|  |  |  |
| --- | --- | --- |
|  | TECNICATURA UNIVERSITARIA EN DISEÑO INTEGRAL DE VIDEOJUEGOS  FACULTAD DE INGENIERÍA  Universidad Nacional de Jujuy |  |

*Profesores:*

*Mg. Ing. Ariel Alejandro Vega*

*Año 2024*

Trabajo Práctico N° 1

Fabrego Lucas Ivan - 0718

**FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS**

Sección Expresiones aritméticas y lógicas

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

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

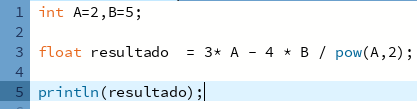
Resolución necesaria en Word:

(3\*A)-(4\*B/(A^2)) 6-(4\*B/4)

6-5

1

Captura de Processing



Ojo: Colocar la captura, no reemplaza que deban agregar a la carpeta el archivo .pde que contiene el código programado.

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

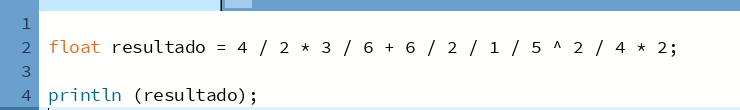
(4 / 2 \* 3 / 6) + (6 / 2 / 1 / (5 ^ 2) / 4 \* 2)

(2 \* 1 / 2) + (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.**

1. b ^ 2 – 4 \* a \* c
2. 3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17
3. (b + d) / (c + 4)
4. (x ^ 2 + y ^ 2) ^ (1 / 2)

Para aclarar que indicamos con” Luego escribirlas como expresiones algebraicas” lo

aplicamos con el punto

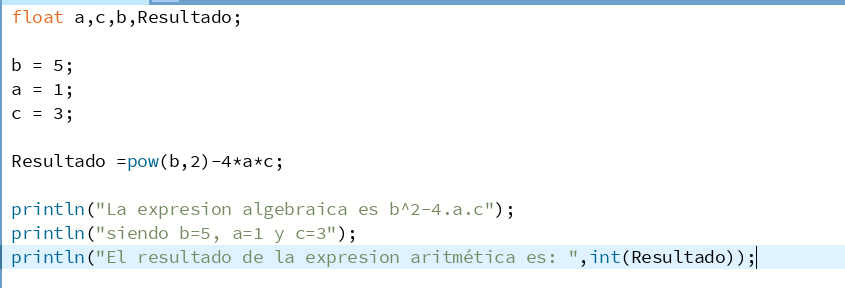
1. B= 5; a=1 ; c=3

5 ^2 – 4 \*1 \* 3

25 – 12

13

Expresión algebraica “𝑏2 − 4. 𝑎. 𝑐”



1. X = 1

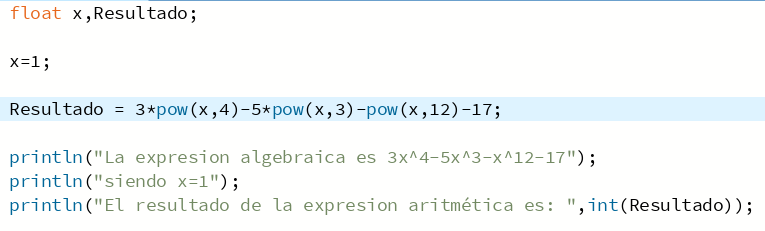
3 \* 1^4 – 5 \* 1^3 – 1 ^12 – 17

3 \* 1 – 5 \* 1 – 1 – 17

3 – 5 – 1 – 17

-20

Expresión Algebraica “



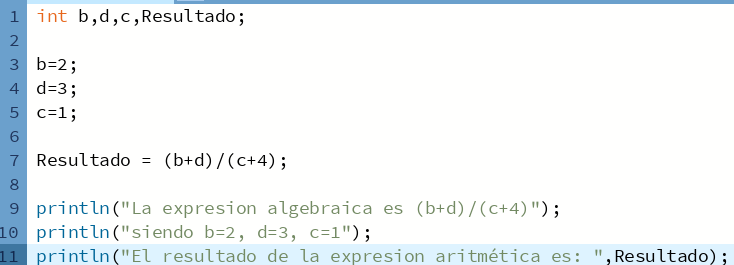
1. b = 2; d =3; c =1

(2 + 3) / (1 + 4)

5/5

1

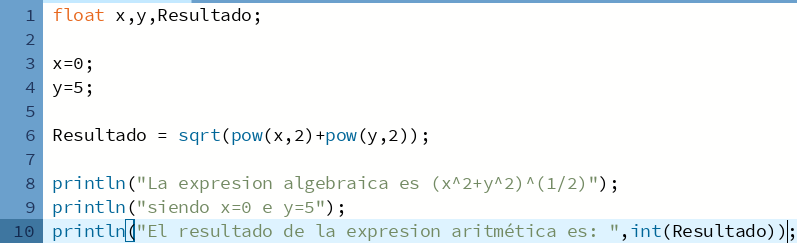
Expresión Algebraica “”



1. x = 0; y = 5

5

Expresión Algebraica “”



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:

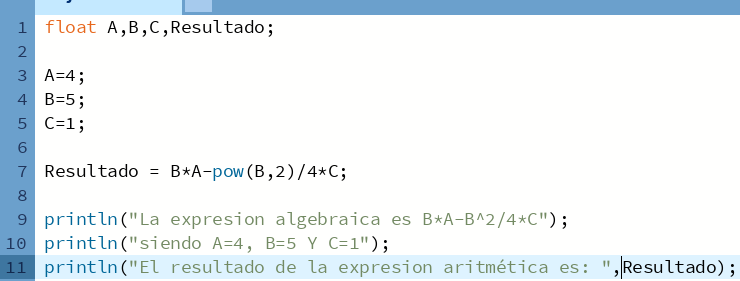
1. B \* A – B ^ 2 / 4 \* C

5 \* 4 – 5 ^2 / 4 \* 1

20 – 25/4

55/4

13,75

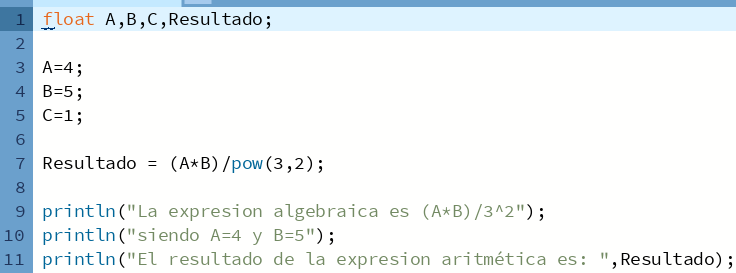


1. (A \* B) / 3 ^ 2

(4 \* 5) / 3 ^ 2

20 / 9

2,222222



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

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

((6 / 2 \* 4 + 10) \* 15) – 6

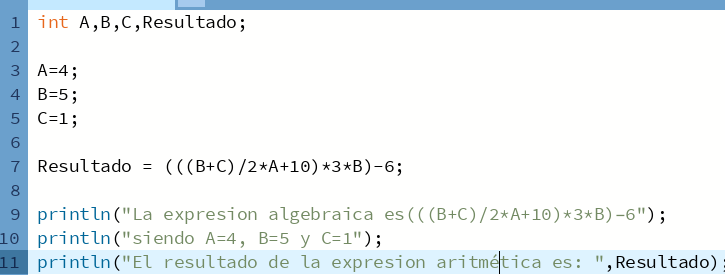
((3 \* 4 + 10) \* 15) -6

(12 + 10) \* 15) – 6

(22 \* 15) – 6

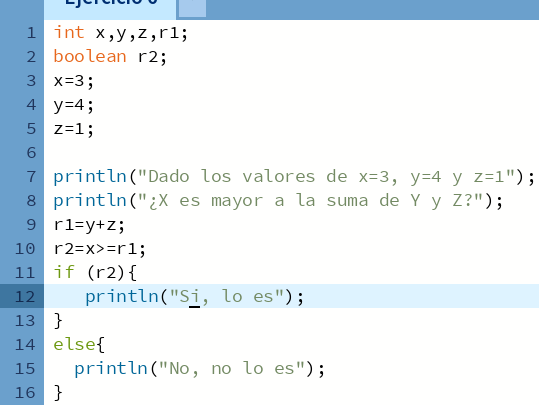
330 – 6

324



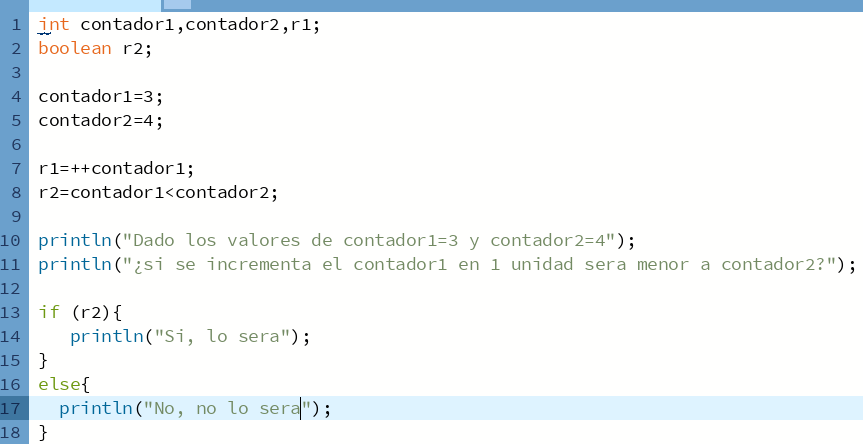
Ejercicio 6: Para x=3, y=4; z=1, evaluar el resultado de R1 = y+z = 4 + 1 = 5

R2 = x >= R1 🡪 3 >= 5 🡪 falso



Ejercicio 7: Para contador1=3, contador2=4, evaluar el resultado de R1 = ++contador1 🡪 R1 = 4

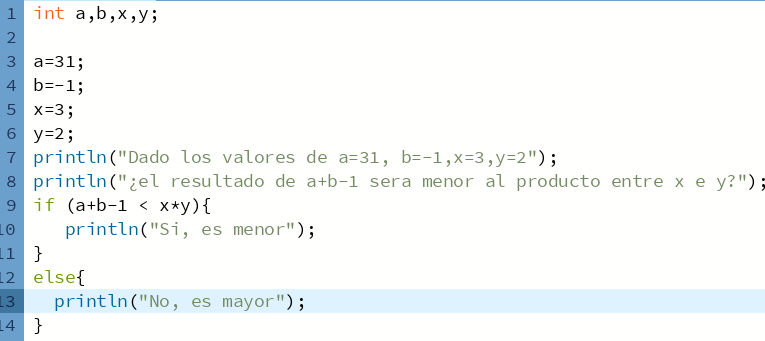
R2 = contador1 < contador2 🡪 falso



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

31 + (-1) – 1 < 3\*2

29 < 6 🡪 Falso



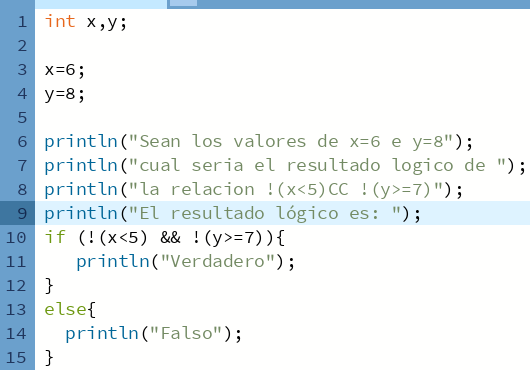
Ejercicio 9: Para x=6, y=8, evaluar el resultado de

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

¡(falso) cc ¡(verdadero)

Verdadero CC falso

falso



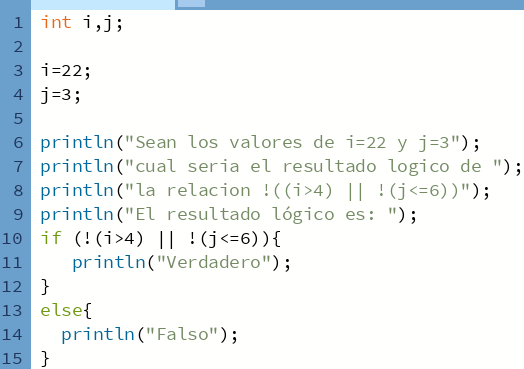
Ejercicio 10: Para i=22, j=3, evaluar el resultado de

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

¡(verdadero) || ¡(verdadero)

Falso o falso

falso



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

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

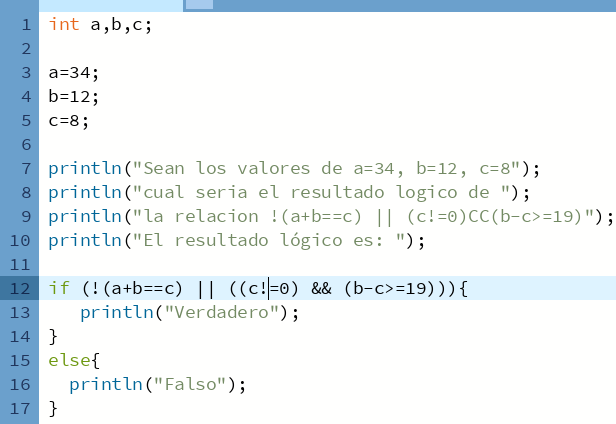
!(34+12==8) || (8!=0) CC (12-8=19)

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

!(falso) || (verdadero) CC (falso)

Verdadero || falso

verdadero



# Sección Análisis – Diseño y Codificación de algoritmos – Aplicación de estructuras de control

Para cada ejercicio, en el archivo Word agregar las secciones de análisis y diseño, mientras que, para la codificación, crear el archivo de Processing.

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.

**Definición del problema:** ingresar por teclado un nombre

**Análisis:**

Datos de entrada

Nombre: cadena de caracteres

Datos de Salida

Mensaje: cadena de caracteres

Mostrar nombre en el saludo

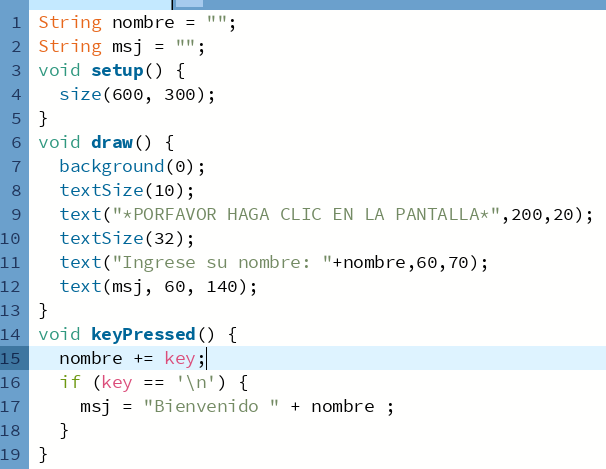
Proceso

Leer nombre ingresado

**Diseño**

|  |
| --- |
| Entidad Usuario |
| Variables  nombre: Cadena de caracteres;  msj: cadena de caracters; |
| Nombre del algoritmo Saludo  nombre🡨””  msj 🡨””  crear pantalla tamaño (x,y)  mostrar “ingrese su nombre”  nombre🡨 nombre + tecla presionada  mostrar “Bienvenido “ + nombre |

**Codificación en processing**

****

Ejercicio 13: Será común resolver problemas utilizando variables. Calcule el perímetro y área de un rectángulo dada su base y su altura.

**Definición del problema**: Calcular el área y el perimetro de un rectángulo.

**Análisis**

Datos entrada

base, altura: real

datos salida

perímetro, área: real

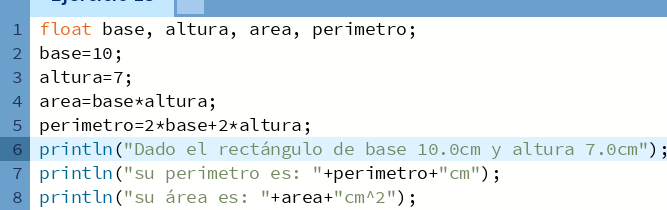
Preceso

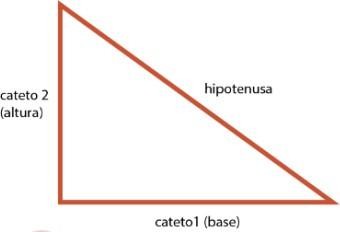
Calcular el área y el perímetro según la base y altura

**Diseño**

|  |
| --- |
| Entidad: Consola |
| Variables  base, altura: real  area, perímetro: real |
| Nombre del algoritmo: Area y perímetro  inicio  Leer base  Leer altura  Perímetro🡨2\*base + 2\*altura  Área🡨 base \* altura  Mostrar perímetro  Mostrar área  fin |

**Codificación en processing**



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

**Descripción del problema**: hallar la hipotenusa de un triángulo rectángulo

**Análisis**

Datos de entrada

Base, altura: real

Datos de salida

Hipotenusa: real

Proceso

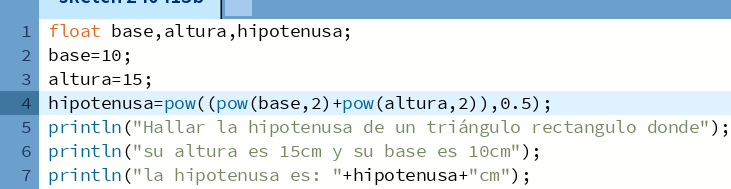
Calcular la hipotenusa con la base y la atura con la formula:

Donde h es Hipotenusa, **b** es la base y **a** es la altura

**Diseño**

|  |
| --- |
| Entidad: Consola |
| Variables  Base,altura,hipotenusa: real |
| Nombre del código: Hallar\_Hipotenusa  Incio  Leer base  Leer altura  Hipotenusa=(base^2+altura^2)^(0.5)  Mostrar hipotenusa  fin |

**Codificación en processing**



Ejercicio 15: 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.

**Descripción del problema**: Resolver la Suma, Resta, Multiplicación y División de 2 numero

**Análisis**:

Datos de entrada

Num1,num2: enteros

Datos de salida

Suma, resta, multiplicación: enteros

División: real

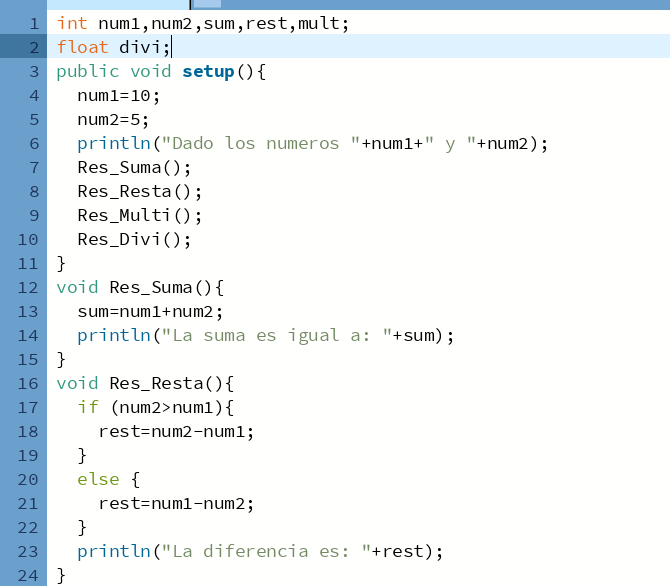
Proceso

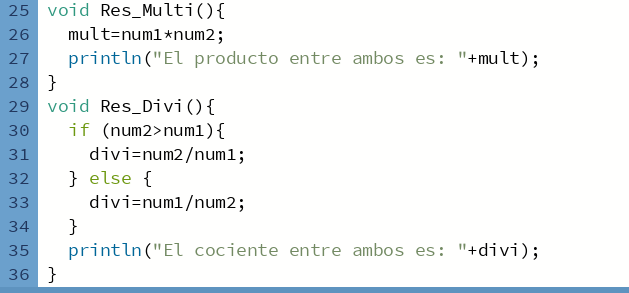
Realizar las operaciones matemáticas de Suma, Resta, multiplicación y división

**Diseño:**

|  |
| --- |
| Entidad: Consola |
| Variables:  Num1, num2, suma, resta, multiplicación: enteros  División: real |
| Nombre de código: operaciones matemáticas  inicio  Leer num1  Leer num2  Llamar Res\_Suma()  Llamar Res\_Resta()  Llamar Res\_Mult()  Llamar Res\_Divi()  fin |
| Nombre de código: Res\_Suma()  Inicio  Suma🡨num1+num2  Mostrar Suma  fin |
| Nombre de código: Res\_Resta()  Inicio  Si num2 > num1 hacer  Resta🡨num2-num1  Sino  Resta🡨num1-num2  finsi  Mostrar resta  Fin |
| Nombre de código: Res\_Mult()  Inicio  Multiplicacion🡨num1\*num2  Mostrar multiplicación  fin |
| Nombre de código: Res\_Divi()  Inicio  Si num2 > num1 hacer  División🡨num2/num1  Sino  División🡨num1/num2  Finsi  Mostrar división  Fin |

**Codificación en processing**





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

**Definición del problema**: Transformar la temperatura Fahrenhit en grados Celsius

**Análisis**

Datos de entrada

temperaturaFahrenheit=real

Datos de salida

TemperaturaCelcius=real

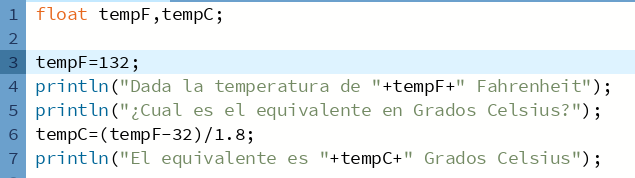
Proceso

 Transformar temperatura Fahrenheit en temperatura Celsius con la formula

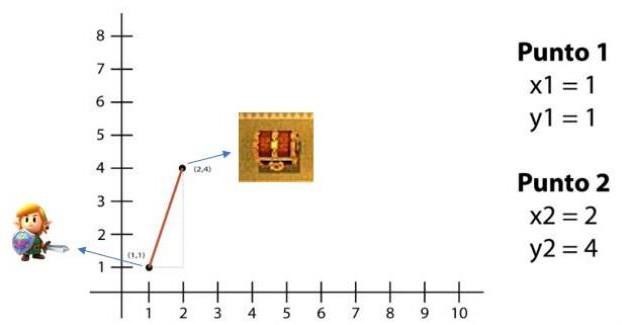
**Diseño**

|  |
| --- |
| Entidad: Consola |
| Variables  tempF,tempC: real |
| Nombre del código: TemperaturaFC  Inicio  Leer tempF  tempC🡨(tempF-32)/1.8  mostrar tmpC  fin |

**Codificación en processing**



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.

**Definición del problema**: Hallar la distancia entre circulo y cuadrado. Mover circulo con el mouse

**Análisis**

Datos de entrada

P1, p2,p3,p4: Coordenadas cartecianas

Datos de salida

Distancia: real

Proceso

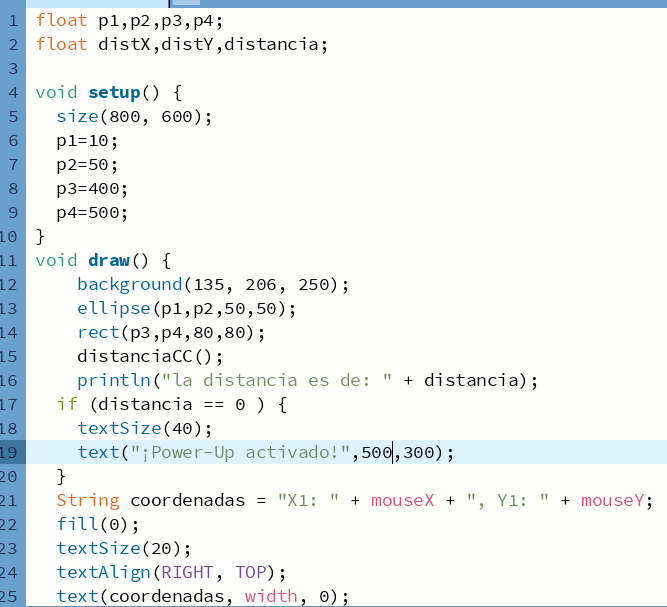
Calcular distancia entre circulo y cuadrado aplicando Pitágoras

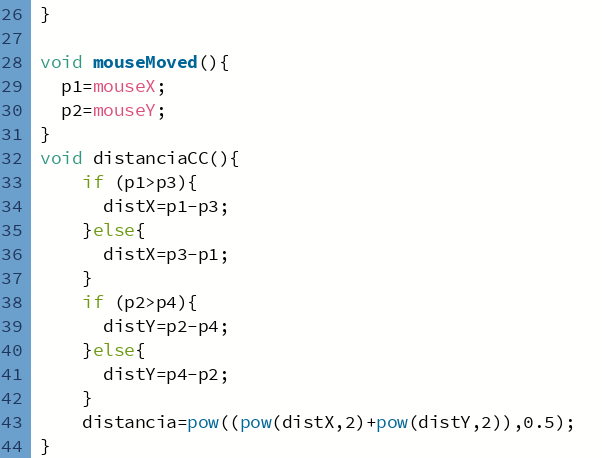
Desaparecer cuadrado cuando sea alcanzado por circulo

**Diseño**

|  |
| --- |
| Entidad: Mouse |
| Variables  p1, p2, p3, p4: cooredenadas cartecianas  distX, distY, distancia: real |
| Nombre del código: Power\_up  Inicio  Leer p1  Leer p2  Leer p3  Leer p4  Circulo(p1,p2)  Cuadrado(p3,p4)  Si p1>p3  distX🡨p1-p3  sino  distX🡨p3-p1  finsi  si p2>p4  distY🡨p2-p4  sino  distY🡨p4-p2  finsi  Distancia<-(distX^2+distY^2)^(0.5)  Mostrar distancia  fin |

**Codificacion en processing**





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.

**Definición del problema**: Obtener raíces de una ecuación en segundo grado

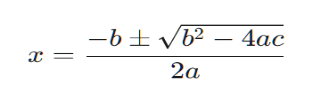
**Análisis:**

Datos de entrada:

a,b,c,disc: real

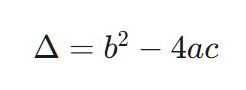
Datos de salida:

x1,x2: real

 Proceso

Resolver la ecuación mediante la formula

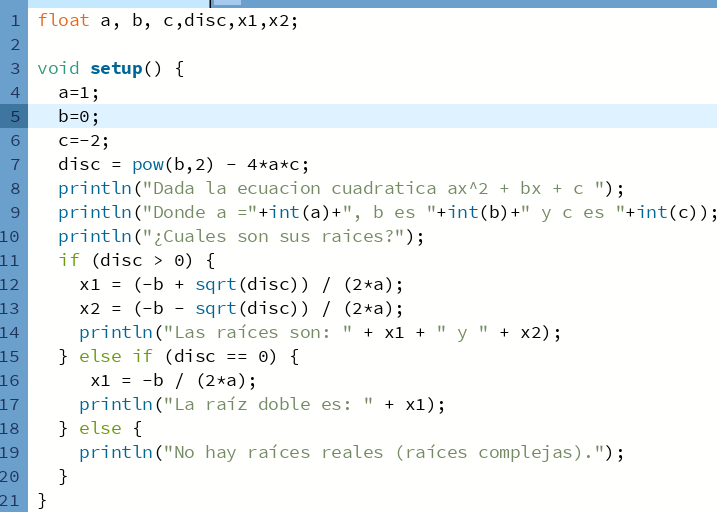
Aplicando las reglas del determinante con formula



**Diseño**

|  |
| --- |
| Entidad: Consola |
| Variables  a, b, c, disc: real  x1, x2: real |
| Nombre del algoritmo: Raices  Inicio  Leer a  Leer b  Leer c  Disc🡨b^2-2\*a\*c  Si disc>0 hacer  X1🡨(-b+(disc)^(0.5))/(2\*a)  X2🡨(-b-(disc)^(0.5))/(2\*a)  Sino  Si disc==0 hacer  X1=-b/(2\*a)  Sino  Mostrar “la ecuación no tiene raíces reales”  Finsi  Finsi  fin |

**Codificación en processing**



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



**Definición del problema**: Dibujar en un lienzo una elipse sobre una línea que se desplaza verticalmente de extremo a extremo

**Análisis:**

Datos de entrada

P1, p2: enteros

Datos de salida

Dibujo línea horizontal

Dibujo circulo sobre linea

Movimiento de figuras

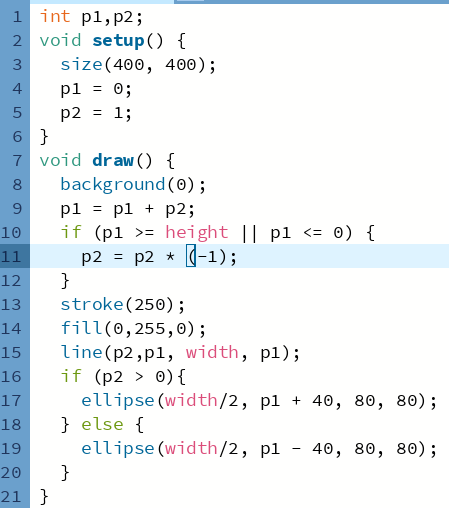
Proceso

Crear una línea móvil con un círculo en el medio

**Diseño**

|  |
| --- |
| Entidad: Consola |
| Variables  P1, p2: Entero |
| Nombre código: Figura\_movil  Incio  Definir lienzo (400,400)  Leer p1  Leer p2  P1🡨p1+p2  Si (p1>= alturaLienzo || p1 <=0) entonces  P2=p2\*(-1)  Fin si  Línea (p2, p1, ancholienzo, p1)  Si (p2>0) entonces  Elipse (anchoLienzo/2, p1+40, 80, 80)  Sino  Elipse (anchoLienzo/2, p1-40, 80, 80)  Finsi  fin |

**Código en Processing**



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:



**Definición del problema:** Dibujar en el lienzo, rectángulos de la misma medida usando estructuras iterativas

**Análisis:**

Datos entrada

CoorRect: Coordenadas Cartesianas;

AltLienzo: Entero;

AncLienzo: Entero;

AltoRec: Entero;

AnchoRec: Entero;

DistanciaR: Entero;

Datos Salida

Rectangulos dibujados en el lienzo

Proceso

Dibujar los rectángulos

**Diseño**

|  |
| --- |
| Entidad: Lienzo |
| Variables:  CoorRect: Coordenadas;  AltLienzo,AncLienzo: Entero;  AltoRec,AnchoRec,DistanciaR: Entero; |
| Nombre del Algoritmo: Rectangulos en Lienzo  Algoritmo  Inicio  AltLienzo 🡨 420  AncLienzo 🡨440  AltoRec 🡨 20  AnchoRec 🡨 40  DistanciaR 🡨 20  Para x🡨CoorRect.x hasta x<AncLienzo ; x+=(AnchoRec+Distancia)){  Para y🡨CoorRect.y hasta y<AltLienzo ; y+=(AltoRec+Distancia)  Dibujar rectángulo en (x,y,AnchoRec,AltoRec);  finPara  finPara  Fin |

**Código en Processing**



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



El tamaño del lienzo es size(500,500). La estructura while() se ejecuta dentro de la función setup(). La condición es que solo se dibuje dentro del lienzo. Utilice variables que puedan ayudar a la construcción del dibujo, por ej: x, y, anchoEscalon, altoEscalon, etc.

**Definición del problema**: Dibujar escalones con un punto rojo en el vertice

**Análisis**

Datos de entrada

Pa, pb, pc, pd: coordenadas cartesianas en 2d

DistanciaLinea: entero

Datos de salida

Dibujo línea horizontal

Dibujo línea vertical

Dibujo punto rojo

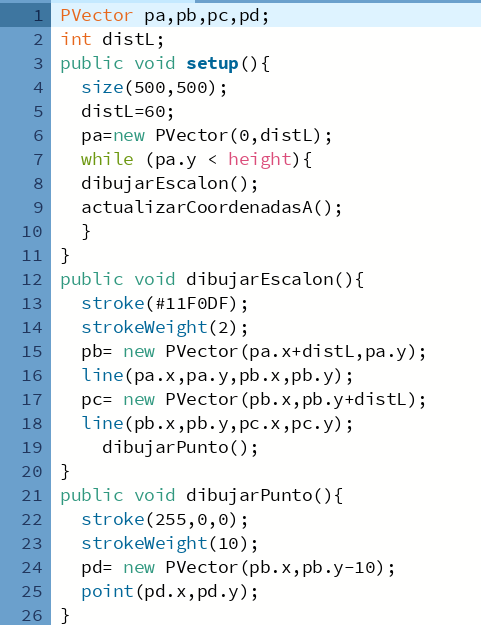
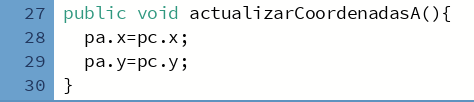
Proceso

* Dibujar una línea horizontal entre los puntos pa y pb
* Dibujar una línea vertical entre los puntos pb y pc
* Dibujar un punto en la siguiente posición: x= posición en x de pb, y=posición en y de b -5 unidades
* Actualizar coordenadas de pa como pa.x=pc.x y pa.y=pc.y

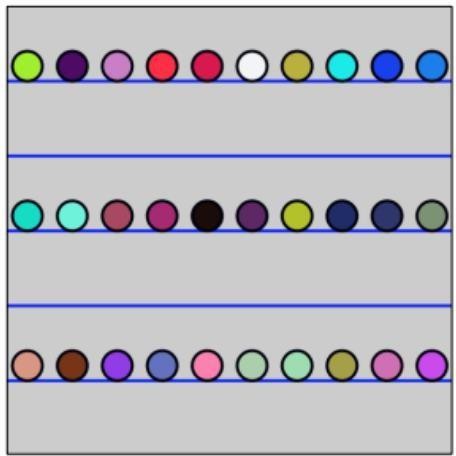
**Diseño**

|  |
| --- |
| **Entidad**: Escalon |
| **Variables**  Pa, pb, pc, pd: coordenadas cartesianas  distL: entero |
| **Nombre del algoritmo** Escalon  inicio  Lienzo (500,500)  Leer distL  Definimos pa  Mientras (pa.y < alturaLienzo) hacer  Llamar dibujarEscalon()  Llamar actalizarCoordenadasA()  finMientras  fin |
| **Nombre del algoritmo** dibujarEscalon()  Inicio  Definir pb (pa.x+distL,pa.y)  Dibujar línea (pa.x,pa.y,pb.x,pb.y)  Definir pc (pb.x,pb.y+distL)  Dibujar linea vertical (pb.x,pb.y,pc.x,pc.y)  Llamar dibujarPunto()  fin |
| **Nombre del algoritmo** dibujarPunto()  Incio  Definir pd = (pb.x,pb.y-10)  Dibujar punto en (pd.x,pd.y)  fin |
| **Nombre del algoritmo** actualizarCoordenadasA()  Incio  Pa.x=pc.x  Pa.y=pc.y  fin |

**Codigo en processing**

Ejercicio 22: Utilizando la estructura de control repetitiva do-while. Replique la siguiente imagen



La imagen debe ser construida desde la función setup(). Defina el tamaño del lienzo en size(600,600), verticalmente 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.

**Definición del problema**: Dibujar puntos sobre líneas espaciadas de forma escalada

**Análisis:**

Datos de entrada

lineaX, lineaY, puntoX, puntoY: enteros

distanciaPunto: entero

Datos de Salida

Dibujo de líneas horizontales intercaladas en eje y

Dibujo de puntos sobre línea

Proceso

Dibujar línea en punto inicial (LineaX,lineaY) y punto final Ancho de lienzo

Dibujar puntos sobre línea

Repetir intercalando la aparición de los puntos en líneas de por medio

**Diseño**

|  |
| --- |
| Entidad: Dibujo |
| Variables  lineaX, lineaY, puntoX, puntoY: enteros  distanciaPunto: entero |
| Nombre del programa Linea\_circulo  Inicio  Definir lienzo (600,600)  Leer lineaX  Leer lineaY  Leer cirY  Leer distC  Hacer  cirX🡨distC  hacer  crear línea (lineaX,lineaY,anchoLienzo,lineaY)  crear circulo (cirX,cirY,45,45)  cirX🡨distC\*2  mientras (cirX < anchoLienzo)  lineaY🡨lineaY+100  cirY🡨cirY+200  mientras (lineaY < alturaLienzo)  finsi |

**Código en Processing**

