

Assegnamenti

Scegliere 6 esercizi da implementare dalla lista seguente. Tra i 6 scelti ci deve essere almeno un esercizio di ogni gruppo A, B, C, D, E.

Tutti gli esercizi devono essere risolti garantendo la miglior efficienza di esecuzione e il principio del privilegio minimo. Il codice sorgente deve essere ben indentato e commentato. Gli output devono essere formattati correttamente e gli eventuali errori di input da parte dell'utente devono essere tutti gestiti da programma. Se viene allocata memoria dinamica, deve essere immediatamente liberata quando non più necessaria. Non deve esserci memoria allocata, ma non utilizzata.

Tutte le domande concernenti questi assegnamenti devono essere inviate ai docenti tramite l'apposito forum aperto su Moodle in modo che tutti gli studenti possano beneficiare delle risposte.

La consegna delle soluzioni può avvenire in qualunque momento ed al più tardi una settimana prima dello scritto di programmazione funzionale (test a risposte multiple). Deve essere consegnato il codice sorgente C e tutti i file di input usati per testare il programma. Il calendario (data, ora e luogo) della discussione dell'assegnamento verrà comunicata dal docente tramite email.

GRUPPO A (massimo 0.4 punti ad esercizio):

==== Es A1 ====

==== Es A2 ====

GRUPPO B (massimo 0.4 punti ad esercizio):

==== Es B1 ====

==== Es B2====

GRUPPO C (massimo 0.4 punti ad esercizio):

==== Es C1 ====

==== Es C2====

GRUPPO D (massimo 0.7 punti ad esercizio):

==== Es D1 ====

==== Es D2 ====

GRUPPO E (massimo 0.4 punti ad esercizio):

==== Es E1 ====

==== Es E2 ====


GRUPPO A:

==== Es A1 ====

Scrivere un programma che riceve in ingresso da tastiera un numero int N e ne calcola la rappresentazione in complemento a 2 (32 bit). Dopo aver ottenuto la rappresentazione C2 il programma chiede all'utente quale tra le seguenti operazioni eseguire:

1. leggere un nuovo numero int N da tastiera
2. visualizzare la rappresentazione in C2 dell'ultimo N inserito partendo dal bit più significativo (MSB)
3. calcolare l'opposto dell'ultimo N ricevuto in ingresso e visualizzarlo in decimale
4. visualizzare la rappresentazione in C2 dell'opposto dell'ultimo N ricevuto in ingresso partendo dal bit più significativo (MSB).
5. terminare l'esecuzione del programma

Realizzare il menù testuale mediante un array di puntatori a funzioni.

Eseguire tutti i controlli di correttezza sia sui dati ricevuti in input che sulle scelte effettuate dall'utente per eseguire le operazioni sopra elencate. Le stampe devono essere formattate e precedute da una descrizione di cosa si sta stampando. All'uscita dal programma stampare un messaggio di saluto. Dopo ogni output riprodurre il menu di scelte e proseguire. 

Esempio di esecuzione:

Inserire un numero intero:

5

Scegliere un'opzione (1 - Nuovo numero; 2 - Stampa rappresentazione C2; 3 - Calcola opposto; 4 - Stampa rappresentazione C2 dell'opposto; 5 - Esci):

2

La rappresentazione di 5 in C2 MSB è: 0000 0000 0000 0000 0000 0000 0000 0101

Scegliere un'opzione (1 - Nuovo numero; 2 - Stampa rappresentazione C2; 3 - Calcola opposto; 4 - Stampa rappresentazione C2 dell'opposto; 5 - Esci):

4

La rappresentazione dell'opposto di 5 in C2 MSB è: 1111 1111 1111 1111 1111 1111 1111 1011

Scegliere un'opzione (1 - Nuovo numero; 2 - Stampa rappresentazione C2; 3 - Calcola opposto; 4 - Stampa rappresentazione C2 dell'opposto; 5 - Esci):

3

L'opposto di 5 è: -5

Scegliere un'opzione (1 - Nuovo numero; 2 - Stampa rappresentazione C2; 3 - Calcola opposto; 4 - Stampa rappresentazione C2 dell'opposto; 5 - Esci):

1

Inserisci un numero intero:

sgdfgdfs

Errore. Inserisci un numero intero.

7

Scegliere un'opzione (1 - Nuovo numero; 2 - Stampa rappresentazione C2; 3 - Calcola opposto; 4 - Stampa rappresentazione C2 dell'opposto; 5 - Esci):

==== Es A2 ====

Si scriva un programma ricorsivo in C per giocare al gioco "indovina un numero". Il programma ha un menù testuale iniziale con due funzionalità (1 - gioco con due giocatori, 2 - gioco con tre giocatori, 3 - esci). Un giocatore inserisce da tastiera un numero segreto (tra 1 e M) dove M è definita come una costante simbolica. Ad ogni tentativo gli altri giocatori a turno tentano di indovinare il numero inserito, ed il programma può rispondere "esatto", "troppo alto" oppure "troppo basso". Se nessuno dei giocatori indovina il numero entro $\log_2(M)$ passi, il programma stampa "Ha vinto il banco!" e torna al menù principale. Se, al contrario, il numero è stato indovinato, il programma stampa "Ha vinto il giocatore n. 1" oppure "Ha vinto il giocatore n. 2" a seconda che abbia indovinato il primo giocatore che inizia i tentativi o il secondo; poi il programma torna al menù principale. Realizzare il menù testuale con un array di puntatori a funzioni.

Esempio con M=100 (numero indovinato):

Numero segreto: 44

Indovina il numero!

Tentativo numero 1: 15

Troppo basso

Tentativo numero 2: 80

Troppo alto

Tentativo numero 3: 44

Ha vinto il giocatore n. 1

Esempio con M=100 (numero non indovinato):

Numero segreto: 44

Indovina il numero!

Tentativo numero 1: 15

Troppo basso

Tentativo numero 2: 80

Troppo alto

Tentativo numero 3: 54

Troppo alto

...

Ha vinto il banco!

GRUPPO B:

==== Es B1 ====

Scrivere un programma C che legge da file una matrice quadrata $N \times N$ di interi ($N \geq 4$). La dimensione della matrice non è nota, viene calcolata dal file in base al numero di elementi letti. Il nome del file è passato come primo parametro esterno da linea di comando. Il programma trova tutte le sequenze contigue di valori in ogni riga o ogni colonna di almeno M occorrenze. M è letto da tastiera (con controllo dell'input) e deve essere almeno 1 e minore di N . Il programma stampa poi gli indici delle righe e delle colonne che contengono tali sequenze.

Esempio $N=4$, $M=2$

1 0 4 0

2 3 4 9

1 3 4 5

0 0 1 0

Le sequenze di valori contigue da individuare sono 4 4 4 in colonna 2, 3 3 in colonna 1, 0 0 in riga 4. Le altre ripetizioni non sono consecutive.

L'output deve essere formattato come segue:

Ci sono 3 occorrenze consecutive di 4 in colonna 2 a partire dalla riga 0

Ci sono 2 occorrenze consecutive di 3 in colonna 1 a partire dalla riga 1

Ci sono 2 occorrenze consecutive di 0 in riga 3 a partire dalla colonna 0

Ordinare la stampa dei risultati dal maggior numero di occorrenze in ordine decrescente. A numero di occorrenze identico, stampare prima le colonne e poi le righe, in ordine decrescente del numero di colonna/riga. Le occorrenze identiche di lunghezza sulla stessa riga/colonna vengono stampate in ordine crescente della posizione di partenza.

Le matrici vengono memorizzate usando allocazione dinamica della memoria.

==== Es B2====

Generare un array di N stringhe sull'alfabeto [A-Za-z] dove N è il primo parametro passato da linea di comando di tipo int. Per ogni stringa il programma genera una lunghezza casuale tra 4 e SIZE (costante simbolica) e genera i corrispondenti caratteri della stringa ancora in modo pseudocasuale. Si deve minimizzare l'occupazione di memoria per la rappresentazione delle stringhe. Il programma ha come secondo parametro passato da linea di comando il nome di un file che contiene una lista di parole con il primo carattere maiuscolo. Determinare quante parole del file sono generabili usando caratteri delle stringhe nell'array. Le parole devono essere processate in ordine di apparizione nel file. Un carattere può essere usato solo una volta e la soluzione deve essere case-sensitive. Stampare quindi l'elenco di parole generate e l'elenco di caratteri delle stringhe dell'array non utilizzati.

Esempio:

N=5,
Il file parole.txt contiene le parole:
Programmazione
Facile
Uso
Int
Di
Nodo
Lista

Il programma genera l'array di stringhe:

a F i D K m T
b t u s
o U I L s K x
n c t T m H H
P k z h

L'output del programma:

Si possono generare 3 parole:

*Uso
Int
Di*

Caratteri rimasti:

a F K m T b t u L s K x c T m H h P k z h

GRUPPO C:

==== Es C1 ====

Scrivere un programma che riceve come primo parametro da linea di comando il nome di un file CSV da cui legge informazioni sull'osservazione di fenomeni astronomici. Il formato di ogni linea è codice evento, posizione, data, ora. Non è noto il numero di eventi registrato nel file. Il programma legge il file e fa tutti i controlli di correttezza. I codici evento devono essere distinti tra loro. Una volta letto il file, si vogliono determinare le zone astrali in cui si ha la maggior frequenza di eventi. Ogni posizione distinta nel file determina il centro di una zona astrale, che avrà il raggio D. Il parametro D viene passato come secondo parametro dalla linea di comando. Il programma deve calcolare per ogni zona astrale il numero di eventi nella zona, poi stampare le 10 zone con maggiore frequenza, insieme alla corrispondente frequenza di eventi, in ordine decrescente del numero di eventi. A numero di eventi uguale si stampi in ordine della data del primo evento incontrato in quella zona. Se il file contiene meno di 10 zone si devono stampare tutte.

Per semplicità usare coordinate e distanze nello spazio Euclideo tridimensionale.

La soluzione deve consumare la minor quantità possibile di memoria: scegliere la struttura dati più adatta tra array e liste, considerando la possibilità di avere un numero molto grande di eventi.

Esempio:

Il file eventi.csv contiene i dati:

565,4,6,8,11/10/2017,11:30

756,5,9,1,12/12/2017,10:40

765,-8,-2,-1,06/01/2018,23:23

786,-9,-3,-8,07/01/2018,16:19

835,-4,-6,-2,08/01/2018,00:12

Il valore di D è 8.

Il programma stampa le zone in ordine di frequenza di eventi:

Coordinate zona	Numero eventi
(-8,-2,-1)	3
(4,6,8)	2
(5,9,1)	2
(-9,-3,-8)	2
(-4,-6,-2)	2

==== Es C2====

Si scriva un programma in grado di analizzare il contenuto di un file di testo e di calcolare la distribuzione di frequenza della lunghezza delle varie parole in esso contenute (sequenze di caratteri alfanumerici separate da spazi o punteggiatura). Il programma riceve da linea di comando il nome del file da analizzare e produce in uscita una tabella con le frequenze, espresse in percentuale. Infine stampa l'elenco delle parole presenti nel file in ordine alfabetico e associa a ciascuna un istogramma orizzontale di lunghezza proporzionale alla frequenza della parola. Il risultato viene stampato su un file il cui nome è specificato come secondo argomento della linea di comando e ha il formato parola, istogramma (sequenza di asterischi).

Ad esempio il file di testo:

Biology is a science, but what exactly is science? What does the study of biology share with other scientific disciplines? Science (from the Latin scientia, meaning knowledge) can be defined as knowledge that covers general truths or the operation of general laws, especially when acquired and tested by the scientific method. It becomes clear from this definition that the application of the scientific method plays a major role in science. The scientific method is a method of research with defined steps that include experiments and careful observation.

produce il seguente output:

Lunghezza	Frequenza (%)
1	3.45%
2	14.94%
3	12.64%
...	...

a	***
acquired	*
and	**
application	*
...	...

GRUPPO D:

==== Es D1 ====

Scrivere un programma C per la gestione di un magazzino.

Per ogni prodotto, si ha il codice unico (di tipo numerico) che lo individua, il nome, il prezzo in euro e la quantità disponibile in magazzino.

Si vogliono poter eseguire le seguenti operazioni:

1. variazione della quantità di un prodotto, il cui codice è inserito da tastiera;
2. variazione del prezzo di un prodotto, il cui codice è inserito da tastiera;
3. inserimento di un nuovo prodotto, il cui codice è inserito da tastiera;
4. eliminazione di un prodotto, il cui codice è inserito da tastiera;
5. ricerca di un prodotto, il cui codice è inserito da tastiera;
6. stampare i prodotti (in ordine decrescente del codice) la cui quantità è sotto un valore soglia inserito da tastiera.

Eseguire tutti i controlli di correttezza sia sui dati ricevuti in input che sulle scelte effettuate dall'utente per eseguire le operazioni sopra elencate. Le stampe devono essere formattate e precedute da una descrizione di cosa si sta stampando. All'uscita dal programma stampare un messaggio di saluto. Dopo ogni output riprodurre il menu di scelte e proseguire. Il menu delle scelte deve essere realizzato mediante un array di puntatori a funzioni.

Inizialmente il programma acquisisce tutte le informazioni relative ai prodotti dal file elencoProdotti.csv.

Il file contiene i campi: codice, nome, prezzo in formato CSV.

La quantità di ciascun prodotto deve essere inizializzata con un intero pseudocasuale nell'intervallo 0-100000, imponendo un seme pari a 10.

Mantenere la lista di prodotti ordinato per il loro codice (i codici devono essere ordinati in modo crescente), non vi deve essere spreco di memoria.

L'utente farà le scelte da tastiera e al termine della scelta deve essere stampate tutti i dettagli del prodotto inserito/modificato/cancellato.

Quando l'utente sceglie di uscire dal programma, stampare a video i risultati nel seguente ordine:

- il numero totale di prodotti disponibili in magazzino;
- il numero totale di prodotti sotto la soglia;

Creare un file di testo (chiamato numeroTotale.txt) contenente il numero totale di prodotti diversi disponibili in magazzino e il numero totale di prodotti sotto la soglia.

Creare un file di testo (chiamato daOrdinare.bin) contenente l'elenco dei prodotti sotto la soglia compreso il numero di prodotti da ordinare per raggiungere la soglia+1. Usare l'ultima soglia inserita dall'utente. Se non è mai stata inserita usare una soglia default uguale a 10.

Includere codice, nome prodotto, numero prodotti da ordinare.

Esempio di esecuzione:

Scegliere un'opzione (1 - variazione della quantità di un prodotto; 2 - variazione del prezzo di un prodotto; 3 - inserimento di un nuovo prodotto; 4 - eliminazione di un prodotto; 5 - Ricerca di un prodotto; 6 - Stampa prodotti sotto al soglia; 7 - Esci):

2

Inserisci il codice del prodotto:

100

Inserisci il nuovo prezzo:

2500

Dettagli prodotto:

100 Notebook 2500 500

Scegliere un'opzione (1 - variazione della quantità di un prodotto; 2 - variazione del prezzo di un prodotto; 3 - inserimento di un nuovo prodotto; 4 - eliminazione di un prodotto; 5 - Ricerca di un prodotto; 6 - Stampa prodotti sotto al soglia; 7 - Esci):

7

Bye bye!

==== Es D2 ====

Si supponga di dover gestire le statistiche dei piloti durante una gara di Formula Uno.

Il programma deve prendere in input il nome di un file di testo contenente la griglia di partenza dei piloti: costruttore e pilota in ordine di partenza.

Il programma deve leggere da un file .csv le informazioni relative ai tempi di ogni pilota. In ogni riga vi è il nome del pilota, seguito dalle velocità massime raggiunte da questo in una delle 4 fasi. Ogni riga ha la seguente forma:

"Pilota" "Intermediate 1" "Intermediate 2" "Finish line" "Speed Trap"

I nomi dei file devono essere dati in input al programma attraverso i parametri della funzione main.

Successivamente, il programma deve prendere in input, da un terzo file, i piloti con la relativa tempistica *alla fine della propria gara e creare una lista ordinata*. ~~Tale input serve per aggiornare la lista non appena il pilota raggiunge il traguardo (Finish line) o si ritira dalla gara.~~

La lista deve essere *creata e* mantenuta in ordine crescente di tempo (quindi in ordine di arrivo dei piloti), i piloti che non hanno raggiunto il traguardo avranno tempo 0 e dovranno essere situati in coda alla lista (chi si ritira per primo è ultimo nella classifica finale).

Nota:

- Il nome del costruttore e il nome del pilota devono essere una stringa di al più 30 caratteri.
- il tempo è composto da 4 interi (ore, minuti, secondi, millesimi): ore, minuti e secondi separati dal simbolo “:”, i millesimi sono separati dal simbolo “.”.
- Il nome dell'intermedio è una stringa di al più 20 caratteri.
- Lo speed trap è il punto più veloce del circuito di gara. È generalmente verso la fine del rettilineo più lungo, e poco prima del punto di rottura dell'angolo.

La classifica generale (Pilota, Costruttore, Griglia, Tempi) è ottenuta leggendo la griglia di partenza e la lista ordinata dei piloti alla fine della gara e deve essere stampata in ordine decrescente di posizione e deve essere memorizzata in un file in formato .csv.

Suggerimento: per ogni elemento letto nella lista ordinata dei piloti chiamare una funzione che restituisca sia la posizione nella griglia di partenza che il nome del costruttore.

Il programma ha tre funzionalità (1. Podio; 2. Miglior pilota; 3. esci) così definite:

- la frase "Podio: " seguita dal nome dei tre piloti e i rispettivi costruttori che saliranno sul podio in ordine di arrivo;
- la frase "Miglior pilota: " seguita dal nome del pilota che ha raggiunto la massima velocità in uno degli intermedi, nello step finale o nello "speed trap".

Ogni qualvolta che un pilota raggiunge il traguardo o si ritira dalla gara, la lista contenente la griglia di partenza deve essere aggiornata rimuovendo il pilota e il relativo costruttore dalla lista. Al termine della gara, la lista deve quindi essere vuota.

Se viene allocata memoria dinamica, questa deve essere liberata.

Non vi deve essere memoria allocata ma non utilizzata.

Esempio di esecuzione:

Esempio di file di testo contenente i dati relativi alla griglia di partenza (costruttore, pilota):

Red Bull

Ricciardo

Red Bull

Verstappen

Mercedes

Hamilton

Ferrari
 Vettel
 Mercedes
 Bottas
 Ferrari
 Raikkonen
 Renault
 Hulkenberg
 Renault
 Sainz
 Sauber
 Leclerc
 Sauber
 Ericsson
 Force India
 Ocon
 McLaren
 Alonso
 Force India
 Perez
 Toro Rosso
 Hartley
 McLaren
 Vandoorne
 Haas
 Magnussen
 Williams
 Stroll
 Haas
 Grosjean
 Williams
 Sirotkin
 Toro Rosso
 Gasly

Esempio di file .csv:

Pilota	Intermediate 1	Intermediate 2	Finish line	Speed Trap
Alonso	279.4	279.2	246.8	332.6
Bottas	302.5	288.1	252.2	343.5
Ericsson	294.3	290.4	257.5	356.0
Gasly	304.8	288.1	253.9	360.3
Grosjean	299.6	289.0	254.4	360.7
Hamilton	295.8	287.3	251.6	358.0
Hartley	288.6	289.3	253.0	347.6
Hulkenberg	302.2	287.1	252.0	347.0

Leclerc	306.2	290.3	256.5	360.0
Magnussen	301.5	289.0	253.8	359.1
Ocon	299.8	289.9	252.4	364.9
Perez	291.4	285.4	249.3	360.6
Raikkonen	314.5	293.4	258.1	362.6
Ricciardo	305.2	289.7	256.7	352.5
Sainz	293.8	284.2	252.8	346.0
Sirotkin	291.9	291.8	253.1	359.8
Stroll	298.6	291.2	253.1	356.5
Vandoorne	302.9	288.2	253.4	354.2
Verstappen	297.0	289.9	254.7	350.7
Vettel	312.4	295.4	260.4	360.1

Esempio di dati contenente le informazioni finali della gara:

Ricciardo
0:00:00.000
Perez
0:00:00.000
Verstappen
1:38:28.851
Vettel
1:39:28.852
Raikkonen
1:40:28.853
Hamilton
1:41:28.854
Bottas
1:42:28.855
Sainz
0:00:00.000
Hulkenberg
1:42:28.856
Leclerc
1:43:28.857
Vandoorne
1:43:28.858
Ericsson
1:49:28.859
Gasly
1:50:28.860
Ocon
1:50:28.861
Stroll
1:51:28.862
Sirotkin
1:52:28.863

Hartley
1:52:28.864
Magnussen
1:53:28.865
Grosjean
1:53:28.866
Alonso
0:00:00.000

File .csv da produrre in output la classifica finale:

Pilota	Costruttore	Griglia	Tempi
Ricciardo	Red Bull	1	0:00:00.000
Perez	Force India	13	0:00:00.000
Sainz	Renault	8	0:00:00.000
Alonso	McLaren	12	0:00:00.000
Grosjean	Haas	18	1:53:28.866
Magnussen	Haas	16	1:53:28.865
Hartley	Toro Rosso	14	1:52:28.864
Sirotkin	Williams	19	1:52:28.863
Stroll	Williams	17	1:51:28.862
Ocon	Force India	11	1:50:28.861
Gasly	Toro Rosso	20	1:50:28.860
Ericsson	Sauber	10	1:49:28.859
Vandoorne	McLaren	15	1:43:28.858
Leclerc	Sauber	9	1:43:28.857
Hulkenberg	Renault	7	1:42:28.856
Bottas	Mercedes	5	1:42:28.855
Hamilton	Mercedes	3	1:41:28.854
Raikkonen	Ferrari	6	1:40:28.853
Vettel	Ferrari	4	1:39:28.852
Verstappen	Red Bull	2	1:38:28.851

Nota che Ricciardo è l'ultimo in classifica perché è stato il primo a ritirarsi dalla gara, così come Perez è il penultimo perché è stato il secondo a ritirarsi dalla gara. Il primo in classifica è Verstappen che è il primo ad aver raggiunto il traguardo.

Stampare a terminale:

Podio:

Verstappen

Vettel

Raikkonen

Miglior pilota:

Ocon

GRUPPO E:

==== Es E1 ====

Scrivere un programma C che realizza un automa a stati finiti per il riconoscimento delle stringhe generate dall'espressione regolare $[A-Z]\{4\}ab+[^d]\{3\}$. Realizzare l'automa usando array di puntatori a funzioni. Il programma riceve in ingresso una stringa da tastiera (controllare che l'input contenga solo i caratteri dell'alfabeto del linguaggio regolare generato dall'espressione regolare) e stampa "stringa appartenente al linguaggio" oppure "stringa non appartenente al linguaggio". Se la stringa appartiene al linguaggio, verifica se esiste una sottostringa della forma abDD mediante un altro automa a stati finiti che prende in input la stringa data in input precedentemente.

Esempio di input

AGTSabi1U

stringa appartenente al linguaggio

nessuna sottostringa abDD

Esempio di input

AGTSabDDz

stringa appartenente al linguaggio

sottostringa abDD presente

==== Es E2 ====

Scrivere un programma C che realizza un automa a stati finiti per il riconoscimento delle stringhe generate dalla grammatica regolare rappresentata in forma BNF

$S \rightarrow [a-z]S \mid [0-1]S \mid 1$

Realizzare l'automa usando array di puntatori a funzioni. Il programma riceve in ingresso una stringa da tastiera (controllare che l'input contenga solo i caratteri dell'alfabeto del linguaggio regolare generato dalla grammatica regolare) e stampa "stringa appartenente al linguaggio" oppure "stringa non appartenente al linguaggio". Se la stringa appartiene al linguaggio, verifica se esistono sottostringhe della forma "011" e se si stampa tutte le posizioni a cui tali sottostringhe iniziano nella stringa originale.

Esempio di input

ahgf011jh011hkew1

Esempio di output

stringa appartenente al linguaggio

posizioni 4, 9