Caratteri e stringhe

Esercizi risolti

1 Esercizio: “Conta 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:

- per ognuna delle lettere dell’alfabeto, il numero di volte che la lettera compare nella stringa
- il numero di consonanti presenti nella stringa
- il numero di vocali presenti nella stringa.

Soluzione

```
1  /* PROGRAMMAZIONE IN C */
2
3  /* File: contavocaliconsonanti.c */
4  /* Soluzione proposta esercizio "Conta vocali e consonanti" */
5
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <string.h>
9
10 int main(void)
11 {
12     const int MAXDIM = 100 ;       /* dimensione massima stringa di caratteri */
13     const int NUMLETTERE = 26 ;    /* numero di lettere dell'alfabeto */
14
15     char frase[MAXDIM +1] ;        /* stringa di caratteri inserita */
16     int lung_stringa ;             /* lunghezza della stringa inserita */
17     int vocali, consonanti ;       /* contatori numero di vocali e di consonanti */
18     int contatori[NUMLETTERE];     /* memorizza il numero di occorrenze per
19                                       ogni lettera */
20     int posizione_alfabeto ;       /* posizione nell'alfabeto di una lettera */
21     int i ;                        /* indice dei cicli */
22
23
24     /* LEGGI LA FRASE INSERITA DA TASTIERA */
25     printf ("Inserisci una frase di al massimo %d caratteri: ", MAXDIM) ;
26     gets(frase) ;
27
28     /* CALCOLA LA LUNGHEZZA DELLA FRASE */
29     lung_stringa = strlen(frase) ;
30
31     /* STAMPA LA FRASE INSERITA */
32     printf("La frase inserita e': ") ;
33     puts(frase) ;
34     printf("La frase contiene %d caratteri (inclusi gli spazi)\n", lung_stringa) ;
35
36     /* AZZERA IL VETTORE DEI CONTATORI. OGNI CELLA DI QUESTO VETTORE E'
37     ASSOCIATA A UNA LETTERA DELL'ALFABETO. LA CELLA 0 ALLA LETTERA A,
```

```
38      LA CELLA 1 ALLA B E COSI' VIA */
39
40      for ( i=0; i<NUMLETTERE; i++ )
41          contatori[i] = 0 ;
42
43      /* ANALIZZA LA FRASE LETTERA PER LETTERA E AGGIORNA IL VETTORE DEI CONTATORI */
44      for ( i=0; i<lung_stringa; i++ )
45      {
46          if ( frase[i] >= 'A' && frase[i] <= 'Z' )
47          {
48              /* IL CARATTERE ESAMINATO E' UNA LETTERA MAIUSCOLA POICHE'
49              IL SUO CODICE ASCII E' COMPRESO TRA QUELLI DELLE LETTERE A E Z.
50              PER RICAVARE LA CELLA DEL VETTORE "contatori" DA INCREMENTARE
51              DEVI IDENTIFICARE LA POSIZIONE DELLA LETTERA NELL'ALFABETO.
52              POICHE' I CODICI ASCII DELLE LETTERE MAIUSCOLE SONO CONSECUTIVI,
53              BASTERA' SOTTRARRE AL CARATTERE ESAMINATO IL CODICE ASCII DELLA
54              PRIMA LETTERA DELL'ALFABETO ('A') */
55
56              posizione_alfabeto = frase[i] - 'A' ;
57              contatori[posizione_alfabeto] ++ ;
58          }
59          else
60          {
61              if ( frase[i] >= 'a' && frase[i] <= 'z' )
62              {
63                  /* IL CARATTERE ESAMINATO E' UNA LETTERA MINUSCOLA POICHE'
64                  IL SUO CODICE ASCII E' COMPRESO TRA QUELLI DELLE LETTERE a E z.
65                  PER RICAVARE LA CELLA DEL VETTORE "contatori" DA INCREMENTARE
66                  DEVI IDENTIFICARE LA POSIZIONE DELLA LETTERA NELL'ALFABETO.
67                  POICHE' I CODICI ASCII DELLE LETTERE MINUSCOLE SONO CONSECUTIVI,
68                   BASTERA' SOTTRARRE AL CARATTERE ESAMINATO IL CODICE ASCII DELLA
69                  PRIMA LETTERA DELL'ALFABETO ('a') */
70
71                  posizione_alfabeto = frase[i] - 'a' ;
72                  contatori[posizione_alfabeto] ++ ;
73              }
74          }
75      }
76
77      /* STAMPA I CONTATORI DELLE VARIE LETTERE */
78      for ( i=0; i<NUMLETTERE; i=i+1 )
79          printf ("La␣lettera␣%c␣compare␣%d␣volte␣\n",
80                  'A'+i , contatori[i]) ;
81
82      /* CALCOLA IL NUMERO DI VOCALI */
83      /* SOMMA IL NUMERO DI OCCORRENZE PRESENTI NEL VETTORE "contatori"
84      NELLE CELLE ASSOCIATE ALLE LETTERE A, E, I, O, U, Y */
85      vocali = contatori['A'-'A'] + contatori['E'-'A'] + contatori['I'-'A'] +
86               contatori['O'-'A'] + contatori['U'-'A'] + contatori['Y'-'A'] ;
87
88      /* CALCOLA IL NUMERO DI CONSONANTI */
89      /* IL NUMERO DI CONSONANTI SI OTTIENE SOTTRAENDO DAL NUMERO COMPLESSIVO
90      DI OCCORRENZE DI TUTTE LE LETTERE, IL NUMERO COMPLESSIVO DI VOCALI */
91
92      consonanti = 0 ;
93      for ( i=0; i<NUMLETTERE; i=i+1 )
```

```
94          consonanti = consonanti + contatori[i] ;
95
96      consonanti = consonanti - vocali ;
97
98      /* STAMPA IL NUMERO DI VOCALI E CONSONANTI */
99      printf ("Il␣numero␣di␣vocali␣e':␣%d\n", vocali) ;
100     printf ("Il␣numero␣di␣consonanti␣e':␣%d\n", consonanti) ;
101     exit(0) ;
102 }
```

2 Esercizio: "Sostituisci carattere"

Scrivere un programma in linguaggio C che legga una frase introdotta da tastiera. La frase è terminata dall'introduzione del carattere di invio e contiene complessivamente al più 100 caratteri. Il programma deve svolgere le seguenti operazioni:

- visualizzare la frase inserita
- costruire una nuova frase in cui tutte le occorrenze del carattere `'.'` sono sostituite con il carattere di ritorno di linea `'\n'`. Il programma deve memorizzare la nuova frase in una opportuna variabile
- visualizzare la nuova frase.

Soluzione

```
1  /* PROGRAMMAZIONE IN C */
2
3  /* File: sostituiscicarattere.c */
4  /* Soluzione proposta esercizio "Sostituisci carattere" */
5
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <string.h>
9
10 int main(void)
11 {
12     const int MAXDIM = 100 ;           /* dimensione max stringa di caratteri */
13
14     char frase[MAXDIM + 1] ;           /* stringa di caratteri inserita */
15     char frasemodificata[MAXDIM + 1] ; /* nuova stringa modificata */
16     int lung_stringa ;                 /* lunghezza della stringa inserita */
17     int i ;                            /* indice dei cicli */
18
19     /* LEGGI LA FRASE INSERITA DA TASTIERA */
20     printf ("Inserisci␣una␣frase␣di␣al␣massimo␣%d␣caratteri:␣", MAXDIM) ;
21     gets(frase) ;
22
23     /* CALCOLA LA LUNGHEZZA DELLA FRASE */
24     lung_stringa = strlen(frase) ;
25
26     /* STAMPA LA FRASE INSERITA */
27     printf("La␣frase␣inserita␣e':␣") ;
28     puts(frase) ;
29     printf("La␣frase␣contiene␣%d␣caratteri␣(inclusi␣gli␣spazi)\n", lung_stringa) ;
30
31     /* ANALIZZA LA FRASE INSERITA CARATTERE PER CARATTERE. RICOPIA LA FRASE
```

```
32      NELLA STRINGA "frase modificata". SE LA STRINGA INSERITA CONTIENE IL
33      CARATTERE ".", SOSTITUISCILO CON IL CARATTERE DI RITORNO DI LINEA "\n" */
34      for ( i=0; i<lung_stringa; i=i+1 )
35      {
36          if ( frase[i] == '.' )
37              frasemodificata[i] = '\n' ;
38          else
39              frasemodificata[i] = frase[i] ;
40      }
41      frasemodificata[lung_stringa] = '\0' ;
42
43      /* STAMPA LA FRASE MODIFICATA */
44      printf("La frase modificata e': \n") ;
45      puts(frasemodificata) ;
46      exit(0) ;
47  }
```

3 Esercizio: "Codifica di una parola"

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 deve svolgere le seguenti operazioni:

- visualizzare la frase inserita;
- costruire una nuova frase tale che ogni lettera vocale presente nella frase di partenza sia seguita dalla lettera `'f'` (se la vocale è minuscola) o dalla lettera `'F'` (se la vocale è maiuscola) nella nuova frase. Il programma deve memorizzare la nuova frase in una opportuna variabile.
- visualizzare la nuova frase.

Ad esempio, la frase `VacAnze di NaTAle` diviene `VafcAFnzef dif NafTAFlef`.

Soluzione

```
1   /* PROGRAMMAZIONE IN C */
2
3   /* File: codificadiunaparola.c */
4   /* Soluzione proposta esercizio "Codifica di una parola" */
5
6   #include <stdio.h>
7   #include <stdlib.h>
8   #include <string.h>
9   #include <ctype.h>
10
11  int main(void)
12  {
13      const int MAXDIM = 100 ;           /* dimensione max stringa di caratteri */
14
15      char frase[MAXDIM + 1] ;             /* stringa di caratteri inserita */
16      char frasemodificata[2*MAXDIM + 1] ; /* nuova stringa modificata */
17      int lung_stringa ;                   /* lunghezza della stringa inserita */
18      int i, j ;                           /* indici dei cicli */
19
20      /* LEGGI LA FRASE INSERITA DA TASTIERA */
21      printf ("Inserisci una frase di al massimo %d caratteri: ", MAXDIM) ;
```

```
22      gets(frase) ;
23
24      /* CALCOLA LA LUNGHEZZA DELLA FRASE */
25      lung_stringa = strlen(frase) ;
26
27      /* STAMPA LA FRASE INSERITA */
28      printf("La␣frase␣inserita␣e':␣") ;
29      puts(frase) ;
30      printf("La␣frase␣contiene␣%d␣caratteri␣(inclusi␣gli␣spazi)\n", lung_stringa) ;
31
32      /* COSTRUISCI LA NUOVA FRASE */
33      /* L'INDICE "i" E' USATO PER SCORRERE LA STRINGA "frase". L'INDICE "j" E'
34      USATO PER SCORRERE LA STRINGA "frasemodificata" */
35      for ( i=0, j=0; i<lung_stringa; i++ )
36      {
37          /* RICOPIA IL CARATTERE IN "frase[i]" nella cella "frasemodificata[j]" */
38          /* INCREMENTA IL CONTATORE "j" PER ACCEDERE ALLA CELLA SUCCESSIVA
39          NELLA STRINGA "frasemodificata" */
40          frasemodificata[j] = frase[i] ;
41          j = j + 1 ;
42
43          /* SE "frase[i]" CONTIENE UNA VOCALE MINUSCOLA,
44          INSERISCI IL CARATTERE "f" NELLA CELLA "frasemodificata[j]" */
45          /* INCREMENTA IL CONTATORE "j" PER ACCEDERE ALLA CELLA SUCCESSIVA
46          NELLA STRINGA "frasemodificata" */
47          if ( frase[i] == 'a' || frase[i] == 'e' || frase[i] == 'i'
48                  || frase[i] == 'o' || frase[i] == 'u')
49          {
50              frasemodificata[j] = 'f' ;
51              j = j + 1 ;
52          }
53          else
54          {
55              /* SE "frase[i]" CONTIENE UNA LETTERA VOCALE IN CARATTERE MAIUSCOLO,
56              INSERISCI IL CARATTERE "F" NELLA CELLA "frasemodificata[j]" */
57              /* INCREMENTA IL CONTATORE "j" PER ACCEDERE ALLA CELLA SUCCESSIVA
58              NELLA STRINGA "frasemodificata" */
59              if ( frase[i] == 'A' || frase[i] == 'E' || frase[i] == 'I'
60                      || frase[i] == 'O' || frase[i] == 'U' )
61              {
62                  frasemodificata[j] = 'F' ;
63                  j = j + 1 ;
64              }
65          }
66      }
67      frasemodificata[j] = '\0' ;
68
69      /* STAMPA LA FRASE MODIFICATA */
70      printf("La␣frase␣modificata␣e':␣") ;
71      puts(frasemodificata) ;
72      exit(0) ;
73  }
```

4 Esercizio: “Primo carattere maiuscolo”

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 deve svolgere le seguenti operazioni:

- visualizzare la frase inserita
- costruire una nuova frase in cui il primo carattere di ciascuna parola nella frase di partenza è stato reso maiuscolo. Tutti gli altri caratteri devono essere resi minuscoli. Il programma deve memorizzare la nuova frase in una opportuna variabile
- visualizzare la nuova frase.

Ad esempio la frase `cHe bElLA gIOrnaTa` diviene `Che Bella Giornata`.

Soluzione

```
1  /* PROGRAMMAZIONE IN C */
2
3  /* File: primocaratteremaiuscolo.c */
4  /* Soluzione proposta esercizio "Primo carattere maiuscolo" */
5
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <ctype.h>
9  #include <string.h>
10
11 int main(void)
12 {
13     const int MAXDIM = 100 ;       /* dimensione massima stringa di caratteri */
14
15     char frase[MAXDIM +1] ;        /* stringa di caratteri inserita */
16     char nuovafrase[MAXDIM +1] ;   /* stringa di caratteri modificata */
17     int lung_stringa ;             /* lunghezza della stringa inserita */
18     int i ;                        /* indice dei cicli */
19
20     /* LEGGI LA FRASE INSERITA DA TASTIERA */
21     printf ("Inserisci una frase di al massimo %d caratteri: ", MAXDIM) ;
22     gets(frase) ;
23
24     /* CALCOLA LA LUNGHEZZA DELLA FRASE */
25     lung_stringa = strlen(frase) ;
26
27     /* STAMPA LA FRASE INSERITA */
28     printf("La frase inserita e': ") ;
29     puts(frase) ;
30     printf("La frase contiene %d caratteri (inclusi gli spazi)\n", lung_stringa) ;
31
32     /* COSTRUISCI LA NUOVA FRASE */
33     for ( i=0; i<lung_stringa; i++ )
34     {
35         /* IL CARATTERE "frase[i]" E' LA PRIMA LETTERA DI UNA PAROLA SE IL
36         CARATTERE PRECEDENTE ("frase[i-1]") ERA UNO SPAZIO OPPURE SE E' IL PRIMO
37         CARATTERE DELLA FRASE (OSSIA i==0). IN QUESTO CASO IL CARATTERE "frase[i]"
38         E' CONVERTITO IN CARATTERE MAIUSCOLO. IN TUTTI GLI ALTRI CASI IL CARATTERE
```

```
39          "frase[i]" E' CONVERTITO IN CARATTERE MINUSCOLO */
40          if ( (i==0) || isspace(frase[i-1]) )
41              nuovafrase[i] = toupper(frase[i]) ;
42          else
43              nuovafrase[i] = tolower(frase[i]) ;
44      }
45      nuovafrase[lung_stringa] = '\0' ;
46
47      /* STAMPA LA FRASE MODIFICATA */
48      printf("La frase modificata e': ") ;
49      puts(nuovafrase) ;
50      exit(0);
51  }
```

5 Esercizio: "Conversione binario decimale"

Scrivere un programma in linguaggio C che legga da tastiera un numero binario puro sotto forma di una stringa di caratteri (0 o 1) lunga al massimo 24 bit. Il programma deve:

- controllare che la stringa inserita sia corretta, vale a dire composta solo da caratteri 0 e 1
- convertire il numero binario inserito nell'equivalente valore decimale
- stampare sul video il valore decimale.

Soluzione

```
1   /* PROGRAMMAZIONE IN C */
2
3   /* File: conversionebindec.c */
4   /* Soluzione proposta esercizio "Conversione binario decimale" */
5
6   #include <stdio.h>
7   #include <stdlib.h>
8   #include <string.h>
9
10  int main(void)
11  {
12      const int MAXDIM = 24 ;        /* dimensione massima stringa di caratteri */
13
14      char binario[MAXDIM + 1] ;     /* stringa contenente il numero binario */
15
16      int num_cifre ;                /* numero di cifre nel numero binario */
17      int decimale ;                 /* numero decimale risultante */
18      int corretto ;                 /* flag per la ricerca */
19      int i ;                        /* indice dei cicli */
20
21      /* LEGGI IL NUMERO BINARIO */
22      printf("Inserisci un numero binario puro di al massimo %d cifre: ", MAXDIM) ;
23      gets(binario) ;
24
25      /* CALCOLA IL NUMERO DI CIFRE DEL NUMERO BINARIO */
26      num_cifre = strlen(binario) ;
27
28      /* VISUALIZZA IL NUMERO INSERITO */
29      printf("Il numero binario inserito e' %s e contiene %d cifre\n",
```

```
30              binario, num_cifre);
31
32      /* VERIFICA SE IL NUMERO INSERITO CONTIENE SOLO CARATTERI 0 E 1 */
33      /* IL NUMERO BINARIO NON E' CORRETTO SE CONTIENE ALMENO UNA CIFRA DIVERSA
34      SIA DA 0 CHE DA 1 */
35      corretto = 1 ;
36      for ( i=0 ; i<num_cifre; i++ )
37          if ( binario[i]!='0' && binario[i]!='1' )
38              corretto = 0 ;
39
40      if ( corretto == 0 )
41          printf("Il numero binario inserito non e' valido\n") ;
42      else
43      {
44          /* CONVERTI IL NUMERO BINARIO NEL NUMERO DECIMALE CORRISPONDENTE */
45          decimale = 0 ;
46          for ( i=0; i<num_cifre; i++)
47          {
48              if ( binario[i] == '1' )
49                  decimale = 2*decimale + 1 ;
50              else
51                  decimale = 2*decimale ;
52          }
53
54          /* STAMPA IL RISULTATO */
55          printf("Il valore decimale e': %d\n", decimale) ;
56      }
57      exit(0) ;
58  }
```

6 Esercizio: "Parola palindroma"

Scrivere un programma in linguaggio C che riceve in ingresso una parola inserita da tastiera. Si consideri che la parola può contenere sia caratteri maiuscoli che caratteri minuscoli, e complessivamente al massimo 30 caratteri. Il programma deve svolgere le seguenti operazioni:

- visualizzare la parola inserita
- aggiornare la parola in modo che tutti i caratteri siano minuscoli. Il programma deve visualizzare la parola ottenuta
- verificare se la parola è palindroma. Una parola è palindroma se può essere letta indifferentemente da sinistra verso destra e da destra verso sinistra. Ad esempio, le seguenti parole sono palindrome: `otto`, `madam`.

Soluzione

```
1   /* PROGRAMMAZIONE IN C */
2
3   /* File: palindroma.c */
4   /* Soluzione proposta esercizio "Parola palindroma" */
5
6   #include <stdio.h>
7   #include <stdlib.h>
8   #include <ctype.h>
9   #include <string.h>
```

```
10
11  int main(void)
12  {
13      const int MAXDIM = 30 ;      /* dimensione massima stringa di caratteri */
14
15      char parola[MAXDIM+1] ;      /* stringa di caratteri inserita */
16      int numcaratteri ;           /* numero di caratteri della stringa inserita */
17      int palindroma ;             /* flag per la ricerca */
18      int i, j ;                   /* indici dei cicli */
19
20
21      /* LEGGI LA STRINGA DI CARATTERI INSERITA DA TASTIERA */
22      printf("Inserisci una parola di al massimo %d caratteri: ", MAXDIM) ;
23      scanf("%s", parola) ;
24
25      /* VISUALIZZA LA STRINGA DI CARATTERI INSERITA */
26      printf("La parola inserita e': %s \n", parola) ;
27
28      /* LEGGI IL NUMERO DI CARATTERI DELLA STRINGA */
29      numcaratteri = strlen(parola) ;
30      printf("La parola contiene %d caratteri\n", numcaratteri) ;
31
32      /* CONVERTI TUTTI I CARATTERI DELLA STRINGA IN CARATTERI MINUSCOLI */
33      for ( i=0; i < numcaratteri ; i++ )
34          parola[i] = tolower(parola[i]) ;
35
36      /* VISUALIZZA LA STRINGA DI CARATTERI DOPO LA CONVERSIONE */
37      printf("La parola inserita scritta solo con caratteri in minuscolo e': %s\n",
38             parola) ;
39
40      /* VERIFICA SE LA STRINGA "parola" E' PALINDROMA */
41
42      /* INIZIALIZZA IL FLAG "palindroma". IL FLAG ASSUME I VALORI
43      -- "palindroma" E' UGUALE A 1 SE "parola"  E' PALINDROMA
44      -- "palindroma" E' UGUALE A 0 SE "parola"  NON E' PALINDROMA
45      */
46      palindroma = 1 ;
47
48      /* IL CICLO FOR SCANDISCE LA STRINGA DI CARATTERI "parola" E VERIFICA
49      SE E' PALINDROMA L'INDICE "i" SCORRE LA PRIMA META' DI "parola". L'INDICE
50      "j" SCORRE LA SECONDA META' DI "parola" PARTENDO DALL'ULTIMO CARATTERE.
51      LA RICERCA TERMINA QUANDO SI TROVA SI VERIFICA CHE LA STRINGA "parola"
52      NON E' PALINDROMA O QUANDO SONO STATI CONSIDERATI TUTTI I CARATTERI
53      DI "parola" */
54
55      for ( i=0, j=numcaratteri - 1 ;
56              i< numcaratteri/2 && palindroma==1;
57              i++, j-- )
58      {
59          if ( parola[i] != parola[j] )
60              palindroma = 0 ;
61      }
62
63      /* STAMPA DEL RISULTATO */
64      if ( palindroma == 1 )
65          printf("La parola e' palindroma\n") ;
```

```
66      else
67          printf("La␣parola␣non␣e'␣palindroma\n") ;
68
69      exit(0) ;
70  }
```

7 Esercizio: "Ricerca sottostringa"

Si scriva un programma in linguaggio C che riceva in ingresso due parole inserite da tastiera. Si consideri che ciascuna parola può contenere al massimo 30 caratteri. Il programma deve verificare se la seconda parola inserita è contenuta almeno una volta all'interno della prima parola (ossia se la seconda parola è una sottostringa della prima parola).

Soluzione

```
1   /* PROGRAMMAZIONE IN C */
2
3   /* File: ricercasottostringa_v1.c */
4   /* Soluzione proposta esercizio "Ricerca sottostringa" */
5
6   #include <stdio.h>
7   #include <stdlib.h>
8   #include <string.h>
9
10  int main(void)
11  {
12      const int MAXDIM = 30 ;            /* dimensione max stringa di caratteri */
13
14      char parola1[MAXDIM + 1] ;         /* prima stringa di caratteri */
15      char parola2[MAXDIM + 1] ;         /* seconda stringa di caratteri */
16      int lung_stringa1, lung_stringa2 ; /* lunghezza delle due stringhe */
17
18      /* LEGGI LA PRIMA PAROLA INSERITA DA TASTIERA */
19      printf ("Inserisci␣una␣parola␣di␣al␣massimo␣%d␣caratteri:␣", MAXDIM) ;
20      gets(parola1) ;
21
22      /* CALCOLA LA LUNGHEZZA DELLA PAROLA */
23      lung_stringa1 = strlen(parola1) ;
24
25      /* STAMPA LA PAROLA INSERITA */
26      printf("La␣parola␣%s␣contiene␣%d␣lettere\n", parola1, lung_stringa1) ;
27
28      /* LEGGI LA SECONDA PAROLA INSERITA DA TASTIERA */
29      printf ("Inserisci␣una␣parola␣di␣al␣massimo␣%d␣caratteri:␣", MAXDIM) ;
30      gets(parola2) ;
31
32      /* CALCOLA LA LUNGHEZZA DELLA PAROLA */
33      lung_stringa2 = strlen(parola2) ;
34
35      /* STAMPA LA PAROLA INSERITA */
36      printf("La␣parola␣%s␣contiene␣%d␣lettere\n", parola2, lung_stringa2) ;
37
38      /* VERIFICA SE "parola2" E' CONTENUTA IN "parola1" */
39      if ( lung_stringa1 < lung_stringa2 )
40          printf("La␣seconda␣parola␣e'␣piu'␣lunga␣della␣prima␣parola␣\n") ;
41      else
```

```
42      {
43          if ( strstr(parola1, parola2) != NULL )
44              printf("La␣seconda␣parola␣e'␣contenuta␣nella␣prima␣\n") ;
45          else
46              printf("La␣seconda␣parola␣non␣e'␣contenuta␣nella␣prima␣\n") ;
47      }
48      exit(0) ;
49  }
```

Soluzione alternativa

```
1   /* PROGRAMMAZIONE IN C */
2
3   /* File: ricercasottostringa_v2.c */
4   /* Soluzione proposta esercizio "Ricerca sottostringa" */
5
6   #include <stdio.h>
7   #include <stdlib.h>
8   #include <string.h>
9
10  int main(void)
11  {
12      const int MAXDIM = 30 ;            /* dimensione massima stringa di caratteri */
13
14      char parola1[MAXDIM + 1] ;         /* prima stringa di caratteri inserita */
15      char parola2[MAXDIM + 1] ;         /* seconda stringa di caratteri inserita */
16      int lung_stringa1, lung_stringa2 ; /* lunghezza delle due stringhe inserite */
17
18      int contenuto, finito ;            /* flag per la ricerca */
19      int i, j ;                         /* indici dei cicli */
20
21      /* LEGGI LA PRIMA PAROLA INSERITA DA TASTIERA */
22      printf ("Inserisci␣una␣parola␣di␣al␣massimo␣%d␣caratteri:␣", MAXDIM) ;
23      gets(parola1) ;
24
25      /* CALCOLA LA LUNGHEZZA DELLA PAROLA */
26      lung_stringa1 = strlen(parola1) ;
27
28      /* STAMPA LA PAROLA INSERITA */
29      printf("La␣parola␣%s␣contiene␣%d␣lettere\n", parola1, lung_stringa1) ;
30
31      /* LEGGI LA SECONDA PAROLA INSERITA DA TASTIERA */
32      printf ("Inserisci␣una␣parola␣di␣al␣massimo␣%d␣caratteri:␣", MAXDIM) ;
33      gets(parola2) ;
34
35      /* CALCOLA LA LUNGHEZZA DELLA PAROLA */
36      lung_stringa2 = strlen(parola2) ;
37
38      /* STAMPA LA PAROLA INSERITA */
39      printf("La␣parola␣%s␣contiene␣%d␣lettere\n", parola2, lung_stringa2) ;
40
41      /* VERIFICA SE "parola2" E' CONTENUTA IN "parola1" */
42      if ( lung_stringa1 < lung_stringa2 )
43          printf("La␣seconda␣parola␣e'␣piu'␣lunga␣della␣prima␣parola␣\n") ;
44      else
45      {
46          /* IL CICLO FOR ESTERNO SCORRE LA STRINGA "parola1".
```

```
47          PER OGNI CARATTERE "parola1[i]" IL CICLO FOR INTERNO ANALIZZA LA
48          LA SOTTOSTRINGA CONTENENTE I CARATTERI COMPRESI TRA "parola1[i]"
49          E "parola1[i+lung_stringa2-1]", E VERIFICA SE TALE SOTTOSTRINGA
50          E' UGUALE A "parola2" */
51
52          /* IL FLAG "finito==1" INDICA LA CONDIZIONE DI FINE RICERCA.
53          IL FLAG E' INIZIALIZZATO A 0 E VIENE ASSEGNATO A 1 SE "parola2" E'
54          CONTENUTA IN "parola1" */
55          finito = 0 ;
56          for ( i=0; i+(lung_stringa2-1)<lung_stringa1 && finito==0; i++ )
57          {
58              /* "j" E' L'INDICE DEL CICLO FOR INTERNO. VIENE UTILIZZATO PER
59              SCORRERE I CARATTERI DELLA SOTTOSTRINGA "parola2" E DELLA
60              SOTTOSTRINGA CONTENENTE I CARATTERI COMPRESI TRA "parola1[i]"
61              E "parola1[i+lung_stringa2-1]" */
62
63              /* IL FLAG "contenuto==1" INDICA CHE LE DUE SOTTOSTRINGHE SONO
64              UGUALI. IL FLAG E' INIZIALIZZATO A 1 E VIENE ASSEGNATO A 0 SE
65              ALMENO UN CARATTERE "parola1[i+j]" NELLA SOTTOSTRINGA E' DIVERSO
66              DAL CORRISPONDENTE CARATTERE "parola2[j]" */
67              contenuto = 1 ;
68              for ( j=0; j<lung_stringa2 && contenuto==1; j++ )
69              {
70                  if ( parola1[i+j] != parola2[j] )
71                      contenuto = 0 ;
72              }
73
74              /* SE AL TERMINE DEL CONFRONTO TRA LE DUE STRINGHE "contenuto" E'
75              ANCORA UGUALE A 1, ALLORA "parola2" E' CONTENUTA IN "parola1".
76              IL FLAG "finito" VIENE AGGIORNATO, E SI CONCLUDE LA RICERCA */
77              if ( contenuto==1 )
78                  finito = 1 ;
79          }
80      }
81
82      /* STAMPA IL RISULTATO */
83      if ( contenuto == 1 )
84          printf("La␣seconda␣parola␣e'␣contenuta␣nella␣prima␣\n") ;
85      else
86          printf("La␣seconda␣parola␣non␣e'␣contenuta␣nella␣prima␣\n") ;
87
88      exit(0) ;
89  }
```

8 Esercizio: "Sostituisci sottostringa"

Si scriva un programma in linguaggio C che riceva in ingresso due parole inserite da tastiera. Si consideri che ciascuna parola può contenere al massimo 30 caratteri. Il programma deve sostituire ogni occorrenza della seconda parola nella prima parola con una sequenza di caratteri `'*'`.

Ad esempio, inserite le parole `abchdfffchdtlchd` e `chd`, il programma deve visualizare la parola `ab***fff***tl***`.

Soluzione

```
1   /* PROGRAMMAZIONE IN C */
```

```c

/* File: sostituiscisottostringa.c */
/* Soluzione proposta esercizio "Sostituisci sottostringa" */

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

int main(void)
{
    const int MAXDIM = 30 ;            /* dimensione max stringa di caratteri */

    char parola1[MAXDIM + 1] ;         /* prima stringa di caratteri inserita */
    char parola2[MAXDIM + 1] ;         /* seconda stringa di caratteri inserita */
    int lung_stringa1, lung_stringa2 ; /* lunghezza delle due stringhe inserite */

    int contenuto ;                    /* flag per la ricerca */
    int i, j ;                         /* indici dei cicli */


    /* LEGGI LA PRIMA PAROLA INSERITA DA TASTIERA */
    printf ("Inserisci una parola di al massimo %d caratteri: ", MAXDIM) ;
    gets(parola1) ;

    /* CALCOLA LA LUNGHEZZA DELLA PAROLA */
    lung_stringa1 = strlen(parola1) ;

    /* STAMPA LA PAROLA INSERITA */
    printf("La parola %s contiene %d lettere\n", parola1, lung_stringa1) ;


    /* LEGGI LA SECONDA PAROLA INSERITA DA TASTIERA */
    printf ("Inserisci una parola di al massimo %d caratteri: ", MAXDIM) ;
    gets(parola2) ;

    /* CALCOLA LA LUNGHEZZA DELLA PAROLA */
    lung_stringa2 = strlen(parola2) ;

    /* STAMPA LA PAROLA INSERITA */
    printf("La parola %s contiene %d lettere\n", parola2, lung_stringa2) ;


    /* VERIFICA SE "parola2" E' CONTENUTA IN "parola1" */
    if ( lung_stringa1 < lung_stringa2 )
        printf("La seconda parola e' piu' lunga della prima parola \n") ;
    else
    {
        /* IL CICLO FOR ESTERNO SCORRE LA STRINGA "parola1".
        PER OGNI CARATTERE "parola1[i]" IL CICLO FOR INTERNO ANALIZZA LA
        LA SOTTOSTRINGA CONTENENTE I CARATTERI COMPRESI TRA "parola1[i]"
        E "parola1[i+lung_stringa2-1]", E VERIFICA SE TALE SOTTOSTRINGA
        E' UGUALE A "parola2" */

        for ( i=0; i+(lung_stringa2-1)<lung_stringa1; i++ )
        {
            /* "j" E' L'INDICE DEL CICLO FOR INTERNO. VIENE UTILIZZATO PER
```

```
58              SCORRERE I CARATTERI DELLA SOTTOSTRINGA "parola2" E DELLA
59              SOTTOSTRINGA CONTENENTE I CARATTERI COMPRESI TRA "parola1[i]" E
60              "parola1[i+lung_stringa2-1]" */
61
62              /* IL FLAG "contenuto==1" INDICA CHE LE DUE SOTTOSTRINGHE SONO
63              UGUALI.
64              IL FLAG E' INIZIALIZZATO A 1 E VIENE ASSEGNATO A 0 SE ALMENO UN
65              CARATTERE "parola1[i+j]" NELLA SOTTOSTRINGA E' DIVERSO DAL
66              CORRISPONDENTE CARATTERE "parola2[j]" */
67              contenuto = 1 ;
68              for ( j=0; j<lung_stringa2 && contenuto==1; j++ )
69              {
70                  if ( parola1[i+j] != parola2[j] )
71                      contenuto = 0 ;
72              }
73
74              /* SE AL TERMINE DEL CONFRONTO TRA LE DUE STRINGHE "contenuto" E'
75              ANCORA UGUALE A 1, ALLORA "parola2" E' CONTENUTA IN "parola1".
76              SOSTITUISCI ALLORA TUTTI I CARATTERI COMPRESI TRA "parola1[i]"
77              E "parola1[i+lung_stringa2-1]" CON IL CARATTERE '*' */
78              if ( contenuto==1 )
79              {
80                  for ( j=0; j<lung_stringa2; j++ )
81                      parola1[i+j] = '*' ;
82
83                  /*PER OTTIMIZZARE LA RICERCA SALTA NELLA STRINGA "parola1"
84                  LA SOTTOSEQUENZA DI ASTERISCHI APPENA INSERITA */
85                  i = i + lung_stringa2 - 1 ;
86              }
87          }
88      }
89
90      /* STAMPA IL RISULTATO */
91      printf("La␣parola␣risultante␣e'␣%s␣\n", parola1) ;
92
93      exit(0) ;
94  }
```