

Laboratorio 4 - Cicli annidati, Stringhe/Rune

Cicli annidati - Qual è l'output?

Quanti asterischi stampa il seguente programma?

```
package main

import "fmt"

func main() {
    var n int = 3

    var i, j int
    for i = 0; i < n; i++ {
        for j = 0; j < n; j++ {
            fmt.Print("*")
        }
    }
    fmt.Println()
}
```

Cicli annidati - Qual è l'output?

Quanti asterischi stampa il seguente programma?

```
package main

import "fmt"

func main() {
    var n int = 3

    var i, j, z int
    for i = 0; i < n; i++ {
        for j = 0; j < n; j++ {
            for z = 0; z < n; z++ {
                fmt.Print("*")
            }
        }
    }
    fmt.Println()
}
```

Cicli annidati - Qual è l'output?

Quanti asterischi stampa il seguente programma?

```

package main

import "fmt"

func main() {
    var n int = 3

    var i, j int
    for i = 0; i < n; i++ {
        for j = i+1; j < n; j++ {
            fmt.Print("*")
        }
    }
    fmt.Println()
}

```

Cicli annidati - Trova l'errore

Questo programma dovrebbe stampare 4 asterischi ma non funziona correttamente. Qual è l'errore?

```

package main

import "fmt"

func main() {
    var n int = 2

    var i, j int
    for i = 0; i < n; i++ {
        for j = 0; j < n; i++ {
            fmt.Print("*")
        }
    }
    fmt.Println()
}

```

Cicli annidati - Trova l'errore

Questo programma dovrebbe stampare 4 asterischi ma non funziona correttamente. Qual è l'errore?

```

package main

import "fmt"

```

```
func main() {
    var n int = 2

    var i, j int
    for i = 0; i < n; i++ {
        for j = 0; j < n; j-- {
            fmt.Print("*")
        }
    }
    fmt.Println()
}
```

Stringhe/Rune - Qual è l'output?

Qual è l'output del seguente programma?

```
package main

import "fmt"

func main() {
    var x int
    var y int = 'a'
    x = 'A'
    fmt.Print(x, " ")
    for x := 1; x < 10; x++ {
        fmt.Print(x+y, " ")
    }
    fmt.Println()
}
```

Stringhe/Rune - Qual è l'output?

Qual è l'output del seguente programma?

```
package main

import (
    "fmt"
)

func main() {

    s := "Papà!"

    var s1 string
```

```

    for _, carattere := range s {
        s1 += string(carattere)
    }
    fmt.Println(s1)

    var s2 string
    for _, carattere := range s {
        s2 = string(carattere) + s2
    }
    fmt.Println(s2)
}

```

Stringhe/Rune - Qual è l'output?

Analizzate l'output del seguente programma.

```

package main

import "fmt"

func main() {

    s := "Ciao René!"
    numerocaratteri := 0
    numeroiterazione := 0
    for i, c := range s {
        fmt.Print("Iterazione ", numeroiterazione, ": In posizione ", i,
            " inizia la sottosequenza di byte che codifica il carattere ",
            string(c), "\n")
        numerocaratteri++
        numeroiterazione++
    }
    fmt.Println("Numero iterazioni:", numeroiterazione)
    fmt.Println("Numero di byte utilizzati per rappresentare la stringa:", len(s))
    fmt.Println("Numero di caratteri che definiscono la stringa:", numerocaratteri)
}

```

Osservazioni:

In generale, una stringa è una sequenza di byte. Nelle esercitazioni di laboratorio assumeremo sempre, se non esplicitato altrimenti, che una stringa sia una sequenza di byte che rappresenta una sequenza di caratteri codificati secondo lo standard Unicode/UTF-8:

- Ogni carattere è rappresentato da una sequenza di bit definita dallo standard Unicode/UTF-8 la cui lunghezza varia da 1 a 4 byte (1 byte = 8

bit). Per i caratteri considerati nello standard US-ASCII, integrato nello standard Unicode, la sequenza di bit è lunga 1 byte. Per un carattere non considerato nello standard US-ASCII, la sequenza di bit può essere lunga da 2 a 4 byte (ad esempio, la sequenza di bit prevista dallo standard Unicode/UTF-8 per rappresentare il carattere è è lunga 2 byte).

- In generale, il numero di caratteri che definiscono una stringa `s` è minore o uguale al numero di byte utilizzati per rappresentare la stringa `s` (`len(s)`).
- Per lunghezza di una stringa `s` si intende il numero di byte utilizzati per rappresentare `s` (`len(s)`).

Ad ogni iterazione del ciclo definito dal costrutto `for range`, `i` (variabile di tipo `int`) indica la posizione in cui inizia la sottosequenza di byte che rappresenta il carattere `string(c)`. `c` è una variabile di tipo `rune`, il cui valore è un intero corrispondente al codice Unicode del carattere.

Stringhe/Rune - Qual è l'output?

Qual è l'output del seguente programma?

```
package main

import "fmt"

func main() {

    s := "Ciao, come va?"
    // s è interamente definita da caratteri considerati nello standard US-ASCII

    for i := 0; i < len(s); i++ {
        fmt.Print(string(s[i]))
    }
    fmt.Println()

    s = "Ciao, come è andata?"
    // s non è interamente definita da caratteri considerati nello standard US-ASCII

    for i := 0; i < len(s); i++ {
        fmt.Print(string(s[i]))
    }
    fmt.Println()

}
```

Osservazioni:

- Data una stringa `s`, `s[i]` è il byte in posizione `i` nella sequenza di byte che rappresenta `s`. In generale, `s[i]` **non** codifica un carattere.

- In generale, per esaminare in sequenza i caratteri che definiscono una stringa si deve utilizzare il costrutto `for range`.

1 Cicli annidati - Quadrato di asterischi

Scrivere un programma che legga da **standard input** un numero intero `n` e stampi a video un quadrato di `n` asterischi intervallati da spazi come mostrato nell'**Esempio di esecuzione**.

Suggerimento: potete utilizzare due cicli `for` annidati.

Esempio d'esecuzione:

```
$ go run quadratoasterischi.go
Inserisci un numero: 3
* * *
* * *
* * *
```

2 Cicli annidati - Quadrato a righe alterne (1)

Scrivere un programma che legga da **standard input** un numero intero `n` e che, come mostrato nell'**Esempio di esecuzione**, stampi a video un quadrato di `n` righe costituite ciascuna da `n` simboli intervallati da spazi, alternando fra loro righe costituite solo da simboli `*` (asterisco) intervallati da spazi e righe costituite solo da simboli `+` (più) intervallati da spazi.

Suggerimento: potete utilizzare due cicli `for` annidati ed utilizzare l'operatore `%` per distinguere le righe pari da quelle dispari.

Esempio d'esecuzione:

```
$ go run quadrato_righe_alterne_1.go
Inserisci un numero: 5
* * * * *
+ + + + +
* * * * *
+ + + + +
* * * * *
```

3 Cicli annidati - Quadrato a righe alterne (2)

Scrivere un programma che legga da **standard input** un numero intero `n` e che, come mostrato nell'**Esempio di esecuzione**, stampi a video un quadrato di `n` righe costituite ciascuna da `n` simboli intervallati da spazi, alternando fra loro righe costituite solo da simboli `*` (asterisco) intervallati da spazi, righe costituite solo da simboli `+` (più) intervallati da spazi e righe costituite solo da simboli `o` (lettera o) intervallati da spazi.

Esempio d'esecuzione:

```
$ go run quadrato_righe_alterne_2.go
Inserisci un numero: 5
* * * * *
+ + + + +
o o o o o
* * * * *
+ + + + +
```

4 Stringhe/Rune - Analisi lettere maiuscole/minuscole

Scrivere un programma che legga da **standard input** una stringa senza spazi e, considerando **solamente** l'insieme delle lettere dell'alfabeto inglese, stampi

- il numero di lettere maiuscole;
- il numero di lettere minuscole;
- il numero di consonanti;
- il numero di vocali.

Suggerimento: Le lettere dell'alfabeto inglese sono considerate nello standard US-ASCII (integrato nello standard Unicode). Per i caratteri considerati nello standard US-ASCII, il codice Unicode varia tra 0 e 127. In particolare, per le lettere dell'alfabeto inglese il codice Unicode varia negli intervalli:

- [65, 90] per le lettere MAIUSCOLE (con 'A'=65 e 'Z'=90)
- [97, 122] per le lettere minuscole (con 'a'=97 e 'z'=122)

Esempio d'esecuzione:

```
$ go run analisi.go
Ciaoà
Maiuscole: 1
Minuscole: 3
Vocali: 3
Consonanti: 1
```

```
$ go run analisi.go
Certo!Sto,bene!iiiiii
Maiuscole: 2
Minuscole: 10
Vocali: 5
Consonanti: 7
```

```
$ go run analisi.go
aaAA
Maiuscole: 2
Minuscole: 2
```

Vocali: 4
Consonanti: 0

5 Stringhe/Rune - Trasformazione lettere maiuscole/minuscole

Scrivere un programma che legga da **standard input** una stringa e, considerando l'insieme delle lettere dell'alfabeto inglese, ristampi a video la stringa due volte: la prima volta in maiuscolo e la seconda volta in minuscolo.

Esempio d'esecuzione:

```
$ go run trasforma.go
TestoDiProva!!!
Testo maiuscolo: TESTODIPROVA!!!
Testo minuscolo: testodiprova!!!

$ go run trasforma.go
Testo_Di_PrOvA....
Testo maiuscolo: TESTO_DI_PROVA....
Testo minuscolo: testo_di_prova....
```

6 Stringhe/Rune - Parola palindroma

Definizione: Una parola è palindroma se può essere letta normalmente, da sinistra verso destra, sia viceversa, cioè da destra verso sinistra.

Scrivere un programma che:

- legga da **standard input** una stringa senza spazi;
- stampi a video il messaggio **Palindroma** nel caso in cui la stringa letta sia palindroma e **Non palindroma** altrimenti.

Esempio d'esecuzione:

```
$ go run palindroma.go
anna
Palindroma

$ go run palindroma.go
anni
Non palindroma

$ go run palindroma.go
osso
Palindroma
```



```
$ go run palindroma.go
èssè
Palindroma
```

```
$ go run palindroma.go
èsse
Non palindroma
```

7 Cicli annidati - Triangoli

Scrivere un programma che legga da **standard input** un intero $n > 1$ e stampi, utilizzando il carattere *, il perimetro di due triangoli rettangoli con base e altezza di lunghezza n , posizionati come mostrato nell'**Esempio d'esecuzione**.

Esempio d'esecuzione:

```
$ go run triangoli.go
2
**
 *
 *
**
```

```
$ go run triangoli.go
3
***
 **
  *
  *
 **
***
```

```
$ go run triangoli.go
4
****
 * *
  **
   *
   *
  **
 * *
****
```

```
$ go run triangoli.go 6
*****
 *   *
```

* *
* *
**
*
*
**
* *
* *
* *
