Laurea In Ingegneria dell'Informazione

Esercitazioni Guidate di Tecniche della Programmazione

Note introduttive:

1) Nel titolo di ogni sezione di questo documento è specificato tra parentesi il nome del (o dei) file in cui è proposta una soluzione (se disponibile nella directory "`programmi`" di questa EG).
2) I programmi che scriveremo dovranno essere in accordo con la definizione standard **ANSI C** del linguaggio C.
    a. Se usate un sistema diverso dal DEV, provvedete a che la compilazione avvenga con il compilatore standard C.
    b. Ricordate che un programma C è in un file con estensione "`.c`"
    c. Se usate il DEVC++, per configurare bene la compilazione bisogna
        i. andare nel menù "Tools", selezionare "Compiler Options", scegliere "Settings" e poi "C Compiler"; poi selezionare almeno "Support all ANSI Standard C Programs"
        ii. (se l'interfaccia è in italiano …) andare nel menu' "Strumenti", selezionare "Opzioni di compilazione", "Compilatore", "Generazione di Codice …", "Compilatore C" e poi far apparire "Yes" almeno accanto a "Supporto programmi ANSI standard C".

NB le immagini in queste dispense sono prese dalla versione 4.9.9.2 del DEV. La versione 5.11 (che e' quella corrispondente alla cartella "portable" che abbiamo discusso a lezione è più recente e può differire solo in qualche dettaglio.

1. Esercitazione 1

1.1. Primer per DEVC++ (`ourFirst.c`)

Si comincia dal Primer, cioè il documento di guida iniziale all'uso del DEV, con un esempio. Il documento è disponibile tramite il sito web del corso (Complementi didattici ad una delle prime lezioni …). Inutile copiarlo qui … Prima leggete ed usate quello.

1.2. Capitalism (`capital0.c`)

Scrivere un programma che calcola la crescita di un capitale iniziale (ad esempio 99 euro), cui viene applicato un tasso di interesse composto del 10%, costante per tre anni.
Usare una costante CAPITALE_INIZIALE per rappresentare il valore iniziale del capitale.
Il programma deve stampare lo stato del capitale nei tre successivi suoi incrementi.
Per la prima stampa si usi una istruzione come la seguente:

```
printf ("dopo 12 mesi: %g\n", capitale);
```

Provare il programma per diversi valori del capitale iniziale (4 volte). Annotarsi le proprie osservazioni al termine delle (almeno) 4 prove.
E poi passare ad altro. Cioè al prossimo esercizio …

1.3. Reuse (`ourProgram.c`)

Ripetere l'esercizio visto a lezione: `ourProgram.c`, anche con l'uso delle costanti per assegnare i due numeri. Ci sara' quindi, nel programma, un'istruzione come la seguente

```
primoNumero = NUMERO1;
```

e il resto viene di conseguenza, ma se non è chiaro parliamone mentre state facendo l'esercizio …

Attenzione: quando si salva con "Save As" / "Salva col Nome" … badare a scegliere il tipo giusto di file (C source).

1.4. Moquette (`moquette0.c`)

La moquette costa 10.12 euro al mq. Quanto costa la copertura in moquette per una certa stanza che ha lunghezza e larghezza, rispettivamente, mt. 6.9 e 4.5?

Risolvere il problema, producendo alla fine un programma opportuno in C.

Applicare la soluzione più volte, per diverse dimensioni della stanza.

Scrivere una nota su quel che succede in ogni prova: in particolare … per eseguire il programma su una nuova istanza del problema cosa è stato necessario fare? Sarà utile più tardi…

Se si è in difficoltà nello scrivere … confrontate con il file soluzione (ricordando che quel file contiene una possibile soluzione tra varie possibili …).
Poco più sotto c'è un **suggerimento**, in caso di blocco creativo …

Suggerimento
Scrivere e provare un programma che realizza il seguente algoritmo:

0) le dimensioni della stanza sono rappresentate in due costanti, LUNGHEZZA e LARGHEZZA; decidere qui se rappresentare il costo al mq. Della moquette con una variabile o una costante.
1) calcolare l'area della stanza come area = LUNGHEZZA * LARGHEZZA;
2) calcolare il costo della moquette come costo = area per "costo al mq"
3) mostrare in output il costo
4) fine programma

1.5. Moquette da input (moquette1.c)

Stesso problema del punto precedente. Ma ora lunghezza e larghezza devono essere chiesti in input.

yes, scanf()

I formati di conversione da usare nelle chiamate di scanf() sono

- **`%d`** per le espressioni intere, cioè le espressioni la cui valutazione dà un valore intero,
- **`%f`** per i valori float,
- **`%lf`** per i valori double, e
- **`%c`** per i valori char).

Scrivere l'algoritmo (anche all'interno del file .c, sotto forma di un commento iniziale).

Poi realizzare il programma (probabilmente riusando molto del programma precedente).

Provare il programma per diversi input e diversi costi al MQ.

Poi …

Provare il programma anche per diversi costi al MQ.

Scrivere una nota su quel che succede in ogni prova: in particolare … per eseguire il programma su una istanza del problema in cui c'è un diverso costo al MQ, cosa è stato necessario fare?

Sarà utile più tardi…

1.6. Capitalism reloaded (capital1.c)

Provare il programma realizzato nella precedente sezione Capitalism, diverse volte, almeno 4, per diversi valori dell'interesse che viene applicato. Che succede se l'interesse è il 3%, il 12%, o lo 0.34%?

Se le istruzioni del precedente esercizio sono state seguite fedelmente, c'è qualcosa di fastidioso che succede.

Considerare anche le note scritte in precedenza …

COSA SUCCEDE DI SCOMODO????

Elencare qualsiasi cosa che non quadra o che dà una sensazione di insoddisfazione …

Poi si può proseguire con la prossima pagina

In linea teorica la cosa fastidiosa è nel fatto di dover correggere il programma ogni volta che si cambia saggio di interesse (o il costo al mq nell'esercizio sulla moquette …).

Ad esempio per risolvere il problema con un saggio di valore 0.34, dobbiamo riscrivere il programma, mettendo 0.34 (in tutti i punti) dove prima c'era (ad esempio) 12, e lasciando tutto il resto uguale, ottenendo (se siamo ordinati e puliti) un altro file, con nome un po' diverso … ma che in sostanza è il medesimo programma.

Per risolvere questo problema potremmo far diventare il saggio di interesse una variabile, da leggere in input … ma … attenzione, questo lo abbiamo capito, così non è questo che viene chiesto in questo nuovo esercizio …

Questo è l'esercizio.
A partire dal programma scritto precedentemente per "Capitalism", riscrivere il programma in mod che

1) capitale iniziale sia un dato ricevuto in input
2) una costante SAGGIO rappresenti il valore dell'interesse applicato.
3) la stampa dei valori crescenti del capitale avvenga mediante un'istruzione come la seguente

```
printf ("dopo %d mesi: %g\n", numeroMesi, capitale);
```

Scrivere il programma e provarlo diverse volte, per diversi saggi di interesse: il fenomeno fastidioso di prima è attutito? Da quale dei tre cambiamenti che abbiamo fatto sopra (1,2,3) dipende questo miglioramento?

1.7. Non potrebbe mancare …

Scrivere e provare un programma che, ricevendo i coefficienti a, b, c, di un'equazione di secondo grado, ne calcoli le radici reali (assumiamo inizialmente che ci siano solo radici reali).

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()`**, appartenente alla libreria di funzioni matematiche (bisognerà includere nel programma la libreria … il file header … il file .h `math.h`).

Scrivere prima un algoritmo per risolvere il problema … un suggerimento è un po' più sotto;

STOP!!

PRIMA scrivere un tentativo di algoritmo;

POI guardare il suggerimento, ok?

CHE AVEVO DETTO?

0) Ci serviranno variabili per i coefficienti (scegliamo i nomi per bene e indichiamoli qui …); poi una variabile per calcolare il delta quadro (ad esempio `deltaQuadro`), e poi le variabili per le soluzioni (es. `x1`, `x2`)
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 deltaQuadro
5) calcolare il valore della soluzione x_1
6) calcolare il valore della soluzione x_2
7) stampare le soluzioni
8) FINE

Una soluzione possibile è nel file `eq2.c`.

1.8. formati di conversione (in output)

Estensione dell'esercizio precedente.

Tutto come prima ma, tampare un messaggio nel caso in cui non ci sono soluzione reali
Provare a rifletterci, e farlo.

(Questa estensione dell'esercizio potrebbe essere affrontata più facilmente dopo l'esercizio 1.12 (il penultimo di questo documento).

Comunque per questo esercizio una possibile soluzione è in `eq2B.c`

Quando avete pensato un po' alla soluzione, prima di confrontarla con `eq2B.c`, **correggete il programma in `eq3.c`** (è un programma con errori e l'esperienza è utile)

1.9. formati di conversione (in output)

`%d, %f, %g,` possono essere usati per tabulare dati (ad esempio scrivere numeri su un certo numero di cifre, o usare un certo numero di cifre decimali).
Anche consultando il libro, fare pratica con questi aspetti dell'uso della printf.
Un possibile insieme di esperimenti è in `formatiConversione.c`.

1.10. Fahrheneit 2451 (tabellaFahrenheit0.c)

Scrivere un programma che, produce una tabella di conversioni di temperature (Celsius e Farheneit). Ecco un esempio di output.

```
tabella temperature da 18 con passo 4
-----------+------------
 celsius   | farheneit
      18   |     64.400
      22   |     71.600
      26   |     78.800
      30   |     86.000
-----------+------------
FINE
```

Fate venire l'output esattamente così.
Poi se volete fate un altro programma in cui produrrete un output che vi piaccia di più.

La conversione da celsius a fahrenheit è calcolata come segue:

```
fahrenheit = (celsius*9)/5 + 32;
```

Usare 4 costanti, per la temperatura iniziale in celsius, il passo, il rapporto 9/5 e la differenza 32.0

1.11. Fahrheneit 3451, di più (tabellaFahrenheit1.c)

Il programma fa cose simili al precedente … ma

1) I dati sulla temperatura di partenza e sul passo con cui incrementarla vengono forniti da INPUT. Quindi si usa scanf() … e opportune variabili

2) Vengono tabulate anche le temperature Rankine, che si calcolano a partire da Fahrenheit come °R = °F + 459,67

Cercare di ripetere (e anche di migliorare l'output di esempio qui appresso …

```
fornire temperatura d'inizio e passo: 22 5

+-----------+------------+------------+
| celsius   | fahrenheit  |   rankine   |
|      22   |     71.600  |    530.600  |
|      27   |     80.600  |    539.600  |
|      32   |     89.600  |    548.600  |
|      37   |     98.600  |    557.600  |
+-----------+------------+------------+
FINE
```

1.12. reciproco.c

Il problema consiste nel ricevere da input un numero e stampare il relativo reciproco.
Il fatto è che non sempre il reciproco è legittimo (se il numero è zero, 1/0 non si fa).

Scrivere un programma che risolve questo problema, sapendo che la seguente istruzione

```
if (num!=0)
  printf ("il reciproco e` %g\nn", 1.0/num);
else printf("non si puo` calcolare il reciproco!\n");
```

viene eseguita come di seguito:

- **se** il valore contenuto nella variabile num e` diverso da zero,
  - **allora** viene eseguita l'istruzione **`printf ("il reciproco e` %g\nn", 1.0/num);`**
  - **altrimenti** viene eseguita l'istruzione **`printf("non si puo` calcolare il reciproco!\n");`**

1.13. full fledged Equazione di secondo grado

Il programma riceve i coefficienti, calcola le soluzioni e le stampa, con commenti in output che spiegano se le soluzioni sono coincidenti, reali, complesse coniugate …