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 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:

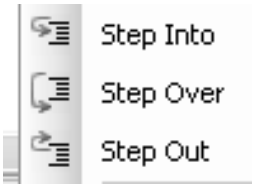
- Eseguire passo passo ogni istruzione
- Controllare lo "stato" del nostro programma
  - Visualizzare il contenuto delle variabili (monitoraggio)
  - Visualizzare lo stack delle chiamate a funzione
  - …

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")

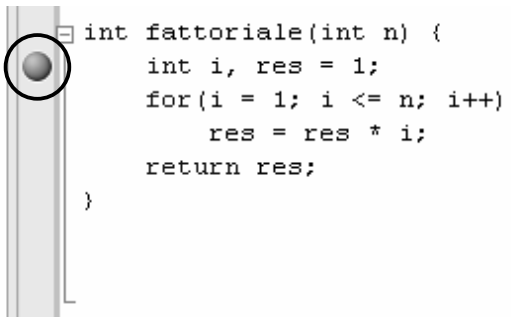


- E se l'istruzione chiama una funzione?
  - Step Into ➔ continua il debug entrando nel codice della funzione
  - Step Over ➔ continua il debug ripartendo dal punto immediatamente successivo alla chiamata di funzione (ovvero esattamente dopo la restituzione del valore)
- Se sono all'interno di una funzione,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

Breakpoints (1)

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

Breakpoint
Click per aggiungerne uno



```
int fattoriale(int n) {
    int i, res = 1;
    for(i = 1; i <= n; i++)
        res = res * i;
    return res;
}
```

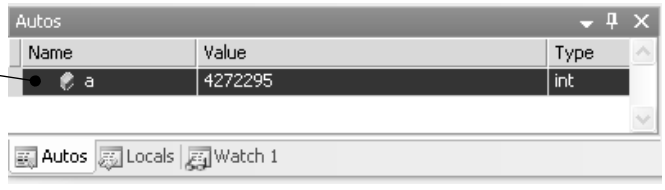
Breakpoints (2)

- 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 cera variabile è uguale a 0)
  - Menu Debug ➔ Windows ➔ Breakpoints apre la finestra di configurazione e definizione dei vari breakpoints

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

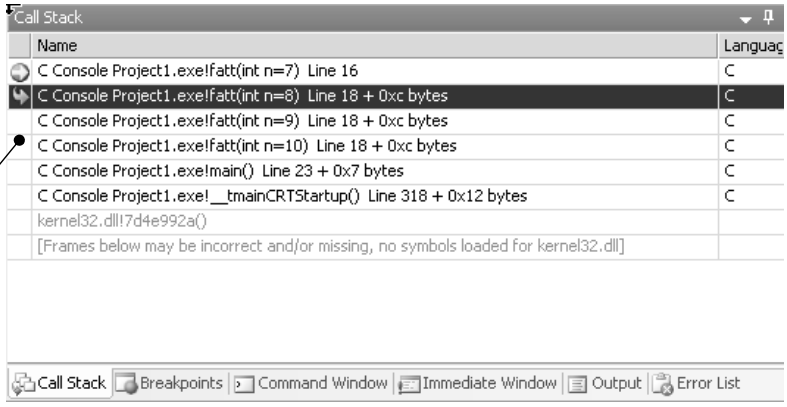


  - Local
    - Visualizza il contenuto delle variabili "locali", ovvero tutte quelle visibili all'interno della funzione corrente (nota: in caso di scope innestati con variabili con lo stesso nome, compaiono ripetizioni!)
  - Watch
    - Permette di inserire il nome della variabile da monitorare (occhio agli scope!)
    - E' 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)
  - E' 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 Rivisitato

- Creare un nuovo progetto per illinguaggio 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;
   system("PAUSE");
}
```

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

Esercizio Rivisitato

- ATTENZIONE: Non copiate il codice direttamente dal pdf, poichè vengono introdotti caratteri strani (non visibili) che rendono incompilabile il programma!!!

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

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

- Cosa succede quando viene eseguita l'istruzione "a=a-3;" ??? Qual'e' il valore di a prima e dopo l'esecuzione della variabile?