Esercizi di Go

23 ottobre 2018

Indice

Convenzione per i nomi dei file Chiamare i file dei programmi con il nome indicato. In particolare la convenzione è: `lez`n`_nomefile.go`, dove n è il numero della lezione di Go (la numero 1 la settimana dell'8 ottobre) e `nomefile` è un nome indicativo del problema proposto, ad esempio `lez1_area_cerchio.go`.

1 Esercizi per la seconda lezione di Go

Argomenti selezione: il costrutto `if`.

1.0.1 pari-dispari

Scrivere un programma Go `lez2_pari_dispari.go` che legge un intero n e, a seconda del valore di n, stampa uno dei messaggi

```
n è pari
```

oppure

```
n è dispari
```

Esempi di esecuzione

```
numero?
4
4 è pari

numero?
15
15 è dispari
```

1.0.2 maggiore

Scrivere un programma Go `lez2_maggiore.go` che legge due interi e stampa il maggiore.

Annotazioni In Go è possibile lo scambio diretto di valori tra due variabili:

```
a, b = b, a
```

Esempi di esecuzione

```
due numeri interi:
-3 12
12 è il maggiore

due numeri interi:
15 5
15 è il maggiore
```

1.0.3 indovina 10

Scrivere un programma Go `lez2_indovina_1_10.go` che fissa un numero intero tra 1 e 10 da indovinare, legge un intero da standard input e,

- se il numero in input e fuori dall'intervallo 1-10, stampa "Valore non valido";
- se il numero è quello fissato, stampa "Hai indovinato!";
- altrimenti stampa "Ciao".

Esempi di esecuzione

se il numero fissato è 2:

```
un numero intero:
2
Hai indovinato!

un numero intero:
7
Ciao

un numero intero:
22
Valore non valido
```

1.0.4 validità voto

Scrivere un programma Go `lez2_voto_valido.go` che legge un numero intero. Se il numero non è compreso tra 0 e 30, stampa "voto non valido".

Esempio di esecuzione

```
voto?
42
voto non valido

voto?
28
```

1.0.5 tasse

Scrivere un programma Go `lez2_tasse.go` che chiede reddito e stato civile (0 per non coniugato, 1 per coniugato) e calcola le tasse da pagare secondo la seguente tabella:

non coniugato		coniugato	
0 - 32000	10%	0 - 64000	10%
$>$ 32000	25%	$>$ 64000	25%

Scrivere una seconda soluzione utilizzando un if solo (quindi senza annidamenti) e gli operatori logici.

1.0.6 voti in lettere

Scrivere un programma Go `lez2_esiti.go` che associa voti in lettere a punteggi secondo la seguente tabella:

90 - 100	A
80 - 89	B
70 - 79	C
60 - 69	D
0 - 59	F

Più precisamente il programma riceve in ingresso un valore intero tra 0 e 100 e stampa su monitor il voto in lettere corrispondente.

Annotazioni Potendo dare per scontato che il valore *voto* fornito in ingresso sarà *voto* <= 100, se il primo if è

if(90 <= voto)

come sarà la codizione dell'else-if che lo segue? È necessario controllare che il voto sia minore di 90 oltre che maggiore o uguale a 80? Cioè:

if(80 <= voto && voto < 90)

O basta verificare solo che il voto sia maggiore o uguale a 80? Cioè:

if(80 <= voto)

Scegliere la soluzione che fa il minor numero possibile di controlli.

Esempi di esecuzione

```
Voto?
100
A

Voto?
65
D
```

1.0.7 fizz buzz

Problema Scrivere un programma Go `lez2_fizz_buzz.go` che riceve in ingresso un numero intero e stampa "Fizz" se il numero è multiplo di 3, "Buzz" se il numero è multiplo di 5, il numero stesso altrimenti.

Annotazioni Si osservi che le condizioni non sono mutuamente esclusive e che quindi non si prestano a essere gestite con un `if ...else if ....`

Esempi di esecuzione

```
numero?
5
Buzz
```

```
numero?
4
4

numero?
15
Fizz Buzz

numero?
6
Fizz
```

1.0.8 anno bisestile

Scrivere un programma Go `lez2_bisestile.go` che legge un intero n corrispondente all'anno di una data, e determina se l'anno è bisestile o no.

Annotazioni Il calendario gregoriano si applica dal 1582, anno della sua introduzione. Benché, in via teorica, sia possibile estenderlo anche agli anni precedenti, per questi, di norma, si usa il calendario giuliano.

Sono bisestili tutti gli anni divisibili per 4, compresi quelli secolari, dal 4 al 1580. Per gli anni precedenti non si applicano gli anni bisestili. Per gli anni dal 1582 (calendario gregoriano) sono bisestili:

- gli anni non secolari il cui numero è divisibile per 4;
- gli anni secolari il cui numero è divisibile per 400.

Per esempio, il 1896 e il 1996 sono stati entrambi bisestili (non secolari divisibili per 4), il 1800 e il 1900 non lo sono stati (secolari non divisibili per 400), mentre il 1600 e il 2000 lo sono stati (secolari divisibili per 400).

Esempi di esecuzione

```
anno:
1996
bisestile

anno:
1900
non bisestile

anno:
2000
bisestile

anno:
1998
non bisestile
```

1.0.9 validità di una data

versione 1 Scrivere un programma Go `lez2_data_valida1.go` che legge due interi g e m rappresentanti giorno e mese dell'anno, e verifica che sia una data valida. In questa versione si assuma che tutti i mesi abbiano 31 giorni.

versione 2 Scrivere una seconda versione `lez2_data_valida2.go` in cui si tiene conto del fatto che solo i mesi 1, 3, 5, 7, 8, 10, 12 hanno 31 giorni, che i mesi 4, 6, 9, 11 ne hanno 30, e si assuma che febbraio ne abbia sempre 28.

versione 3 Scrivere una terza versione `lez2_data_valida3.go` che legge tre interi g, m e a (anno) e tiene conto anche degli anni bisestili.

1.0.10 sconti

Consideriamo la seguente tabella che associa uno sconto a una persona a seconda della sua età

età	sconto
0 - 9	50%
10 - 16	30%
17 - 25	20%
26 - 45	30%
46 - 59	20%
60 - 75	30%
76	50%

Scrivere un programma Go `lez2_sconti.go` che legge un intero $n >= 0$, rappresentante l'età di una persona, e stampa lo sconto associato. Si assuma che il numero inserito sia maggiore o uguale a 0 (quindi non è necessario fare il controllo $n >= 0$).

Annotazioni Si noti che gli sconti possono assumere solamente i valori 20, 30 e 50. Quindi il codice deve avere la struttura:

```
leggi n
if(CONDIZIONE_1)
stampa 20
else if( CONDIZIONE_2)
stampa 50
else
stampa 30
```

dove CONDIZIONE_1 e CONDIZIONE_2 vanno opportunamente definite usando gli operatori booleani.

Esempi di esecuzione

```
Età:
```

```
18
sconto: 20

Età:
35
sconto: 30

Età:
101
sconto: 50
```

2 Esercizi supplementari

2.0.1 codifica del tipo int

Scrivere un programma Go `lez2_tipo_int.go` che verifica se il tipo int è codificato con 32 o 64 bit.

Suggerimento sfruttare le costanti MaxInt del package "math" (vedere la documentazione) e l'overflow.

2.0.2 indovina un numero

Scrivere un programma Go `lez2_indovina.go` che fissa un numero intero da indovinare, legge un intero da standard input e, se il numero è quello fissato, stampa "Hai indovinato!", altrimenti stampa "Ciao".

Esempi di esecuzione

se il numero fissato è 22:

```
un numero intero:
22
Hai indovinato!


un numero intero:
9
Ciao
```

2.0.3 validità orario

Scrivere un programma Go `lez2_orario.go` che controlla la validità dei dati di inizio e fine di un impegno e stampa un messaggio solo nel caso la fine preceda l'inizio. Al termine il programma stampa "STOP".

Si assuma che gli impegni siano a ore intere (tipo int).

Indicazioni Mettere fuori dall'if le istruzioni la cui esecuzione non è condizionata dagli orari letti.

Esempi di esecuzione

```
Ora inizio:
20
Ora fine:
18
Attenzione, la fine è prima dell'inizio!
STOP

Ora inizio:
20
Ora fine:
21
STOP
```

2.0.4 categorie

Scrivere un programma Go `lez2_categoria.go` che chiede l'età (un numero intero) e a seconda dell'età fornita, stampa "junior" (se l'età è inferiore a 18), "senior" (se è maggiore o uguale a 65), "adult" (se è tra 18 e 64).

Esempi di esecuzione

```
età?
25
adult

età?
14
junior

età?
68
senior
```

2.0.5 equazione di II grado

Scrivere un programma Go `lez2_radici_eq.go` che, dati tre numeri a, b, c in input, determina le radici dell'equazione di secondo grado $ax^2 + bx + c = 0$. In particolare

- se l'equazione non ha soluzioni reali, stampa "nessuna soluzione";
- se l'equazione ha una soluzione reale, stampa "una soluzione per x = . . . ";

- se l'equazione ha due soluzioni reali, stampa "x1 = ...; x2 = ...";

2.0.6 appartenenza a un rettangolo

Scrivere un programma Go `lez2_punto_in_rettangolo.go` che legga sei interi (coordinate dei punti A e B diagonali di un rettangolo e coordinate di un punto P) e che calcola se il punto P è fuori o dentro il rettangolo AB.

2.0.7 appartenenza a un cerchio

Scrivere un programma Go `lez2_punto_in_cerchio.go` che legga 5 interi (coordinate dei punti C del centro di un cerchio, raggio e coordinate di un punto P) e che calcola se il punto P è fuori o dentro il cerchio

2.0.8 appartenenza a una retta

Scrivere un programma Go `lez2_punto_su_retta.go` che legga 4 float (parametri della retta m e q, coordinate di un punto P) e che calcola se il punto P appartiene alla retta.

2.0.9 float intero?

Scrivere un programma Go `lez2_float_intero.go` che prenda in ingresso un float64 e determini se esso sia anche un int. (Esempio: 5.0 è anche un intero, mentre 5.31 no)

(Opzionale: se la risposta è no arrotondarlo all'intero più vicino, senza usare il package math)