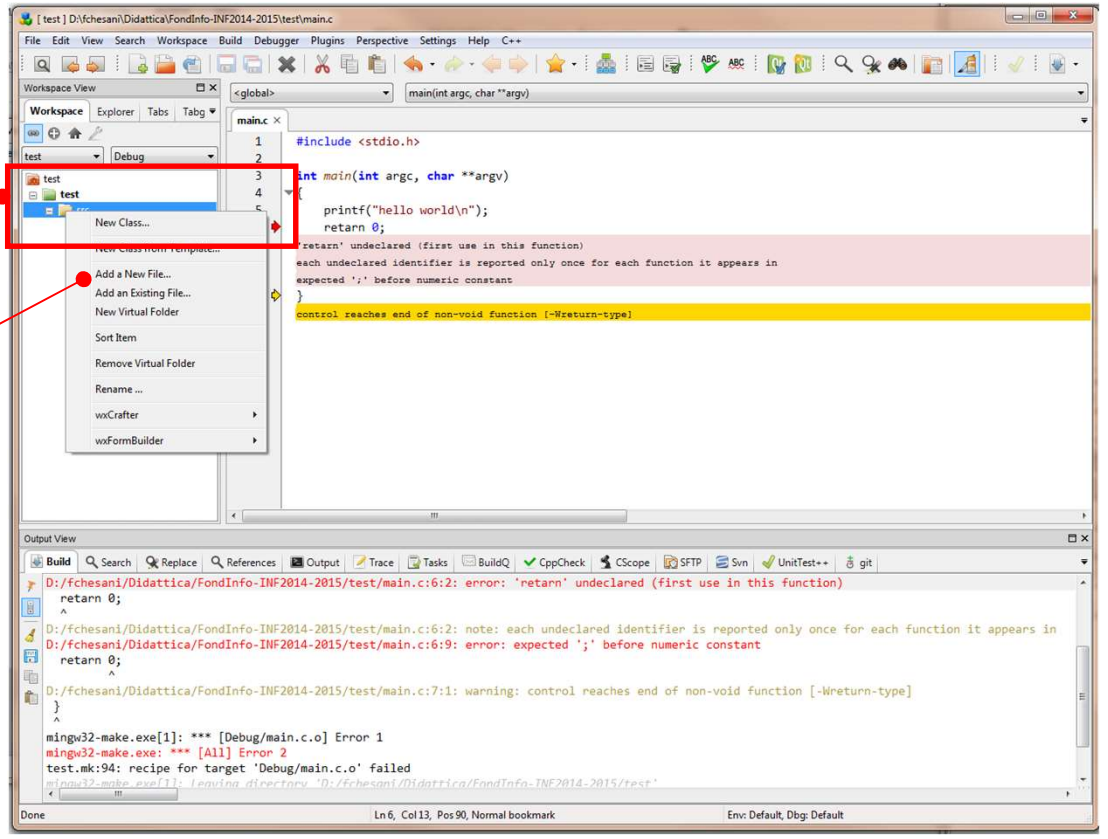
- Contiene la ***lista degli errori e dei warning*** rilevati dal compilatore
- Gli **errori** sono situazioni gravi, che rendono impossibile compilare il programma
- I **warning** sono situazioni in cui qualcosa di strano è stato rilevato dal compilatore, che però riesce a compilare comunque… ma spesso sono sintomi di errori non trascurabili

- **Un programma ben fatto, al momento della compilazione:**
  - **Non contiene errori**
  - **Non genera warning**

CodeLite: Aggiungere file ad un progetto

1. Selezionare la cartella relativa a un file header o un file sorgente

2. Tasto dx ➔ Add a New File



3. Si apre la finestra di creazione file
   - Selezionare "C Source File" o "Header File"

CodeLite: Compilazione/Linking ed esecuzione

- Usando i menu apposite
- Usando le toolbar
- ... tutto analogo a quanto già visto per Visual Studio

Debug di un programma

- Col termine “Debug” si intende una fase di sviluppo del software, nella quale si cerca di eliminare gli errori dal programma
- Due tipi di errori:
  - **Errori sintattici**, rilevati sempre dal compilatore in fase di compilazione
  - **Errori semantici**, difficilmente rilevabili

    Esempio: un programma deve eseguire la somma di due numeri, ma il programmatore in un momento di distrazione ha usato il simbolo di operazione “-” invece del simbolo “+”

    CONSEGUENZE: il programma è sintatticamente corretto, ma non esegue ciò che è stato richiesto

Debug di un programma

- Il programmatore deve essere in grado, per ogni istruzione del proprio programma, di prevedere che cosa farà tale istruzione, cioè
- **Il programmatore deve conoscere in anticipo gli effetti derivanti dall'eseguire una certa istruzione**

IDEA: per ogni istruzione del programma:

a) Calcolo quali siano gli effetti nell'eseguire l'istruzione

b) Eseguo tale istruzione

c) Verifico che gli effetti siano effettivamente ciò che mi aspettavo

Se la verifica fallisce, ho trovato un errore ☺!

Uso del debug

L'ambiente di sviluppo ci mette a disposizione una serie di funzionalità per:

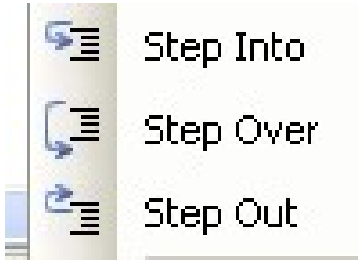
- Eseguire passo passo ogni istruzione
- Controllare lo "stato" del nostro programma
  - Visualizzare il contenuto delle variabili (monitoraggio)
  - Visualizzare lo ***stack delle chiamate a funzione***
    (imparerete presto che cosa significa…)
  - …

Lancio di un programma e debug

- Premere il pulsante con l'icona "play" per lanciare il programma
- Il programma viene lanciato in modalità debug (a indicare che è ancora sotto test)
  - Da un punto di vista dell'esecuzione non cambia niente…
  - …ma vi dà la possibilità di andare a controllare il vostro codice istruzione per istruzione

Passi di debug

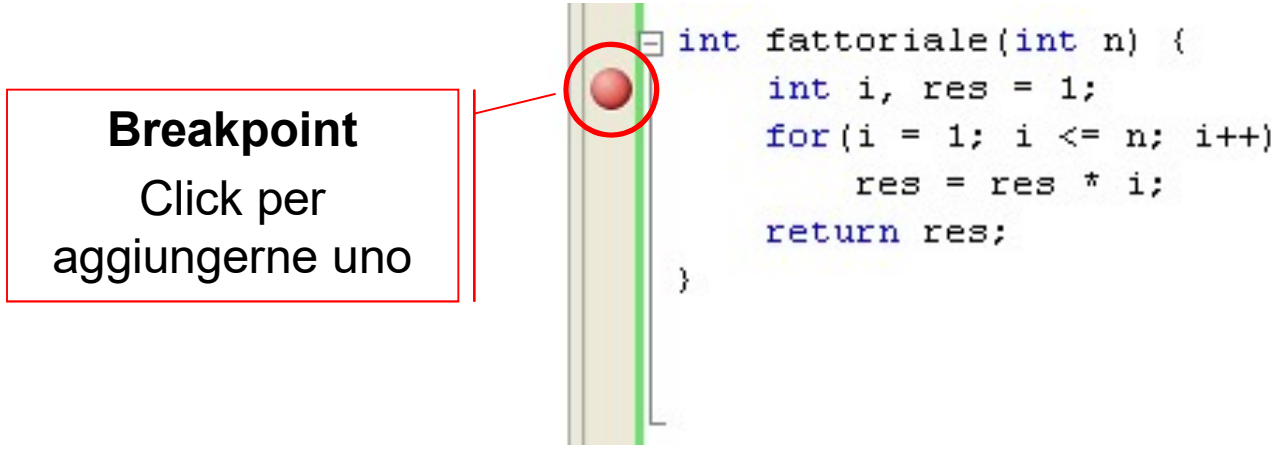
- È possibile controllare l'esecuzione istruzione per istruzione (usando uno dei tasti "step …")
  - Basta premere uno di essi per lanciare il programma passo passo (al posto di "play")



- E se l'istruzione chiama una funzione?
  - Step Into ➔ continua il debug entrando nel codice della funzione
  - Step Over ➔ continua il debug ripartendo dal punto immediatamente successivo alla chiamata di funzione (ovvero esattamente dopo la restituzione del valore)
- Se sono all'interno di una funzione, con Step Out posso continuare il debug all'istruzione che segue **`return`** della funzione

Breakpoints (1)

- Cominciare il debug dall'inizio del programma può essere scomodo…
- Possiamo inserire dei breakpoint
  - Punti del programma che ci interessa monitorare
    - Il programma esegue normalmente fino al breakpoint, poi passa in "modalità debug"



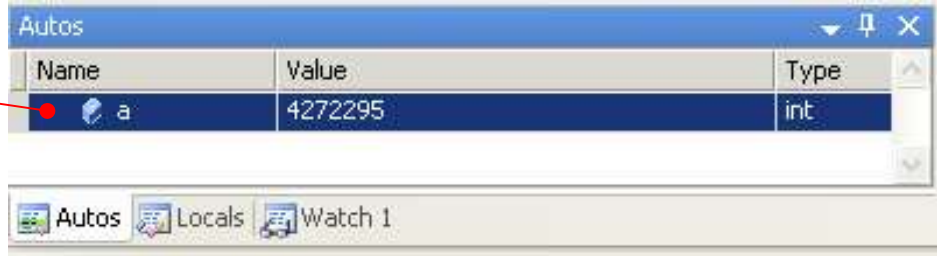
Breakpoints (2)

- Una volta bloccata l'esecuzione al raggiungimento di un breakpoint si può decidere
  - Di continuare l'esecuzione normalmente fino al prossimo breakpoint (pulsante "play")
  - Di continuare il debug istruzione per istruzione (con uno dei vari "step …")
- Nota: i breakpoint possono essere associati a condizioni e altre proprietà configurabili (es: si può indicare di attivare il breakpoint solo se una certa variabile è uguale a 0)
  - Menu Debug ➔ Windows ➔ Breakpoints apre la finestra di configurazione e definizione dei vari breakpoint

Monitoraggio variabili

- Tre finestre di monitoraggio delle variabili
  - Auto
    - Visualizza il contenuto delle variabili definite ***all'interno dello scope corrente*** (e anche il valore di ritorno all'uscita da una funzione)

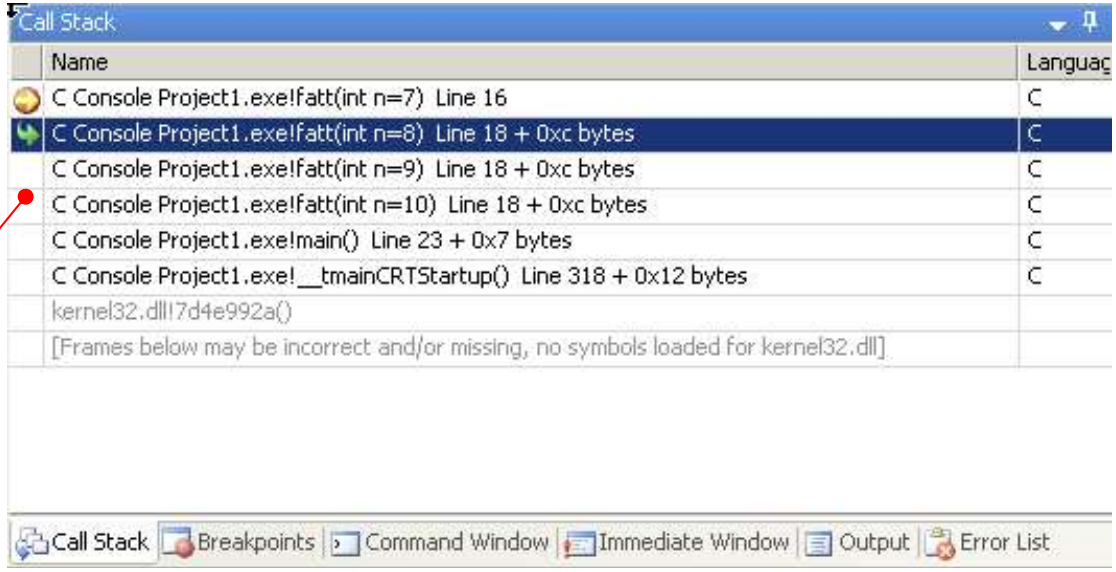
Es: valore della variabile 'a' prima dell'inizializzazione



  - Local
    - Visualizza il contenuto delle ***variabili "locali", ovvero tutte quelle visibili all'interno della funzione corrente*** (nota: in caso di scope innestati con variabili con lo stesso nome, compaiono ripetizioni)
  - Watch
    - Permette di inserire il ***nome della variabile da monitorare*** (attenzione agli scope)
    - È possibile anche monitorare espressioni (es: a+b)

Finestra Call Stack

- Permette di visualizzare lo stack delle chiamate a funzione
  - Alla chiamata di una funzione viene aggiunta una riga che mostra il ***valore dei parametri attuali***
  - All'uscita di una funzione rimozione della riga (in cima)
  - È possibile selezionare una qualsiasi delle righe, e le finestre di monitoraggio delle variabili recuperano lo stato corrispondente
- Provare con funzioni ricorsive!



Call stack del fattoriale ricorsivo…

Esercizio

- Creare un nuovo progetto per il linguaggio C (a tal scopo, utilizzare il progetto vuoto disponibile sul sito del corso)
- Nel file sorgente main.c, scrivere il seguente codice:

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

int main() {
   int a;
   a = 2+3;
   printf("Hello world!");
   a = a-3;
   return 0;
}
```

- Compilare il programma
- Eseguire il programma
- Settare un break point all'istruzione "a= 2+3;"
- Rieseguire il programma utilizzando il debug, e verificare che cosa succede ad ogni istruzione…

Appendice A:
Che cosa fare se…
(MS Visual Studio)

Cosa fare se …

- **… non compare il Solution Explorer**
  1. Menù “View”
  2. Selezionare la voce “Solution Explorer”
- **… non compare la finestra “Output” in basso**
  1. Menù “View”
  2. Selezionare la voce “Output”
- **… non compare la finestra “Error List” in basso**
  1. Menù “View”
  2. Selezionare la voce “Error List”

Cosa fare se …

- **… non compare la “Build Palette”**
  1. Cliccare con il tasto di destra del mouse un punto qualunque sulla barra dei bottoni o dei menu
  2. Selezionare la voce “Build”
- **… non compare la “Debug Palette”**
  1. Cliccare con il tasto di destra del mouse un punto qualunque sulla barra dei bottoni o dei menu
  2. Selezionare la voce “Debug”

Appendice B:
Creare un progetto per il C (MS VIsual Studio)

C++ vs. C

- Visual Studio supporta il C++
- Visual Studio supporta *in modo non diretto* anche il C…

Per creare un nuovo progetto per programmare in C, si crea un nuovo progetto C++

Attenzione: il linguaggio C++ consente alcune "scorciatoie" di sintassi che PERO' in C sono **VIETATE**!!!

Creare un nuovo progetto C++ Visual Studio 2010

1.a) “New project” button

1.b) Seleziono “Visual C++”

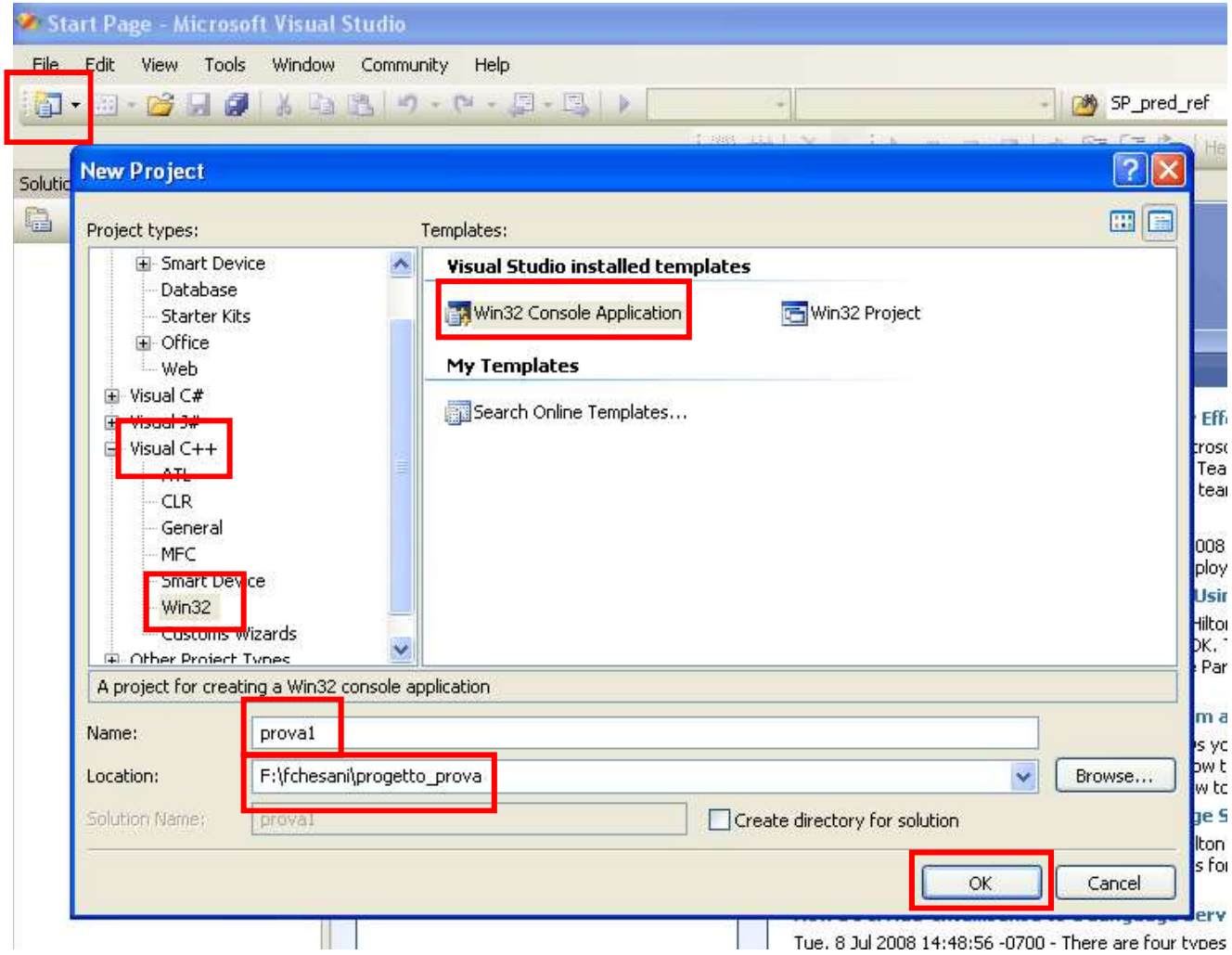
1.c) Seleziono la categoria “Win32”

1.d) Come tipo di progetto, seleziono “Win32 Console Application”

1.e) Specifico il nome di progetto

1.f) Specifico dove voglio salvare il progetto

1.g) Clicco “OK”



Creare un nuovo progetto C++ Visual Studio 2017

1.a) “New project” button

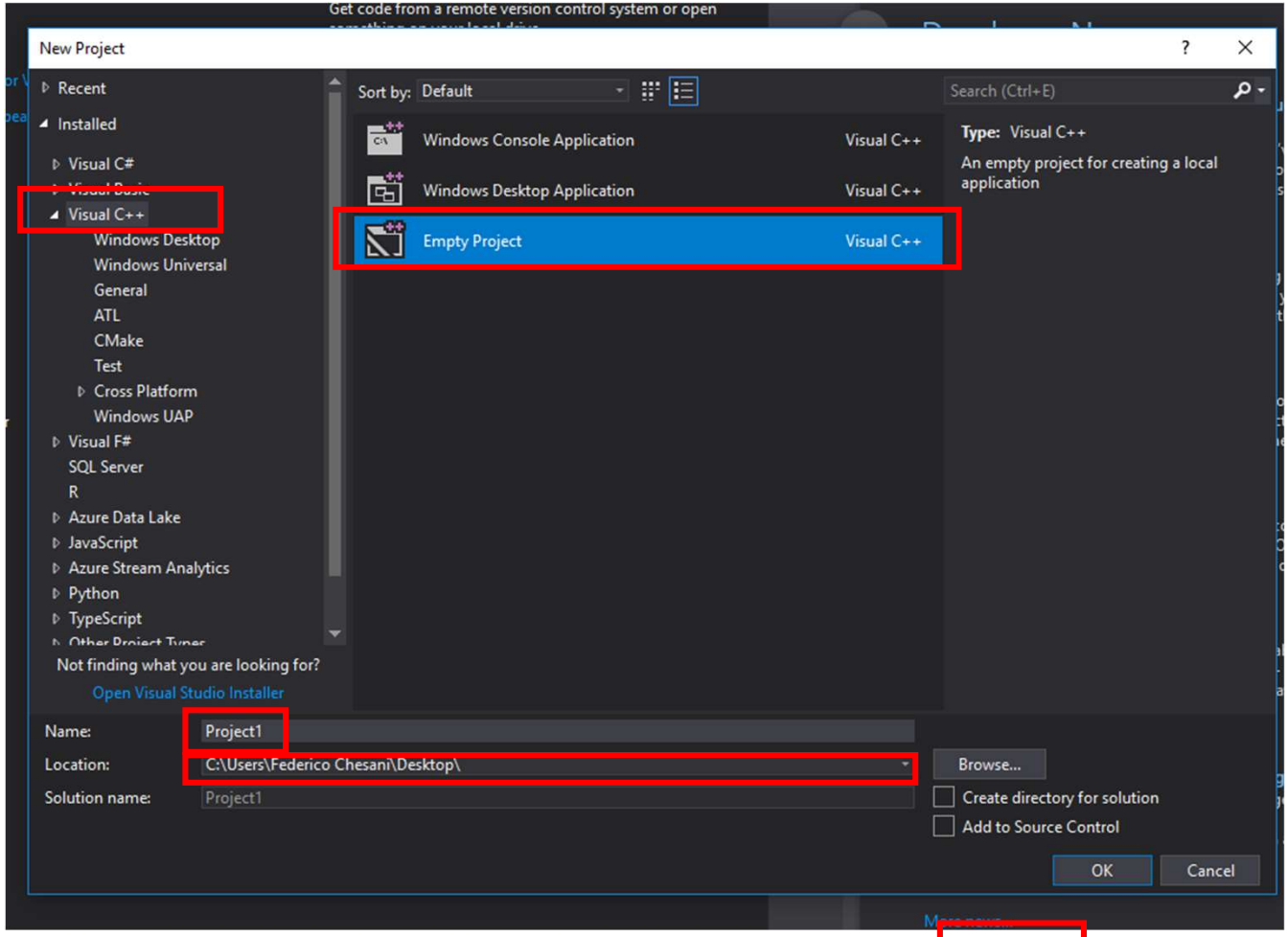
1.b) Seleziono “Visual C++”

1.c) Come tipo di progetto, seleziono “Empty project”

1.d) Specifico il nome di progetto

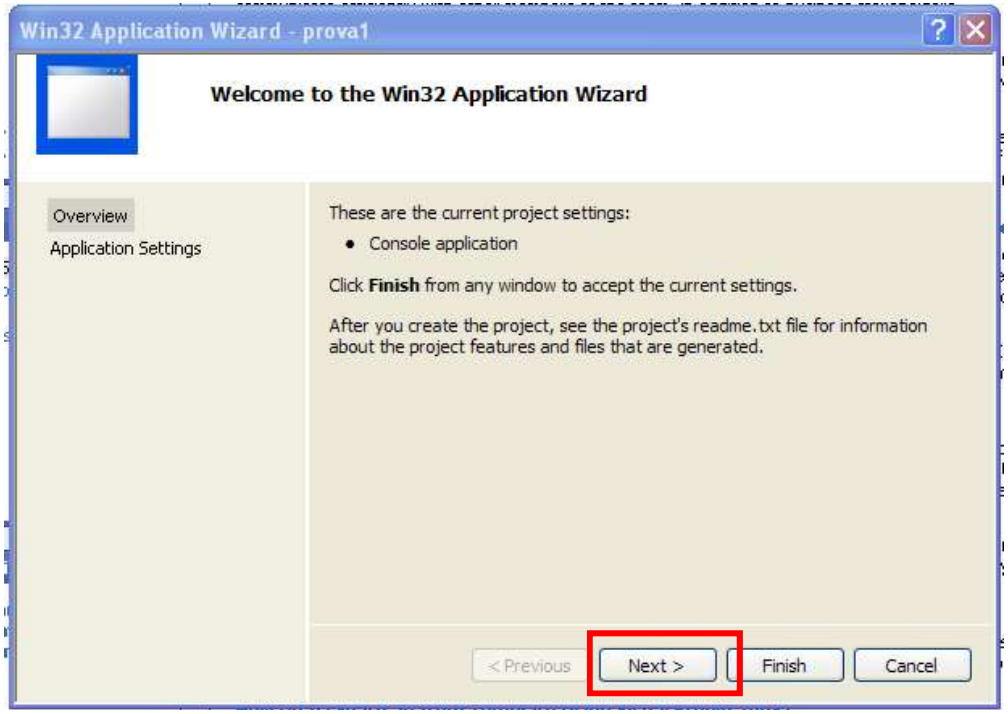
1.e) Specifico dove voglio salvare il progetto

1.f) Clicco “OK”



Creare un nuovo progetto C++ Visual Studio 2010

1.h) Configuro correttamente il progetto… in questa schermata mi limito a cliccare “Next >”

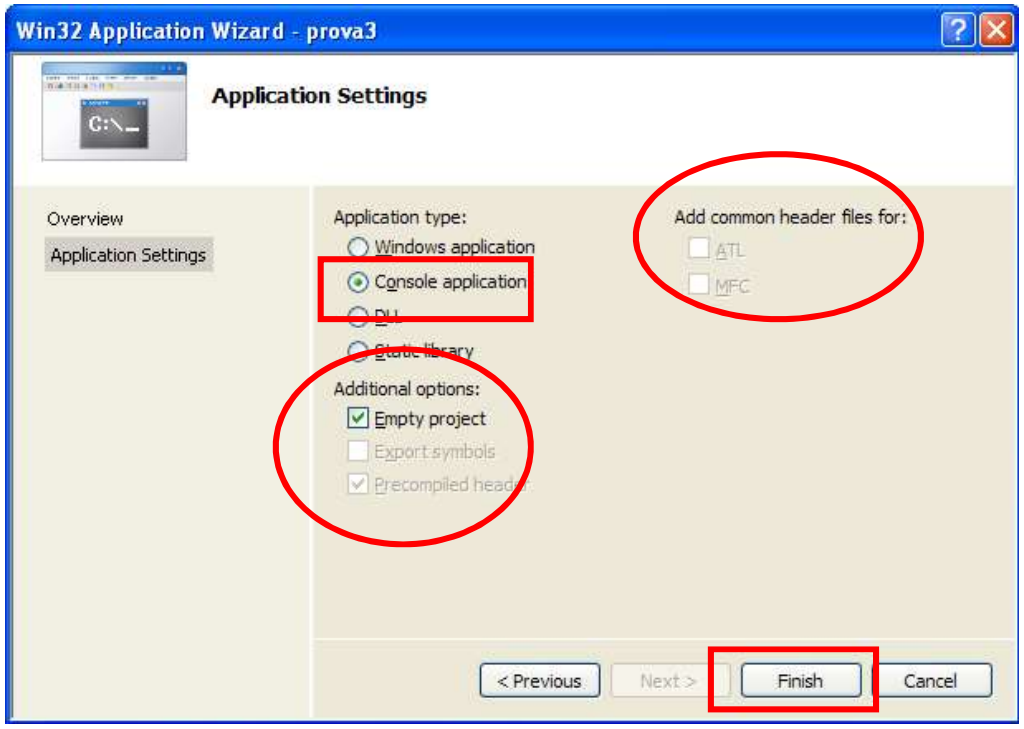


Creare un nuovo progetto C++ Visual Studio 2010

1.i) Specifico come Application type ancora “Console application”

1.j) Seleziono come Additional options la voce “Empty Project”

1.k) Clicco su “Finish”



Creare un nuovo progetto C++ Visual Studio 2017

Se si seleziona un empty project, non c'è alcun wizard....

Appendice C:
Warning su funzioni di libreria "deprecated"

Warning su funzioni di libreria "deprecated"

- Aggiungere in testa ad ogni file, come primissime istruzioni per il pre-compilatore:
- `#define _CRT_SECURE_NO_DEPRECATE`

Volendo, potete aggiungere anche (in alternativa):

- `#define _CRT_SECURE_NO_WARNINGS`