Coding challenge

Bisakah membuat **Lingkaran** pada ***Terminal*** atau ***Console*** dengan memanfaatkan ***Trigonometry*** :)

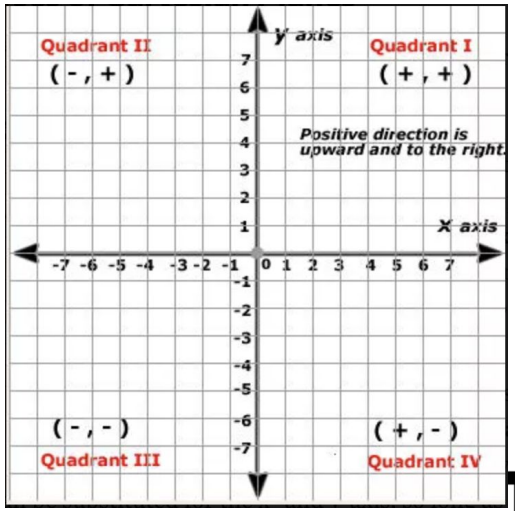
wuriyan.to

Cartesian Plane

Pada dasarnya semua hal yang bisa merepresentasikan **Cartesian Plane** bisa kita gambar sebuah **Lingkaran**.

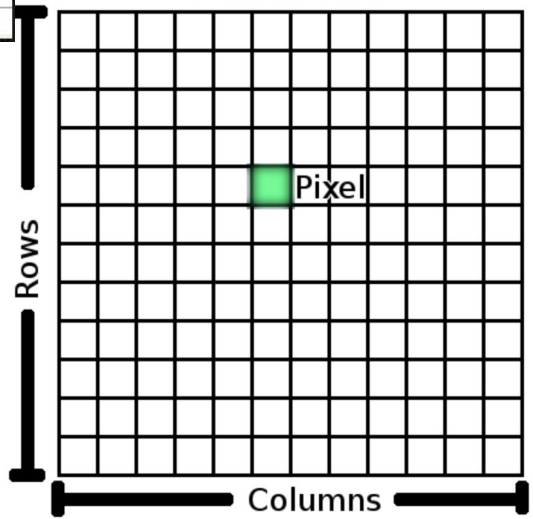
Contoh:

- Salah satunya tentu saja **raster image (contoh: PNG, JPG)**. Apa itu raster image? raster image ialah citra digital yang tersusun dari banyak **Pixel**. Susunan Pixel tersebut direpresentasikan dalam matrix 2 dimensi dengan koordinat (**X: 0, Y: 0**) berada pada pojok kiri atas. Sama dengan koordinat pada layar komputer kita.
- HTML5 Canvas
- Array 2 Dimensi. Sama halnya dengan raster image, array 2 dimensi juga bisa merepresentasikan cartesian plane. Sehingga bisa kita gambar sebuah lingkaran.



Cartesian Plane

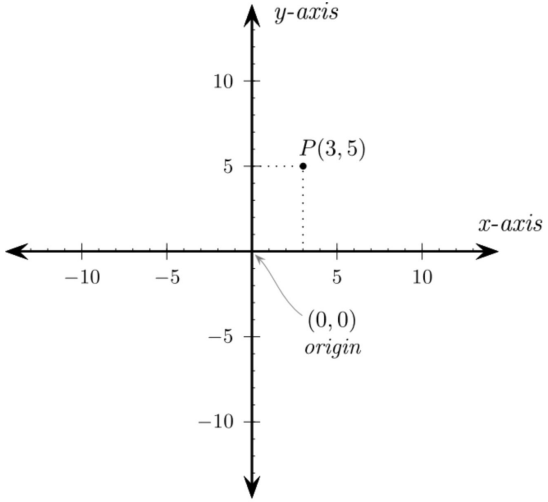
Pixel (Rows sebagai Y, Column sebagai X)



Computer coordinate system

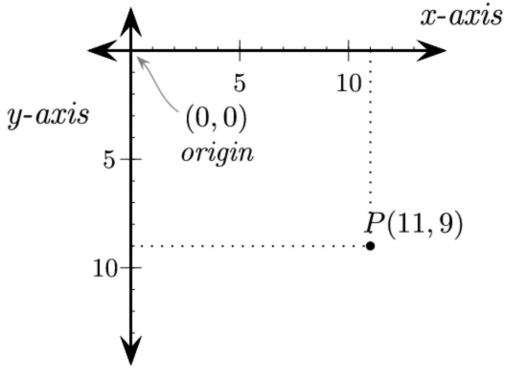
Jika anda datang dari ilmu Matematika murni, ketika melihat sistem koordinat pada komputer akan sedikit membingungkan. Karena pada sistem koordinat komputer tidak terdapat istilah **kuadran**. Jadi semua poin akan bernilai positif dan dalam satu kuadran.

Pada gambar disamping, sistem koordinat komputer dimulai dari pojok kiri atas, dengan **X dimulai dari kiri ke kanan** dan **Y dari atas ke bawah**. Nilai **X** semakin ke kanan semakin bertambah dan sebaliknya. Nilai **Y** semakin ke bawah semakin bertambah dan sebaliknya.



Sistem koordinat normal

Sistem koordinat komputer

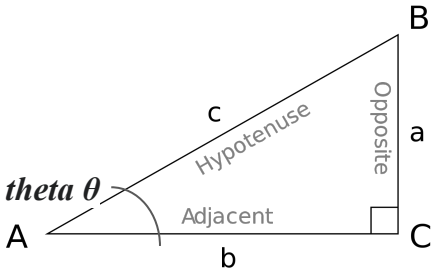


Trigonometry

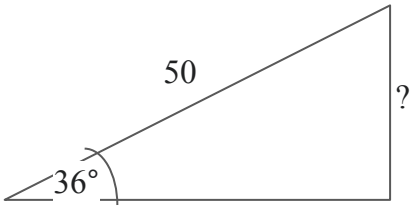
Pada pelajaran Matematika SMP kita sudah diperkenalkan dengan 3 dari banyak fungsi Trigonometri yang paling sering kita manfaatkan. **Sin**, **Cos** dan **Tan**. Termasuk beberapa fungsi *inverse* atau kebalikannya. Seperti **Arcsin**, **Arcos** dan **Arctan**.

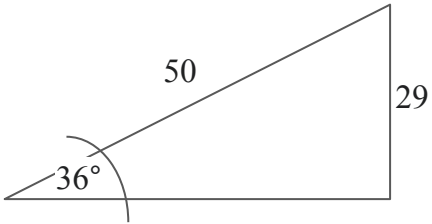
Trigonometry adalah salah satu cabang ilmu Matematika yang sangat penting bagi banyak industri. Sebut saja penerbangan, kelautan, otomasi mesin. Dan tentu saja sangat penting pada bidang ilmu komputer, seperti pada pengolahan citra digital, *Computer vision*, *Game engine*.

Kita tidak akan membahas detail tentang Trigonometri, hanya akan kita bahas sedikit beberapa persamaan yang akan kita gunakan sesuai tema materi ini.



$\sin\theta = a/c$
$\cos\theta = b/c$
$\tan\theta = a/b$





Kita akan coba buktikan beberapa persamaan diatas. Pada gambar disamping diketahui sudut kemiringan segitiga tersebut **θ = 36°**, nilai **hypotenuse atau c = 50** dan nilai yang ingin kita ketahui adalah **opposite atau a**. Sehingga kita bisa menggunakan $\sin(\theta) = a/c$.

sin(θ) = a/c

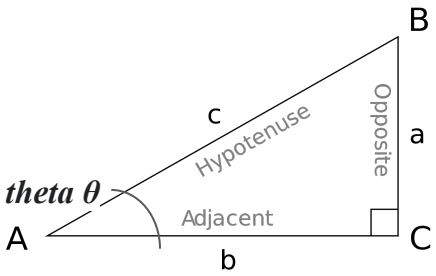
sin(36°) = a/50
a = sin(36°) . 50

sin(36°) = 0.58

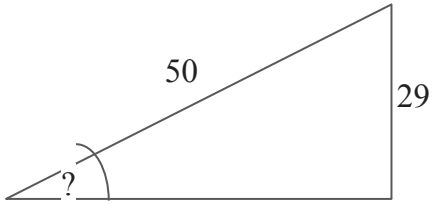
a = 0.58 x 50

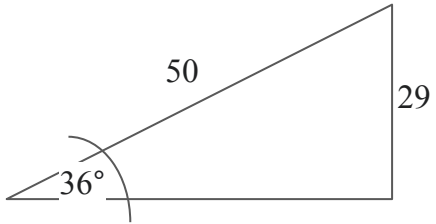
a = 29

Trigonometry (inverse function)



$sin\ \theta = a/c$
$cos\ \theta = b/c$
$tan\ \theta = a/b$





Sama halnya, misal ketika kita ingin mengetahui berapa **sudut kemiringan,** kita bisa memanfaatkan fungsi ***inverse*** trigonometri. Pada gambar disamping diketahui nilai **hypotenuse atau c = 50, opposite atau a = 29** dan **nilai yang ingin kita ketahui adalah sudut kemiringannya**. Kita masih menggunakan persamaan $sin\ \theta = a/c$.

sin(θ) = a/c

sin(θ) = 29/50

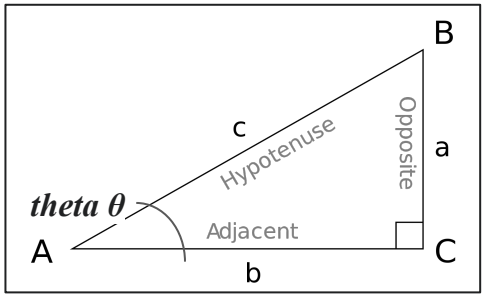
29/50 = 0.58

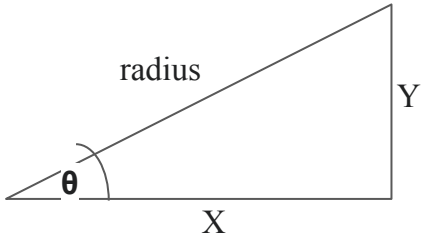
sin(θ) = 0.58

θ = arcsin(0.58)

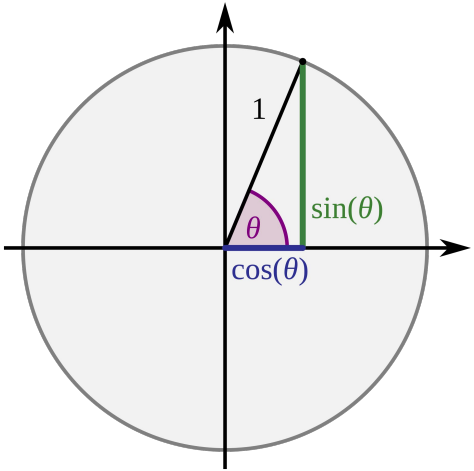
θ = 36°

X , Y value ?





$\sin\theta = a/c$
$\cos\theta = b/c$
$\tan\theta = a/b$



Dengan memanfaatkan persamaan-persamaan trigonometri yang sudah kita bahas sedikit diatas, kita bisa mendapatkan setiap nilai **X** dan **Y** dengan **hanya kita mengetahui radius dan sudut kemiringannya**, sehingga kita bisa menentukan pada koordinat mana akan kita akan meletakan sesuatu.

mendapatkan nilai X:
cos(θ) = X/radius

X = cos(θ) . radius

mendapatkan nilai Y:
sin(θ) = Y/radius

Y = sin(θ) . radius

Nilai theta (θ) adalah hasil kalkulasi dari 0 - 360. Yaitu nilai dari semua kemungkinan sudut.

Dengan mengulangi perhitungan **X** dan **Y dengan for loop misalnya, kita bisa mendapatkan setiap koordinat dan membentuk lingkaran penuh seperti gambar di samping.**

Array, List, dan Vector

Array adalah salah satu struktur data yang sangat sering kita gunakan, terutama untuk menyimpan lebih dari satu data dengan tipe data yang sama dalam satu tempat.

Sebenarnya kita bisa menggunakan struktur data lain seperti **List** atau **Vector**. Selama tipe data tersebut bisa menyimpan data lebih dari satu dimensi.

Dengan memanfaatkan **Array**, **List** atau **Vector 2 dimensi** kita bisa menyimpan koordinat data, sehingga kita bisa merepresentasikan **Cartesian Plane**.

Pada ada 2 dimensi disamping kita membuat array dengan ukuran **[31 x 31]** dan setiap *index*-nya yang dimulai dari 0 merepresentasikan koordinat. ***Rows*** merepresentasikan sumbu **Y** dan ***Column*** merepresentasikan sumbu **X**. Koordinat **center = round(31/2) = 15**.

Data Array [31x31]:

```
[
  0 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
  1 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
  2 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
  3 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
  4 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
  5 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
  6 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
  7 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
  8 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
  9 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 10 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 11 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 12 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 13 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 14 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 15 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 16 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 17 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 18 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 19 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 20 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 21 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 22 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 23 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 24 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 25 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 26 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 27 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 28 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 29 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
 30 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 29, 30]
]
```

Implementation (pseudocode 1)

```
radius = 10

n = 31

circleDot = “*”
planeDot = “#”

cartesian string[][]

for y = 0; y <= n; y++:
    cartI string[]
    for x = 0; x <= n; x++:
      cartI[y][x] = planeDot
    cartesian.push(cartI)
```

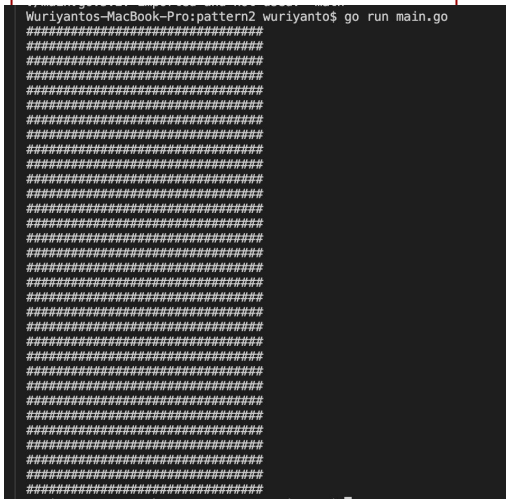
- **radius** lingkaran kita set dengan value 10, lebih kecil dari 31, yaitu ukuran board.

- ukuran board = 31, sehingga kita memiliki matrix [31x31]

- **circleDot *** nantinya akan kita gunakan untuk membentuk lingkaran
- **planeDot #** nantinya akan kita gunakan untuk mengisi board

- **cartesian**, variabel array dua dimensi *string* yang akan kita isi dengan tanda **#**

- nested loop kita gunakan untuk mengisi cartesian board kita dengan tanda **#**
- **cartI**, scope variabel pada loop terluar, kita gunakan sebagai value dari sumbu X



Dari pseudocode diatas bisa kita dapatkan hasil seperti di samping.

Implementation (pseudocode 2)

```
int radius = 10

int n = 31

circleDot = “*”
planeDot = “#”

cartesian string[][]

for y = 0; y <= n; y++:
    cartI string[]
    for x = 0; x <= n; x++:
      cartI[y][x] = planeDot
    cartesian.push(cartI)

float xx = 0
float yy = 0

for t = 0; t <= 360; t++:
   xx = round(radius*cos(t))
   yy = round(radius*sin(t))

   cartesian[abs(yy), abs(xx)] = circleDot
```

- radius lingkaran kita set dengan value 10, lebih kecil dari 31, ukuran board.

- ukuran board = 31, sehingga kita memiliki matrix [31x31]

- circleDot ***** nantinya akan kita gunakan untuk membentuk lingkaran
- planeDot **#** nantinya akan kita gunakan untuk mengisi board

- **cartesian**, array dua dimensi string yang akan kita isi dengan tanda **#**

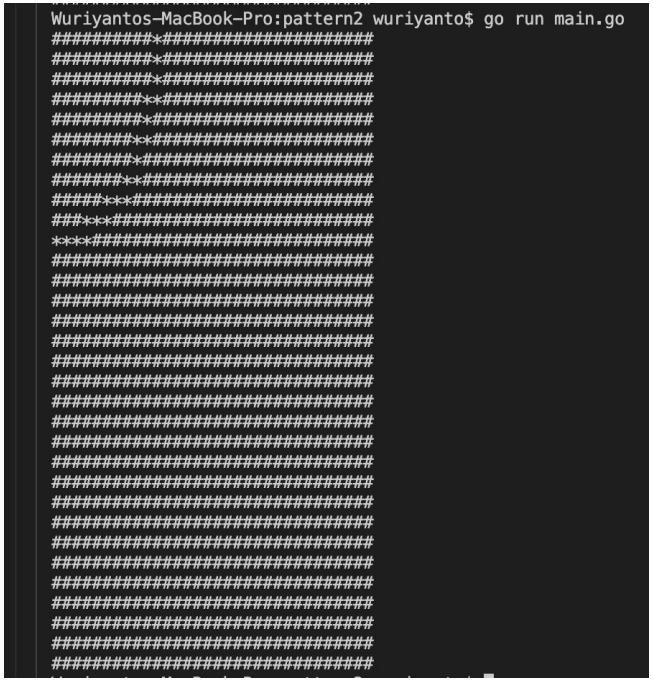
- nested loop kita gunakan untuk mengisi cartesian board kita dengan tanda **#**
- **cartI**, scope variabel pada loop terluar, kita gunakan sebagai value dari sumbu X

- **xx** akan kita gunakan untuk menyimpan setiap koordinat **x** hasil dari kalkulasi **X = cos θ . radius**
- **yy** akan kita gunakan untuk menyimpan setiap koordinat **y** hasil dari kalkulasi **Y = sin θ . radius**

- loop kita gunakan untuk kalkulasi setiap nilai theta **θ**
- set value xx, kita round, karena kita hanya membutuhkan nilai bulat
- set value yy kita round, karena kita hanya membutuhkan nilai bulat

- kita replace tanda **#** di setiap koordinat yang sesuai dengan tanda *****. Sebelumnya value **xx** dan **yy** kita set ke nilai *absolute*, karena kita hanya membutuhkan nilai positif,

Hasil dari (pseudocode 2)



Dari hasil pseudocode ke 2 kita hanya bisa menampilkan ¼ dari lingkaran kita.

Hal tersebut terjadi karena koordinat *center* dari lingkaran yang kita buat dimulai dari **0, 0**.

Akan kita perbaiki pada pseudocode ke 3.

Implementation (pseudocode 3)

```
int radius = 10

int n = 31

circleDot = “*”
planeDot = “#”

cartesian string[][]

for y = 0; y <= n; y++:
    cartI string[]
    for x = 0; x <= n; x++:
      cartI[y][x] = planeDot
    cartesian.push(cartI)

float xx = 0
float yy = 0

for t = 0; t <= 360; t++:
   xx = n/2 + round(radius*cos(t))
   yy = n/2 + round(radius*sin(t))

   cartesian[abs(yy), abs(xx)] = circleDot
```

- radius lingkaran kita set dengan value 10, lebih kecil dari 31, ukuran board.

- ukuran board = 31, sehingga kita memiliki matrix [31x31]

- **circleDot *** nantinya akan kita gunakan untuk membentuk lingkaran
- **planeDot #** nantinya akan kita gunakan untuk mengisi board

- cartesian, array dua dimensi string yang akan kita isi dengan tanda **#**

- nested loop kita gunakan untuk mengisi cartesian board kita dengan tanda **#**
- **cartI**, scope variabel pada loop terluar, kita gunakan sebagai value dari sumbu X

- **xx** akan kita gunakan untuk menyimpan setiap koordinat **x** hasil dari kalkulasi **X = cos θ . radius**
- **yy** akan kita gunakan untuk menyimpan setiap koordinat **y** hasil dari kalkulasi **Y = sin θ . radius**

- loop kita gunakan untuk kalkulasi setiap nilai theta **θ**
- set value xx, kita round, karena kita hanya membutuhkan nilai bulat. Dan untuk fixing problem sebelumnya kita tambahkan nilai tengah dari board yang kita buat 30/2 = 15 dengan setiap nilai xx. Sehingga posisi lingkaran akan berada ditengah.
- set value yy kita round, karena kita hanya membutuhkan nilai bulat. Dan untuk fixing problem sebelumnya kita tambahkan nilai tengah dari board yang kita buat 30/2 = 15 dengan setiap nilai yy. Sehingga posisi lingkaran akan berada ditengah.

- kita replace tanda **#** di setiap koordinat yang sesuai dengan tanda *. Sebelumnya value **xx** dan **yy** kita set ke nilai *absolute*, karena kita hanya membutuhkan nilai positif,

Note: Text merah menandakan perbaikan dari pseudocode sebelumnya

Hasil dari (pseudocode 3)

Seperti inilah hasil akhirnya. Lingkaran tidak bulat sempurna karena ukuran board dan grid kita tidak presisi seperti pada *raster image* misalnya.

Real implementation with Javascript and HTML5 Canvas

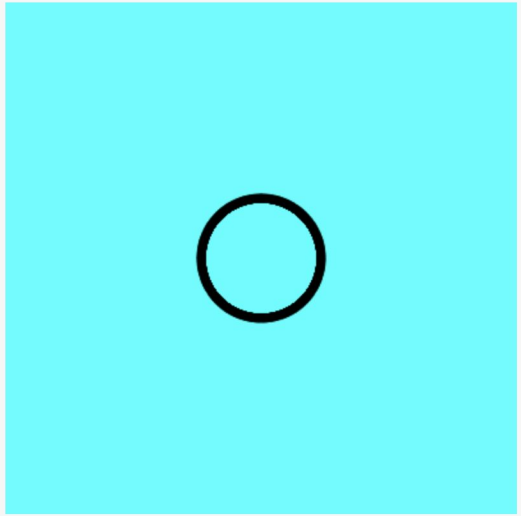
Seperti inilah hasil implementasi dengan menggunakan HTML5 canvas dan Javascript. Hasilnya bulat sempurna.

JavaScript

```
var theCanvas = document.getElementById('theCanvas');
var ctx = theCanvas.getContext('2d');
ctx.fillStyle = 'cyan';
ctx.fillRect(0, 0, 400, 400);

var r = 50;

for (var t = 0; t <= 360; t++) {
  var x = 400/2 + (r * Math.cos(t));
  var y = 400/2 + (r * Math.sin(t));
  console.log(x)

  ctx.lineTo(x, y);


}

ctx.stroke();
```

Output

Run with JS



Real implementation with Golang

Untuk implementasi dengan Go, anda bisa melihatnya full disini

https://gist.github.com/wuriyanto48/ff4318a955fbe7e7ea28414181c0133e

Selamat Mencoba …