Laboratorio 7 - Stringhe II

1 Testo concatenato

Scrivere un programma che:

- legga da **standard input** un testo su più righe;
- termini la lettura quando, premendo la combinazione di tasti `Ctrl+D`, viene inserito l'indicatore End-Of-File (EOF);
- ristampi il testo letto.

Nota: la stampa del testo deve essere effettuata solamente dopo aver letto per intero il testo.

Oltre alla funzione `main()`, il programma deve definire e utilizzare le seguenti funzioni:

- una funzione `LeggiTesto() string` che legge da **standard input** un testo su più righe e terminato dall'indicatore EOF, restituendo un valore `string` in cui è memorizzato il testo letto.

Esempio d'esecuzione:

```
$ go run ristampa.go
Inserisci testo (termina con CTRL+D):
Testo di prova
disposto su più righe

Alcune possono essere anche righe vuote
Testo letto:
Testo di prova
disposto su più righe

Alcune possono essere anche righe vuote
```

2 Qual è l'output?

Qual è l'output del seguente programma?

```go
package main

import (
    "fmt"
)

const NumeroLettere = 'Z' - 'A' + 1

func main() {

    var carattere rune = 'Z'

    carattere += 2
    carattere -= 'A'
    carattere %= NumeroLettere
    carattere += 'A'

    fmt.Printf("%c\n", carattere)
}
```

3 Statistiche testo

Scrivere un programma che:

- legga da **standard input** un testo su più righe;
- termini la lettura quando, premendo la combinazione di tasti `Ctrl+D`, viene inserito da **standard input** l'indicatore End-Of-File (EOF);
- stampi a video le seguenti statistiche relative al testo letto:
  i. il numero di parole presenti nel testo;
  ii. la lunghezza media delle parole presenti nel testo.

Nota: una parola è una sequenza di caratteri consecutivi rappresentanti delle lettere. Numeri, punteggiatura e caratteri di spaziatura intervallano parole diverse.

Oltre alla funzione `main()`, il programma deve definire e utilizzare le seguenti funzioni:

- una funzione `LeggiTesto() string` che legge da **standard input** un testo su più righe terminato dall'indicatore EOF (`CTRL+D`), restituendo un valore `string` in cui è memorizzato il testo letto;
- una funzione `StatisticheParole(s string) (int, int)` che riceve in input un valore `string` nel parametro `s` e restituisce due valori `int` pari rispettivamente al numero di parole presenti in `s` e alla loro lunghezza totale.

Suggerimento: per sapere se un carattere rappresenta una lettera utilizza la funzione `unicode.IsLetter()` del package `unicode`. Utilizza `go doc unicode.IsLetter` per scoprire il suo funzionamento.

Esempio d'esecuzione:

```
$ go run statistiche.go
Inserisci un testo su più righe (termina con Ctrl+D):
Testo di prova

su cui calcolare le statistiche.
Test 01: prova.prova0prova
Statistiche:
Numero parole: 12
Lunghezza media: 4.833333333333333
```

4 Testo invertito

Scrivere un programma che legga da **standard input** un testo formato da un numero variabile di righe, terminando la lettura quando viene inserita da **standard input** una riga vuota (`""`), e lo ristampi dall'ultimo carattere al primo.

Scrivere un programma che:

- legga da **standard input** un testo su più righe;
- termini la lettura quando viene inserita da **standard input** una riga vuota (`""`);
- ristampi il testo letto (riga vuota esclusa) dall'ultimo carattere al primo.

Oltre alla funzione `main()`, il programma deve definire e utilizzare le seguenti funzioni:

- una funzione `LeggiTesto() string` che legge da **standard input** un testo su più righe e terminato da una riga vuota (`""`), restituendo un valore `string` in cui è memorizzato il testo letto (riga vuota esclusa);
- una funzione `InvertiStringa(s string) string` che riceve in input un valore `string` nel parametro `s` e ne inverte l'ordine dei caratteri, ovvero restituisce un valore `string` in cui il primo carattere è l'ultimo che definisce `s`, il secondo carattere è il penultimo che definisce `s`, ... e l'ultimo carattere è il primo che definisce `s`.

Esempio d'esecuzione:

```
$ go run stringainvertita.go
Inserisci un testo su più righe (termina con riga vuota):
Testo di prova
disposto su due righe
```

```
Testo invertito:
ehgir eud us otsopsid
avorp id otseT
```

5 Numero nascosto

Scrivere un programma che legga da **standard input** una riga di testo e stampi in output il doppio del numero nascosto all'interno della riga di testo, ovvero il doppio del numero che si ottiene concatenando le cifre presenti all'interno della riga di testo. Il programma non stampa nulla se non è presente alcun numero nascosto.

Suggerimento: Per convertire una stringa in un numero intero utilizzate la funzione `strconv.Atoi()` del package `strconv`. Invece, per sapere se un carattere è una cifra utilizzate la funzione `unicode.IsDigit()` del package `unicode`.

Oltre alla funzione `main()`, il programma deve definire e utilizzare le seguenti funzioni:

- una funzione `LeggiTesto() string` che legge da **standard input** una riga di testo, restituendo un valore `string` in cui è memorizzato il testo letto;
- una funzione `NumeroNascosto(testo string) (int, error)` che riceve in input un valore `string` nel parametro `testo` e restituisce due valori:
  - il primo valore è un numero intero che rappresenta il numero nascosto all'interno del testo. Se il testo in input non contiene alcun numero il valore restituito deve essere `0`;
  - il secondo valore è l'eventuale errore restituito dalla funzione `strconv.Atoi()`.

Esempio d'esecuzione:

```
$ go run numero_nascosto.go
Ci8ao 97com3 va?
Doppio del numero nascosto: 17946 (8973 * 2)

$ go run numero_nascosto.go
Ch3 831 t3mp0
Doppio del numero nascosto: 766260 (383130 * 2)

$ go run numero_nascosto.go
c140n3
Doppio del numero nascosto: 2806 (1403 * 2)

$ go run numero_nascosto.go
nessun numero nascosto
```

6 Linguaggio farfallino

Nel linguaggio farfallino ciascuna vocale non accentata (`vocale`) viene sostituita da una sequenza di tre caratteri `vocale-f-vocale`. Per esempio, la vocale `a` viene sostituita dalla sequenza `afa`, la vocale `e` dalla sequenza `efe` e così via. Se una vocale è maiuscola, anche la sequenza di tre caratteri che sostituisce la vocale deve essere definita da caratteri maiuscoli (ad esempio, la vocale `A` viene sostituita dalla sequenza `AFA`).

Scrivere un programma che:

- legga da **standard input** un testo su più righe;
- termini la lettura quando, premendo la combinazione di tasti `Ctrl+D`, viene inserito da **standard input** l'indicatore End-Of-File (EOF);
- ristampi il testo letto dopo averlo tradotto in linguaggio farfallino.

Oltre alla funzione `main()`, il programma deve definire e utilizzare le seguenti funzioni:

- una funzione `LeggiTesto() string` che legge da **standard input** un testo su più righe e terminato

dall'indicatore EOF, restituendo un valore `string` in cui è memorizzato il testo letto;
- una funzione `TraduciCarattereInFarfallino(c rune) string` che riceve in input un valore `rune` nel parametro `c` e restituisce un valore `string` che rappresenta la traduzione in linguaggio farfallino di `c`;
- una funzione `TraduciTestoInFarfallino(s string) string` che riceve in input un valore `string` nel parametro `s` e restituisce un valore `string` che rappresenta la traduzione in linguaggio farfallino di `s`.

Suggerimento: per verificare se un carattere è una vocale potete utilizzare i costrutti `if` o `switch`, oppure utilizzare la funzione `strings.ContainsRune()` del package `strings`.

Esempio d'esecuzione:

```
$ go run farfallino.go
Inserisci un testo su più righe (termina con CTRL+D):
Questo è un testo di prova
da trasformare IN ALFABETO FARFALLINO
Risultato:
Qufuefestofo èfè ufun tefestofo difi profovafa
dafa trafasfoformafarefe IFIN AFALFAFABEFETOFO FAFARFAFALLIFINOFO
```

7 Garibaldi

Scrivere un programma che:

- legga da **standard input** un testo su più righe;
- termini la lettura quando viene inserita da **standard input** una riga vuota (`""`);
- ristampi il testo letto (riga vuota esclusa) sostituendo tutte le vocali con delle `u`.

Oltre alla funzione `main()`, il programma deve definire e utilizzare le seguenti funzioni:

- una funzione `LeggiTesto() string` che legge da **standard input** un testo su più righe e terminato da una riga vuota (`""`), restituendo un valore `string` in cui è memorizzato il testo letto (riga vuota esclusa);
- una funzione `TrasformaCarattere(c rune) rune` che riceve in input un valore `rune` nel parametro `c` e restituisce un valore `rune` uguale a `u` se `c` è una vocale, e uguale a `c` altrimenti.
- una funzione `Garibaldi(s string) string` che riceve in input un valore `string` nel parametro `s` e restituisce un valore `string` in cui ogni carattere è trasformato utilizzando la funzione `TrasformaCarattere()`.

Esempio d'esecuzione:

```
$ go run garibaldi.go
Inserisci un testo su più righe (termina con riga vuota):
Garibaldi fu ferito
fu ferito in una gamba,
Garibaldi che comanda
che comanda i bersaglier

Risultato trasformazione:
Gurubuldu fu furutu
fu furutu un unu gumbu,
Gurubuldu chu cumundu
chu cumundu u bursugluur
```

8 Spaziatura

Scrivere un programma che:

- legga da **standard input** un testo su più righe;

- termini la lettura quando, premendo la combinazione di tasti `Ctrl+D`, viene inserito da **standard input** l'indicatore End-Of-File (EOF);
- ristampi il testo letto con spaziatura, ovvero inserendo uno spazio `' '` tra ogni coppia di caratteri che **non** sono spazi.

Oltre alla funzione `main()`, il programma deve definire e utilizzare le seguenti funzioni:

- una funzione `LeggiTesto() string` che legge da **standard input** un testo su più righe e terminato dall'indicatore EOF, restituendo un valore `string` in cui è memorizzato il testo letto;
- una funzione `Spazia(s string) string` che riceve in input un valore `string` nel parametro `s` e restituisce un valore `string` che rappresenta la versione spaziata di `s`.

Suggerimento: per sapere se un carattere è uno spazio utilizzate la funzione `unicode.IsSpace` del package `unicode`.

Esempio d'esecuzione:

```
$ go run spaziatura.go
Inserisci un testo su più righe (termina con CTRL+D):
Testo di prova
da stampare con spaziatura
Risultato:
T e s t o  d i  p r o v a
d a  s t a m p a r e  c o n  s p a z i a t u r a
```

9 Il cifrario di Cesare

Giulio Cesare usava per le sue corrispondenze riservate un codice di sostituzione molto semplice, nel quale la lettera chiara viene sostituita dalla lettera che la segue di tre posti nell'alfabeto: la lettera A è sostituita dalla D, la B dalla E, e così via fino alle ultime lettere che sono cifrate con le prime.

Chiaro: 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

Cifrato: 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

Più in generale si dice cifrario di Cesare un codice nel quale ogni lettera del messaggio chiaro viene spostata di un numero fisso `k` di posti (non necessariamente tre), dove `k` è detto **chiave di cifratura**.

Scrivere un programma che:

- legga da **standard input** un numero intero `k` (la chiave di cifratura);
- legga da **standard input** un messaggio in chiaro su più righe, terminando la lettura quando, premendo la combinazione di tasti `Ctrl+D`, viene inserito da **standard input** l'indicatore End-Of-File (EOF);
- stampi il messaggio cifrato (ottenuto con chiave di cifratura `k`) corrispondente al messaggio in chiaro letto.

Oltre alla funzione `main()`, il programma deve definire e utilizzare le seguenti funzioni:

- una funzione `LeggiNumero() int` che legge da **standard input** un numero intero e ne restituisce il valore;
- una funzione `LeggiTesto() string` che legge da **standard input** un testo su più righe e terminato dall'indicatore EOF, restituendo un valore `string` in cui è memorizzato il testo letto;
- una funzione `CifraCarattere(c rune, chiave int) rune` che riceve in input un valore `rune` nel parametro `c` ed un valore `int` nel parametro `chiave`, e restituisce un valore `rune` uguale a `c` nel caso in cui `c` non sia una lettera dell'alfabeto inglese, uguale al valore cifrato corrispondente a `c` (ottenuto con chiave di cifratura `chiave`) altrimenti. In particolare, il valore cifrato deve essere minuscolo se `c` è minuscolo e maiuscolo se `c` è maiuscolo;
- una funzione `CifraTesto(s string, chiave int) string` che riceve in input un valore `string` nel parametro `s` ed un valore `int` nel parametro `chiave`, e restituisce un valore `string` ottenuto cifrando ogni carattere di `s` tramite la funzione `CifraCarattere()`.

Esempio d'esecuzione:

```
$ go run cifrario_cesare.go
Inserisci un numero: 1
Inserisci un testo su più righe (termina con CTRL D):
Testo di esempio
diviso su righe diverse
Testo cifrato:
Uftup ej ftfnqjp
ejwjtp tv qsjhif ejwfstf

$ go run cifrario_cesare.go
Inserisci un numero: -2
Inserisci un testo su più righe (termina con CTRL D):
AbC

dEf
Testo cifrato:
YzA

bCd
```