

Esercizio filtro

Scrivere un programma che legga da **riga di comando** un numero naturale positivo e, come mostrato nell'**Esempio di esecuzione**, stampi a video ogni cifra del numero inferiore a quella che la segue.

Si assuma un ordinamento delle cifre da sinistra verso destra. Ad esempio, dato il numero 496:

- 4 è la prima cifra del numero; la cifra 4 è seguita dalla cifra 9;
- 9 è la seconda cifra del numero; la cifra 9 è seguita dalla cifra 6;
- 6 è la terza e ultima cifra del numero; la cifra 6 non è seguita da nessun'altra cifra.

Si assuma inoltre che il valore letto da **riga di comando** sia nel formato corretto.

Si noti che l'ultima cifra del numero letto non viene mai stampata.

Esempio d'esecuzione:

```
$ go run esercizio_filtro.go 344
3
```

```
$ go run esercizio_filtro.go 34456
345
```

```
$ go run esercizio_filtro.go 3
```

```
$ go run esercizio_filtro.go 37778
37
```

Test automatico

L'esercizio filtro è considerato esatto **solo se** rispetta le specifiche date e **solo se** passa il test automatico fornito

Inizializzazione del test automatico Al fine di poter eseguire il test automatico, è prima necessario eseguire **una volta sola** ed **esclusivamente nella cartella relativa a questo esercizio** il comando: `go mod init filtro` ottenendo il seguente output:

```
$ go mod init filtro
go: creating new go.mod: module filtro
go: to add module requirements and sums:
    go mod tidy
```

Esecuzione del test automatico Successivamente sarà possibile eseguire il test automatico utilizzando il comando `go test`. L'esercizio passa il test automatico se il comando `go test` restituisce **PASS**, come nel seguente output:

```
$ go test
PASS
ok
```

Invece, nel caso in cui l'output dovesse restituire **FAIL**, come nel seguente esempio, significa che almeno un caso tra quelli riportati nell'esempio d'esecuzione non è stato eseguito in modo corretto, ed il filtro è considerato **errato**. In tal caso `go test` restituirà anche l'output atteso dal programma e l'output effettivo che il programma produce. Questo dato è utile per capire in cosa il risultato del programma differisce dall'output atteso.

```
$ go test
--- FAIL: TestFiltro (0.00s)
    esercizio_filtro_test.go:40:
        ...
```

Esercizio 1

Parte 1

Scrivere un programma che legga da **riga di comando** una stringa rappresentante una funzione quadratica nella forma $ax^2+bx+c=0$, con a, b, c interi e diversi da 0.

Esempio:

$5x^2+1x+2=0$

Il programma deve stampare su standard output quante e quali sono le soluzioni reali di questa equazione. Ogni soluzione deve essere approssimata usando 3 cifre decimali.

Suggerimento: Per stampare un numero con virgola approssimando alle prime 3 cifre decimali è possibile utilizzare `fmt.Printf("Numero: %.3f", num)`

Le soluzioni si possono trovare applicando la seguente formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Ne risulta che il numero di soluzioni reali totali di un'equazione quadratica possono essere zero, se il discriminante, ovvero la parte sotto la radice quadrata, è negativo; una, se il discriminante è nullo; oppure due altrimenti. Si ricorda che la funzione `sqrt` per calcolare la radice quadrata è disponibile nel package `math`. È possibile utilizzare la funzione `split` del package `strings` per suddividere una stringa in parti separate da una sottostringa specificata.

Parte 2

Estendere la Parte 1 in modo tale da leggere da **riga di comando** anche due valori reali **soglia** ed **epsilon** e stampare quali soluzioni siano maggiori del valore **soglia** di almeno **epsilon**

Esempio d'esecuzione:

```
$ go run esercizio_1.go 4x^2+10x+1=0 -1 0.01
Esistono due soluzioni reali: -0.104 e -2.396
La soluzione -0.104 è maggiore della soglia
```

```
$ go run esercizio_1.go 4x^2-10x+1=0 2.3 0.05
Esistono due soluzioni reali: 2.396 e 0.104
La soluzione 2.396 è maggiore della soglia
```

```
$ go run esercizio_1.go 1x^2+10x+25=0 -6 1
Esiste un'unica soluzione reale: -5.000
```

```
$ go run esercizio_1.go 5x^2+1x+2=0 -1 0.01
Non ci sono soluzioni reali
```

Esercizio 2

Parte 1

Scrivere un programma che:

- legga da **standard input** una stringa **s**. Si assuma che la stringa non contenga caratteri di spaziatura.
- verifichi che la stringa sia composta di sole lettere minuscole nello standard di caratteri ASCII. Qualora non fosse così il programma deve terminare senza stampare nulla.
- stampi a schermo, tutte le sottosequenze della stringa **s** nelle quali le lettere siano in ordine crescente (si considerino solamente sottosequenze di almeno 2 lettere).

Come mostrato nell'**Esempio d'esecuzione**, ciascuna sottosequenza deve essere stampata un'unica volta, riportando il numero di volte in cui la sottosequenza appare in **s**.

Oltre alla funzione `main()`, deve essere definita ed utilizzata almeno la funzione `Sottostringhe(s string) map[string]int`, che riceve in input un valore **string** nel parametro **s**, e restituisce un valore `map[string]int` in cui, per ogni sottosequenza di **s** di almeno 2 lettere, sia memorizzato il numero di volte in cui essa appare in **s**.

Parte 2

Estendere la Parte 1 in modo tale che le sottosequenze vengano stampate in ordine lessicale crescente. A tale scopo, si utilizzi la libreria `sort`.

Esempio d'esecuzione:

```
$ go run esercizio_2.go
abcdef
output:
ab 1
abc 1
abcd 1
abcde 1
abcdef 1
bc 1
bcd 1
bcde 1
bcdef 1
cd 1
cde 1
cdef 1
de 1
def 1
ef 1
```

```
$ go run esercizio_2.go
fedcba
output:
```

```
$ go run esercizio_2.go
abcababab
output:
ab 4
abc 1
bc 1
```

```
$ go run esercizio_2.go
abbèh
```

```
$ go run esercizio_2.go
01010101
```

```
$ go run esercizio_2.go
ABCD
```

Esercizio 3

Un'utenza di telefonia mobile è identificata dal numero del telefono mobile e da un codice che identifica la scheda SIM che gli corrisponde.

Ne consegue che, in un dato istante temporale, ad un numero di telefono mobile possa corrispondere una ed una sola scheda SIM.

Comunque, nel corso del tempo, ad un numero di telefono mobile possono corrispondere diversi codici relativi a schede SIM.

L'utenza di telefonia mobile attiva è quella caratterizzata dal codice più recente relativo ad una scheda SIM.

Parte 1

Scrivere un programma che:

- legga da **standard input** una sequenza di righe di testo;
- termina la lettura quando, premendo la combinazione di tasti **Ctrl+D**, viene inserito da **standard input** l'indicatore End-Of-File (EOF).

Ogni riga di testo è nel formato

`NUMERO_TELEFONO;CODICE_SIM`

dove `NUMERO_TELEFONO` è il numero telefonico (valore composto da sole cifre) di una utenza, mentre `CODICE_SIM` è il codice che identifica la scheda SIM associata (valore composto da caratteri alfanumerici).

Ciascuna riga di testo descrive quindi un'utenza di telefonia mobile. Le utenze di telefonia mobile sono specificate in ordine cronologico: la prima riga descrive l'utenza di telefonia mobile meno recente, l'ultima riga descrive l'utenza di telefonia mobile più recente.

Definire la struttura `Utenza` per memorizzare il numero telefonico e il codice della SIM associata.

Implementare le funzioni:

- `LeggiUtenze()` (`utenze []Utenza`) che:
 1. legge da **standard input** una sequenza di righe di testo, terminando la lettura quando viene letto l'indicatore End-Of-File (EOF);
 2. restituisce un valore `[]Utenza` nella variabile `utenze` in cui è memorizzata la sequenza di istanze del tipo `Utenza` inizializzate con i valori letti da **standard input**.

Parte 2

Il registro telefonico di un operatore è la lista di tutte le utenze telefoniche passate (non attive) e presenti (attive). Il registro contiene, per ogni numero

di telefono, l'elenco di SIM associate in ordine cronologico. Definire il tipo di dato `RegistroTelefonico` (alias di `map[string][]string`) che associa ad ogni numero di telefono tutte le SIM in ordine cronologico.

Implementare le funzioni:

- `InizializzaRegistro()` (`registro RegistroTelefonico`) che inizializza un'istanza di tipo `RegistroTelefonico` (un registro telefonico vuoto) e la restituisce all'interno della variabile `registro`;
- `AggiungiUtenza(registro RegistroTelefonico, utenza Utenza)` (`registroAggiornato RegistroTelefonico`) che riceve in input un'istanza di tipo `RegistroTelefonico` nella variabile `registro` e un'istanza di tipo `Utenza` nella variabile `utenza`. Se il numero dell'utenza non è presente nel registro, viene aggiunta una nuova voce. Altrimenti, viene solamente aggiunta la nuova sim al numero di telefono. La funzione restituisce il registro telefonico aggiornato nella variabile `registroAggiornato`.
- `Identifica(registro RegistroTelefonico, telefono string)` (`codiceSIM string`) che riceve in input un'istanza di tipo `RegistroTelefonico` nella variabile `registro` e un valore di tipo `string` nella variabile `telefono` (un numero di telefono mobile). La funzione restituisce il codice della SIM (presente in `registro`) corrispondente a `telefono` nella variabile `codiceSIM`. Se in `registro` sono presenti più codici SIM corrispondenti a `telefono`, viene restituito il codice più recente, ovvero quello che nella sequenza di input è letto per ultimo. Se in `registro` non è presente alcun codice SIM corrispondente a `telefono`, viene restituito il valore `""`.

Scrivere un programma che:

- Richiami la funzione `LeggiUtenze` per leggere da standard input una sequenza di utenze
- Usi le funzioni `InizializzaRegistro` e `AggiungiUtenza` per inizializzare il registro ed inserirvi ogni utenza letta da standard input
- Stampi per ogni numero di telefono nel registro che inizia con 338, la SIM più recente restituita dalla funzione `Identifica`.

Esempio d'esecuzione: Nota bene: siccome l'output del programma è lungo, vengono riportate solo le prime tre righe e le ultime tre righe separate da [...] per brevità.

```
$ go run esercizio_3.go < registro1.txt
Il numero 338245567 è associato alla sim 464c1f62ff7f8dafff0a20e6bb2fa20e
Il numero 338345567 è associato alla sim 5fa42ef883cf1f749b55c213dda314ab
Il numero 338345467 è associato alla sim 809eb5802ccf66708614f75cd8800bc7
[...]
Il numero 338145167 è associato alla sim edaf427ea16089fc1892eccfd39b94b6
Il numero 338245067 è associato alla sim 23e8e8d3cff934d8302db861b3d4a14e
Il numero 338245467 è associato alla sim 5e1d885e4e462d556f807bbd29f3e62a
```

```
$ go run esercizio_3.go < registro2.txt
Il numero 338445367 è associato alla sim e7e818dd24e61103189fc39a73e98c4d
Il numero 338445267 è associato alla sim 2babaeb631887cd90b82e56e1fcdba7b
Il numero 338345467 è associato alla sim 7514c20e15d760dc5f69e9aefe0784d8
[...]
Il numero 338345267 è associato alla sim f2de3e746944aa8fd1798e4c97baf751
Il numero 338145167 è associato alla sim 202620525af16637f8551c5d1407bb9d
Il numero 338545567 è associato alla sim 9556aa951e6a1e5315fa0d65addfa5cb
```