Esercitazioni di Programmazione I
canale A-D

Alessio Malizia

malizia@di.uniroma1.it

I docenti

- Docente: T. Calamoneri

  calamo@di.uniroma1.it

  Ricevimento: lun. 14.00-15.30

  (mandando prima una e-mail)

- Esercitatore: A. Malizia

  malizia@di.uniroma1.it

  Ricevimento: gio. 14.00-16.00

  (mandando prima una e-mail)

Alcune regole di questo corso

- Non c'è obbligo di frequenza, quindi chi non è interessato non deve rimanere
- Necessario seguire le regole di comportamento (non si mangia, non si ride, non si gioca, non si arriva in ritardo e non si va via in anticipo…)
- Laboratori: scopo ed orari
- Esercizi per casa obbligatori (correz. automatica tutti + manuale alcuni) - soluzioni durante le esercitazioni
- Tutoraggio
- Orario di ricevimento
- Pagina web (twiki.di.uniroma1.it)
- Libri di testo…

Modalità d'Esame

- Esercizi settimanali obbligatori (non entrano nella valutazione, ma senza di essi non si può fare l'esame).

  Controllo automatico della correttezza e dell'"originalità"
- Un esame scritto, che può essere sostituito da due prove di esonero (fortemente raccomandate, soprattutto agli studenti del primo anno)
- Un esame orale

Per superare l'esame è sufficiente studiare settimanalmente e fare tutti gli esercizi proposti (non solo quelli da consegnare!)

Ambiente di programmazione p.13

L'ambiente di programmazione è un insieme di strumenti che facilitano la scrittura dei programmi e la verifica della loro correttezza

- Editor:
  per costruire programmi sorgente, cioè scritti in un linguaggio di alto livello
- Preprocessore:
  per inglobare le direttive, che consistono di solito nell'inclusione di altri files e nella sostituzione di simboli speciali con un testo
- Compilatore:
  per tradurre in un programma oggetto, scritto in un linguaggio eseguibile. Possibili errori… A volte: Interprete
- Linker:
  per collegare i vari moduli del programma, separatamente compilati
- Loader:
  per caricare il programma in memoria
- Debugger:
  per controllare l'esecuzione ed eliminare errori

Storia del C p.8

- C
  - Creato da Ritchie a partire da due precedenti linguaggi di programmazione, BCPL e B
  - Usato per sviluppare UNIX ed altri sistemi operativi
  - Dalla fine degli anni 70 il C si è evoluto nel "Tra ditional C"
- Standardizzazione
  - Molte piccole variazioni del C, molte delle quali tra loro incompatibili
  - Comitato formato per creare una definizione ”non ambigua, e indipendente dalla macchina”
  - Standard creato nel 1989, aggiornato nel 1999

Librerie p.9

I programmi C consistono di moduli detti funzioni.

- Un programmatore può creare le funzioni di cui ha bisogno
  - Vanataggio: il programmatore sa esattamente come funzionano
  - Svantaggio: dispendio di tempo
- I programmatori spesso usano funzioni di libreria
- Per evitare di “reinventare la ruota”:
  - se una funzione esiste già, è di solito meglio usarla piuttosto che riscriverla
  - le funzioni di libreria sono scritte attentamente, guardando all’efficienza e alla portabilità

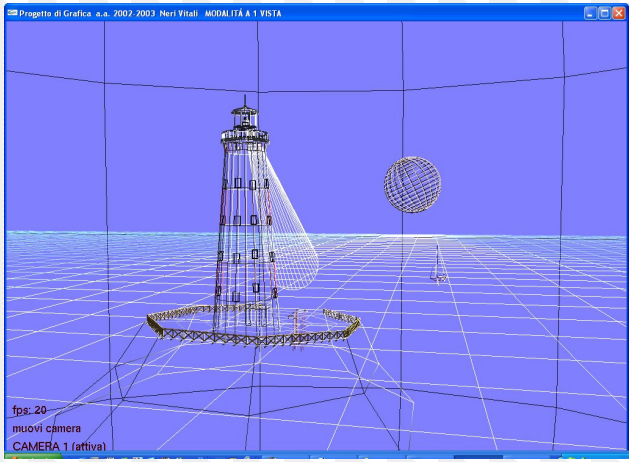
Il linguaggio C

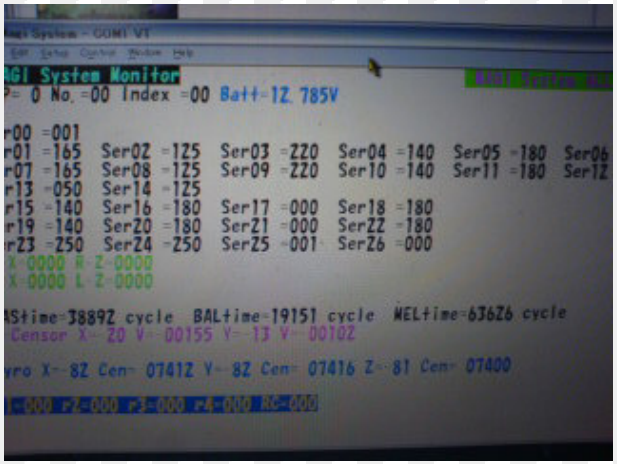
- Variabili (celle di memoria) tramite identificatori simbolici
- Istruzioni di assegnamento (espressioni e operatori aritmetici)
- Istruzioni di input e output
- Istruzioni condizionali (condizioni)
- Istruzioni iterative

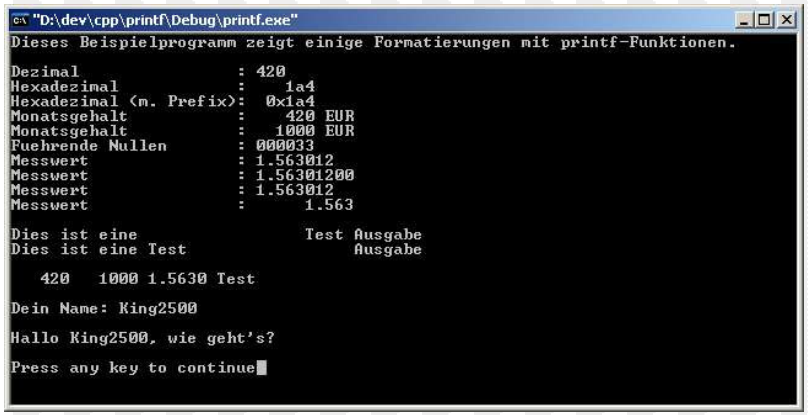
Note sul C

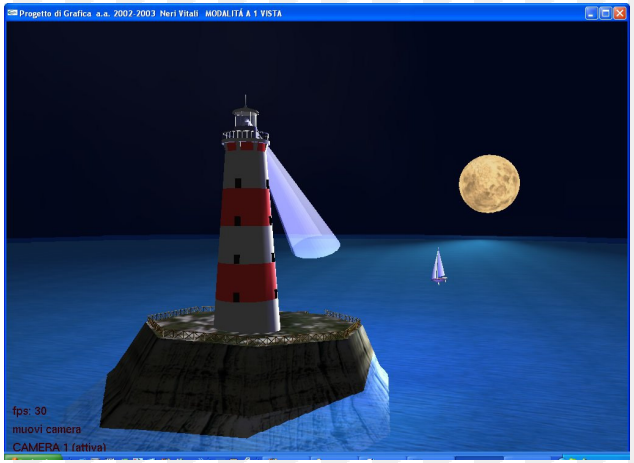
- Chiarezza
  - Il C consente di scrivere programmi molto compatti, tuttavia essi sono spesso difficili da leggere, da capire e da modificare
- il C è un linguaggio portabile
  - i programmi C possono spesso essere eseguiti su diversi computers e piattaforme, ma…
- In questo corso studieremo il C ma
  - alcuni dettagli non saranno spiegati
  - per avere ulteriori dettagli tecnici si può:
    - leggere la documentazione standard del C
    - leggere il libro di Kernigan and Ritchie

Applicazioni









Compilatori (1)

- La GNU Compiler Collection (GCC). Software libero!
- Cygwin: un ambiente UNIX per windows. Comprende tutti i compilatori della *suite* GCC (C, C++, Java, Ada, Fortran, ...). Software libero!
- DEV C++ builder
- Il compilatore di "C++ Builder" (gratuito, ma non libero).

Compilatori (2)

- http://gcc.gnu.org/
- http://www.fsf.org/philosophy/free-sw.it.html
- http://sources.redhat.com/cygwin/
- http://www.borland.com/bcppbuilder/freecompiler/
- http://www.bloodshed.net/dev/ (DEV C++)

C e Compilazione

- Obiettivi :
- Conoscere le caratteristiche di un linguaggio di programmazione
- Comprendere il significato della compilazione, del codice sorgente, del codice oggetto, di un programma eseguibile

Linguaggi di programmazione

- **Un programma è una sequenza di istruzioni scritte in un linguaggio di programmazione.**
- **Esistono centinaia di linguaggi di programmazione:**
  - C; C++; Java; Fortran; Assemby; Pascal; Cobol;
  - Ada; VB; PERL; C#; Matlab; Modula2.
- **Vi sono linguaggi direttamente comprensibili dall'elaboratore.**
- **Altri necessitano di traduzioni, cioè di sequenze di operazioni che rendono il programma eseguibile dall'elaboratore.**

Categorie di linguaggi

- **I linguaggi di programmazione possono essere distinti in:**
  - linguaggi MACCHINA;
  - linguaggi ASSEMBLY;
  - linguaggi di ALTO LIVELLO.

Linguaggio macchina

- **Ogni processore può direttamente comprendere il suo linguaggio macchina.**
- **Possiamo intenderlo come il linguaggio naturale del processore.**
- **Dipende strettamente dal processore.**
- **E' di difficile leggibilità.**
- **Esempio**
  - 1010….10101
  - 1111….10101
  - 1011….10101
  - 1111….10101

Linguaggio assembly (1)

- **E' la traduzione del significato delle istruzioni del linguaggio macchina in un formato più leggibile.**
- **Esempio :** z=x+y;
- Linguaggio Macchina
  - 1010….10101
  - 1111….10101
  - 1011….10101
  - 1111….10101
- Assemby
  - LOAD X
  - LOAD Y
  - ADD
  - STORE Z

Linguaggio assembly (2)

- **Ha semplificato molto il compito dei primi programmatori.**
- **Un semplice programma assembly è composto da moltissime istruzioni.**
- **Anche semplici operazioni fra variabili, come z=x+y, implicano la scrittura di più istruzioni.**

Linguaggio di alto livello

- **Permette di esprimere operazioni elementari con una sola istruzione (es.: funzioni matematiche).**
- **Permette di scrivere programmi facilmente comprensibili dal programmatore.**
- **E' un linguaggio molto vicino al linguaggio naturale del programmatore (inglese).**
- **Es:** if done then print_results()

Compilazione

PROGRAMMATORE

Programmi comprensibili dal programmatore
(ALTO LIVELLO)

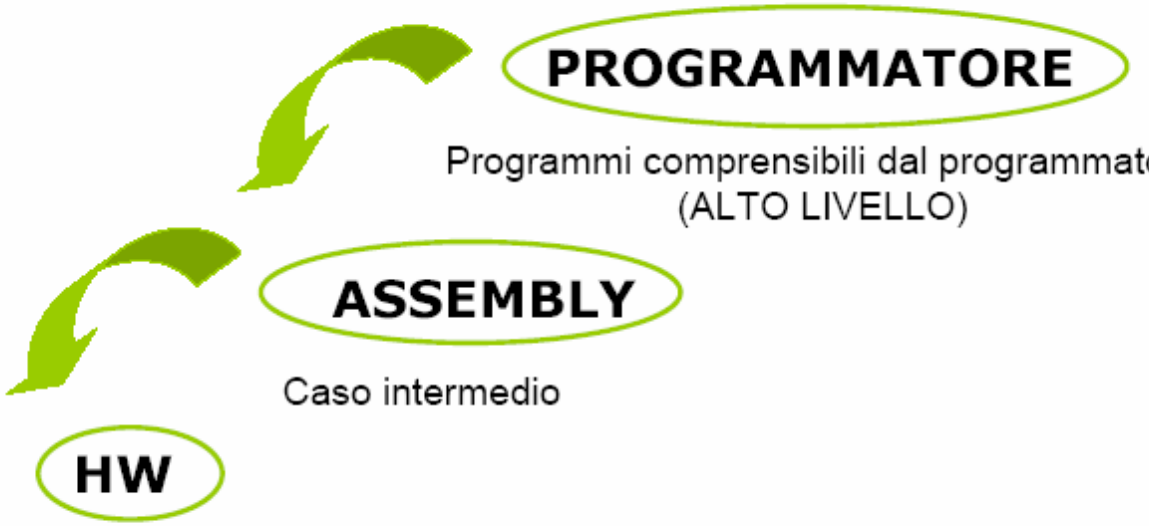
ASSEMBLY

Caso intermedio

HW

Programmi eseguibili direttamente dal processore
(LINGUAGGI MACCHINA)

Compilazione (2)



Compilazione: processo di traduzione che porta il programma ad essere eseguibile.

Quali Linguaggi?

- *Oggetto di questo corso sono i linguaggi di programmazione di ALTO LIVELLO.*
- *Linguaggio C*

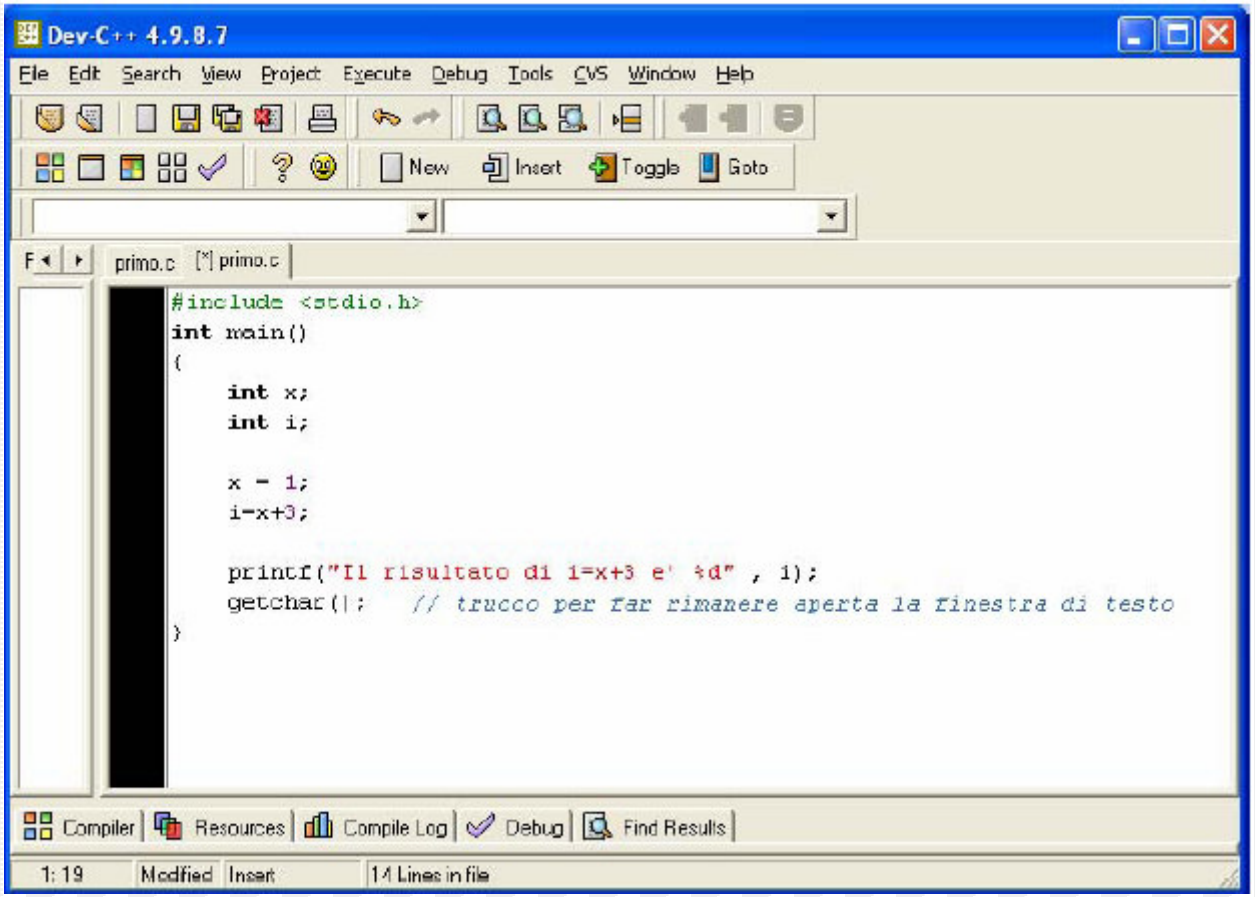
Perché il linguaggio C in questo corso?

- **Linguaggio di ALTO LIVELLO.**
- **Permette di studiare aspetti vicini all'HW.**

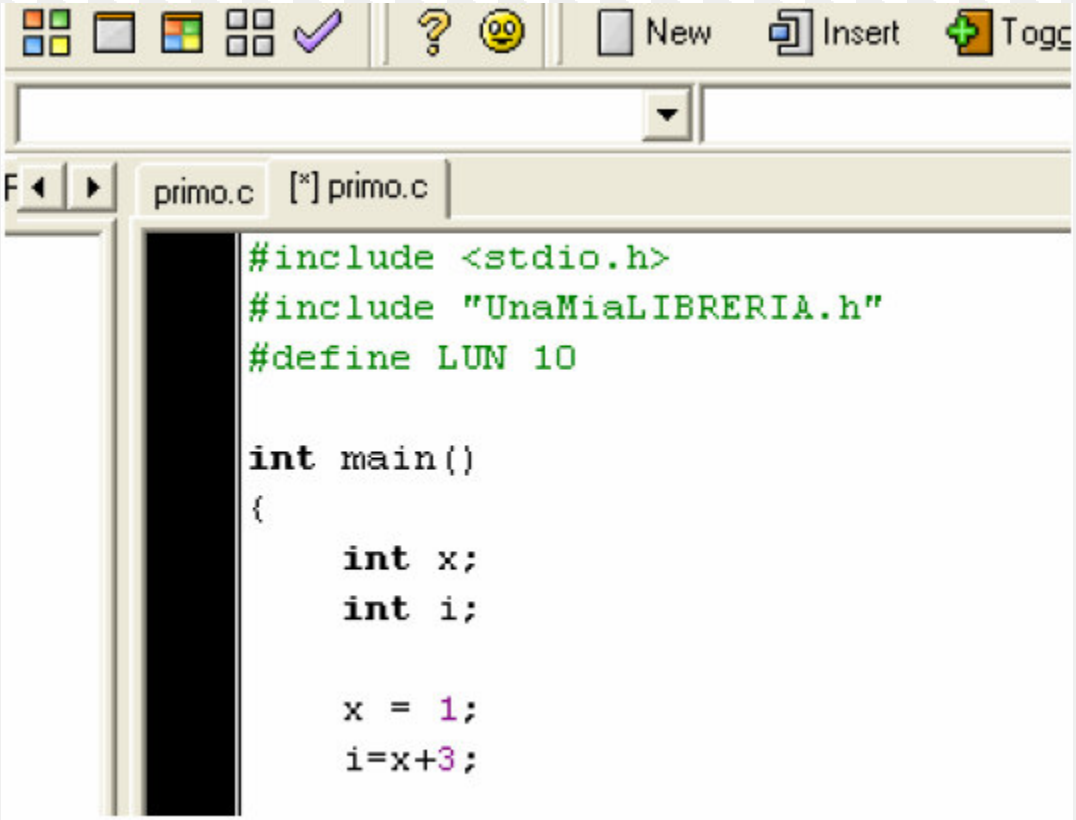
Rendere eseguibile un programma

- **Per rendere eseguibile un programma C su un calcolatore servono 6 passi:**
  - 1. editazione;
  - 2. pre-elaborazione;
  - 3. compilazione;
  - 4. collegamento;
  - 5. caricamento;
  - 6. esecuzione.

Editazione


Dev-C++ 4.9.8.7
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
primo.c [*] primo.c
#include <stdio.h>
int main()
{
int x;
int i;
x = 1;
i=x+3;
printf("Il risultato di i=x+3 e' %d" , i);
getchar(); // trucco per far rimanere aperta la finestra di testo
}
Compiler Resources Compile Log Debug Find Results
1: 19 Modified Insert 14 Lines in file

Pre-elaborazione



Compilazione (1)

- **Partendo da questo esempio di programma C :**

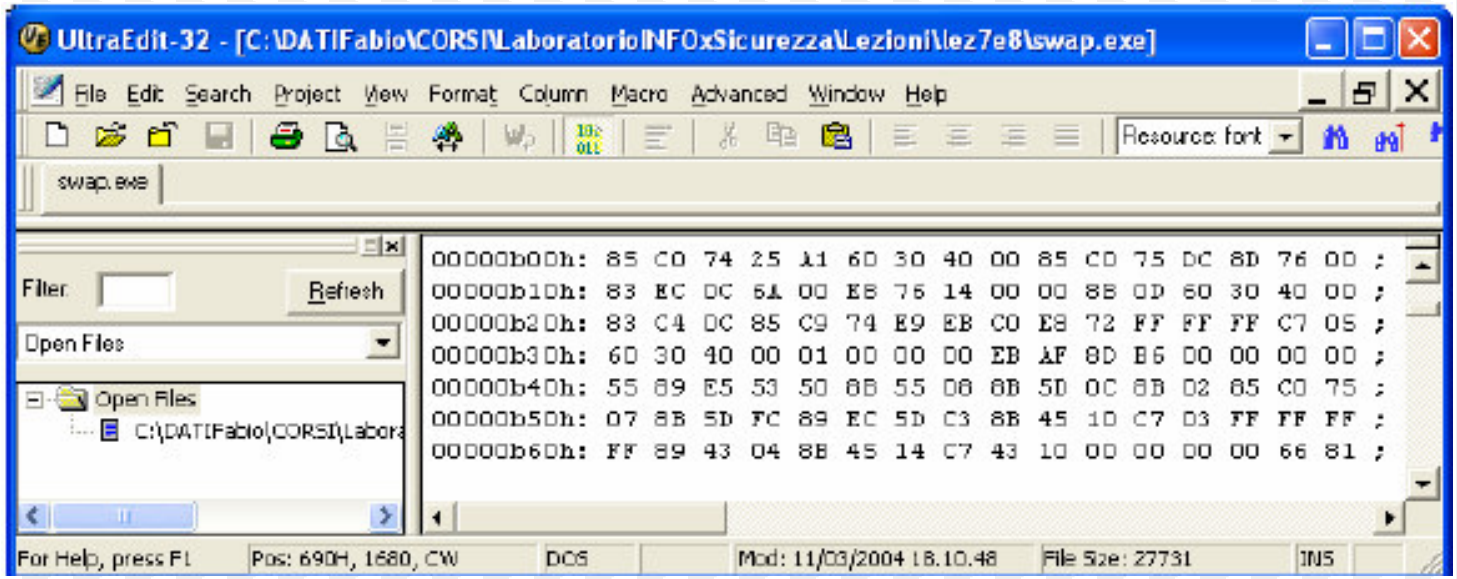
```c
#include <stdio.h>
int main()
{
    int x;
    int i;

    x = 1;
    i=x+3;

    printf("Il risultato di i=x+3 e' %d" , i);
    getchar();   // trucco per far rimanere aperta la finestra di testo
}
```

Compilazione (2)

- **L'istruzione i = x +3; diventa:**
  - mov eax, [x]
  - add eax, 3
  - mov [i], eax
- eax è un registro del processore;
- [x] è il contenuto della cella di memoria di indirizzo x;
- (Microsoft assembler).

Compilazione (3)

- mov eax, [x]
- add eax, 3
- mov [i], eax

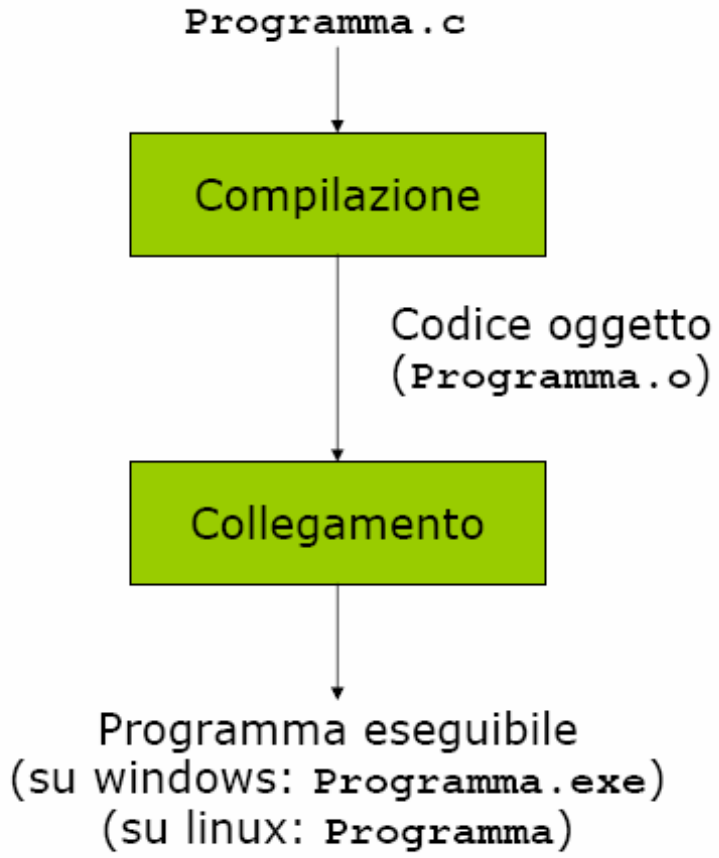


Collegamento (linking)(1)

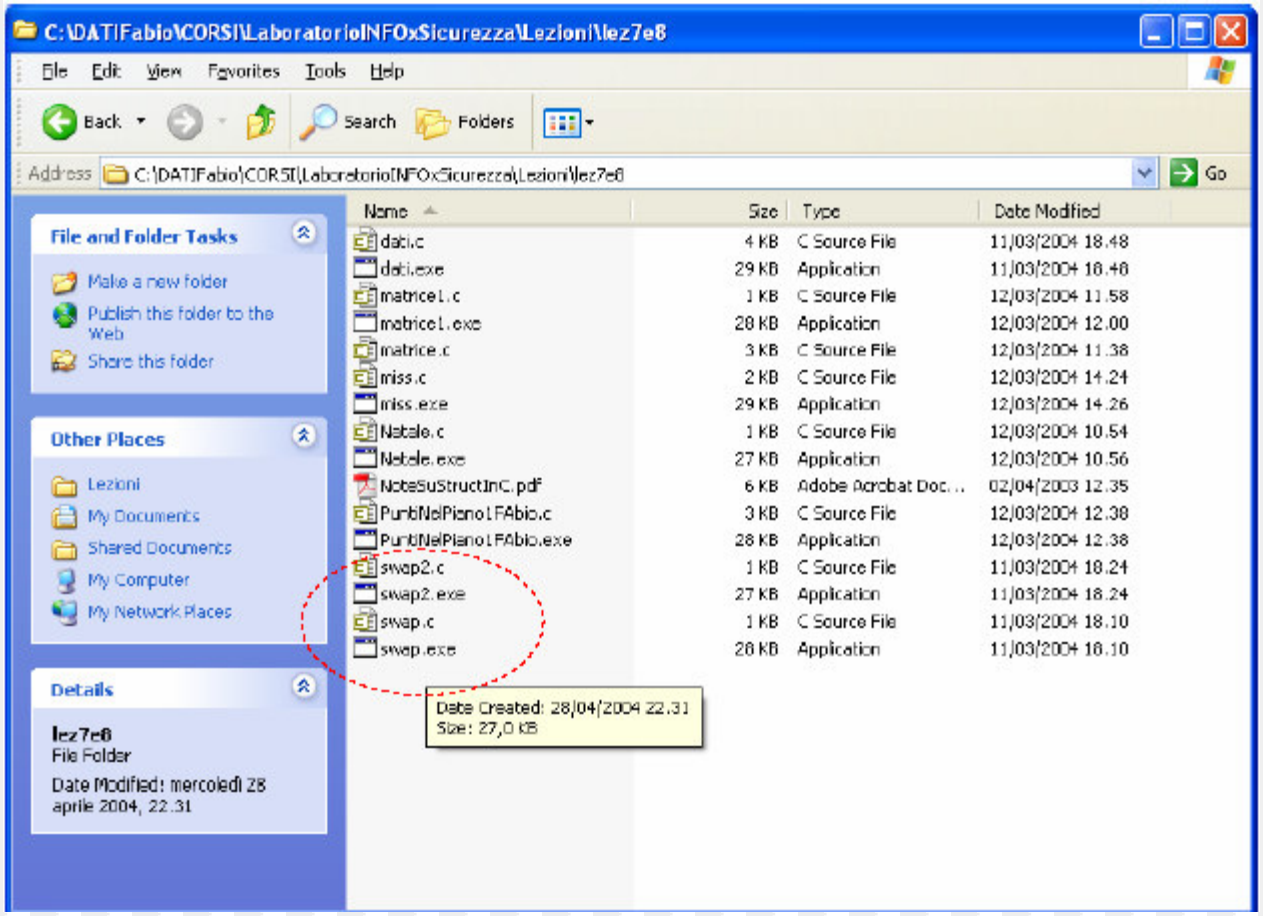
- **Viene incluso il codice delle funzioni presenti nelle librerie:**

```
#include <stdio.h>
int main()
{
    int x;
    int i;

    x = 1;
    i=x+3;

    printf("Il risultato di i=x+3 e' %d" , i);
    getchar();   // trucco per far rimanere aperta la finestra di testo
}
```

Collegamento (linking) (2)



Esecuzione


C:\DATI\Fabio\CORSI\LaboratorioINFOxSicurezza\Lezioni\lez7e8
File Edit View Favorites Tools Help
Back Search Folders
Address C:\DATI\Fabio\CORSI\LaboratorioINFOxSicurezza\Lezioni\lez7e8
Go
File and Folder Tasks
Make a new folder
Publish this folder to the Web
Share this folder
Other Places
Lezioni
My Documents
Shared Documents
My Computer
My Network Places
Details
lez7e8
File Folder
Date Modified: mercoledì 28 aprile 2004, 22.31
Name Size Type Date Modified
dati.c 4 KB C Source File 11/03/2004 18.48
dati.exe 29 KB Application 11/03/2004 18.46
matrice1.c 1 KB C Source File 12/03/2004 11.58
matrice1.exe 28 KB Application 12/03/2004 12.00
matrice.c 3 KB C Source File 12/03/2004 11.38
miss.c 2 KB C Source File 12/03/2004 14.24
miss.exe 29 KB Application 12/03/2004 14.26
Natale.c 1 KB C Source File 12/03/2004 10.54
Natale.exe 27 KB Application 12/03/2004 10.56
NoteSuStructInC.pdf 6 KB Adobe Acrobat Doc... 02/04/2003 12.35
PuntiNelPiano1FAbio.c 3 KB C Source File 12/03/2004 12.38
PuntiNelPiano1FAbio.exe 28 KB Application 12/03/2004 12.38
swap2.c 1 KB C Source File 11/03/2004 18.24
swap2.exe 27 KB Application 11/03/2004 18.24
swap.c 1 KB C Source File 11/03/2004 18.10
swap.exe 28 KB Application 11/03/2004 18.10
Date Created: 28/04/2004 22.31
Size: 27,0 KB

Ambienti di sviluppo

- Obiettivi :
  - Essere in grado di compilare programmi negli ambienti di sviluppo:
    - DEVC++, gcc, .Net
  - Riuscire a trovare e comprendere la documentazione delle funzioni nelle librerie

Piattaforma DEVC++

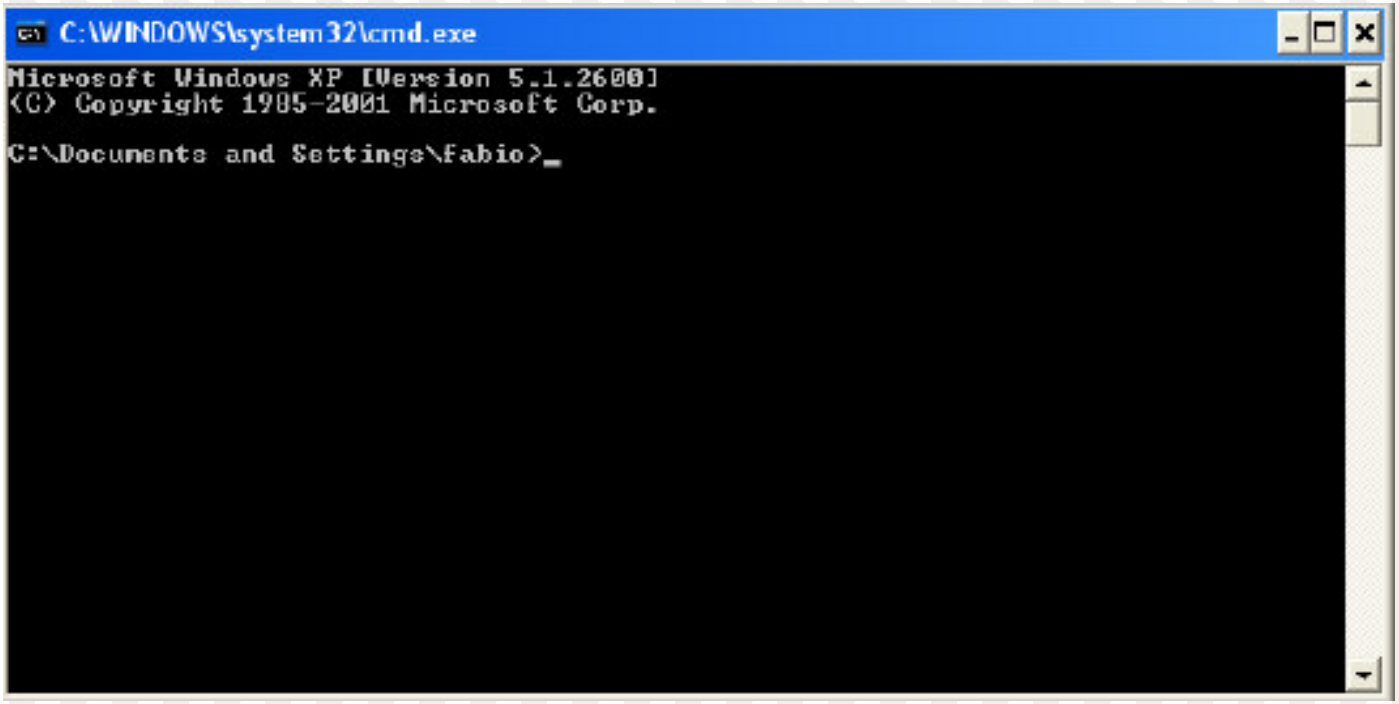
- **E' un ambiente semplice e diffuso.**
- **E' utile per muovere i primi passi nella programmazione.**
- **E' un software freeware.**
- **E' scaricabile da:**
  - **http://www.bloodshed.net/dev/devcpp.html**
  - **(http://www.bloodshed.net/dev/devcpp.html)**

Compilatore gcc

- **E' un compilatore freeware.**
- **E' compreso nel pacchetto DEVC++.**
- **E' scaricabile al seguente URL :**
  - **http://gcc.gnu.org .**
- **E' multipiattaforma (DOS, WINDOWS, LINUX, ALPHA, POWERPC, SUN, OS/2).**

Caratteristiche del gcc

- **Non è un ambiente grafico: si usa da riga di comando.**



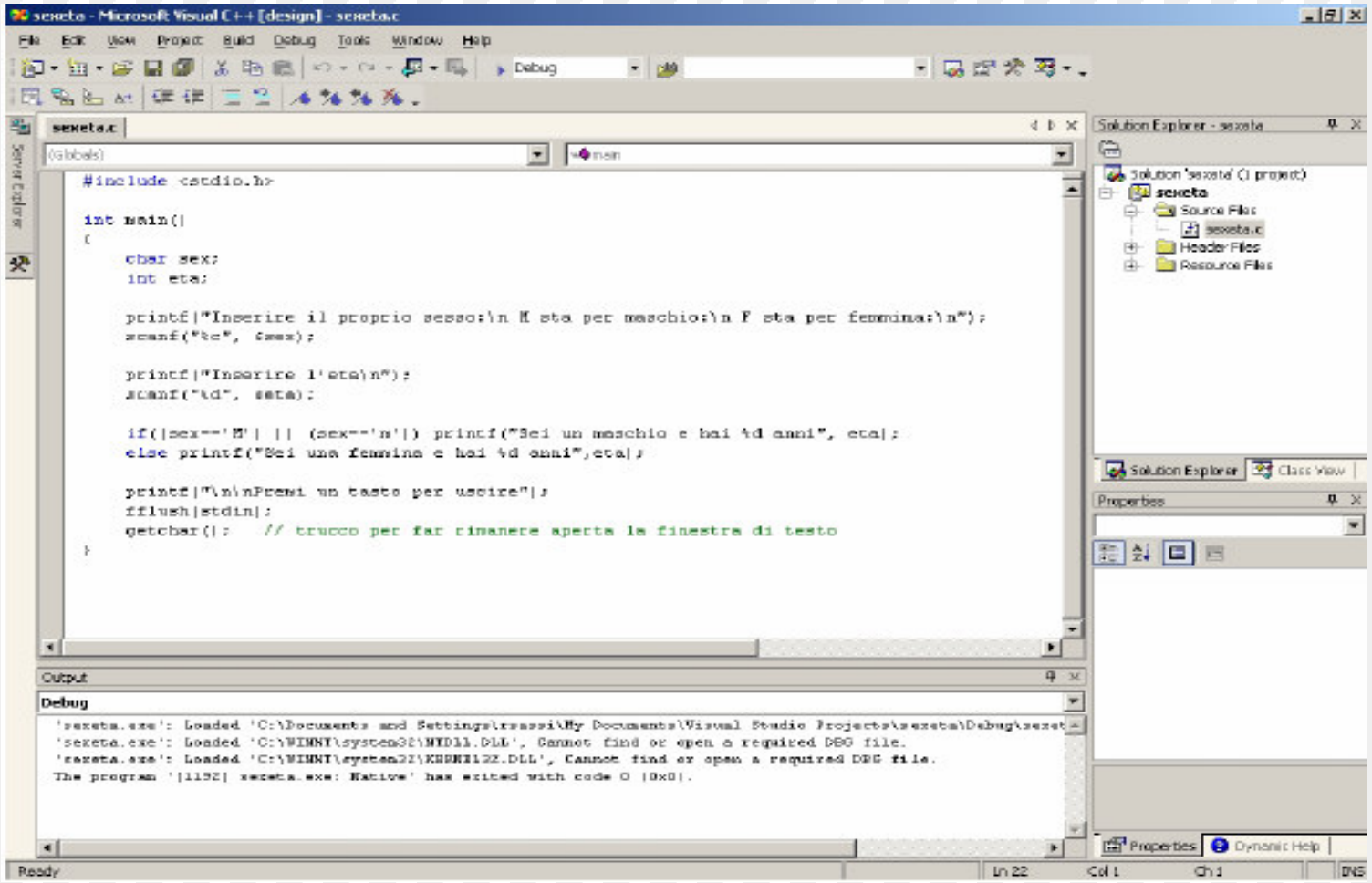
Piattaforma .Net

- **Prodotta dalla Microsoft (non freeware).**
- **Piuttosto diffusa.**
- **Impiegata per progetti di medie-grandi dimensioni.**
- **Integrata con notevoli librerie.**
- **Può essere impiegata per scopi didattici in questo corso.**

Progetto: significato in .Net

- **Non si può compilare un file .c in un passaggio, occorre CREARE UN PROGETTO.**
- **Un progetto è un insieme di FILE ed una struttura di DIRECTORY per la compilazione e l'editazione.**
- **Ciò deriva dal fatto che di solito programmi composti da migliaia di righe vengono divisi in MODULI SEPARATI.**
- **La piattaforma .Net può creare eseguibili per diverse applicazioni.**

Vista della piattaforma .Net



Esempio

- # include <stdio.h>
- int main() {
- printf("Hello World!");
- getchar();
- }
- **Salvare nel file helloword.c**

Struttura di un programma C

- **1. DIRETTIVE PER IL PRE-COMPILATORE**
- **2. Funzione MAIN:**
  - 1. parte DICHIARATIVA;
  - 2. dichiarazione delle COSTANTI;
  - 3. dichiarazione delle VARIABILI;
  - 4. parte ESECUTIVA.

Direttive per il pre-compilatore

- **Sono indirizzate al pre-compilatore e devono essere eseguite prima della compilazione.**
- **Iniziano con il carattere speciale "#" e sono moltissime.**
- **Quelle che ci interessano sono principalmente**
- **due:**
  - – includere le librerie
  - (es: "#include <stdio.h>");
  - – definire delle etichette
  - (es: "#define COSTANTE 10").
- **Vanno inserite in testa al programma (per motivi di buona programmazione)**

Posizione direttive pre-compilatore

- **# include <stdio.h>**
- int main() {
- printf("Hello World!");
- getchar();
- }

Funzione main()

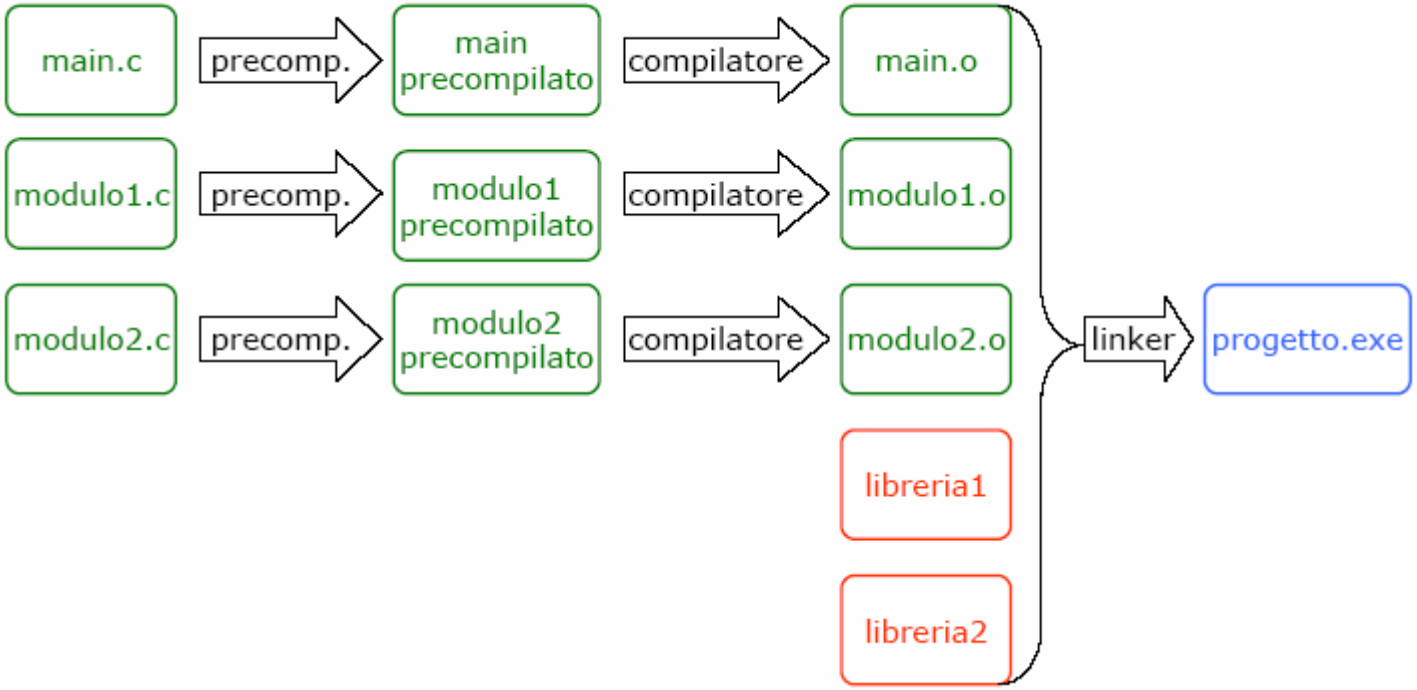
- **Contiene il programma vero e proprio che deve essere eseguito ed è divisa in più parti.**
- int main() {
- printf("Hello World!");
- getchar();
- }

Parte esecutiva

- **In essa sono contenute le istruzioni.**
  - **Alcune di queste sono funzioni predefinite,**
  - **altre appartengono a librerie standard, altre**
  - **ancora sono definite dal programmatore.**

- printf("Hello World!");
- getchar();

Cosa succede durante la compilazione?



Debugging (1)

- Strumento importantissimo per:
  - controllare il funzionamento del programma;
  - correggere eventuali errori.

- Il compilatore lavora in modo diverso durante la fase di debug.
- Tiene traccia di:
  - tutte le chiamate a funzione;
  - valore delle variabili;
  - istruzioni.

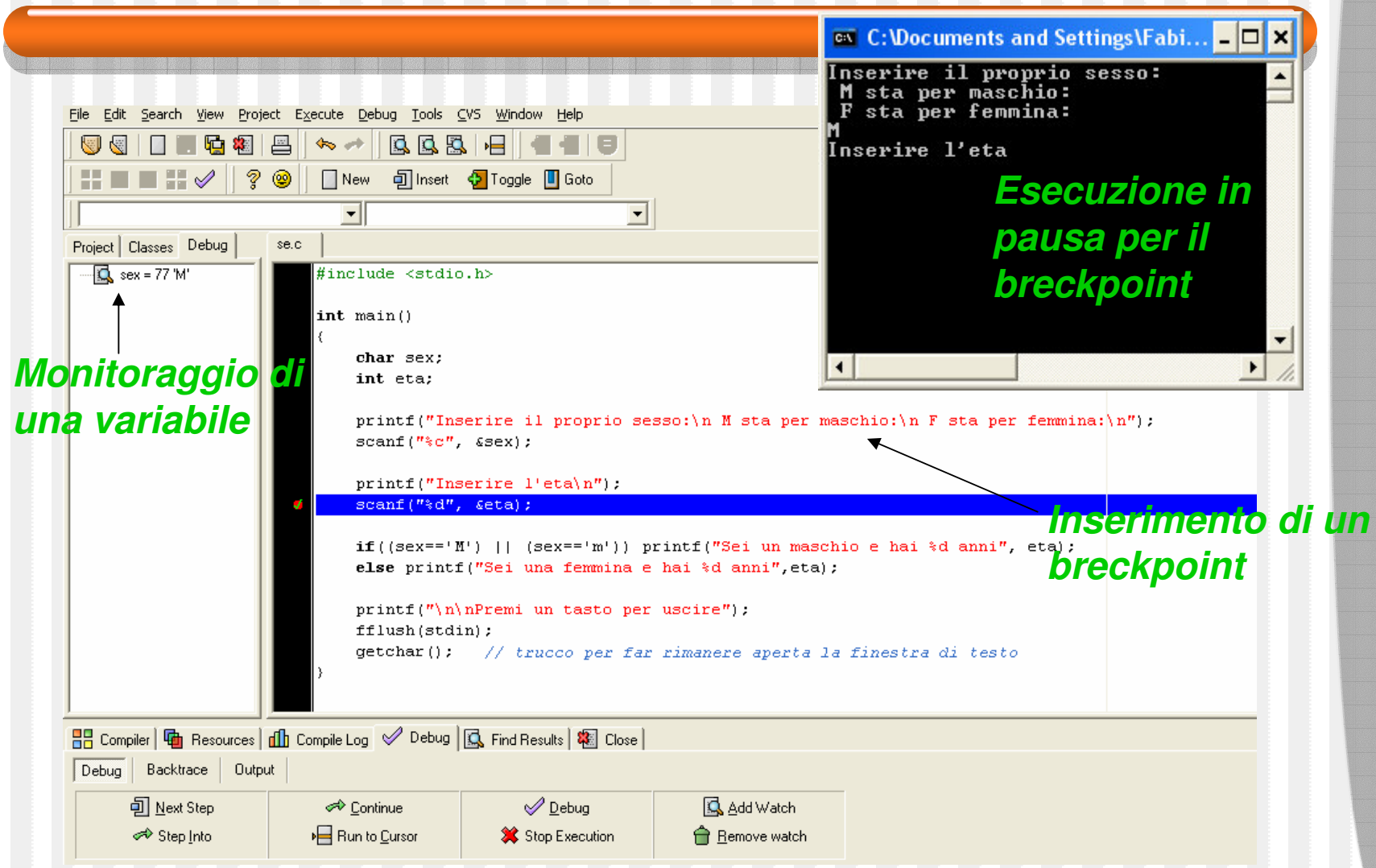
Debugging (2)

- Le possibilità che offre un normale debugger sono:
  - inserire dei **breakpoint** nel codice;
  - procedere una istruzione alla volta (**stepping**);
  - mostrare il **valore delle variabili**;
  - **backtraking**
    - - elenco delle funzioni che sono state chiamate prima di un breakpoint o una interruzione.
  - **finestra della CPU**
    - - monitorare i registri della CPU.

Abilitare il programma di debug

- dalla Ver. 4.9.9.1 il debugger si è già abilitato. Controllalo su
  → ***DevC++*** → ***Tools*** → ***Compiler Option*** → ***riquadro Settings*** → ***Linker:*** `Generate debugging information:YES`
- Nelle versioni vecchie (e nel dubbio) meglio abilitare il debugger a mano per evitare noie.
  → *DevC++* → *Tools* → *Compiler Option*
  → Inserire il check
  `Add the following commands when calling compiler`
  → Aggiungere *esattamente* questi parametri
  -g3 -Wall

Esempio di debug



Ringraziamenti

- **Fabio Scotti**
  - **Politecnico di Milano - Dipartimento di tecnologie dell'informazione**