# 

# Trabajo Práctico N°1: Operadores Metodología de Programación

## Ejercicios:

**Ejercicio 1**: Evaluar(obtener resultado) la siguiente expresión para A = 2 y B = 5

3\* A - 4 \* B / A ^ 2

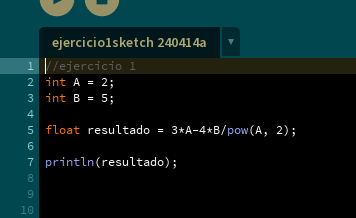
Resolución necesaria en Word:

(3\*A)-(4\*B/(A^2))

6-(4\*B/4)

6-5

1



**Ejercicio 2**: Evaluar la siguiente expresión 4 / 2 \* 3 / 6 + 6 / 2 / 1 / 5 ^ 2 / 4 \* 2

(4/2\*3/6)+(6/2/1/(5^2)/4\*2)

(2\*3/6)+(3/1/25/4\*2)

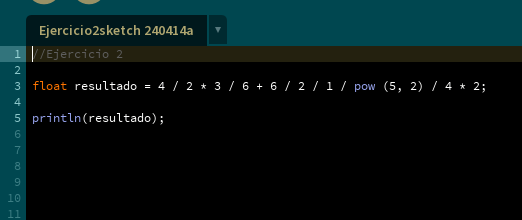
(6/6)+(3/25/4\*2)

1+(0.12/4\*2)

1+(0.03\*2)

1+0.06

1.06



**Ejercicio 4**: Evaluar las siguientes expresiones aritméticas, para lo cual indicar en el caso de las variables, el valor indicado. Luego escribirlas como expresiones algebraicas.

a = 1, b = 2, c = 3, d = 4, X = 5, y = 6, x = 7

1. (b^2)–(4\*a\*c)

4-(4\*c)

4-12

-8

1. (3 \*( X ^ 4))–(5 \* (X ^ 3) + X \*12)– 17

(3\*625)-((5\*125)+(5\*12))-17

1875-(625+60)-17

1875-685-17

1190-17

1173

1. (b + d) / (c + 4)

6/7

0.85

1. ((x ^ 2) +( y ^ 2)) ^ (1 / 2)

(49+36) ^0.5

85^0.5

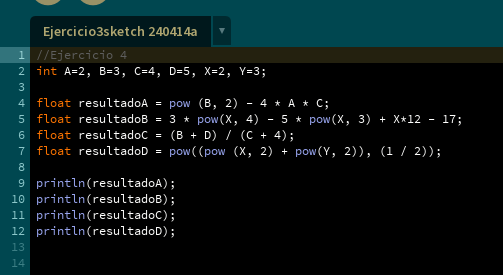
9.21

a)

b)

c)

d)



**Ejercicio 5:** Si el valor de A es 4, el valor de B es 5 y el valor de C es 1, evaluar las siguientes expresiones:

1. (B\*A)–(B^(2/4\*C))

20-(B^(0.5\*C))

20-(B^0.5)

20-2.23

17,76

1. (A\*B)/(3^2)

20/9

2.22

1. ((((B+C)/2\*A)+10)\*3\*B)–6  
   (((6/2\*A)+10)\*3\*B)-6

(((3\*A)+10)\*3\*B)-6

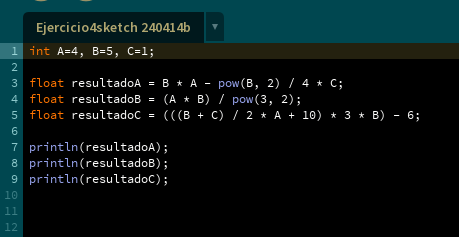
((12+10)\*3\*B)-6

(22\*3\*B)-6

(66\*B)-6

330-6

324



**Ejercicio 6:** Para x=3, y=4; z=1, evaluar el resultado de

R1 = y+z

R1 = 5

R2 = x >= R1

R2 = (3 >= 5)

R2= false



**Ejercicio 7**: Para contador1=3, contador2=4, evaluar el resultado de

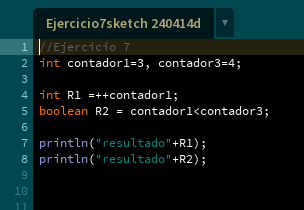
R1 = ++contador1

R1= 4

R2 = contador1 < contador2

R2= 4<4

R2=false



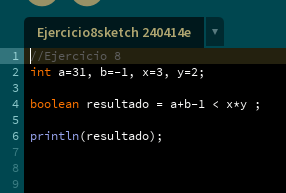
**Ejercicio 8:** Para a=31, b=-1; x=3, y=2, evaluar el resultado de

a+b-1 < x\*y

31-1<6

30<6

false

****

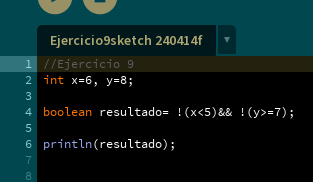
**Ejercicio 9:** Para x=6, y=8, evaluar el resultado de

!(x<5)&& !(y>=7)

!(false) and !(true)

(true) and (false)

False

****

**Ejercicio 10**: Para i=22,j=3, evaluar el resultado de

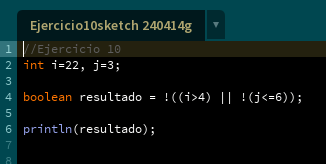
!((i>4) || !(j<=6))

!(true or !(true))

!(true or false)

!(true)

false

****

**Ejercicio 11:** Para a=34, b=12,c=8, evaluar el resultado de

!(a+b==c) || (c!=0)&&(b-c>=19)

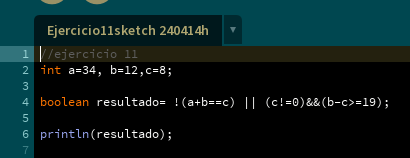
!(46==c) or (true) and (4>=19)

!(false) or (true) and (false)

true or (true and false)

true OR false

true



**Ejercicio 12**:

Definicion del problema: mostrar un saludo con el nombre del usuario

Analisis:

* Datos de Entrada:

-nombre del usuario: string

* Datos de Salida:

-saludo con nombre del usuario: string

* Proceso:

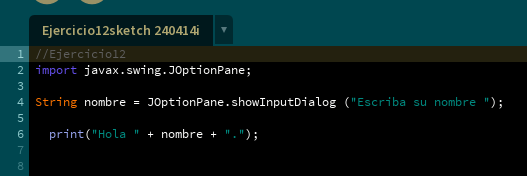
¿Quién debe realizar el proceso?: el usuario con el programa

¿Cuál es el proceso que realiza?:

Recibir el nombre del usuario para realizar un saludo con su nombre

**Diseño:**

| Entidad: usuario |
| --- |
| Variables:   * nombreUsuario: string * saludo: string |
| Nombre de Algoritmo: saludar\_usuario  Proceso del Algoritmo:   1. Inicio 2. Importar JOptionPane del paquete Java 3. *Leer* nombre 4. *Mostrar* saludo en (“¡Hola”+nombre+”!”) 5. fin |



**Ejercicio 13**:

Definicion del problema: calcular el primetro y área de un rectangulo

Analisis:

* Datos de Entrada:

-base: int

-altura: int

* Datos de Salida:

-perimetro: int

-area: int

* Proceso:

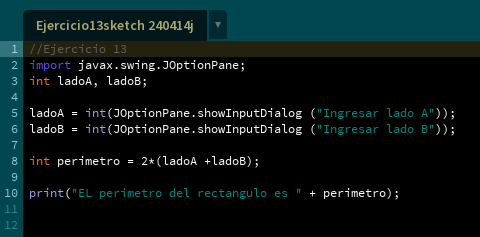
¿Quién debe realizar el proceso?: el programa

¿Cuál es el proceso que realiza?:

Calcular el perímetro y área de un rectángulo con la base y altura dadas

**Diseño:**

| Entidad: programa |
| --- |
| Variables:   * base: int * altura: int * perímetro: int * area: int |
| Nombre de Algoritmo: calcular\_rectangulo  Proceso del Algoritmo:   1. Inicio 2. importar JOptionPane del paquete Java 3. *Leer* base 4. *Leer* altura 5. *Mostrar* perímetro del rectángulo igual a (base\*2+altura\*2) 6. *Mostrar* area del rectángulo igual a (base\*altura) 7. fin |



**Ejercicio 14**:

Definicion del problema: obtener hipotenusa de un triangulo

Analisis:

* Datos de Entrada:

-cateto1: int

-cateto2: int

* Datos de Salida:

-hipotenusa: float

* Proceso:

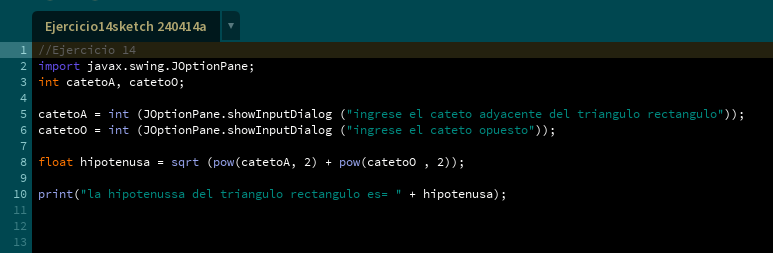
¿Quién debe realizar el proceso?: el programa

¿Cuál es el proceso que realiza?:

Calcular la hipotenusa de un triangulo con sus dos catetos dados

**Diseño:**

| Entidad: programa |
| --- |
| Variables:   * cateto1: int * cateto2: int * hipotenusa: float |
| Nombre de Algoritmo: calcular\_hipotenusa  Proceso del Algoritmo:   1. Inicio 2. importar JOptionPane del paquete Java 3. *Leer* cateto1 4. *Leer* cateto2 5. hipotenusa🡨 raiz de (cateto1^2 + cateto2^2) 6. *Mostrar* hipotenusa del triangulo 7. fin |



**Ejercicio 15:**

Definicion del problema: calcular la suma, resta, multiplicacion y división de dos números dados

Analisis:

* Datos de Entrada:

-numeroA: int

-numeroB: int

* Datos de Salida:

-resultadoSuma: float

-resultafoResta: float

-resultadoMulti: float

-resultadoDiv: float

* Proceso:

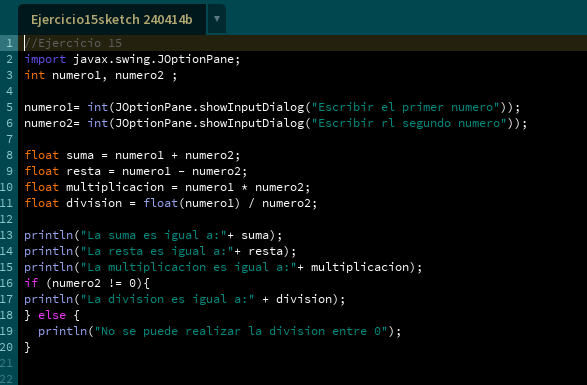
¿Quién debe realizar el proceso?: el programa

¿Cuál es el proceso que realiza?:

Calcular la suma, resta, multiplicación y división de dos números dados

**Diseño:**

| Entidad: programa |
| --- |
| Variables:   * numeroA: int * numeroB: int * resultadoSuma: float * resultafoResta: float * resultadoMulti: float * resultadoDiv: float |
| Nombre de Algoritmo: calcular\_resultado  Proceso del Algoritmo:   1. Inicio 2. importar JOptionPane del paquete Java 3. *Leer* numeroA 4. *Leer* numeroB 5. resultadoSuma🡨 (numeroA+numeroB) 6. resultafoResta🡨 (numeroA-numeroB) 7. resultadoMulti🡨 (numeroA\*numeroB) 8. resultadoDiv🡨 (numeroA/numeroB) 9. *Mostrar* resultadoSuma 10. *Mostrar* resultadoResta 11. *Mostrar* resultadoMulti 12. *Mostrar* resultadoDiv 13. fin |



**Ejercicio 16:**

Definicion del problema: convertir temperatura Fahrenheit en grados Celsius

Analisis:

* Datos de Entrada:

-gradoFah: int

* Datos de Salida:

-gradoCel: float

* Proceso:

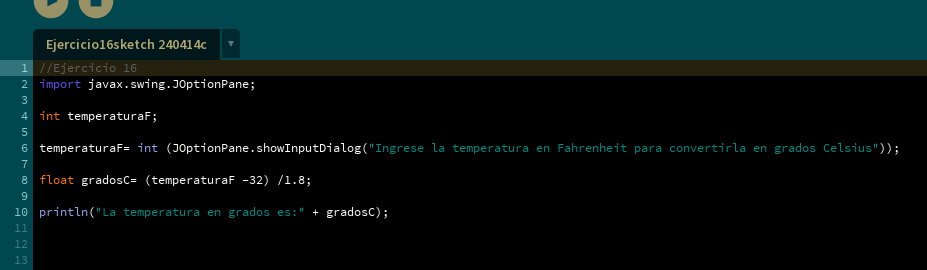
¿Quién debe realizar el proceso?: el programa

¿Cuál es el proceso que realiza?:

Convertir los grados Fahrenheit dados en grados Celsius

**Diseño:**

| Entidad: programa |
| --- |
| Variables:   * gradoFah: int * gradoCel: float |
| Nombre de Algoritmo: transformar\_celsius  Proceso del Algoritmo:   1. inicio 2. importar JOptionPane del paquete Java 3. *Leer* gradoFah 4. gradoCel🡨 (gradoFah-32) / 1.8 5. *Mostrar* gradoCel 6. fin |



**Ejercicio 17**:

Definicion del problema: calcular la distancia entre Link y la caja

Analisis:

* Datos de Entrada:

-ancho, alto de Lienzo: int

-coordenadasLink: coordenadas cartesianas

-coordenadasCaja: coordenadas cartesianas

-ancho, alto de Link: int

-ancho, alto de Caja: int

-catetoA, catetoO: coordenadas cartesianas

-colorLink: color

-colorCaja: color

* Datos de Salida:

-distanciaLinkCaja: float

* Proceso:

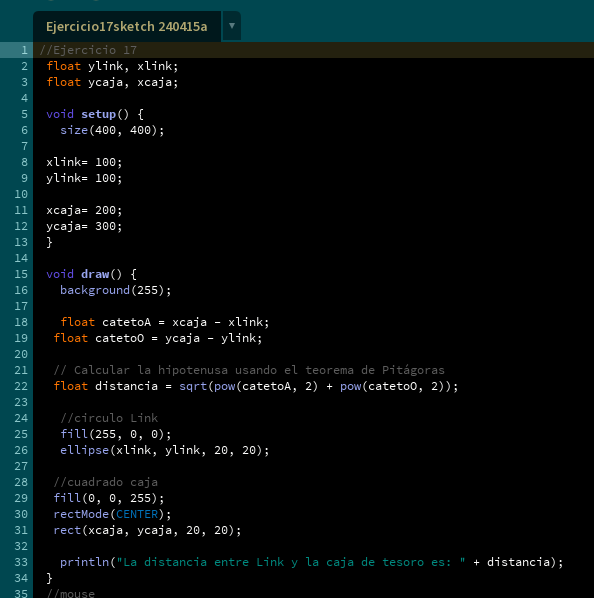
¿Quién debe realizar el proceso?: el programa

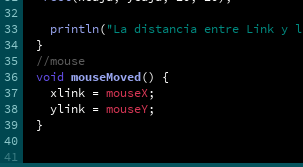
¿Cuál es el proceso que realiza?:

Colocar en un lienzo un cuadrado estatico y un circulo controlado por el mouse que dependiendo la posición de este se calcule la distancia entre ambos

**Diseño:**

| Entidad: programa |
| --- |
| Variables:   * anchoLienzo, altoLienzo: int * coordenadasLink: coordenadas cartesianas * coordenadasCaja: coordenadas cartesianas * anchoLink, altoLink: int * anchoLink, altoCaja: int * catetoA, catetoO: coordenadas cartesianas * colorLink: color * colorCaja: color * distanciaLinkCaja: float |
| Nombre de Algoritmo: calcular\_distanciaLinkCaja  Proceso del Algoritmo:   1. inicio 2. anchoLienzo🡨 400 3. altoLienzo🡨 400 4. coordenadaXCaja🡨 200 5. coordenadaYCaja🡨 300 6. altoLink🡨20 7. anchoLink🡨20 8. altoCaja🡨20 9. anchoCaja🡨20 10. catetoA🡨 xCaja – xLink 11. catetoB🡨 yCaja – yLink 12. distanciaLinkCaja🡨 raiz de (catetoA^2 + catetoB^2) 13. *Leer* coordenadasLink 14. *Mostrar* distanciaLinkCaja |





**Ejercicio 18:**

Definicion del problema: crear un algoritmo que resuelva raíces de ecuaciones de segundo grado y analize su discriminante

Analisis:

* Datos de Entrada:

-numeroA: float

-numeroB: float

-numeroC:float

* Datos de Salida:

-resultadoEcua:

-discriminante:

* Proceso:

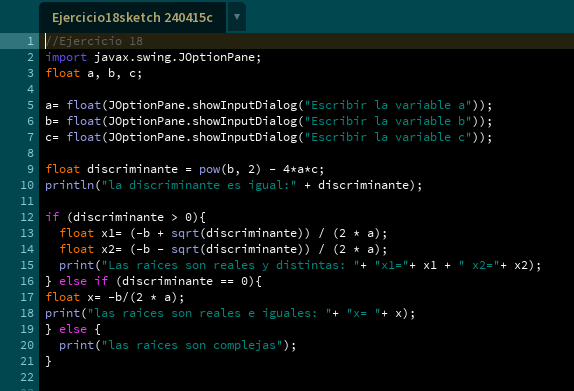
¿Quién debe realizar el proceso?: el programa

¿Cuál es el proceso que realiza?:

Calcular la ecuación de segundo grado dada y analizar su discriminante

**Diseño:**

| Entidad: programa |
| --- |
| Variables:   * numeroA: float * numeroB: float * numeroC: float * resultafoEcua: float * discriminante: float * x1, x2, x; float |
| Nombre de Algoritmo: calcular\_ecuacion  Proceso del Algoritmo:   1. Inicio 2. importar JOptionPane del paquete Java 3. Leer numeroA 4. Leer numeroB 5. Leer numeroC 6. discriminante🡨 numeroB^2- 4\*numeroA\*numeroC 7. si la discriminante es mayor a cero se realizara 8. x1🡨(-numeroB+raíz del(discriminante))/(2\*a) 9. x2🡨(-numeroB-raíz del(discriminante))/(2\*a) 10. Mostrar las raíces reales y distintas x1, x2 11. si la discriminante es igual a cero se realizara 12. x🡨 -numeroB/(2\*a) 13. Mostrar la raíz real e igual x 14. si no cumple ninguna de las condiciones anteriores se realizara 15. Mostrar las raíces son complejas 16. fin |



**Ejercicio 19:**

Definicion del problema: dibujar una línea que toque la parte superior de un circulo y hacer que los dos se muevan juntos indefinidamente de arriba a abajo de los bordes del lienzo

Analisis:

* Datos de Entrada:

- anchoLienzo, altoLienzo: int

- yLlinea, yElipse, velocidad: float

- direccionLinea, direcciónElipse: int

* Datos de Salida:

-movimiento de circulo con linea

* Proceso:

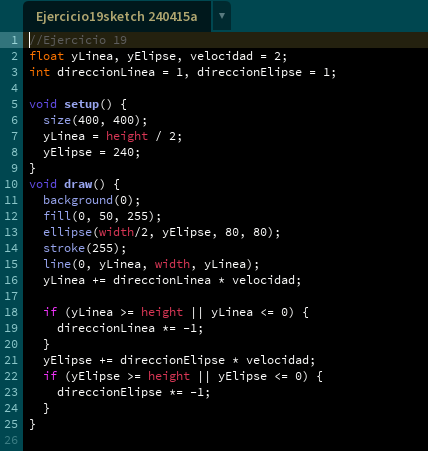
¿Quién debe realizar el proceso?: el programa

¿Cuál es el proceso que realiza?:

En un lienzo muestra una línea y un circulo a los cuales se les otorga cierta velocidad y la capacidad de rebotar entre los limites del lienzo

**Diseño:**

| Entidad: programa |
| --- |
| Variables:   * anchoLienzo, altoLienzo: int * yLlinea, yElipse, velocidad: float * direccionLinea, direcciónElipse: int |
| Nombre de Algoritmo: rebotar\_circulo\_linea   1. Proceso del Algoritmo: 2. Inicio 3. velocidad🡨2 4. direccionLinea🡨 1 5. direccionElipse🡨 1 6. yElipse🡨 240 7. yLinea🡨 altura de elipse/2 8. coordenada inicio para la elipse respecto a y 9. coordenada inicio para la linea respecto a y 10. velocidad por la que se moverá la línea por el eje y 11. si la línea es igual o mayor a la altura del lienzo rebotara inviertiendo su movimiento a -1 12. si la elipse es igual o mayor a la altura del lienzo rebotara inviertiendo su movimiento a -1 |



**Ejercicio 20:**

Definicion del problema: dibujar una serie de rectángulos idénticos en un lienzo

Analisis:

* Datos de Entrada:

- coordenadasRect: coordenadas cartesianas

-ancho, alto, distanciaEntreRect: enteros

-anchoLienzo, altoLienzo: enteros

-rect\_color: color

* Datos de Salida:

-rectangulos dibujados

* Proceso:

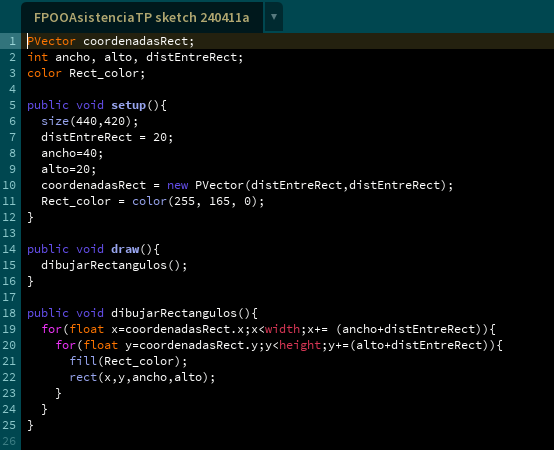
¿Quién debe realizar el proceso?: el programa

¿Cuál es el proceso que realiza?:

Dibujar una serie de rectángulos con determinado espaciado y color entre ellos en un lienzo con un determinado tamaño

**Diseño:**

| Entidad: programa |
| --- |
| Variables:   * coordenadasRect: coordenadas * ancho, alto, distanciaEntreRect: enteros * anchoLienzo, altoLienzo: enteros * rect\_color: color |
| Nombre de Algoritmo: dibujar\_rectangulos  Proceso del Algoritmo:   1. danchoLienzo← 440 2. altoLienzo← 420 3. distanciaEntreRect←20 4. ancho←40 5. alto← 20 6. color←color(255, 165, 0) 7. para x← coordenadasRect.x hasta anchoLienzo con paso (ancho+distanciaEntreRect) 8. hacer 9. para y←coordenadasRect.y hasta altoLienzo con paso (alto+distanciaEntreRect) 10. hacer 11. rellenar con color los rectangulos 12. dibujar un rectángulo en (x.coordenadasRect.y) con dimensiones ancho y alto 13. fin para 14. fin para |



**Ejercicio 21:**

Definicion del problema: dibujar una serie de escalones donde en cada borde hay un punto rojo

Analisis:

* Datos de Entrada: coordenadas cartesianas en 2D

-puntoA

-puntoB

-puntoC

-puntoD

* Datos de Salida:

-EL dibujo en la línea horizontal

-El dibujo en la línea vertical

-El dibujo del punto rojo

* Proceso:

Dibujar una linea horizontal entre los puntos A y B, con distancia igual a distLinea

Dibujar una linea vertical entre los puntos B y C, con distancia igual a distLinea

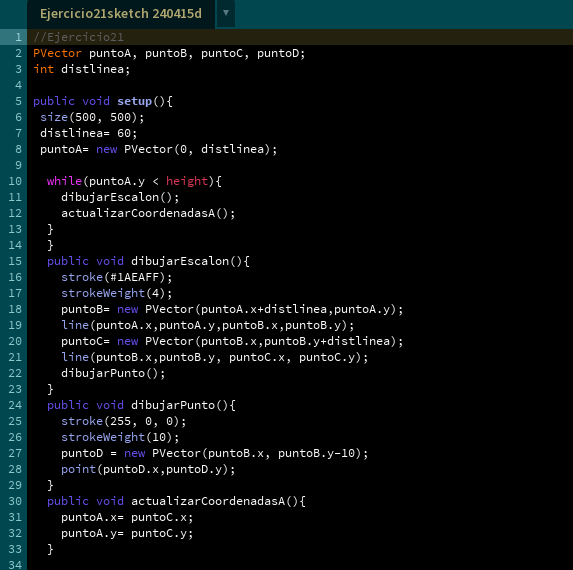
Dibujar un punto en la siguiente posicion: x=posicion en x de B, y =posición en y de B -5 unidades

Actualizar las coordenadas de punta con las de puntoC

Repetir desde el principio hasta que la coordenada en y de puntoA sea mayor que el alto del lienzo

**Diseño:**

| Entidad: Escalon |
| --- |
| Variables: coordenadas cartesianas en 2D   * puntoA * puntoB * puntoC * puntoD * distLinea: entero |
| Nombre de Algoritmo: dibujar\_escalon  Proceso del Algoritmo:   1. inicio 2. dibujar una linea horizontal entre los puntos A y B, con distancia distLinea 3. dibujar una linea vertical entre los puntos B y C, con distancia distLinea 4. dibujar\_circulo 5. fin |
| Nombre algoritmo: dibujar\_circulo  Proceso del Algoritmo:   1. inicio 2. dibujar un punto en la siguiente posicion: x= posicion en x de B, y = posicion en y de B - 10 3. fin |



**Ejercicio 22:**

Se divide el lienzo en franjas de igual medida, se deben dibujar los círculos sobre cada línea de por medio es decir en la línea 1 se dibujan círculos con distanciamiento, en la línea 2 no se dibuja y así sucesivamente. Las líneas tienen un color fijo, los círculos asumen colores aleatorios.Datos de entrada: coordenadasRect: coordenadas cartesianas.

Analisis:

* Datos de Entrada:

- coordenadasRect: coordenadas cartesianas

-ancho, alto, distanciaEntreRect: enteros

-anchoLienzo, altoLienzo: enteros

-rect\_color: color

* Datos de Salida:

-lineas con círculos dibujado

* Proceso:

¿Quién debe realizar el proceso?: el programa

¿Cuál es el proceso que realiza?:

Dibujar una serie de de 6 lineas de manera horizontal a través del lienzo, de las cuales solamente en las líneas impares tendrán círculos de colores arriba suyo

**Diseño:**

| Entidad: programa |
| --- |
| Variables:   * coordenadasRect: coordenadas * ancho, alto, distanciaEntreRect: enteros * anchoLienzo, altoLienzo: enteros * rect\_color: color |
| Nombre de Algoritmo: dibujar\_lineas\_pelotas  Proceso del Algoritmo:   1. danchoLienzo←440 2. altoLienzo← 420 3. distanciaEntreRect←20 4. ancho←40 5. alto←20 6. color←color(255, 165, 0) 7. para x ← coordenadasRect.x hasta anchoLienzo con paso (ancho+distanciaEntreRect) 8. hacer 9. para y←coordenadasRect.y hasta altoLienzo con paso (alto+distanciaEntreRect) 10. hacer 11. rellenar con color los rectangulos 12. dibujar un rectángulo en (x.coordenadasRect.y) con dimensiones ancho y alto 13. fin para 14. fin para |

