|  |  |  |
| --- | --- | --- |
| Texto  Descripción generada automáticamente con confianza media | TECNICATURA UNIVERSITARIA EN DISEÑO INTEGRAL DE VIDEOJUEGOS  FACULTAD DE INGENIERÍA  Universidad Nacional de Jujuy |  |

Trabajo Práctico

N°1

Flores Milagros Soledad– LU /TUV000475

*Profesor:*

*Mg. Ing. Ariel Alejandro Vega*

*Año 2024*

**FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS**

REGLAMENTO

Resolver cada ejercicio en un archivo Word y luego programarlo en Processing. En el caso

de la programación crear un archivo por ejercicio.

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

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

Resolución necesaria en Word:

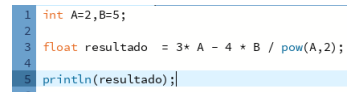
(3\*A)-(4\*B/(A^2))

6-(4\*B/4)

6-5

1

Captura de Processing



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

**Resolución**

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

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

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

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

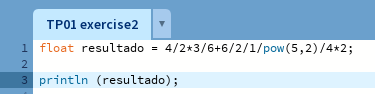
1 + (0.12 / 4 \* 2)

1 + (0.03\* 2)

1 + (0.06)

1.06

**Captura de Processing**



**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 Separar en términos y resolver de izq. a derecha.

b) 3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17 Separar en términos, resolver en orden potencia, multiplicación y división, luego suma y resta.

c) (b + d) / (c + 4) Los paréntesis tienen prioridad para luego realizar la división.

d) (x ^ 2 + y ^ 2) ^ (1 / 2) Se resuelve las potencias de x e y, luego realizar la suma.

**Resolución**

a) b ^ 2 – 4 \* a \* c **b2 – 4.a.c**

3 ^ 2 – 4 \* 2 \* 1

9 – 8 \* 1

9 – 8

1

b) 3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17 **3 x4 – 5 x3 + x 12 – 17**

3 \* 1^ 4 – 5 \* 1^ 3 + 1\* 12 –17

3 \*1 - 5\* 1 + 12 – 17

3 - 5 + 12 – 17

- 2 + 12 – 17

--7

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

(3 + 6) / (2+ 4)

9/6

1.5

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

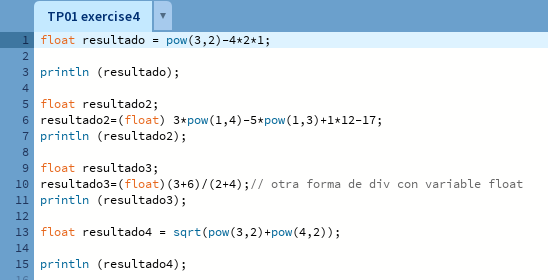
(3 ^ 2 + 4 ^ 2) ^ (1 / 2)

(9+ 16) ^ (1 / 2)

(25) ^ (1 / 2)

5

**Captura de Processing**



**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

**Resolución**

1. 5 \* 4– 5^ 2 / 4 \* 1

20 – 25/ 4 \* 1

20 – 6.25\* 1

20– 6.25

13.75

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

20/9

2.22…

c) (((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

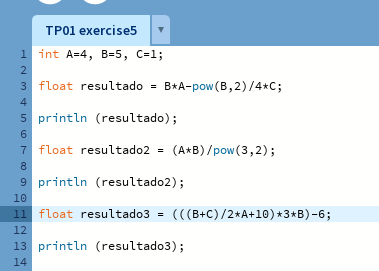
(22 \*3 \*5) – 6

(66\*5) – 6

(330) – 6

324

**Captura de Processing**

****

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

R1 = y+z

R2 = x >= R1

**Resolución**

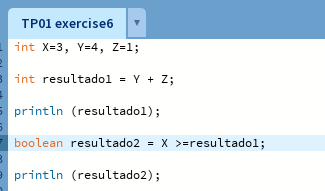
R1 = 4+1

R1 = 5

R2 = 3 >= 5

R2 = 5 False

**Captura de Processing**



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

R1 = ++contador1

R2 = contador1 < contador2

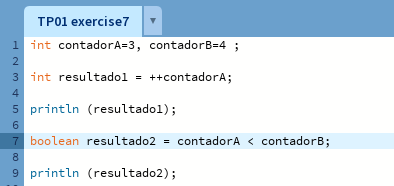
**Resolución**

R1 = ++3

R1 = 4

R2 = 4 < 4 False

**Captura de Processing**



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

a+b-1 < x\*y

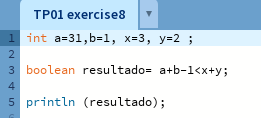
**Resolución**

31+ 1-1 < 3\*2

32-1 < 6

31 < 6 False

**Captura de Processing**

****

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

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

**Resolución**

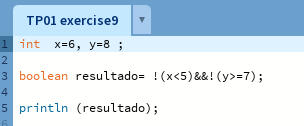
!(6<5)CC !(8>=7)

!(False)CC !(True)

True CC False

False

**Captura de Processing**



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

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

**Resolución**

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

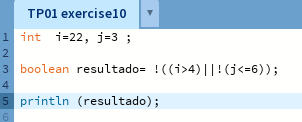
!((True) || !(True))

!((True) || (False))

!(True)

False

**Captura de Processing**



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

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

**Resolución**

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

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

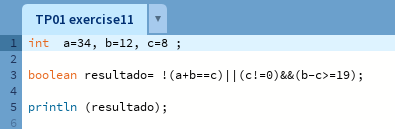
!(False) || (True)CC(False)

(True) || (True)CC(False)

(True) || (False)

True

**Captura de Processing**



**Sección Análisis – Diseño y Codificación de algoritmos – Aplicación de estructuras de**

**control**

Ejercicio 1**2:** 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 un nombre y saludarlo

**Análisis**:

* Datos de Entrada: nombre de la persona
* Datos de Salida: “Buen día ……” …. :nombre de la persona
* Proceso: Unir el mensaje en conjunto con el nombre del usuario

**Diseño**:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Usuario |
| **VARIABLES**  nombreUsuario: String //  mensajeBienvenida: String // |
| **NOMBRE ALGORITMO**: saludar\_al\_usuario  **PROCESO DEL ALGORITMO**   1. Mostrar “Ingrese su nombre: ” 2. Leer nombreUsuario 3. mensajeBienvenida ← “Buen día ” + nombreUsuario 4. Mostrar mensajeBienvenida |

**Captura de 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 perímetro y área de rectángulo a partir de base y altura.

**Análisis**:

* Datos de Entrada: Base, altura: Real.
* Datos de Salida: Perímetro y área del rectángulo: Real.
* Proceso: Ingresar los valores correspondientes a la base y la altura del rectángulo.

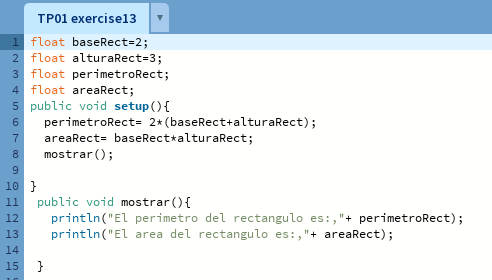
Formula del perímetro: 2\*(baseRect+alturaRect)

Formula del área: baseRect\*alturaRect

**Diseño**:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Rectángulo |
| **VARIABLES**  baseRect: Real// almacena el valor de la base del rectángulo.  alturaRect: Real// almacena el valor de la altura del rectángulo.  perimetroRect: Real// almacena el resultado obtenido en el cálculo del perímetro del rectángulo.  areaRect: Real// almacena el resultado obtenido en el cálculo del área del rectángulo. |
| **NOMBRE ALGORITMO**: calcular\_perimetro\_area  **PROCESO DEL ALGORITMO**   1. *Leer* baseRect 2. *Leer* alturaRect 3. perimetroRect← 2\*(baseRect+alturaRect) 4. areaRect← (baseRect\*alturaRect) 5. Mostrar “El perímetro del rectángulo es:”+perimetroRect 6. *Mostrar* “El área del rectángulo es:”+areaRect |

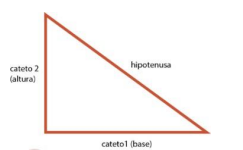
**Captura de Processing**



**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



**Definición del Problema**: Obtener el valor de la hipotenusa dado ala los valores correspondientes a sus catetos.

**Análisis**:

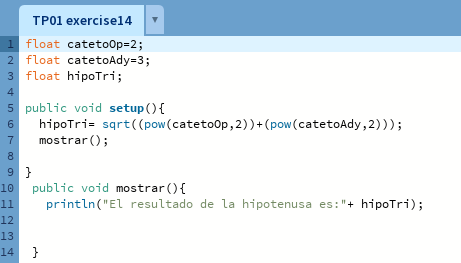
* Datos de Entrada: catetoOp (base), catetoAdy (altura): Real.
* Datos de Salida: hipotenusa: Real.
* Proceso: A partir de los valores respectivamente de los cateto del triángulo. En lo siguiente aplicar el teorema de Pitágoras para conocer el valor de la hipotenusa.

Teorema de Pitágoras: [(catetoOp^2)+(catetoAdy^2)]^1/2

**Diseño**:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Triangulo |
| **VARIABLES**  catetoOp: Real//  catetoAdy: Real//  hipoTri: Real// |
| **NOMBRE ALGORITMO**: calcular\_hipotenusa\_triangulo  **PROCESO DEL ALGORITMO**   1. *Leer* catetoOp 2. *Leer* catetoAdy 3. hipoTri←[(catetoOp^2)+(catetoAdy^2)]^1/2 4. *Mostrar* “El resultado de la hipotenusa es:”+ hipoTri |

**Captura de Processing**



**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.

**Definición del Problema**: Calcular a partir de dos números para obtener el resultado de las operaciones de la suma, resta, multiplicación y división.

**Análisis**:

* Datos de Entrada: numeroA, numeroB: Real
* Datos de Salida: Resultado de la suma, resta, multiplicación, división.
* Proceso: Partiendo de los valores correspondiente a numeroA y numeroB, relizar las siguientes operaciones:

Ecuación de la suma: numeroA+ numeroB

Ecuación de la resta: numeroA- numeroB

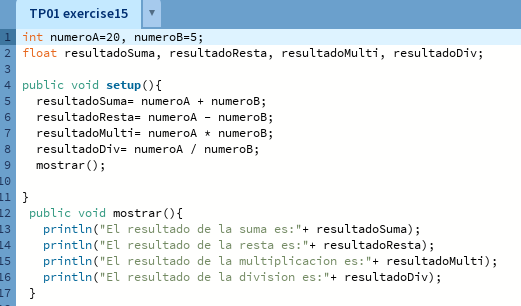
Ecuación de la multiplicación: numeroA\*numeroB

Ecuación de la división: numeroA/ numeroB

**Diseño**:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Matemático |
| **VARIABLES**  numeroA: Int//  numeroB: Int//  resutadoSuma: Int//  resultadoResta: Int//  resultadoMulti: Int//  resultadoDiv: Int// |
| **NOMBRE ALGORITMO**: sumar\_numeros  **PROCESO DEL ALGORITMO**   1. *Leer* numeroA 2. *Leer* numeroB 3. resultadoSuma← numeroA+ numeroB 4. *Mostrar* “El resultado de la suma:”+ resultadoSuma   **NOMBRE ALGORITMO**: restar\_numeros  **PROCESO DEL ALGORITMO**   1. *Leer* numeroA 2. *Leer* nuemroB 3. resultadoResta← numeroA- numeroB 4. *Mostrar* “El resultado de la resta es:”+ resultadoResta   **NOMBRE ALGORITMO**: multiplicar\_numeros  **PROCESO DEL ALGORITMO**   1. *Leer* numeroA 2. *Leer* numeroB 3. resultadoMulti← numeroA\* numeroB 4. *Mostrar* “El resultado de la multiplicacion es:”+ resultadoMulti   **NOMBRE ALGORITMO**: dividir\_numeros  **PROCESO DEL ALGORITMO**   1. *Leer* numeroA 2. *Leer* numeroB 3. resultadoDiv← numeroA/ numeroB 4. *Mostrar* “El resultado de la división es:”+ resultadoDiv |

**Captura de 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**: Convertir una temperatura Fahrenheit en grados Celsius.

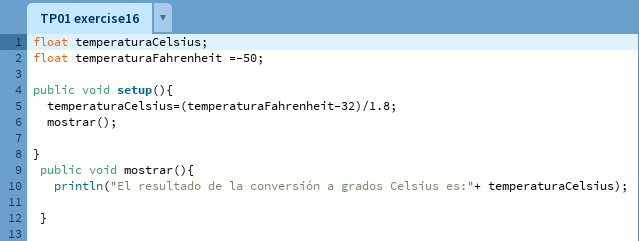
**Análisis**:

* Datos de Entrada: temperaturaFahrenheit: Real.
* Datos de Salida: temperaturaCelsius: Real.
* Proceso: conversión de la temperatura Fahrenheit a grados Celsius.

**Diseño**:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Temperatura |
| **VARIABLES**  temperaturaFahrenheit: Real//  temperaturaCelsius: Real// |
| **NOMBRE ALGORITMO**: convertir\_temperatura\_Fahrenheit  **PROCESO DEL ALGORITMO**   1. *Leer* temperaturaFahrenheit 2. temperaturaCelsius←( temperaturaFahrenheit-32)/1.8 3. *Mostrar* “El resultado de la conversión a grados Celsius es:”+ temperaturaCelsius |

**Captura de 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

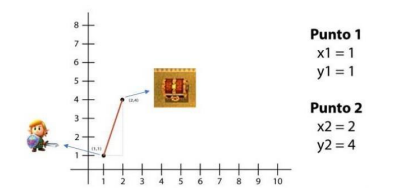
en la posición (𝑥2, 𝑦2). Si observa con detenimiento se observa la conformación de un

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**: Calcular la distancia entre el personaje y el objeto encontrado.

**Análisis**:

* Datos de Entrada: Puntos x1, y1; x2,y2
* Datos de Salida: Distancia entre los puntos
* Proceso: Dado a los puntos: puntoX1, puntoX2, puntoY1, puntoY2.

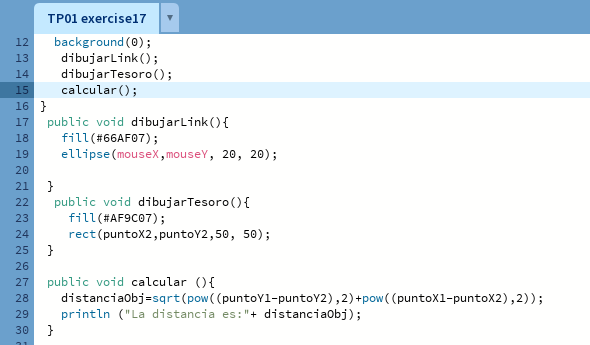
Se debe aplicar el teorema de Pitágoras mediante el cálculo de los catetos correspondiente de la siguiente manera:

Ecuación del teorema de Pitágoras: [(puntoY1-puntoY2)^2+(puntoX1-puntoX2)^2]^1/2

**Diseño**:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: ObjetoEncontrado |
| **VARIABLES**  puntoX1, puntoX2, puntoY1, puntoY2.: Real//  distaciaObj: Real// |
| **NOMBRE ALGORITMO**: encontrar\_distancia\_objeto\_persoanje  **PROCESO DEL ALGORITMO**   1. *Leer* puntoY1 2. *Leer* puntoY2 3. *Leer* puntoX1 4. *Leer* puntoX2 5. distanciaObj← [(puntoY1-puntoY2)^2+(puntoX1-puntoX2)^2]^1/2 6. *Mostrar* distanciaObj |

**Captura de 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 las raíces de una ecuación cuadrática a partir del análisis de su discriminante

**Análisis**:

* Datos de Entrada: númeroA, númeroB, númeroC: Real
* Datos de Salida: mensajeResultante: String
* Proceso: En una ecuación de segundo grado **numeroA\*x^2 numeroB\*x numero\*C**, para determinar los valores de **x** que cortan al eje de las abscisas, se utiliza la fórmula de la ecuación cuadrática:

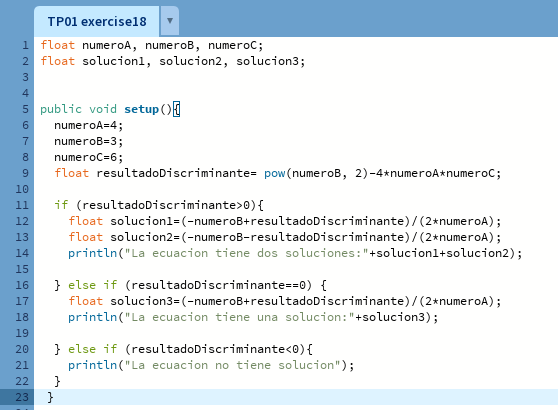
Para identificar cuantas soluciones existen para la ecuación cuadrática, es posible con el uso del discriminante (D): o bien **numeroB^24\*numeroA\*numeroC**

* Si D>0: La ecuación tiene dos soluciones
* Si D=0: La ecuación tiene una sola solución
* Si D<0: La ecuación no tiene solución

**Diseño**:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: AlgoritmoEcuacionCuadratica |
| **VARIABLES**  numeroA, numeroB, numeroC: Real// almacena el valor del número que se ingresó.  resultadoDiscriminante: Real//almacena el valor del resultado de la discriminante. |
| **NOMBRE ALGORITMO**: calcular\_ecuacion\_raices  **PROCESO DEL ALGORITMO**   1. *Leer* numeroA 2. *Leer* numeroB 3. *Leer* numeroC 4. resultadoDiscriminante ← numeroB^24\*numeroA\*numeroC 5. **según\_sea** (resultadoDiscriminante) hacer 6. **caso resultadoDiscriminante>0:** 7. solucion1 ←numeroBresultadoDiscriminante)/2\*numeroA 8. solucion2 ← (numeroBresultadoDiscriminante)/2\*numeroA 9. “La ecuación tiene dos soluciones:”+ solucion1, solucion2 10. sentencia de ruptura 11. **caso resultadoDiscriminante=0:** 12. Solucion3 ← (numeroBresultadoDiscriminante)/2\*numeroA 13. “La ecuación tiene una solución:”+ solucion1 14. sentencia de ruptura 15. **otros resultadoDiscriminante<0:** 16. “La ecuación no tiene solución.” 17. sentencia de ruptura 18. **fin\_según** 19. fin |

**Captura de 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

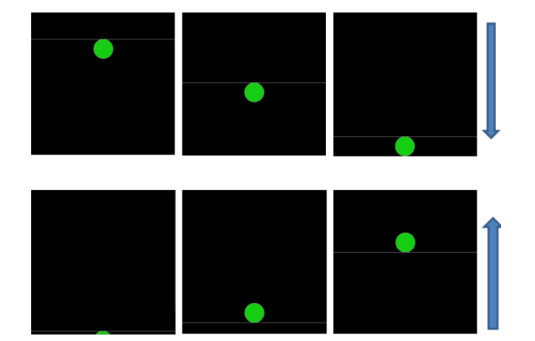
la elipse. Mantenga en cero el valor para background(). Cuando la línea supere la posición

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 una línea y una elipse en el medio de la misma donde las variables se actualicen haciendo que se desplacen de arriba hacia abajo y viceversa a su vez que el circulo sea arrastrado en conjunto.

**Análisis**:

* Datos de Entrada: distancia, incremento

anchoCir, altoCir

* Datos de Salida: Línea y circulo dibujados en el lienzo con desplazamientos.
* Proceso: Actualizar las variables del circulo y la línea de tal manera que se vea el desplazamiento incrementando sus respectivos valores y en caso contrario volviendo negativo su valor.

**Diseño**:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Algoritmo |
| **VARIABLES**  anchoCir, altoCir, distancia, incremento: entero// |
| **NOMBRE ALGORITMO**: dibujar\_circulo\_linea\_desplazamiento  **PROCESO DEL ALGORITMO**  1*.*anchoCir ← 80  2.altoCir ←80  3*.*distancia ← 40  4. incremento ←1  5. y ←y+ incremento  6. **si** ← ((y>=height)||(y<=0)) **entonces**  7. incremento ←incremento\*-1  8. distancia ←distancia\*-1  9. **fin\_si**  10. fin |

**Captura de Processing**

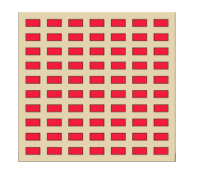


**Ejercicio 20:** Dibuje en toda la extensión del lienzo de (440, 420) rectángulos de idénticas

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 rectángulos en el lienzo utilizando estructuras iterativas.

**Análisis**:

* Datos de Entrada: coordRect: coordenadas cartesianas

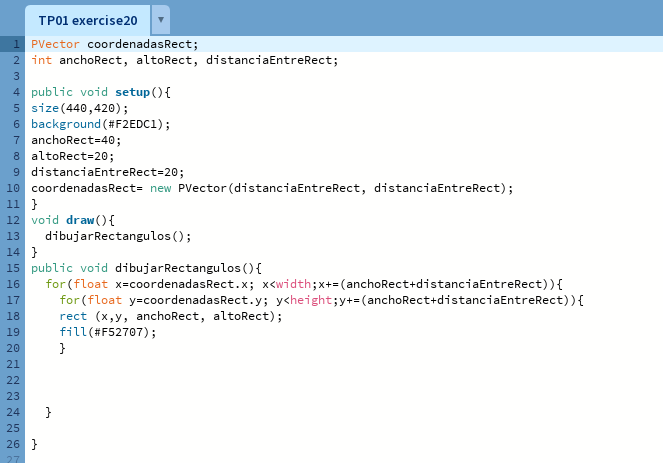
altoRect, anchoRect, distanciaEntreRect

* Datos de Salida: Rectángulos dibujados en el lienzo.
* Proceso: Dibujar los rectángulos.

**Diseño**:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Lienzo |
| **VARIABLES**  coordenadasRect: coordenadas//  anchoRect, altoRect, distanciaEntreRect: entero//  anchoLienzo(width), altoLienzo(height): entero// |
| **NOMBRE ALGORITMO**: dibujar\_rectangulos  **PROCESO DEL ALGORITMO**  1*.*anchoLienzo ← 440  2.altoLienzo ←420  3*.*distanciaEntreRect ← 20  4. anchoRect ←40  5. altoRect ←20  6. **para** x ← coordenadasRect.x **hasta** width con paso (anchoRect+distanciaEntreRect) **hacer**  7. **para** y ← coordenadasRect.y **hasta** height con paso (anchoRect+distanciaEntreRect) **hacer**  8. dibujar rectángulo en (x,y,anchoRect, altoRect)  9. **fin\_para**  10. fin |

**Captura de Processing**



**Ejercicio 21:** Utilizando la estructura de control repetitiva while() dibuje la siguiente imagen

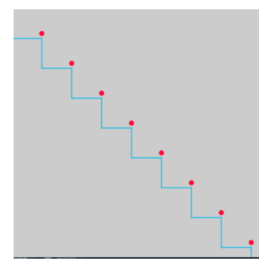
utilizando líneas que forman escalones y sobre cada borde de escalón se dibuje un punto

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 líneas que de tal manera formen escalones y en el borde de cada escalón se dibuje un punto.

**Análisis**:

* Datos de Entrada: x,y

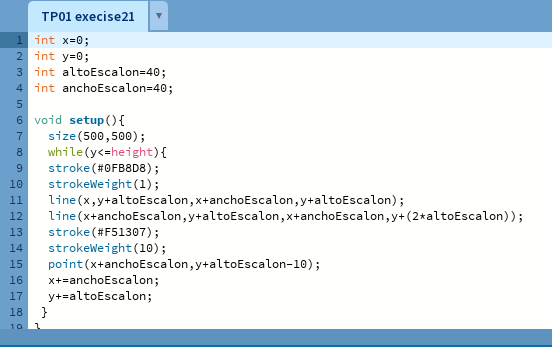
anchoEscalon, altoEscalon

* Datos de Salida: Escalón y puntos dibujados en el lienzo de processing.
* Proceso: Dibujar las líneas en formas escalonada con sus respectivos puntos al borde.

**Diseño**:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Lienzo |
| **VARIABLES**  anchoEscalon, altoEscalon, x, y: entero//  altoLienzo, anchoLienzo: entero// |
| **NOMBRE ALGORITMO**: dibujar\_puntos\_lineas\_escalonada  **PROCESO DEL ALGORITMO**  1*.*anchoLienzo ← 500  2.altoLienzo ←500  3*.* x← 0  4. y←0  5. altoEscalon←40  6. anchoEscalon ←40  7. **mientras** ← (y<=heigh) **hacer**  8. x ←x + anchoEscalon  9. y ←y+ altoEscalon  10. **fin\_mientras**  11. fin |

**Captura de Processing**

****

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

imagen

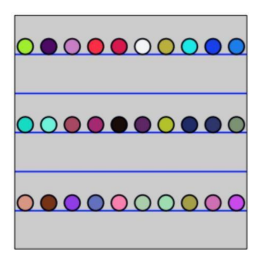
La imagen debe ser construida desde la función setup(). Defina el tamaño del lienzo en

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 líneas y círculos de colores de manera que los círculos queden intercalados por cada línea de por medio.

**Análisis**:

* Datos de Entrada: anchoLienzo, altoLienzo

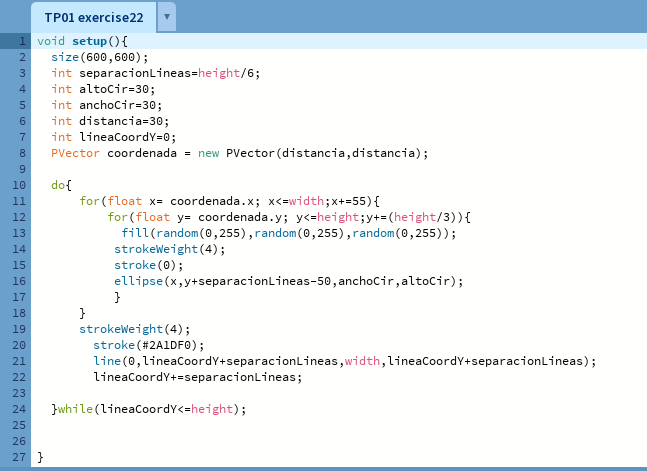
separacionLineas

* Datos de Salida: Líneas y círculos de colores dibujados en el lienzo.
* Proceso: Dibujar las líneas con los círculos de por medio.

**Diseño**:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Lienzo |
| **VARIABLES**  anchoLienzo, altoLienzo, anchoCir, altoCir, lineaCoordY, distancia: entero//  separacionLineas: Real//  coordenada: Coordenadas// |
| **NOMBRE ALGORITMO**: dibujar\_lineas\_circulos  **PROCESO DEL ALGORITMO**  1*.*anchoLienzo ← 600  2.altoLienzo ←600  3*.*anchoCir ← 30  4. altoCir ←30  5. distancia ←30  6. lineaCoordY ← 0  7. **hacer**  8. **para** x ← coordenada.x **hasta** width con paso (x+=55) **hacer**  9. **para** y ← coordenada.y **hasta** height con paso (y+=(height/3) **hacer**  10. dibujar ellipse (x,y+separacionLineas-50, anchoCir, altoCir)  11. **fin\_para**  12. dibujar línea en ( 0, lineaCoordY +separacionLineas,width,lineaCoordY+separacionLineas)  13. lineaCoordY←lineaCoordY+ separacionLineas  14. **mientras** ←(lineaCoordY<=height)  15. fin |

**Captura de Processing**



Conclusión

En un principio me resulto un tanto complejo poder ordenar y comenzar a organizar como o por donde debería empezar a resolver los ejercicios del trabajo practico, pero con los apuntes proporcionados en la cátedra (textos, videos, cuestionarios), pude entender y clarificar ciertas ideas haciendo que el proceso para el inicio de la resolución se fuera alivianado en el recorrido, tanto en el diseño de análisis de la problematización de cada ejercicio como en la práctica de programación en processing.

Aun así, todo el proceso fue un ir y venir debido a que encontraba ciertos errores o veía que podía aplicar otra forma de resolución en processing para ponerlo en práctica.

Bibliografía

\_Apuntes de catedra: archivos pdf “Problema y solución”, “Programación”, “Fases en la resolución de problemas mediante algoritmos “,” Expresiones aritméticas”, “Bifurcaciones” e “Estructuras iterativas”.

\_Videos de YouTube en relación al tema de programación con processing.