Tutorato di programmazione

Azzeramento - Prima esercitazione

Indice

1 Conversione temperatura

Considerate il seguente programma `gradi.go`. Il programma legge una temperatura in gradi Fahrenheit e stampa l'equivalente in gradi centigradi.

```
1  package main
2
3  import "fmt"
4
5  func main() {
6          var fahr float64
7          const FATTORE_SCALA, ZERO = 5.0 / 9.0, 32.0
8
9          fmt.Println("Inserisci la temperatura in gradi fahrenheit: ")
10         fmt.Scan(&fahr)
11         cels := (fahr - ZERO) * FATTORE_SCALA
12         fmt.Println("Gradi Celsius equivalenti:", cels)
13 }
```

Analizzate il codice sorgente e rispondete per iscritto alle seguenti domande. Se avete dubbi, potete testarlo, eseguendolo su casi di input significativi e modificandolo.

1. Perché `FATTORE_SCALA` e `ZERO` sono dichiarate `const`?
2. A cosa serve l'istruzione alla riga 9?
3. A cosa serve l'istruzione alla riga 10?
4. Nella definizione di riga 7, è possibile sostituire `5.0` con `5`? È possibile sostituire `9.0` con `9`? È possibile sostituirli entrambi contemporaneamente? Giustificate la risposta.
5. Di che tipo è la variabile `cels`? Giustificate la risposta.
6. A cosa serve la variabile `cels`? È davvero necessaria? Provate a modificare il programma in modo da non usarla (e senza sostituirla con un'analoga variable con altro nome!).

2 Tre casi

Considerate il seguente programma `tre_casi.go`.

```
1  package main
2
3  import (
4          "fmt"
5          "os"
6  )
7
8  func main() {
9          var n int
10
11         fmt.Println("Inserisci un numero:")
12         fmt.Scan(&n)
13
14         switch n {
15         case 1:
16                 os.Args[2], os.Args[3] = os.Args[3], os.Args[2]
17         case 2:
18                 os.Args[1], os.Args[3] = os.Args[3], os.Args[1]
19         case 3:
20                 os.Args[1], os.Args[2] = os.Args[2], os.Args[1]
21         }
22         fmt.Println(os.Args[1], os.Args[2], os.Args[3])
23 }
```

Analizzate il codice sorgente e rispondete per iscritto alle seguenti domande. Se avete dubbi, potete testarlo, eseguendolo su casi di input significativi e modificandolo.

1. Cosa si aspetta in input il programma? Distinguete tra standard input e argomenti da linea di comando.
2. Descrivete cosa stampa il programma se riceve da standard input 5.
3. Scrivi il valore di `os.Args[0]` se si eseguono da terminale i comandi:

   ```
   go build tre_casi.go
   ./tre_casi 7 12 6
   ```

4. Qual è l'output del programma quando si eseguono da terminale i comandi precedenti e si inserisce 2 da standard input?
5. Riassumete il comportamento del programma con una frase del tipo: "Il programma legge . . . da riga di comando e . . . da standard input, quindi . . . infine stampa su standard output . . . ". La frase deve tenere conto dei vari casi che si possono verificare.

3 Classificazione triangoli

Considerate il seguente programma `triangoli.go`. Il programma riceve da standard input le lunghezze dei lati di un triangolo e classifica il triangolo come equilatero, isoscele o scaleno.

```
1  package main
2
3  import "fmt"
4
5  func main() {
6          var a, b, c float64
7
8          fmt.Println("Inserisci tre numeri separati da spazi:")
9          fmt.Scanln(&a, &b, &c)
10         if a == b && a == c {
11                 fmt.Println("Il triangolo e` ____________________")
12         } else if a == b || a == c || b == c {
13                 fmt.Println("Il triangolo e` ____________________")
14         } else {
15                 fmt.Println("Il triangolo e` ____________________")
16         }
17 }
```

1. Completate le righe 10, 12 e 14 in modo che il programma stampi l'output corretto.
2. Modificate il programma affinché prima di procedere con la classificazione verifichi questa proprietà fondamentale: in un triangolo, ogni lato è più corto della somma degli altri due. Salvate il programma con nome `triangoli_bis.go` e caricatelo sul sito upload.

4 Numeri

Considerate il seguente programma `numeri.go`.

```
1  package main
2
3  import "fmt"
4
5  func main() {
6          var n int
7          fmt.Scan(&n)
8
9          x := 0
10         for n != 0 {
11                 x += n
12                 fmt.Scan(&n)
13         }
14         fmt.Println(x)
15 }
```

Analizzate il codice sorgente e rispondete per iscritto alle seguenti domande

1. Cosa si aspetta in input il programma?
2. Cosa deve avvenire affinché il programma termini?
3. Date un nome più significativo alla variabile `x`.
4. Il programma tiene traccia della sequenza dei numeri letti? Come?
5. Riassumete il comportamento del programma con una frase del tipo: "Il programma legge da standard input . . . , calcola . . . e stampa . . . ".
6. Modificate il programma in modo che legga esattamente 10 numeri diversi da 0, stampi il risultato e termini. Salvate il nuovo programma con nome `numeri_bis.go` e caricatelo sul sito di upload.
7. Modificate il programma `numeri.go` in modo che che legga da standard input una sequenza di interi terminata da 0 e calcoli la media dei numeri della sequenza. Salvate il nuovo programma con nome `numeri_ter.go` e caricatelo sul sito di upload. *Suggerimento: cosa deve fare il programma nel caso in cui l'input sia formato solo dal numero 0?*

5 Array

Considerate il seguente programma `array.go`. Analizzate il codice sorgente e rispondete per iscritto alle seguenti domande.

```
1  package main
2
3  import "fmt"
4
5  func main() {
6          var s [10]int
7          var n int
8
9          fmt.Println("Inserisci 10 numeri:")
10         for i := 0; i < 10; i++ {
11                 fmt.Scan(&s[i])
12         }
13         fmt.Println("Inserisci un numero:")
14         fmt.Scan(&n)
15
16         p := -1
17         for i, z := range s {
18                 if z == n {
19                         p = i
20                 }
21         }
22         fmt.Println(p)
23 }
```

1. Cosa rappresentano le variabili `i` e `p` usate nel ciclo `for` nelle righe 16-20?
2. Riassumete con una frase cosa fa il programma.
3. Perché la variabile `p` è inizializzata a -1?
4. Scrivete un esempio di input su cui il programma stampa -1.
5. Riuscite a modificare il programma in modo da evitare l'uso della variabile `p` e in particolare dell'assegnamento in riga 19? Salvate il programma con nome `array_bis.go` e caricatelo sul sito di upload.
6. Modificate il programma in modo che legga da riga di comando il numero da memorizzare nella variabile `n`. Salvate il programma con nome `array_ter.go` e caricatelo sul sito di upload. Suggerimenti: 1) potrebbe esservi utile la funzione `Atoi` del package `strconv`; 2) è necessario memorizzare tutti i numeri inseriti da standard input?

6 Matrice

Considerate il seguente programma `matrice.go`. Analizzate il codice e rispondete per iscritto alle seguenti domande

```
1  package main
2  import "fmt"
3
4  func main() {
5          const R, C = 3, 4
6          var a [R][C]int
7
8          for i := 0; i < R; i++ {
9                  for j := 0; j < C; j++ {
10                         fmt.Scanln(&a[i][j])
11                 }
12         }
13
14         for i := 0; i < R; i++ {
15                 for j := 0; j < C; j++ {
16                         a[i][j] *= 2
17                 }
18         }
19
20         fmt.Println(a)
21 }
```

1. Cosa indica la variabile `i`?
2. A cosa serve il ciclo nella riga 15?
3. Riassumete in una frase cosa fa il programma.

7 Il cifrario di Cesare

Svetonio nella *Vita dei dodici Cesari* racconta che Giulio Cesare usava per le sue corrispondenze riservate un codice di sostituzione molto semplice. Il seguente programma `cifrario.go` implementa il cifrario di Cesare: il programma legge da standard input il testo da cifrare e un numero k (detto *chiave* di cifratura) ed emette in output il testo cifrato. Analizzate il codice sorgente e rispondete per iscritto alle domande sotto.

```
1  package main
2
3  import (
4          "bufio"
5          "fmt"
6          "os"
7          "unicode"
8  )
9
10 func main() {
11         var k int
12         var s string
13         const ALPHABET_LEN = 26
14
15         scanner := bufio.NewScanner(os.Stdin)
16         fmt.Println("Inserisci la stringa da cifrare:")
17         if scanner.Scan() {
18                 s = scanner.Text()
19         }
20         fmt.Println("Inserisci la chiave:")
21         fmt.Scan(&k)
22
23         for _, char := range s {
24                 if unicode.IsLetter(char) {
25                         var baseLetter int
26                         if char >= 'a' && char <= 'z' {
27                                 baseLetter = 'a'
28                         } else {
29                                 baseLetter = 'A'
30                         }
31                         n := int(char) - baseLetter
32                         n = (n + k) % ALPHABET_LEN
33                         char = rune(baseLetter + n)
34                 }
35                 fmt.Print(string(char))
36         }
37 }
```

1. Senza eseguire il programma, stabilite qual è l'output quando il programma riceve da standard input
   `allo Zoo`
   `5`
2. A cosa serve l'underscore alla riga 23? Che valore assumerebbe `char` nel caso in cui l'underscore non fosse presente?
3. Qual è lo scopo dell'istruzione `if` nella riga 24?
4. A cosa serve il blocco `if-else` nelle righe 26-30?
5. Riscrivete la condizione dell'`if` di riga 26 usando un metodo del package `unicode`.
6. A cosa serve l'assegnamento nella riga 31?
7. A cosa serve l'assegnamento nella riga 32?
8. Spiegate quale è il valore assunto da `char` in seguito all'assegnamento di riga 33.