Laboratorio di Programmazione: Linguaggio C
Lezione 18 del 31 marzo 2014

Damiano Macedonio

Esercizio 1

Scrivere un programma C che implementa una rubrica.

Una `rubrica` è una struttura formata da un array di strutture `contatto` (max. 10) e un intero rappresentante il numero di contatti inseriti.

Una struttura `contatto` è formata da: una stringa `nome` (max. 128 caratteri), una stringa `numero` (max. 128 caratteri). La definizione della struttura va messa nel file header `rubrica.h`.

Sulla struttura `rubrica` sono definite le seguenti funzioni (file `rubrica.h` + `rubrica.c`):

`struct rubrica aggiungi_contatto(struct rubrica r, struct contatto c)` Se vi è ancora spazio aggiunge il contatto `c` alla rubrica `r`, altrimenti ritorna un messaggio di errore.

`void elenca_contatti(struct rubrica r)` Stampa tutti i contatti contenuti nella rubrica.

`void trova_numero(struct rubrica r, char nome[])` Stampa il numero di telefono associato al contatto `nome` oppure un messaggio di errore se il nome non è presente in rubrica.

Le funzioni che richiedono all'utente una stringa e l'operazione da eseguire vanno isolate in un file `input.c` (+ `input.h`).

`void chiedi_stringa(char desc[], char s[])` Richiede all'utente una stringa di max 128 caratteri da salvare nell'array `s`. Usare la stringa `desc` per personalizzare il messaggio di richiesta.

`char chiedi_operazione()` Stampa le operazioni possibili (tra cui uscire) e richiede all'utente quale vuole eseguire.

Esercizio 2 [Cifrario di Cesare]

Scrivere un programma C che implementa un *cifrario di Cesare.* Il cifrario di Cesare è uno degli algoritmi crittografici più antichi. È anche detto cifrario a sostituzione: viene fissata una chiave k e il testo cifrato si ottiene sostituendo ciascuna lettera del testo in chiaro con la lettera che si trova k posizioni dopo nell'alfabeto (*in modo circolare*).

Esempio Fissiamo la chiave $k = 3$.

Corrispondeza tra caratteri:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c

Esempio di codifica:

P	r	o	g	r	a	m	m	a	z	i	o	n	e	1
S	u	r	j	u	d	p	p	d	c	l	r	q	h	1

Il programma deve essere composto da

- `cifrario.h` + `cifrario.c` contenenti rispettivamente la dichiarazione e la definizione delle funzioni per la cifratura (encode) e decifratura (decode).

  `void encode(char s[], int k)` Usata per cifrare la stringa `s` usando come chiave il valore `k`. La stringa risultato viene memorizzata in `s`. Devono essere rispettate le maiuscole e minuscole della parola originale. I caratteri che non sono lettere dell'alfabeto, non devono subire alcuna modifica

  `void decode(char s[], int n)` usata per decifrare la stringa `s` usando come chiave il valore `k`. La stringa risultato viene memorizzata in `s`.

- `main.c` che richiede all'utente una stringa (vedi funzione `readLine()`), un intero da usare come chiave e un'operazione (`E`/`D`) e ritorna la stringa cifrata/decifrata. Il file `main.c` può contenere funzioni ausiliarie.

Esercizio 3 [Ordinamento di Stringhe]

Scrivere un programma C che richiede all'utente di digitare 10 stringhe di lunghezza massima 128 e stampa a video le stringhe ordinate.

- Le stringhe vanno memorizzate in un array.
- Per eseguire l'ordinamento, potete usare uno degli algoritmi di ordinamento visti.

Il programma deve essere composto da

- `sort.h` + `sort.c` contenenti rispettivamente la dichiarazione e la definizione della funzione per l'ordinamento.

  `void sort(char s[][129], int n)` Per il confronto tra stringhe usare la funzione `strcmp` della libreria standard (ricordarsi di `#include <string.h>`). Il parametro `n` rappresenta il numero di stringhe nell'array `s` e permette di rendere la funzione generica. *Attenzione:*
  - `char s[n][m]` è un array di `n` stringhe, ciascuna di lunghezza massima `m`; l'elemento `s[i]` è a tutti gli effetti una stringa di lunghezza massima `m`.
  - per assegnare un array ad un altro array è necessario copiare i singoli elementi; bisogna quindi implementare una funzione!

- `main.c` richiede all'utente 10 stringhe e le ristampa a video nel corretto ordine lessicografico.