Laboratorio di informatica
Ingegneria meccanica

Esercitazione 6 – 7 novembre 2007

Correzione di un programma

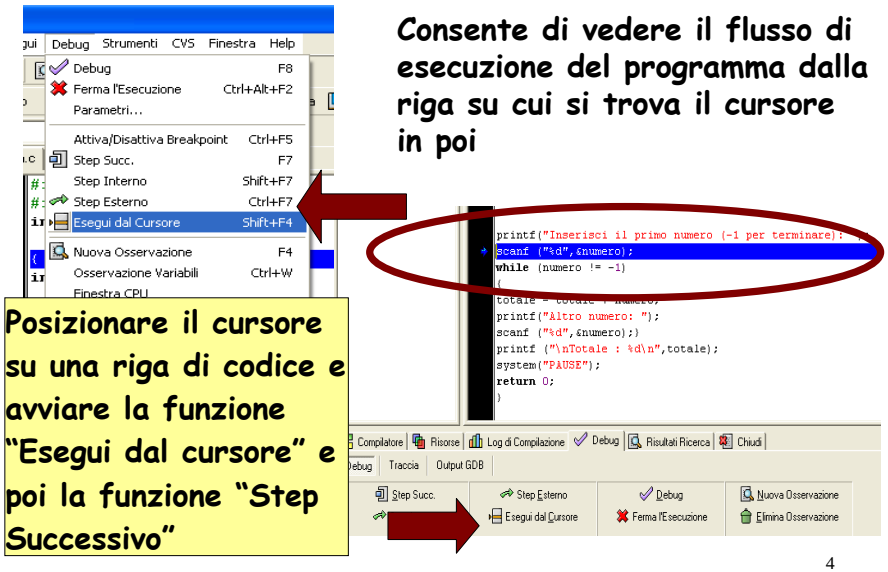
Un programma può contenere errori sintattici e/o errori logici.
Gli errori sintattici vengono segnalati dal compilatore e non viene generato il codice eseguibile
Gli errori logici sono più difficili da individuare poiché viene generato il codice eseguibile, ma il comportamento del programma non corrisponde a quello atteso (il programma può produrre un errore a tempo di esecuzione o può produrre un risultato sbagliato anche solo per alcuni valori dell'input).

Esistono degli strumenti (debugger) che aiutano ad individuare gli errori logici, mediante l'analisi del flusso di esecuzione del programma e del valore assunto dalle variabili durante l'esecuzione.

Debugger: strumenti principali

- Esecuzione passo passo
- Breakpoint
- Ispezione delle variabili

Debugger: esecuzione dal cursore



Consente di vedere il flusso di esecuzione del programma dalla riga su cui si trova il cursore in poi

Posizionare il cursore su una riga di codice e avviare la funzione "Esegui dal cursore" e poi la funzione "Step Successivo"

Debugger: breakpoint

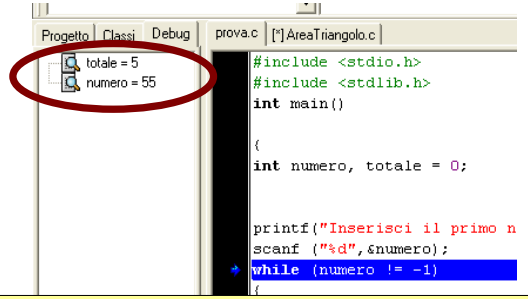
I breakpoint sono punti di interruzione che si possono inserire all'interno del codice del programma. Ad ogni breakpoint incontrato il programma si arresta permettendo di analizzare il valore assunto dalle variabili o di attivare l'esecuzione passo passo a partire da quel punto

```
scanf ("%d" , &num);
while ( num != -1 ) /* controllo valore sentinella */
{
cont= cont + 1;
printf ("Inserisci un altro numero:\n");
scanf( "%d" , &num ); /* modifica valore sentinella */
}
printf ("Numero di interi letti = %d\n", cont);
system("pause");
return 0;
}
```

Posizionare il cursore su una riga di codice e avviare la funzione "attiva un breakpoint".

Debugger: ispezione delle variabili

Consente di ispezionare il valore delle variabili durante l'esecuzione del programma.



Avviare il debugger e posizionare il mouse sopra una variabile. La variabile viene aggiunta nella finestra di osservazione dell'area progetto. Alternativamente si può impostare l'osservazione selezionando la voce "Nuova Osservazione" nel programma ed inserendo il nome della variabile

Esempio su cui eseguire il debug

```
#include <stdio.h>
#include <stdlib.h>
int main( )
{
    int num, cont = 0 ;
    printf ("Inserire un intero positivo (-1 per terminare)\n");
    scanf ("%d" , &num);
    while ( num != -1 ) /* controllo valore sentinella */
    {
            cont= cont + 1;
            printf ("Inserire un intero positivo (-1 per terminare)\n");
            scanf( "%d" , &num ); /* modifica valore sentinella */
    }
    printf ("Numero di interi letti = %c\n", cont);
    system("pause");
    return 0;
}
```


Esercizio 1

Scrivere un programma che, acquisiti da stdin
- un primo intero $1 \leq n \leq 15$
- n interi positivi, uno per volta, da assegnare agli elementi successivi di un array a di 15 interi
- un secondo intero i da ricercare in a,

visualizzi su stdout
- le posizioni di occorrenza di i tra i primi n elementi di a e il loro numero complessivo c, se $c \geq 1$
- oppure -1 e 0, se $c = 0$

Verificare il funzionamento del programma nei seguenti casi:
n = 1 ed i presente 0, 1, o n volte
n = 15 ed i presente 0, 1, o n volte

Esercizio 2

Scrivere un programma che, dopo aver acquisito da stdin 15 interi in un array a di 15 interi e due ulteriore interi i e j, con $0 \leq i < j \leq 15$, ricerchi in a il minimo e il massimo tra gli elementi nelle posizioni comprese tra i e j e visualizzi su stdout

- il valore del minimo
- il valore del massimo
- la posizione (compresa tra i e j) di prima occorrenza del minimo
- la posizione (compresa tra i e j) di prima occorrenza del massimo

Esercizio 3

Scrivere un programma che, letta da stdin una sequenza di caratteri terminata da invio, per ciascun carattere della sequenza esegue la seguente azione:

- se il carattere è una lettera minuscola stampa su stdout la corrispondente lettera maiuscola
- se il carattere è una lettera maiuscola stampa su stdout la corrispondente lettera minuscola
- in tutti gli altri casi stampa uno spazio

Nella rappresentazione ASCII

- le lettere maiuscole hanno codici compresi tra 65 ('A') e 90 ('Z')
- le lettere minuscole hanno codici compresi tra 97 ('a') e 122 ('z')

```
Inserire una sequenza di caratteri terminata da
invio:
abc12.Ief2g
ABC   iEF G
Premere un tasto per continuare . . .
```


Esercizio 4

Scrivere un programma C che legga da stdin una sequenza di numeri positivi la cui lunghezza non è nota a priori, terminata da un numero negativo. Per ogni numero letto il programma deve stampare su stdout la media di tutti i numeri letti fino a quel momento.

```
Inserisci un numero positivo (o negativo per terminare): 2.2
Media attuale (1 numero/i): 2.200000
Inserisci un numero positivo (o negativo per terminare): 3.3
Media attuale (2 numero/i): 2.750000
Inserisci un numero positivo (o negativo per terminare): 5.5
Media attuale (3 numero/i): 3.666667
Inserisci un numero positivo (o negativo per terminare): 0
Media attuale (4 numero/i): 2.750000
Inserisci un numero positivo (o negativo per terminare): -1
Premere un tasto per continuare . . .
```


Esercizio 5

Leggere da stdin una sequenza di 0 e 1 terminata da 2 (acquisire i valori uno per volta) e stampare la lunghezza della più lunga sottosequenza di soli 0 presente nella sequenza letta

Esempio: per la sequenza

0 0 1 0 0 0 1 1 1 1 0 0 2

la risposta cercata è 3

Algoritmo Esercizio 5

Variabili utilizzate (tipo intero):
bit: valore letto, contatore: numero di 0 accumulati
lmax: massima lunghezza sottosequenza di 0

- contatore=0; lmax=0 (inizializzazione)
- leggi un numero (valore registrato in bit)
- finché bit è diverso da 2
  - se bit è pari a 0:
    - incrementa contatore
    - se contatore > lmax: lmax=contatore
  - altrimenti
    - contatore=0
  - leggi un altro numero
- stampa lmax

Esercizio 6

Un intero N > 1 è detto primo se i suoi unici divisori sono 1 e N

Scrivere un programma che legge da stdin un intero e determina se è primo

Algoritmo (*inefficiente*): provare se tra i numeri compresi tra 2 e N-1 c'è un divisore di N

Algoritmo(*inefficiente*) Esercizio 6

Variabili utilizzate (tipo intero):
numero: valore letto, provadiv: possibile divisore di numero, trovatodiv: diventa vero (1) se si trova un divisore di numero

- provadiv=2; trovatodiv=0 (inizializzazione)
- leggi valore (registrato in numero)
- finché provadiv < numero
  - se provadiv divide numero: trovato=1
  - provadiv=provadiv+1
- se trovato=1: numero non è primo
  altrimenti: numero è primo

Esercizio 7

Scrivere una programma che, letto da stdin un numero intero positivo N, calcola la parte intera della radice quadrata di N.

```
Inserire un numero intero positivo: 7

Parte intera della radice quadrata di 7 = 2

Premere un tasto per continuare . . .
```

Esercizio 8

Scrivere un programma che, letti da stdin un intero positivo *n* e un numeri reale x, calcola lo sviluppo di Taylor di ordine *n* della funzione e^x, dato dalla seguente formula:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!}$$

```
Calcolo dello sviluppo di Taylor di ordine n di e(x)
Inserire x: 1.5
Inserire n: 6
4.477539
Premere un tasto per continuare . . .
```