# Laboratorio 5 - Tipi di dato

## 1 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 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 Trova l'errore

Questo programma dovrebbe stampare il valore delle variabili `a`, `b`, `c` e `d` ma contiene degli errori. Corregere 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 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 Operatori e precedenza

Si consideri il seguente frammento di codice.

```
var a, b, c int = 10, 5, 4
fmt.Println( a / b * c )
```

Quale valore viene stampato a video?

# 6 Divisione per zero

Si consideri il seguente frammento di codice.

```
var numero float64 = 0
fmt.Println(numero)

fmt.Println(0 / numero)

numero = 1 / numero
fmt.Println(numero)

numero = 1 / numero
fmt.Println(numero)

numero = -(1 / numero)
fmt.Println(numero)

numero = numero - numero
fmt.Println(numero)
```

Quali valori vengono stampati a video? Quali valori verrebbero stampati a video se la variabile `numero` fosse di tipo `int`?

# 7 Overflow

Qual è l'output del seguente programma?

```
import "fmt"

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

# 8 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)

    fmt.Println("complex64:", complex(-math.MaxFloat32, -math.MaxFloat32),
        complex(math.MaxFloat32, math.MaxFloat32))
    fmt.Println("complex128:", complex(-math.MaxFloat64, -math.MaxFloat64),
        complex(math.MaxFloat64, math.MaxFloat64))

}
```

# 9 Uguaglianza tra valori

Scrivere un programma che:
1) Legga da **standard input** un numero reale.
2) Calcoli la radice quadrata del numero letto (sia `x` la variabile di tipo `float64` in cui è memorizzato il valore; la radice quadrata di del valore reale può essere calcolata utilizzando la funzione `math.Sqrt` del package `math` nel seguente modo `radiceQuadrata := math.Sqrt(x)`).

3) Stampi a video `x, "uguale a", radiceQuadrata, "*", radiceQuadrata, "\n"` nel caso in cui `radiceQuadrata*radiceQuadrata` sia uguale a `x` e `x, "diverso da", radiceQuadrata, "*", radiceQuadrata, "\n"` altrimenti.

Supponendo che l'utente inserisca da **standard input** il valore `9`, qual è l'output del programma?

Supponendo che l'utente inserisca da **standard input** il valore `10`, qual è l'output del programma? Perché?

**Esempio d'esecuzione:**

```
$ go run uguaglianza.go
Inserisci un numero: 4
4 uguale a 2 * 2
```

# 10 Retta

Scrivere un programma che legga da **standard input** un valore intero e due valori reali:

- il primo valore è il seme (seed) `s` da utilizzare per inizializzare il generatore di numeri casuali;
- il secondo ed il terzo valore sono il coefficiente angolare `m` e il termine noto `q` di una retta `r: y = m*x + q` sul piano cartesiano.

Una volta terminata la fase di lettura, il programma deve generare per 10 volte una coppia di valori reali `px` e `py` che rappresentano rispettivamente l'ascissa e l'ordinata di un punto sul piano cartesiano e, per ciascun punto:

1. determinare se, a meno di una costante `EPSILON = 1.0e-9`, il punto sta sopra, sotto, o appartiene alla retta `r`;
2. stampare a video il relativo messaggio (come mostrato nell'**Esempio di esecuzione**).

I valori `px` e `py` devono essere compresi nell'intervallo [0, 10.0).

Oltre alla funzione `main()`, il programma deve utilizzare almeno due tra le seguenti funzioni:

```
const EPSILON = 1.0e-9

/* La funzione `func ÈXMaggioreDiY(x, y float64) bool` riceve in input due
valore reali nei parametri `x` e `y` e restituisce `true` se `x` è maggiore
di `y` a meno di una costante EPSILON */
/* ÈXMaggioreDiY(5.0,4.999) -> true */
/* ÈXMaggioreDiY(5.0,4.9999999999) -> false */
func ÈXMaggioreDiY(x, y float64) bool {
    return (x - y) > EPSILON
}

/* La funzione `func ÈXUgualeAY(x, y float64) bool` riceve in input due
valore reali nei parametri `x` e `y` e restituisce `true` se `x` è uguale
di `y` a meno di una costante EPSILON */
/* ÈXUgualeAY(5.0,4.999) -> false */
/* ÈXUgualeAY(5.0,4.9999999999) -> true */
func ÈXUgualeAY(x, y float64) bool {
    return math.Abs(x - y) <= EPSILON
}

/* La funzione `func ÈXMaggioreDiY(x, y float64) bool` riceve in input due
valore reali nei parametri `x` e `y` e restituisce `true` se `x` è minore
di `y` a meno di una costante EPSILON */
/* ÈXMinoreDiY(4.999,5.0) -> true */
/* ÈXMinoreDiY(4.9999999999,5.0) -> false */
func ÈXMinoreDiY(x, y float64) bool {
    return (x - y) < -EPSILON
}
```

*Nota:* Inizializzando il generatore dei numeri casuali con l'istruzione

```
/* inizializzazione del generatore di numeri casuali */
rand.Seed(s)
```

la sequenza dei numeri casuali generati durante l'esecuzione del programma dipenderà dal valore `s`, quindi due esecuzioni successive del programma considereranno la stessa sequenza di numeri casuali se l'utente inserirà lo stesso valore per `s`.

*Suggerimento:* per generare in modo casuale un valore reale, potete utilizzare la funzione `func Float64()` del package `math/rand`, che restituisce un valore casuale compreso tra [0.0,1.0).

**Esempio d'esecuzione:**

```
$ go run retta.go
Inserisci s: 2
Inserisci m e q: 1 0

Punto (0.8364831721292811,1.325271527168901) - Il punto sta sopra la retta
Punto (0.25744012651422055,0.5948870460174552) - Il punto sta sopra la retta
Punto (3.071353082885974,4.638274793965569) - Il punto sta sopra la retta
Punto (2.125002747927654,1.030771445518701) - Il punto sta sotto la retta
Punto (1.059760133310769,0.6344848690006494) - Il punto sta sotto la retta
Punto (3.09889044262683,4.311133779707338) - Il punto sta sopra la retta
Punto (1.843213441859697,1.9013752063486216) - Il punto sta sopra la retta
Punto (3.1086212867763456,4.552073840109743) - Il punto sta sopra la retta
Punto (2.151010640152745,0.06284952382418317) - Il punto sta sotto la retta
Punto (2.6797238953528506,4.4750914842551826) - Il punto sta sopra la retta

$ go run retta.go
Inserisci s: 2
Inserisci m e q: 1 1

Punto (0.8364831721292811,1.325271527168901) - Il punto sta sotto la retta
Punto (0.25744012651422055,0.5948870460174552) - Il punto sta sotto la retta
Punto (3.071353082885974,4.638274793965569) - Il punto sta sopra la retta
Punto (2.125002747927654,1.030771445518701) - Il punto sta sotto la retta
Punto (1.059760133310769,0.6344848690006494) - Il punto sta sotto la retta
Punto (3.09889044262683,4.311133779707338) - Il punto sta sopra la retta
Punto (1.843213441859697,1.9013752063486216) - Il punto sta sotto la retta
Punto (3.1086212867763456,4.552073840109743) - Il punto sta sopra la retta
Punto (2.151010640152745,0.06284952382418317) - Il punto sta sotto la retta
Punto (2.6797238953528506,4.4750914842551826) - Il punto sta sopra la retta
```

# 11 Troncamento

Scrivere un programma che legga da **standard input** un numero reale e ne stampi il valore **troncato** alla seconda cifra decimale.

*Suggerimento:* Il valore troncato alla seconda cifra decimale di un numero reale può essere ottenuto effettuando le seguenti operazioni:

1. Moltiplicare il numero reale per 100
2. Convertire il valore ottenuto al passo 1) in un valore di tipo `int`
3. Riconvertire il valore ottenuto al passo 2) in un valore di tipo `float64`
4. Dividere il valore ottenuto al passo 3) per 100

oppure:

1. Moltiplicare il numero reale per 100
2. Utilizzare la funzione `math.Trunc` del package `math` per troncare all'intero valore ottenuto al passo 1) (si utilizzi il comando `go doc math.Trunc` per capire come utilizzare la funzione)
3. Dividere il valore ottenuto al passo 2) per 100

**Esempio d'esecuzione:**

```
$ go run troncamento.go
Inserisci il valore da troncare: 10.34762
Valore troncato = 10.34

$ go run troncamento.go
Inserisci il valore da troncare: 8.34267
Valore troncato = 8.34
```

# 12 Arrotondamento

Scrivere un programma che legga da **standard input** un numero reale e ne stampi il valore **arrotondato** alla seconda cifra decimale.

*Suggerimento:* Il valore arrotondato alla seconda cifra decimale di un numero reale può essere ottenuto effettuando le seguenti operazioni:

1. Moltiplicare il numero reale per 100
2. Sommare 0.5 al valore ottenuto al passo 1)
3. Convertire il valore ottenuto al passo 2) in un valore di tipo `int`
4. Riconvertire il valore ottenuto al passo 3) in un valore di tipo `float64`
5. Dividere il valore ottenuto al passo 4) per 100

oppure:

1. Moltiplicare il numero reale per 100
2. Utilizzare la funzione `math.Round` del package `math` per arrontondare all'intero valore ottenuto al passo 1) (si utilizzi il comando `go doc math.Round` per capire come utilizzare la funzione)
3. Dividere il valore ottenuto al passo 2) per 100

**Esempio d'esecuzione:**

```
$ go run arrotondamento.go
Inserisci il valore da arrotondare: 10.34762
Valore arrotondato = 10.35

$ go run arrotondamento.go
Inserisci il valore da arrotondare: 8.32467
Valore arrotondato = 8.32
```

# 13 Troncamento/Arrotondamento generalizzati

Scrivere una versione generalizzata dei programmi **Troncamento** e **Arrotondamento** che legga da **standard input** un intero `n` oltre al numero reale. L'intero `n` specifica che il troncamento e l'arrotondamento devono avvenire alla `n`-esima cifra decimale.

**Esempio d'esecuzione:**

```
$ go run generalizzazione.go
Inserisci il valore: 10.34762
Inserisci il numero di cifre dopo la virgola: 4
Valore troncato = 10.3476
Valore arrotondato = 10.3476

$ go run generalizzazione.go
Inserisci il valore:  10.34762
Inserisci il numero di cifre dopo la virgola: 3
Valore troncato = 10.347
Valore arrotondato = 10.348
```

# 14 Classifica triangolo

Scrivere un programma che legga da **standard input** sei valori reali:

- il primo ed il secondo valore, `xA` e `yA`, rappresentano rispettivamente l'ascissa e l'ordinata di un punto `A` sul piano cartesiano;
- il terzo ed il quarto valore, `xB` e `yB`, rappresentano rispettivamente l'ascissa e l'ordinata di un punto `B` sul piano cartesiano;
- il quinto ed il sesto valore, `xC` e `yC`, rappresentano rispettivamente l'ascissa e l'ordinata di un punto `C` sul piano cartesiano.

Una volta terminata la fase di lettura, il programma deve stampare a video (come mostrato nell'**Esempio di esecuzione**), se il triangolo `ABC` definito dai segmenti/lati `AB`, `BC` e `AC` è equilatero, iscoscele o scaleno.

Un triangolo è equilatero se ha tutti e tre i lati di lunghezza uguale.

Un triangolo è isoscele se ha solo due lati di lunghezza uguale.

Un triangolo è scaleno se ha tutti e tre i lati di lunghezza diversa.

La lunghezza di ciascun lato di un triangolo è pari alla distanza euclidea tra gli estremi del lato.

Per esempio, la lunghezza del lato `AB` del triangolo `ABC` è pari alla distanza euclidea tra i punti `A` e `B`: $((xA\text{-}xB)^2 + (yA\text{-}yB)^2)^{1/2}$.

Le lunghezze dei lati del triangolo vanno confrontate a meno di una costante `EPSILON = 1.0e-9`.

Oltre alla funzione `main()`, devono essere definite ed utilizzate almeno le seguenti funzioni:

- una funzione `Distanza(x1, y1, x2, y2 float64) float64` che riceve in input:
  - due valori `float64` nei parametri `x1` e `y1` che rappresentano rispettivamente l'ascissa e l'ordinata di un punto `P1` sul piano cartesiano;
  - due valori `float64` nei parametri `x2` e `y2` che rappresentano rispettivamente l'ascissa e l'ordinata di un punto `P2` sul piano cartesiano;

  e restituisce un valore `float64` pari alla distanza euclidea tra i punti `P1` e `P2`.

**Esempio d'esecuzione:**

```
$ go run classifica.go
Inserisci i valori dell'ascissa e dell'ordinata del punto A: -6 1
Inserisci i valori dell'ascissa e dell'ordinata del punto B: -1 1
Inserisci i valori dell'ascissa e dell'ordinata del punto C: -3.5 5.330127018922193

Il triangolo ABC è equilatero.
Lunghezza del lato: 5

$ go run classifica.go
Inserisci i valori dell'ascissa e dell'ordinata del punto A: -6 1
Inserisci i valori dell'ascissa e dell'ordinata del punto B: -1 1
Inserisci i valori dell'ascissa e dell'ordinata del punto C: -3.5 5.33

Il triangolo ABC è isocele.
I lati di lunghezza uguale sono AC e BC.
Lunghezza dei lati AC e BC: 4.999889998789973
Lunghezza del lato AB: 5

$ go run classifica.go
Inserisci i valori dell'ascissa e dell'ordinata del punto A: 1 1
Inserisci i valori dell'ascissa e dell'ordinata del punto B: 3 3
Inserisci i valori dell'ascissa e dell'ordinata del punto C: 1 5

Il triangolo ABC è isocele.
I lati di lunghezza uguale sono AB e BC.
```

```
Lunghezza dei lati AB e BC: 2.8284271247461903
Lunghezza del lato AC: 4

$ go run classifica.go
Inserisci i valori dell'ascissa e dell'ordinata del punto A: 1 1
Inserisci i valori dell'ascissa e dell'ordinata del punto B: 1 5
Inserisci i valori dell'ascissa e dell'ordinata del punto C: 5 1

Il triangolo ABC è isocele.
I lati di lunghezza uguale sono AB e AC.
Lunghezza dei lati AB e AC: 4
Lunghezza del lato BC: 5.656854249492381

$ go run classifica.go
Inserisci i valori dell'ascissa e dell'ordinata del punto A: 1 1
Inserisci i valori dell'ascissa e dell'ordinata del punto B: 1 5
Inserisci i valori dell'ascissa e dell'ordinata del punto C: 3 3

Il triangolo ABC è isocele.
I lati di lunghezza uguale sono AC e BC.
Lunghezza dei lati AC e BC: 2.8284271247461903
Lunghezza del lato AB: 4

$ go run classifica.go
Inserisci i valori dell'ascissa e dell'ordinata del punto A: 1 1
Inserisci i valori dell'ascissa e dell'ordinata del punto B: 1 4
Inserisci i valori dell'ascissa e dell'ordinata del punto C: 1.5 1.5

Il triangolo ABC è scaleno.
Lunghezza del lato AB: 3
Lunghezza del lato BC: 2.5495097567963922
Lunghezza del lato AC: 0.7071067811865476
```

# 15 Triangolo

Scrivere un programma che legga da **standard input** un valore intero e sei valori reali:

- il primo valore è il seme (seed) `s` da utilizzare per inizializzare il generatore di numeri casuali;
- il secondo ed il terzo valore sono il coefficiente angolare `m1` e il termine noto `q1` di una retta `r1: y = m1*x + q1` sul piano cartesiano su cui giace la base `AB` di un triangolo `ABC`;
- il quarto ed il quinto valore sono il coefficiente angolare `m2` e il termine noto `q2` di una retta `r2: y = m2*x + q2` sul piano cartesiano su cui giace il lato `BC` di un triangolo `ABC`;
- il sesto ed il settimo valore sono il coefficiente angolare `m3` e il termine noto `q3` di una retta `r3: y = m3*x + q3` sul piano cartesiano su cui giace il lato `AC` di un triangolo `ABC`.

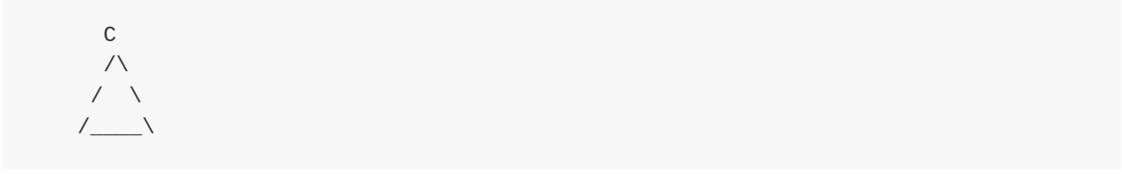
Una volta terminata la fase di lettura, il programma deve generare per 10 volte una coppia di valori reali `px` e `py` che rappresentano rispettivamente l'ascissa e l'ordinata di un punto sul piano cartesiano e, per ciascun punto:

1. determinare se, a meno di una costante `EPSILON = 1.0e-9`, il punto sta all'interno o all'esterno del triangolo `ABC`;
2. stampare a video il relativo messaggio (come mostrato nell'**Esempio di esecuzione**).

I valori `px` e `py` devono essere compresi nell'intervallo [0, 10.0).

Si assuma che:

- i valori introdotti dall'utente che definiscono le rette `r1`, `r2`, ed `r3` permettano di caratterizzare un triangolo `ABC` in cui la base `AB` sia parallela all'asse delle ascisse, ed il vertice `C`, opposto alla base, sia al di sopra della retta `r1` su cui giace la base del triangolo:



```
    A       B
```

- i valori introdotti dall'utente che definiscono le rette `r1`, `r2`, ed `r3` permettano di definire un triangolo `ABC` che contenga almeno un punto con coordinare `px` e `py` compresi nell'intervallo [0, 10.0).

*Suggerimento:* Un punto non appartiene al triangolo se sta al di sotto della retta `r1`, al di sopra della retta `r2`, o al di sopra della retta `r3`.

**Esempio d'esecuzione:**

```
$ go run triangolo.go
Inserisci s: 3
Inserisci m1 e q1: 0 2
Inserisci m2 e q2: -1 15
Inserisci m3 e q3: 1 0

Punto (7.199826688373036,6.526308027999122) - Il punto sta all'interno del triangolo.
Punto (9.419605585830052,7.681370946252233) - Il punto sta all'esterno del triangolo.
Punto (8.935331264200833,2.190716471450932) - Il punto sta all'interno del triangolo.
Punto (4.27633484874847,5.076860623844484) - Il punto sta all'esterno del triangolo.
Punto (3.250879908255019,4.684369767626347) - Il punto sta all'esterno del triangolo.
Punto (3.503080765299786,7.956635723501489) - Il punto sta all'esterno del triangolo.
Punto (8.170357818404945,3.493485636935447) - Il punto sta all'interno del triangolo.
Punto (9.603601912420224,4.76799474008711) - Il punto sta all'interno del triangolo.
Punto (4.994476561730858,7.24266412974525) - Il punto sta all'esterno del triangolo.
Punto (4.009944732483285,4.053135323201041) - Il punto sta all'esterno del triangolo.

$ go run triangolo.go
Inserisci s: 5
Inserisci m1 e q1: 0 1
Inserisci m2 e q2: -1 10
Inserisci m3 e q3: 1 0

Punto (8.038448294975284,5.196192472270505) - Il punto sta all'esterno del triangolo.
Punto (9.79385566880146,5.967488889192919) - Il punto sta all'esterno del triangolo.
Punto (4.793723986112438,7.002671557670805) - Il punto sta all'esterno del triangolo.
Punto (4.689419113522166,8.5793442542724) - Il punto sta all'esterno del triangolo.
Punto (2.9902926479032783,9.943523044874215) - Il punto sta all'esterno del triangolo.
Punto (0.01626037515838685,6.51527635860278) - Il punto sta all'esterno del triangolo.
Punto (5.443642101331307,1.303299350890999) - Il punto sta all'interno del triangolo.
Punto (4.965806911581438,2.9708160500789433) - Il punto sta all'interno del triangolo.
Punto (6.277694594891373,4.120684542231575) - Il punto sta all'esterno del triangolo.
Punto (1.2876742934360954,7.403669183503507) - Il punto sta all'esterno del triangolo.
```

# 16 Cerchio

Scrivere un programma che legga da **standard input** un valore intero e tre valori reali:

- il primo valore è il seme (seed) `s` da utilizzare per inizializzare il generatore di numeri casuali;
- il secondo ed il terzo valore, `xC` e `yC`, rappresentano rispettivamente l'ascissa e l'ordinata di un punto `C` sul piano cartesiano: il centro di un cerchio;
- il quarto valore, `raggio`, è il raggio del cerchio con centro `C`.

Una volta terminata la fase di lettura, il programma deve generare per `1.000.000` di volte una coppia di valori reali `px` e `py` che rappresentano rispettivamente l'ascissa e l'ordinata di un punto sul piano cartesiano e, come mostrato nell'**Esempio di esecuzione**:

1. per ogni punto che, a meno di una costante `EPSILON = 1.0e-6`, giace sulla circoferenza del cerchio con centro `C`, deve stampare a video il relativo messaggio;

2. deve stampare l'ascissa e l'ordinata del punto che, a meno di una costante `EPSILON = 1.0e-6`:

   a) è all'esterno del cerchio;

   b) ha distanza massima dal centro `C`;

3. deve stampare l'ascissa e l'ordinata del punto che, a meno di una costante `EPSILON = 1.0e-6`:

   a) è all'interno al cerchio;

   b) ha distanza minima dal centro `C`;

I valori `px` e `py` devono essere compresi rispettivamente negli intervalli `[xC-raggio, xC+raggio)` e `[yC-raggio, yC+raggio)`.

**Esempio d'esecuzione:**

```
$ go run cerchio.go
Inserisci s: 1
Inserisci i valori dell'ascissa e dell'ordinata del punto C (il centro del cerchio): 5 5
Inserisci il valore del raggio: 1

Il punto (4.15954073279876,5.541874673524392) giace sulla circonferenza del cerchio.

Il punto (4.0003279909645535,4.001517285035352) è quello all'esterno del cerchio che ha d
Distanza: 1.4129090054678466

Il punto (4.999592781833336,5.0000449681220855) è quello all'interno del cerchio che ha d
Distanza: 0.00040969350405560887

$ go run cerchio.go
Inserisci s: 3
Inserisci i valori dell'ascissa e dell'ordinata del punto C (il centro del cerchio): 4 4
Inserisci il valore del raggio: 0.5

Il punto (4.497366185594927,3.948755997236377) giace sulla circonferenza del cerchio.
Il punto (4.253280394019776,3.568897687168935) giace sulla circonferenza del cerchio.
Il punto (3.9590078930698014,3.501683012145147) giace sulla circonferenza del cerchio.
Il punto (3.8031515161989375,4.4596205489968055) giace sulla circonferenza del cerchio.

Il punto (4.499978695636602,4.4994651760442474) è quello all'esterno del cerchio che ha d
Distanza: 0.7067136323656061

Il punto (4.0002120210822625,3.9996262973943617) è quello all'interno del cerchio che ha
Distanza: 0.0004296586747461331
```