14BHD INFORMATICA, A.A. 2021/2022

Esercitazione di Laboratorio 11

Obiettivi dell'esercitazione

- Elaborazione di dati usando le più comuni operazioni sugli insiemi
- Uso di un dizionario per fare ricerche in una tabella
- Uso di strutture complesse per la memorizzazione dei dati

Contenuti tecnici

- Creazione e uso di insiemi e dizionari
- Unione, intersezione e differenza tra insiemi
- Accesso ai valori e scansione degli elementi di un dizionario

---

Da risolvere in laboratorio

Esercizio 1. A) Scrivete un programma che conti le occorrenze di ciascuna parola presente in un file di testo, il cui nome è inserito da tastiera. Si assuma che il file contenga solo caratteri alfabetici o di spaziatura. B) Successivamente, migliorate il programma in modo che visualizzi le 100 parole più comuni (in caso di parità alla posizione 100, è indifferente quali parole si stampino). [P8.2] [P8.3]

Esercizio 2. Scrivete un programma che chieda all'utente di fornire due stringhe, per poi visualizzare (evitando ripetizioni di caratteri nella stampa):

- i caratteri che compaiono in entrambe le stringhe;
- i caratteri che compaiono in una stringa ma non nell'altra;
- le lettere che non compaiono in nessuna delle due stringhe.

Suggerimento: trasformare una stringa in un insieme di caratteri. [P8.9]

Esercizio 3. Un array sparso è una sequenza di numeri, la maggior parte dei quali è zero. Un modo efficiente per memorizzare un array sparso è un dizionario, nel quale le chiavi sono le posizioni in cui sono presenti valori diversi da zero, e i valori sono i corrispondenti valori nella sequenza. Per esempio, la sequenza 0 0 0 0 0 4 0 0 0 2 9 sarebbe rappresentata dal dizionario `{5:4, 9:2, 10:9}`. Scrivete una funzione, `sparseArraySum`, i cui argomenti sono due di tali dizionari, *a* e *b*, e che restituisca un array sparso che ne sia il vettore *somma*: il suo valore nella posizione *i* è la somma dei valori di *a* e *b* nella posizione *i*. I dizionari del programma chiamante *non* devono essere modificati. [P8.22]

Esercizio 4. Realizzate il *crivello di Eratostene*: una funzione che calcola i numeri primi, nota anche nell'Antica Grecia. Scegliete un numero intero, n: questa funzione calcolerà tutti i numeri primi fino a n. Per prima cosa, create un insieme ed inseritevi tutti i numeri interi da 2 a n, poi eliminate dall'insieme tutti i multipli di 2, tranne 2 (cioè 4, 6, 8, 10, 12, …), quindi eliminate tutti i multipli di 3, tranne 3 (cioè 6, 9, 12, 15, …) e proseguite così, cancellando ogni volta i multipli del valore minimo presente nell'insieme, fino al numero $\sqrt{n}$. I numeri rimasti nell'insieme sono quelli richiesti. [P8.4]

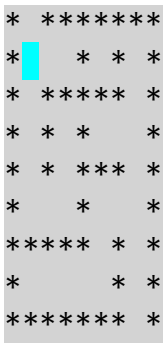
Da risolvere a casa

Esercizio 5. Scrivete un programma "censore", che legga un file (`bad_words.txt`) contenente una lista di "parolacce" (come "sesso", "droga", "C++" e così via), una per riga, inserendole in un insieme. Leggere poi un altro file di testo: il programma deve riscrivere il secondo file letto, generandone un terzo nel quale tutte le lettere di ciascuna parolaccia (comprese quelle nelle sotto-parole contenenti parolacce) siano state sostituite da un numero di asterischi pari alla sua lunghezza. [P8.14]

Esercizio 6. Scrivete un programma che legga i dati dal file di testo[1] `rawdata_2004.txt` e li inserisca in un dizionario le cui chiavi sono nomi di nazioni e i cui valori sono redditi annui pro capite. Si noti che nel file i campi sono separati da un carattere di tabulazione `'\t'`. Poi, il programma deve chiedere all'utente di fornire nomi di nazioni, per visualizzare i valori corrispondenti. Il programma termina quando l'utente scrive `quit`. [P8.17]
Aggiornamento: all'indirizzo https://www.cia.gov/the-world-factbook/field/real-gdp-per-capita/country-comparison potete scaricare (bottone "Download Data") i dati analoghi, aggiornati al 2021, in formato CSV. Vi suggeriamo di provare a risolvere lo stesso esercizio lavorando sul file CSV.

Esercizio 7. Scrivete un programma che legga un file di testo[2] (`maze.txt`) contenente l'immagine di un labirinto, come il seguente, in cui gli asterischi sono muri invalicabili e gli spazi sono corridoi percorribili.

[1] Il file `rawdata_2004.txt` è scaricabile da https://drive.google.com/file/d/1SVN7K4B7Bq_zN5lBUJMlvBieF_syxXzO/view?usp=sharing
[2] Un esempio di file `maze.txt` è scaricabile da https://drive.google.com/file/d/1Luq-ijXAMCbQu620uoTAg-NY-6eoP4wu/view?usp=sharing



Generate un dizionario le cui chiavi siano tuple del tipo (riga, colonna) delle posizioni dei corridoi e i cui valori siano insiemi di posizioni di corridoi adiacenti. Nell'esempio di labirinto qui presentato, (1, 1) (quadratino azzurro) ha come corridoi adiacenti {(1, 2), (0, 1), (2, 1)}. Visualizzate il dizionario. [P8.20]

Esercizio 8. Completate il programma dell'esercizio precedente in modo che trovi una via d'uscita dal labirinto a partire da un punto qualsiasi.
Costruite un nuovo dizionario le cui chiavi siano le posizioni dei corridoi e i cui valori siano tutti uguali alla stringa "?".
Poi scandite le chiavi di tale dizionario. Per ogni chiave che si trova al *confine* del labirinto (cioè rappresenta un'uscita), sostituite la "?" con un valore ("N", "E", "S" o "W") che indichi la direzione del percorso di uscita.
A questo punto, ripetete la scansione delle chiavi il cui valore è rimasto "?" e verificate se hanno almeno un adiacente il cui valore non sia "?", usando il primo dizionario per localizzare gli adiacenti. Trovato uno di tali adiacenti, sostituite la stringa "?" con la direzione dell'adiacente.
Se durante una di tali scansioni non riuscite a compiere nessuna di queste sostituzioni, interrompete l'algoritmo.
Infine, visualizzate il labirinto così ottenuto: in ogni posizione di corridoio sarà presente la direzione che porta più rapidamente a un'uscita. Ad esempio:

```
*N*******
*NWW*?*S*
*N*****S*
*N*S*EES*
*N*S***S*
*NWW*EES*
*****N*S*
*EEEEN*S*
*******S*
```

[P8.21]