www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari

```
/*
 * Corso di Fondamenti di Informatica
 * Esercizio:
 * tema d'esame (informatica 1 per allievi informatici).
 */

/* NOTA IMPORTANTE: i commenti non sono completi perché si fa riferimento al testo del tema d'esame, qui di seguito
riportato.

Esercizio A (modificato per eliminare i riferimenti a funzioni e liste dinamiche). La soluzione della versione non
modificata dell'esercizio verra' illustrata più avanti nel corso.
Un testo e' rappresentato come array di 100 caratteri. I caratteri utili nell'array sono quelli che precedono il (primo)
carattere '\0'. Dati due testi, si vuole identificare la più lunga sequenza S di caratteri consecutivi (tra i quali possono
comparire eventuali spazi) che e' contenuta in entrambi i testi.
Ad esempio, dati i testi "alternativamente" e "eternita", la più lunga sequenza contenuta in entrambi e' "tern" di lunghezza
4, mentre le sequenza "te" -anch'essa contenuta in entrambi- e' di lunghezza 2.  Dunque S sarà la sequenza "tern".
Detta L la lunghezza della sequenza S, se esistono più sequenze di lunghezza L  comuni ai due testi si utilizzi un criterio
qualsiasi per scegliere una sola tra esse.

Si scriva un programma C che, dati due testi definiti all'interno del programma stesso, trovi la sequenza di lunghezza
massima contenuta in entrambi i testi, la memorizzi e la stampi. La sequenza restituita e' anch'essa di tipo testo, pertanto
va memorizzata sotto forma di array di caratteri terminati da '\0'.
*/


#include <stdio.h>
#define MAX_LUNG_TESTO 100
/* massima lunghezza che ciascuno dei due testi da confrontare può avere,
 * incluso il carattere di terminazione '\0': dunque L <= MAX_LUNG_TESTO - 1 */

/* NOTA: nel linguaggio C si usano array di caratteri per contenere STRINGHE,
 * ovvero sequenze di caratteri. Per definizione, in C una stringa e' un array
 * di char contenente una sequenza di zero o più caratteri seguiti da un
 * carattere '\0', usato come TERMINATORE. Perciò se la stringa da memorizzare
 * e' lunga N caratteri, occorre utilizzare un array di almeno N+1 celle, in modo
 * da poter correttamente inserire il terminatore (che deve essere presente
 * anche nel caso la stringa non contenga caratteri utili).
 * Tutte le funzioni delle librerie standard del C (stdio.h, string.h, ...) che
 * lavorano su stringhe (ad esempio printf, strcpy, strlen,...) si aspettano che
 * la suddetta convenzione sulla rappresentazione di stringhe di caratteri
 * attraverso array di caratteri venga rispettata dagli argomenti loro passati,
```

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari

```
 * e a loro volta i loro valori di uscita rispettano tale convenzione. Ciò vale
 * in particolare per la funzione printf, quando usata con il carattere di
 * conversione %s nella stringa di controllo.
 * Quali caratteri seguano il '\0' nell'array che contiene una stringa e', dal
 * punto di vista delle funzioni C che operano su stringhe, indifferente. */

int main()
{
  char TestoA[MAX_LUNG_TESTO] = "alternativamente";
  char TestoB[MAX_LUNG_TESTO] = "eternita";
  /* i due testi da confrontare; notare che il testo dell'esercizio NON
   * richiedeva che i testi venissero immessi da tastiera */
  /* NOTA: inizializzare un array in questo modo inserisce nell'array tutti
   * i caratteri della stringa tra doppi apici PIU' UN CARATTERE '\0' dopo
   * di essi. Pertanto TestoA e TestoB rispettano i vincoli imposti dal
   * testo dell'esercizio. */

  char TestoComune[MAX_LUNG_TESTO];
  /* il testo comune a TestoA e TestoB, da trovare */
  int CursoreA, CursoreB, CursoreComune;
  /* CursoreA e CursoreB sono usati per scorrere i caratteri dei due testi;
         * inoltre nella fase finale CursoreA e' usato per scorrere gli elementi di
         * TestoA che costituiscono il testo comune. CursoreComune e' usato per
         * scorrere gli elementi di TestoComune */
  int CursoreALocale, CursoreBLocale;
  /* per ogni coppia di posizioni di CursoreA e CursoreB scorrono i caratteri
   * successivi dei due testi per determinare se coincidono */
  int InizioTestoComune;
  /* indice (in TestoA) del carattere iniziale del testo comune più lungo
   * trovato finora */
  int MaxLunghezza;
  /* lunghezza del più lungo testo comune ai due testi originali finora
   * trovato */
  int LunghezzaLocale;
  /* lunghezza del frammento di testo comune attualmente in esame */

      /* scorriamo il testo A un carattere alla volta; per ogni carattere di A,
       * scorriamo B per trovare lo stesso carattere; ogni volta che lo troviamo,
       * procediamo parallelamente sui due testi, un carattere alla volta, per
       * determinare quanto e' lungo il tratto coincidente; */
```

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari

```
MaxLunghezza = 0;
CursoreA = 0;

while ( '\0' != TestoA[CursoreA] )
  /* finché non si e' raggiunta la fine del testo A */
  {
    CursoreB = 0;

    while ( '\0' != TestoB[CursoreB] )
                      /* finché non si e' raggiunta la fine del testo B */
                              {
                                if ( TestoB[CursoreB] == TestoA[CursoreA] )
                                  /* se i caratteri correnti dei due testi sono uguali... */
                                  {
                                /* ...inizia il confronto dei caratteri successivi */
                                    LunghezzaLocale = 1;
                                    CursoreALocale = CursoreA+1;
                                    CursoreBLocale = CursoreB+1;

          while ( ('\0' != TestoA[CursoreALocale]) &&
          ('\0' != TestoB[CursoreBLocale]) &&
          (TestoA[CursoreALocale] == TestoB[CursoreBLocale]) )
          {
            ++LunghezzaLocale;
            ++CursoreALocale;
            ++CursoreBLocale;
          }
                              /* a questo punto LunghezzaLocale rappresenta la lunghezza del
                                                  * testo comune che parte da CursoreA (in TestoA) e CursoreB
                                                  * (in TestoB) */

                              if ( LunghezzaLocale > MaxLunghezza)
                                                 /* se il testo comune appena trovato e' il più lungo trovato
                                                  * finora */
                                                        {
                                                          InizioTestoComune = CursoreA;
                                                          MaxLunghezza = LunghezzaLocale;
                                                        }
                                        }
                              ++CursoreB;
                              }
```

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari

```
                ++CursoreA;
              }

  /* a questo punto InizioTestoComune indica il carattere iniziale (nel testo
   * A) del testo comune e MaxLunghezza e' la lunghezza del testo comune. Tale
   * testo va copiato in TestoComune */

  /* inizio fase di copia del testo comune in TestoComune: */
  CursoreComune = 0;
  CursoreA = InizioTestoComune;

  while (CursoreA < InizioTestoComune + MaxLunghezza)
    {
      TestoComune[CursoreComune] = TestoA[CursoreA];
      ++CursoreA;
      ++CursoreComune;
    }

  TestoComune[CursoreComune] = '\0';
  /* inserisce il carattere di terminazione in TestoComune */

  /* Nota: il ciclo non e' stato eseguito se MaxLunghezza = 0, ovvero se non
   * esiste alcun testo comune. In tal caso TestoComune[0] = '\0',
   * conformemente a quanto richiesto dall'esercizio: infatti in questa
   * situazione TestoComune deve contenere una stringa vuota */

  printf("\nIl testo comune e': '%s'\n\n", TestoComune);

  return(0);
}
```

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari