

Laboratorio 6 - Recap di Tipi, Funzioni, Rune - Switch

1 (Tipi) Qual è l'output?

Qual è l'output del seguente programma?

```
package main
import "fmt"
func main() {
    var x int = 10
    var y float64 = 2.5
    t := 100
    t, z, k := x/int(y), float64(x)/y, t*2
    fmt.Println(t, z, k)
}
```

2 (Tipi) Qual è l'errore?

Si consideri il seguente frammento di codice.

```
package main
import "fmt"
func main() {
    var a, b int
    fmt.Scan(&a, &b)
    var c ???
    c = a == b

    if c {
        fmt.Println("uguali")
    } else {
        fmt.Println("diversi")
    }
}
```

Di che tipo deve essere la variabile `c` affinché la compilazione del codice non generi errori?

3 (Tipi) Trova l'errore

Questo programma dovrebbe stampare il valore delle variabili `a`, `b`, `c` e `d` ma contiene degli errori. Correggere gli errori e verificare che l'esecuzione produca l'output desiderato.

```
package main
import "fmt"
func main() {
```

```

var a int = 4
var b float64 = 12.5
var c int
c := a + b

var d float64
d = a/c

fmt.Println(a, b, c, d)
}

```

4 (Tipi) Valore di default delle variabili

Che cosa succede se si prova a utilizzare una variabile senza averla inizializzata? Si esegua il seguente programma per stampare a video i valori di default (valori iniziali/zero-values) per i tipi `int`, `float64`, `string` e `bool`.

```

package main
import "fmt"
func main() {
    var numeroIntero int
    var numeroReale float64
    var valoreLogico bool
    fmt.Print("Valore di default per il tipo int: ", numeroIntero, "_\n")
    fmt.Print("Valore di default per il tipo float64: ", numeroReale, "_\n")
    fmt.Print("Valore di default per il tipo bool: ", valoreLogico, "_\n")
}

```

5 (Tipi) Operatori e precedenza

Si consideri il seguente frammento di codice.

```

package main
import "fmt"
func main() {
    var a, b, c int = 10, 5, 4
    fmt.Println( a / b * c )
}

```

Quale valore viene stampato a video?

6 (Tipi) Overflow

Qual è l'output del seguente programma?

```

package main
import "fmt"
func main() {

```

```

var (
    a, b int8 = 30, 100
)
somma := a + b
fmt.Println("La somma di", a, "e", b, "è", somma)
}

```

7 (Tipi) Valori rappresentabili

Il seguente programma stampa a video i limiti dei valori rappresentabili con i diversi tipi di dato numerici. Per ogni tipo di dato, i corrispondenti limiti sono definiti come costanti nel package `math`. Si stampi a video il valore di tali limiti eseguendo il programma.

```

package main
import (
    "fmt"
    "math"
)
func main() {
    fmt.Println("uint8:", 0, math.MaxUint8)
    fmt.Println("uint16:", 0, math.MaxUint16)
    fmt.Println("uint32:", 0, math.MaxUint32)
    fmt.Println("uint64:", 0, uint64(math.MaxUint64))
    fmt.Println("int8:", math.MinInt8, math.MaxInt8)
    fmt.Println("int16:", math.MinInt16, math.MaxInt16)
    fmt.Println("int32:", math.MinInt32, math.MaxInt32)
    fmt.Println("int64:", math.MinInt64, math.MaxInt64)
    /* Floating-point limit values:
       - Max is the largest finite value representable by the type.
       - SmallestNonzero is the smallest positive, non-zero value
         representable by the type. */
    fmt.Println("float32 - SmallestNonzero:", math.SmallestNonzeroFloat32)
    fmt.Println("float32:", -math.MaxFloat32, math.MaxFloat32)
    fmt.Println("float64 - SmallestNonzero:", math.SmallestNonzeroFloat64)
    fmt.Println("float64:", -math.MaxFloat64, math.MaxFloat64)
}

```

8 (Switch) Qual è l'output?

Supponendo che l'utente inserisca da **standard input** il valore 10, cosa dovrebbe produrre in output il seguente programma? E se invece inserisse 4?

```

package main

import "fmt"

```

```
func main() {

    var num int = 1

    switch num {
    case 1:
        fmt.Println("Uno")
    case 2:
        fmt.Println("Due")
    default:
        fmt.Println("Ne uno, ne due")
    }
}
```

9 (Switch) Qual è l'output?

Supponendo che l'utente inserisca da **standard input** il valore 10, cosa dovrebbe produrre in output il seguente programma? E se invece inserisse 4?

```
package main

import "fmt"

func main() {

    var voto int

    fmt.Scan(&voto)

    switch voto {
    default:
        fmt.Println("Insufficiente!")
    case 10:
        fallthrough
    case 9:
        fmt.Println("Ottimo!")
    case 8:
        fmt.Println("Distinto!")
    case 7:
        fmt.Println("Buono!")
    case 6:
        fmt.Println("Sufficiente!")
    }
}
```

10 (Switch) Qual è l'output?

Supponendo che l'utente inserisca da **standard input** il valore 9, cosa dovrebbe produrre in output il seguente programma? E se invece inserisse 13?

```
package main

import "fmt"

func main() {

    var n int

    fmt.Scan(&n)

    var somma int

    for i:=0; i<=n; i++ {

        switch i%2 {
        case 0:
            fmt.Println(i, "pari!")
        case 1:
            continue
        }

        somma += i
    }

    fmt.Println("Somma =", somma)
}
```

11 (Switch) Qual è l'output?

Supponendo che l'utente inserisca da **standard input** il valore 9, cosa dovrebbe produrre in output il seguente programma? E se invece inserisse 13?

```
package main

import "fmt"

func main() {

    var n int

    fmt.Scan(&n)
```

```

var somma int

for i:=0; i<=n; i++ {

    switch i%2 {
    case 0:
        fmt.Println(i, "pari!")
    case 1:
        break
    }

    somma += i
}

fmt.Println("Somma =", somma)
}

```

12 (Switch) Qual è l'output?

Supponendo che l'utente inserisca da **standard input** il valore 3, cosa dovrebbe produrre in output il seguente programma? E se invece inserisse 100?

```

package main

import "fmt"

func main() {

    var n int

    fmt.Scan(&n)

    accumulatore := 1

    for i:=1; i<=n; i++ {

        switch {
        case i<5:
            accumulatore *= i
        case i<10:
            accumulatore += i
        }

    }

    fmt.Println("Accumulatore =", accumulatore)
}

```

```
}
```

13 (Funzioni) Tabelline

Scrivere un programma che legga da **standard input** una sequenza di numeri interi compresi tra 1 e 9 (estremi inclusi) e stampi per ognuno di essi la tabellina corrispondente. Il programma si interrompe quando viene inserito in input un numero non valido (inferiore a 1 o superiore a 9) stampando **Programma terminato..**

Oltre alla funzione `main()`, il programma deve definire e utilizzare le seguenti funzioni: `* Tabellina(numero int) bool` che riceve in input un valore intero nel parametro `numero`. Se `numero` è compreso tra 1 e 9 (estremi inclusi), la funzione stampa la tabellina associata e restituisce un valore logico `true`. Altrimenti, la funzione non stampa nulla e restituisce un valore logico `false`.

Esempio d'esecuzione:

```
$ go run tabellina.go
Inserisci un numero: 6
Tabellina del 6: 0 6 12 18 24 30 36 42 48 54 60
Inserisci un numero: 4
Tabellina del 4: 0 4 8 12 16 20 24 28 32 36 40
Inserisci un numero: 13
Programma terminato.
```

14 (Funzioni) Scacchiera

Scrivere un programma che legga da **standard input** un numero intero e, come mostrato nell'**Esempio d'esecuzione**, stampi a video una scacchiera di dimensione `n x n` utilizzando i caratteri `*` (asterisco) e `+` (più).

Oltre alla funzione `main()`, il programma deve definire e utilizzare le seguenti funzioni: `* StampaRigaInizioAsterisco(lunghezza int)` che riceve in input la lunghezza della riga da stampare nel parametro `lunghezza` e stampa in modo alternato i caratteri `*` e `+` (partendo dal carattere `*`); `* StampaRigaInizioPiù(lunghezza int)` che riceve in input la lunghezza della riga da stampare nel parametro `lunghezza` e stampa in modo alternato i caratteri `+` e `*` (partendo dal carattere `+`); `* StampaScacchiera(dimensione int)` che riceve in input la dimensione della scacchiera da stampare nel parametro `dimensione` e stampa la scacchiera utilizzando le funzioni `StampaRigaInizioAsterisco()` e `StampaRigaInizioPiù()`. Se `dimensione <= 0`, non stampa nulla.

Esempio d'esecuzione:

```
$ go run scacchiera.go
Inserisci la dimensione: 4
```

```
****
****
****
****
```

```
$ go run scacchiera.go
Inserisci la dimensione: 6
*****
*****
*****
*****
*****
*****
```

```
$ go run scacchiera.go
Inserisci la dimensione: -1
```

```
$ go run scacchiera.go
Inserisci la dimensione: 0
```

15 (Funzioni) Numeri primi gemelli

Definizione: Due numeri primi p e q sono gemelli se $p = q + 2$.

Scrivere un programma che legga da **standard input** un numero intero **soglia** e stampi tutti i numeri primi gemelli tali che p sia inferiore a **soglia**. Se **soglia** ≤ 0 il programma deve stampare **La soglia inserita non è positiva**.

Oltre alla funzione `main()`, il programma deve definire e utilizzare le seguenti funzioni: * una funzione `ÈPrimo(n int) bool` che riceve in input un valore intero nel parametro `n` e restituisce `true` se `n` è primo e `false` altrimenti; * una funzione `NumeriPrimiGemelli(limite int)` che riceve in input un valore intero nel parametro `limite` e stampa tutte le coppie di numeri primi gemelli tali che p sia inferiore a `limite` (cfr. Definizione); la funzione deve utilizzare la funzione `ÈPrimo()`.

Esempio d'esecuzione:

```
$ go run numeri_primi_gemelli.go
Inserisci un numero: -4
La soglia inserita non è positiva
```

```
$ go run numeri_primi_gemelli.go
Inserisci un numero: 10
Numeri primi gemelli inferiori a 10
(3,5) (5,7)
```



```
$ go run numeri_primi_gemelli.go
Inserisci un numero: 20
Numeri primi gemelli inferiori a 20
(3,5) (5,7) (11,13) (17,19)
```

16 (Funzioni) Terna pitagorica

Definizione: Se a , b e c sono numeri naturali e $a^2 + b^2 = c^2$, si dice che la terna di numeri a , b e c è una terna pitagorica.

Scrivere un programma che legga da **standard input** un intero `soglia`>0 e stampi a video tutte le terne pitagoriche tali che $a < \text{soglia}$, $b < \text{soglia}$ e $c < \text{soglia}$.

Oltre alla funzione `main()`, devono essere definite ed utilizzate almeno le seguenti funzioni: `* ÈTernaPitagorica(a int, b int, c int) bool` che riceve in input tre valori interi nei parametri a , b e c , e restituisce `true` se c^2 è uguale a $a^2 + b^2$ e `false` altrimenti; `* TernePitagoriche(soglia int)` che riceve in input un valore intero nel parametro `soglia` e stampa tutte le terne pitagoriche inferiori a `soglia`; la funzione deve utilizzare la funzione `ÈTernaPitagorica()`.

Esempio d'esecuzione:

```
$ go run terna_pitagorica.go
Inserisci la soglia: 10
Terne pitagoriche:
(3, 4, 5)
(4, 3, 5)
```

```
$ go run terna_pitagorica.go
Inserisci la soglia: 20
Terne pitagoriche:
(3, 4, 5)
(4, 3, 5)
(5, 12, 13)
(6, 8, 10)
(8, 6, 10)
(8, 15, 17)
(9, 12, 15)
(12, 5, 13)
(12, 9, 15)
(15, 8, 17)
```

17 (Rune) Carte

Sapendo che al codice Unicode 127153 (associato alla rappresentazione in bit Unicode/UTF-8 `'\U0001F0B1'`) corrisponde il simbolo “asso di cuori”, e che i

codici successivi corrispondono alle carte successive (2 di cuori, 3 di cuori, ...), scrivere un programma che stampi tutte le carte da gioco dall'asso di cuori al 10 di cuori.

Suggerimento: Un carattere è una variabile di tipo **rune**, il cui valore è un intero corrispondente al codice Unicode del carattere. Le istruzioni equivalenti `var c rune = 127153` e `var c rune = '\U0001F0B1'` servono a definire la variabile `c` di tipo **rune** ed inizializzarla al valore “asse di cuori”. Per stampare la carta da gioco “asse di cuori” si può utilizzare l’istruzione `fmt.Print(string(c))` o `fmt.Printf("%c", c)`.

```
$ go run carte.go
Simbolo: - Codice numerico in base 10: 127153
Simbolo: - Codice numerico in base 10: 127154
Simbolo: - Codice numerico in base 10: 127155
Simbolo: - Codice numerico in base 10: 127156
Simbolo: - Codice numerico in base 10: 127157
Simbolo: - Codice numerico in base 10: 127158
Simbolo: - Codice numerico in base 10: 127159
Simbolo: - Codice numerico in base 10: 127160
Simbolo: - Codice numerico in base 10: 127161
Simbolo: - Codice numerico in base 10: 127162
```

18 (Rune) Spaziatura caratteri

Scrivere un programma che: 1. legga da **standard input** una stringa senza spazi ed interamente definita da lettere dell’alfabeto inglese; 2. stampi la stessa stringa in modo tale che ogni lettera sia separata da quella successiva da uno spazio.

Esempio d’esecuzione:

```
$ go run spazia.go
Inserisci una stringa di testo: CiaoMondo!
C i a o M o n d o !
```

19 (Rune) Stringa alternata

Scrivere un programma che legga da **standard input** due stringhe senza spazi `s1` e `s2` e stampi a video la stringa creata alternando i caratteri delle stringhe `s1` e `s2`.

A tal fine utilizzare una funzione `StringheAlternate(s1, s2 string) (risultato string)`

Esempio: Se `"ciao!"` e `"MONDO"` sono le stringhe lette, allora la stringa stampata a video deve essere `"cMiOaN0D!0"`.

Si assuma che le stringhe lette siano interamente definite da caratteri considerati nello standard US-ASCII.

Se le stringhe lette non sono definite dallo stesso numero di caratteri, si deve utilizzare il carattere '-' come carattere di riempimento:

Esempio: Se "ciao" e "mondo!", sono le stringhe lette, allora la stringa stampata video deve essere "cmioanod-o-!".

Esempio d'esecuzione

```
$ go run stringa_alternata.go
ciao
mondo
cmioanod-o
```

```
$ go run stringa_alternata.go
ciaone
mondo
cmioanodnoe-
```

```
$ go run stringa_alternata.go
esame
go
egsoa-m-e-
```

20 (Switch) Analisi lettere maiuscole/minuscole

Scrivere un programma che legga da **standard input** una stringa senza spazi e, considerando l'insieme delle lettere dell'alfabeto inglese, stampi * il numero di vocali maiuscole; * il numero di vocali minuscole; * il numero di consonanti maiuscole; * il numero di consonanti minuscole.

A tal fine definire la funzione: 'èVocale(l rune) bool'.

- Utilizzare il costrutto switch per il controllo delle vocali nella funzione 'èVocale'
- Utilizzare le funzioni 'unicode.IsLetter' e 'unicode.IsUpper' del package 'unicode' per controllare che un carattere sia una lettera valida/maiuscola, rispettivamente.

Esempio d'esecuzione:

```
$ go run analisi.go
Ciao
Vocali maiuscole: 0
Consonanti maiuscole: 1
Vocali minuscole: 3
```

```
Consonanti minuscole: 0
```

```
$ go run analisi.go
Certo!Sto,bene
Vocali maiuscole: 0
Consonanti maiuscole: 2
Vocali minuscole: 5
Consonanti minuscole: 5
```

```
$ go run analisi.go
aaAA
Vocali maiuscole: 2
Consonanti maiuscole: 0
Vocali minuscole: 2
Consonanti minuscole: 0
```

21 (Switch) Menu

Scrivere un programma che permetta di ordinare al fast food con consegna a domicilio. Il programma deve chiedere iterativamente da **standard input** il numero corrispondente al tipo di articolo (1 patatine,2 hamburger o 3 cocacola) e la quantità richiesta. Con 0 viene terminata la lettura e viene stampato il costo dell'ordine. I prezzi sono 2€ per le patatine 5€ per gli hamburger e 2€ per la cocacola. In più ci sono 2€ di spesa per la consegna.

Si utilizzi il costrutto switch per selezionare l'articolo ed aggiornare il totale in base al numero inserito.

Esempio d'esecuzione:

```
$ go run menu.go
Cosa vuoi ordinare?
    1. patatine 2€
    2. hamburger 5€
    3. cocacola 2€
    0. termina
1
patatine? ottimo, quante?
3
Cosa vuoi ordinare?
    1. patatine 2€
    2. hamburger 5€
    3. cocacola 2€
    0. termina
2
hamburger? ottimo, quanti?
2
```

Cosa vuoi ordinare?

- 1. patatine 2€
- 2. hamburger 5€
- 3. cocacola 2€
- 0. termina

3

cocacola? ottimo, quante?

1

Cosa vuoi ordinare?

- 1. patatine 2€
- 2. hamburger 5€
- 3. cocacola 2€
- 0. termina

0

Sono 18 euro + 2 di consegna. Totale: 20