**FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS (TUDIVJ)**

**Trabajo Práctico 1**

**Alumno:**

Gonzalez Ariki Sebastian

DNI - 47005600

LU - TUV000452

**Profesor:**

Mg. Ing. Ariel Alejandro Vega

[**Ejercicio 1 3**](#_4z8ukkdc7zz2)

[**Ejercicio 2 4**](#_6rjf3ae6np1x)

[**Ejercicio 3 4**](#_w57ypu7l1j2j)

[**Ejercicio 4 5**](#_e8p34ay20fc2)

[**Ejercicio 5 7**](#_nw13wrmgcv9h)

[**Ejercicio 6 8**](#_j931xdezey9x)

[**Ejercicio 7 9**](#_8bse0l5o7j4v)

[**Ejercicio 8 10**](#_5f3faecdn9h)

[**Ejercicio 9 10**](#_gajiicps9n8g)

[**Ejercicio 10 11**](#_of2475vao3w8)

[**Ejercicio 11 12**](#_qvaz2z5dqsg8)

[**Ejercicio 12 12**](#_ya347b81tetg)

[**Ejercicio 13 13**](#_1mh2sne3tyok)

[**Ejercicio 14 15**](#_3md6l1snjlf)

[**Ejercicio 15 16**](#_c63u6p5hmy87)

[**Ejercicio 16 18**](#_4x5advob0t8j)

[**Ejercicio 17 19**](#_gxy67mblcppm)

[**Ejercicio 18 23**](#_s4cvulnrdcaa)

[**Ejercicio 19 24**](#_7ax3m1mipzh)

[**Ejercicio 20 27**](#_67sko6gashs4)

[**Ejercicio 21 29**](#_jgqvezba7oo2)

[**Fuentes Biográficas 33**](#_sykacnyier4x)

# **Ejercicio 1**

Evaluar (obtener resultado) la siguiente expresión para A = 2 y B = 5

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

**Desarrollo del punto**

3\* 2 - 4 \* 5 / 2 ^ 2

3\* 2 - 4 \* 5 / 4

6 - 20 / 4

6 - 5

1



# **Ejercicio 2**

Evaluar la siguiente expresión

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

**Desarrollo del punto**

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

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

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

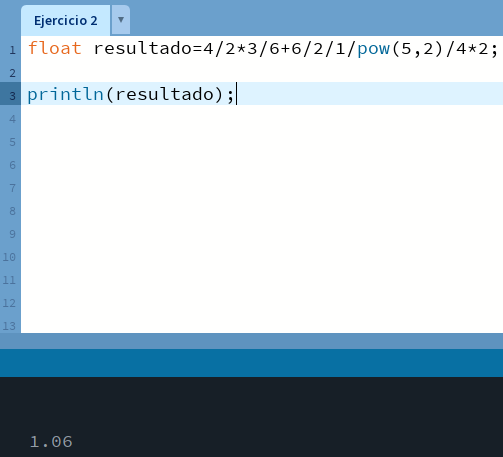
6 / 6 + 3 / 25 / 4 \* 2

1 + 0.12 / 4 \* 2

1 + 0.03 \* 2

1 + 0.06

1.06



# **Ejercicio 3**

Escribir las siguientes expresiones algebraicas como expresiones algorítmicas (en su forma aritmética dentro del algoritmo). En este caso no se pide evaluarlas ni programarlas.

**Desarrollo del punto**

??

# **Ejercicio 4**

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) b ^ 2 – 4 \* a \* c

b) 3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17

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

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

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

**Desarrollo del punto**

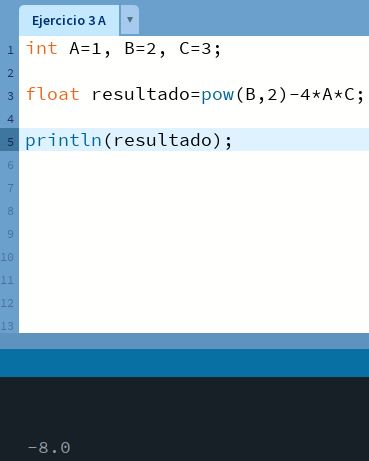
1. 2 ^ 2 – 4 \* 1 \* 3

4 – 4 \* 1 \* 3

4 – 4 \* 3

4 – 12

-8

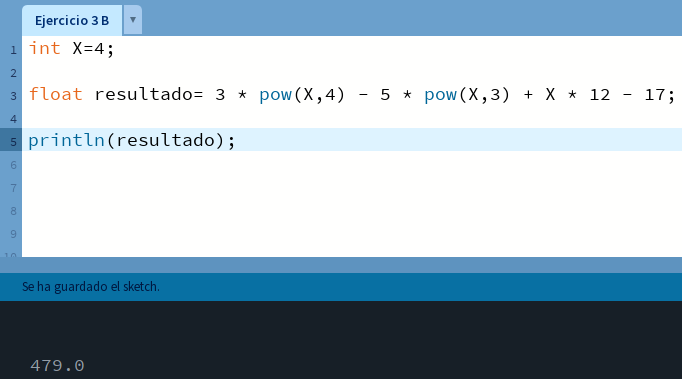


1. 3 \* 4 ^ 4 – 5 \* 4 ^ 3 + 4 \* 12 – 17

3 \* 256 – 5 \* 64 + 4 \* 12 – 17

768 – 320 + 48 – 17

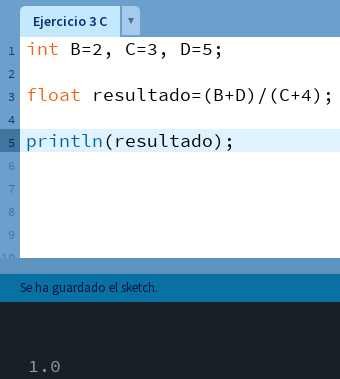
479



1. (2 + 5) / (3 + 4)

7 / 7

1

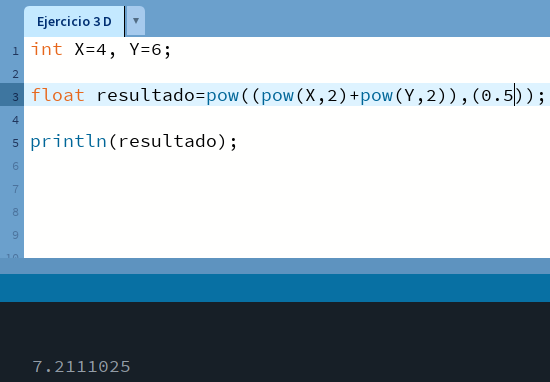
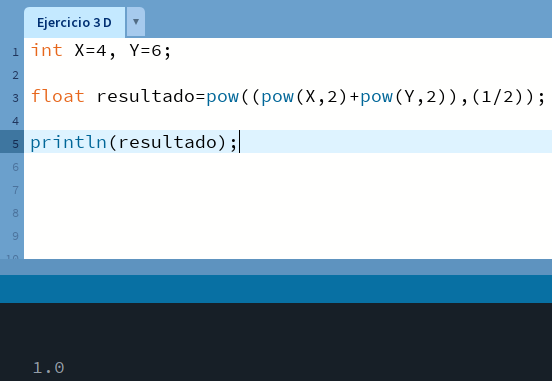


1. (4 ^ 2 + 6 ^ 2) ^ (1 / 2)

(16 + 36) ^ (1 / 2)

(52) ^ (1 / 2)

7.21



# **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:

a) B \* A – B ^ 2 / 4 \* C

b) (A \* B) / 3 ^ 2

c) (((B + C) / 2 \* A + 10) \* 3 \* B) – 6

**Desarrollo del punto**

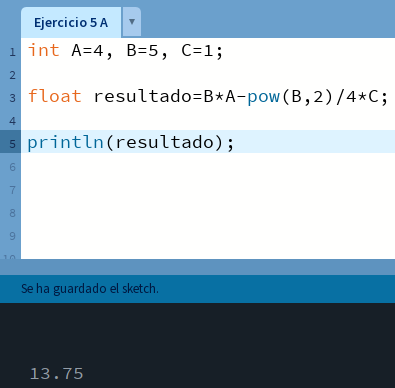
1. 5 \* 4 – 5 ^ 2 / 4 \* 1

5 \* 4 – 25 / 4 \* 1

20 – 25 / 4

20 – 6.25

13.75

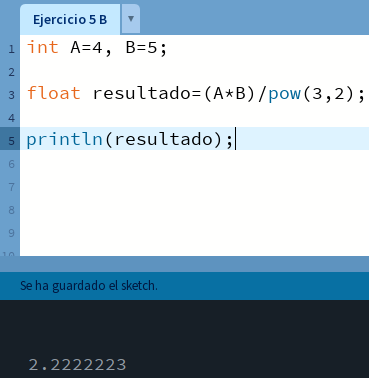


1. (4 \* 5) / 3 ^ 2

(20) / 3 ^ 2

20 / 9

2.2222



1. (((5 + 1) / 2 \* 4 + 10) \* 3 \* 5) – 6

((6 / 2 \* 4 + 10) \* 3 \* 5) – 6

((3 \* 4 + 10) \* 3 \* 5) – 6

((12 + 10) \* 3 \* 5) – 6

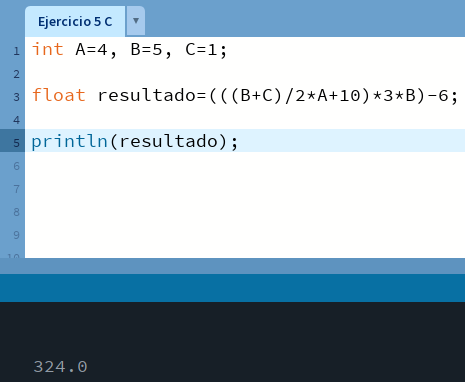
(22 \* 3 \* 5) – 6

(66 \* 5) – 6

(66 \* 5) – 6

330 – 6

324



# **Ejercicio 6**

Para x=3, y=4; z=1, evaluar el resultado de

R1 = y+z

R2 = x >= R1

**Desarrollo del punto**

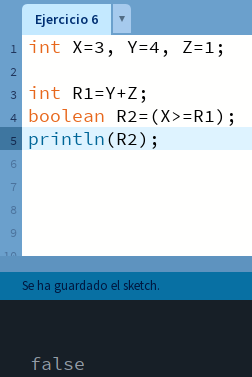
R1 = 4+1

R1 = 5

R2 = x >= R1

R2 = 3 >= 5

falso



# 

# **Ejercicio 7**

Para contador1=3, contador2=4, evaluar elresultado de

R1 = ++contador1

R2 = contador1 < contador2

**Desarrollo del punto**

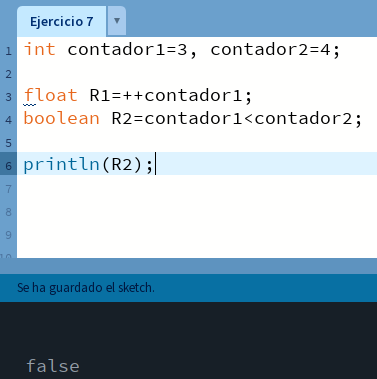
R1 = ++contador1

R1 = ++3

R2 = contador1 < contador2

R2 = 4 < 4

falso



# **Ejercicio 8**

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

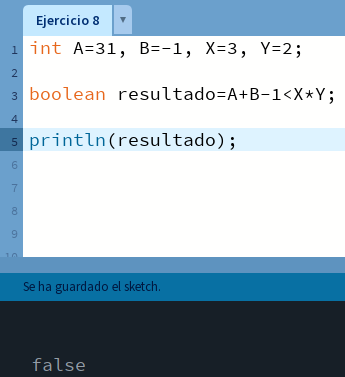
a+b-1 < x\*y

**Desarrollo del punto**

31+(-1)-1 < 3\*2

29 < 6

falso



# **Ejercicio 9**

Para x=6, y=8, evaluar el resultado de

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

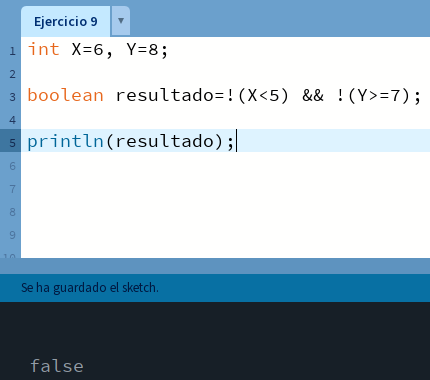
**Desarrollo del punto**

!(6<5) CC !(8>=7)

!falso CC !verdadero

verdadero CC falso

falso



# **Ejercicio 10**

Para i=22,j=3, evaluar el resultado de

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

**Desarrollo del punto**

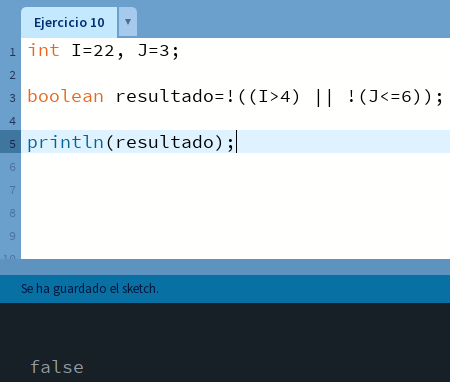
!((22>4) || !(3<=6))

!(verdadero || !verdadero)

!(verdadero || falso)

!verdadero

falso



# **Ejercicio 11**

Para a=34, b=12,c=8, evaluar el resultado de

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

**Desarrollo del punto**

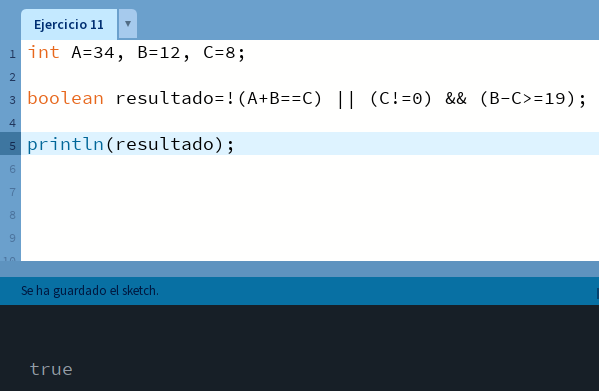
!(34+12==8) || (8!=0) CC (12-8>=19)

!(46==8) || (8!=0) CC (4>=19)

verdadero || verdadero CC falso

verdadero || falso

verdadero



# **Ejercicio 12**

Un problema sencillo. Deberá pedir por teclado al usuario un nombre y posteriormente realizará la presentación en pantalla de un saludo con el nombre indicado.

**Desarrollo del punto**

**Definición del Problema**:

Escribir en pantalla un saludo usando el nombre indicado

**Análisis**:

**Datos de Entrada**:

nombre

**Datos de Salida**:

saludo incluyendo el nombre elegido

**Proceso**:

**¿Quién debe realizar el proceso?**: Lienzo (Processing)

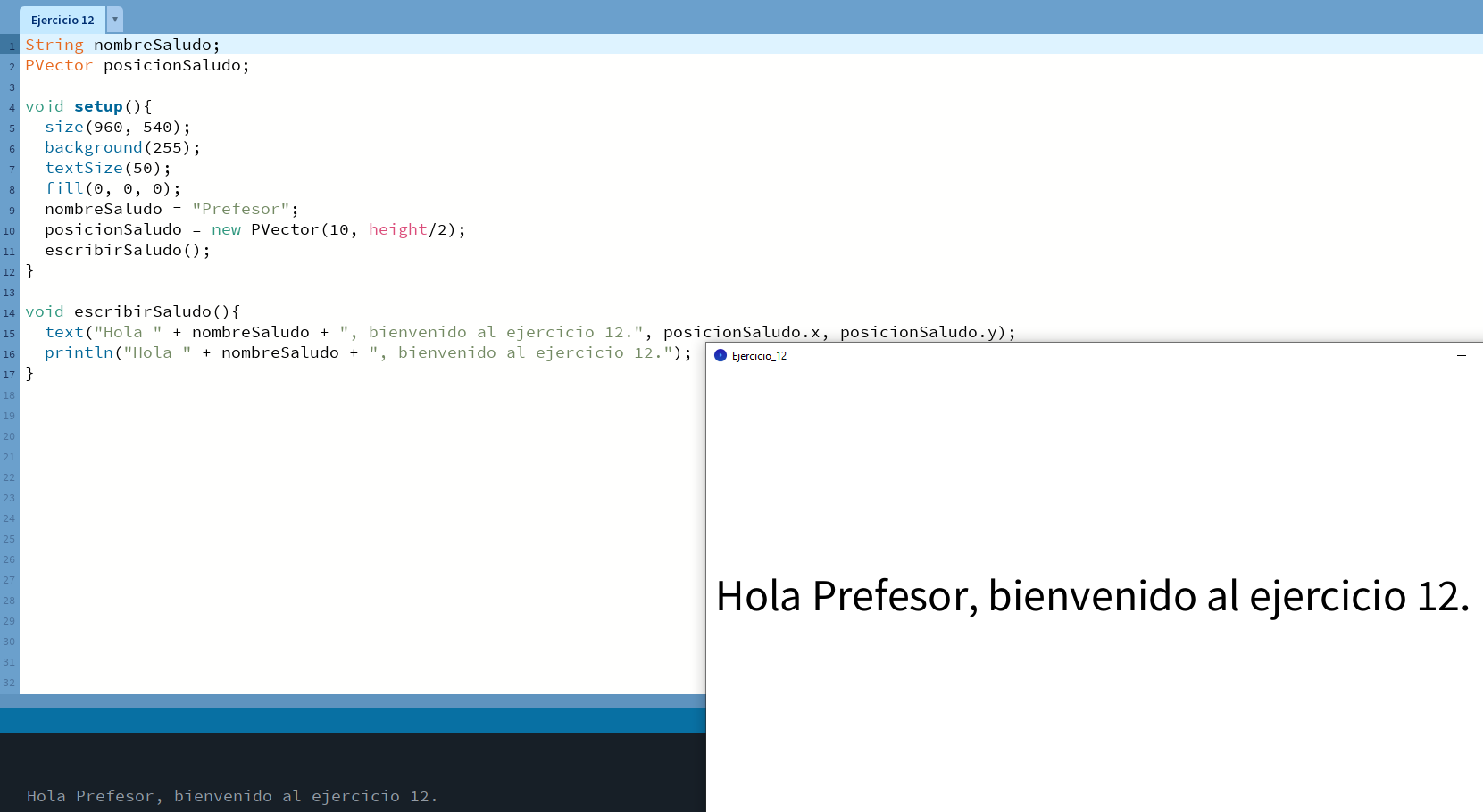
**¿Cuál es el proceso que se debe realizar?**

hacer una variable para que guarde el nombre y luego dibujar en el lienzo un texto que incluya el saludo

text(“Hola ” + nombreSaludo + “, bienvenido al ejercicio 12.”, posicionSaludo.x, posicionSaludo.y)

**Diseño**:

| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Lienzo (Processing) |
| --- |
| **VARIABLES**:  nombreSaludo: texto  posicionSaludo: coordenadas |
| **NOMBRE ALGORITMO**: escribirSaludo  **PROCESO DEL ALGORITMO**   1. *Leer* nombreSaludo 2. *Leer* posicionSaludo 3. *Escribir* (“Hola ” + nombreSaludo + “, bienvenido al ejercicio 12.”, posicionSaludo.x, posicionSaludo.y) |



# **Ejercicio 13**

Será común resolver problemas utilizando variables. Calcule el perímetro y área de un rectángulo dada su base y su altura.

**Desarrollo del punto**

**Definición del Problema**:

Calcule el perímetro y área de un rectángulo dada su base y su altura.

**Análisis**:

**Datos de Entrada**:

ancho y alto del rectángulo

**Datos de Salida**:

perimetro y area del rectangulo

**Proceso**:

**¿Quién debe realizar el proceso?**: Processing

**¿Cuál es el proceso que se debe realizar?**

para calcular el perímetro tenemos que sumar ancho y alto del rectángulo dos veces

perímetro = ancho + alto + ancho + alto

para calcular área tenemos que multiplicar el ancho por el alto del rectángulo

area = ancho \* alto

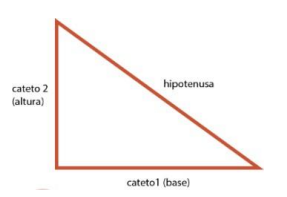
**Diseño**:

| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Processing |
| --- |
| **VARIABLES**:  anchoRect, altoRect: reales  perimetro, area: reales |
| **NOMBRE ALGORITMO**: calcularPerimetro  **PROCESO DEL ALGORITMO**   1. *Leer* anchoRect 2. *Leer* altoRect 3. perimetroRect ← (anchoRect + altoRect) \* 2 4. *Mostrar* perimetroRect |
| **NOMBRE ALGORITMO**: calcularArea  **PROCESO DEL ALGORITMO**   1. *Leer* anchoRect 2. *Leer* altoRect 3. areaRect ← anchoRect \* altoRect 4. *Mostrar* areaRect |



# **Ejercicio 14**

Una ayuda importante al momento de resolver problemas con algoritmos es asumir que su gran amigo son las matemáticas. Obtenga la hipotenusa de un triángulo rectángulo conociendo sus catetos



**Desarrollo del punto**

**Definición del Problema**:

Obtenga la hipotenusa de un triángulo rectángulo.

**Análisis**:

**Datos de Entrada**:

cateto1, cateto2

**Datos de Salida**:

hipotenusa

**Proceso**:

**¿Quién debe realizar el proceso?**: Processing

**¿Cuál es el proceso que se debe realizar?**

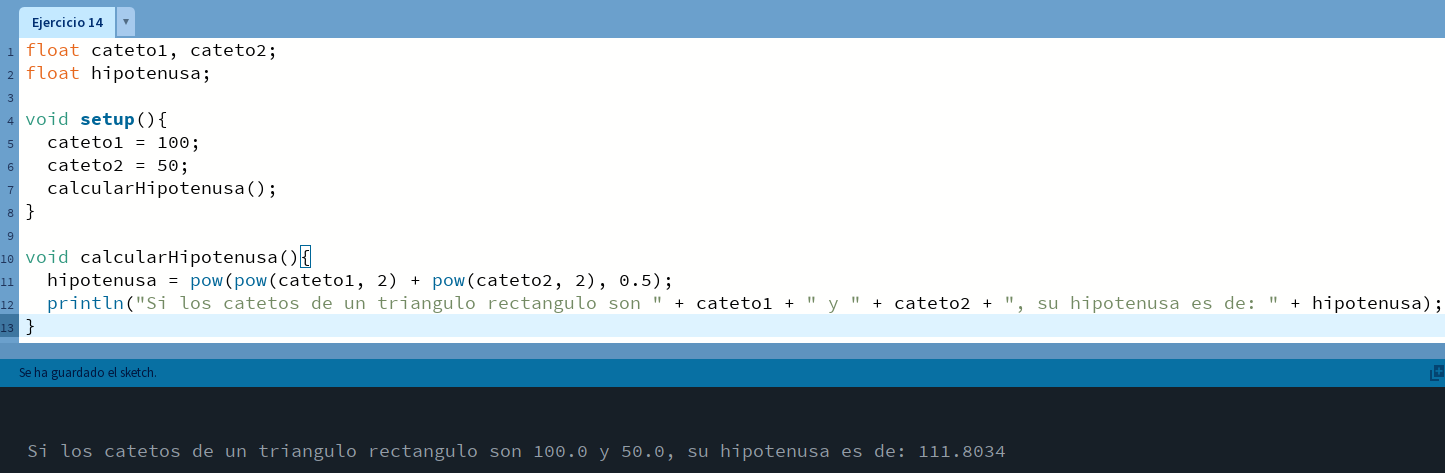
para obtener la hipotenusa tenemos que aplicar la siguiente formula

hipotenusa ^ 2 = cateto1 ^ 2 + cateto2 ^ 2

hipotenusa = √(cateto1 ^ 2 + cateto2 ^ 2)

**Diseño**:

| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Processing |
| --- |
| **VARIABLES**:  cateto1, cateto2: reales  hipotenusa: real |
| **NOMBRE ALGORITMO**: calcularHipotenusa  **PROCESO DEL ALGORITMO**   1. *Leer* cateto1 2. *Leer* cateto2 3. hipotenusa ← √(cateto1 ^ 2 + cateto2 ^ 2) 4. *Mostrar* hipotenusa |



# **Ejercicio 15**

Si viste algo de los apuntes y vídeos, esto debería ser muy fácil de resolver. Dados dos números permita calcular la suma, resta, multiplicación y división de estos. Considere que cada una de estas operaciones es un algoritmo cuando realice el diseño. Obviamente muestre los resultados.

**Desarrollo del punto**

**Definición del Problema**:

Calcular la suma, resta, multiplicación y división entre dos números.

**Análisis**:

**Datos de Entrada**:

numero1, numero2

**Datos de Salida**:

resultadoSuma, resultadoResta, resultadoMulti, resultadoDivi

**Proceso**:

**¿Quién debe realizar el proceso?**: Processing

**¿Cuál es el proceso que se debe realizar?**

para este ejercicio se deben hacer 4 algoritmos

suma:

resultadoSuma = numero1 + numero2

resta:

resultadoResta = numero1 - numero2

multiplicación

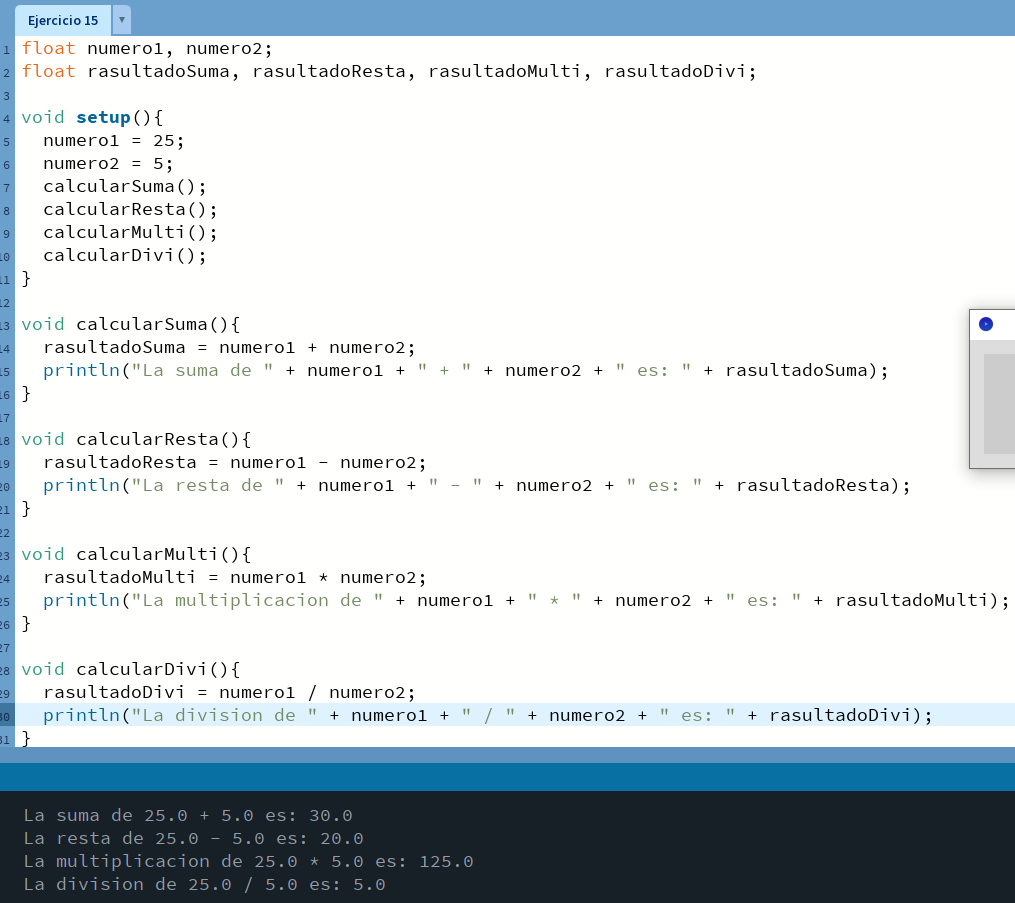
resultadoMulti = numero1 \* numero2

división

resultadoDivi = numero1 / numero2

**Diseño**:

| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Processing |
| --- |
| **VARIABLES**:  numero1, numero2: reales  resultadoSuma, resultadoResta, resultadoMulti, resultadoDivi: reales |
| **NOMBRE ALGORITMO**: calcularSuma  **PROCESO DEL ALGORITMO**   1. *Leer* numero1 2. *Leer* numero2 3. resultadoSuma ← numero1 + numero2 4. *Mostrar* resultadoSuma |
| **NOMBRE ALGORITMO**: calcularResta  **PROCESO DEL ALGORITMO**   1. *Leer* numero1 2. *Leer* numero2 3. resultadoResta ← numero1 - numero2 4. *Mostrar* resultadoResta |
| **NOMBRE ALGORITMO**: calcularMulti  **PROCESO DEL ALGORITMO**   1. *Leer* numero1 2. *Leer* numero2 3. resultadoMulti ← numero1 \* numero2 4. *Mostrar* resultadoMulti |
| **NOMBRE ALGORITMO**: calcularDivi  **PROCESO DEL ALGORITMO**   1. *Leer* numero1 2. *Leer* numero2 3. resultadoDivi ← numero1 / numero2 4. *Mostrar* resultadoDivi |



# **Ejercicio 16**

Necesitamos convertir una temperatura Fahrenheit en grados Celsius. Si no conoce la forma en la que se realiza esta conversión, debería investigarlo; para eso sirve la etapa de análisis. Pero como somos buenos, daremos una ayuda



**Desarrollo del punto**

**Definición del Problema**:

Convertir una temperatura Fahrenheit en grados Celsius

**Análisis**:

**Datos de Entrada**:

temperaturaFahren

**Datos de Salida**:

temperaturaCelsius

**Proceso**:

**¿Quién debe realizar el proceso?**: Processing

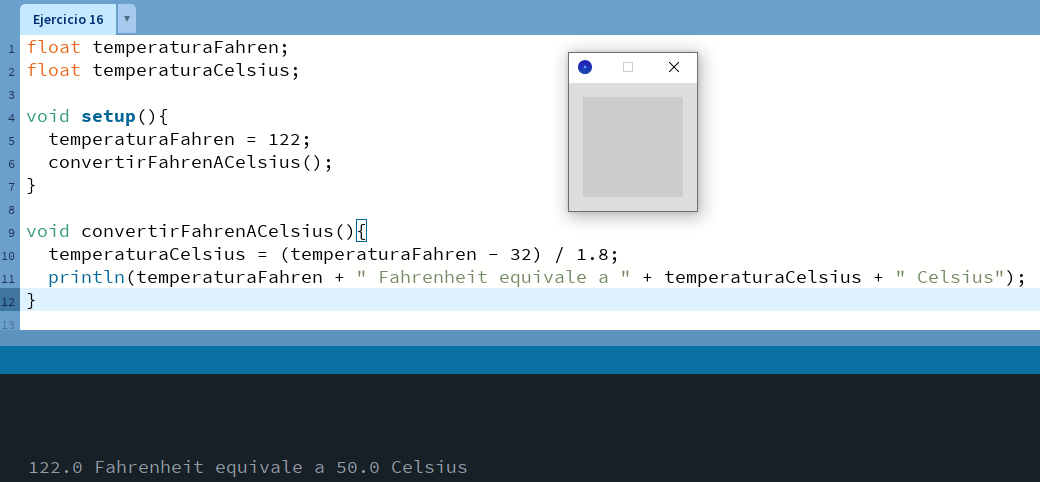
**¿Cuál es el proceso que se debe realizar?**

utilizar la siguiente formula:

temperaturaCelsius = (temperaturaFahren - 32) / 1.8

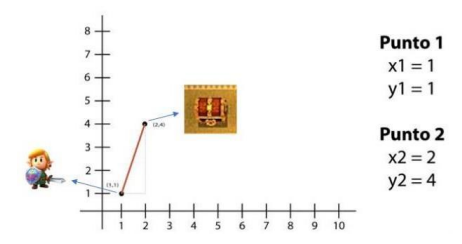
**Diseño**:

| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Processing |
| --- |
| **VARIABLES**:  temperaturaFahren: real  temperaturaCelsius: real |
| **NOMBRE ALGORITMO**: convertirFahrenACelsius  **PROCESO DEL ALGORITMO**   1. *Leer* temperaturaFahren 2. temperaturaCelsius ← (temperaturaFahren - 32) / 1.8 3. *Mostrar* temperaturaCelsius |



# **Ejercicio 17**

Si queremos representar personajes o power ups (premios) en la pantalla debemos primero ubicarlos en alguna posición dentro de la pantalla. Imagine que está en un juego donde un power up desaparece porque el personaje se acerca a una distancia de x unidades, sin importar por donde se acerque. Por tanto, para que desaparezca, en primer lugar, hay que determinar esa distancia. La forma de representar la posición de un objeto en la pantalla es a través de las coordenadas de un punto. Suponga que la posición de Link está representada por la coordenada (𝑥1, 𝑦1) , mientras que las de la caja de tesoro se halla en la posición (𝑥2, 𝑦2). Si observa con detenimiento se observa la conformación de un triángulo rectángulo, por lo que es posible aplicar Pitágoras para obtener la distancia



Para esto debe calcular el tamaño de los catetos y luego aplicar el teorema. Halle la distancia entre ambos objetos. Cuando programe, represente a Link con un Circulo, y al tesoro con un cuadrado. Además, mueva a Link mediante el mouse.

**Desarrollo del punto**

**Definición del Problema**:

Calcular la distancia entre dos puntos

**Análisis**:

**Datos de Entrada**:

posicionPunto1, posicionPunto2

**Datos de Salida**:

distanciaEntrePuntos

**Proceso**:

**¿Quién debe realizar el proceso?**: Processing

**¿Cuál es el proceso que se debe realizar?**

para calcular la distancia entre dos puntos tenemos que formar un triángulo rectángulo con estos puntos para aplicar pitágoras

el cateto1 seria

(posicionPunto2.x - posicionPunto1.x)

el cateto2 seria

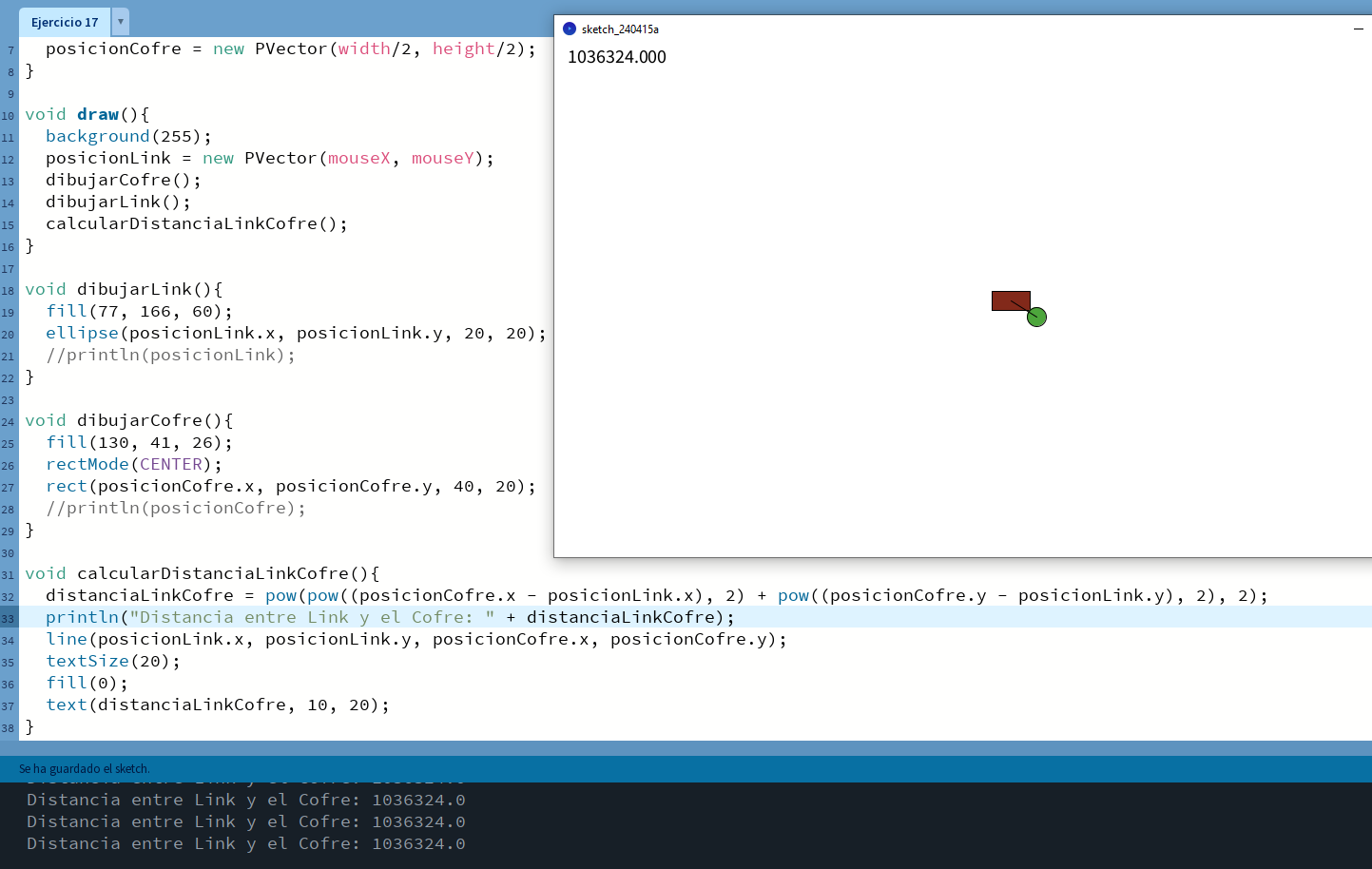
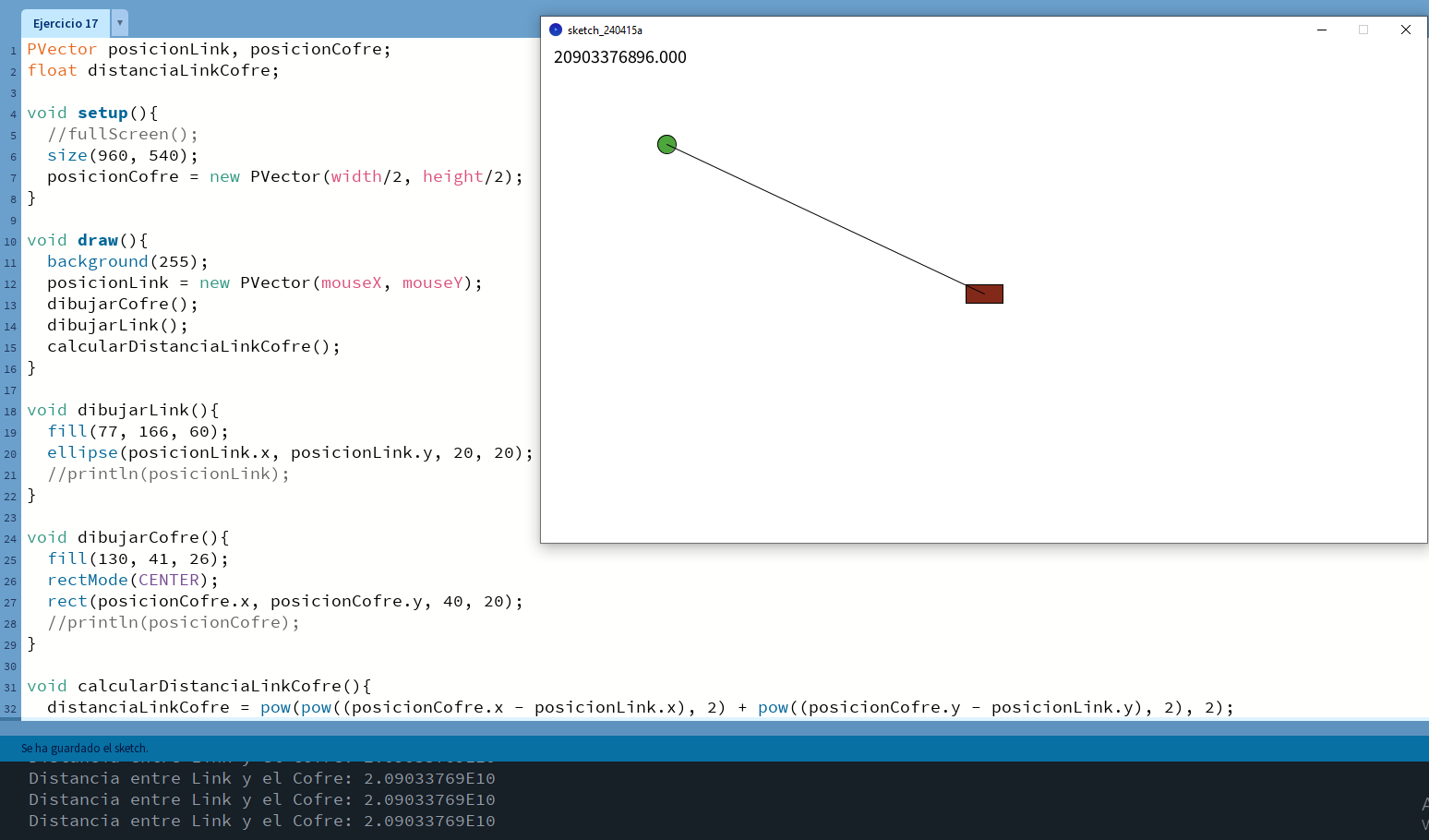
(posicionPunto2.y - posicionPunto1.y)

entonces utilizamos la formula de pitagoras (hipotenusa ^ 2 = cateto1 ^ 2 + cateto2 ^ 2) y quedaria asi:

distanciaEntrePuntos = √((posicionPunto2.x - posicionPunto1.x) ^ 2 + (posicionPunto2.y - posicionPunto1.y) ^ 2)

**Diseño**:

| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Lienzo (Processing) |
| --- |
| **VARIABLES**:  posicionLink, posicionCofre: coordenadas  distanciaLinkCofre: real |
| **NOMBRE ALGORITMO**: dibujarLink  **PROCESO DEL ALGORITMO**   1. posicionLink ← (mouseX, mouseY) 2. Dibujar una elipse (posicionLink.x, posicionLink.y, 20, 20) |
| **NOMBRE ALGORITMO**: dibujarCofre  **PROCESO DEL ALGORITMO**   1. *Leer* posicionCofre 2. Dibujar un rectángulo (posicionCofre.x, posicionCofre.y, 40, 20) |
| **NOMBRE ALGORITMO**: calcularDistanciaLinkCofre  **PROCESO DEL ALGORITMO**   1. distanciaLinkCofre← √((posicionCofre.x - posicionLink.x) ^ 2 + (posicionCofre.y - posicionLink.y) ^ 0.5) 2. *Mostrar* distanciaLinkCofre |



# **Ejercicio 18**

Desarrolle el análisis y diseño de un algoritmo que permita obtener las raíces de una ecuación de segundo grado. Además, utilice la estructura según para el análisis de la discriminante de la ecuación cuadrática. Obviamente codifique en Processing.

**Desarrollo del punto**

**Definición del Problema**:

desarrollar un algoritmo que resuelva una ecuación de segundo grado

**Análisis**:

**Datos de Entrada**:

numeroA, numeroB, numeroC

**Datos de Salida**:

x1, x2

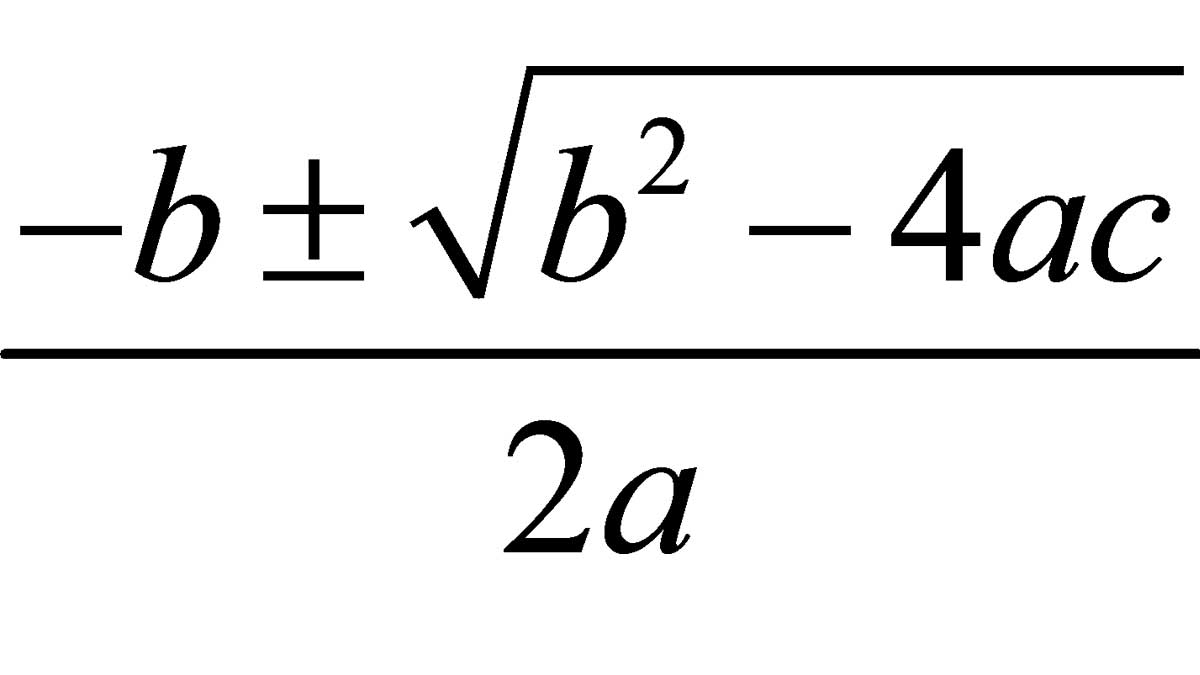
**Proceso**:

**¿Quién debe realizar el proceso?**: Processing

**¿Cuál es el proceso que se debe realizar?**

primero tendríamos que determinar el resultado de la discriminante para saber que calculo realizar

utilizaremos esta formula general



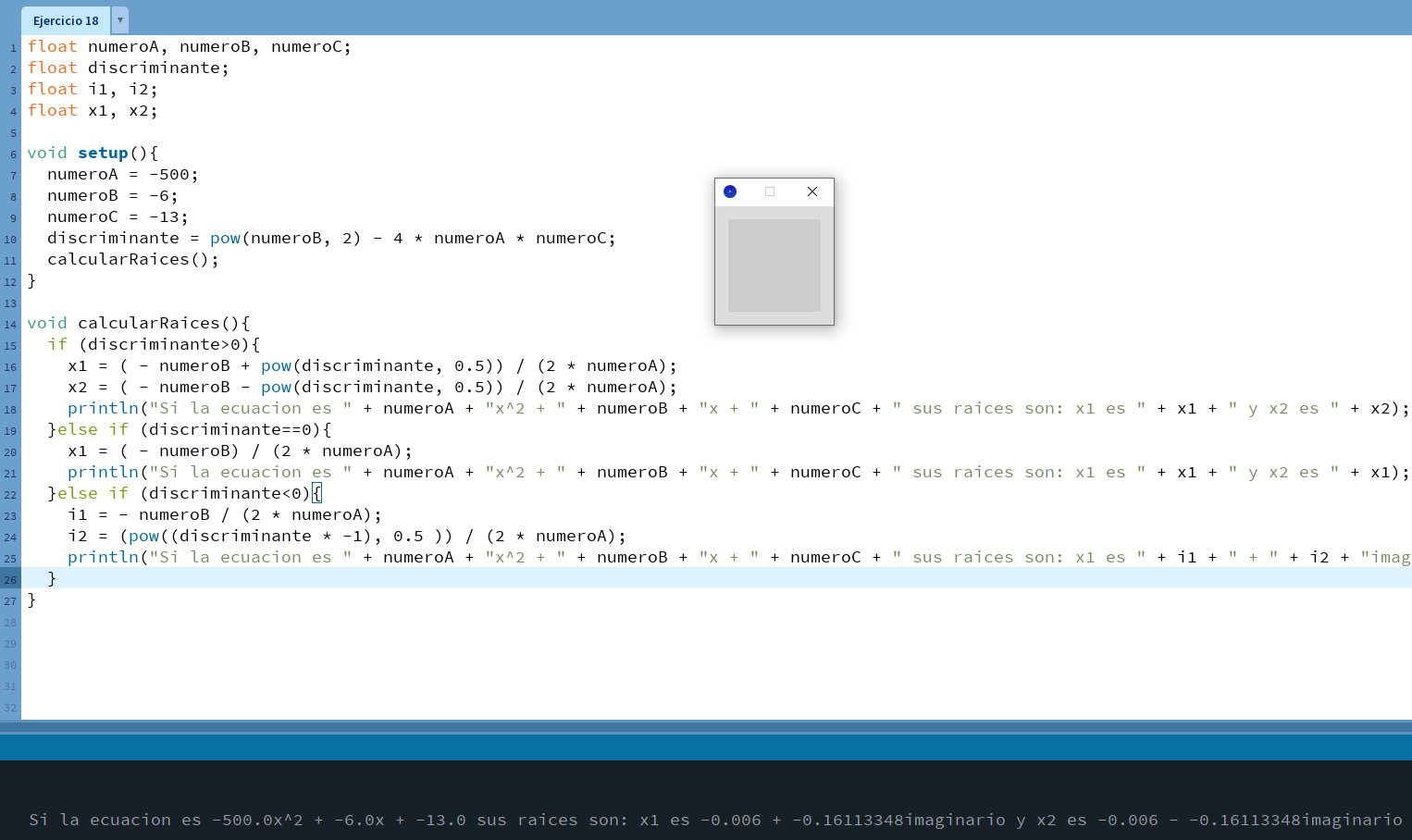
si la discriminante es mayor a 0 podemos sacar el resultado de x1 y x2

si la discriminante es igual a 0 podemos sacar el resultado de x

si la discriminante es menor a 0 el resultado será imaginario

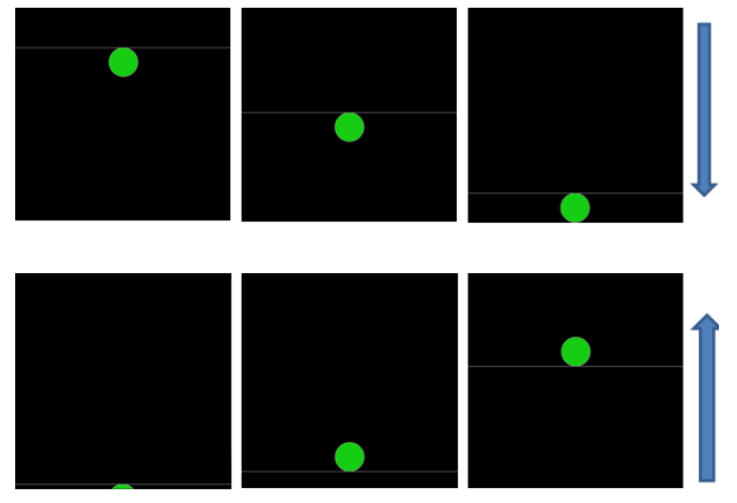
**Diseño**:

| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Processing |
| --- |
| **VARIABLES**:  numeroA, numeroB, numeroC: reales  discriminante: real  i1, i2: reales  x1, x2: reales |
| **NOMBRE ALGORITMO**: calcularRaices  **PROCESO DEL ALGORITMO**   1. *Leer* numeroA 2. *Leer* numeroB 3. *Leer* numeroC 4. discriminante ← numeroB ^ 2 - 4 \* numeroA \* numeroC 5. **segun\_sea** (discriminante) **hacer** 6. **caso** discriminante > 0:    1. x1 ← ( - numeroB + discriminante ^ 0.5) / (2 \* numeroA)    2. x2 ← ( - numeroB - discriminante ^ 0.5) / (2 \* numeroA)    3. *Mostrar* x1 y x2 7. **caso** discriminante = 0:    1. x1 ← ( - numeroB) / (2 \* numeroA)    2. *Mostrar* x1 8. **caso** discriminante < 0:    1. i1 ← - numeroB / (2 \* numeroA)    2. i2 ← ((discriminante \* -1) ^ 0.5 ) / (2 \* numeroA)    3. *Mostrar* i1 e i2 9. **fin\_segun** |



# **Ejercicio 19**

Declare las variables necesarias para dibujar una línea que se dibuja desde las coordenadas iniciales del lienzo y se extiende por todo el ancho. Sobre el punto medio de la línea y a una distancia de 40px (en sentido vertical desde la línea) dibuje una elipse que tenga como ancho 80px y de alto 80px. Dentro de la función draw(), actualice las variables necesarias para que la línea desde su inicio se mueva en dirección hacia abajo arrastrando la elipse. Mantenga en cero el valor para background(). Cuando la línea supere la posición de la altura del lienzo, debe invertir su sentido, es decir dirigirse hacia arriba arrastrando la elipse. Cuando la línea alcance nuevamente el valor 0 para su posición en y, el desplazamiento debe ser hacia abajo y así sucesivamente. El lienzo debería verse como en las siguientes figuras



**Desarrollo del punto**

**Definición del Problema**:

dibujar una linea con una elipse en el medio moviéndose de arriba a abajo

**Análisis**:

**Datos de Entrada**:

diámetro elipse, posY línea y elipse, dirección del movimiento

**Datos de Salida**:

el movimiento de la línea con la elipse encima dibujados en un lienzo

**Proceso**:

**¿Quién debe realizar el proceso?**: Lienzo (Processing)

**¿Cuál es el proceso que se debe realizar?**

tenemos que hacer un sistema para que aumente la posY pero a partir de un punto cambie el sentido

podemos hacer que si el valor dirección del movimiento es positivo aumenta la posY

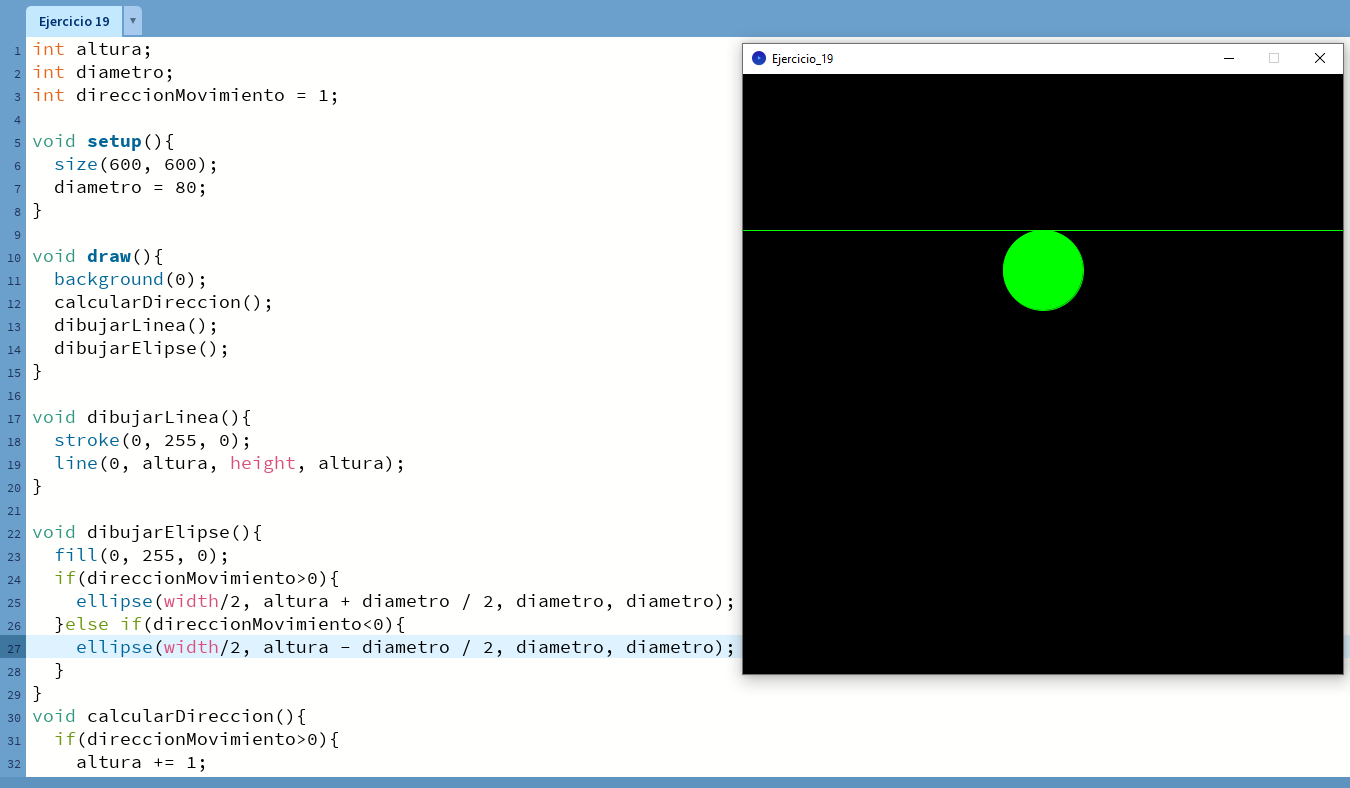
y si es negativo disminuye la posY

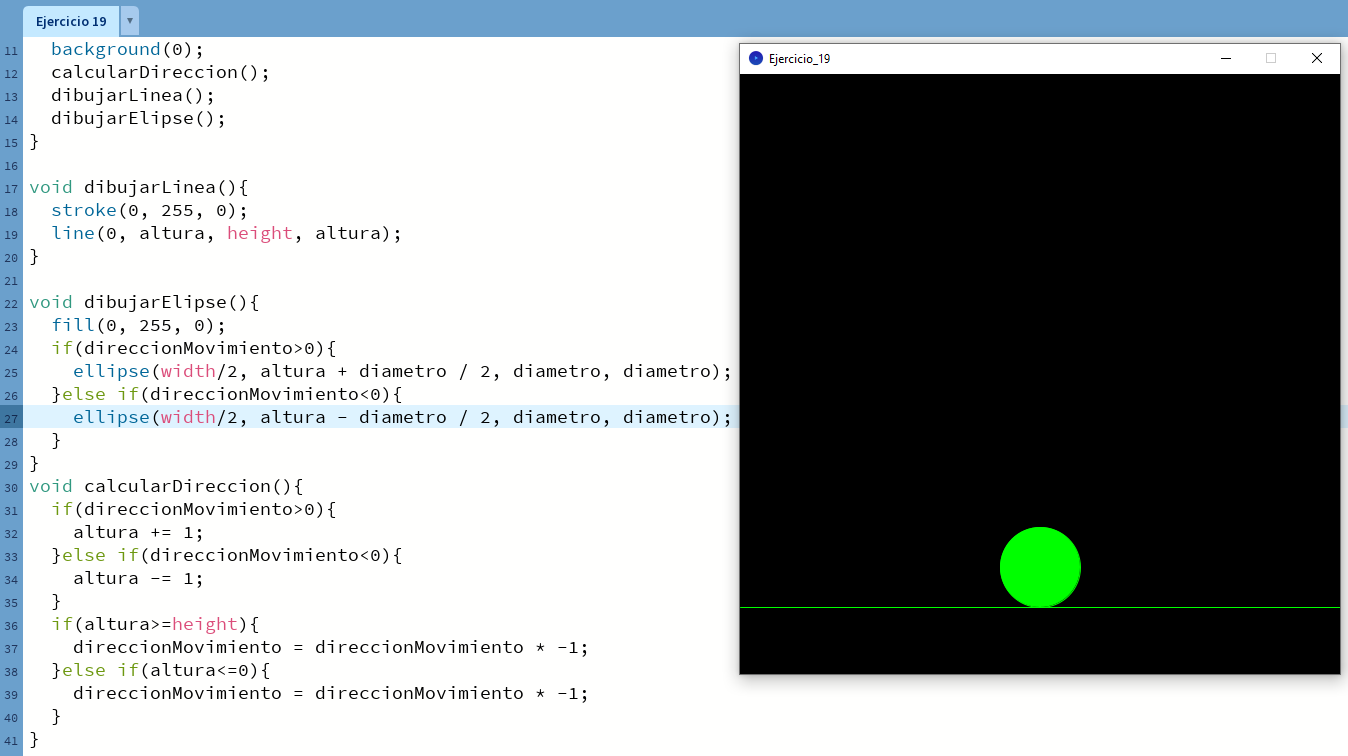
ahora para cambiar dirección del movimiento hacemos que cuando posY sea mayor a 0 sea positivo y cuando pase el borde invierte el valor, pasándolo a negativo (dirección del movimiento \* -1) y a la vuelta cuando dirección del movimiento sea negativo al llegar a 0 se invierte de nuevo el valor, pasándolo a positivo (dirección del movimiento \* -1)

solo queda cambia si la elipse va pos encima o debajo de la línea dependiendo también de la dirección del movimiento

**Diseño**:

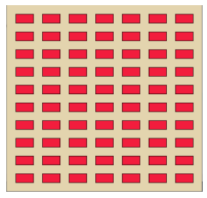
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Lienzo (Processing) |
| --- |
| **VARIABLES**:  posY: entero  diametro: entero  direccionMovimiento: entero |
| **NOMBRE ALGORITMO**: draw  **PROCESO DEL ALGORITMO**   1. *Leer* posY 2. *Leer* diametro 3. *Leer* direccionMovimiento 4. calcularDireccion 5. dibujarLinea 6. dibujarElipse |
| **NOMBRE ALGORITMO**: calcularDireccion  **PROCESO DEL ALGORITMO**   1. **si** (direccionMovimiento>0) **hacer**    1. posY ← posY + 1 2. **si\_no** (direccionMovimiento<0) **hacer**    1. posY ← posY - 1 3. **fin\_si** 4. **si** (posY>=height) **hacer**    1. direccionMovimiento ← direccionMovimiento \* -1 5. **si\_no** (direccionMovimiento<0) **hacer**    1. direccionMovimiento ← direccionMovimiento \* -1 6. **fin\_si** |
| **NOMBRE ALGORITMO**: dibujarLinea  **PROCESO DEL ALGORITMO**   1. Dibujar linea (0, posY, height, posY) |
| **NOMBRE ALGORITMO**: dibujarElipse  **PROCESO DEL ALGORITMO**   1. **si** (direccionMovimiento>0) **hacer**     1. Dibujar elipse (width/2, posY + diametro / 2, diametro, diametro) 2. **si\_no** (direccionMovimiento<0) **hacer**    1. Dibujar elipse (width/2, posY - diametro / 2, diametro, diametro) 3. **fin\_si** |





# **Ejercicio 20**

Dibuje en toda la extensión del lienzo de (440, 420) rectángulos de idénticas medidas (40 ancho y 20 de alto) y que mantengan una distancia de 20 pixeles entre ellos tanto horizontal como verticalmente. Utilice la estructura de control repetitiva for. El lienzo debería verse así:



**Desarrollo del punto**

**Definición del Problema**:

Dibujar rectángulos (de la misma medida y distancia cada rectángulo) en un lienzo usando estructura iterativa for

**Análisis**:

**Datos de Entrada**:

medidas de los rectángulos, distancia entre rectángulos, posición de rectángulos

**Datos de Salida**:

los rectángulos dibujados en un lienzo de manera ordenada

**Proceso**:

**¿Quién debe realizar el proceso?**: Lienzo (Processing)

**¿Cuál es el proceso que se debe realizar?**

Dibujar el primer rectángulo, para dibujar el siguiente cambiar sus coordenadas (coordenadas x del rectángulo += medida del rectángulo (ancho) + distancia entre rectángulos) repetir el proceso hasta el borde del lienzo formando una fila, luego cambiar la coordenada y para comenzar otra fila abajo< así hasta el borde del lienzo

**Diseño**:

| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Lienzo (Processing) |
| --- |
| **VARIABLES**:  anchoLienzo, altoLienzo: enteros  distanciaEntreRect, anchoRect, altoRect: enteros  coordenadasRect: coordenadas |
| **NOMBRE ALGORITMO**: dibujarRectangulos  **PROCESO DEL ALGORITMO**   1. *Leer* anchoLienzo 2. *Leer* altoLienzo 3. *Leer* distanciaEntreRect 4. *Leer* anchoRect 5. *Leer* altoRect 6. *Leer* coordenadasRect 7. **para** y ← coordenadasRect.y **hasta** altoLienzo **incremento** altoRect + distanciaEntreRect 8. **para** x ← coordenadasRect.x **hasta** anchoLienzo **incremento** anchoRect + distanciaEntreRect 9. Dibujar un rectángulo (coordenadasRect.x, coordenadasRect.y, anchoRect, altoRect) 10. **fin\_para** 11. **fin\_para** |

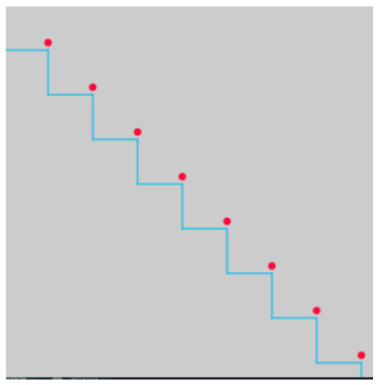


# **Ejercicio 21**

Utilizando la estructura de control repetitiva while() dibuje la siguiente imagen

utilizando líneas que forman escalones y sobre cada borde de escalón se dibuje un punto

de color rojo



**Desarrollo del punto**

**Definición del Problema**:

dibujar líneas con patrón de escalones con un punto en cada esquina

**Análisis**:

**Datos de Entrada**:

coordenada izquierda de la linea1 (puntoA), coordenada derecha de la linea1 (puntoB), coordenada abajo de la linea2 (puntoC), distancia de la línea, coordenada del punto (punto D)

**Datos de Salida**:

los escalones dibujados con los puntos dibujados en el lienzo

**Proceso**:

**¿Quién debe realizar el proceso?**: Lienzo (Processing)

**¿Cuál es el proceso que se debe realizar?**

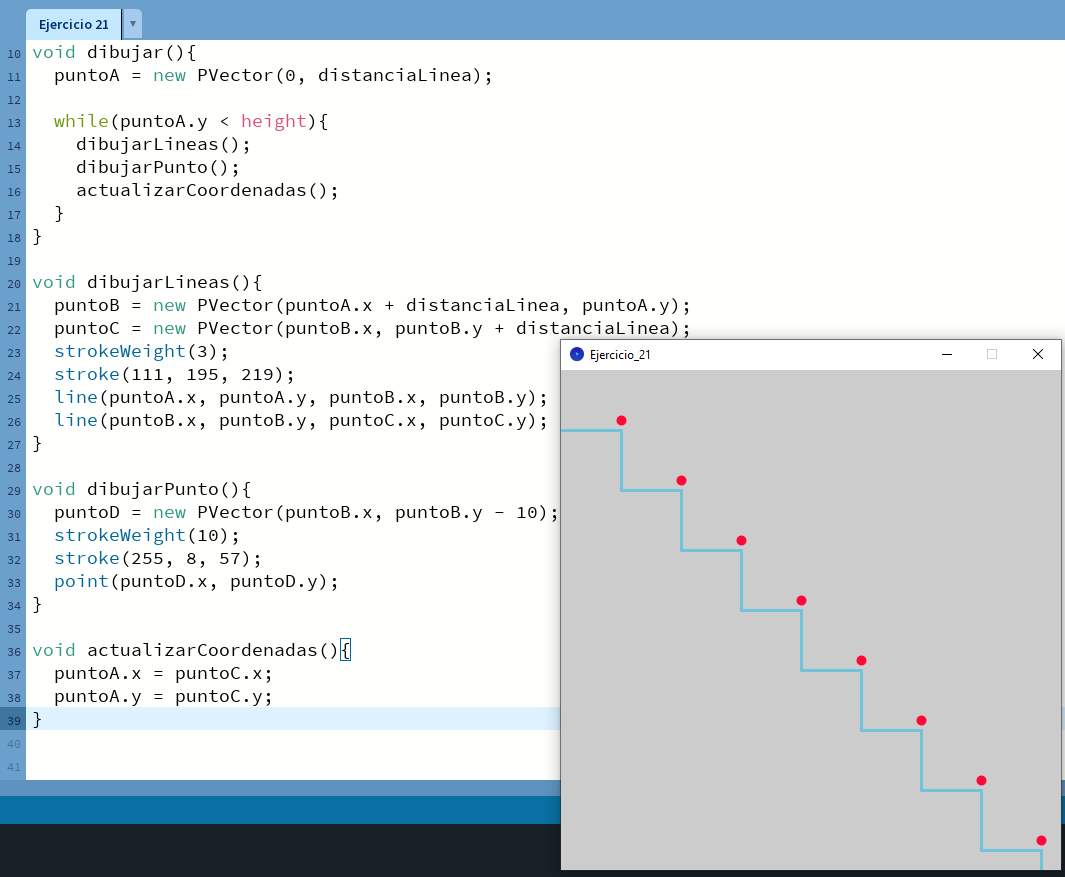
dibujar una línea horizontal entre con las coordenadas de los puntos A y B

dibujar una línea vertical entre con las coordenadas de los puntos B y C

dibujar un punto en el puntoD (mismas coordenadas x del puntoB, pero un poco menos en las coordenadas y)

**Diseño**:

| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Lienzo (Processing) |
| --- |
| **VARIABLES**:  puntoA, puntoB, puntoC, puntoD: coordenadas  distanciaLinea: entero |
| **NOMBRE ALGORITMO**: dibujar  **PROCESO DEL ALGORITMO**   1. *Leer* distanciaLinea 2. puntoA ← (0, distanciaLinea) 3. **mientras** (puntoA.y > height) **hacer**    1. dibujarLineas    2. dibujarPunto    3. actualizarCoordenadas 4. **fin\_mientras** |
| **NOMBRE ALGORITMO**: dibujarLineas  **PROCESO DEL ALGORITMO**   1. puntoB ← (puntoA.x + distanciaLinea, puntoA.y) 2. puntoC ← (puntoB.x, puntoB.y + distanciaLinea) 3. dibujar linea (puntoA.x, puntoA.y, puntoB.x, puntoB.y) 4. dibujar linea (puntoB.x, puntoB.y, puntoC.x, puntoC.y) |
| **NOMBRE ALGORITMO**: dibujarPunto  **PROCESO DEL ALGORITMO**   1. puntoD ← (puntoB.x, puntoB.y - 10) 2. dibujar punto (puntoD.x, puntoD.y) |
| **NOMBRE ALGORITMO**: actualizarCoordenadas  **PROCESO DEL ALGORITMO**   1. puntoA.x ← puntoC.x 2. puntoA.y ← puntoC.y |



# **Fuentes Biográficas**

[clase 03 09 04 24 Fudnamentos de Programación Orientada a objetos](https://youtu.be/YHpW_HMP_PE?si=L373GL9QjVTCPc2X)

[Ejemplo de análisis - diseño y codificación de un ejercicio del Trabajo Práctico](https://youtu.be/_YblzDgoAus)

[07 Fases en la resolución de problemas mediante algoritmos](https://virtual.unju.edu.ar/pluginfile.php/616478/mod_resource/content/6/03%20Fases%20en%20la%20resoluci%C3%B3n%20de%20problemas%20con%20algoritmos.pdf)

[11 Bifurcaciones - Expresiones lógicas - Estructuras selectivas](https://virtual.unju.edu.ar/pluginfile.php/616669/mod_resource/content/5/06%20Bifurcaciones%20-%20Expresiones%20L%C3%B3gicas%20-%20Estructuras%20selectivas.pdf)

[12 Estructuras de control selectivas en Processing](https://youtu.be/Nr3NdAxjqsE)

[14 Estructuras de Control Iterativas](https://virtual.unju.edu.ar/pluginfile.php/617801/mod_resource/content/3/08%20Estructuras%20Iterativas.pdf)

[18 Estructora while en Processing](https://youtu.be/dysxlptjAx0)

[20 Estructura for en Processing](https://youtu.be/kApd8u4l0tU)

[21 Ejemplo de for anidado](https://youtu.be/r-Olf82ViNE)