# 2. Seconda esercitazione autoguidata

Le soluzioni agli esercizi, le versioni di programmi dati nel testo delle esercitazioni e quant'altro sono raggiungibili tramite la pagina web del corso.

**COMPILAZIONE in ANSI C**
Qui usiamo una definizione standard del C, che si chiama ANSI C. Quindi dobbiamo assicurarci che il compilatore "usi" anche lui la medesima definizione. Per esserne sicuri, bisogna assegnare opportunamente una certa opzione del TurboC: aprire il menù OPTIONS, selezionare ***Compiler*** e poi ***Source***. Nella finestra di scelta che appare, selezionate ANSI C.

## *2.1. SCRITTURA E VERIFICA DI UN PROGRAMMA.*

Usando i metodi spiegati nella prima esercitazione, adesso scriviamo un

**programma che calcola le due radici di un'equazione di secondo grado (si suppone che l'equazione abbia solo due radici reali, cioè delta>0)**

Se a, b, c sono i coefficienti dell'equazione, le radici si calcolano usando la radice quadrata del delta. In questo esercizio, quindi, si dovrà usare la funzione `sqrt()`, appartenenente alla libreria di funzioni matematiche (bisognerà includere nel programma il file `math.h`).

Ecco un algoritmo per la soluzione:

1) stampare la richiesta per il primo coeff e leggerlo;
2) idem per il secondo coeff;
3) idem per il terzo coeff;
4) calcolare il valore di delta
5) calcolare il valore della soluzione $x_1$
6) calcolare il valore della soluzione $x_2$
7) stampare le soluzioni

Una soluzione proponibile è nel file `eq2.c`. Scrivere il programma e provarlo. Durante la sperimentazione del programma può esser utile quel che è scritto nelle TRE sezioni successive.

## *2.2. SPOSTAMENTO E RIDIMENSIONAMENTO DI UNA FINESTRA*

Se già non è aperta, aprire la Finestra di Output. Posizionarla in modo tale da vederla contemporaneamente alla Finestra di Edit.

### Attivazione di una finestra già aperta

Usando il tasto **F6**, oppure cliccando sulla finestra di interesse, si può attivare l'una o l'altra delle finestre aperte sullo schermo. La finestra attiva si distingue per una doppia linea di contorno.

### Spostamento della finestra

Selezionare la finestra da muovere e attivare il comando ***Size/Move*** del menù **WINDOW**.
Premendo i tasti freccia la finestra si muove di conseguenza.

In alternativa si può cliccare una volta con il mouse sul bordo superiore della finestra. Mantenendo premuto il tasto di clic e muovendo il mouse la finestra si muove di conseguenza (questa è la funzione di *drag* = trascinamento).

### Ridimensionamento della finestra

Selezionare la finestra da ridimensionare e attivare il comando ***Size/Move*** del menù **WINDOW**.

Premendo il tasto delle maiuscole ⇑ (“shift”) **e** contemporaneamente un tasto freccia, la finestra cambia dimensione conseguentemente alla freccia premuta.
Per smettere di posizionare/ridimensionare, premere invio.

Con il **mouse**, per ridimensionare la finestra attiva "cliccare" (con il pulsante sinistro) l'angolo inferiore destro e muovere il mouse mantenendo premuto il pulsante.

## 2.3. ESECUZIONE PASSO-PASSO

Un programma TurboC può essere eseguito una linea alla volta, premendo ripetutamente il tasto **F7** *("trace”)* oppure il tasto **F8** *(“step”).*
La linea che sta per essere eseguita sarà evidenziata con un colore diverso nel testo.

(Le differenze tra F7 e F8 non sono importanti per ora.)

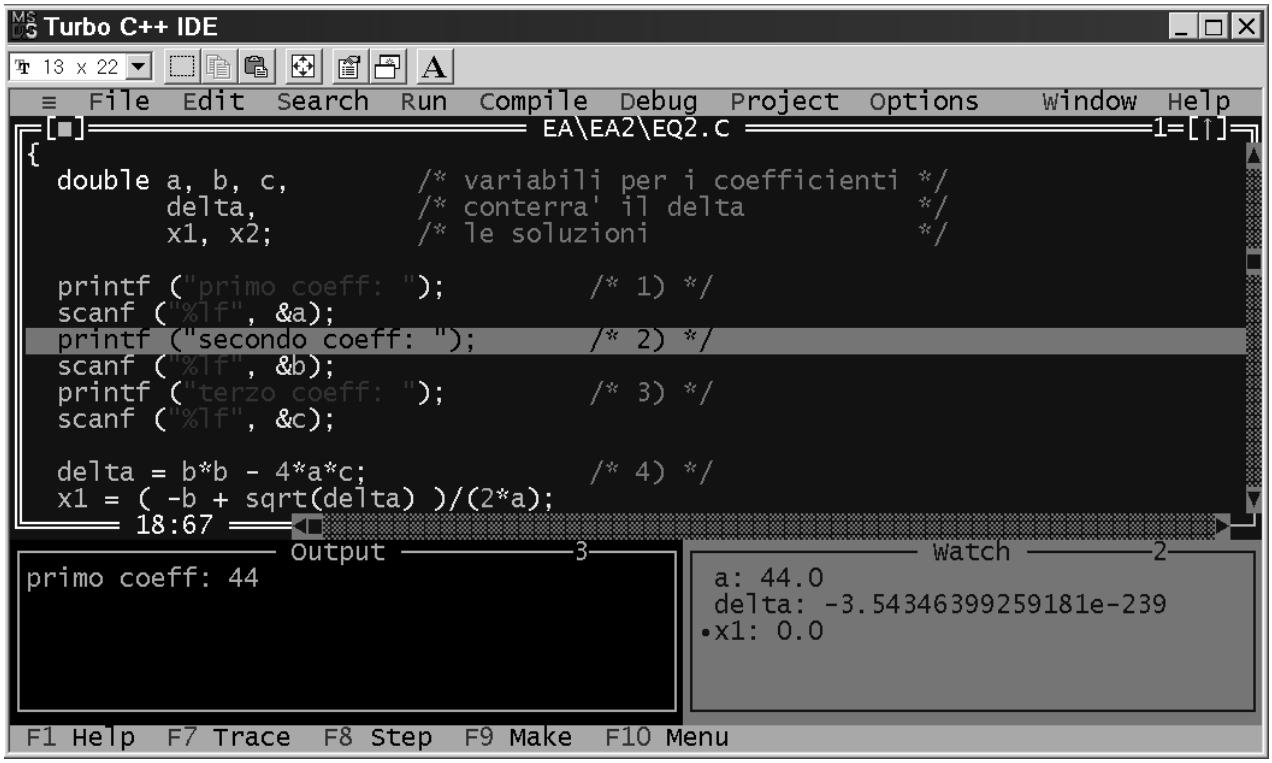
Se sulla linea ci sono più istruzioni, tutte verranno eseguite e si passerà alla porssima linea. In effetti è consigliabile mettere una sola istruzione per linea, così vedremo l’esecuzione del programma “una istruzione alla volta” (*passo-passo*).

**Provare ad eseguire passo-passo il programma fatto prima.**

Eseguendo le istruzioni di lettura il programma
- passa allo schermo di esecuzione (***user screen***),
- attende un dato da leggere (perciò, nel nostro caso, si deve inserire un numero e premere invio),
- e poi torna a mostrare le istruzioni del programma.

## 2.4. CONTROLLO DEL VALORE DELLE VARIABILI (WATCH)

Il valore di espressioni (e quindi anche di variabili del programma) può essere tenuto costantemente sotto controllo mentre si esegue passo-passo il programma. Ciò è utilissimo durante la ricerca di errori di codifica o logici, non rilevabili automaticamente.




**esecuzione passo-passo con watches**:

in figura, usando ripetutamente F7, abbiamo eseguito la prima printf e poi la prima scanf; ora siamo posizionati sulla seconda printf. Quando premeremo F7 questa istruzione verrà eseguita e ci si posizionerà sulla successiva scanf. Nell’eseguire la scanf, apparirà lo user screen (tutto schermo nero) con la richiesta del secondo input etc...

Inserire la variabile che rappresenta il primo coefficiente dell'equazione (nella soluzione proposta è `a`). Appare una finestra dal titolo "Watches" con l' identificatore in questione.

Per comodità (vedi figura successiva), posizionare e dimensionare la finestra "Watches" in modo da poterla vedere contemporaneamente alle altre due (Output ed Edit).

Eseguendo ora il programma, passo-passo, si vede come, accanto all'identificatore in questione, appaia il suo valore attuale, cioè il valore che è effettivamente presente nella locazione di memoria associata all'identificatore). Tipicamente, prima che qualche istruzione abbia assegnato un valore ad una certa variabile, questa contiene valori non significativi (e imprevedibili).

Rieseguendo il procedimento (**Debug**-***Add watch***), aggiungere le altra variabili usate nel programma (suggeriamo quelle che abbiamo chiamato `delta, x1, x2`, nella soluzione proposta). La finestra Watches si può comunque aprire con il comando ***Watch*** del menù **Window**.

Eseguire adesso un' altra volta il programma con **F7** o **F8**, osservando come cambiano i valori delle variabili ispezionate durante l' esecuzione.

### *2.5. ESERCIZIO (eq2B.c)*

Il programma sviluppato prima è limitato dall'assunzione che l'equazione di secondo grado presenti esclusivamente soluzioni reali. Se eseguiamo quel programma con coefficienti, ad esempio 12 1 1, il delta è negativo e l'esecuzione della funzione `sqrt()` su esso produce errori (quella funzione, ovviamente è prevista solo per argomenti non negativi).

Modificare il programma per l'equazione di secondo grado, in modo che, quando il delta risulta negativo, le soluzioni non vengano calcolate, ma ci si limiti a stampare un messaggio del tipo `"tsk, tsk ... soluzioni complesse coniugate"`.

Una prima soluzione potrebbe consistere nel mettere "sub iudice" i punti 5) 6) e 7) dell'algoritmo presentato sopra:

5) **se** delta>=0 calcolare la prima radice
6) **se** delta>=0 calcolare la seconda radice
7) **se** delta>=0 stampare $x_1$ e $x_2$ **altrimenti** stampare il messaggio negativo

Provare a programmare questa soluzione, testandola con differenti esecuzioni; soluzione proposta in `eq2B.c`.

### *2.6. ESERCIZIO (eq2C.c)*

Stesso problema dell'esercizio precedente; però cerchiamo di applicare una soluzione più elegante: l'espressione/condizione `(delta>=0)` deve essere calcolata una volta sola.

Provare a programmare questa soluzione, testandola con differenti esecuzioni; soluzione proposta in `eq2C.c`.

### *2.7. ESERCIZIO (formati1.c)*

Ricordiamo i principali formati di stampa utilizzabili per i numeri reali:

| | |
|---|---|
| `%f` | (floating point, forma decimale normale: `4572.93376912`) |
| `%e` | (forma esponenziale: `4.57293376912e+3`) |
| `%g` | forma generale (seleziona automaticamente una delle altre, in base alle dimensioni della variabili e dell'output) |

Ogni volta che usiamo questi formati, possiamo anche specificare delle "formattazioni" dell'output:

- il numero di posizioni su cui il numero verrà stampato;
- il numero di cifre frazionarie da mostrare (le cifre dopo il punto decimale).

`%n.mf` significa che il numero verrà stampato su `n` posizioni (cioè riempiendo `n` caratteri), con `m` cifre frazionarie. (Il punto decimale è uno degli `n` caratteri).

Per fare pratica con questi formati, analizzare il codice (ed eseguirlo!) del programma `formati1.c`. (Si guarda il codice, lo si esegue, si capisce perché certe cose vengono stampate in un certo modo ...). Chi vuole può sperimentare stampe diverse (magari usando un file copia di quello originale).

### *2.8. ESERCIZIO (farh.c)*

Scrivere un programma che, ricevuti in input un valore di temperatura iniziale (`ti`, in gradi celsius) e un passo (`ps`), entrambi interi, produce in output una tabella contenente le temperature celsius `ti, ti+ps, ti+2ps` e `ti+3ps` insieme alla corrispondenti farheneit. Sotto si vede un esempio di output del programma. Si ricorda che se C è una temperatura celsius, la corrispondente temperatura farheneit è F=(9/5)C+32.

Nel programma, definire due simboli costanti `RAPPORTO` (con valore 9.0/5.0) e `DIFF` (32). Che succede se si usa una definizione di RAPPORTO come (9/5) invece? Ricordare l'overloading dell'operatore /: 9/5 è un valore intero e vale 1)

```
fornire temperatura d'inizio e passo: 30 2

-----------+------------
 celsius   | farheneit
      30   |     86.000
      32   |     89.600
      34   |     93.200
      36   |     96.800
-----------+------------
FINE
```

### *2.9. ESERCIZI (segmento.c, max2.c, max3.c)*

Scrivere un programma che, ricevuti in input gli estremi di un segmento della retta reale (due numeri razionali), calcoli e stampi la lunghezza del segmento, come valore assoluto della differenza tra gli estremi. Soluzione proposta in `segmento.c`.

Scrivere un programma che, ricevuti in input due numeri interi, assegni ad una variabile `max` il massimo tra i due e poi lo stampi. Soluzione proposta in `max2.c`.

Scrivere un programma che, ricevuti in input tre numeri interi, assegni ad una variabile `max` il massimo tra i tre e poi lo stampa. Soluzione con *if-else innestati*, proposta in `max3.c`.

### *2.10. TECNICA DEL MASSIMO PARZIALE (max4.c, max3c.c)*

Scrivere un programma che, ricevuti in input quattro numeri interi, utilizzando la tecnica del massimo parziale, trova e stampa il valore massimo. Soluzione proposta in `max4.c` (con inizializzazione di `maxparz` a `0`). Questa soluzione soffre del problema che se i numeri dati in input sono negativi il risultato stampato è 0. Si tratta di una soluzione limitata alle sole quaterne di numeri positivi (o nulli). In `max4b.c` è proposta la soluzione più generale, con inizializzazione di `maxparz` con il primo dei numeri dati).