4 Array e stringhe

Soluzioni

Soluzione dell'esercizio 4.1

```c
#define LEN 256

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

int main()
{
    int i, j;

    char msg[LEN+1];
    char fsg[LEN+1];

    printf("Inserisci una parola da convertire: ");
    scanf("%s", msg);

    for (i = 0, j = 0; msg[i] != '\0'; i++, j++) {
        //in ogni caso copio lettera per lettera
        fsg[j] = msg[i];

        //se trovo una vocale
        if (msg[i] == 'a' || msg[i] == 'e' || msg[i] == 'i' || msg[i] == 'o' || msg[i]
            == 'u') {
            //inserisco la 'f'
            j++;
            fsg[j] = 'f';

            //ripeto la vocale dopo la 'f'
            j++;
            fsg[j] = msg[i];
        }
    }

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

```c
#define LEN 256

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

/*
   //esempio di un passo
    i = 1

    //shift in avanti di 2 posizioni

    0 1 2 3 4
    c i a o \0

                  j = len = 4
          +------+
          |      |
          |      v
    c i a o \0 ? \0
```

```
             j = len-1 = 3
          +---+
          |   |
          |   v
    c i a o ? o \0

            j = len-2 = 2
        +---+
        |   |
        |   v
    c i a o a o \0

          j = len-3 = 1 == i = 1 (stop)
      +---+
      |   |
      |   v
    c i a i a o \0

    //inserimento della f
    c i a i a o \0

    c i f i a o \0
 */

int main()
{
    int i, j;
    int len, len_iniziale;

    char msg[LEN+1];

    printf("Inserisci una parola da convertire: ");
    scanf("%s", msg);

    i = 0;

    //attenzione, non posso usare la strlen() perche' la lunghezza della
    //stringa cambia durante il ciclo
    while (msg[i] != '\0') {
        if (msg[i] == 'a' || msg[i] == 'e' || msg[i] == 'i' || msg[i] == 'o' || msg[i]
            == 'u') {
            //controllo di avere abbastanza spazio per le due lettere
            //aggiuntive

            if (strlen(msg) + 2 < LEN) {
                //prima faccio spazio per 2 lettere aggiuntive, spostando tutto
                //in avanti di 2

                len = strlen(msg);

                //dall'ultima lettera alla corrente (inclusa)
                //j = len -> posizione terminatore
                for (j = len; j >= i; j--)
                    msg[j+2] = msg[j];

                //devo inserire 'f' in posizione i+1
                msg[i+1] = 'f';

                //avendo spostato in avanti di 2 posizioni, la prossima lettera
                //nella parola originale non e' la i+1-esima ma la i+3-esima
                i = i + 3;
            } else
```

```
                printf("Non e` possibile convertire tutta la parola\n");
        } else
            i++; //non vocale, quindi vado semplicemente avanti un passo
    }

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

Soluzione dell'esercizio 4.2

```
#define LEN 256

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

int main() {
    char frase[LEN+1];

    //indice
    int i;
    int len;

    printf("Inserisci una frase\n");
    gets(frase);

    len = strlen(frase);

    /*
     Scorro l'array in due direzioni opposte e controllo se le lettere non sono
     uguali. Mi basta trovare due lettere diverse per uscire dal ciclo e
     dichiarare la frase come non palindroma.

     Esempio (caso di lunghezza dispari):

     "abbabba"

     len = 7

     len/2 = 3 (int)

     Ad ogni ciclo, i e len-i-1 prendono i seguenti valori:

     i = 0 1 2

     len-i-1 = 6 5 4

     Quindi i confronti fatti sono:

     frase[0] == frase[6] --> vero
     frase[1] == frase[5] --> vero
     frase[2] == frase[4] --> vero

     prima di uscire dal ciclo i viene incrementata a 3

     3 = i < len/2 = 3 --> falso, quindi palindroma

     --

     Esempio (caso di lunghezza pari):
```

```
     "sabbas"

     len = 6

     len/2 = 3

     Ad ogni ciclo, i e len-i-1 prendono i seguenti valori:

     i = 0 1 2

     len-i-1 = 5 4 3

     Quindi i confronti fatti sono:

     frase[0] == frase[5] --> vero
     frase[1] == frase[4] --> vero
     frase[2] == frase[3] --> vero

     prima di uscire dal ciclo i viene incrementata a 3

     3 = i < len/2 = 3 --> falso

     --

     Esempio (non palindroma):

     "sabcas"

     len = 6

     len/2 = 3

     Ad ogni ciclo, i e len-i-1 prendono i seguenti valori:

     i = 0 1 2

     len-i-1 = 5 4 3

     Quindi i confronti fatti sono:

     frase[0] == frase[5] --> vero
     frase[1] == frase[4] --> vero
     frase[2] == frase[3] --> falso --> si esce dal ciclo

     il contatore i non viene incrementato, quindi

     2 = i < len/2 = 3 --> vero
    */

    for (i = 0; i < len/2 && frase[i] == frase[len-i-1]; i++) {
        /*
          non faccio nulla - equivale a scrivere:

          for (i = 0; i < len/2 && frase[i] == frase[len-i-1]; i++);
        */
    }

    if (i < len/2) //uscita prematura?
        printf("non ");

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

Soluzione dell'esercizio 4.3

```c
#define DIM 10

#include <stdio.h>

int main()
{
    int i;
    int j; //indice usato solo per chiarezza, ma non necessario

    int indice[DIM];
    int valori[DIM];
    int dim;

    //acquisizione della dimensione
    do {
        printf("Quanti elementi vuoi inserire (max: %d)? ", DIM);
        scanf("%d", &dim);

        if (dim > DIM)
            printf("Il valore inserito e' troppo grande!\n");
    } while (dim > DIM);

    //acquisizione valori
    for (i = 0; i < dim; i++) {
        printf("Inserire il %do valore: ", i+1);
        scanf("%d", &valori[i]);
    }

    //acquisizione indice
    for (i = 0; i < dim; i++) {
        printf("Qual e' il %do valore che vuoi stampare? ", i+1);

        //tutti i valori dell'indice devono puntare ad elementi validi
        // quindi all'interno dei limiti 0,dim
        do {
            scanf("%d", &indice[i]);

            if (indice[i] > dim || indice[i] < 0)
                printf("Il valore inserito non e' in [0, %d]", dim);
        } while (indice[i] > dim || indice[i] < 0);

        //si assume che l'utente che vuole indicare il primo elemento (ad
        // esempio), inserisca il numero 1, quindi dobbiamo decrementare
        // tutto di 1: 1o, 2o, 3o... -> 0, 1, 2
        indice[i] = indice[i] - 1;
    }

    //stampa in ordine stabilito dall'indice
    for (i = 0; i < dim; i++)
    {
        j = indice[i];

        printf("%d\n", valori[j]);
    }
}
```

Soluzione dell'esercizio 4.4

```c
#define DIM 20
#define INF 18
#define SUP 30

#include <stdio.h>
#include <math.h> //per sqrt(), radice quadrata, e pow(), potenza

int main()
{
    int i;
    int dim;
    int voti[DIM];
    int min;
    int max;

    //statistiche
    float media = 0.0;
    float media_troncata;
    float scarto_medio = 0.0;
    float scarto;
    float dev_std;
    float var;

    //acquisizione della dimensione
    do {
        printf("Quanti elementi vuoi inserire (max: %d)? ", DIM);
        scanf("%d", &dim);

        if (dim > DIM)
            printf("Il valore inserito e' troppo grande!\n");
    } while (dim > DIM);

    //acquisizione voti con ricerca di min max
    for (i = 0; i < dim; i++) {
        do {
            printf("Inserire il %do voto: ", i+1);
            scanf("%d", &voti[i]);

            if (voti[i] < INF || voti[i] > SUP)
                printf("\nIl valore inserito e' fuori dagli estremi [%d, %d]!\n",
                       INF, SUP);
        } while (voti[i] < INF || voti[i] > SUP);

        if (voti[i] < min)
            min = voti[i];

        if (voti[i] > max)
            max = voti[i];

        media = media + voti[i];
    }

    //i valori sono tutti positivi, quindi posso eliminare min/max semplicemente
    media_troncata = media - min - max;

    //calcolo di media e media troncata
    media = media/dim;
    media_troncata = media_troncata/(dim-2);

    //calcolo scarti quadratici dalla media
    for (i = 0; i < dim; i++) {
        scarto = (voti[i] - media);
```

```
        scarto_medio = scarto_medio + pow(scarto, 2);
    }

    //calcolo varianza (va bene anche con "dim")
    //http://it.wikipedia.org/wiki/Varianza
    var = scarto_medio/(dim-1);

    //calcolo deviazione standard
    //http://it.wikipedia.org/wiki/Deviazione_standard
    dev_std = sqrt(var);

    //calcolo scarto medio
    scarto_medio = scarto_medio/dim;

    //stampa
    printf(
        "\n\nSTATISTICHE VOTI:\n\n"
        "Esami sostenuti: %.2f\n"
        "Media: %.2f\n"
        "Media troncata: %.2f\n"
        "Varianza: %.2f\n"
        "Deviazione standard: %.2f\n",

        dim,
        media,
        media_troncata,
        scarto_medio,
        var,
        dev_std);
}
```

Soluzione dell'esercizio 4.5

```
#define DIM 30

#include <stdio.h>

int main()
{
    int i, j, k = 0;
    int dim;
    int trovato;
    int lista[DIM];
    int insieme[DIM];

    //acquisizione della dimensione
    do {
        printf("Quanti elementi vuoi inserire (max: %d)? ", DIM);
        scanf("%d", &dim);

        if (dim > DIM)
            printf("Il valore inserito e' troppo grande!\n");
    } while (dim > DIM);

    for (i = 0; i < dim; i++) {
        printf("Inserire il %do elemento: ", i+1);
        scanf("%d", &lista[i]);
    }

    //per ogni elemento
```

```
    for (i = 0; i < dim; i++) {
        trovato = 0;

        //scorro la lista fino a che trovo lo stesso elemento
        for (j = 0; !trovato && j < k; j++)
            trovato = (lista[i] == insieme[j]);

        //se non trovato, lo inserisco e incremento l'indice
        if (!trovato) {
            insieme[k] = lista[i];
            k++;
        }
    }

    // stampo la lista
    printf("lista = [");
    for (i = 0; i < dim; i++) {
        printf("%d ", lista[i]);
    }
    printf("]\n");

    //stampo l'insieme - k ora  la lunghezza dell'insieme
    printf("insieme = {");
    for (i = 0; i < k; i++) {
        printf("%d ", insieme[i]);
    }

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

Soluzione dell'esercizio 4.6

```
#define DIM 30

#include <stdio.h>

int main()
{
    // indici per scansione
    int i, j;

    // dimensione effettiva degli array
    int dim_a, dim_b;

    // flag utile per le ricerche di elementi
    int trovato;

    // array e insiemi A e B
    int lista[DIM];
    int A[DIM];
    int B[DIM];

    // dimensione effettiva degli insiemi
    int len_a = 0;
    int len_b = 0;
    int len_u = 0; //dimensione unione
    int len_i = 0; //dimensione intersezione
    int len_d = 0; //dimensione differenza

    // altri insiemi
```

```
int unione[2*DIM];    // A u B
int intersezione[DIM];// A ^ B
int differenza[DIM];  // B \ A

/*
 * ACQUISIZIONE DELLA DIMENSIONE DELLA PRIMA LISTA
 */
do {
    printf("Quanti elementi vuoi inserire nella prima lista (max: %d)? ",
           DIM);
    scanf("%d", &dim_a);

    if (dim_a > DIM)
        printf("Il valore inserito  troppo grande!\n");
} while (dim_a > DIM);

/*
 * ACQUISIZIONE DELLA PRIMA LISTA DI ELEMENTI
 */
for (i = 0; i < dim_a; i++) {
    printf("Inserire il %do elemento: ", i+1);
    scanf("%d", &lista[i]);
}

/*
 * CONVERSIONE IN INSIEME A
 */
for (i = 0; i < dim_a; i++) {
    trovato = 0;

    //ricerco il valore i-esimo nella lista
    for (j = 0; !trovato && j < len_a; j++)
        trovato = (lista[i] == A[j]);

    //se non trovato, lo inserisco e incremento l'indice
    if (!trovato) {
        //gi che ci sono, lo inserisco anche nell'unione
        unione[len_a] = lista[i];

        A[len_a] = lista[i];
        len_a++;
    }
}

/*
 * A QUESTO PUNTO A COINCIDE CON L'INSIEME UNIONE, AL QUALE DOVR
 * AGGIUNGERE GLI ELEMENTI DI B SENZA RIPETIZIONI
 */
len_u = len_a;

// stampa l'insieme A
printf ("Insieme A = {");
for (i = 0; i < len_a; i++)
    printf("%d ", A[i]);
printf ("}\n");

/*
 * ACQUISIZIONE DELLA DIMENSIONE DELLA SECONDA LISTA
 */
do {
    printf("Quanti elementi vuoi inserire nella seconda lista (max: %d)? ",
           DIM);
```

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

        if (dim_b > DIM)
            printf("Il valore inserito  troppo grande!\n");
    } while (dim_b > DIM);

    /*
     * ACQUISIZIONE DELLA SECONDA LISTA DI ELEMENTI
     */
    for (i = 0; i < dim_b; i++) {
        printf("Inserire il %do elemento: ", i+1);
        scanf("%d", &lista[i]);
    }

    /*
     * CONVERSIONE IN INSIEME B E CALCOLO UNIONE, INTERSEZIONE,
     * DIFFERENZA
     */
    for (i = 0; i < dim_b; i++) {
        trovato = 0;

        //ricerco il valore i-esimo nella lista
        for (j = 0; !trovato && j < len_b; j++)
            trovato = (lista[i] == B[j]);

        //se non trovato, lo inserisco e incremento l'indice
        if (!trovato) {
            B[len_b] = lista[i];
            len_b++;
        }

        /*
         * UNIONE: Tutti gli elementi in A e tutti gli elementi in B
         * senza ripetizioni.
         */
        trovato = 0;

        //ricerco il valore i-esimo nella lista
        for (j = 0; !trovato && j < len_u; j++)
            trovato = (unione[j] == lista[i]);

        //se non trovato, allora va aggiunto all'unione
        if (!trovato) {
            unione[len_u] = lista[i];
            len_u++;
        }

        /*
         * INTERSEZIONE: Tutti gli elementi che sono sia in A che in
         * B, senza ripetizioni.
         */
        trovato = 0;

        //uso len_b-1 perch len_b  gi stato incrementato
        for (j = 0; !trovato && j < len_b; j++)
            trovato = (A[j] == B[len_b-1]);

        //trovato anche in A: allora fa parte dell'intersezione
        if (trovato) {
            trovato = 0;

            //controllo per non inserire duplicati in intersezione
```

```
            for (j = 0; !trovato && j < len_i; j++)
                trovato = (intersezione[j] == B[len_b-1]);

            if (!trovato) {
                intersezione[len_i] = B[len_b-1];
                len_i++;
            }
        } else { //non trovato in A: allora fa parte della differenza
            /*
             * DIFFERENZA: Tutti gli elementi che sono in B, tranne
             * quelli che sono in A.
             */
            trovato = 0;

            //controllo per non inserire duplicati in differenza
            for (j = 0; !trovato && j < len_d; j++)
                trovato = (differenza[j] == B[len_b-1]);

            if (!trovato) {
                differenza[len_d] = B[len_b-1];
                len_d++;
            }
        }
    }

    // stampa l'insieme B
    printf ("B = {");
    for (i = 0; i < len_b; i++)
        printf("%d ", B[i]);
    printf ("}\n");

    // stampa l'insieme unione
    printf ("unione = {");
    for (i = 0; i < len_u; i++)
        printf("%d ", unione[i]);
    printf ("}\n");

    // stampa l'insieme intersezione
    printf ("intersezione = {");
    for (i = 0; i < len_i; i++)
        printf("%d ", intersezione[i]);
    printf ("}\n");

    // stampa l'insieme differenza
    printf ("differenza = {");
    for (i = 0; i < len_d; i++)
        printf("%d ", differenza[i]);
    printf ("}\n");
}
```

Soluzione dell'esercizio 4.7

```
#define DIM 10

#include <stdio.h>

int main()
{
    //dichiarazione variabili
    int i;
```

```
    int dim;
    int min_idx = 0; //assumiamo che min sia il primo elemento
    int max_idx = 0; //assumiamo che max sia il primo elemento

    float valori[DIM];

    //acquisizione della dimensione
    do {
        printf("Quanti elementi vuoi inserire (max: %d)? ", DIM);
        scanf("%d", &dim);

        if (dim > DIM)
            printf("Il valore inserito e' troppo grande!\n");
    } while (dim > DIM);

    //acquisizione valori con ricerca di min max
    for (i = 0; i < dim; i++) {
        printf("Inserire il %do valore: ", i+1);
        scanf("%f", &valori[i]);

        if (valori[i] < valori[min_idx])
            min_idx = i;

        if (valori[i] > valori[max_idx])
            max_idx = i;
    }

    //stampa dei valori con indicazione del massimo e del minimo
    for (i = 0; i < dim; i++) {
        if (i == min_idx && i == max_idx)
            printf("- +|");
        else {
            if (i == min_idx)
                printf("-  |");

            if (i == max_idx)
                printf("  +|");
        }

        if (i != min_idx && i != max_idx)
            printf("   |");

        printf(" %.2f\n", valori[i]);
    }
}
```

Soluzione dell'esercizio 4.8

```
#define LEN 160
#define KMIN 0
#define KMAX 10

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

int main() {
    char messaggio[LEN+1];

    int i;
    int chiave;
```

```
    int len;

    printf(
        "Scrivere un SMS su una riga (massimo %d caratteri):\n"
        "---premere invio per terminare il messaggio---\n", LEN);
    gets(messaggio);

    do {
        printf("Inserire la chiave di cifratura: intero in [%d, %d]: ",
               KMIN, KMAX);
        scanf("%d", &chiave);

        if (chiave < KMIN || chiave > KMAX)
            printf("Chiave non valida!");
    } while (chiave < KMIN || chiave > KMAX);

    len = strlen(messaggio);

    printf("Messaggio cifrato:\n");

    //cifratura (scorrimento al contrario + chiave)
    for (i = 0; i < len+1; i++)
        printf("%c", messaggio[len-i] + chiave);
    printf("\n\n");

    //alternativa alla fflush(stdin), che su alcuni terminali
    //potrebbe non funzionare correttamente
    while(getchar() != '\n');

    //decifratura
    printf(
        "Scrivere l'SMS cifrato su una riga (massimo %d caratteri):\n"
        "---premere invio per terminare il messaggio---\n", LEN);
    gets(messaggio);

    len = strlen(messaggio);

    for (i = len; i > 0; i--)
        printf("%c", messaggio[i-1] - chiave);
    printf("\n");
}
```

Soluzione dell'esercizio 4.9

```
#define LEN 256

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

/*
  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
*/

int 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 4.10

```
#define LEN 100

#include <stdio.h>

int main()
{
    int i, j;

    /*

      str1[0] = 'c';
      str1[1] = 'i';
      str1[2] = 'a';
      str1[3] = 'o';
      str1[4] = '\0';

      str2[0] = 'm';
      str2[1] = 'a';
      str2[2] = 'r';
      str2[3] = 'e';
      str2[4] = '\0';

      cat[0] = str1[0] = 'c';
      cat[1] = str1[1] = 'i';
      cat[2] = str1[2] = 'a';
      cat[3] = str1[3] = 'o';
      cat[4] = str2[0] = 'm';
      cat[5] = str2[1] = 'a';
      cat[6] = str2[2] = 'r';
      cat[7] = str2[3] = 'e';
      cat[8] = '\0';

     */

    char str1[LEN+1];
    char str2[LEN+1];
    char cat[2*LEN+1];

    printf("str1 = ");
    gets(str1);

    printf("str2 = ");
    gets(str2);

    for (i = 0; str1[i] != '\0'; i++) {
        printf("cat[%d] = str[%d] = %c\n", i, i, str1[i]);
        cat[i] = str1[i];
    }

    for (j = 0; str2[j] != '\0'; j++) {
        printf("cat[%d] = str[%d] = %c\n", i+j, i, str2[j]);
        cat[i+j] = str2[j];
    }
```

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