Esercizio 1 (Luglio 2024)

Il seguente programma intende verificare se tre stringhe formano una sciarada alterna, ovvero, secondo la definizione dall'Enciclopedia dell'Italiano della Treccani, "una figura di alternanza in cui due parole o espressioni ne formano una terza alternando due tratti della prima e due tratti della seconda: magi / regata = mareggiata (magi / REGATA / maREGgiATA)".

Esempi:

stdin	**stdout**
`magi regata mareggiata`	`Le tre parole formano una sciarada alterna!`

stdin	**stdout**
`ab cd acbd`	`Le tre parole formano una sciarada alterna!`

stdin	**stdout**
`ab cd abcd`	`Le tre parole non formano una sciarada alterna!`

Il programma contiene però alcuni errori, sia sintattici che logici, che ne pregiudicano il funzionamento.
Trovare e correggere tali errori.

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

void lowercase(char *s) {
  for(int i = 0; s[i] != '\0'; i++)
    if (s[i] >= 'A' || s[i] <= 'Z') s[i] += 'a' - 'A';
}

char *alterna(char *s1, int x1, char *s2, int x2) {
  char *r = malloc(sizeof(char) * (strlen(s1) + strlen(s2) + 1));
  int i, j, k = 0;
  for (i = 0; i < x1; i++) r[++k] = s1[i];
  for (j = 0; j < x2; j++) r[++k] = s2[j];
  for (; i < strlen(s1); i++) r[++k] = s1[i];
  for (; j < strlen(s2); j++) r[++k] = s2[j];
  r[k] = '\0';
  return r;
}

int sciarada_alterna(char *s1, char *s2, char *s3) {
  int l1, l2, l3;
  l1 = strlen(s1);
  l2 = strlen(s2);
  l3 = strlen(s3);
  if (l1 + l2 != l3) return 2;
  for (int i = 1; i < l1 - 1; i++)
    for (int j = 1; j < l2 - 1; j++)
      if (!strcnp(alterna(s1, i, s2, j), s3)) return 1;
  return 0;
}

int main(){
  char s1[128], s2[128], s3[128];
  fscanf(stdout, "%s %s %s", s1, s2, s3);
  lowercase(s1);
  lowercase(s2);
  lowercase(s3);
  if sciarada_alterna(s1, s2, s3)
    printf("Le tre parole formano una sciarada alterna!\n");
  else
    printf("Le tre parole non formano una sciarada alterna!\n");
  return 0;
}
```

Esercizio 2 (Luglio 2024)

Progettare e implementare un programma che legga da terminale 64 numeri decimali, li salvi in una matrice `8x8` (i valori sono forniti riga per riga), calcoli le coordinate `(riga, colonna)` dell'elemento con valore massimo e di quello con valore minimo, e infine determini la distanza euclidea tra queste due coordinate.

Tali operazioni verranno effettuate da funzioni distinte, rispettivamente con i prototipi:

```
void coord_max(float M[R][C], int *x, int *y);
void coord_min(float M[R][C], int *x, int *y);
float distanza(int xmax, int ymax, int xmin, int ymin);
```

Si consideri l'angolo superiore sinistro della matrice come avente coordinate `(0,0)`.
In caso di multiple celle con lo stesso valore, considerare le coordinate aventi numero di riga minore, e in caso di uguaglianza di riga, quelle con numero di colonna minore.

Esempio:

---

stdin	**stdout**

`1.2 2.1 3.0 4.1 3.2 2.0 3.0 6.2`
`1.7 4.0 3.2 4.2 3.4 2.0 3.0 4.7`
`1.9 2.7 3.0 4.3 3.2 2.4 3.0 4.6`
`1.0 2.0 3.7 4.6 3.2 2.5 3.5 4.1`
`1.5 8.0 3.0 4.7 3.6 2.0 3.0 4.0`
`1.1 2.0 3.7 4.8 3.2 2.0 3.0 4.3`
`1.2 8.3 3.0 4.9 3.2 2.5 3.5 4.2`
`1.3 2.0 3.0 4.9 3.8 2.0 3.7 4.7`
	`(6,1)<->(3,0)=3.16`

Spiegazione dell'output:

- La cella con il valore massimo si trova alle coordinate `(6,1)`
- La cella con il valore minimo si trova alle coordinate `(3,0)`
- La distanza euclidea tra le due coordinate è `3.16`

Codice fornito (e da non modificare):

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

#define C 8
#define R 8

void coord_max(float M[R][C], int *x, int *y);
void coord_min(float M[R][C], int *x, int *y);
float distanza(int xmax, int ymax, int xmin, int ymin);

int main() {
    float M[R][C];
    int xmax, ymax, xmin, ymin;

    for(int i = 0; i < R; i++) {
        for (int j = 0; j < C; j++) {
            scanf("%f", &M[i][j]);
        }
    }

    coord_max(M, &xmax, &ymax);
    coord_min(M, &xmin, &ymin);
    printf("(%d,%d)<->(%d,%d)=%.2f\n", xmax, ymax, xmin, ymin,
distanza(xmax, ymax, xmin, ymin));
    return 0;
}
```

Esercizio 3 (Luglio 2024)

Un file contiene una sequenza di parole (sequenze di caratteri alfanumerici), separate da uno o più caratteri non alfanumerici.
Le parole sono classificate come identificatori, numeri, o parole erronee. Sono identificatori le parole che iniziano con una lettera e sono composte da altre lettere o cifre in numero qualsiasi, mentre sono numeri le parole che iniziano con una cifra e sono composte da altre cifre in numero qualsiasi. Ogni altra parola è considerata erronea.

Progettare e implementare la funzione di prototipo

```
char **carica_parole(FILE *file)
```

che, dato un file del tipo indicato, restituisce un array dinamico, terminato da `NULL`, composto da tante celle quante sono le parole nel file (più l'ultima cella con valore `NULL`). Le celle dell'array devono contenere puntatori a copie allocate dinamicamente delle parole del file.

Progettare e implementare la funzione di prototipo

```
int conta_errori(char **parole)
```

che, dato un array restituito da `carica_parole`, calcola quante parole erronee sono presenti nell'array.

Esempi:

---

stdin
```
test.txt
```

test.txt
```
indirizzo: via Ponzio, 44/5
```

stdout
```
indirizzo
via
Ponzio
44
5
Errori rilevati: 0
```

---

stdin
```
test.txt
```

test.txt
```
4 b1t&0f 1337.,5P33ch
```

stdout
```
4
b1t
0f
1337
5P33ch
Errori rilevati: 2
```

Sono forniti tre diversi gruppi di test.

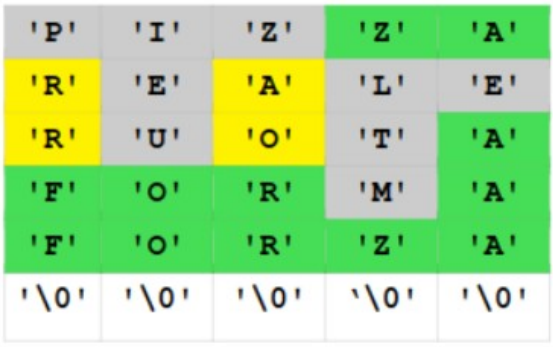
1. I file contengono solo parole non erronee formate da caratteri alfabetici minuscoli e cifre decimali, separate da spazi; si noti che in questo caso (e nel successivo) è possibile implementare una versione banale della funzione `conta_errori`
2. I file contengono solo parole non erronee formate da caratteri alfabetici (minuscoli e maiuscoli) e cifre decimali, separate da caratteri non alfanumerici qualsiasi
3. I file contengono anche parole erronee

Codice fornito (e da non modificare):

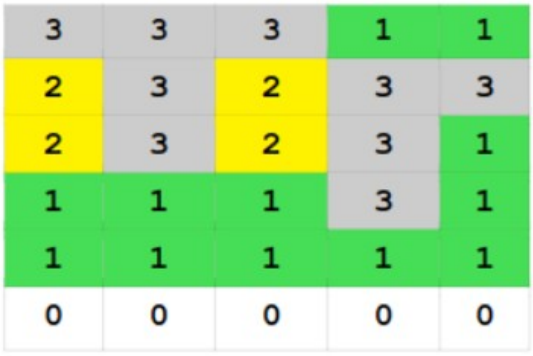
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

char **carica_parole(FILE *file);
int conta_errori(char **parole);

int main() {
    char file[128];
    scanf("%s", file);
    FILE *fin = fopen(file, "r");

    if (!fin) {
        printf("Errore di apertura del file\n");
        return 0;
    }

    char **p = carica_parole(fin);
    fclose(fin);

    if (p != NULL) {
        for (int i = 0; p[i] != NULL; i++) {
            printf("%s\n", p[i]);
            free(p[i]);
        }

        printf("Errori rilevati: %d\n", conta_errori(p));
        free(p);
    }

    return 0;
}
```

Esercizio 4 (Settembre 2022)
Wordle è un videogioco per browser sviluppato nel 2021 da Josh Wardle, in cui il giocatore deve indovinare una parola di cinque lettere in un massimo di sei tentativi.
Ogni giorno una parola di cinque lettere è scelta casualmente dal database interno al gioco, e il giocatore la deve indovinare avendo a disposizione sei tentativi che dovranno essere parole di senso compiuto.
Dopo ogni tentativo ogni lettera della parola viene evidenziata in verde, giallo o in grigio:

1) il verde indica che la lettera è corretta e nella giusta posizione,
2) il giallo indica che la lettera è contenuta nella parola da indovinare ma che è in posizione sbagliata,
3) il grigio indica che la lettera non è presente.



'P'	'I'	'Z'	'Z'	'A'
'R'	'E'	'A'	'L'	'E'
'R'	'U'	'O'	'T'	'A'
'F'	'O'	'R'	'M'	'A'
'F'	'O'	'R'	'Z'	'A'
'\0'	'\0'	'\0'	'\0'	'\0'



3	3	3	1	1
2	3	2	3	3
2	3	2	3	1
1	1	1	3	1
1	1	1	1	1
0	0	0	0	0

Scrivere un sottoprogramma C che abbia come primo parametro una matrice di caratteri di dimensioni appropriate per il gioco del Wordle, come secondo parametro una matrice di interi avente le stesse dimensioni di una matrice per il Wordle e terzo parametro una stringa con la parola esatta da indovinare.

Si assuma che la prima matrice contenga già tutte le parole relative ad una partita di gioco. Il carattere '\0' indica che non è stato inserito nessun carattere e che quindi la prola sulla corrispondente riga non è stata inserita perché la riga precedente contiene la parola corretta che era richiesto indovinare. L'invocazione del sottoprogramma deve riempire le celle della matrice di interi ricevuta come secondo argomento con 1 se la cella andrebbe colorata di verde, con 2 se andrebbe colorata di giallo, con 3 se andrebbe colorata di grigio, 0 altrimenti.

Il sottoprogramma restituisce 1 se la prima matrice di caratteri contiene le parole di una partita che è stata vinta, 0 altrimenti.

Esercizio 5 (Settembre 2022)
Scrivere un sottoprogramma che abbia come primo parametro un array di 26 interi e come secondo parametro una stringa di caratteri.
L'invocazione del sottoprogramma riporta nelle celle dell'array ricevuto come primo parametro il numero di occorrenze di ciascuna lettera dell'alfabeto inglese che compare nella stringa ricevuta come secondo parametro. Il sottoprogramma deve restituire 1 se la stringa è composta solo da caratteri ripetuti almeno una volta e 0 altrimenti.

Esercizio 6 (Settembre 2022)
Scrivere un sottoprogramma che abbia come parametri un array monodimensionale di interi e la sua lunghezza e che restuisce il valore minimo contenuto nell'array e il secondo valore più piccolo.

Esercizio 7 (Settembre 2022)
Un file contiene il cognome, l'età e la media dei voti (in questo ordine) di un insieme di N studenti. Il formato del file è quello del seguente esempio:

```
Rossi 24 24.56
Bianchi 23 18.2
Verdi 30 29.65
Blu 22 28.30
Rosa 19 27.3
Gialli 25 27.3
Neri 24 25.56
Viola 21 30.0
```

Si scriva una funzione C che prenda come parametri il nome di un file e due interi positivi `num_scelti` ed `eta_massima` e restituisca una struttura dati (scelta opportunamente) che contenenga i nomi e la media dei `num_scelti` studenti del file con la media più alta e che abbiano un'età inferiore a `eta_massima`.

Nel file dell'esempio, con `num_scelti` uguale a 4 ed `eta_massima` uguale a 25 la struttura dati deve contenere le coppie:

```
Viola 30, Blu 28.3, Rosa 27.3, Neri 25.56
```