Presentación Estrategia didáctica y proyecto

Elaborado por Plata Luna Iveth Vanessa.

2023

Estrategia didáctica

Nombre del participante	Plata Luna Iveth Vanessa
Asignatura	Cibernética y Computación 2
Año o semestre en que imparte	6to. semestre
Horas clase a la semana	4 horas
Unidad	2
Aprendizajes	Arreglos Resuelve problemas que involucren el uso de los arreglos unidimensionales en los métodos de una Clase.

Problemática que se abordará a través del problema.	Una de las grandes problemáticas en la enseñanza aprendizaje es incentivar a los alumnos. Algunas estrategias que podemos usar es el trabajo a través de proyectos; pero para esto, el docente tiene que ser ejemplo, esto quiere decir que el docente tiene que involucrarse en proyectos o diseñar estos. Al presentarle al alumno un proyecto desarrollado por el profesor pudiera generar cierta curiosidad al alumno por saber más. Por tal motivo, una de las estrategias utilizadas es mostrar una aplicación práctica en donde se utilicen los aprendizajes a ver en la sesión, con el fin de que el en el alumno o la alumna se le despierte la motivación por conocer más. Se ejemplificará el uso estructuras de datos en una aplicación en diferentes lenguajes de programación (arreglos unidimensionales en Java y de listas en Python) a través de un proyecto titulado Radar. El presente proyecto es un radar, rastrea objetos a través de un sensor ultrasónico en un rango de 360 grados y 20 centímetros. Los datos obtenidos se envían por medio del puerto serial y son codificados y almacenados en listas para que posteriormente la computadora represente de manera gráfica y en formato texto la ubicación en coordenadas polares y cartesianas los objetos con respecto al sensor ultrasónico. La computadora también calculará las distancias entre los objetos rastreados.

Justificación. (porque considera que el programa en python o Julia puede apoyar al alumno a entender o lograr el aprendizaje)	Las estructuras de datos como los arreglos y listas se utilizan con frecuencia para optimizar la manipulación y almacenamiento de datos en todos los lenguajes de programación. En la enseñanza aprendizaje es un tema complejo, porque involucra conocimientos previos como el concepto de variables, tipos de datos, estructura de repetición for, variables de acumulación e incremento, en algunos casos los operadores unarios. Sin embargo, el programa de Python puede ayudar a ejemplificar de una manera mucho más fácil las listas y dar un paso en la comprensión de los índices dentro de los arreglos.

Producto esperado **(Después de haber explicado, haber realizado alguna actividad guiada y/o dejar una actividad extraclase, ¿Qué evidencia tiene que entregar para ser evaluada?**	**Los alumnos y las alumnas** entregarán una reflexión sobre los usos que se les puede dar a los arreglos, ventajas y desventajas entre Python y Java con respecto a los arreglos y listas. También **los alumnos y las alumnas** realizarán un ejercicio en donde se utilice un arreglo unidimensional. Por último, **los alumnos y las alumnas** entregarán el planteamiento e implementación de un programa en donde se utilice por lo menos un arreglo unidimensional.
Recursos materiales /Herramientas TIC	• Computadora o laptop, • Software: IDE (Entorno de desarrollo integrado) www.replit.com. • Conexión a internet, • Plataforma educativa: Teams • Videoproyector. • Pizarrón, • Plumo gis • Video • Lista de cotejo para evaluar el desarrollo del proyecto, a manera de cronograma, para apoyar en el seguimiento del mismo.
Tiempos de realización.	Dos horas en clase y dos extra-clase.

Presentación del problema a resolver	Los alumnos y alumnas plantearán e implementarán en Java una solución a una problemática en donde involucre arreglos, con el apoyo de algunos ejemplos que muestre el docente tanto en Python como en Java.
Inicio de la Sesión	El **docente junto con los alumnos** realizará un repaso de los conocimientos previos a través de lluvia de ideas. **Anexo 1 Presentación conocimientos previos.** El **docente** mostrará el mismo proyecto implementado en diferentes lenguajes de programación (Java y Python). **Anexo 2 Presentación proyecto implementado en Pytho y Java.** El **docente** realizará algunas comparaciones entre ambos lenguajes con respecto a los arreglos (en Java) y listas (en Python). **Anexo 2 Presentación proyecto implementado en Pytho y Java.** El **docente** realizará una explicación sobre los fundamentos de los arreglos y las listas. **Anexo 3 Presentación arreglos unidimensionales.** **Las alumnas y los alumnos de manera individual** reflexionarán sobre las aplicaciones que tienen los arreglos y las listas. **Anexo 4 Preguntas para la reflexión** **Las alumnas y los alumnos de manera individual** identificaran las diferencias entre ambos lenguajes de programación (Python y Java) con respecto a los arreglos y listas. **Anexo 4 Preguntas para la reflexión.** **Las alumnas y los alumnos en equipo** plantearán algunas propuestas a desarrollar. **Anexo 4 Preguntas para la reflexión.** **Preguntas para la reflexión** ¿Cuáles son las principales ventajas y desventajas de Python y Java? Comenta algunas diferencias entre Python y Java con respecto a los arreglos y listas. Lista algunas propuestas en donde se puede utilizar los arreglos y listas.

Desarrollo de la sesión	**Las alumnas y los alumnos de manera individual** realizarán un ejercicio en relación con el tema de arreglos. **Anexo 3 Presentación arreglos unidimensionales**. En una papelería se necesita determinar el subtotal, IVA y total de diez productos. • Realiza un programa en Java donde se solicite el monto (similar a la estructura calificaciones[], nada más que esta estructura se llama montos[]) de cada uno de los productos y los almacena en un arreglo de diez elementos double montos[10]). • En el mismo bucle se acumulará poco a poco el subtotal (similar a la variable suma, nada más que esta se llama subtotal) por cada producto que pase por el lector de código de barras en una variable que lleva su mismo nombre. • Al finalizar el bucle el usuario imprimirá el subtotal, calculará el IVA multiplicando el subtotal por el 0.16 (afuera del bucle se realiza la siguiente multiplicación subtotal * 0.16, el resultado del producto es el IVA) y se sumará el subtotal más el IVA, teniendo como resultado el total (total = subtotal + IVA). • Es importante desplegar en pantalla el monto de cada producto (monto[i]), el subtotal, IVA y total). **Las alumnas y los alumnos en equipo** implementarán su propuesta en el lenguaje Java.
Cierre de la sesión	**Las alumnas y los alumnos en equipo** exponen sus propuestas implementadas en lenguaje Java.

Evaluación	A. Lista de cotejo para la reflexión. B. Lista de cotejo para el ejercicio: “Papelería”. C. Lista de cotejo para la propuesta e implementación

A. Lista de cotejo para la reflexión

	Sí	No
La alumna y el alumno encuentra aplicaciones en las estructuras: listas y arreglos.		
La alumna y el alumno identifica las ventajas, desventajas del mismo proyecto implementado en diferentes lenguajes de programación (Python y Java).		
Las alumnas y los alumnos plantean una propuesta a implementar.		

B. Lista de cotejo para el ejercicio: “Papelería”

La alumna y el alumno:	Sí	No
Utiliza arreglos en la implementación de su programa.		
presentan sus dudas o investigan para solucionar estas.		
compila a bytecode el ejercicio.		

C. Lista de cotejo para la propuesta e implementación

La alumna y el alumno:	Sí	No
son creativos en su propuesta.		
utilizan arreglos en la implementación de su programa.		
presentan sus dudas o investigan para solucionar estas.		
exponen su proyecto.		

Referencias	Foundation, P. S. (2023). Python. Obtenido de https://docs.python.org/3/ Joyanes Aguilar, Luis, autor Programación en Java 6 : algoritmos, programación orientada a objetos e interfaz gráfica de usuario / México : McGraw-Hill Interamericana, c2011 Torrente Artero, Óscar, autor Arduino - curso práctico de formación [México] : Alfaomega, RC Libros, 2013
Anexos	Anexo 2 Presentación proyecto implementado en Pytho y Java. Anexo 3 Presentación arreglos unidimensionales. Anexo 1 Presentación conocimientos previos.

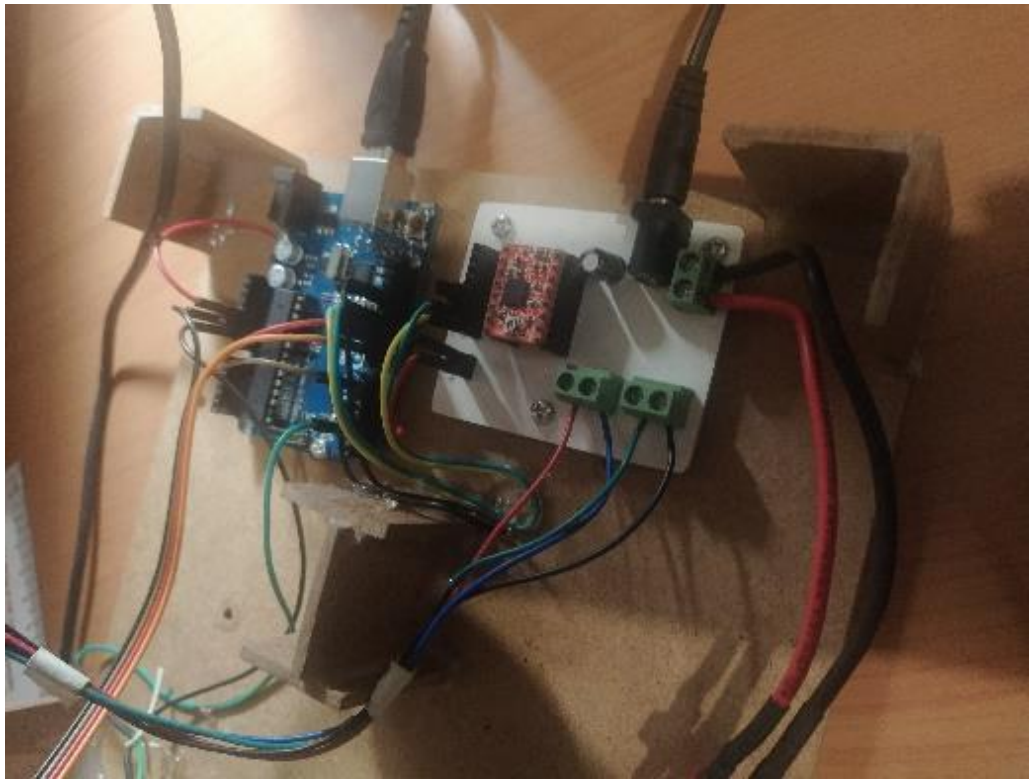
Anexo 2 Presentación proyecto implementado en Pytho y Java.

Descripción del proyecto

- El presente proyecto es un radar, rastrea objetos a través de un sensor ultrasónico en un rango de 360 grados y 20 centímetros. Los datos obtenidos se envían por medio del puerto serial y son codificados y almacenados en listas para que posteriormente la computadora represente de manera gráfica y en formato texto la ubicación en coordenadas polares y cartesianas los objetos con respecto al sensor ultrasónico. La computadora también calculará las distancias entre los objetos rastreados.

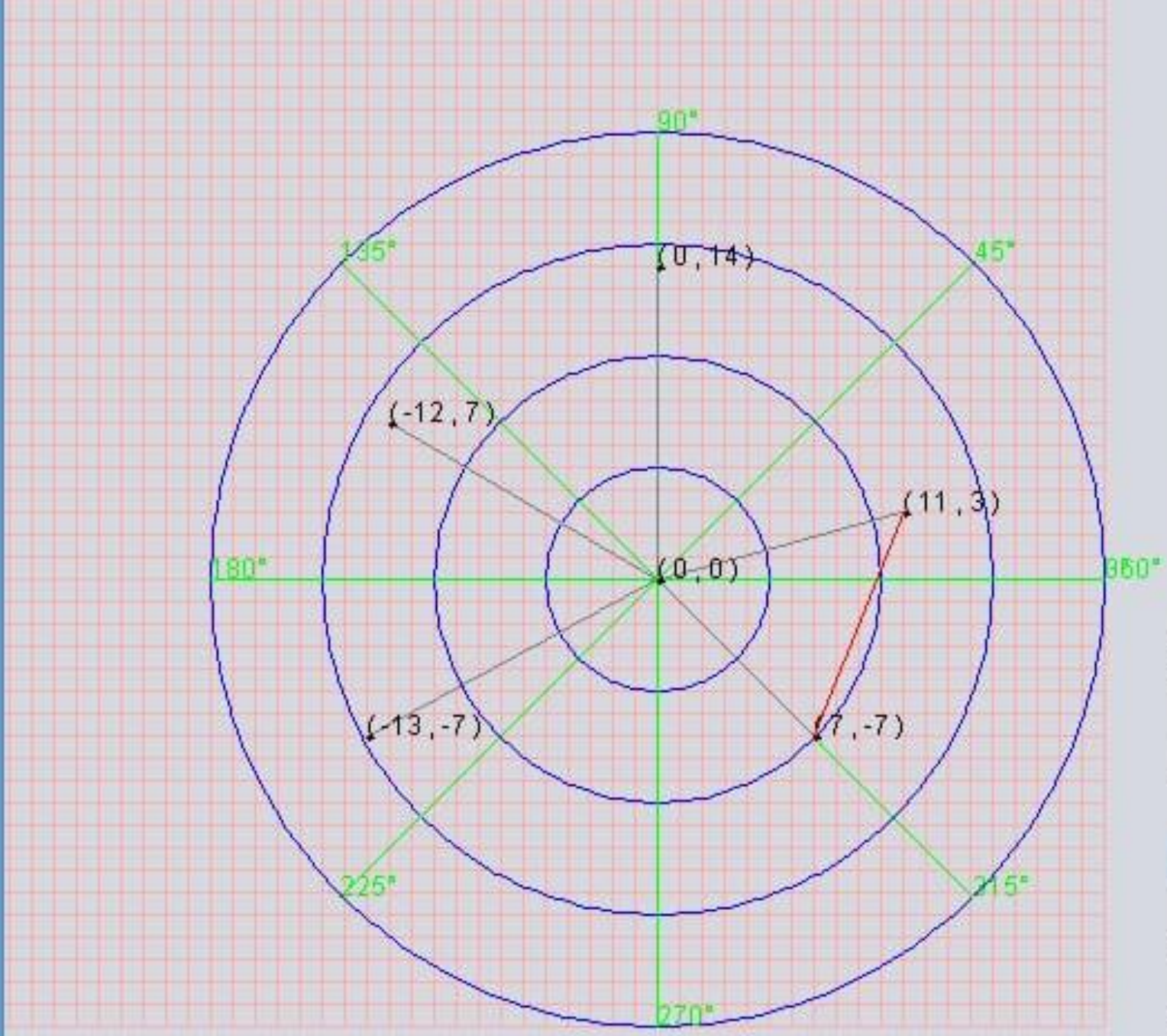
Radar ultrasónico (sensor ultrasónico, motor a pasos, placa Arduino)

ARDUINO



SENSOR ULTRASÓNICO Y MOTOR A PASOS

Proyecto implementado en Java



COM6 ▼ Conectar

R5rD11dG315g

Objeto	Distanc...	Grados	x	y
1	12	19	11	3
2	15	89	0	14
3	15	149	-12	7
4	15	208	-13	-7
5	11	315	7	-7

Graficar objetos

Calcular distancias el

Objeto 1 y Objeto 2

1 ▼ 5 ▼

Calcular distancia

Coordenadas polares

Objeto 1 Objeto 2

(12 , 19) (11 , 315)

COM6 ▼ Conectar

R5rD11dG315g

Objeto	Distanc...	Grados	x	y
1	12	19	11	3
2	15	89	0	14
3	15	149	-12	7
4	15	208	-13	-7
5	11	315	7	-7

Graficar objetos

Calcular distancias el

Objeto 1 y Objeto 2

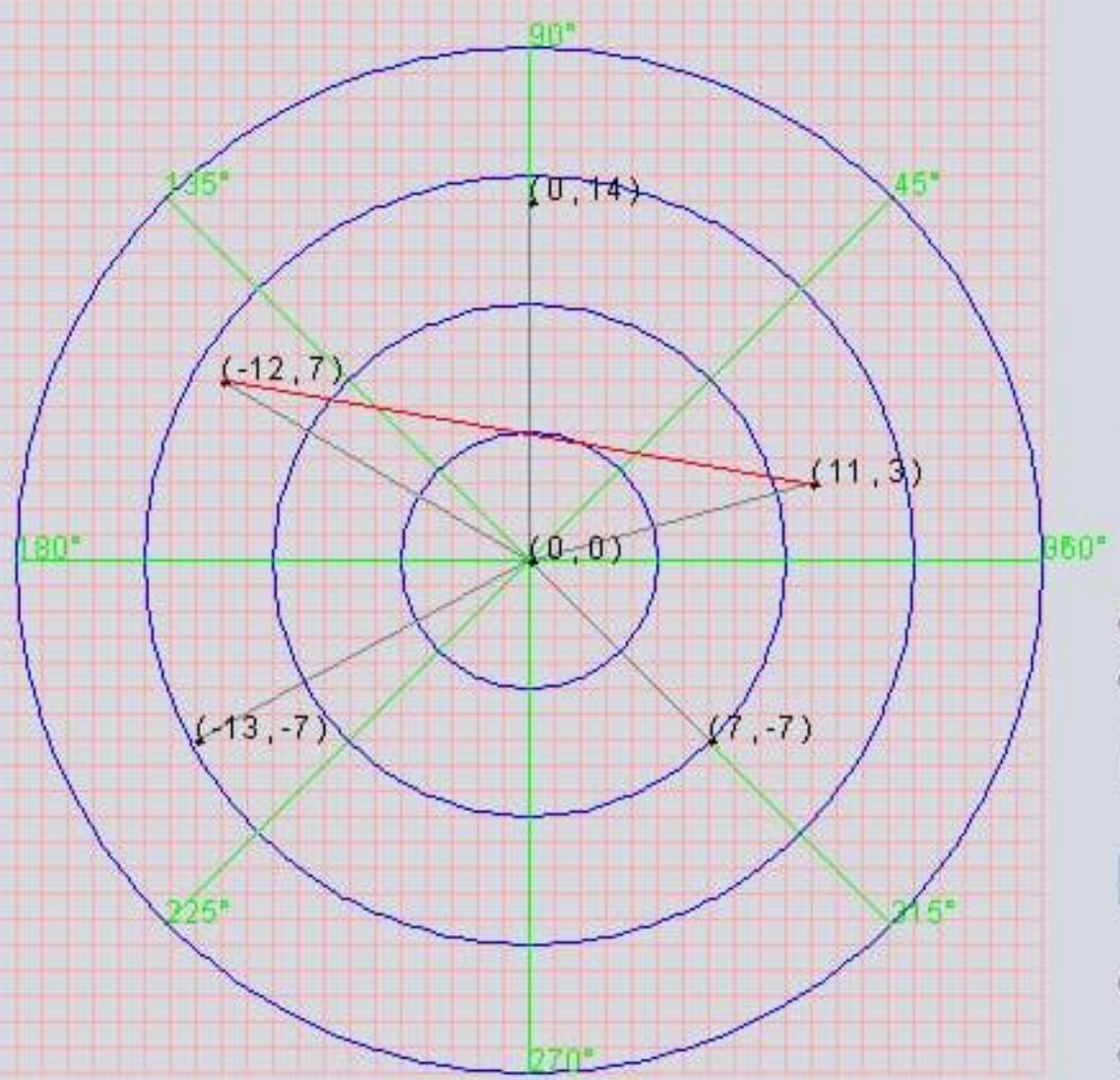
2 ▼ 3 ▼

Calcular distancia

Coordenadas polares

Objeto 1 Objeto 2

(15 , 89) (15 , 149)



COM6 ▼ Conectar

R5rD11dG315g

Objeto	Distanc...	Grados	x	y
1	12	19	11	3
2	15	89	0	14
3	15	149	-12	7
4	15	208	-13	-7
5	11	315	7	-7

Graficar objetos

Calcular distancias el

Objeto 1 y Objeto 2

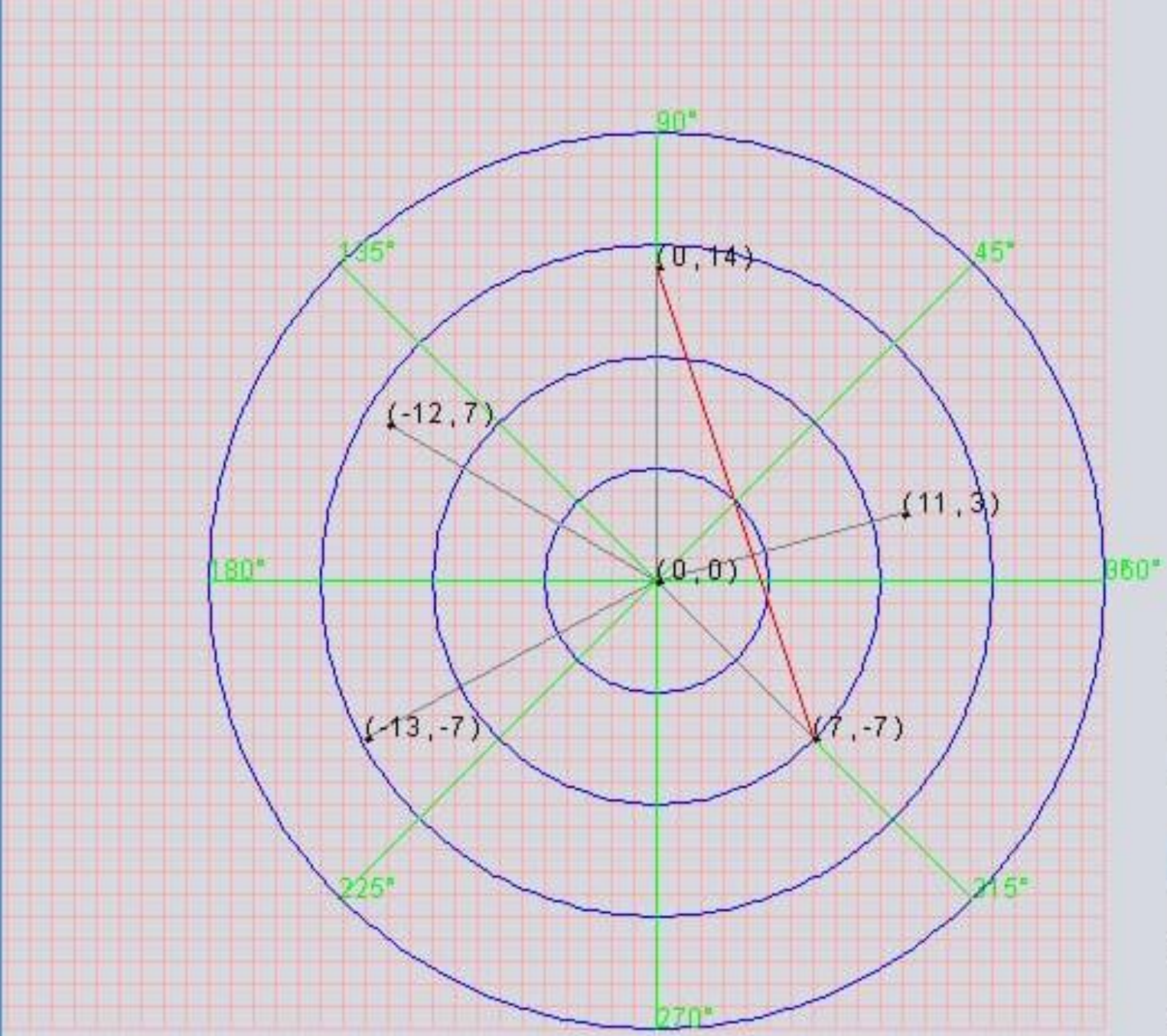
1 ▼ 3 ▼

Calcular distancia

Coordenadas polares

Objeto 1 Objeto 2

(12 , 19) (15 , 149)



COM6 ▼ Conectar

R5rD11dG315g

Objeto	Distanc...	Grados	x	y
1	12	19	11	3
2	15	89	0	14
3	15	149	-12	7
4	15	208	-13	-7
5	11	315	7	-7

Graficar objetos

Calcular distancias el

Objeto 1 y Objeto 2

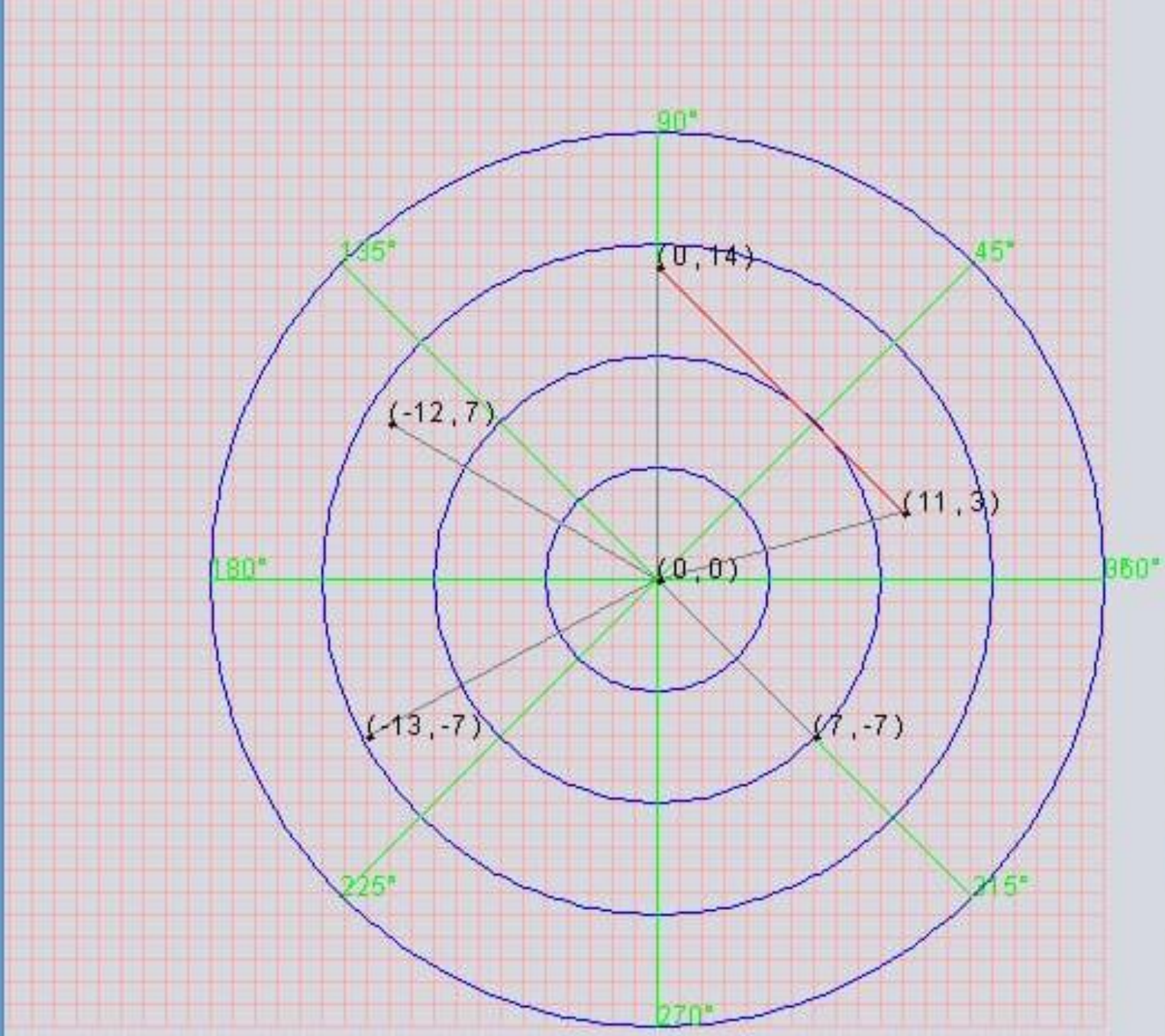
2 ▼ 5 ▼

Calcular distancia

Coordenadas polares

Objeto 1 Objeto 2

(15 , 89) (11 , 315)



COM6 ▼ Conectar

R5rD11dG315g

Objeto	Distanc...	Grados	x	y
1	12	19	11	3
2	15	89	0	14
3	15	149	-12	7
4	15	208	-13	-7
5	11	315	7	-7

Graficar objetos

Calcular distancias el

Objeto 1 y Objeto 2

1 ▼ 2 ▼

Calcular distancia

Coordenadas polares

Objeto 1 Objeto 2

(12 , 19) (15 , 89)

COM6 ▼ Conectar

R5rD11dG315g

Objeto	Distanc...	Grados	x	y
1	12	19	11	3
2	15	89	0	14
3	15	149	-12	7
4	15	208	-13	-7
5	11	315	7	-7

Graficar objetos

Calcular distancias el

Objeto 1 y Objeto 2

1 ▼ 4 ▼

Calcular distancia

Coordenadas polares

Objeto 1 Objeto 2

(12 , 19) (15 , 149)

COM6 ▼ Conectar

R5rD11dG315g

Objeto	Distanc...	Grados	x	y
1	12	19	11	3
2	15	89	0	14
3	15	149	-12	7
4	15	208	-13	-7
5	11	315	7	-7

Graficar objetos

Calcular distancias el

Objeto 1 y Objeto 2

4 ▼ 5 ▼

Calcular distancia

Coordenadas polares

Objeto 1 Objeto 2

(15 , 208) (11 , 315)

Proyecto implementado en Python

La imagen 2 representa las cadenas de carácter que recibe el programa Python desde el puerto serial.

```
['R1rD11dG8g\r\n', 'R2rD20dG64g\r\n', 'R3rD15dG108g\r\n', 'R4rD17dG157g\r\n', 'R5rD6dG275g\r\n']
```

In [96]:
```
##Autor Plata Luna Iveth Vanessa
##Importamos la librería para el uso del puerto serial y time par el menejo del tiempo.
import serial
import time
```

In [97]:
```
##El programa se conecta al puerto serial COM6 a 9600 baudios dando tiempos de demora para recibir datos.
serialArduino= serial.Serial("COM6", 9600 )
time.sleep(1)
```

In [98]:
```
#La variable cad almacenará de manera temporal los datos obtenidos del puerto serial.
cad =""
```

In [99]:
```
#Definimos una lista para almacenar los datos del puerto serial. La variable i es de control.
lista=[]
i = 0
```

In [102]:
```
#Segmento de código que obtiene datos desde el puerto serial y lo transforma a ASCII. El bucle termina hasta que se obtene una 'f
while not(cad[0:1] == 'f') :
    cad = serialArduino.readline().decode('ascii')
    print(cad)
    if not(cad[0:1] == 'f'):
        lista.append(cad)
        i=i+1
```

Posteriormente el programa decodifica por medio de funciones para el manejo de cadenas de caracteres (búsqueda de caracteres y extracción de subcadenas). Al finalizar la decodificación se almacena los grados y la distancia de los objetos en listas.

```
R1rD11dG8g

R2rD20dG64g

R3rD15dG108g

R4rD17dG157g

R5rD6dG275g

índice de caracteres
[0, 0, 0, 0, 0]
[2, 2, 2, 2, 2]
[3, 3, 3, 3, 3]
[6, 6, 6, 6, 5]
[7, 7, 7, 7, 6]
[9, 10, 11, 11, 10]
Número de dígitos
[1, 1, 1, 1, 1]
[2, 2, 2, 2, 1]
[1, 2, 3, 3, 3]
Resultados
['1', '2', '3', '4', '5']
['11', '20', '15', '17', '6']
['8', '64', '108', '157', '275']
```

In [107]:

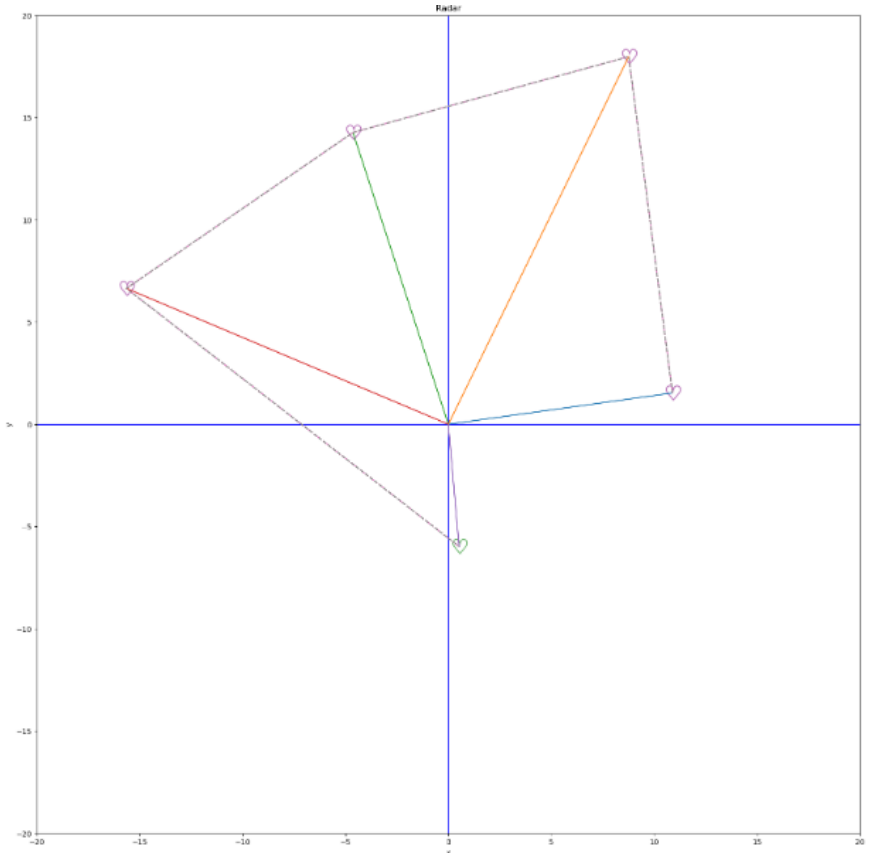
```python
#Definimos listas que servirán a decodificar la información del puerto serial.
indexO = []
indexo= []
indexD = []
indexd = []
indexG = []
indexg = []
numDigO = []
numDigD = []
numDigG = []
objetoS=[]
distanciaS=[]
gradosS=[]

#Segmento de código que decodifica las cadenas de caracter obtenidas desde el puerto serial.
#Se utiliza funciones para la manipulación de cadenas, como buscar un caracter y extraer subcadenas.
#Los resultados obtenidos se almacenan en listas.
for n in range(0, len(lista), 1):
    print(lista[n])
    cadena=lista[n]

    indexO.append(cadena.find('R'))
    indexo.append(cadena.find('r'))
    indexD.append( cadena.find('D'))
    indexd.append(cadena.find('d'))
    indexG.append(cadena.find('G'))
    indexg.append(cadena.find('g'))

    numDigO.append (indexo[n] - indexO[n] - 1)
    numDigD.append (indexd[n] - indexD[n] - 1)
    numDigG.append (indexg[n] - indexG[n] - 1)

    objetoS.append(cadena[(indexO[n]+1):(indexO[n]+1+numDigO[n])])
    distanciaS.append(cadena[(indexD[n]+1):(indexD[n]+1+numDigD[n])])
    gradosS.append(cadena[(indexG[n]+1):(indexG[n]+1+numDigG[n])])


print("índice de caracteres")
print(indexO)
print(indexo)
print(indexD)
print(indexd)
print(indexG)
print(indexg)
print("Número de dígitos")
print(numDigO)
print(numDigD)
print(numDigG)
print("Resultados")
print(objetoS)
print(distanciaS)
print(gradosS)
```

Continuando con el programa, con los datos obtenidos y almacenados (grados, distancia y objetos) se calcula las coordenadas cartesianas, para que posteriormente se represente un plano de manera gráfica.

```
[1, 2, 3, 4, 5]
[11, 20, 15, 17, 6]
[8, 64, 108, 157, 275]
[10.892948756157274, 8.76742293578155, -4.63525491562421, -15.648582508691486, 0.5229344564859473]
[1.5309041105607197, 17.97588092598334, 14.265847744427305, 6.642429184317654, -5.977168188550474]
```

In [108]:
```python
#En este segmento de código se convierten las coordenadas polares a coordenadas cartesianas.
#Para esta función utilizamos la libreria math.
import math
objeto = []
distancia = []
grados = []

for n in range(0, len(lista), 1):

    objeto.append (int (objetoS[n]))
    distancia.append (int (distanciaS[n]))
    grados.append(int (gradosS[n]))

x=[]
y=[]
for n in range(0, len(lista), 1):
    x.append(distancia[n] * math.cos(math.radians(grados[n])));
    y.append(distancia[n] * math.sin(math.radians(grados[n])));
print(objeto)
print(distancia)
print(grados)
print(x)
print(y)
```

Para graficar utilizamos la librería numpy y matplotlib. Utilizamos las listas de los objetos, grados, distancia y coordenadas para graficar.



In [109]:
```
#Para graficar utilizamos las librerias mumpy y matplotlib.
import numpy as np
import matplotlib.pyplot as plt
```

In [110]:
```
#Es importante convertir la lista a un tipo arreglo para poder graficar.
xnp = np.array(x)
ynp = np.array(y)
print(xnp)
print(ynp)
```

```
[ 10.89294876   8.76742294  -4.63525492 -15.64858251   0.52293446]
[ 1.53090411 17.97588093 14.26584774  6.64242918 -5.97716819]
```

In [111]:
```
#Configuración del gráfico


fig, ax = plt.subplots(figsize = (20,20))

plt.xlabel("x")
plt.ylabel("y")
plt.xlim(-20,20)
plt.ylim(-20,20)
plt.title("Radar")

plt.hlines(0,-20,20, color="blue")
plt.vlines(0,-20,20, color="blue")

for n in range(0, len(lista), 1):
    plt.plot([0,x[n]],[0,y[n]])

#Configuración del color
color = np.where((xnp<= 0)  , "red", "blue")
color = np.where((ynp <= 0)  , "green", "purple")


plt.scatter(xnp, ynp, c=color, label=color, s=500, marker=r'$\heartsuit$', alpha=0.4 )
plt.plot(xnp, ynp, linestyle="dotted")
plt.plot(xnp, ynp, linestyle="dashdot")
plt.plot(xnp, ynp, linestyle="dashed")
plt.show()
```

También el programa en Python muestra los resultados de las coordenadas de cada objeto, la distancia con respecto al radar, los grados y el objeto.

```
Objeto:  1
Distancia en la que se encuentra el objeto:  11
Grados en la que se encuentra el objeto:  8
Coordenadas en el plano cartesiano: [  10.892948756157274  ,  1.5309041105607197  ]


Objeto:  2
Distancia en la que se encuentra el objeto:  20
Grados en la que se encuentra el objeto:  64
Coordenadas en el plano cartesiano: [  8.76742293578155  ,  17.97588092598334  ]


Objeto:  3
Distancia en la que se encuentra el objeto:  15
Grados en la que se encuentra el objeto:  108
Coordenadas en el plano cartesiano: [  -4.63525491562421  ,  14.265847744427305  ]


Objeto:  4
Distancia en la que se encuentra el objeto:  17
Grados en la que se encuentra el objeto:  157
Coordenadas en el plano cartesiano: [  -15.648582508691486  ,  6.642429184317654  ]


Objeto:  5
Distancia en la que se encuentra el objeto:  6
Grados en la que se encuentra el objeto:  275
Coordenadas en el plano cartesiano: [  0.5229344564859473  ,  -5.977168188550474  ]
```

In [112]:
```python
#En este segmento de código se despliegan los datos obtenidos desde el puerto serial, objeto, grados, distancia.
#También se imprime las coordenadas cartesianas.
for n in range(0, len(lista), 1):
    print("Objeto: " , objeto[n])
    print("Distancia en la que se encuentra el objeto: ", distancia[n])
    print("Grados en la que se encuentra el objeto: " , grados[n])
    print("Coordenadas en el plano cartesiano: [ " , x[n] , " , " , y[n] , " ]")
    print("\n")
```

Para continuar con el cálculo de las distancias entre los objetos, primero hacer todas las posibles combinaciones que puede tener cada objeto. Para esto creamos algunas listas que ayudaran a la generación de todas las posibilidades.

```
objetos:  [1, 2, 3, 4, 5]
Serie objetos:  [1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5]
combinaciones:  [1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

In [113]:

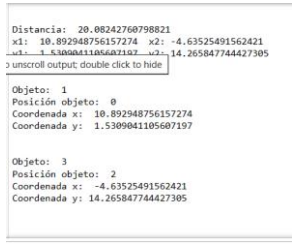
```python
#Para calcular la distancia entre dos objetos es importante crear todas las posibles combinaciones entre los objetos.
#Generamos algunas listas para las posibles combinaciones.
objetos=[]
combinaciones=[]
serieObjetos=[]

for n in range(1, len(lista)+1, 1):
    objetos.append(n)

for m in range(1, len(objetos)+1, 1):
    for n in range(1, len(objetos)+1, 1):
        serieObjetos.append(m)

for m in range(1, len(lista)+1, 1):
    for n in range(1, len(lista)+1, 1):
        combinaciones.append(n)

print("objetos: ",objetos)
print("Serie objetos: ", serieObjetos)
print("combinaciones: ",combinaciones)
```

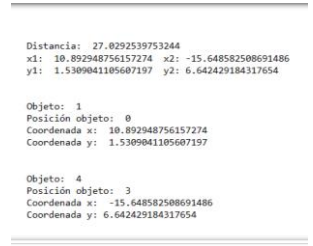
A continuación, llenamos todas las listas () con sus posibilidades.

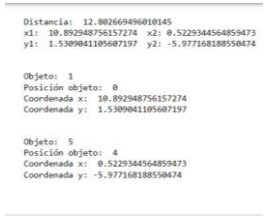
```
Combinación objetosXY1 :  [[1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5]]
Posición objetosXY1 :  [0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4]
x1: [10.892948756157274, 10.892948756157274, 10.892948756157274, 10.892948756157274, 10.892948756157274, 8.76742293578155, 8.76
742293578155, 8.76742293578155, 8.76742293578155, 8.76742293578155, -4.63525491562421, -4.63525491562421, -4.63525491562421, -
4.63525491562421, -4.63525491562421, -15.648582508691486, -15.648582508691486, -15.648582508691486, -15.648582508691486, -15.64
8582508691486, 0.5229344564859473, 0.5229344564859473, 0.5229344564859473, 0.5229344564859473, 0.5229344564859473]
y1: [1.5309041105607197, 1.5309041105607197, 1.5309041105607197, 1.5309041105607197, 1.5309041105607197, 17.97588092598334, 17.
97588092598334, 17.97588092598334, 17.97588092598334, 17.97588092598334, 14.265847744427305, 14.265847744427305, 14.26584774442
7305, 14.265847744427305, 14.265847744427305, 6.642429184317654, 6.642429184317654, 6.642429184317654, 6.642429184317654, 6.642
429184317654, -5.977168188550474, -5.977168188550474, -5.977168188550474, -5.977168188550474, -5.977168188550474]


Combinación objetosXY2 :  [[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]]
Posición ObjetosXY2:  [0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4]
x2: [10.892948756157274, 8.76742293578155, -4.63525491562421, -15.648582508691486, 0.5229344564859473, 10.892948756157274, 8.76
742293578155, -4.63525491562421, -15.648582508691486, 0.5229344564859473, 10.892948756157274, 8.76742293578155, -4.635254915624
21, -15.648582508691486, 0.5229344564859473, 10.892948756157274, 8.76742293578155, -4.63525491562421, -15.648582508691486, 0.52
29344564859473, 10.892948756157274, 8.76742293578155, -4.63525491562421, -15.648582508691486, 0.5229344564859473]
y2: [1.5309041105607197, 17.97588092598334, 14.265847744427305, 6.642429184317654, -5.977168188550474, 1.5309041105607197, 17.9
7588092598334, 14.265847744427305, 6.642429184317654, -5.977168188550474, 1.5309041105607197, 17.97588092598334, 14.26584774442
7305, 6.642429184317654, -5.977168188550474, 1.5309041105607197, 17.97588092598334, 14.265847744427305, 6.642429184317654, -5.9
77168188550474, 1.5309041105607197, 17.97588092598334, 14.265847744427305, 6.642429184317654, -5.977168188550474]
```

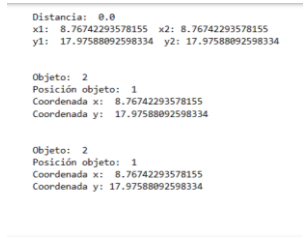
In [114]:
```
#Para continuar con las posibles combinaciones, llenamos las listas con todas las posibilidades.
posObjetosXY1=[]
posObjetosXY2=[]
combObjetosXY1=[]
combObjetosXY2=[]
combX1=[]
combX2=[]
combY1=[]
combY2=[]


combDistanciaObjetos=[]

for n in range(0, len(serieObjetos), 1):
    print("Objeto: ",serieObjetos[n],"Posición objeto: ", objeto.index(serieObjetos[n]),"Coordenada x: ",x[objeto.index(serieObje
    posObjetosXY1.append(objeto.index(serieObjetos[n]))
    combX1.append(x[objeto.index(serieObjetos[n])])
    combY1.append(y[objeto.index(serieObjetos[n])])
combObjetosXY1.append(serieObjetos)

print("\n")
for m in range(0, len(combinaciones), 1):
    print("Objeto: ", combinaciones[m],"Posición objeto: ", objeto.index(combinaciones[m]),"Coordenada x: ",x[objeto.index(combir
    posObjetosXY2.append(objeto.index(combinaciones[m]))
    combX2.append(x[objeto.index(combinaciones[m])])
    combY2.append(y[objeto.index(combinaciones[m])])
combObjetosXY2.append(combinaciones)

print("\n")

print("Combinación objetosXY1 : ",combObjetosXY1)
print("Posición objetosXY1 : ",posObjetosXY1)
print("x1:", combX1)
print("y1:", combY1)
print("\n")

print("Combinación objetosXY2 : ",combObjetosXY2)
print("Posición ObjetosXY2: ",posObjetosXY2)
print("x2:", combX2)
print("y2:", combY2)
```

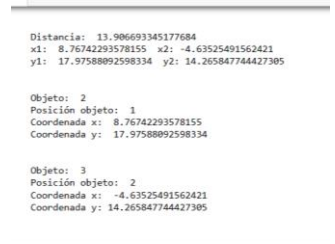
En el antepenúltimo paso, el programa calcula todas las distancias que puede existir entre los objetos.

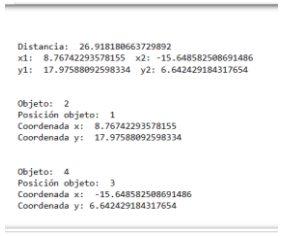


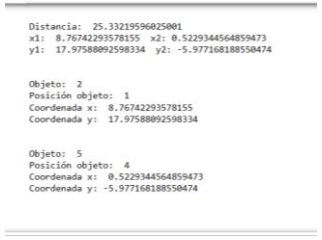


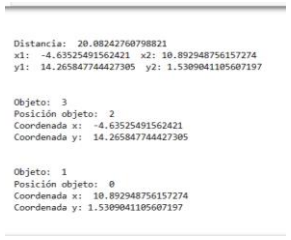


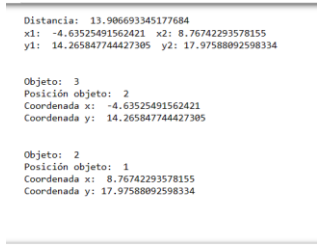


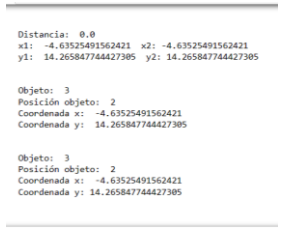


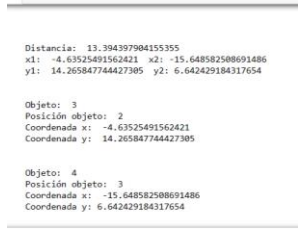


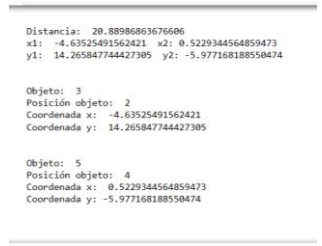


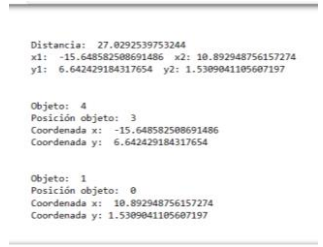


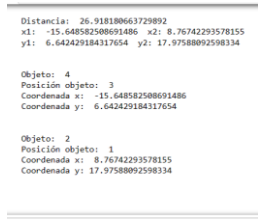


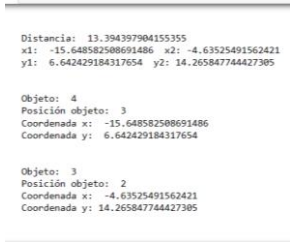

Distancia: 0.0
x1: -4.63525491562421 x2: -4.63525491562421
y1: 14.265847744427305 y2: 14.265847744427305
Objeto: 3
Posición objeto: 2
Coordenada x: -4.63525491562421
Coordenada y: 14.265847744427305
Objeto: 3
Posición objeto: 2
Coordenada x: -4.63525491562421
Coordenada y: 14.265847744427305

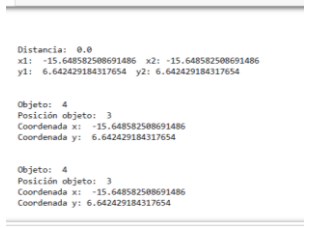

Distancia: 13.394397904155355
x1: -4.63525491562421 x2: -15.648582508691486
y1: 14.265847744427305 y2: 6.642429184317654
Objeto: 3
Posición objeto: 2
Coordenada x: -4.63525491562421
Coordenada y: 14.265847744427305
Objeto: 4
Posición objeto: 3
Coordenada x: -15.648582508691486
Coordenada y: 6.642429184317654

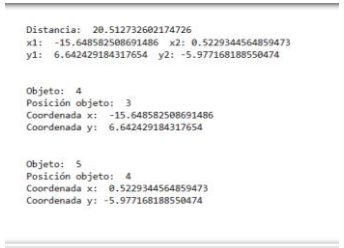

Distancia: 20.88986863676606
x1: -4.63525491562421 x2: 0.5229344564859473
y1: 14.265847744427305 y2: -5.977168188550474
Objeto: 3
Posición objeto: 2
Coordenada x: -4.63525491562421
Coordenada y: 14.265847744427305
Objeto: 5
Posición objeto: 4
Coordenada x: 0.5229344564859473
Coordenada y: -5.977168188550474

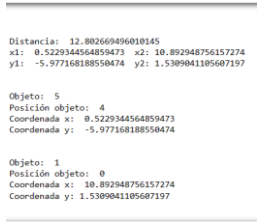

Distancia: 27.0292539753244
x1: -15.648582508691486 x2: 10.892948756157274
y1: 6.642429184317654 y2: 1.5309041105607197
Objeto: 4
Posición objeto: 3
Coordenada x: -15.648582508691486
Coordenada y: 6.642429184317654
Objeto: 1
Posición objeto: 0
Coordenada x: 10.892948756157274
Coordenada y: 1.5309041105607197

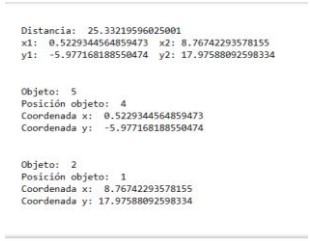

Distancia: 26.918180663729892
x1: -15.648582508691486 x2: 8.76742293578155
y1: 6.642429184317654 y2: 17.97588092598334
Objeto: 4
Posición objeto: 3
Coordenada x: -15.648582508691486
Coordenada y: 6.642429184317654
Objeto: 2
Posición objeto: 1
Coordenada x: 8.76742293578155
Coordenada y: 17.97588092598334

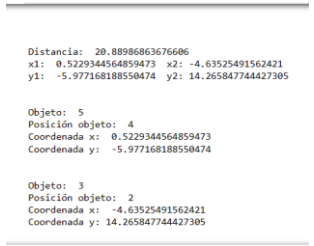

Distancia: 13.394397904155355
x1: -15.648582508691486 x2: -4.63525491562421
y1: 6.642429184317654 y2: 14.265847744427305
Objeto: 4
Posición objeto: 3
Coordenada x: -15.648582508691486
Coordenada y: 6.642429184317654
Objeto: 3
Posición objeto: 2
Coordenada x: -4.63525491562421
Coordenada y: 14.265847744427305

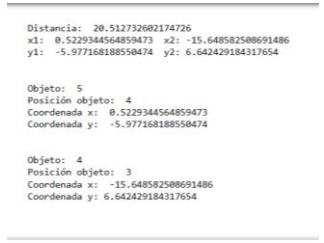

Distancia: 0.0
x1: -15.648582508691486 x2: -15.648582508691486
y1: 6.642429184317654 y2: 6.642429184317654
Objeto: 4
Posición objeto: 3
Coordenada x: -15.648582508691486
Coordenada y: 6.642429184317654
Objeto: 4
Posición objeto: 3
Coordenada x: -15.648582508691486
Coordenada y: 6.642429184317654


Distancia: 20.512732602174726
x1: -15.648582508691486 x2: 0.5229344564859473
y1: 6.642429184317654 y2: -5.977168188550474
Objeto: 4
Posición objeto: 3
Coordenada x: -15.648582508691486
Coordenada y: 6.642429184317654
Objeto: 5
Posición objeto: 4
Coordenada x: 0.5229344564859473
Coordenada y: -5.977168188550474


Distancia: 12.802669496010145
x1: 0.5229344564859473 x2: 10.892948756157274
y1: -5.977168188550474 y2: 1.5309041105607197
Objeto: 5
Posición objeto: 4
Coordenada x: 0.5229344564859473
Coordenada y: -5.977168188550474
Objeto: 1
Posición objeto: 0
Coordenada x: 10.892948756157274
Coordenada y: 1.5309041105607197


Distancia: 25.33219596025001
x1: 0.5229344564859473 x2: 8.76742293578155
y1: -5.977168188550474 y2: 17.97588092598334
Objeto: 5
Posición objeto: 4
Coordenada x: 0.5229344564859473
Coordenada y: -5.977168188550474
Objeto: 2
Posición objeto: 1
Coordenada x: 8.76742293578155
Coordenada y: 17.97588092598334

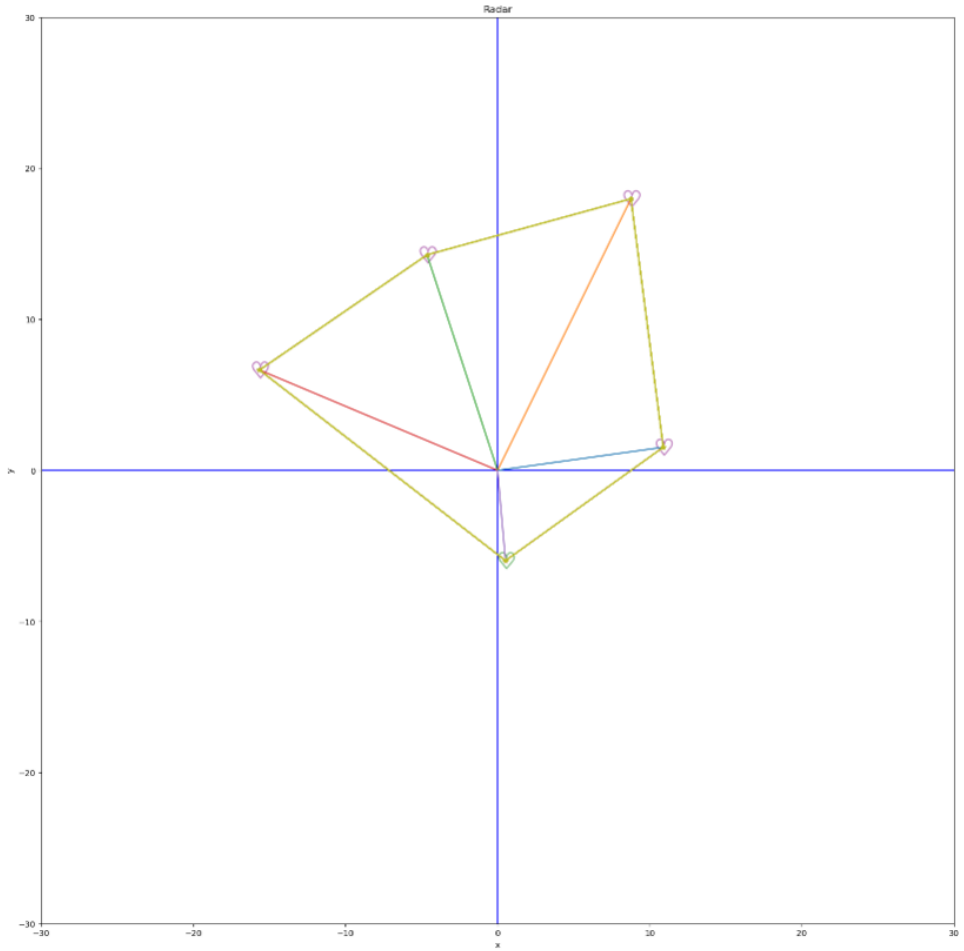

Distancia: 20.88986863676606
x1: 0.5229344564859473 x2: -4.63525491562421
y1: -5.977168188550474 y2: 14.265847744427305
Objeto: 5
Posición objeto: 4
Coordenada x: 0.5229344564859473
Coordenada y: -5.977168188550474
Objeto: 3
Posición objeto: 2
Coordenada x: -4.63525491562421
Coordenada y: 14.265847744427305


Distancia: 20.512732602174726
x1: 0.5229344564859473 x2: -15.648582508691486
y1: -5.977168188550474 y2: 6.642429184317654
Objeto: 5
Posición objeto: 4
Coordenada x: 0.5229344564859473
Coordenada y: -5.977168188550474
Objeto: 4
Posición objeto: 3
Coordenada x: -15.648582508691486
Coordenada y: 6.642429184317654

In [116]:

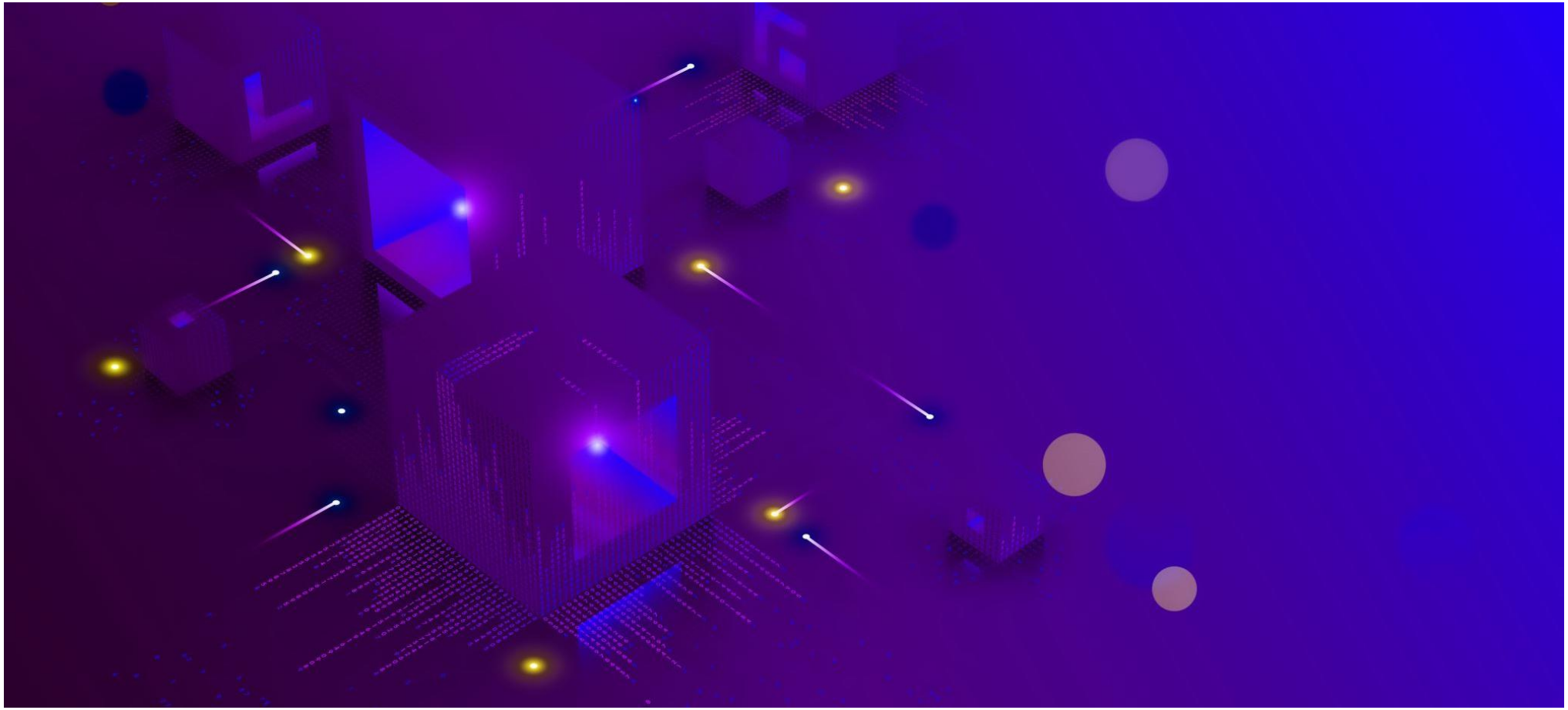
```
#En este segmento de código se calcula la distancia de todas las posibles combinaciones entre los objetos y
#se despliegan los resultados.
print(objetos)
print(x)
print(y)
print("\n")
print("\n")
for m in range(0, len(combinaciones), 1):
    print("\n")
    print("\n")
    combDistanciaObjetos.append(float(math.sqrt((combX1[m]-combX2[m])**2+(combY1[m]-combY2[m])**2)))
    print("Distancia: ", float(math.sqrt((combX1[m]-combX2[m])**2+(combY1[m]-combY2[m])**2)) )
    print("x1: ",combX1[m]," x2:", combX2[m], )
    print("y1: ",combY1[m]," y2:",combY2[m])
    print("\n")
    print("Objeto: ",serieObjetos[m])
    print("Posición objeto: ", objeto.index(serieObjetos[m]))
    print("Coordenada x: ",x[objeto.index(serieObjetos[m])])
    print("Coordenada y: ",y[objeto.index(serieObjetos[m])])
    print("\n")
    print("Objeto: ", combinaciones[m])
    print("Posición objeto: ", objeto.index(combinaciones[m]))
    print("Coordenada x: ",x[objeto.index(combinaciones[m])])
    print("Coordenada y:",y[objeto.index(combinaciones[m])])

print(combDistanciaObjetos)
print("\n")
```

Por último, volvemos a graficar los objetos en el plano cartesiano.



In [117]:
```
#Volvemos a graficar.

fig, ax = plt.subplots(figsize = (20,20))



plt.xlabel("x")
plt.ylabel("y")
plt.xlim(-30,30)
plt.ylim(-30,30)
plt.title("Radar")

plt.hlines(0,-30,30, color="blue")
plt.vlines(0,-30,30, color="blue")

for n in range(0, len(lista), 1):
    plt.plot([0,x[n]],[0,y[n]])

#Configuración del color
color = np.where((xnp<= 0)  , "red", "blue")
color = np.where((ynp <= 0)  , "green", "purple")


plt.scatter(xnp, ynp, c=color, label=color, s=500, marker=r'$\heartsuit$', alpha=0.4 )
ax.plot(combX1,combY1, marker='*')
ax.plot(combX2,combY2, marker='*')


ax.plot(combX1,combY1, marker='*')
ax.plot(combX2,combY2, marker='*')
plt.show()
```

Anexo 3 Presentación arreglos unidimensionales.



Arreglos unidimensionales

Resuelve problemas que involucren el uso de arreglos unidimensionales en los métodos de una Clase.

Desarrollo de programas que involucren el uso de arreglos unidimensionales en los métodos de una clase.

Elaboró: Plata Luna Iveth Vanessa

Definición

10.5	9.6	10.5	8.5	9.6
[0]	[1]	[2]	[3]	[4]

- Un arreglo unidimensional es un conjunto de datos, con la misma estructura (tipo de datos) y con el mismo nombre.
- El usuario define el tipo de dato que almacenará la estructura.
- Se puede leer y escribir este tipo de estructura de datos a través de su nombre y para identificar cada elemento se realiza a través de su índice, empezando con cero.

Definición de un arreglo

- Sintaxis:

TipoDeDato**[]** nombreDelArreglo;

- Ejemplos:

int**[]** conjuntoEnteros; //se define un arreglo que almacena un conjunto de enteros.

Toda esta estructura se llama conjuntoEnteros. Puede almacenar hasta cinco elementos, cada casilla se identifica por un índice y este empieza desde cero.

double[] calificaciones; //Se define un arreglo que almacena un conjunto de doubles.

boolean[] arregloBooleano;

char[] arregloCaracteres;

long[] arregloEnteroLargo;

En contraste con una variable:

double numero; //definición de una variable

Inicialización de un arreglo

- Inicializar es crear espacio en memoria RAM para almacenar el conjunto de datos.
- double[] calificaciones; //Declarar
- **calificaciones = new double[5] //instanciar**
- //Este arreglo se instancia con cinco elementos.

10.5	9.6	10.5	8.5	9.6
[0]	[1]	[2]	[3]	[4]
calificaciones[0]	calificaciones[1]	calificaciones[2]	calificaciones[3]	calificaciones[4]

¿Qué pasaría si no tengo la estructura de arreglo en Java?

- Tengo que declarar los elementos del arreglo uno a uno.

- double calificaciones1; //variable
- double calificaciones2; //variable
- double calificaciones3; //variable
- double calificaciones4; //variable
- double calificaciones5; //variable

Tipos de arreglos

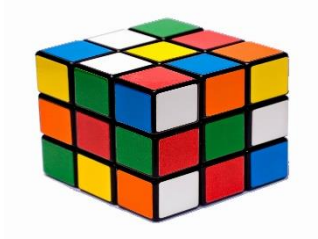
- Unidimensional – Vector (Una sola fila y varias columnas)

[0]	**[1]**	**[2]**	**[3]**	**[4]**

Bidimensionales [2][5] o matrices (filas y columna)
[filas][columnas]

[0][0]	**[0][1]**	**[0][2]**	**[0][3]**	**[0][4]**
[1][0]	[1][1]	[1][2]	[1][3]	[1][4]

Tridimencionales[][][]



Multidimensionales [2][5][]…[]

Filas Horizontal →
Columnas Vertical ↓

Almacenamiento de datos en un arreglo

```
int[] conjuntoEnteros;

conjuntoEnteros = new int[5];

conjuntoEnteros[0]=20;

conjuntoEnteros[1]=21;

conjuntoEnteros[2]=22;

conjuntoEnteros[3]=23;

conjuntoEnteros[4]=24;

conjuntoEnteros[5]=25; //error de
desbordamiento
```

Estructura **conjuntoEntero**

Memoria	20	21	22	23	24
Índice	[0]	[1]	[2]	[3]	[4]

Almacenamiento de datos de un arreglo a través de estructuras repetitivas

```
for(int i=0; i<5; i++){

    System.out.println("Favor de teclear el número para el conjuntoEnteros[" + i + "] = ");

    conjuntoEnteros[i] = teclado.nextInt();

}
```

Pantalla
Primera iteración:
Favor de teclear el número para el conjuntoEntero[0] =
5
Segunda iteración:
Favor de teclear el número para el conjuntoEntero[1] =
9

Tercera iteración:
Favor de teclear el número para el conjuntoEntero[2] =
2

Cuarta iteración:
Favor de teclear el número para el conjuntoEntero[3] =
8

Quinta iteración:
Favor de teclear el número para el conjuntoEntero[4] =
12

Memoria	**5**	**9**	**2**	**8**	**12**
Índice	conjuntoEntero [0]	conjuntoEntero [1]	conjuntoEntero [2]	conjuntoEntero [3]	conjuntoEntero [4]

Despliegue de información de un arreglo

```
public class Main{

     public static void main(String[] args) {

    int[] conjuntoEnteros = {10, 11, 12, 13, 14};

        System.out.println(conjuntoEnteros[0]);

     System.out.println(conjuntoEnteros[1]);

     System.out.println(conjuntoEnteros[2]);

     System.out.println(conjuntoEnteros[3]);

     System.out.println(conjuntoEnteros[4]);

     }//cierra el método principal main
```

Pantalla

10
11
12
13
14

espliegue de información de un arreglo a avés de estructuras repetitivas

```
public class Main

{

      public static void main(String[] args) {

            int[] conjuntoEnteros = {10, 11, 12, 13, 14};

            for(int i=0; i<5; i++){

                System.out.println("conjuntoEnteros[" + i + "] = " + conjuntoEnteros[i]);

      }//cierra el ciclo for

      }//cierra el método principal main

}//cierra la clase
```

Pantalla

```
conjuntoEnteros[0]=10
conjuntoEnteros[1]=11
conjuntoEnteros[2]=12
conjuntoEnteros[3]=13
conjuntoEnteros[4]=14
```

Memoria	10	11	12	13	14
Índice	[0]	[1]	[2]	[3]	[4]

```
import java.util.Scanner;

public class Main

{

        public static void main(String[] args) {

                Scanner teclado = new Scanner(System.in);

                //Llena el arreglo

                int conjuntoEnteros[] = {10, 11, 12, 13, 14};

                //Despliega información del arreglo

                for(int i=0; i<5; i++){

                    System.out.println("conjuntoEnteros[" + i+"] = " + conjuntoEnteros[i]);

                }

                //Llena el arreglo de manera manual

                conjuntoEnteros[0]=20;

                conjuntoEnteros[1]=21;

                conjuntoEnteros[2]=22;

                conjuntoEnteros[3]=23;

                conjuntoEnteros[4]=24;

                //Despliega la información del arreglo

                for(int i=0; i<5; i++){

                    System.out.println("ConjuntoEnteros[" + i +"] =" + conjuntoEnteros[i]);

                }

                //Utiliza un bucle para llenar el arreglo

                for(int i=0; i<5; i++){

                    System.out.println("Favor de teclear el numero para conjuntoEnteros["+ i + "] = ");
```

Ejemplo

- Elabora un programa en Java donde se pida siete calificaciones del semestre, se almacene en un arreglo estas calificaciones, se acumula la suma de las calificaciones en una variable llamada suma, al finalizar el bucle el programa realizará el promedio e imprimirá las **calificaciones y el promedio.**

```
import java.util.Scanner;

public class Main
{
	public static void main(String[] args) {
		Scanner teclado = new Scanner(System.in);
		double suma=0, promedio;
		double[] calificaciones = new double[7];

		for(int i=0; i<7; i++){

		    System.out.println("Favor de introducir la calificación " + (i+1) + ":");
		    calificaciones[i] = teclado.nextDouble();
		    suma = calificaciones[i] + suma;
		}

		promedio = suma/7;
		System.out.println("El promedio es:" + promedio);
		for(int i=0; i<7; i++){

		    System.out.println("CalificaciÃ³n " + (i+1) + ": " + calificaciones[i]);
		}
	}//Cierra main
}//cierra Main
```

```
Favor de introducir la calificación 1:

4

Favor de introducir la calificación 2:
7

Favor de introducir la calificación 3:
8

Favor de introducir la calificación 4:
1

Favor de introducir la calificación 5:
9

Favor de introducir la calificación 6:
4

Favor de introducir la calificación 7:
10

Promedio: 6.14

Calificación 1: 4
Calificación 2: 7
Calificación 3: 8
Calificación 4: 1
Calificación 5: 9
Calificación 6: 4
Calificación 7: 10
```

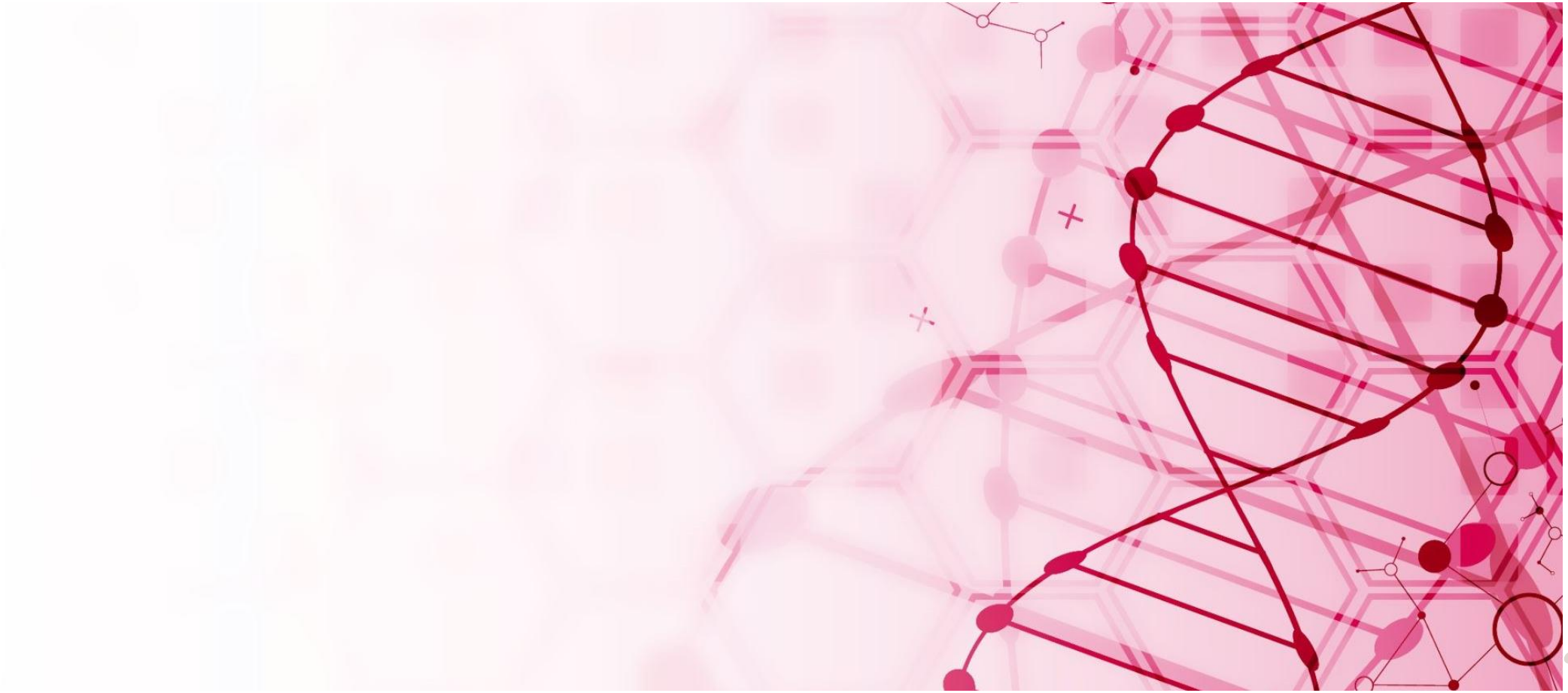
Memoria	4	7	8	1	9	4	10
Índice	[0]	[1]	[2]	[3]	[4]	[5]	[6]

Ejercicio

- En una papelería se necesita determinar el subtotal, IVA y total de diez productos.

- Realiza un programa en Java donde se solicite el monto (similar a la estructura calificaciones[], nada más que esta estructura se llama montos[]) de cada uno de los productos y los almacena en un arreglo de diez elementos double montos[10]).
- En el mismo bucle se acumulará poco a poco el subtotal (similar a la variable suma, nada más que esta se llama subtotal) por cada producto que pase por el lector de código de barras en una variable que lleva su mismo nombre.
- Al finalizar el bucle el usuario imprimirá el subtotal, calculará el IVA multiplicando el subtotal por el 0.16 (afuera del bucle se realiza la siguiente multiplicación subtotal * 0.16, el resultado del producto es el IVA) y se sumará el subtotal más el IVA, teniendo como resultado el total (total = subtotal + IVA).
- Es importante desplegar en pantalla el monto de cada producto (monto[i]), el subtotal, IVA y total).

Anexo 1 Presentación conocimientos previos.



Elaborado por

Estructuras de repetición

En la programación utilizamos estructuras repetitivas para repetir una secuencia de instrucciones n veces, sin necesidad de repetir estas sentencias.

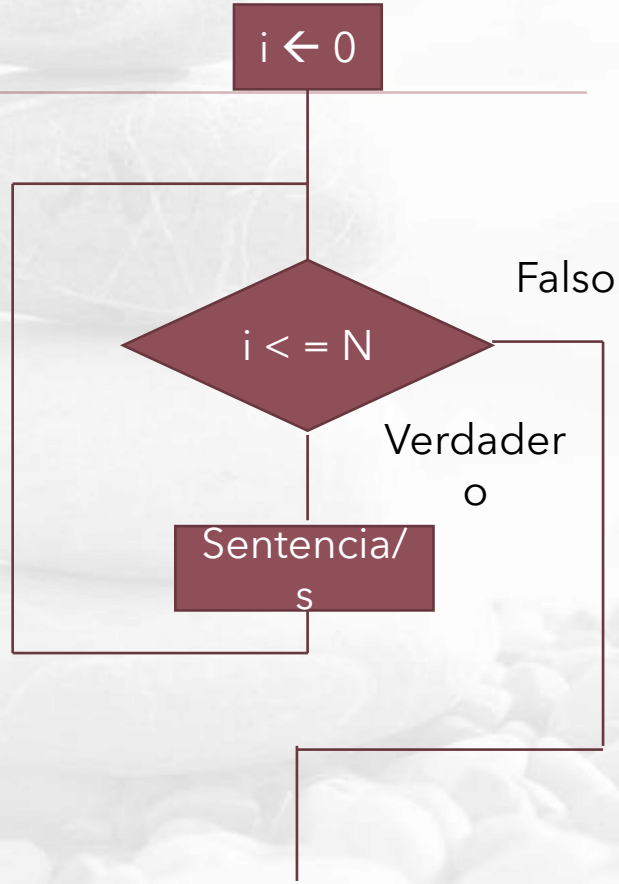
Este tipo de estructuras llamados bucle nos permite reducir líneas de código.

Por ejemplo, si deseo escribir cien veces en pantalla “Estoy bien”, puedo escribir solamente una línea dentro de la estructura de repetición.

En esta presentación vamos a revisar las tres estructuras de repetición: for, while y do-while.

Estructura repetitiva for: sintaxis y diagrama de flujo

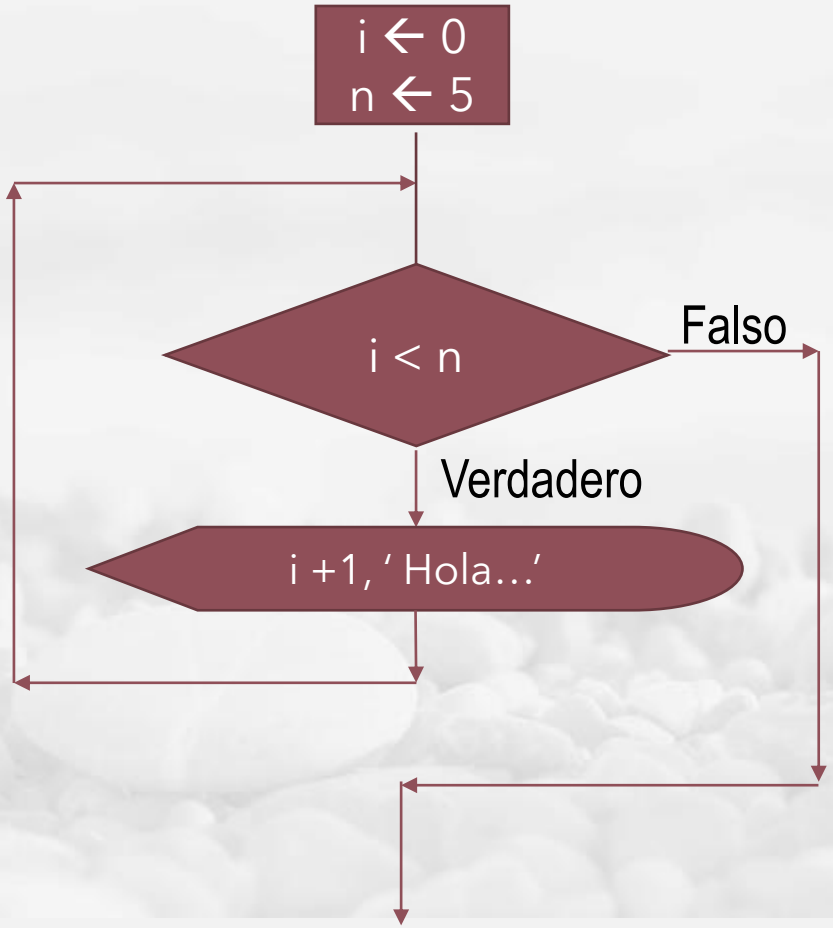
```
for (int i=valorInicial; condición; incremento){
        Sentencias;
}
```



- La sentencia o estructura for repite la ejecución de una o varias sentencias un número fijo de veces, previamente establecido.
- Esta estructura (for) necesita una variable de control del bucle que es necesariamente de tipo ordinal, ya que el bucle se ejecuta mientras la variable de control toma una serie consecutiva de valores de tipo ordinal, comprendidos entre dos valores extremos (inferior y superior).
- La condición es de tipo lógico, donde los resultados son verdadero o falso, puedes usar operaciones relacionales (igual, mayor, menor mayor o igual, menor o igual, diferente, entre otros).

Estructura repetitiva for: ejemplo1

Este programa imprime cinco veces “Hola…”.




```
n=5;

for (int i=0; i < n; i++){

        System.out.println((i+1) + “Hola…”);

}
```

Estructura repetitiva for: ejemplo1

```
n=5;

for (int i=0; i < n; i++){

        System.out.println((i+1) + "Hola…");

}
```

Este programa imprime cinco veces "Hola…".

1. Inicializa la variable, contiene el valor máximo, cinco.

   n=5;

2. Inicializa la variable de control con el valor inicial, cero.

   int i=0;

3. Compara la variable de control con la variable que almacena el valor final.

   i < n;

1. Si es verdad la condición imprime "Hola…".

   System.out.println((i+1) + " Hola…");

   El valor de i no se modifica en memoria, solo se muestra la suma i+1 en pantalla, pero no altera la variable como i++

2. Incrementa el valor en una posición.

   i++

   i++ se encuentra abreviado, es lo mismo que esta estructura:

   i = i + 1

3. Regresa a la condición y evalúa esta.

   i < n

   Recuerda que el valor de la variable i cambio en un posición por el incremento.

4. Si la condición es verdadera vuelve a entrar al bucle, pero si es falsa sale del bucle.

Prueba de escritorio: primera iteración.

i	0	1		
n	5			

1. Inicializa la variable n en cinco.
2. Inicializa la variable i en cero.
3. Evalúa la condición i < n, cero es menor que cinco, 0 < 5. Si es verdad la condición inicia el bucle, se puede identificar el inicio del bucle con las llaves "{". En caso contrario que no cumpla con el bucle se salta hasta el final del bucle identificado con una llaves de cierre "}" y continua el flujo del código de manera descendente. Es verdad la condición (0 < 5) por lo tanto entra al bucle.
4. Imprime i +1, esto es cero mas uno, 0 + 1 = 1, entonces imprime uno, pero este valor no modifica el valor de la variable i, solo despliega la suma de i + 1 en pantalla. También despliega en pantalla "Hola…"
5. La variable i incrementa en una posición su valor con la instrucción i++. Entonces la variable i en vez de almacenar el valor de cero ahora tiene uno.

   Es como si se expresara de la siguiente manera la siguiente instrucción i++ de la siguiente forma:

   i = i + 1, i almacena el valor de cero mas uno, esto es uno, y después se asigna el valor de la suma en la variable que se encuentra de lado izquierdo del operador de asignación "=".
6. Termina el bucle al encontrar las llaves de cierre "}" y regresa al punto tres (3).

Prueba de escritorio: segunda iteración.

i	0	1	2
n	5		

3. Evalúa la condición i < n, uno es menor que cinco, 1<5. Si es verdad la condición inicia el bucle. En caso contrario que no cumpla con el bucle se salta hasta el final del bucle y continua el flujo del código de manera descendente. Es verdad la condición (1 < 5) por lo tanto entra al bucle.
4. Imprime i +1, esto es uno mas uno, 1 + 1 = 2 ,entonces imprime dos, pero este valor no modifica el valor de la variable i, solo despliega la suma de i + 1 en pantalla. También despliega en pantalla “Hola…”
5. La variable i incrementa en una posición su valor con la instrucción i++. Entonces la variable i en vez de almacenar el valor de uno ahora tiene dos.
6. Termina el bucle al encontrar las llaves de cierre “}” y regresa al punto tres (3).

Prueba de escritorio: tercera iteración.

i	0	1	2	3	
n	5				

3. Evalúa la condición i < n, dos es menor que cinco, 2<5. Si es verdad la condición inicia el bucle. En caso contrario que no cumpla con el bucle se salta hasta el final del bucle y continua el flujo del código de manera descendente. Es verdad la condición (2<5), por lo tanto entra al bucle.
4. Imprime i + 1, esto es dos mas uno, 2 + 1 = 3, entonces imprime tres, pero este valor no modifica el valor de la variable i, solo despliega la suma de i + 1 en pantalla. También despliega en pantalla “Hola…”
5. La variable i incrementa en una posición su valor con la instrucción i++. Entonces la variable i en vez de almacenar el valor de dos ahora tiene tres.
6. Termina el bucle al encontrar las llaves de cierre “}” y regresa al punto tres (3).

Prueba de escritorio: cuarta iteración

i	0	1	2	3	4
n	5				

3. Evalúa la condición i < n, tres es menor que cinco, 3<5. Es verdad la condición (3<5), por lo tanto entra al bucle.
4. Imprime i + 1, esto es tres mas uno, 3 + 1 = 4, entonces imprime cuatro. También despliega en pantalla “Hola…”
5. La variable i incrementa en una posición su valor con la instrucción i++. Entonces la variable i en vez de almacenar el valor de tres ahora tiene cuatro.
6. Termina el bucle al encontrar las llaves de cierre “}” y regresa al punto tres (3).

Prueba de escritorio: quinta iteración

i	0	1	2	3	4	5
n	5					

3. Evalúa la condición i < n, cuatro es menor que cinco, 4<5. Es verdad la condición (4<5), por lo tanto entra al bucle.
4. Imprime i + 1, esto es cuatro mas uno, 4 + 1 = 5, entonces imprime cinco. También despliega en pantalla “Hola…”
5. La variable i incrementa en una posición su valor con la instrucción i++. Entonces la variable i en vez de almacenar el valor de cuatro ahora tiene cinco.
6. Termina el bucle al encontrar las llaves de cierre “}” y regresa al punto tres (3).

Prueba de escritorio: sexta iteración

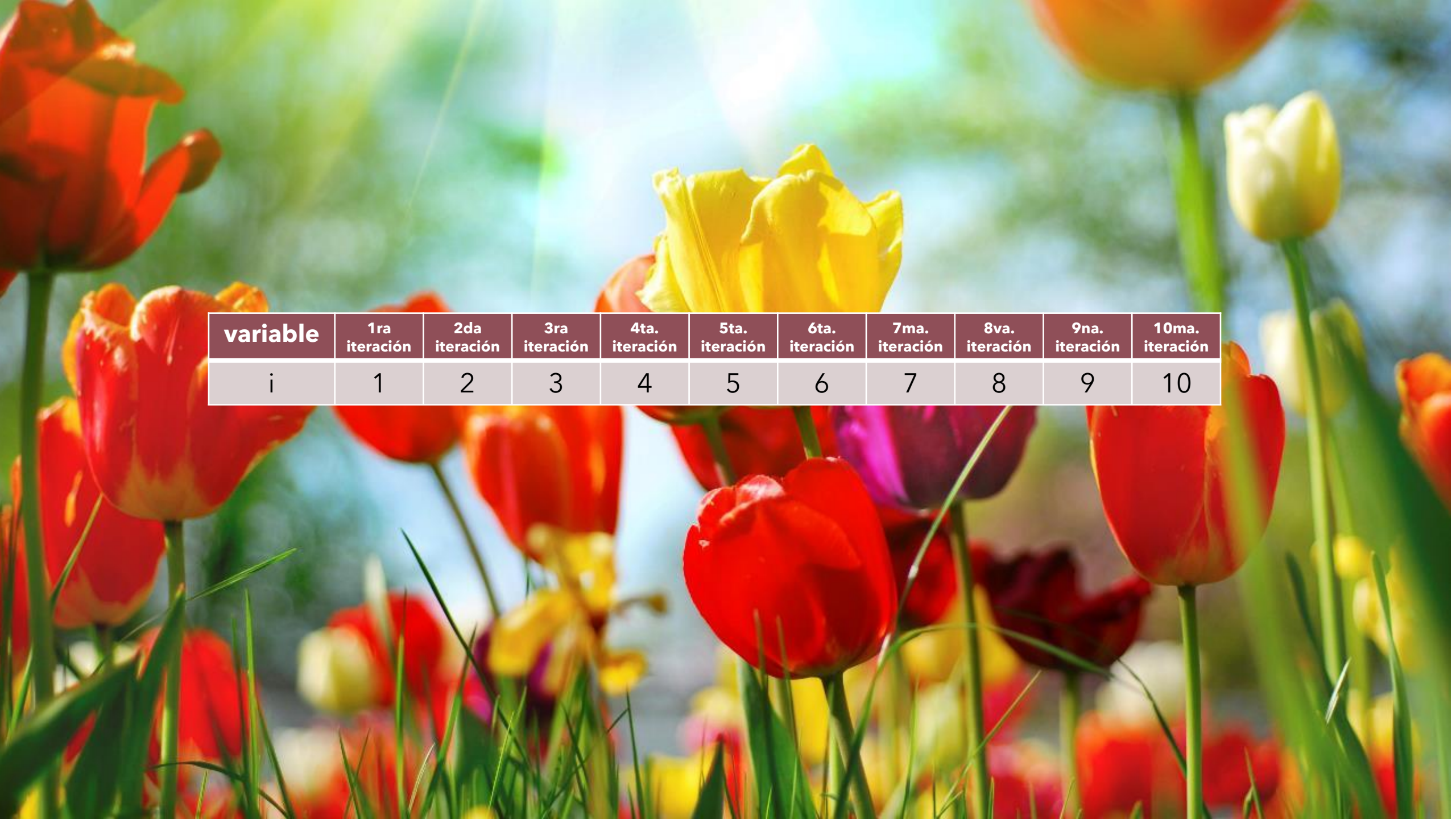
i	0	1	2	3	4	5
n	5					

- Evalúa la condición i < n, cinco es menor que cinco, 5<5. Es falsa la condición (5<5), por lo tanto no puede entrar al bucle, el programa se salta al final del bucle, esto lo podemos ubicar donde se encuentra las llaves de cierre del bucle "}" y continua con el flujo del programa.

Ejercicio1. Prueba de escritorio estructura for

- Realiza la prueba de escritorio del siguiente código.

```
import java.util.Scanner;
public class TablaDeMultiplicar
{
    public static void main(String[] args){
        int numero;
        Scanner leer = new Scanner(System.in);
        System.out.println("¿Cuál es la tabla que deseas desplegar?");
        numero = leer.nextInt();
        for(int i=1; i<=10; i++){
            System.out.println(numero + " x " + i + " = " + numero*i + "\n");
        }
    }
}
```



variable	1ra iteración	2da iteración	3ra iteración	4ta. iteración	5ta. iteración	6ta. iteración	7ma. iteración	8va. iteración	9na. iteración	10ma. iteración
i	1	2	3	4	5	6	7	8	9	10

1ra. iteración

Variable	1ra. iteración
i	1

2da. iteración

Variable	1ra.	2da.
i	1	2

3ra. iteración

Variable	1ra.	2da.	3ra.
i	1	2	3

4ta. iteración

Variable	1ra.	2da.	3ra.	4ta.
i	1	2	3	4

5ta. iteración

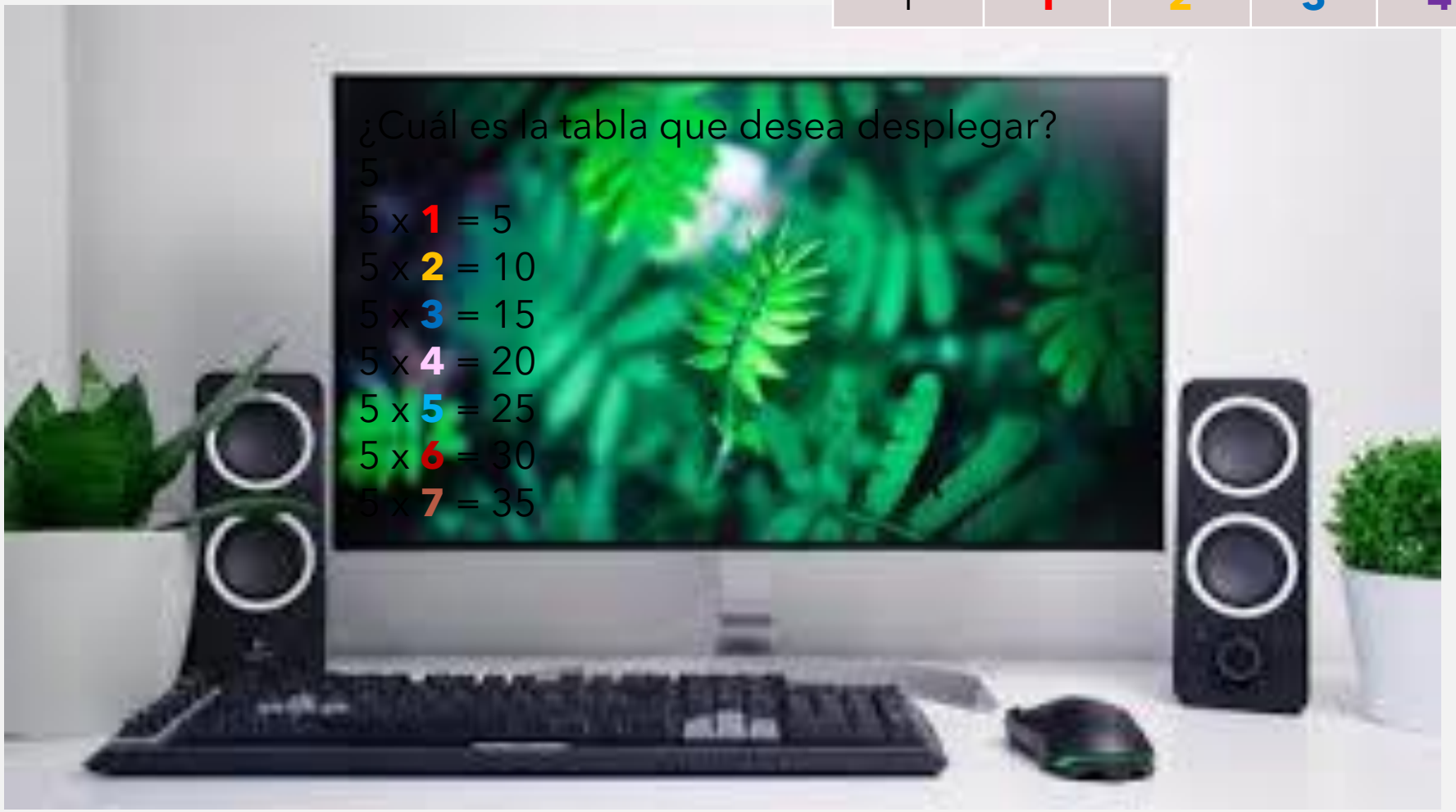
Variable	1ra.	2da.	3ra.	4ta.	5ta.
i	1	2	3	4	5

6ta. iteración

Variable	1ra.	2da.	3ra.	4ta.	5ta.	6ta.
i	1	2	3	4	5	6

7ma. iteración

Variable	1ra.	2da.	3ra.	4ta.	5ta.	6ta.	7ma.
i	1	2	3	4	5	6	7

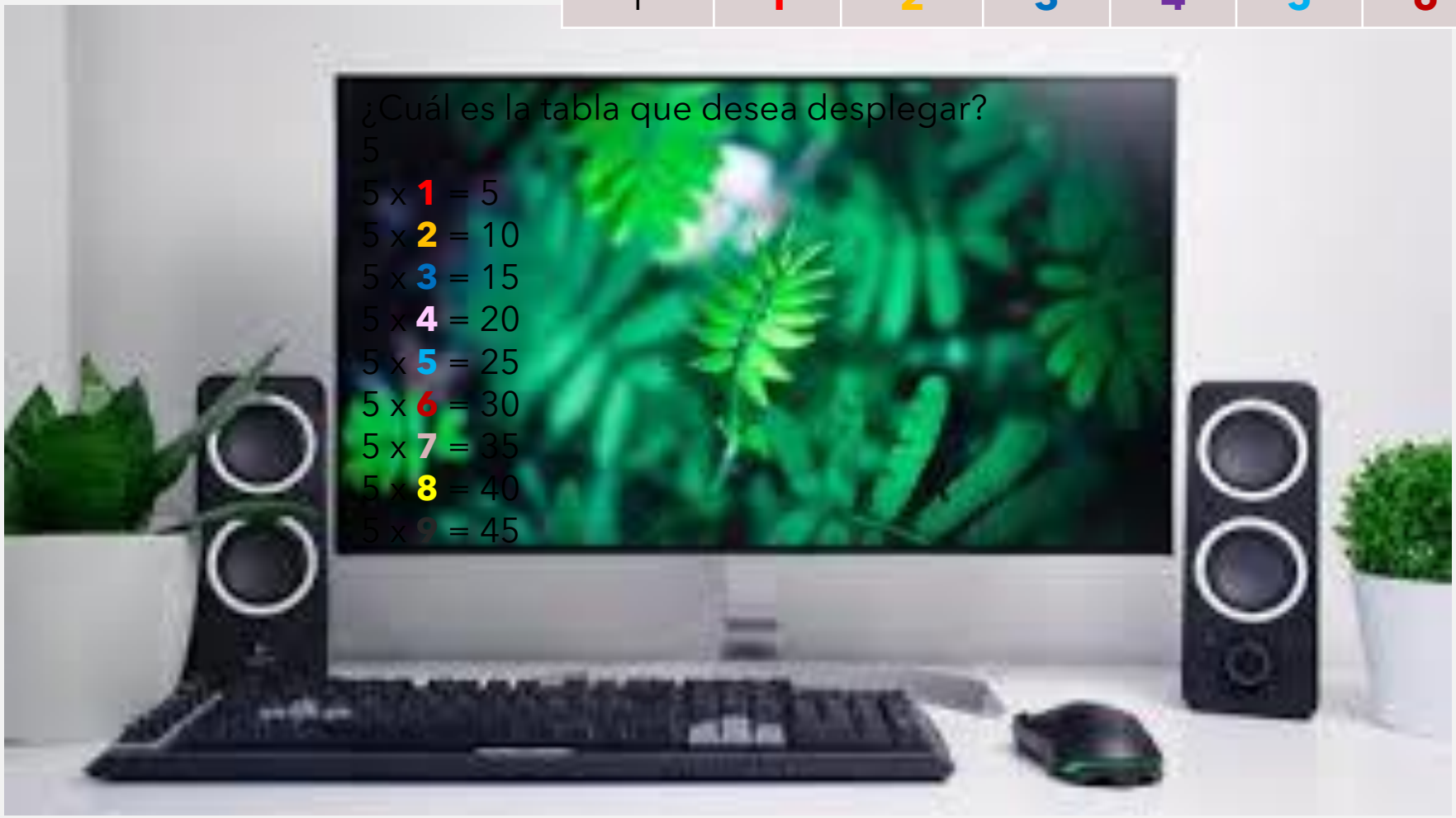


8va. iteración

Variable	1ra.	2da.	3ra.	4ta.	5ta.	6ta.	7ma.	8va.
i	1	2	3	4	5	6	7	8

9na. iteración

Variable	1ra.	2da.	3ra.	4ta.	5ta.	6ta.	7ma.	8va.	9na.
i	1	2	3	4	5	6	7	8	9



10ma. iteración

Variable	1ra.	2da.	3ra.	4ta.	5ta.	6ta.	7ma.	8va.	9na.	10ma.
i	1	2	3	4	5	6	7	8	9	10