Laboratorio 6 - Stringhe, Strutture, Puntatori

(Stringhe) Qual è l'output?

Qual è l'output del seguente programma?

```go
package main

import "fmt"

func main() {

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

    fmt.Println(string(s[0]) + string(s[len(s)-2]) + string(s[len(s)-1]))
}
```

(Funzioni con output multipli, errori) Qual è l'output?

Qual è l'output del seguente programma?

```go
package main

import "fmt"

func main() {

    d, err := divide(5.0, 3.0)
    if err != nil {
        fmt.Println(err)
        return
    }

    fmt.Println(d)
}

func divide(a, b float64) (float64, error) {
    if b == 0.0 {
        return 0.0, fmt.Errorf("Errore, divisione per zero!")
    }
    return a/b, nil
}
```

(Stringhe) Qual è l'output?

Qual è l'output del seguente programma?

```go
package main

import "fmt"

func main() {

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

    fmt.Println(s[6:10] + string(s[len(s)-1]))
    fmt.Println(s[:5] + s[len(s)-4:])

}
```

(Strutture) Qual è l'output?

Qual è l'output del seguente programma?

```go
package main

import (
    "fmt"
)

type Persona struct {
    Nome    string
    Cognome string
}

func main() {

    var p1 Persona
    p1 = Persona{"Rick", "Sanchez"}

    var p2 Persona
    p2.Nome = "Jerry"
    p2.Cognome = "Smith"

    p3 := Persona{Nome: "Morty"}

    p2 = p3

    fmt.Println(p1, p2)
}
```

(Puntatori) Qual è l'output?

Qual è l'output del seguente programma?

```go
package main

import "fmt"

func main() {
    var a, b, c int
    var ptr *int
    a, b, c = 10, 15, 20
    ptr = &a
    *ptr += 10
    a += 10
    ptr = &b
    *ptr += 10
    *ptr = c
    *ptr += 10
    fmt.Println(a, b, c)
}
```

(Puntatori) Qual è l'output?

Qual è l'output di questo programma?

```go
package main

import "fmt"

func main() {
    var a, b = 15, 20
    f(&a, &b)
    fmt.Println(a, b)
}
func f(x, y *int) {
    *x, *y = *y, *x
}
```

(Puntatori) Trova l'errore

Questo programma dovrebbe stampare `30 20` ma non genera l'output desiderato. Corregere e verificare che l'esecuzione del programma generi l'output atteso.

```go
package main

import "fmt"
```

```go
func main() {
    var a, b int = 10, 20
    var ptr *int
    ptr = a
    *ptr = *ptr + b
    fmt.Println(a, b)
}
```

(Puntatori) Trova l'errore

Questo programma dovrebbe stampare `70 20` ma non genera l'output desiderato. Corregere e verificare che l'esecuzione del programma generi l'output atteso.

```go
package main

import "fmt"

func main() {
    var a, b int = 10, 20
    var ptr *int
    *ptr = 50
    ptr = &a
    *ptr += b
    fmt.Println(a, b)
}
```

(Puntatori) Qual è l'output?

Qual è l'output di questo programma?

```go
package main

import (
    "fmt"
    "strings"
)

type Persona struct {
    Nome, Cognome string
}

func main() {

    p := &Persona{"Rick", "Sanchez"}

    f(*p)
```

```go
    fmt.Println(*p)
    g(p)
    fmt.Println(*p)
}

func f(p Persona) {
    p.Nome, p.Cognome = strings.ToUpper(p.Nome), strings.ToUpper(p.Cognome)
}

func g(p *Persona) {
    p.Nome, p.Cognome = strings.ToUpper(p.Nome), strings.ToUpper(p.Cognome)
}
```

1 (Stringhe) Triangolo

Scrivere un programma che legga da **standard input** un numero intero `n` e, come mostrato nell'**Esempio di esecuzione**, stampi a video un triangolo rettangolo con base e altezza di lunghezza `n` utilizzando il carattere `*` (asterisco).

Se `n` è negativo o nullo, anziché stampare il triangolo il programma deve stampare un messaggio d'errore.

Nota: Si utilizzi un solo ciclo `for` ed una variabile di tipo `string`.

Esempio d'esecuzione:

```
$ go run triangolo.go
-2
Dimensione non sufficiente

$ go run triangolo.go
5
*
**
***
****
*****
```

2 (Stringhe) Ripetizione

Scrivere un programma che: 1. legga da **standard input** un numero intero `n` ed una stringa senza spazi; 2. stampi `n` volte la stringa letta, intervallando le `n` occorrenze della stringa con il carattere `-` (meno).

Esempio d'esecuzione:

```
$ go run ripetizione.go
```

```
5 test
test-test-test-test-test
```

3 (Stringhe) Unisci stringhe

Scrivere un programma che: 1. Legga da **standard input** una sequenza di stringhe senza spazi, una per linea. 2. La lettura termina quando viene letta una stringa contenente il solo carattere `"0"` e viene stampata una stringa che le unisce tutte (separate con spazi).

Esempio d'esecuzione:

```
$ go run unisci_stringhe.go
inserisci una stringa:
hello
inserisci una stringa:
world!
inserisci una stringa:
0
stringa completa: hello world!
```

4 (Stringhe) Maiuscole

Scrivere un programma che: 1. Legga da **standard input** una sequenza di stringhe senza spazi, una per linea. 2. Stampi ciascuna stringa convertita in maiuscolo. A tal fine è possibile utilizzare la funzione `strings.ToUpper` del package `strings`. 3. La lettura termina quando viene letta una stringa contenente il solo carattere `"0"` e viene stampato `ciao`.

Esempio d'esecuzione:

```
$ go run maiuscole.go
inserisci una stringa:
pippo
stringa in maiuscolo: PIPPO
inserisci una stringa:
0
ciao
```

5 (Stringhe) Somma stringhe

Scrivere un programma che: 1. Legga da **standard input** una sequenza di stringhe senza spazi, una per linea. 2. Converta ciascuna stringa in numero intero. A tal fine è possibile utilizzare la funzione `strconv.Atoi` del package `strconv`. 3. La lettura termina quando viene letta una stringa che non sia convertibile in intero (ovvero che contenga caratteri diversi da numeri interi) e viene stampata la somma dei numeri inseriti.

Suggerimento: E' possibile consultare la documentazione relativa al funzionamento della funzione `strconv.Atoi` mediante il comando `go doc strconv.Atoi` (da riga di comando)

Esempio d'esecuzione:

```
$ go run somma_stringhe.go
inserisci un intero:
3
inserisci un intero:
4
inserisci un intero:
a
somma: 7
```

6 (Stringhe) Ultimo cognome

Scrivere un programma che: 1. Legga da **standard input** una sequenza di cognomi (stringhe) senza spazi, uno per linea. 2. La lettura termina quando viene letta una stringa contenente il solo carattere `"0"` e viene stampato l'ultimo cognome in ordine alfabetico.

Suggerimento: Cosa produce il seguente spezzone di codice `Go`?

```go
var a string = "a"
var b string = "b"

fmt.Println(a > b)
```

Esempio d'esecuzione:

```
$ go run ultimo_cognome.go
inserisci un cognome:
rossi
inserisci un cognome:
verdi
inserisci un cognome:
bianchi
inserisci un cognome:
0
ultimo cognome: verdi
```

7 (Stringhe) Numero nascosto

Scrivere un programma che legga da **standard input** una stringa di testo senza spazi e stampi in output il doppio del numero nascosto all'interno della stringa di testo, ovvero il doppio del numero che si ottiene concatenando le cifre

presenti all'interno della stringa 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 la seguente funzione: * 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
Ci8ao97com3va?
Doppio del numero nascosto: 17946 (8973 * 2)

$ go run numero_nascosto.go
Ch3831t3mp0
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
```

8 (Strutture/puntatori) Frazioni

Al fine di modellare l'entità matematica `frazione`, si definisca un nuovo tipo `Frazione` come una struttura avente due campi `numeratore` e `denominatore` di tipo `int`.

Scrivere un programma che legga da **standard input** due interi `n` e `d` e restituisca la rappresentazione della frazione in formato stringa (i.e. `n/d`) inizializzando una `Frazione` utilizzando `n` come numeratore e `d` come denominatore.

Implementare le funzioni:

- `NuovaFrazione(numeratore, denominatore int) *Frazione` che restituisce un puntatore all'istanza del tipo `Frazione` inizializzata in base ai valori dei parametri `numeratore` e `denominatore`;

- `String(f Frazione) string` che riceve in input un'instanza del tipo `Frazione` nel parametro `f` e restituisce un valore `string` che corrisponde alla rappresentazione `string` di `f` nel formato `numeratore/denominatore`;

Suggerimento: E' possibile convertire un intero in stringa utilizzando il metodo `strconv.Itoa()` del pacchetto `strconv`.

Esempio d'esecuzione:

```
$ go run frazione.go
Inserisci due numeri interi: 34 18
Frazione: 34/18

$ go run frazione.go
Inserisci due numeri interi: 1 2
Frazione: 1/2
```

9 (Strutture/puntatori) Riduzione di una frazione ai minimi termini

Si consideri il tipo `Frazione` dell'esercizio precedente.

Si modifichi il programma in modo tale che: * riduca ai minimi termini la frazione; * stampi a video, nel formato `numeratore/denominatore`, la frazione ridotta ai minimi termini.

Oltre alla funzione `main()` e alle funzioni implementate nell'esercizio precedente, devono essere definite ed utilizzate almeno le seguenti funzioni: * una funzione `Riduci(f *Frazione)` che riceve in input un'instanza del tipo `Frazione` (puntatore) nel parametro `f` e, se necessario, modifica opportunamente il valore dei campi `f.numeratore` e `f.denominatore` affinché `f` rappresenti una frazione ridotta ai minimi termini.

Esempio d'esecuzione:

```
$ go run riduci.go 10 10
1/1

$ go run riduci.go 34 18
17/9

$ go run riduci.go 12 36
1/3
```

10 (Stringhe) Conversione da base 10 a base X (in [2,16])

Scrivere un programma che legga da **standard input** un numero in base 10 `n` e la nuova base `b` in cui `n` dovrà essere convertito e stampi a video il risultato

della conversione.

Oltre alla funzione `main()`, il programma deve definire e utilizzare: * una funzione `convert10ToBaseX(n, b uint) (string, bool)` che riceve in input due valori `uint` corrispondenti al numero in base 10 da convertire `n` e la base di arrivo `b`. La funzione restituisce il valore convertito in base `b` sottoforma di stringa (i valori oltre la base 10 contengono delle lettere) e un booleano. Se la base non è nell'intervallo 2 <= b <= 16 la funzione ritornerà `"", false`

Come funziona la conversione di base?

- l'algoritmo di conversione si basa sui resti della divisione di `n` per `b`, che rappresentano le cifre del risultato della conversione.
- si ricordi che se `b > 10` si dovranno usare delle lettere
- Assumendo che base sia al più 16, l'insieme delle possibili cifre sarà: `"0123456789ABCDEF"`

L'algoritmo di conversione può essere definito come di seguito:

```
Esempio: n = 31 e b = 16

1. Prendiamo il resto della divisione di n per b
    31 % 16 = 15

2. Prendiamo il quoziente della divisione di n per b
    dividiamo 31 / 16 = 1

3. Poniamo n uguale al quoziente e torniamo al punto 1 (finchè il quoziente è diverso da 0)


usiamo i resti ottenuti al punto 1. come indici della stringa "0123456789ABCDEF" per reperi

risultato: 31 (base 10) equivale a 1F (base 16)
```

Esempio di Esecuzione:

```
$ go run fromBase10.go 31 16
31 in base 16: 1F

$ go run fromBase10.go 298 16
298 in base 16: 12A

$ go run fromBase10.go 7 2
298 in base 16: 111
```

11 (Stringhe) Conversione da base X (in [2,16]) a base 10

Scrivere un programma che legga da **standard input** un valore `n` e la rispettiva base `b` (in [2,16]) e restituisca il valore di `n` convertito in base 10.

Oltre alla funzione `main()`, il programma deve definire e utilizzare: * una funzione `convertFromBaseXTo10(n string, b int) (int, bool)` che riceve in input un valore `string` e un `int` corrispondenti alla cifra `n` espressa in base `b`. La funzione restituisce il valore convertito in base 10 sottoforma di intero e un booleano. Se la base `b` non è nell'intervallo 2 <= b <= 16 oppure `n` contiene cifre non appartenenti alla base `b` (ad es, se b == 2 n può contenere solo le cifre '0' e '1'), la funzione ritornerà `0, false`, `numeroInBase10, true`, altrimenti.

Suggerimento: l'algoritmo di conversione si basa sulla rappresentazione posizionale dei numeri; si consideri la stringa `s := "0123456789ABCDEF"` l'indice `idx` (posizione nella stringa) di un elemento `c` nella stringa `s` può essere ottenuto mediante la funzione `idx = strings.Index(s, string(c))` del pacchetto `strings`. Es. `strings.Index(s, "E")` restituirà il valore `14`.

```
Esempio: "12A" in base b = 16 viene convertita in base 10 con:

b10 = 10 * 16^0 + 2 * 16^1 + 1 * 16^2  (la cifra 'A' occupa la posizione 10 nella stringa, I
```

Esempio di Esecuzione:

```
$ go run toBase10.go
  Inserisci un valore e la corrispondente base (in [2,16]) da convertire in base 10: 12A 16
  12A in base 10: 298

  go run toBase10.go
  Inserisci un valore e la corrispondente base (in [2,16]) da convertire in base 10: 1F 16
  1F in base 10: 31

  go run toBase10.go
  Inserisci un valore e la corrispondente base (in [2,16]) da convertire in base 10: 111 2
  111 in base 10: 7
```