Laboratorio di Calcolo per Fisici, Settima esercitazione

Canale D-K, Docente: Dott.ssa Livia Soffi,
Esercitatori: Prof. S. Rahatlou e Dott.ssa Giulia D'Imperio

Lo scopo della settima esercitazione di laboratorio è di esercitarsi con l'utilizzo delle **stringhe**. Analizzeremo dei crittogrammi, cioe' delle parole crittografate, "offuscate" in modo da non essere leggibili per chiunque non fosse autorizzato.

Il **cifrario di Cesare** è uno dei più antichi algoritmi crittografici di cui si abbia traccia storica. È un cifrario a sostituzione monoalfabetica, in cui ogni lettera del testo in chiaro è sostituita, nel testo cifrato, dalla lettera che si trova tre posizioni dopo nell'alfabeto. All'epoca di Giulio Cesare era un algoritmo efficace e veniva usato per inviare le sue corrispondenze.
Nell'alfabeto latino esteso, che ha 26 caratteri, la corrispondenza tra carattere in chiaro e carattere cifrato è la seguente:

Carattere in chiaro	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Carattere criptato	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

► **Prima parte** Scrivete un programma **cesare.c** per decriptare una parola cifrata con cifrario di Cesare. Il programma prendera' in input una stringa (la parola cifrata) e stampera' su schermo da parola in chiaro. Si utilizzino nel seguito solo le lettere maiuscole.

- Stampare un breve messaggio (max 1 riga) che spieghi all'utente lo scopo del programma.
- Chiedere all'utente di inserire una parola cifrata e acquisirla sotto forma di stringa. Salvare la stringa in un array di char di lunghezza 20.
- Stampare la parola cifrata appena acquisita.
- Contare e stampare la lunghezza della parola inserita. La lunghezza si intende pari al numero di caratteri effettivi contenuti nella stringa, escluso il carattere di terminazione '\0'.
- Tramite un costrutto *do-while* ripetere i passi precedenti se la lunghezza della parola e' maggiore di 12. Proseguire poi con le istruzioni seguenti.

- Contare e stampare quante vocali e quante consonanti contiene la stringa inserita. Si consideri "Y" come una vocale e "X", "W", "K" e "J" come consonanti.

  Suggerimento: L'operatore di confronto "==" si può utilizzare anche con i char, ad esempio, data una variabile char di nome a, si può scrivere: if(a=='Z'){...blocco di istruzioni...}

- Decriptare la parola cifrata. Per farlo dichiarare una nuova stringa di lunghezza 20. Per ciascuno dei caratteri che compongono la parola cifrata, assegnare al carattere della nuova stringa la lettera che si trova tre posizioni prima la lettera di partenza nell'alfabeto latino esteso.

  Suggerimento: Se così facendo si finisce prima della 'A', la sequenza riparte dalla Z. Cioè "BDB" viene decifrato in "YAY".

- Stampare la parola in chiaro ottenuta.

Utilizzare il programma scritto per decriptare le seguenti parole:

- FHVDUH
- RUR
- ILVLFD
- LQJHJQL
- FRPSXWHU
- NDBDN

- Il descrittore per le stringhe e' **%s**. Si ricorda che nell'utilizzo di scanf(..) per le stringhe il simbolo "&" non va inserito:

  char stringa[10];
  scanf("%s",stringa);

- Il carattere '\0' indica la fine della stringa.

Laboratorio di Calcolo per Fisici, Settima esercitazione - Addendum

Canale D-K, Docente: Dott.ssa Livia Soffi,
Esercitatori: Prof. S. Rahatlou e Dott.ssa Giulia D'Imperio

► Seonda parte

Utilizzare il programma **cesare.c** per decriptare le seguenti parole. Le stesse, lette una dopo l'altra nell'ordine riportato, vi diranno cosa vi viene chiesto di fare in questa seconda parte.

- YHULILFDUH
- VH
- OD
- SDUROD
- GHFULSWDWD
- ULVXOWD
- SDOLQGURPD

PROGRAMMA IN C ESEGUIBILE.

```
#include<stdio.h>
#include<math.h>
#define LEN 20
#define MAX 13

int main() {
  char word[LEN]={'\0'};
  char decript[LEN]={'\0'};
  int lenght=0, vocali=0, consonanti=0;
  int vero=0;

  printf("Benvenuto!\nQuesto programma prende in input una parola cifrata e la decifra
        attraverso il 'Cifrario di Cesare'.\nInserisci una parola da decifrare di massimo 12
        lettere maiuscole.\npassword: ");


 do {
  for(int i=0; i<LEN; i++) {
    word[i]='\0';
  }                         //azzero l'array per ogni errore
   scanf("%s", word);
 if(word[MAX]!='\0') printf("Errore. Inserisci una parola di lunghezza valida.\npassword: ");
 } while(word[MAX]!='\0');     //acquisisco la parola e controllo se l'inserimento è corretto


  printf("Hai inserito la parola: '%s'.\n", word);

  for(int j=0; word[j]!='\0'; j++) {
    lenght++;                 //conto la lunghezza della parola

   if(word[j]=='A' || word[j]=='E' || word[j]=='I' || word[j]=='O' || word[j]=='U' || word[j]=='Y')
        vocali++;
   else consonanti++;
  }                         //sfrutto lo stesso ciclo for per contare vocali e consonanti della parola


  printf("La lunghezza della parola inserita è %d, ed è formata da %d vocali e %d
        consonanti.\n", lenght, vocali, consonanti);


  for(int k=0; word[k]!='\0'; k++) {
    decript[k] = 0;         //azzero ogni carattere prima di operarci sopra

    if((int)word[k]>=68 && (int)word[k]<=90) {
     decript[k] = word[k]-3;
   } else {
   decript[k] = word[k]+23;    //eseguo l'algoritmo in modo che fra A e C non ottengo
        caratteri che non mi servono
    }
  }

// cerca di aggiungere sempre a mano il carattere di fine stringa '\0'

  printf("La parola decriptata è '%s'", decript);
```

```
/* ADDENDUM:

   "VERIFICARE SE LA PAROLA DECRIPTATA RISULTA PALINDROMA" */

        for(int t=0; decript[t]!='\0'; t++) {       //controllo per ogni carattere se primo=ultimo,
                                                    secondo=penultimo, ecc...
         if(decript[t]==decript[lenght-(1+t)]) {
        vero++;
        }
       }

       if(vero==lenght) printf(" ed è palindroma.\n\n");
       else printf(" e non è palindroma.\n\n");

        return 0;
      }
```