FONDAMENTI DI INFORMATICA

Prof. Maristella Matera - A.A. 2024/2025

Laboratorio di Programmazione in C

Laboratorio 4

Es. 1 Gioco di prestigio: il mistero delle 21 carte

■ *Gestione dei file, struct, funzioni*

Il seguente programma definisce al suo interno la struttura `Carta` che rappresenta una carta da gioco. La struttura è costituita dal valore della carta (singolo carattere) e dal seme (singolo carattere).

Se il valore della carta è una lettera (`J`, `Q`, `K`, `A`), questa deve essere maiuscola (per semplicità, rappresentare il `10` come `D`). I semi delle carte sono indicati con la notazione italiana (`C` = Cuori, `Q` = Quadri, `F` = Fiori, `P` = Picche).

Il programma estrae casualmente 21 carte da un mazzo di 52 carte da gioco. Le mostra a video e chiede all'utente di pensare a una delle carte. Successivamente le divide in tre mazzetti sullo schermo e chiede all'utente in quale di questi è presente la carta pensata. Questo procedimento viene ripetuto tre volte in totale.

Al termine il programma rivela a video la carta pensata dall'utente (magia!).

Spiegazione del trucco ed esempio di esecuzione

Partendo dalla prima carta in cima al mazzo, i mazzetti sono ottenuti disponendo una carta alla volta da sinistra a destra su tre colonne. A ogni iterazione, il mazzetto contenente la carta scelta deve essere posizionato in mezzo agli altri due, ottenendo un nuovo mazzo. Al termine delle tre iterazioni, la carta scelta occuperà l'undicesima posizione nel mazzo.

Estraiamo 21 carte dal mazzo:

`DP QF 4P 6P 2C KP QP 9C QQ 7C KQ 3P 6Q 8F 9F 3C KF AQ 3F DF 5Q`

Ipotizziamo che lo spettatore scelga **l'asso di quadri** (`AQ`):

Prima iterazione: `AQ` finisce in mazzetto 3	**Seconda iterazione:** `AQ` finisce in mazzetto 1	**Terza iterazione:** `AQ` finisce in mazzetto 3
`DP QF 4P`	`DP 6P QP`	`6P 6Q 4P`
`6P 2C KP`	`7C 6Q 3C`	`3P 5Q 9C`
`QP 9C QQ`	`3F 4P KP`	`KF DP 7C`
`7C KQ 3P`	`QQ 3P 9F`	`3F QQ AQ`
`6Q 8F 9F`	`AQ 5Q QF`	`2C 8F QP`
`3C KF AQ`	`2C 9C KQ`	`3C KP 9F`
`3F DF 5Q`	`8F KF DF`	`QF KQ DF`

Mazzo finale: `AQ` finisce in posizione 11

`6P 3P KF 3F 2C 3C QF 4P 9C 7C AQ QP 9F DF 6Q 5Q DP QQ 8F KP KQ`

Il programma, implementando opportune funzioni, deve:

- aprire il mazzo di 52 carte da gioco contenuto nel file binario `mazzoPolimi.bin` fornito con questo testo e memorizzarlo in una struttura adeguata
- estrarre a caso 21 carte dal mazzo inserendole in un array di carte
- salvare il mazzo di carte ottenuto in un file binario chiamato `mazzoStudente.bin`
- eseguire il gioco di prestigio all'utente secondo il procedimento riportato sopra

Suggerimenti

- per la generazione di numeri pseudocasuali è possibile utilizzare la funzione `rand()` unitamente alla funzione `srand()`. Entrambe le funzioni sono contenute in `stdlib.h`.
- La funzione `srand()` prende come unico parametro un numero intero che fa da "seme" per l'algoritmo di generazione di numeri pseudocasuali (come seme può essere utilizzato il numero di secondi trascorsi dal 1 gennaio 1970, restituito dalla funzione `time()` contenuta in `time.h`) e va eseguita prima della chiamata a `rand()`.
- Per ottenere il valore `int i` rappresentato da un carattere numerico `char c` si può utilizzare l'operazione `i = c - '0'`.
- Un interessante approfondimento sul principio matematico del trucco (con esecuzione del gioco) è disponibile qui: `https://www.youtube.com/watch?v=d7dg7gVDWyg`

Es. 2 Partitina a Mastermind?

■ *Cicli, stringhe, condizioni, funzioni*

Sviluppare un programma in linguaggio C che permetta di giocare al gioco del Mastermind. Il programma deve generare casualmente una sequenza di 4 caratteri numerici (che possono essere ripetuti all'interno della sequenza), e il giocatore ha a disposizione un massimo di 10 tentativi per indovinarla.

Per ogni tentativo, il giocatore deve inserire una sequenza di 4 numeri. Il programma mostrerà il numero di tentativi rimanenti e fornirà un riscontro per ciascun carattere della sequenza inserita, utilizzando le seguenti convenzioni:

- **0**: Il carattere è presente nella sequenza generata ed è nella posizione corretta.
- **+**: Il carattere è presente nella sequenza generata ma si trova in una posizione diversa.
- **–**: Il carattere non è presente nella sequenza generata.

Il programma termina in uno dei seguenti casi:

- Il giocatore indovina la sequenza corretta: il programma stampa un messaggio di vittoria.
- Il giocatore esaurisce i tentativi: il programma stampa la sequenza corretta che doveva essere indovinata e un messaggio di sconfitta.

Verificare che la sequenza inserita dal giocatore contenga esattamente 4 caratteri numerici, e assicurarsi che il programma gestisca correttamente l'input non valido, richiedendo una nuova sequenza in caso di errore.

Suggerimenti:

- Utilizzare la funzione `rand()` per generare i caratteri della sequenza in modo casuale, senza dimenticare di inizializzare il generatore di numeri casuali con `srand(time(NULL))`. Entrambe le funzioni sono contenute in `stdlib.h`.
- Per eseguire la ricerca di un carattere all'interno di una stringa, potete utilizzare la funzione `strchr()` contenuta nella libreria `string.h`, che prende in input la stringa da verificare e il carattere da cercare.
- Per verificare se un carattere è di tipo numerico, alfanumerico, alfabetico e così via potete utilizzare le funzioni condizionali presenti nella libreria `ctype.h`.

Es. 3 Libreria custom per stringhe

■ *Programmazione modulare, stringhe*

Creare una libreria personalizzata `stringsTools.h` che contenga le seguenti funzioni riutilizzabili in altri programmi:

- `void invertiStringa(char *str)`
  inverte una stringa passata come argomento
- `void toUppercase(char *str)`
  converte tutti i caratteri di una stringa in maiuscolo
- `void rimuoviSpazi(char *str)`
  elimina tutti gli spazi bianchi di una stringa
- `int isPalindromo(char *str)`
  verifica se la stringa è un palindromo
- `int contaVocali(char *str)`
  conta il numero di vocali presenti in una stringa

Le prime tre funzioni modificano la stringa originale ricevuta come argomento.

Scrivere un programma `libreria_stringhe.c` includendo la libreria implementata. Il programma riceve una stringa (che può contenere spazi) da riga di comando nel `main()` (usando `argc` e `argv[]` e controllando che sia stato passato almeno un parametro), esegue le funzioni implementate nella libreria e stampa a video il risultato di ciascuna operazione.

Suggerimenti

- nell'implementazione delle funzioni è possibile utilizzare le librerie predefinite `ctype.h` e `string.h` per sfruttare alcune loro funzioni;
- per semplicità, mantenete tutti i file (programma principale e librerie) **nella stessa cartella**;
- ricordiamo il procedimento base per lo svolgimento dell'esercizio:
  1. definire l'interfaccia `stringsTools.h` contenente i prototipi delle funzioni richieste
  2. definire la libreria `stringsTools.c` che contiene l'implementazione delle funzioni richieste
  3. creare l'object file `stringsTools.o` dal Terminale con il comando: `gcc -o stringsTools.o -c stringsTools.c`
  4. implementare il programma `libreria_stringhe.c`
  5. compilare il programma principale linkando l'object file da Terminale, usando il comando: `gcc libreria_stringhe.c stringsTools.o -o libreria_stringhe`
- a ogni modifica della libreria e del programma principale, prima di testare il tutto ricordatevi sempre di **salvare tutti i file e rigenerare l'object file della libreria** `stringsTools.h`, utilizzando il Terminale per compilare correttamente il programma, in questo modo:
  1. rigenerare l'object file `stringsTools.o` dal Terminale con il comando: `gcc -o stringsTools.o -c stringsTools.c`
  2. ricompilare il programma principale linkando l'object file, usando `gcc libreria_stringhe.c stringsTools.o -o libreria_stringhe`
- per eseguire il programma principale da Terminale, usate il comando `./libreria_stringhe`
- è possibile passare al programma una stringa contenente spazi bianchi da riga di comando racchiudendola tra virgolette o apici (attenzione all'escaping per i caratteri speciali, per semplicità potete prevedere solo l'utilizzo di stringhe alfanumeriche senza punteggiatura);

Es. 4 Hotel da incubo (TdE del 14 giugno 2023)

■ *Struct, gestione dei file, funzioni*

L'app **Hotel da incubo** individua i peggiori hotel di una città. L'app si basa sulle recensioni che i clienti rilasciano online, su una scala da 1 a 10, per tre parametri: servizi (`s`), pulizia (`pu`), posizione (`po`). Si vogliono definire quindi le funzioni che permettono di ritrovare l'hotel peggiore. Si tenga conto che per ogni hotel della città, è disponibile un file di testo che memorizza le recensioni dei clienti. La prima riga del file riporta il nome dell'hotel, le righe successive riportano ognuna, in modo anonimo, triple di voti assegnati da un cliente ai 3 criteri `s`, `pu` e `po`. Ad esempio, per l'Hotel Excelsior:

```
Excelsior
8 9 7
9 10 10
5 6 5
```

Punto 3.1 (5 punti) - Si scriva la funzione `voto_hotel` che, ricevuto come parametro il nome di un file con le recensioni di un singolo hotel, e ogni altro parametro ritenuto necessario, restituisca un voto aggregato per l'hotel ottenuto nel seguente modo:

1. Si calcola la media per il criterio `s`, la media per il criterio `pu`, la media per il criterio `po` (per esempio, per l'Hotel Excelsior: `media s = 7.3; media pu = 8.3; media po = 7.3`)
2. Si calcola e si restituisce al chiamante la somma delle tre medie (es. per l'Hotel Excelsior: `22.9`)

Punto 3.2 (5 punti) - Si supponga che un file aggiuntivo funga da indice e memorizzi i nomi dei file testuali con le recensioni di vari hotel di una città (non più di 50 hotel). Ogni riga memorizza il nome di un solo file. Ad esempio:

```
excelsior.txt
roma.txt
italia.txt
... ...
```

Si scriva la funzione `incubo` che, ricevuto come parametro il nome del file indice, e ogni altro parametro ritenuto necessario, individui l'hotel con il punteggio più basso. Per ogni file elencato nel file indice, la funzione:

1. Richiama la funzione `voto_hotel` definita al punto precedente per calcolare il voto aggregato di ogni hotel. Memorizza i valori ottenuti in un array i cui elementi sono `struct` con campi `<nome hotel, voto complessivo>`.
2. A partire dall'array costruito, individui e restituisca al chiamante l'hotel con il punteggio più basso.

Punto 3.3 (2 punti) - Si scriva il `main()` del programma, completo di ogni dichiarazione, in modo che, ricevuto come parametro sulla linea di comando il nome del file indice, invochi in modo opportuno le funzioni definite ai punti precedenti, quindi stampi un messaggio per visualizzare il nome e il voto dell'hotel peggiore.