Programmazione
II Compitino (Preparazione)

16 gennaio 2014

Cognome **Salamazza** Nome **Giovanni**
Matricola **773992**

1. Scrivete un programma che chieda ripetutamente all'utente di inserire degli interi, e che termini quando l'utente inserisce il valore 0; potete assumere che l'utente non inserisca più di 100 interi. Al termine dell'inserimento, il programma deve stampare l'elenco dei valori inseriti saltando le duplicazioni (cioè, se un numero compare nell'input più di una volta, viene ristampato solo la prima volta). Ad esempio, se l'utente inserisce *61 300 506 61 601 300 503 0* il programma deve stampare *61 300 506 601 503*.

```
#include <stdio.h>

int main() {
 int x[100], n=0, i, j;
 do
  scanf("%d",&x[n]);
 while (x[n++]!=0);
 n--;
 for (i=0;i<n;i++) {
  for (j=0;j<i;j++)
   if (x[i]==x[j]) break;
  if (j==i)
   printf("%d\n",x[i]);
 }
 return 0;
}
```

2. Scrivete una funzione che, data una stringa e un carattere, restituisca il numero di volte che il carattere compare nella stringa *non seguito da se stesso*. Ad esempio, se la stringa fosse "la mamma ha fatto la manza" e il carattere fosse 'm', la funzione dovrebbe restituire 3.

```
int f(char *s, char c) {
 int x=0;
 while (*s) {
  if (*s==c && *(s+1)!=c)
   x++;
  s++;
 }
 return x;
}
```

3. Un'azienda commercializza vari prodotti; ogni prodotto è caratterizzato da:

   - un codice merceologico, di 8 caratteri;
   - un nome (una stringa di al più 100 caratteri);
   - un prezzo di vendita (in euro);
   - la data in cui, per la prima volta, quel prodotto è stato messo in vendita;
   - il numero di unità di quel prodotto disponibili in magazzino.

   Definite una struttura C, di nome `prodotto`, che rappresenti un prodotto.

```
typedef struct {
 int g, m, a;
} data;
typedef struct {
 char codice[8];
 char nome[101];
 double euro;
 data commercializzazione;
 int numero;
} prodotto;
```

   Ora scrivete una funzione che, dato un array di prodotti e un numero n (il numero di elementi dell'array effettivamente utilizzati), calcoli per ogni prodotto il valore di magazzino (cioè, la quantità presente in magazzino moltiplicata per il prezzo unitario) e restituisca il massimo di tali valori.

```
double maxval(prodotto a[], int n) {
 int i;
 double cand;
 cand=a[0].euro*a[0].numero;
 for (i=1;i<n;i++)
  if (a[i].euro*a[i].numero>cand)
   cand=a[i].euro*a[i].numero;
 return cand;
}
```

4. Scrivete un programma che, ricevendo sulla riga di comando due numeri interi, diciamo a e b, produca e stampi un valore pseudo-casuale nell'insieme $\{a, a+1, \ldots, b\}$. Ad esempio, se supponete che il programma si chiami `acaso`, un tipico esempio di esecuzione potrebbe essere

   [gnagna]>./acaso 5 15
   `6`
   [gnagna]>

   Suggerimento. Vi ricordiamo che la funzione `atoi` (dichiarata in `stdlib.h`), data una stringa contenente una sequenza di cifre, restituisce il corrispondente numero intero. Ad esempio, `atoi("452")` restituisce il numero 452. Ricordiamo, inoltre, che sempre in `stdlib.h` c'è una funzione di nome `rand()` che restituisce un valore pseudo-casuale da 0 a `RAND_MAX`.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
 int from, to;
 from=atoi(argv[1]);
 to=atoi(argv[2]);
 printf("%d\n",from+rand()%(to-from+1));
 return 0;
}
```

5. Scrivete una funzione con tre argomenti: il primo è un intero (x) il secondo e il terzo sono un puntatori a intero (y e z). La funzione deve confrontare l'intero contenuto in x (diciamo a) con quelli cui si riferiscono y e z (diciamo b e c, rispettivamente):

   - se $b \leq a \leq c$, la funzione non deve fare nulla;
   - se $b \leq c \leq a$, la funzione deve mettere a nell'intero cui si riferisce z;
   - se $a \leq b \leq c$, la funzione deve mettere a nell'intero cui si riferisce y;
   - in tutti gli altri casi, deve incrementare sia il contenuto di y che il contenuto di z.

```
void f(int x, int *y, int *z) {
 if (*y<=x && x<=*z);
 else if (*y<=*z && *z<=x) *z=x;
 else if (x<=*y && *y<=*z) *y=x;
 else {
  (*y)++;
  (*z)++;
 }
}
```

6. Supponete di avere dichiarato un tipo `persona` come segue:

```
typedef struct {
 char nome[ 500 ], cognome[ 500 ];
 char sesso; /* vale 'm' oppure 'f' */
 int anno_di_nascita;
} persona;
```

Scrivete una funzione che, dato l'anno corrente, un array di persone e la sua lunghezza, stabilisca l'età media di maschi e femmine e memorizzi tali due valori in due `double` di cui la funzione riceve come argomenti gli indirizzi.

```
void eta_media(int anno, persona x[], int n, double *media_m, double *media_f) {
        int i, nm, nf;

        *media_m =*media_f=0.0;
        nm=nf=0;
        for (i=0; i<n; i++)
                if (x[i].sesso=='m') {
                        (*media_m)+=anno-x[i].anno_di_nascita;
                        nm++;
                }
                else {
                        (*media_f)+=anno-x[i].anno_di_nascita;
                        nf++;
                }
        *media_m/=nm;
        *media_f/=nf;
}
```

7. Data la definizione di `persona` (vedi sopra), scrivete una funzione che riceve un array di persone e la sua lunghezza, e restituisce quante persone hanno un dato cognome (passato, anch'esso, come argomento).

```
int quante(persona x[], int n, char *cogn) {
 int i, c;

 for (i=c=0; i<n; i++)
  if (strcmp(x[i].cognome,cogn)==0)
   c++;
 return c;
}
```