4 Stringhe e Matrici

Stringhe Le stringhe di caratteri sono gestite in C come dei vettori di `char` con alla fine un "tappo" dato dal carattere `'\0'`. E' possibile acquisire un'intera stringa di caratteri in una sola istruzione grazie all'istruzione `scanf('%s', variabile);` oppure `gets(variabile)`. Entrambi devono essere seguiti dall'istruzione `fflush(stdin);` che serve ad evitare errori nella successiva acquisizione di stringhe. Mentre l'istruzione `scanf('%s', variabile);` ferma la sua acquisizione al primo spazio, l'istruzione `gets(variabile)` inserisce nella variable tutto l'input fino al primo invio.

Poichè le stringhe necessitano di un tappo, ogni volta che ne dichiaro una devo aggiungere un elemento per il tappo, ossia se voglio una stringa lunga `N_MAX` dovrò dichiarare un array di `N_MAX+1` elementi.

Esiste una libreria di C che gestisce le stringhe `string.h` essa ci permette di:

- ottenere la lunghezza di una stringa con la funzione `strlen(stringa);`
- confrontare due stringhe `strcmp(stringa1, stringa2)` e restituisce zero se sono uguali, un numero negativo se `stringa1` viene prima in ordine lessicografico di `stringa2` e un numero positivo se `stringa2` viene prima in ordine lessicografico di `stringa1` (se i caratteri sono tutti maiuscoli o tutti minuscoli);
- copiare una stringa `stringa2` in un'altra stringa `stringa1` con la funzione `strcpy(stringa1, stringa2);`
- concatenare alla stringa `stringa1` la stringa `stringa2` con la funzione `strcat(stringa1,stringa2);`

Matrici Le matrici in C vengono definite come vettori di vettori. Ad esempio per dichiarare una matrice bidimensionale dovremo scrivere:

```
#define DIM1 n_righe
#define DIM2 n_colonne

void main(){

tipo_variabile nome_variabile[DIM1][DIM2];

istruzioni;
```

```
}
```

dove `n_righe` e `n_colonne` sono degli interi che specificano il numero di righe e colonne della matrice. Nel caso di dimensioni superiori a 2, dovremo mettere tante parentesi quadre e specificare le grandezze massime per ogni dimensione.

4.1 Esercizi

Esercizio 4.1

Date due stringhe immesse dall'utente, implementare la stessa funzionalità della `strcat()` senza utilizzare la funzione stessa e senza utilizzare la `strlen()`. Considerare delle stringhe in ingresso di al massimo 20 caratteri.

Esercizio 4.2

1. Scrivere un programma che determini se una frase acquisita da tastiera è palindroma. Controllare se sono palindrome le seguenti frasi (senza spazi e maiuscole):
   - Eri un nano non annuire
   - Ad una vera pia donna dei simili fili misi e annodai: pareva nuda
   - O mordo tua nuora, o ari un autodromo
   - Occorre portar aratro per Rocco
2. (Bonus) Modificare il programma precedente per considerare stringhe in input contenenti anche caratteri speciali e spazi.

Esercizio 4.3

Implementare la seguente variante del cifrario di Cesare[1]. Dopo aver acquisito una messaggio di massimo 160 caratteri (alfabetici minuscoli), il programma dovrà chiedere all'utente una chiave (numero intero K, $1 < K < 25$).

Il programma stamperà il messaggio cifrato, ottenuto traslando ogni lettera di K posizioni in avanti (dove K è la chiave).

[1] `http://it.wikipedia.org/wiki/Cifrario_di_Cesare`

Dopo aver stampato il messaggio cifrato, il programma chiede all'utente di inserire un messaggio (che si assume sia stato cifrato con lo stesso algoritmo di cui sopra). Tale messaggio sarà dunque decifrato dal programma, il quale dovrà svolgere l'operazione inversa.

Infine, il messaggio decifrato sarà stampato a video.

Suggerimento: i caratteri minuscoli dalla a alla z corrispondono nella tabella ASCII alle cifre intere dalla 97 alla 122.

Esercizio 4.4

1. Scrivere un programma che converta una stringa in alfabeto farfallino. Nell'alfabeto farfallino, ogni vocale è raddoppiata, e tra le due vocali è inserita la lettera 'f'. Ad esempio, "ciao" diventa "cifiafaofo".

2. (bonus) Scrivere il medesimo programma utilizzando soltanto una variabile per memorizzare la stringa tradotta (ovvero senza copiare la traduzione in una nuova variabile).

Esercizio 4.5

Scrivere un programma che considera solo le lettere maiuscole e minuscole di una stringa letta da tastiera (di dimensione massima 256 elementi). Il programma dovrà calcolare le occorrenze di ogni carattere nella stringa. Le occorrenze saranno poi stampate a video a video come istogramma con gli asterischi nel seguente modo:

```
str = Ciao Come Stai? Non c'c' male GRAZIE! Mi trovo molto bene in questa citta'.

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 |
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 |
```

Esercizio 4.6

Scrivere un programma che determini se due parole acquisite da tastiera sono una l'anagramma dell'altra.

Esercizio 4.7

Scrivere un programma che chieda all'utente la dimensione effettiva di una matrice quadrata, che dovrà poi essere acquisita dal programma.

Successivamente, il programma dovrà controllare se la matrice è diagonale (i.e., nessuno zero sulla diagonale principale e solo zeri fuori da essa).

Estendere il programma per controllare anche se la matrice inserita è simmetrica.

Esercizio 4.8

Scrivere un programma che inizializzi una matrice 10×10 con i valori della tavola pitagorica.

Dopodiché, il programma dovrà stampare la matrice seguendo un percorso a spirale in senso orario, a partire dalla cella $m[0][0]$.

Per semplicità si può assumere la matrice quadrata.

Esercizio 4.9

(TdE Febbraio 2012, variante) Data m una matrice di dimensione $n \times n$ (costante simbolica) di numeri interi nell'intervallo $[0, 9]$:

1. si scriva un frammento di programma in linguaggio C (con le relative dichiarazioni di variabili necessarie al corretto funzionamento), che trovi il numero più frequente (si ipotizzi che tale numero sia unico).

2. si scriva un frammento che memorizzi in un vettore e stampi a schermo tutti i numeri presenti nella matrice con valore minore del valore più frequente e successivamente memorizzi in un altro vettore tutti quelli con valore maggiore di quello più frequente e li stampi;

3. aggiungere al programma di cui sopra un frammento di codice che legga tutti e soli i valori memorizzati nel secondo dei due vettori e stampi a video se i valori che vi sono contenuti sono o meno monotoni crescenti, ovvero se per ogni i si ha $a_i \leq a_{i+1}$.

Esercizio 4.10

Scrivere un programma che acquisisca una matrice di interi positivi con `DIM_R` righe e `DIM_C` colonne. Il prgramma deve stampare l'indice della riga della matrice con somma massima.

Soluzioni

Soluzione dell'esercizio 4.1

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

#define MAX_LEN 20

void main(){

    char str1[MAX_LEN+1], str2[MAX_LEN+1];
    char str12[2*MAX_LEN+1];
    int lungh1, lungh2;
    int i;

    printf("Inserire la prima stringa: ");
    gets(str1);
    fflush(stdin);

    printf("Inserire la seconda stringa: ");
    gets(str2);
    fflush(stdin);

    // strlen prima stringa
    lungh1 = 0;
    for (i = 0; str1[i] != '\0'; i++)
        lungh1++;

    // strlen seconda stringa
    lungh2 = 0;
    for (i = 0; str2[i] != '\0'; i++)
        lungh2++;

    // concatenazione delle due stringhe
    for (i = 0; i < lungh1; i++){
        str12[i] = str1[i];
    }
    for (i = 0; i < lungh2; i++){
        str12[i+lungh1] = str2[i];
    }
    str12[lungh1+lungh2] = '\0';

    // stampa della stringa concatenata
    printf("%s\n", str12);

}
```

Soluzione dell'esercizio 4.2

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

#define LEN 256

void main(){

    char frase[LEN+1];
```

```
    int palindromo;
    int n_caratteri;
    int i,j;

    printf("\nInserire la frase: ");
    gets(frase);
    fflush(stdin);

    n_caratteri = strlen(frase),

    // ciclo di controllo se palindromo,
    // scorro contemporaneamente la stringa nei due versi
    palindromo = 1;
    j = n_caratteri - 1;
    for (i = 0; (i < n_caratteri/2) && (palindromo == 1); i++) {
        // controllo che i due caratteri siano uguali
        if (frase[i] != frase[j])
            palindromo = 0;
        j--;
    }

    if (palindromo == 1)
        printf("E' palindromo\n\n");
    else
        printf("Non e' palindromo\n\n");

}
```

Soluzione dell'esercizio 4.3

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

#define MAX_LEN 160

void main(){

    char messaggio[MAX_LEN+1];
    char cifrato[MAX_LEN+1];
    int lungh;
    int chiave;
    int i;
    char cestino;

    // acquisizione messaggio da cifrare
    printf("Inserire un messaggio: ");
    gets(messaggio);
    fflush(stdin);
    lungh = strlen(messaggio);

    // acquisizione di una chiave valida
    do {
        printf("Inserire una chiave: ");
        scanf("%d", &chiave);
    } while ( chiave < 1 || chiave > 25 );
    scanf("%c", &cestino);

    // ciclo per cifrare il messaggio
    for (i = 0; i <= lungh; i++) {
        if (messaggio[i] > 96 && messaggio[i] < 123) {
```

```
            cifrato[i] = (messaggio[i] - 97 + chiave) % 26 + 97;
        } else
            cifrato[i] = messaggio[i];
    }

    printf("Messaggio cifrato: ");
    printf("%s\n", cifrato);

    // acquisizione messaggio cifrato
    printf("Inserire un messaggio cifrato: ");
    gets(cifrato);
    fflush(stdin);

    // ciclo per decifrare il messaggio
    for (i = 0; i <= lungh; i++) {
        if (cifrato[i] > 96 && cifrato[i] < 123)
            messaggio[i] = (cifrato[i] - 97 + (26 - chiave) ) % 26 + 97;
        else
            messaggio[i] = cifrato[i];

    }

    printf("Messaggio in chiaro: ");
    printf("%s\n", messaggio);
}
```

Soluzione dell'esercizio 4.4

1.

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

#define LEN 256


void main(){

    char parola[LEN+1];
    char parola_tradotta[LEN+1];
    int n,i,j;
    int accettabile;

    // acquisizone
    printf("\nInserire la parola da tradurre: ");
    scanf("%s",parola);

    n = strlen(parola);

    j = 0;
    accettabile = 1;
    // ciclo di traduzione
    for (i = 0; i <= n; i++) {
        // controllo che la stringa non superi la massima lunghezza
        if (j < LEN+1) {
            // copio lettera nella stringa tradotta
            parola_tradotta[j] = parola[i];
            // se  una vocale devo inserire la f e raddoppiarla
            if (parola[i] == 'a' || parola[i] == 'e' || parola[i] == 'i' ||
                parola[i] == 'o' || parola[i] == 'u') {
                parola_tradotta[j+1] = 'f';
                parola_tradotta[j+2] = parola[i];
```

```
                j = j + 3;
            }
            else
                j++;
        }
        else
            accettabile = 0;

    }

    if (accettabile == 1)
        printf("Parola tradotta: %s\n\n",parola_tradotta);
    else
        printf("Parola troppo lunga da tradurre\n\n");

}
```

2.

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

#define LEN 256

int main(){

    int i, k, len;
    char str[LEN];

    // acquisizione
    do {
        printf("Inserire una frase da tradurre: ");
        gets(str);
        len = strlen(str);
    } while (len > LEN);

    // lettura da sx a dx
    for (i = 0; i <= len; i++) {

        // controllo se la lettera  vocale
        if (str[i] == 'a'||
            str[i] == 'e'||
            str[i] == 'i'||
            str[i] == 'o'||
            str[i] == 'u') {

            // shift a dx di 2 posizioni
            for(k = len; k > i ; k--)
                str[k + 2] = str[k];

            // aggiungo la f e la vocale
            str[i+1] = 'f';
            str[i+2] = str[i];

            i = i + 2;
            len = len + 2;
        }
    }

    printf("%s\n", str);
}
```

Soluzione dell'esercizio 4.5

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

#define LEN 256

/*
  Tabella ascii:

  0 nul    1 soh    2 stx    3 etx    4 eot    5 enq    6 ack    7 bel
  8 bs     9 ht    10 nl    11 vt    12 np    13 cr    14 so    15 si
 16 dle   17 dc1   18 dc2   19 dc3   20 dc4   21 nak   22 syn   23 etb
 24 can   25 em    26 sub   27 esc   28 fs    29 gs    30 rs    31 us
 32 sp    33  !    34  "    35  #    36  $    37  %    38  &    39  '
 40  (    41  )    42  *    43  +    44  ,    45  -    46  .    47  /
 48  0    49  1    50  2    51  3    52  4    53  5    54  6    55  7
 56  8    57  9    58  :    59  ;    60  <    61  =    62  >    63  ?
 64  @    65  A    66  B    67  C    68  D    69  E    70  F    71  G
 72  H    73  I    74  J    75  K    76  L    77  M    78  N    79  O
 80  P    81  Q    82  R    83  S    84  T    85  U    86  V    87  W
 88  X    89  Y    90  Z    91  [    92  \    93  ]    94  ^    95  _
 96  `    97  a    98  b    99  c   100  d   101  e   102  f   103  g
104  h   105  i   106  j   107  k   108  l   109  m   110  n   111  o
112  p   113  q   114  r   115  s   116  t   117  u   118  v   119  w
120  x   121  y   122  z   123  {   124  |   125  }   126  ~   127 del
*/

void main(){

    int i, j;
    int ast;

    int hist[25]; //z-a -> 90-65 -> 25
    int HIST[25]; //Z-A -> 122-97 -> 25

    char lettera;
    char str[LEN];

    // inizializzazione dell'istogramma
    for (i = 0; i < 25; i++)
        hist[i] = HIST[i] = 0;

    // acquisizione stringa
    printf("str = ");
    gets(str);

    printf("\n");

    /*
     * Esempio:
     * str[3] = 'd'
     *
     * Bisogna incrementare l'istogramma in posizione 4
     * hist['d'-'a'] = hist[100-97] = hist[3]
     */
    for (i = 0; i < strlen(str); i++) {
        if (str[i] > 'a' && str[i] < 'z')
            hist[str[i]-'a']++;

        if (str[i] > 'A' && str[i] < 'Z')
            HIST[str[i]-'A']++;
```

```
    }

    //per ogni lettera
    for (i = 0; i < 25*2; i++)
    {
        //stampa il giusto numero di asterischi
        if (i < 25) {
            ast = hist[i];
            lettera = 'a' + i;
        } else {
            ast = HIST[i-25];
            lettera = 'A' + i - 25;
        }

        printf("%c | ", lettera);

        for (j = 0; j < ast; j++)
            printf("*");

        printf("\n");
    }
}
```

Soluzione dell'esercizio 4.6

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

#define MAX_LEN 100

void main(){

    char stringa1[MAX_LEN];
    char stringa2[MAX_LEN];
    int i, j;
    int trovato, uguali;
    char cestino;
    int lungh1;

    // acquisizione delle due stringhe
    printf("Inserire la prima stringa: ");
    scanf("%s",&stringa1);
    scanf("%c", &cestino);
    printf("Inserire la seconda stringa: ");
    scanf("%s",&stringa2);
    scanf("%c", &cestino);

    // controllo che la lunghezza sia uguale
    uguali = 1;
    if (strlen(stringa1) != strlen(stringa2))
        uguali = 0;

    // ciclo per scorrere la prima stringa
    for (i = 0; i < strlen(stringa1) && uguali; i++) {
        trovato = 0;
        // ciclo per scorrere la seconda stringa
        for (j = 0; j < strlen(stringa2) && !trovato; j++)
            // se trovo la lettera la cancello dalla stringa 2
            if (stringa1[i] == stringa2[j]){
                stringa2[j] = '-';
```

```
                trovato = 1;
            }
        // se non ho trovato la lettera non sono anagrammi
        if (trovato == 0)
            uguali = 0;
    }

    if (uguali)
        printf("Le due stringhe sono una l'anagramma dell'altra\n");
    else
        printf("No\n");
}
```

Soluzione dell'esercizio 4.7

```
#include <stdio.h>

#define MAX_DIM 20

typedef enum{falso, vero} Booleano;

void main(){

    //dichiarazioni
    int matrice[MAX_DIM][MAX_DIM];
    int dim_matr;
    int i,j;
    Booleano is_diagonale, is_simmetrica;

    //acquisizione dimensione
    do {
        printf("Inserire la dimensione della matrice quadrata: ");
        scanf("%d",&dim_matr);
    } while (dim_matr < 0 || dim_matr > MAX_DIM);

    //acquisizione elementi [i][j]
    for(i = 0; i < dim_matr; i++)
        for(j = 0; j < dim_matr; j++) {
            printf("Inserire l'elemento in posizione [%d][%d]: ",i+1,j+1);
            scanf("%d",&matrice[i][j]);
        }

    //controllo se diagonale
    is_diagonale = vero;
    // controllo elementi sulla diagonale
    for (i = 0; i < dim_matr; i++) {
        if (matrice[i][i] == 0)
            is_diagonale = falso;
        }
    //controllo elementi non sulla diagonale
    if (is_diagonale) {
        for(i = 0; i < dim_matr; i++)
            for(j = 0; j < dim_matr; j++) {
                if (i != j && matrice[i][j] != 0)
                    is_diagonale = falso;
            }
    }

    //controllo se simmetrica
    is_simmetrica = vero;
```

```
    for(i = 0; i < dim_matr; i++) {
        for(j = i+1; j < dim_matr; j++) {
            if (matrice[i][j] != matrice[j][i])
                is_simmetrica = falso;
        }
    }

    //stampa risultato del controllo
    if (is_diagonale)
        printf("La matrice e' diagonale e simetrica\n");
    else {
        if (is_simmetrica)
            printf("La matrice non e' diagonale, ma simmetrica\n");
        else
            printf("La matrice non e' diagonale, ne' simmetrica\n");
    }

}
```

Soluzione dell'esercizio 4.8

```
#include <stdio.h>

#define DIM 10

void main(){

    //dichiarazioni
    int m[DIM][DIM];
    int i, j, inf, sup;

    //inizializzazione della matrice con valori della tavola pitagorica
    for (i = 0; i < DIM; i++) {
        for (j = 0; j < DIM; j++) {
            m[i][j] = (i + 1) * (j + 1);
            printf("%2d ", m[i][j]);
        }
        printf("\n");
    }

    //stampa a spirale

    //limite inferiore (e superiore)
    for (inf = 0; inf < DIM / 2; inf++) {
        sup = DIM - inf - 1;

        printf("\n");

        //da sinistra a destra
        for (j = inf; j < sup; j++)
            printf("%2d ", m[inf][j]);

        printf("\n");

        //dall'alto verso il basso
        for (i = inf; i <= sup; i++)
            printf("%2d ", m[i][sup]);

        printf("\n");
```

```
        //da destra a sinistra
        for (j = sup - 1; j >= inf; j--)
            printf("%2d ", m[sup][j]);

        printf("\n");

        //dal basso verso l'alto
        for (i = sup - 1; i > inf; i--)
            printf("%2d ", m[i][inf]);

        printf("\n");
    }
}
```

Soluzione dell'esercizio 4.9

```
#include <stdio.h>

#define N 5       // dimensione matrice
#define RANGE 10  // da 0 a 9

void main(){

    //dichiarazioni
    int m[N][N];
    int v1[N*N], v2[N*N];
    int freq[RANGE];
    int fmax = 0;
    int valmax;
    int u, z;
    int i, j;

    //inizializzo matrice con valori crescenti (non necessario
    // se non per evitare di inserire a mano i valori)
    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
                m[i][j] = i * j;
            printf("%2d ", m[i][j]);
        }
        printf("\n");
    }

    //inizializzo vettore frequenze
    for (i = 0; i < RANGE; i++)
        freq[i] = 0;

    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            freq[m[i][j]]++;
        }
    }

    //ricerco valore piu frequente
    for (i = 0; i < RANGE; i++)
        if (i == 0 || freq[i] > fmax) {
            valmax = i;    //valore
            fmax = freq[i]; //frequenza
        }

    //spostamento valori
```

```
    u = z = 0;

    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            if (m[i][j] < valmax) {
                v1[u] = m[i][j];
                u++;
            }
            if (m[i][j] > valmax) {
                v2[z] = m[i][j];
                z++;
            }
        }
    }

    //stampa valori
    printf("fmax = %d, valmax = %d\n", fmax, valmax);
    for (i = 0; i < u; i++)
        printf("%d ", v1[i]);

    printf(" (%d elementi)\n", u);

    for (i = 0; i < z; i++)
        printf("%d ", v2[i]);

    printf(" (%d elementi)\n", z);

    //controllo se v2 e' monotono
    i = 0;
    while (i < z-1 && v2[i] <= v2[i+1])
        i++;

    //uscita prematura dal ciclo?
    if (i < z-1)
        printf("Vettore NON monotono crescente.");
    else
        printf("Vettore monotono crescente.");
}
```

Soluzione dell'esercizio 4.10

```
#include <stdio.h>
#define DIM_R 2
#define DIM_C 3

void main(){
    //dichiarazioni
    int M[DIM_R][DIM_C];
    int i, j;
    int somma, max, i_max;
    //acquisizione
    printf("\n");
    for(i = 0; i < DIM_R; i++){
        for(j = 0; j < DIM_C; j++){
            printf("Inserire cella (%d,%d): ", i + 1, j + 1);
            scanf("%d", &M[i][j]);
        }
    }
    //stampa matrice
    printf("\nM = \n");
```

```
    for(i = 0; i < DIM_R; i++){
        printf("[ ");
        for(j = 0; j < DIM_C; j++){
            printf("%d ", M[i][j]);
        }
        printf("]\n");
    }
    //calcolo massimo somma sulle righe
    max = 0;
    i_max = -1;
    for(i = 0; i < DIM_R; i++){
        somma = 0;
        for(j = 0; j < DIM_C; j++){
            somma = somma + M[i][j];
        }
        if(somma > max){
            max = somma;
            i_max = i;
        }
    }
    //stampa risultato
    printf("\nLa somma massima e' %d, alla riga %d\n\n", max, i_max + 1);
}
```