

Laboratorio di algoritmi e strutture dati

Docente: Violetta Lonati

Puntatori - esercizi da svolgere in laboratorio*

1 Array frastagliato

Considerate il codice contenuto nel file frastagliato.c.

```
1 #include <stdio.h>
2 #include <string.h>
3
4 void f( char *a[], int n, int i ) {
5     if ( i < 0 || i > n-1 )
6         return;
7
8     char *p = a[i];
9     a[i] = a[n-1];
10    a[n-1] = p;
11    return;
12 }
13
14 int main( int argc, char **argv ) {
15     char **q;
16     int i;
17
18     scanf( "%d", &i );
19     f( argv + 1, argc - 1, i );
20
21     for ( q = argv + 1; q < argv + argc; q++ )
22         printf( "%s ", *q );
23     printf( "\n" );
24     return 0;
25 }
```

Analizzate il codice sorgente e rispondete per iscritto alle seguenti domande. Se avete dubbi, potete testarlo, eseguendolo su casi di input significativi e modificandolo.

1. Di che tipo è la variabile `p`? *Suggerimento:* provate a definire con `typedef` un tipo `String` sinonimo di `char *`
2. Per cosa viene usata la variabile `p`?
3. Perché il primo argomento dell'invocazione di `f` alla riga 19 è `argv+1` e non `argv`?
4. Come mai il primo parametro della funzione `f` è dichiarato con `char *a[]` mentre il primo argomento passato alla funzione alla riga 19 è `argv` che è dichiarato alla riga 16 con `char** argv`?
5. Senza eseguire il programma al computer, stabilite cosa stampa se viene eseguito con il comando `./a.out zero uno due tre quattro ultimo`

*Ultima modifica 31 ottobre 2019

6. Date un nome più significativo alla funzione `f`.
7. Sarebbe corretto sostituire le righe 8-10 con le righe seguenti? Giustificate la risposta.

```
strcpy( *p, a[i] );
strcpy( a[i], a[n-1] );
strcpy( a[n-1], p );
```

8. Riassumete con una frase cosa fa il programma.

2 Alfabeto farfallino

Quando la vostra docente di laboratorio di algoritmi era bambina, usava a volte, per comunicare con le sue amiche, uno speciale alfabeto, detto *alfabeto farfallino*. L'alfabeto farfallino consiste nel sostituire, a ciascuna vocale, una sequenza di tre lettere della forma vocale-f-vocale. Per esempio, alla lettera *a* viene sostituita la sequenza *afa*, alla lettera *e* la sequenza *efe* e così via.

1. Scrivete un programma, di nome `farf` che, ricevendo come argomento (sulla riga di comando) una parola, ne stampi la traduzione in alfabeto farfallino. Potete assumere che la stringa in input non contenga lettere maiuscole.

Nota: Il programma deve stampare la parola in alfabeto farfallino, ma non è richiesto che la parola tradotta sia memorizzata come stringa.

Esempio di funzionamento

```
$ ./farf mamma
mafammafa
$ ./farf aiuola
afaifiufuofolafa
$ ./farf farfalla
fafarfafallafa
```

2. Modificate il programma in modo che accetti più parole sulla riga di comando.

3 I due valori più grandi

1. Scrivete una funzione con prototipo

```
void max_secondmax ( int a[], int n, int *max, int *second_max )
```

che, dato un array `a` di lunghezza `n`, individui il valore più grande in `a` e il secondo elemento per grandezza in `a`, e li memorizzi negli spazi di memoria puntate da `max` e `second_max`.

Suggerimento: i parametri `max` `secondmax` sono da utilizzare in modo simile ai parametri `h`, `m` e `s` della funzione `split_time`.

2. Testate il vostro programma invocandolo da un `main` strutturato secondo quanto indicato sopra. Notate che gli argomenti passati in corrispondenza di `max` e `secondmax` devono essere indirizzi di memoria utili (ad esempio indirizzi di variabili dichiarate opportunamente).
3. Rappresentate graficamente lo stato della memoria del programma subito prima dell'invocazione della funzione e subito prima del `return`, riportando in aree diverse dei disegni la porzione di memoria relativa alla funzione `main` e quella relativa alla funzione `max_secondmax`.

Nota

Alcuni degli esercizi rimanenti richiedono di scrivere funzioni che manipolano cioè array frastagliati, e cioè con un prototipo del tipo `... f (char *s[], ...)`. Inizializzare tali array da standard input non è banale; consiglio quindi di fare i test usando un main così strutturato:

```
int main( void ) {
char *dict[] = { "ciao", "mondo", "come", "funziona", "bene", "il", "programma" };
int lun = 7, pos;
... f( ... ) ...
}
```

Modificando le inizializzazioni di `dict` e `lun` è possibile testare la funzione con argomenti diversi.

In alternativa le funzioni scritte possono essere testate passando come argomento il vettore frastagliato `argv`. Ricordate che `argv[0]` ha per valore il nome del programma, quindi l'argomento da passare è più precisamente `argv+1`.

Inoltre, ricordate che per effettuare confronti tra stringhe, potete usare la funzione `strcmp` dichiarata nel file di intestazione `string.h`.

4 La parola più piccola

1. Scrivete una funzione con prototipo `int smallest_word_index(char *s[], int n)` che, dato un array `s` lungo `n` di stringhe, restituisca l'indice della parola più piccola (secondo l'ordine alfabetico) contenuta nell'array.

Per effettuare confronti tra stringhe, potete usare la funzione `strcmp` dal file di intestazione `string.h`.

2. Definite usando `typedef` un nuovo tipo `String` sinonimo di `char *` e modificate il programma usando `String` ovunque sia opportuno.
3. Partendo dalla funzione `smallest_word_index` scrivete una nuova funzione con prototipo
`String *smallest_word_address(String s[], int n)`
che, dato un array `s` di `n` stringhe, restituisca l'indirizzo dell'elemento più piccolo dell'array (sempre secondo l'ordine alfabetico).

Suggerimento: riguardate l'esercizio "Puntatore al minimo" e osservate le somiglianze tra i prototipi delle funzioni `smallest` e `smallest_word_address`.

4. Riscrivete il prototipo della funzione `smallest_word_address` senza usare il tipo `String`.

5 La parola minima e la parola massima

Scrivete una funzione con prototipo

```
void smallest_largest( char *s[], int n, char **smallest, char **largest )
```

che, dato un array `s` lungo `n` di stringhe, trovi gli elementi minimo e massimo nell'array (secondo l'ordine alfabetico) e memorizzi gli indirizzi delle loro posizioni nel vettore `s` negli indirizzi di memoria puntati rispettivamente da `smallest` e da `largest`.

Suggerimenti: vedete somiglianze con la funzione `max_secondmax`? Prima di cominciare rappresentate graficamente quale deve essere lo stato della memoria subito dopo l'invocazione della funzione e subito prima del return.

6 Tre volte tre - Appello del 4 luglio 2018

Considerate il codice contenuto nel file `treVolte-bacata.c`. Esaminate il programma, rilevate gli errori e correggeteli. Per ogni correzione, annotate un commento che spiega qual era l'errore e come l'avete corretto.

Il programma deve ricevere una sequenza di stringhe da linea di comando, invocare la funzione `treVolte` per individuare la terza stringa, tra quelle lette, che contiene almeno 3 volte la lettera 'e' (minuscola) e scambiarla con la prima, quindi stampare le stringhe nel nuovo ordine. Se non esiste nessuna stringa che soddisfa la proprietà precedente, il programma deve stampare le stringhe nell'ordine in cui apparivano nella linea di comando. Ad esempio, eseguendo il programma con comando

```
./a.out certamentissimamente signora non vorrei essere scortese posso ripetere le istruzioni  
il programma deve stampare  
ripetere signora non vorrei essere scortese posso certamentissimamente le istruzioni
```

NOTA: effettuate le modifiche necessarie a correggere gli errori ma *senza stravolgere il programma*.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <ctype.h>
4 #include <string.h>
5
6 #define N 100
7
8 void treVolte( char *a[], int n ) {
9     char **p, *q;
10    int conta = 0;
11
12    for ( p = a; p < a + n; p++ ) {
13        int conta_e = 0;
14        char *c;
15        c = *p;
16
17        while ( *c ) {
18            if ( *c == 'e' )
19                conta_e++;
20            if ( conta_e == 3 ) {
21                conta++;
22            }
23            c++;
24        }
25        if ( conta == 3 ) {
26            strcpy( q, p );
27            strcpy( p, a[0] );
28            strcpy ( a[0], q );
29            return;
30        }
31    }
32 }
33
34 int main( int argc, char **argv ) {
35     treVolte( argv + 1, argc - 1 );
36
37     for ( char** p = argv; p < argv + argc; p++ )
38         printf( "%s ", *p );
39     printf( "\n" );
40     return 0;
41 }
```
