**LABORATORIO FONDAMENTI DI INFORMATICA 2-3 DICEMBRE 2013**

**Esercizio 1**
Realizzare un programma che chieda una temperatura in gradi Fahrenheit all'utente e la converta in gradi Celsius.
A tal fine implementare una funzione chiamata "`conversione`" che riceve come unico parametro un numero intero che rappresenta la temperatura in gradi Fahrenheit e restituisce l'equivalente in gradi Celsius come numero decimale. Nella funzione "`main`" del programma scrivere il codice per chiedere all'utente una temperatura in gradi Fahrenheit e quindi stampare a schermo la conversione ottenuta chiamando la funzione "`conversione`".
Formula di conversione: °C = (((°F - 32) * 5.0 ) / 9.0 )

```
Temperatura (Fahrenheit)? 100
100 Fahrenheit = 37.777779 Celsius
```

**Esercizio 2**
Realizzare un programma che chieda all'utente una stringa e un carattere e stampi a schermo una sottostringa in base al carattere inserito secondo le modalità indicate in seguito.
A tal fine realizzare la funzione "`estrazione`" che presi come parametri una stringa "`orig`" (array di char – massimo 50) e un carattere "`carat`" estragga, se esiste, una sottostringa di "`orig`" che parte dal carattere "`carat`" e termina con lo stesso carattere "`carat`" oppure arriva in fondo alla stringa. Questa sottostringa (eventualmente vuota) deve essere copiata in un altro array "`risult`" che viene ritornato dalla funzione "`estrazione`". Nella funzione "`main`" del programma occorre chiedere all'utente una stringa e un carattere e stampare a video la sottostringa che la funzione "estrazione" ritorna passandole i parametri.
NB: per chiedere la stringa all'utente usare la funzione `gets(stringa);` poiché `scanf("%s", &stringa);` si ferma al primo spazio.

```
stringa in ingresso? crapa pelata
carattere?  p
sottostringa:  pa p
carattere?  l
sottostringa:  lata
```

**Esercizio 3**
Realizzare un programma che dopo aver generato un vettore di `DIM` = 10 numeri interi casuali ne analizzi il contenuto contando quanti numeri sono < `SOGLIA` = 10; i restanti numeri (quelli >= `SOGLIA`) vanno smistati in base alla parità: i numeri pari in un vettore e i dispari in un altro.
Il programma deve essere suddiviso nelle seguenti funzioni da implementare:
- funzione "`stampavet`". Riceve come parametri un array "`vett`" di numeri interi di dimensione non specificata e un numero intero "`n`" che indica quanti elementi significativi sono presenti nel vettore. La funzione stampa a video tutti i numeri significativi presenti nell'array separandoli con uno spazio e infine stampa un carattere di "a capo" `'\n'`;
- funzione "`inserisci`". Riceve come parametri un numero intero "`valore`", un vettore di numeri interi "`vett`" e un indice "`i`". La funzione deve inserire nel vettore "`vett`" il valore "`valore`" in posizione "`i`"
- funzione "`analisi`". Riceve come primo parametro un array "`vett`" di `DIM` numeri interi e come secondo parametro un numero intero "`sotto_s`" che deve essere "modificato" dalla funzione (attenzione alla modalità di passaggio del parametro). La funzione deve analizzare il vettore passato come parametro, inserire in "`sotto_s`" quanti numeri in questo vettore sono < `SOGLIA` e popolare tramite la funzione "`inserisci`" una `struct` del tipo specificato sotto che va restituita dalla funzione (attenzione: i numeri < `SOGLIA` non vanno inseriti nei vettori di numeri

pari o dispari):

```
struct risultato {
      int n_pari;  //quantità di numeri pari
      int v_pari[DIM]; //vettore di numeri pari
      int n_dispari;  //quantità di numeri dispari
      int v_dispari[DIM]; //vettore di numeri dispari
}
```

- funzione "`main`". In tale funzione occorre dichiarare un array "`vett`" di massimo `DIM` = 10 numeri interi e poi popolarlo tramite il generatore di numeri pseudocasuali

```
vett[i] = rand() % 100;
```

Il vettore deve essere passato come primo parametro alla funzione "`analisi`" per poi stampare a video quanti numeri sono < `SOGLIA` e il contenuto dei vettori "`v_pari`" e "`v_dispari`" della struct (usando la funzione "`stampavet`").

```
Vettore da analizzare: 1 10 5 28 87 58 77 44 32 59
Quantita' di numeri sotto soglia (< 10): 2
Numeri pari sopra soglia (>= 10): 10 28 58 44 32
Numeri dispari sopra soglia (>= 10): 87 77 59
```

**Esercizio 4**

Realizzare un programma che inserisca in un vettore di numeri interi un numero casuale alla volta mantenendo l'univocità dei numeri: se si tenta di inserire un numero già presente nel vettore le sue occorrenze inserite in precedenza vanno "eliminate".

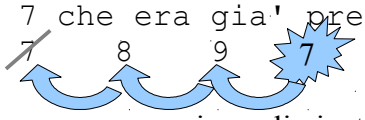
A tal fine implementare la funzione "`memorizza`" che riceve come primo parametro un numero intero "`quantita`" < `DIM` = 100. La funzione genera continuamente numeri pseudocasuali tramite la seguente riga di codice fino a quando ne ha generati un numero pari a "`quantita`":

```
num = rand() % 10;
```

I numeri devono essere memorizzati uno a uno in un vettore di numeri interi "`vett`" (massimo `DIM` 100 elementi) a partire dalla cella 0 in poi.

Per mantenere l'univocità dei numeri, se il numero da inserire compare già nel vettore è necessario eliminare la sua precedente occorrenza sovrascrivendolo, "spostando indietro" ovvero facendo "shift" di una cella tutti gli eventuali numeri che lo seguono. A ogni passo occorre stampare il contenuto del vettore tramite la funzione "`stampavet`" realizzata nell'esercizio 3.

```
Inserisco 7 che era gia' presente!
3    5    7    8    9    7    ===>  3    5    8    9    7
```



Ogni volta che un numero viene eliminato, occorre aumentare una variabile intera "`eliminati`" che conta quanti numeri sono stati eliminati. Tale variabile viene restituita dalla funzione "`memorizza`".

Nella funzione "`main`" occorre chiedere all'utente il valore della variabile "`quantita`" e passarlo come parametro alla funzione "`memorizza`" e quindi stampare a video quanti numeri sono stati eliminati.

```
…
4 8 5 2 3 7 9 0
Inserisco 2 che era gia' presente!
4 8 5 3 7 9 0 2
…
Ho eliminato 12 numeri
```

**Esercizio 5 (Opzionale - variante dell'esercizio 4)**
Realizzare un programma simile al precedente con le seguenti varianti:
- la funzione "`memorizza`" riceve come parametro un numero intero denominato "`sommafinale`"
- l'elaborazione termina quando la somma dei numeri ricevuti è > del parametro "`sommafinale`"
- invece che il mantenimento dell'univocità si richiede il mantenimento dell'ordinamento degli elementi del vettore (dal più piccolo al più grande). Per mantenere l'ordinamento occorre "fare spazio" per il numero da inserire "spostando avanti" di una cella tutti gli eventuali numeri maggiori di esso
- invece che ritornare il numero di elementi eliminati, la funzione "`memorizza`" ritorna il numero di elementi inseriti

```
…
0 3 7 8 9
Inserisco 2 in posizione 1; somma attuale = 29
0 2 3 7 8 9
Inserisco 4 in posizione 3; somma attuale = 33
0 2 3 4 7 8 9
Inserisco 8 in posizione 6; somma attuale = 41
0 2 3 4 7 8 8 9
…
Ho inserito 22 numeri
```