



# Programmazione I

Il Linguaggio C

Esercizi

Daniel Riccio

Università di Napoli, Federico II

03 novembre 2021



# Esercizio – Vocali e consonanti



Scrivere un programma in linguaggio C che legga una frase introdotta da tastiera.

La frase è terminata dall'introduzione del carattere di invio.

La frase contiene sia caratteri maiuscoli che caratteri minuscoli, e complessivamente al più 100 caratteri.

Il programma dovrà stampare su schermo le seguenti informazioni:

- 1) per ognuna delle lettere dell'alfabeto, il numero di volte che la lettera compare nella stringa
- 2) il numero di consonanti presenti nella stringa
- 3) il numero di vocali presenti nella stringa

# Esercizio – Vocali e consonanti



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXDIM 100
#define NUMLETTERE 26
```

```
int main(){
char frase[MAXDIM +1] ;
int lung_stringa ;
int vocali, consonanti ;
int contatori[NUMLETTERE];
int posizione_alfabeto ;
int i ;
```

```
/* dimensione massima stringa di caratteri */
/* numero di lettere dell'alfabeto */
```

```
/* stringa di caratteri inserita */
/* lunghezza della stringa inserita */
/* contatori numero di vocali e di consonanti */
/* memorizza il numero di occorrenze per ogni lettera */
/* posizione nell'alfabeto di una lettera */
/* indice dei cicli */
```

Leggi la frase inserita da tastiera

```
printf ("Inserisci una frase di al massimo %d caratteri: ", MAXDIM) ;
gets(frase) ;
```

# Esercizio – Vocali e consonanti



Calcola la lunghezza della frase

```
lung_stringa = strlen(frase);
```

Stampa la frase inserita

```
printf("La frase inserita e': ");
```

```
puts(frase);
```

```
printf("La frase contiene %d caratteri (inclusi gli spazi)\n", lung_stringa);
```

Azzera il vettore dei contatori. Ogni cella di questo vettore è associata a una lettera dell'alfabeto. La cella 0 alla lettera A, la cella 1 alla B e così via.

```
for ( i=0; i<NUMLETTERE; i++ )  
    contatori[i] = 0 ;
```

Analizza la frase lettera per lettera e aggiorna il vettore dei contatori

```
for ( i=0; i<lung_stringa; i++ ){  
    if ( frase[i] >= 'A' && frase[i] <= 'Z' ){  
        posizione_alfabeto = frase[i] - 'A' ;  
        contatori[posizione_alfabeto] ++ ;  
    }  
}
```

# Esercizio – Vocali e consonanti



Analizza la frase lettera per lettera e aggiorna il vettore dei contatori

```
else{  
    if ( frase[i] >= 'a' && frase[i] <= 'z' ){  
        posizione_alfabeto = frase[i] - 'a' ;  
        contatori[posizione_alfabeto]++ ;  
    }  
}  
}
```

Stampa i contatori delle varie lettere

```
for ( i=0; i<NUMLETTERE; i=i+1 )  
    printf ("La lettera %c compare %d volte \n", 'A'+i , contatori[i]) ;
```

Calcola il numero di vocali. Somma il numero di occorrenze presenti nel vettore «contatori» nelle celle associate alle lettere A, E, I, O, U, Y

```
vocali = contatori['A'-'A'] + contatori['E'-'A'] + contatori['I'-'A'] + contatori['O'-'A'] + contatori['U'-'A'] + contatori['Y'-'A'] ;
```

# Esercizio – Vocali e consonanti



Calcola il numero di consonanti. Il numero di consonanti si ottiene sottraendo il numero complessivo di vocali dal numero complessivo di occorrenze di tutte le lettere

```
consonanti = 0 ;
```

```
for ( i=0; i<NUMLETTERE; i=i+1 )
```

```
    consonanti = consonanti + contatori[i] ;
```

```
consonanti = consonanti - vocali ;
```

Stampa il numero di vocali e consonanti

```
printf ("Il numero di vocali e': %d\n", vocali) ;
```

```
printf ("Il numero di consonanti e': %d\n", consonanti) ;
```

```
exit(0) ;
```

```
}
```

# Esercizio – Statistiche sul testo



Un utente inserisce una serie di frasi da tastiera, su più righe

L'inserimento termina quando l'utente inserisce la parola FINE su una riga da sola

Il programma deve determinare:

- 1) Quante righe sono state inserite dall'utente
- 2) Quanti caratteri sono stati inseriti
- 3) Quanti caratteri alfanumerici sono stati inseriti
- 4) Quante parole sono state inserite

```
Prompt dei comandi

Testo: Nel mezzo del cammin di nostra vita
Testo: mi ritrovai per una selva oscura
Testo: che la diritta via era smarrita.
Testo: FINE
L'utente ha inserito 3 righe
L'utente ha inserito 99 caratteri
L'utente ha inserito 82 caratteri alfanumerici
L'utente ha inserito 19 parole
```

# Esercizio – Statistiche sul testo



L'inserimento termina quando l'utente inserisce la parola FINE su una riga da sola

```
#define MAXRIGHE 2000
#define LUN 80
char testo[MAXRIGHE][LUN] ;
int Nrighe ; /* righe inserite */
char riga[LUN*10] ;
int i, j ;
int caratteri, caralfa, parole ;
```

Quante righe sono state inserite dall'utente

```
Nrighe = 0 ;
do {
    printf("Testo: ");
    gets(riga) ;
    if( strcmp(riga, "FINE")!=0 ){
        /*copia riga in testo[Nrighe] ;*/
        strcpy( testo[Nrighe] , riga ) ;
        Nrighe++ ;
    }
} while( strcmp(riga, "FINE")!=0 ) ;

printf("L'utente ha inserito %d righe\n", Nrighe);
```



# Esercizio – Statistiche sul testo



## Quanti caratteri sono stati inseriti

```
caratteri = 0 ;  
for(i=0; i<Nrighe; i++)  
    caratteri = caratteri + strlen(testo[i]) ;  
  
printf("L'utente ha inserito %d caratteri\n", caratteri) ;
```

## Quanti caratteri alfanumerici sono stati inseriti

```
caralfa = 0 ;  
for(i=0; i<Nrighe; i++){  
    for(j=0; testo[i][j]!='\0'; j++){  
        if( isalnum(testo[i][j]) )  
            caralfa++ ;  
    }  
}  
  
printf("L'utente ha inserito %d caratteri alfanumerici\n", caralfa) ;
```

# Esercizio – Statistiche sul testo



**Quante parole sono state inserite**

```
parole = 0 ;  
for(j=0; j<Nrighe; j++){  
    for(i=0; testo[j][i]!=0; i++){  
        if( isalpha(testo[j][i]) && (i==0 || !isalpha(testo[j][i-1]))){  
            parole ++ ;  
        }  
    }  
}  
printf("L'utente ha inserito %d parole\n", parole) ;
```

# Esercizio – Concorso di intelligenza



In un concorso di intelligenza, **N** giudici esprimono il loro giudizio su **K** candidati.

Il giudizio è un valore numerico tra **0** e **5**

Si scriva un programma in linguaggio C per determinare il candidato più intelligente, ed il giudice più severo

# Esercizio – Concorso di intelligenza



## Azioni:

- 1) Acquisisce il numero di candidati e di giudici
- 2) Acquisisce il voto di ciascun giudice per ciascun candidato
- 3) Somma tutti i voti ottenuti da un candidato
- 4) Somma tutti i voti dati da un giudice
- 5) Il candidato migliore è quello che ha ottenuto il punteggio massimo
- 6) Il giudice più severo è quello che ha dato i voti più bassi (somma)

# Esercizio – Concorso di intelligenza



## Esecuzione

```
✓ ↗ 📋
Quanti candidati ci sono? 3
Quanti giudici ci sono? 4
Immettere i giudizi per il candidato 1
Giudice 1, cosa pensi del candidato 1? 3
Giudice 2, cosa pensi del candidato 1? 2
Giudice 3, cosa pensi del candidato 1? 1
Giudice 4, cosa pensi del candidato 1? 5
Immettere i giudizi per il candidato 2
Giudice 1, cosa pensi del candidato 2? 1
Giudice 2, cosa pensi del candidato 2? 4
Giudice 3, cosa pensi del candidato 2? 3
Giudice 4, cosa pensi del candidato 2? 2
Immettere i giudizi per il candidato 3
Giudice 1, cosa pensi del candidato 3? 4
Giudice 2, cosa pensi del candidato 3? 2
Giudice 3, cosa pensi del candidato 3? 3
Giudice 4, cosa pensi del candidato 3? 5
Il vincitore e' il candidato numero 3
Il giudice piu' severo e' il numero 3
```

# Esercizio – Concorso di intelligenza



		Giudici				
		0	1	2	j	N-1
Candidati	0					
	1					
	2					
	i					
	K-1					

Inseriamo i voti in una matrice Candidati (righe)  $\times$  Giudici (colonne) di dimensione **K**  $\times$  **N**

Il totale del **candidato i-esimo** è rappresentato dalla somma delle colonne alla riga **i-1**

Il totale dei voti assegnati dal **giudice j-esimo** è dato dalla somma delle righe alla colonna **j-1**

**Candidato migliore**  $\rightarrow$  massimo delle K somme sulle colonne

**Giudice più severo**  $\rightarrow$  minimo delle N somme sulle righe

# Esercizio – Concorso di intelligenza



```
#include <stdio.h>
#include <stdlib.h>
#define MAXK 100 /* max n. candidati */
#define MAXN 10 /* max n. giudici */

int main()
{
    int voti[MAXK][MAXN] ;
    int tot[MAXK]={0}; /* somma dei voti per ogni candidato */
    int totg[MAXN]={0}; /* somma dei voti di ogni giudice */
    int K, N ;
    int i, j ;
    int min, max, posmin, posmax ;
```

# Esercizio – Concorso di intelligenza



```
printf("Quanti candidati ci sono? ");
```

```
scanf("%d", &K) ;
```

```
printf("Quanti giudici ci sono? ");
```

```
scanf("%d", &N) ;
```

```
for (i=0; i<K; i++){
```

```
    printf("Immettere i giudizi per il candidato %d\n", i+1);
```

```
    for (j=0; j<N; j++){
```

```
        printf("Giudice %d, cosa pensi del candidato %d? ", j+1,i+1 );
```

```
        scanf("%d", & voti[i][j] ) ;
```

```
    }
```

```
}
```



# Esercizio – Concorso di intelligenza



```
for (i=0; i<K; i++){  
    for (j=0; j<N; j++){  
        tot[i] = tot[i] + voti[i][j] ;  
        totg[j] = totg[j] + voti[i][j] ;  
    }  
}
```

Calcola:

Il voto totale del candidato i-esimo

(somma delle colonne alla riga i-1)

La somma dei voti assegnati da un giudice

(somma delle righe sulla colonna j-1)

```
max = tot[0] ;  
posmax = 0 ;  
  
for (i=1; i<K; i++){  
    if (tot[i]>max){  
        max = tot[i];  
        posmax = i ;  
    }  
}
```

Calcola:

Il candidato con voto massimo

(massimo del vettore **tot**)

```
printf("Il vincitore è il candidato numero %d\n", posmax+1);
```

# Esercizio – Concorso di intelligenza



```
min = totg[0] ;  
posmin = 0 ;
```

```
for (i=1; i<N; i++){  
    if (totg[i]<min){  
        min = totg[i];  
        posmin = i ;  
    }  
}
```

Calcola:  
Il giudice più severo  
(minimo del vettore **totg**)

```
printf("Il giudice più severo è il numero %d\n", posmin+1);
```

```
return 0 ;
```

```
}
```

# Esercizio – Concorso di intelligenza



## Esecuzione

```
✓ ↗ 📋
Quanti candidati ci sono? 3
Quanti giudici ci sono? 4
Immettere i giudizi per il candidato 1
Giudice 1, cosa pensi del candidato 1? 3
Giudice 2, cosa pensi del candidato 1? 2
Giudice 3, cosa pensi del candidato 1? 1
Giudice 4, cosa pensi del candidato 1? 5
Immettere i giudizi per il candidato 2
Giudice 1, cosa pensi del candidato 2? 1
Giudice 2, cosa pensi del candidato 2? 4
Giudice 3, cosa pensi del candidato 2? 3
Giudice 4, cosa pensi del candidato 2? 2
Immettere i giudizi per il candidato 3
Giudice 1, cosa pensi del candidato 3? 4
Giudice 2, cosa pensi del candidato 3? 2
Giudice 3, cosa pensi del candidato 3? 3
Giudice 4, cosa pensi del candidato 3? 5
Il vincitore e' il candidato numero 3
Il giudice piu' severo e' il numero 3
```

# Esercizi



In crittografia 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 un certo numero di posizioni dopo nell'alfabeto.

Scrivere un programma che prenda in input una **stringa** ed un valore intero **N**. Il programma esegue le seguenti operazioni:

- 1) Cifra il testo sostituendo ogni lettera in posizione **i** nell'alfabeto con la lettera in posizione  **$i+N$**
- 2) Stampa la stringa cifrata
- 3) Decifra la stringa codificata con la sostituzione inversa
- 4) Stampa la stringa in chiaro



# Esercizio – Cifrario di Cesare



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

```
#include <string.h>
```

```
#include <ctype.h>
```

```
#define MAX_LEN 100
```

```
#define MAX_ROW 50
```

```
typedef char Stringa[MAX_LEN];
```

```
Stringa Testo[MAX_ROW];
```

```
Stringa Cifrato[MAX_ROW];
```

```
Stringa Decifrato[MAX_ROW];
```

# Esercizio – Cifrario di Cesare



```
int main (){  
    int nrow = 0;  
    int N;  
    int i,j;  
    char c;  
    Stringa riga;  
  
    do {  
        printf("Inserisci un valore intero N [1,25]: ");  
        scanf("%d", &N);  
    }while(N<1 || N>25);
```

Richiediamo un valore per N

Cicliamo fin quando il valore  
di N non è compreso  
nell'intervallo

# Esercizio – Cifrario di Cesare



```
printf ("Inserisci il testo\n\"FINE\" per terminare\n");
```

```
do  
{  
    gets(riga);  
    if (strcmp(riga, "FINE") != 0){  
        strcpy(Testo[nrow], riga);  
        nrow++;  
    }  
}  
while (strcmp(riga, "FINE") != 0);
```

All'interno di un ciclo prendiamo in input una riga.

Se il contenuto di riga è diverso da "FINE", il contenuto di riga viene copiato in Testo

# Esercizio – Cifrario di Cesare



```
// Cifriamo il testo
printf("Cifratura del testo...\n");
for(i=0; i<nrow; i++){ // cicliamo su ogni riga
    j=0;
    while(Testo[i][j] != '\0'){ // scandiamo la riga fino al terminatore '\0'
        c = toupper(Testo[i][j]); // convertiamo il carattere in maiuscolo
        if(c>='A' && c<='Z') // se si tratta di una lettera
            Cifrato[i][j] = (c - 'A' + N)%26 + 'A'; // sostituiamo il carattere
        else
            Cifrato[i][j] = c; // copiamo il carattere senza modificarlo

        ++j;
    }
    Cifrato[i][j] = '\0'; // terminiamo la riga
}
```



# Esercizio – Cifrario di Cesare



```
// Decifriamo il testo
printf("\nDecifratura del testo...\n");
for(i=0; i<nrow; i++){ // cicliamo su ogni riga
    j=0;
    while(Cifrato[i][j] != '\0'){ // scandiamo la riga fino al terminatore '\0'
        c = Cifrato[i][j];
        if(c>='A' && c<='Z'){ // se si tratta di una lettera
            c = Cifrato[i][j] - N; // sostituiamo il carattere
            if(c < 'A') // se sottraendo N il carattere ottenuto precede la 'A' sommiamo 26
                Decifrato[i][j] = c + 26;
            else
                Decifrato[i][j] = c;
        }
        else
            Decifrato[i][j] = c;

        ++j;
    }
    Decifrato[i][j] = '\0';
}
```

# Esercizio – Cifrario di Cesare

```
// stampiamo il testo Cifrato
printf("\nTesto cifrato:\n");
for(i=0; i<nrow; i++)
    printf("%s", Cifrato[i]);

// stampiamo il testo Decifrato
printf("\nTesto decifrato:\n");
for(i=0; i<nrow; i++)
    printf("%s", Decifrato[i]);
}
```

