Fondamenti di Informatica

Ing. Biomedica

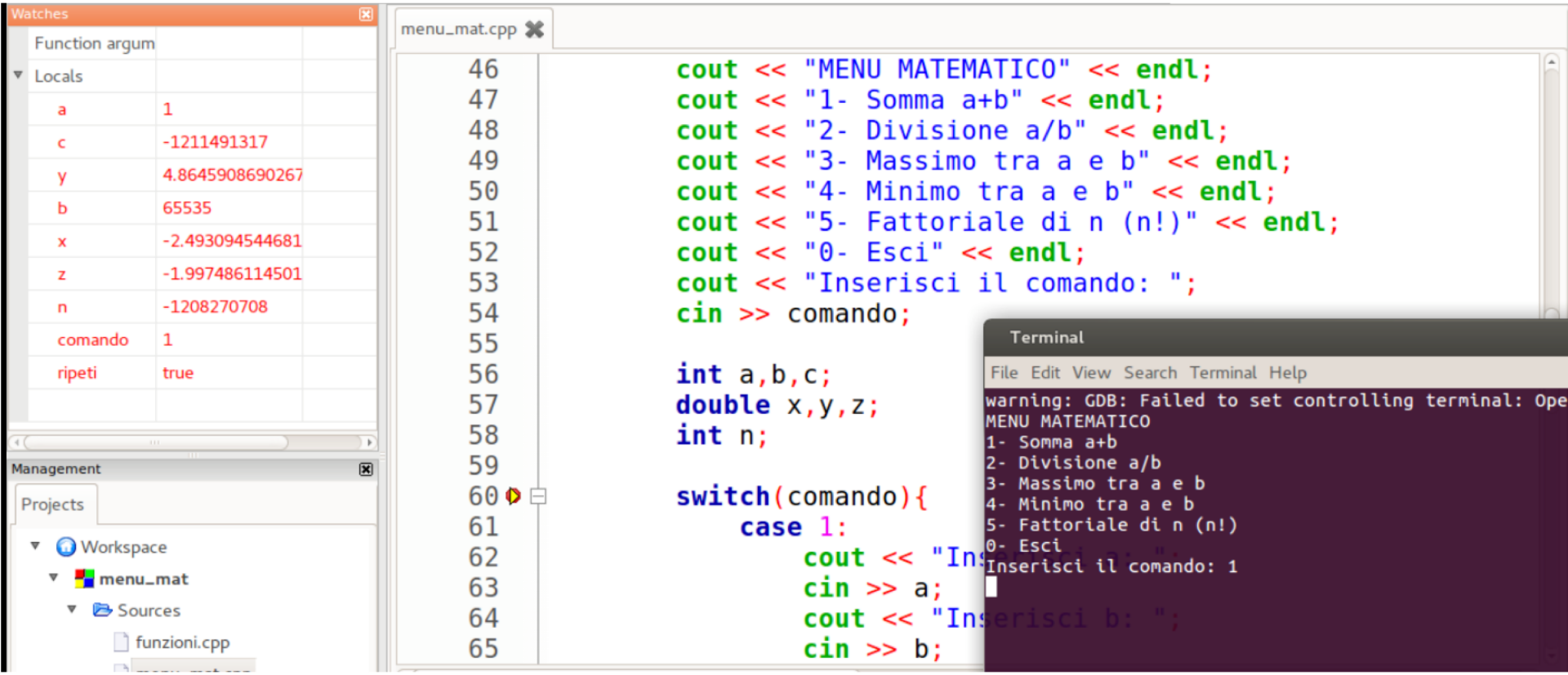
Esercitazione n.5

Debugger & Funzioni Ricorsive

Antonio Arena
antonio.arena@ing.unipi.it

Debugger

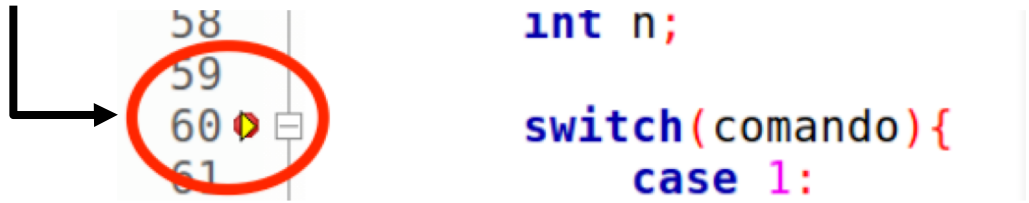
- Cos'è?
  Il debugger è uno strumento molto potente che permette l'analisi e l'eliminazione dei bug *(debugging)*, ovvero gli errori di programmazione.



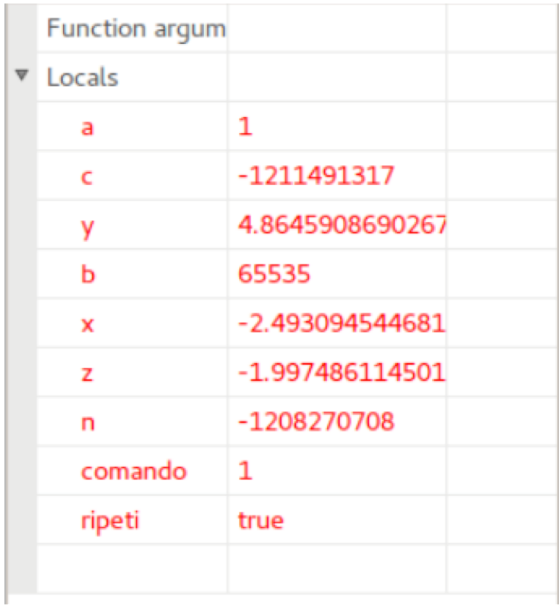
Debugger

- Come funziona?

1. Si comincia ad eseguire il programma.
2. A un certo punto l'esecuzione si «congela» su una qualche istruzione del programma (scelta da chi sta facendo il debugging!!). Il punto in cui il debugger si ferma si chiama *breakpoint.*



3. Non appena il programma si termina, il programmatore può esaminare lo stato delle variabili e di tutta la memoria in generale.



Debugger - Breakpoints

1. I breakpoint possono essere inseriti sia prima dell'esecuzione del programma con il debugger, sia durante l'esecuzione. Si può inserire cliccando a destra del numero di riga in cui ci si vuole fermare.
2. E' importante che ce ne sia almeno uno prima di far partire il debugger, altrimenti l'esecuzione non si fermerà mai per poter essere ispezionato.

```
41  int main(){
42      int comando;
43      bool ripeti = true;
44
45      while(ripeti){
46          cout << "MENU MATEMATICO" << endl;
47          cout << "1- Somma a+b" << endl;
48          cout << "2- Divisione a/b" << endl;
49          cout << "3- Massimo tra a e b" << endl;
50          cout << "4- Minimo tra a e b" << endl;
51          cout << "5- Fattoriale di n (n!)" << endl;
52          cout << "0- Esci" << endl;
53          cout << "Inserisci il comando: ";
54          cin >> comando;
55
56          int a,b,c;
57          double x,y,z;
58          int n;
59
60          switch(comando){
```

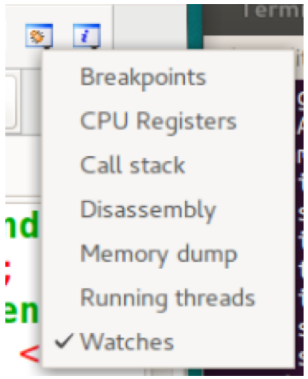
NB: l'ultima istruzione eseguita è quella precedente al breakpoint! Quindi in questo caso, il debugger eseguirà l'istruzione alla riga 48 e si fermerà sulla 49 aspettando un comando dall'utente

Debugger – HOW TO

1. Il debugger ha una sua toolbar apposita. Una volta compilato il programma (tramite il pulsante build), basta premere sul pulsante «Debug/Continue» 

2. Una volta che l'esecuzione si è «congelata», si può proseguire l'esecuzione fino al prossimo breakpoint inserito premento di nuovo il pulsante «Debug/Continue», altrimenti si può andare all'istruzione successiva al breakpoint premente il pulsante «Next Line» 

3. Per poter vedere il contenuto delle variabili dichiarate basta cliccare su «Debugging Windows» -> «Watches» 



4. «Call stack» permette di vedere tutte le chiamate di funzioni fatte nel momento che l'esecuzione viene «congelata»

Funzioni Ricorsive – Esercizio n.1

- Realizzare in C++ una funzione che calcoli la serie armonica (troncata), *ovvero*:

$$f(n) = \sum_{k=1}^{n} \frac{1}{k}$$

i.e. somma dei reciproci dei primi n numeri naturali.

- n va letto da tastiera
- la funzione deve essere **ricorsiva**

Funzioni Ricorsive – Esercizio n.1

- Analizziamo i casi più semplici:
  - f(1) = 1
  - f(2) = 1/1 + 1/2 = f(1) + 1/2
  - f(3) = 1/1 + 1/2 + 1/3 = …

  .

  .
  - f(n) = ?? … ??

- Ricorda: una funzione ricorsiva è una funzione che chiama sé stessa.
- Gli ingredienti sono:
  1. Un caso base dove non c'è la chiamata ricorsiva (qui qual è il caso base?)
  2. Un caso generale dove si richiama la funzione con un input che prima o poi «casca» nel caso base…

Esercizio n.2

- Realizzare in C++ una funzione ricorsiva che calcoli. l'n-esimo numero pari positivo, ovvero

$$f(n) = 2 * (n - 1)$$

n.b. zero è un numero pari!

Ovviamente, in C++ sarebbe banale: si legge n, si calcola 2*(n-1) e si stampa a video il risultato della moltiplicazione.

Ma usando una funzione ricorsiva?

Esercizio n.2

- Analizziamo i casi più semplici:
  - f(1) = 2*(1-1) = 0
  - f(2) = 2*(2-1) = 2 = f(1) + 2
  - f(3) = 2*(3-1) = 4 = f(2) + 2

  .

  .

  - f(n) = ?? … ??

1. Qual è il caso base?

Esercizio n.3

- Realizzare in C++ una funzione ricorsiva che calcoli. l'n-esima potenza di 2, ovvero

$$f(n) = 2^n$$

- Anche qui, analizziamo i casi:
  - f(0) = 2^0 = 1
  - f(1) = 2^1 = 2 = 2 * f(0)
  - f(2) = 2^2 = 4 = 2 * f(1)

  .

  .
  - f(n) = ?? … ??

Esercizio n.4

- Realizzare in C++ una funzione ricorsiva che prende un intero n e stampa a video n asterischi

- Esempio:
  asterischi(5) stampa a video `* * * * *`

  asterischi(1) stampa a video `*`

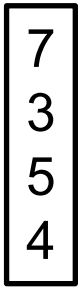
  asterischi(0) stampa a video … niente

Esercizio n.5 – Difficile!

- Realizzare in C++ una funzione ricorsiva che prende un intero e lo stampa a video con tutte le cifre incolonnate (a partire da quella più significativa).

- Esempio:
  scrivi_verticale(12345) stampa a video

scrivi_verticale(7354) stampa a video



scrivi_verticale(2) stampa a video

Esercizio n.5 – Difficile!

- Caso base:
  se il numero letto ha una sola cifra, allora stampa la cifra
- Caso generale:
  se non ha una sola cifra:
  1. stampa il numero incolonnato togliendo l'ultima cifra
  2. scrivi l'ultima cifra

ricorda:

- n/10 da come risultato n con l'ultima cifra rimossa
  1234/10 = 123
- n%10 da come risultato l'ultima cifra di n
  1234%10 = 4

Esercizio n.5 – Difficile!

PseudoCodice

```
stampa_verticale(n)
   se n<10 {
      stampa n
   }
   altrimenti{
      stampa_verticale( «n senza l'ultima cifra» )
      stampa ultima cifra
   }
}
```