|  |  |  |
| --- | --- | --- |
| 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

Apellido y Nombre: Oscar Cesar Ugarte

L.U: 0733

*Profesor:*

*Mg. Ing. Ariel Alejandro Vega*

*Año 2024*

**FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS**

Indice

Ejercicio Página

Ejercicio 01 …………………………………………………………………………………………………… 03

Ejercicio 02 …………………………………………………………………………………………………… 03

Ejercicio 04 …………………………………………………………………………………………………… 03

Ejercicio 05 …………………………………………………………………………………………………… 04

Ejercicio 06 …………………………………………………………………………………………………… 05

Ejercicio 07 …………………………………………………………………………………………………… 05

Ejercicio 08 …………………………………………………………………………………………………… 06

Ejercicio 09 …………………………………………………………………………………………………… 06

Ejercicio 10 …………………………………………………………………………………………………… 07

Ejercicio 11 …………………………………………………………………………………………………… 07

Ejercicio 12 …………………………………………………………………………………………………… 07

Ejercicio 13 …………………………………………………………………………………………………… 08

Ejercicio 14 …………………………………………………………………………………………………… 09

Ejercicio 15 …………………………………………………………………………………………………… 11

Ejercicio 16 …………………………………………………………………………………………………… 13

Ejercicio 18 …………………………………………………………………………………………………… 14

Ejercicio 20 …………………………………………………………………………………………………… 17

Ejercicio 21 …………………………………………………………………………………………………… 18

Ejercicios

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

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

3 \* 2 – 4 \* 5 / 2 ^ 2 = 1

6 – 4 \* 5 / 4

6 – 20 /4

6 – 5 = 1

Captura del Código de Processing

Texto

Descripción generada automáticamente

**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 = 1

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

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

1 + 3 / 25 / 4 \* 2

1 + 0 / 4 \* 2

1 + 0 \* 2

1

Puede dar 1 o 1,06 dependiendo si se guarda en una variable de tipo entero o real.

Captura del Código de Processing

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

**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

𝑏² − 4. 𝑎. 𝑐

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

3x⁴ – 5x³ + X . 12 - 17

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

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

**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

5 \* 4 – 5 ^ 2 / 4 \* 1 = 14

20 – 25 / 4 \* 1

20 – 6 \* 1

14

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

(4 \* 5) / 3 ^ 2

20 / 9

2

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

((( 5 + 1 ) / 2 \* 4 + 10 ) \* 3 \* 5 ) - 6 = 324

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

(( 12 +10 ) \* 3 \*5 ) - 6

( 22 \* 3 \* 5 ) - 6

330 – 6

324

Captura del Código de Processing

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

**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

Captura del Código de Processing

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

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

R1 = ++contador1

R1 = 4

R2 = contador1 < contador2

R2 = 4 < 4

R2 = falso

Captura del Código de Processing

Imagen que contiene Texto

Descripción generada automáticamente

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

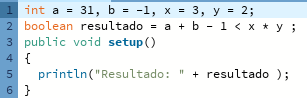
a+b-1 < x \* y

31 + (-1) – 1 < 3 \* 2

29 < 6

falso

Captura del Código de Processing



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

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

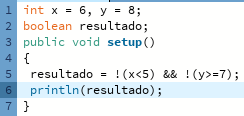
!(6 < 5) && !(8 >= 7)

!falso && !verdadero

verdadero && falso

falso

Captura del Código de Processing



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

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

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

!( verdadero || ! verdadero )

!( verdadero || falso )

! verdadero

Falso

Captura del Código de Processing



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

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

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

!(46==8) || verdadero && (4 >= 19)

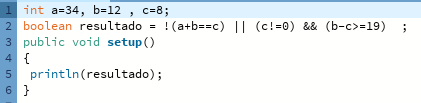
!falso || verdadero && falso

verdadero ||verdadero && falso

verdadero || falso

verdadero

Captura del Código de Processing



**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.

Fase de Análisis

Especificación del problema: Mostrar un mensaje de saludo con el nombre indicado.

Análisis:

Datos de Entrada:

nombre : cadena de texto

Datos de Salida:

Un mensaje de saludo con el nombre del usuario.

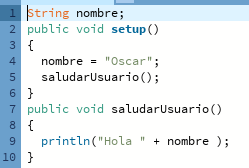
Proceso:

Juntar el nombre del usuario con un mensaje de saludo y después mostrarlo por pantalla.

Fase de Diseño

|  |
| --- |
| Entidad que resuelve el problema: Persona que escribe |
| Variables:  nombre : cadena de texto. |
| Nombre del Algoritmo: saludarUsuario  Proceso del Algoritmo:  Inicio  1. Leer nombre  2. Escribir “Hola ” + nombre  Fin |

Codificación



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

Fase de Análisis

Especificación del problema: Calcular el perímetro y área de un rectángulo.

Análisis:

Datos de Entrada:

base, altura : entero.

Datos de Salida: Un mensaje que muestre el perímetro y el área del rectángulo.

Proceso:

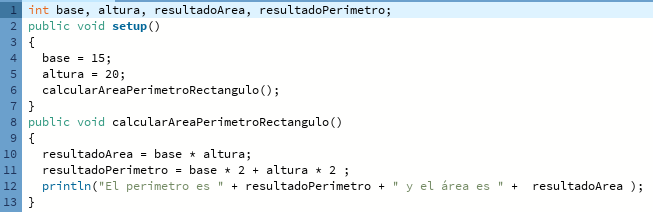
Calcular el perímetro del rectángulo ( base \* 2 + altura \* 2 ) y guardarlo en una variable.

Calcular el área del rectángulo ( base \* altura ) y guardarlo en una variable.

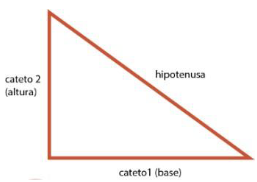
Fase de Diseño

|  |
| --- |
| Entidad que resuelve el problema: Calculadora de área y perímetro del rectángulo |
| Variables:  base, altura, resultadoArea, resultadoPerimetro : entero |
| Nombre del Algoritmo: calcularAreaPerimetroRectangulo  Proceso del Algoritmo:  Inicio  1. Leer altura  2. Leer base  3. resultadoArea <- base \* altura  4. resultadoPerimetro <- base \* 2 + altura \* 2  5. Escribir “El perímetro es ” + resultadoPerimetro + “ y el área es “ + resultadoArea  Fin |
|  |

Codificación



**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.



Fase de Análisis

Especificación del problema: Obtener la hipotenusa de un triángulo rectángulo conociendo sus catetos.

Análisis:

Datos de Entrada:

cateto1, cateto2 : entero

Datos de Salida:

La medida de la hipotenusa del triángulo.

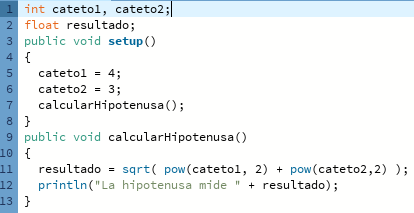
Proceso

Calcular la hipotenusa con la formula (cateto1^2 + cateto2^2) ^(1/2) // A la suma de las potencias cuadrada de los catetos se calcula su raíz cuadrada.

Fase de Diseño

|  |
| --- |
| Entidad que resuelve el problema: Calculadora de hipotenusa de triángulos |
| Variables:  cateto1, cateto2 : entero  resultado : real |
| Nombre del Algoritmo: calcularHipotenusa  Proceso del Algoritmo:  Inicio  1. Leer cateto1  2. Leer cateto2  3. resultado <- (cateto1^2 + cateto2^2) ^ (1/2)  4. Escribir “La hipotenusa mide: ” resultado  Fin |

Codificación



**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.

Fase de Análisis

Especificación del problema: Calcular la suma, resta, multiplicación y división de dos numeros.

Análisis:

Datos de Entrada:

numero1, numero2 : entero.

Datos de Salida: Cuatro mensajes, cada uno mostrando la suma, resta, multiplicación y división.

Proceso:

Sumar 2 números y guardarlos en una variable.

Restar 2 números y guardarlos en una variable.

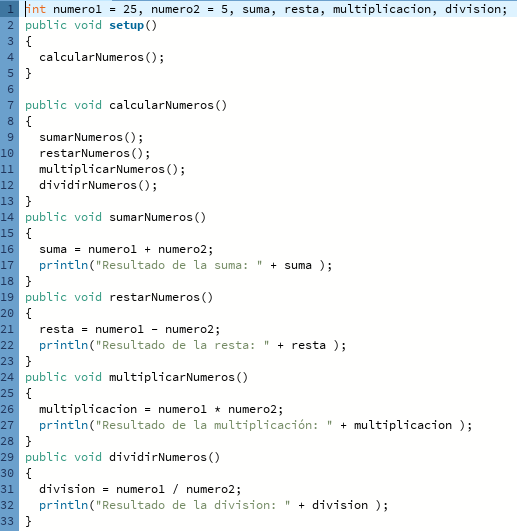
Multiplicar 2 números y guardarlos en una variable.

Dividir 2 números y guardarlos en una variable.

Fase de Diseño

|  |
| --- |
| Entidad que resuelve el problema: Calculadora de dos números |
| Variables:  numero1, numero2, suma, resta, multiplicacion, division : entero |
| Nombre del Algoritmo: calcularNumeros  Proceso del Algoritmo:  Inicio  1. numero1 <- 25  2. numero2 <- 5  3. sumarNumeros()  4. restarNumeros()  5. multiplicarNumeros()  6. dividirNumeros()  Fin  Nombre del Algoritmo: sumarNumeros  Proceso del Algoritmo:  Inicio  1. suma <- numero1 + numero2  2. Escribir “Resultado de la suma: ” + suma  Fin  Nombre del Algoritmo: restarNumeros  Proceso del Algoritmo:  Inicio  1. resta <- numero1 - numero2  2. Escribir “Resultado de la resta: ” + resta  Fin  Nombre del Algoritmo: multiplicarNumeros  Proceso del Algoritmo:  Inicio  1. multiplicacion <- numero1 \* numero2  2. Escribir “Resultado de la multiplicación: ” + multiplicacion  Fin  Nombre del Algoritmo: dividirNumeros  Proceso del Algoritmo:  Inicio  1. division <- numero1 / numero2  2. Escribir “Resultado de la division: ” + division  Fin |

Codificación



**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

**temperaturaCelcius = (temperaturaFahrenheit – 32 ) / 1.8**

Fase de Análisis

Especificación del problema: Obtener la temperatura en grados Celcius conociendo el valor en grados Fahrenheit.

Análisis:

Datos de Entrada:

gradosFahrenheit: real

Datos de Salida:

Un mensaje con el valor en grados celcius.

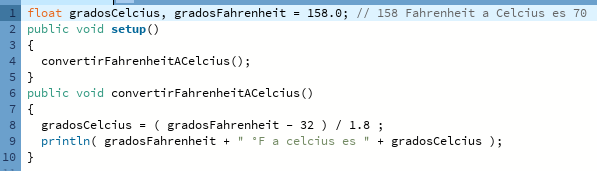
Proceso:

gradosCelcius <- ( gradosFahrenheit - 32 ) / 1.8 // Para obtener los grados Celsius se aplica la formula del ejercicio.

Fase de Diseño

|  |
| --- |
| Entidad que resuelve el problema: Convertidor de grados Fahrenheit a Celcius |
| Variables:  gradosCelcius, gradosFahrenheit : real |
| Nombre del Algoritmo: convertirFahrenheitACelcius  Proceso del Algoritmo:  Inicio  1. gradosFahrenheit <- 158 // 158 Fahrenheit a Celcius es 70  2. gradosCelcius <- ( gradosFahrenheit - 32 ) / 1.8  3. Escribir gradosFahrenheit + “ °F a celcius es “ + gradosCelcius  Fin |

Codificación



**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.

Fase de Análisis

Especificación del problema: Obtener uno o los dos resultados correspondientes de una ecuación cuadrática.

Análisis:

Datos de Entrada:

a, b, c, tipoDiscriminante : entero //El tipoDiscriminante puede tomar valores de 1, 0 o -1. Se inicializa en 0 para evitar preguntar después por el caso de 0.

discriminante, resultado1, resultado2 : real

Datos de Salida:

Uno de cuatro mensajes que puede ser:

1. Se produjo una división por cero.
2. Tiene 2 soluciones reales distintas y debe mostrar los 2 resultados.
3. Tiene una única solución real y debe mostrar ese resultado.
4. Ninguna de las soluciones son números reales.

Proceso:

Primero se asigna los valores de “a”, “b” y “c” y se revisa que “a” no sea 0. Ya que se produciría una división por cero y se mostrar un mensaje avisando.

Si pasa la validación a la variable tipoDiscriminante se le asigna 0 y se calcula el valor de discriminante.

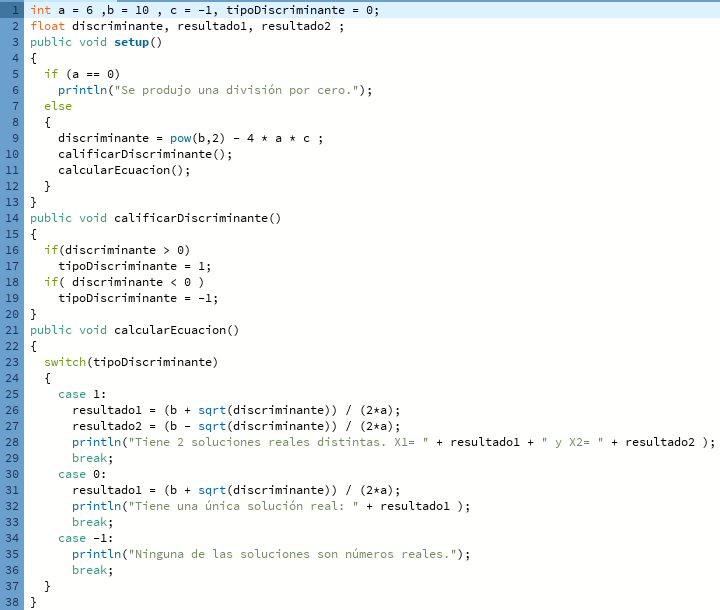
Después se consulta si la discriminante es positiva se le asigna 1 a la variable tipoDiscriminante. Si no es positivo se pregunta si el valor de discriminante es negativo y si se cumple se asigna -1 a la variable tipoDiscriminante. No se pregunta por 0 ya que la variable tipoDiscriminante se inicializó en 0.

Entonces se usa una estructura según para los casos que la determinante diera positivo, cero o negativo. En cada una se realizará el calculo correspondiente para ser mostrado en pantalla.

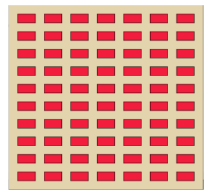
Fase de Diseño

|  |
| --- |
| Entidad que resuelve el problema: Calculadora de Ecuación Cuadratica |
| Variables:  a, b, c, tipoDicreminante : entero //El tipoDicreminante puede tomar valores de 1, 0 o -1. Se inicializa en 0 para evitar preguntar después por 0.  discriminante, resultado1, resultado2 : real |
| Nombre del Algoritmo: calcular  Proceso del Algoritmo:  Inicio  1. a <- 6  2. b <- 10  3. c <- -1  4. **si** ( a == 0 ) **entonces**  5. Escribir “Se produjo una división por cero.”  6. **si\_no**  7. tipoDiscriminante <- 0 // Se inicializa en 0 para no preguntar después por el caso cero.  8. discriminante <- b^2 – 4 \* a \* c  9. calificarDiscriminante()  10. calcularEcuacion()  11. **fin\_si**  Fin |
| Nombre del Algoritmo: calificarDiscriminante  Proceso del Algoritmo:  Inicio  1.  **si** (discriminante > 0) **entonces**  2. tipoDiscriminante <- 1  3. **fin\_si**  4.  **si** (discriminante < 0) **entonces**  5. tipoDiscriminante <- -1  6. **fin\_si**  Fin |
| Nombre del Algoritmo: calcularEcuacion  Proceso del Algoritmo:  Inicio  1. **según\_sea** (tipoDiscriminante) **hacer**  2. **caso 1:**  3. resultado1 = ( b + ( discriminante ^ (1/2) )) / (2\*a)  4. resultado2 = ( b - ( discriminante ^ (1/2) )) / (2\*a)  5. Escribir “Tienes 2 soluciones reales distintas. X1= ” + resultado1 + “ y x2= ” + resultado2  6. sentencia de ruptura  7. **caso 0:**  8. resultado1 = ( b + ( discriminante ^ (1/2) )) / (2\*a)  9. Escribir “Tiene una única solución real: ” + resultado1  10. sentencia de ruptura  11. **caso -1:**  12. Escribir “Ninguna de las soluciones son números reales.”  13. sentencia de ruptura  14. **fin\_segun**  Fin |

Codificación

****

**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:



Fase de Análisis

Especificación del problema: Dibujar en el lienzo rectángulos usando estructuras iterativas.

Análisis:

Dato de Entrada:

coordenadasRectangulo: coordenadas-cartesianas

ancho, alto, distanciaEntreRect, anchoLienzo, altoLienzo : entero

Datos de Salida:

Los rectángulos dibujados.

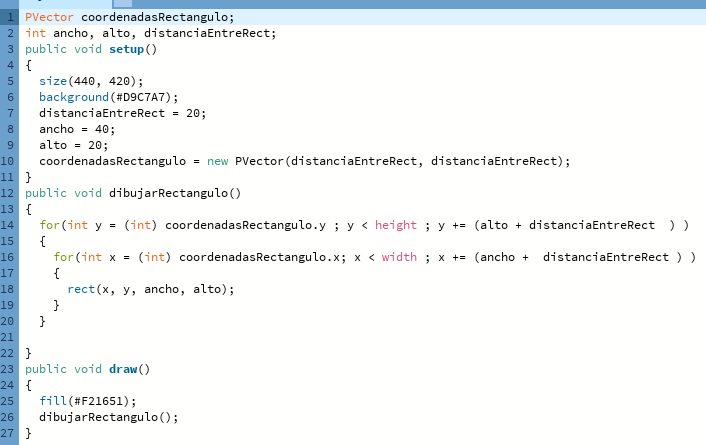
Proceso:

Dibujar los rectángulos.

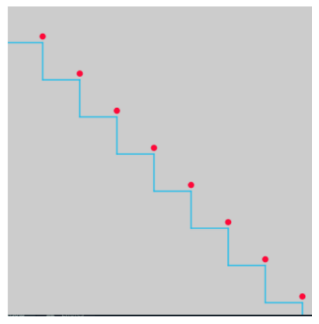
Fase de Diseño:

|  |
| --- |
| Entidad que resuelve el problema: Lienzo |
| Variables:  coordenadasRectangulo : coordenadas  ancho, alto, distanciaEntreRect, anchoLienzo, altoLienzo : entero |
| Nombre del Algoritmo: dibujarRectangulos  Proceso del Algoritmo:  Inicio  1. anchoLienzo <- 440  2. altoLienzo <- 420  3. distanciaEntreRec <- 20  4. ancho <- 40  5. alto <- 20  6. coordenadasRectangulo <- new Pvector (distanciaEntreRect, distanciaEntreRect)  7. **para** y **<-** coordenadasRectangulo.y **hasta** altoLienzo **incremento** (alto + distanciaEntreRec) **hacer**  8. **para** x **<-** coordenadasRectangulo.x **hasta** anchoLienzo **incremento** (ancho + distanciaEntreRec) **hacer**  9. dibujar un rectángulo(x,y) con dimensiones ancho y alto  10. **fin\_para**  11**. fin\_para**  Fin |

Codificación



**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.

Fase de Análisis

Especificación del problema: Dibujar escalones sobre el lienzo y colocar un punto rojo sobre cada escalón.

Análisis:

Datos de entrada:

puntoA, puntoB, puntoC, puntoD : coordenadas cartesianas en 2d

distLinea, altoLienzo, anchoLienzo : entero

Datos de salida:

El dibujo de la línea horizontal correspondiente a la escalera.

El dibujo de la línea vertical correspondiente a la escalera.

El dibujo del punto rojo sobre los escalones.

Proceso:

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

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

Dibujar un punto en la siguiente posición x = posición de x de B, y = posición en y de B – 5 unidades.

Actualizar las coordenadas del puntoA con las del puntoC.

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

Fase de Diseño:

|  |
| --- |
| Entidad que resuelve el problema: Escalon |
| Variables:  puntoA, puntoB, puntoC, puntoD : coordenadas cartesianas en 2d.  distLinea, altoLienzo, anchoLienzo : entero |
| Nombre del Algoritmo: setup  Proceso del Algoritmo:  Inicio  1. altoLienzo <- 500  2. anchoLienzo <- 500  3. distLinea <- 60  4. puntoA = new PVector(0, distLinea);  5. **mientras**(puntoA.y < altoLienzo ) **hacer**  6. dibujarEscalon()  7. actualizarCoordenadasA()  8. **fin\_mientras**  Fin |
| Nombre del Algoritmo: dibujarEscalon  Proceso del Algoritmo:  Inicio  1. Dibujar una línea horizontal entre los puntos A y B, con distancia distlinea.  2. Dibujar una línea vertical entre los puntos B y C, con distancia distLinea.  3. dibujarPunto()  Fin |
| Nombre del Algoritmo: dibujarPunto  Proceso del Algoritmo:  Inicio  1. Dibujar un punto en la siguiente posición: x = posición en x de B, y = posición en y de B – 5 unidades.  Fin |
| Nombre del Algoritmo: actualizarCoordenadasA  Proceso del Algoritmo:  Inicio  1. puntoA.x <- puntoC.x  2. puntoA.y <- puntoC.y  Fin |

Codificación

