Informatica – I Facoltà di Ingegneria

Prova scritta del 23/06/2014

Sia dato uno schema rettangolare contenente lettere dell'alfabeto, **maiuscole**, memorizzato in un file di testo denominato "schema.txt". Si supponga che non vi siano errori di formato e che le linee del file abbiano tutte la stessa lunghezza e **non contengano spazi**. Le dimensioni dello schema non sono note in fase di compilazione, ma devono essere dedotte dal contenuto del file; comunque non possono eccedere le 30 righe e le 50 colonne.

Si realizzi un programma in linguaggio C che:

1. Chieda all'utente di inserire una parola (senza spazi) di al massimo 30 caratteri
2. Ricerchi tale parola nello schema (letto da file e caricato in memoria) nelle tre direzioni:
   - Orizzontale (da sinistra a destra)
   - Verticale (dall'alto al basso)
   - Diagonale (da in alto a sinistra a in basso a destra)
3. Se la parola è stata trovata, stampi a video la posizione iniziale e la posizione finale all'interno della matrice (con le coordinate della posizione in alto a sinistra pari a 1,1), altrimenti chieda nuovamente di inserire una parola.

Esempio

Dato il file "schema.txt" contenente uno schema 10x10 con il seguente testo (nel file non si sono spazi tra una lettera e l'altra):

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

Il programma procederà come segue (dove l'utente inserisce prima la parola "maiuscole" e poi "lettere"):

```
Inserire la parola da cercare: maiuscole
Parola 'maiuscole' non trovata.
Inserire la parola da cercare: lettere
Parola 'lettere' trovata con inizio in posizione 1,3 e fine in posizione 7,9.
```

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

/* TDE-20140623
Sia dato uno schema rettangolare contenente lettere dell'alfabeto, maiuscole, memorizzato in un file
di testo denominato "schema.txt". Si supponga che non vi siano errori di formato e che le linee del
file abbiano tutte la stessa lunghezza e non contengano spazi. Le dimensioni dello schema  non sono
note in fase di compilazione, ma devono essere dedotte dal contenuto del file; comunque non possono
eccedere le 30 righe e le 50 colonne.
Si realizzi un programma in linguaggio C che:
1.      Chieda all'utente di inserire una parola (senza spazi) di al massimo 30 caratteri
2.      Ricerchi tale parola nello schema (letto da file e caricato in memoria) nelle tre direzioni:
-       Orizzontale (da sinistra a destra)
-       Verticale (dall'alto al basso)
-       Diagonale (da in alto a sinistra a in basso a destra)
3.      Se la parola è stata trovata, stampi a video la posizione iniziale e la posizione finale
all'interno della matrice (con le coordinate della posizione in alto a sinistra pari a 1,1),
altrimenti chieda nuovamente di inserire una parola.

-------------------------------------------------
# Esempio

Dato il file "schema.txt" contenente uno schema 10x10 con il seguente testo (nel file non si sono
spazi tra una lettera e l'altra):
CYLXAIPAIT
OJGELALJED
LSUATJIPYN
OVLLKTNYTQ
NIJFASEEOS
NSPAZIERFJ
EXHBXCYIEZ
SCHEMAIGTS
VUZTOCOHSS
YSEOVJHEDB

Il programma procederà come segue (dove l'utente inserisce prima la parola "maiuscole" e poi
 "lettere"):
Inserire la parola da cercare: maiuscole
Parola 'maiuscole' non trovata.
Inserire la parola da cercare: lettere
Parola 'lettere' trovata con inizio in posizione 1,3 e fine in posizione 7,9.

*/

#define N   30
#define M   50

int main()
{
    FILE *fp;
    int i, j, k, z;
    int r, c;
    char m[N][M];
    char ch;
    char parola[N+1];
    int trovato;

    fp = fopen("schema.txt", "r");
    if(fp==NULL) {
        printf("Unable to open file schema.txt\n");
        exit(EXIT_FAILURE);
    }

    i = 0; j = 0;
    /* finchè non viene raggiunta la fine del file */
    while(!feof(fp)) {
        /* leggo un carattere alla volta */
        ch = fgetc(fp);
        /* se e' finita la riga,
           - incremento il contatore delle righe
           - riporto le coordinate a inizio riga: j=0
        */
        if(ch=='\n') {
            i++;
```

```
            /* mi segno quante colonne (j) ho riempito */
            c = j;
            j = 0;
        } else {
        /* altrimenti aggiungo il carattere letto alla riga corrente */
            m[i][j] = ch;
            j++;
        }
    }
    /* mi segno in r quante righe ho riempito */
    r = i;
    fclose(fp);

    for(i=0;i<r;i++) {
        for(j=0;j<c;j++) {
            printf("%c ", m[i][j]);
        }
        printf("\n");
    }

do{
    trovato = 0;

    printf("Inserire la parola da cercare: ");
    scanf("%s", parola);

    /* per ogni riga */
    for(i=0;i<r;i++){
        /* cerco se contiene parola */
        k = 0;
        for(j=0;j<c;j++) {
            /* cioè se ci sono tot caratteri adiacenti
               che coincidono con i caratteri di parola */
            if(m[i][j] == parola[k]) {
                k++;
                if(k==strlen(parola)) {
                    printf("Parola '%s' trovata ");
                    printf("con inizio in posizione %d,%d ", i+1, j-(k-1)+1);
                    printf("e fine in posizione %d,%d\n", i+1, j+1);
                }
            } else {
                k = 0;
            }
        }
    }

    /* per ogni colonna ... idem */
    for(j=0;j<c;j++) {
        k = 0;
        for(i=0;i<r;i++){
            if(m[i][j] == parola[k]) {
                k++;
                if(k==strlen(parola)) {
                    printf("Parola '%s' trovata ");
                    printf("con inizio in posizione %d,%d ", i-(k-1)+1, j+1);
                    printf("e fine in posizione %d,%d\n", i+1, j+1);
                }
            } else {
                k = 0;
            }
        }
    }

    /* per le diagonali e' piu' complicato perche'
       - le diagonali partono sia dalla prima cella di ogni colonna
         sia dalla prima cella di ogni riga
         - quindi spezzo la ricerca in due: parte superiore destra e parte inferiore sinistra
       - per cercare sulle diagonali devo incrementare i e j insieme (nello stesso for)
    */

    /* parte superiore destra */
    for(z=0;z<c;z++) {
        for(i=0,j=z;i<r && j<c;i++,j++){
            if(m[i][j] == parola[k]) {
```

```
                k++;
                if(k==strlen(parola)) {
                    printf("Parola '%s' trovata ");
                    printf("con inizio in posizione %d,%d ", i-(k-1)+1, j-(k-1)+1);
                    printf("e fine in posizione %d,%d\n", i+1, j+1);
                }
            } else {
                k = 0;
            }
        }
    }

    /* parte inferiore sinistra*/
    for(z=0;z<r;z++) {
        /* cerco da m[z][0] (prima colonna della matrice)
           in diagonale (incremento sia i sia j nello stesso ciclo */
        for(i=z,j=0;i<r && j<c;i++,j++){
            if(m[i][j] == parola[k]) {
                k++;
                if(k==strlen(parola)) {
                    printf("Parola '%s' trovata ");
                    printf("con inizio in posizione %d,%d ", i-(k-1)+1, j-(k-1)+1);
                    printf("e fine in posizione %d,%d\n", i+1, j+1);
                }
            } else {
                k = 0;
            }
        }
    }
  if(!trovato){ printf("Parola %s non trovata.\n", parola);}
} while(!trovato);
    return 0;
}
```