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

*Profesor:*

*Mg. Ing. Ariel Alejandro Vega*

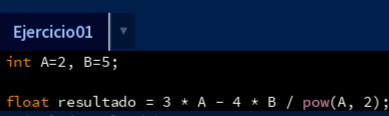
*Año 2024*

Trabajo Práctico N°1

GASPAR, Bruno Ramiro  
TUV000610

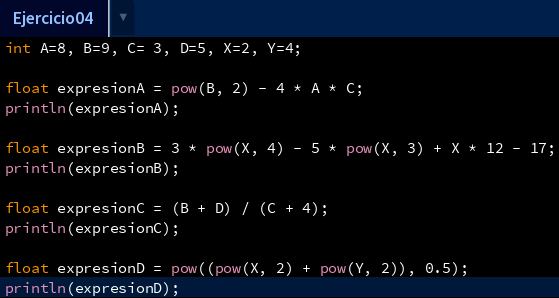
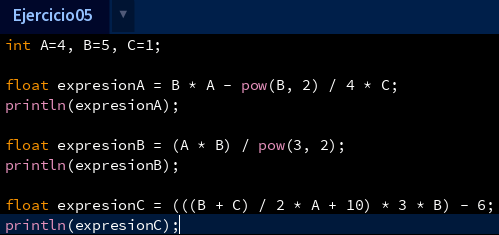
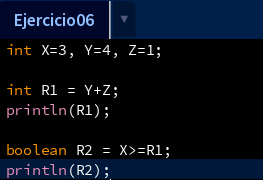
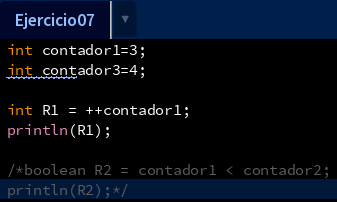
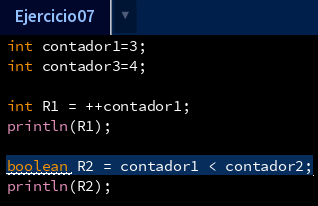
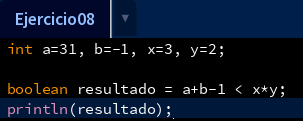
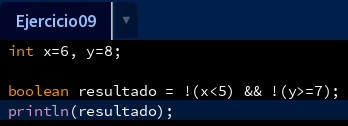
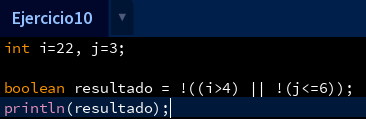
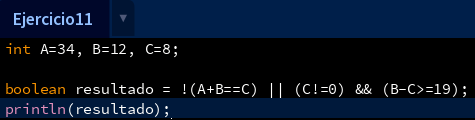
**FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS**

**Indice**  
  
**Sección Expresiones aritméticas y lógicas ………………………………………………………… Pág 3**  
 **Sección Análisis – Diseño y Codificación de algoritmos – Aplicación de estructuras de control …………………………………………………………………………………………………………… Pág 9**

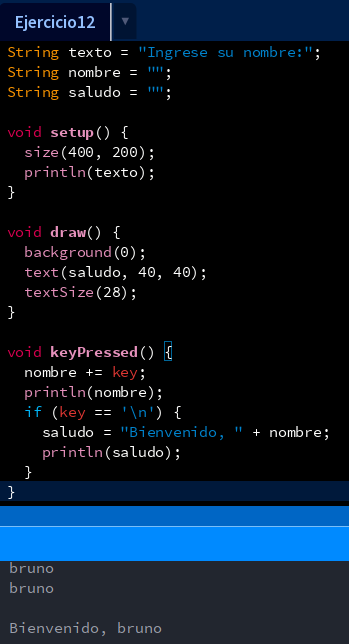
**SECCIÓN EXPRESIONES ARITMÉTICAS Y LÓGICAS**  
  
**Ejercicio 1:** Evaluar(obtener resultado) la siguiente expresión para A=2 y B=5  
  
3 \* A – 4 \* B / A ^ 2  
---------------------------  
(3 \* A) – (4 \* B / (A ^ 2))  
6 – (4 \* B / 4)  
6 – 5  
1  
  
  
  
**Ejercicio 2:** Evaluar la siguiente expresión  
  
4 / 2 \* 3 / 6 + 6 / 2 / 1 / 5 ^ 2 / 4 \* 2  
-----------------------------------------------  
(4 / 2 \* 3 / 6) + (6 / 2 / 1 / (5 ^ 2) / 4 \* 2)  
1 + (6 / 2 / 1 / 25 / 4 \* 2)  
1 + 0,06  
1,06  
  
  
  
**Ejercicio 4:** Evaluar las siguientes expresiones aritméticas, para lo cuál indicar en el caso de las variables, el valor indicado. **Luego escribirlas como expresiones algebraicas.**  
  
A=8, B=9, C=3, D=5, X=2, Y=4

a) B ^ 2 – 4 \* A \* C  
-------------------------  
(B ^ 2) – (4 \* A \* C)  
81 – 96  
–15  
-------------------------  
B2 – 4 x A x C  
  
b) 3 \* X ^ 4 – 5 \* X ^ 3 + X \* 12 – 17  
------------------------------------------------------  
(3 \* (X ^ 4)) – (5 \* (X ^ 3)) + (X \* 12) – 17  
(3 \* 16) – (5 \* 8) + 24 – 17  
48 – 40 + 24 – 17  
72 – 57  
15  
------------------------------------------------------  
3 x X4 – 5 x X3 + X x 12 – 17  
  
c) (B + D) / (C + 4)  
14 / 7  
2  
-----------------------  
B + D

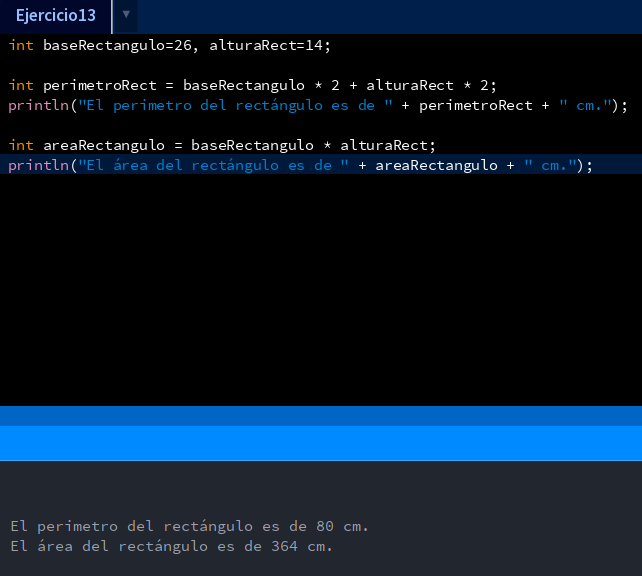
C + 4  
  
d) (X ^ 2 + Y ^ 2) ^ (1/2)  
-------------------------------  
((X ^ 2) + (Y ^2)) ^ (1/2)  
(4 + 16) ^ 0.5

20 ^ 0.5  
4.472136  
-------------------------------  
2√(X2 + Y2)  
  
  
  
  
  
  
**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 ^ 2) / 4 \* C)  
20 – (25 / 4 \* C)  
20 – 6.25  
13.75  
  
b) (A \* B) / 3 ^ 2  
----------------------  
(A \* B) / (3 ^ 2)  
20 / 9  
2.2222223  
  
c) (((B + C) / 2 \* A + 10) \* 3 \* B) – 6  
----------------------------------------------  
((((B+C) / 2 \* A) + 10) \* 3 \* B) – 6   
(((6 / 2 \* A) + 10) \* 3 \* B) – 6  
((12 + 10) \* 3 \* B) – 6  
(22 \* 3 \* B) – 6  
330 – 6  
324  
  
  
  
**Ejercicio 6:** Para X=3, Y=4, Z=1, evaluar el resultado de:  
  
R1 = Y + Z  
R1 = 5  
  
R2 = X >= R1  
R2 = Falso  
  
  
  
**Ejercicio 7:** Para contador1=3 y contador3=4, evaluar el resultado de:  
  
R1 = ++contador1  
R1 = 4  
  
  
  
  
R2 = contador1 < contador2 🡪 No se puede realizar porque “contador2” no existe  
  
  
  
**Ejercicio 8:** Para a=31, b=-1, x=3, y=2, evaluar el resultado de:  
  
a + b - 1 < x \* y  
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))  
!(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)  
!(Falso) || Verdadero CC Falso  
Verdadero || Falso  
Verdadero  
  
  
  
**Sección Análisis – Diseño y Codificación de algoritmos – Aplicación de estructuras de control  
  
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:** Pedir por teclado al usuario un nombre y posteriormente realizar la presentación en pantalla de un saludo con el nombre indicado.  
  
**Análisis:**   
 Datos de Entrada:   
 texto: String  
 nombre: String  
   
 Datos de Salida:  
 saludo: String  
  
 Proceso:  
 ¿Quién debe realizar el proceso? El programa  
 ¿Cuál es el proceso que realiza? saludo: “Bienvenido, “ +  
 nombre  
  
 **Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** El programa |
| **VARIABLES**  texto, nombre: String  saludo: String |
| **NOMBRE ALGORITMO:** saludar\_usuario  **PROCESO DEL ALGORITMO**  texto 🡨 “Ingrese su nombre:“  *Mostrar* texto  nombre 🡨 nombre + *Tecla  Mostrar* nombre  saludo 🡨 “Bienvenido, “ + nombre  *Mostrar* saludo |

**Codificación:**  
 **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 un rectángulo dada su base y su altura.  
  
**Análisis:**  
 Datos de Entrada: baseRectangulo, alturaRect: Entero  
 Datos de Salida: perimetroRect, areaRectangulo: Entero  
 Proceso: ¿Quién debe realizar el proceso? El programa  
 ¿Cuál es el proceso que realiza? Lee la base y la altura  
 del rectángulo para realizar las operaciones   
 correspondientes  
  
**Diseño:**

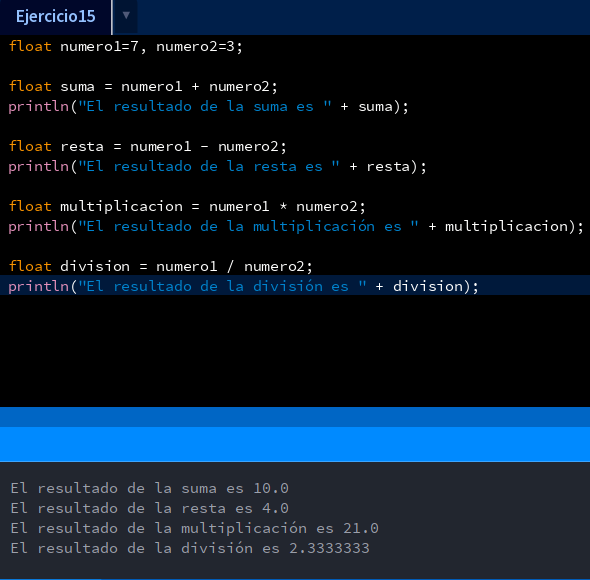
|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** El programa |
| **VARIABLES**  baseRectangulo, alturaRect: Entero  perimetroRect, areaRectangulo: Entero |
| **NOMBRE ALGORITMO:** calcular\_perimetro **PROCESO DEL ALGORITMO**  *Leer* baseRectangulo  *Leer* alturaRect  perimetroRect 🡨 (baseRectangulo \* 2) + (alturaRect \* 2)  *Mostrar* “El perímetro del rectángulo es de “ + perimetroRect +  “ cm.”  **NOMBRE ALGORITMO:** calcular\_area **PROCESO DEL ALGORITMO**  *Leer* baseRectangulo  *Leer* alturaRect  areaRectangulo 🡨 baseRectangulo \* alturaRect  *Mostrar* “El área del rectángulo es de “ + areaRectangulo +  “ cm.” |

**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.  
  
**Definición del problema:** Obtener la hipotenusa de un triángulo rectángulo conociendo sus catetos.  
  
**Análisis:**   
 Datos de Entrada: baseTriangulo, altTriangulo: Real  
 Datos de Salida: hipotenusa: Real  
 Proceso: ¿Quién debe realizar el proceso? El programa  
 ¿Cuál es el proceso que realiza? Lee la base y la altura del  
 triángulo para luego realizar el teorema de Pitágoras  
  
**Diseño:**

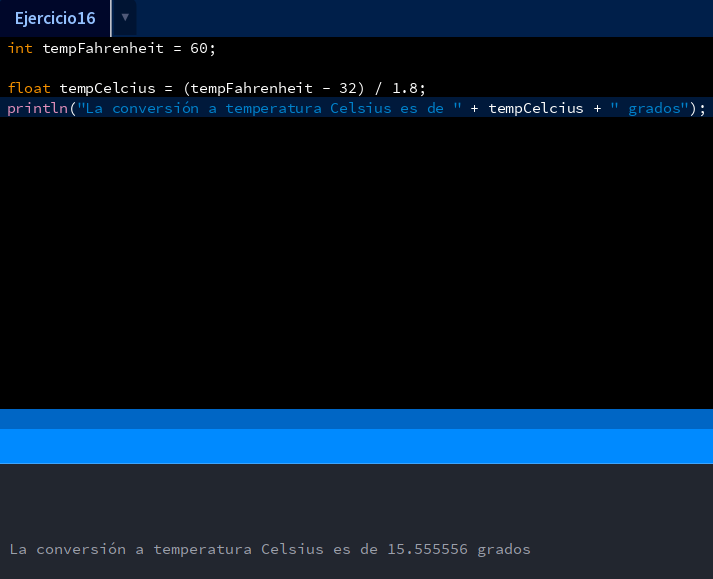
|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** El programa |
| **VARIABLES** baseTriangulo, altTriangulo: Real  hipotenusa: Real |
| **NOMBRE ALGORITMO:** calcular\_hipotenusa **PROCESO DEL ALGORITMO** *Leer* baseTriangulo  *Leer* altTriangulo  hipotenusa 🡨 ((baseTriangulo ^ 2) + (altTriangulo ^ 2)) ^ 0.5  *Mostrar* “La hipotenusa es de “ + hipotenusa + “ cm.” |

**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  
  
**Definición del problema:** Calcular la suma, resta, multiplicación y division entre 2 números  
  
**Análisis:** Datos de Entrada: numero1, numero2: Real  
 Datos de Salida: suma, resta, multiplicacion, division: Real  
 Proceso: ¿Quién debe realizar el proceso? El programa  
 ¿Cuál es el proceso que realiza? Lee los 2 números y  
 realiza la operación correspondiente  
  
**Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** El programa |
| **VARIABLES**  numero1, numero2: Real  suma, resta, multiplicacion, division: Real |
| **NOMBRE ALGORITMO:** resultado\_suma **PROCESO DEL ALGORITMO**  *Leer* numero1  *Leer* numero2  suma 🡨 numero1 + numero2  *Mostrar* “El resultado de la suma es “ + suma  **NOMBRE ALGORITMO:** resultado\_resta **PROCESO DEL ALGORITMO**  *Leer* numero1  *Leer* numero2  resta 🡨 numero1 – numero2  *Mostrar* “El resultado de la resta es “ + resta  **NOMBRE ALGORITMO:** resultado\_multiplicacion **PROCESO DEL ALGORITMO** *Leer* numero1  *Leer* numero2  multiplicacion 🡨 numero1 \* numero2  *Mostrar* “El resultado de la multiplicación es “ + multiplicacion  **NOMBRE ALGORITMO:** resultado\_division **PROCESO DEL ALGORITMO** *Leer* numero1  *Leer* numero2  division 🡨 numero1 / numero2  *Mostrar* “El resultado de la división es “ + division |

**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: temperaturaCelsius = (temperaturaFahrenheit – 32) / 1.8  
  
**Definición del problema:** Convertir una temperature Fahrenheit en grados Celsius.  
  
**Análisis:** Datos de Entrada: tempFahrenheit: Entero  
 Datos de Salida: tempCelsius: Real  
 Proceso: ¿Quién debe realizar el proceso? El programa  
 ¿Cuál es el proceso que realiza? Lee la temperatura que  
 necesitamos convertir antes de hacer la conversion  
  
**Diseño:**

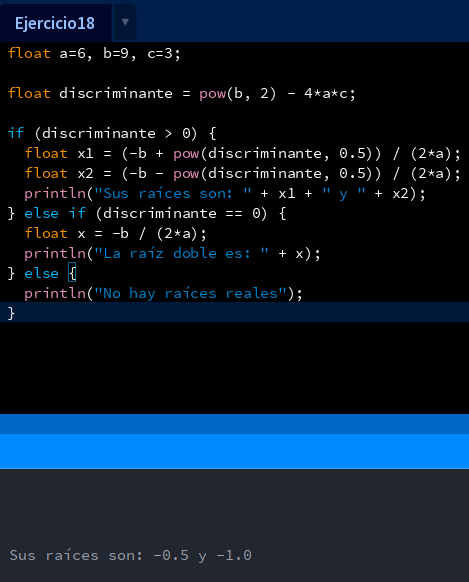
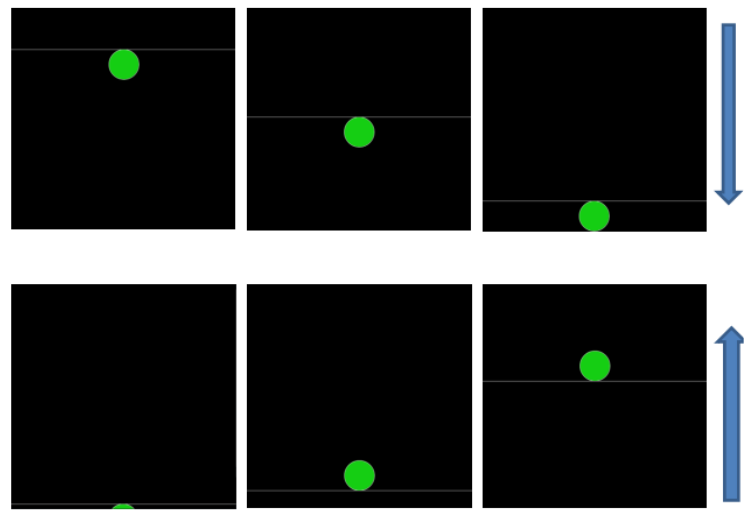
|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** El programa |
| **VARIABLES**  tempFahrenheit: Entero  tempCelsius: Real |
| **NOMBRE ALGORITMO:** conversion\_temperatura **PROCESO DEL ALGORITMO**  *Leer* tempFahrenheit  tempCelsius 🡨 (tempFahrenheit – 32) / 1.8  *Mostrar* “La conversión a temperatura Celsius es de “ +   tempCelsius + “ grados” |

**Codificación:  
  
  
  
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:** Aplicar el teorema de Pitágoras para obtener la distancia entre Link y el tesoro, y así desaparezca una vez que estén a determinada cercanía.  
  
**Análisis:** Datos de Entrada: x1, x2, y1, y2, distanciaTesoro: Entero  
 Datos de Salida: coordX, coordY, distancia: Reales  
 coordLink: String  
 Proceso: ¿Quién debe realizar el proceso? El usuario  
 ¿Cuál es el proceso que realiza? Le da valor a las  
 variables para que luego el programa calcule la distancia  
 entre Link y el tesoro, además de que adquiera la mejora  
 cuando esté a una distancia lo suficientemente cerca  
  
**Diseño:**

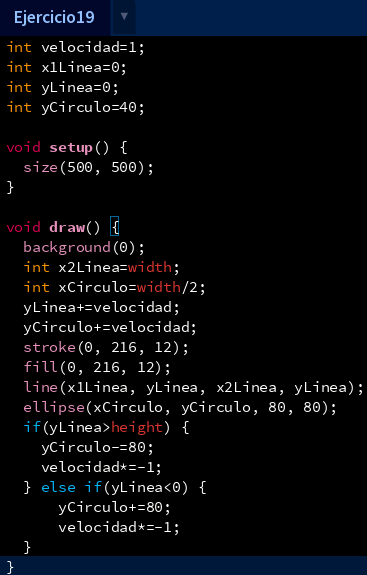
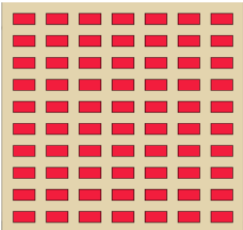
|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** El usuario |
| **VARIABLES**  x1, x2, y1, y2, distanciaTesoro: Entero  coordX, coordY, distancia: Reales  coordLink: String |
| **NOMBRE ALGORITMO:** calcular\_distancia **PROCESO DEL ALGORITMO**  x1 🡨 mouseX  x2 🡨 400  y1 🡨 mouseY  y2 🡨 600  *Leer* x1, x2, y1, y2  elipse (x1, y1, 60, 60)  rectangulo (x2, y2, 80, 80)  coordX 🡨 x2 – x1 // Tamaño de la base del triángulo  coordY 🡨 y2 – y1 // Tamaño de la altura  *Leer* coordX, coordY  distancia 🡨 ((coordX ^ 2) + (coordY ^ 2)) ^ 0.5 // El cálculo de la  distancia entre Link y el tesoro, es decir, la  hipotenusa del triángulo  *Mostrar* “Distancia: “ + distancia  **NOMBRE ALGORITMO:** adquirir\_mejora **PROCESO DEL ALGORITMO**  distanciaTesoro 🡨 50  *Leer* distancia  si (distancia <= distanciaTesoro)  *Mostrar* “Mejora adquirida”  fin\_si |

**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.  
  
**Definición del problema:** Obtener las raíces de una ecuación de segundo grado.  
  
**Análisis:** Datos de Entrada: a, b, c: Real  
 Datos de Salida: discriminante, x1, x2, x: Real  
 Proceso: ¿Quién debe realizar el proceso? El programa  
 ¿Cuál es el proceso que realiza? Lee las variables a, b y c,  
 para realizar la discriminante y determinar sus raices a  
 través de un “según”.  
  
**Diseño:**

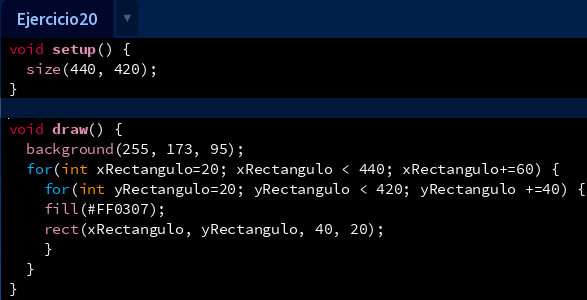
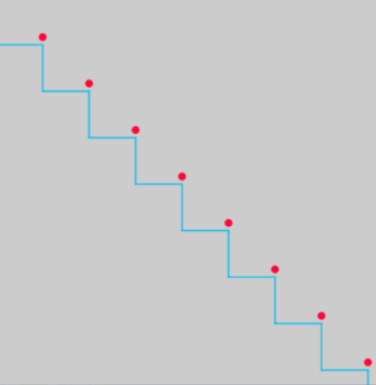
|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** El programa |
| **VARIABLES**  a, b, c: Real  discriminante, x1, x2, x: Real |
| **NOMBRE ALGORITMO:** determinar\_raices **PROCESO DEL ALGORITMO**  a 🡨 6  b 🡨 9  c 🡨 3  *Leer* a  *Leer* b  *Leer* c  discriminante 🡨 b ^ 2 – (4 \* a \* c)  *Leer* discriminante  **según\_sea** (discriminante) hacer  **caso** >0:  x1 🡨 (-b + (discriminante ^ 0.5)) / (2 \* a)  x2 🡨 (-b – (discriminante ^ 0.5)) / (2 \* a)  *Mostrar* “Sus raíces son: “ + x1 + “ y “ + x2  sentencia de ruptura  **caso** =0:  x 🡨 -b / (2 \* a)  *Mostrar* “La raíz doble es: “ + x  sentencia de ruptura  **otros:**  *Mostrar* “No hay raíces reales”  sentencia de ruptura  **fin\_según** |

**Codificación:**  
  
  
  
**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 que vaya desde arriba hacia abajo y ocupe todo el ancho, además, dibujar una elipse de 80px de ancho y 80px de alto, a una distancia vertical de 40px y se mueva junto a la línea.  
  
**Análisis:** Datos de Entrada: velocidad, x1Linea, yLinea, x2Linea, xCirculo,  
 yCirculo: Entero  
 Datos de Salida: velocidad, yLinea, yCirculo  
 Proceso: ¿Quién debe realizar el proceso? El usuario  
 ¿Cuál es el proceso que realiza? Declara las variables y  
 posteriormente instruye al programa para ejecutar el  
 código de acuerdo a la posición y de la línea y del  
 círculo.  
  
**Diseño:**

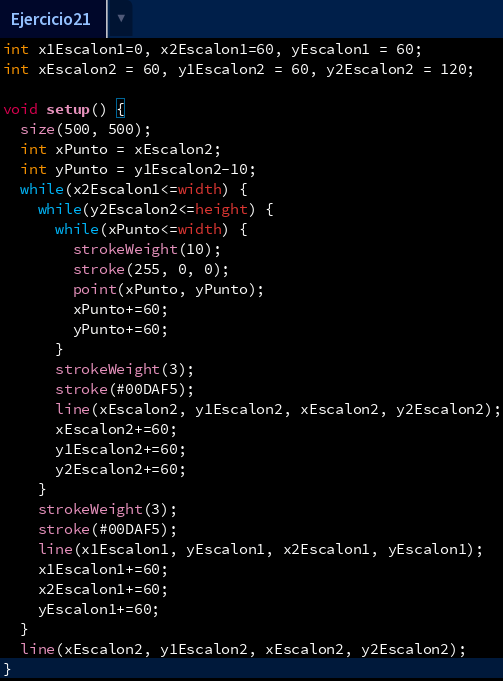
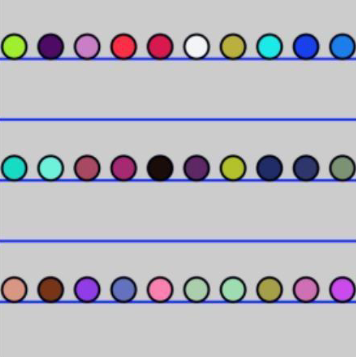
|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** El usuario |
| **VARIABLES**  velocidad, x1Linea, yLinea, x2Linea, xCirculo, yCirculo: Entero |
| **NOMBRE ALGORITMO:** dibujar\_figuras **PROCESO DEL ALGORITMO** velocidad 🡨 1  x1Linea 🡨 0  yLinea 🡨 0  x2Linea 🡨 *Ancho*  xCirculo 🡨 *Ancho* / 2  yCirculo 🡨 40  *Leer* yLinea, velocidad, yCirculo  yLinea 🡨 yLinea + velocidad  yCirculo 🡨 yCirculo + velocidad  *Leer* x1Linea, x2Linea, xCirculo  linea (x1Linea, yLinea, x2Linea, yLinea)  elipse(xCirculo, yCirculo, 80, 80)  **según\_sea** (yLinea) hacer:  **caso** >*Alto*:  yCirculo 🡨 yCirculo – 80  velocidad 🡨 velocidad \* -1  **caso** <0:  yCirculo 🡨 yCirculo + 80  velocidad 🡨 \* -1  **fin\_según** |

**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:  
  
  
  
**Definición del problema:** Dibujar rectangulos de idénticas medidas y que mantengan una distancia de 20 pixeles entre ellos  
  
**Análisis:** Datos de Entrada: xRectangulo, yRectangulo: Entero  
 Datos de Salida: xRectangulo, yRectangulo  
 Proceso: ¿Quién debe realizar el proceso? El usuario  
 ¿Cuál es el proceso que realiza? Declara las variables y  
 les da el valor necesario para indicarle al programa que  
 haga rectángulos mediante la iteración “para”  
  
**Diseño:**

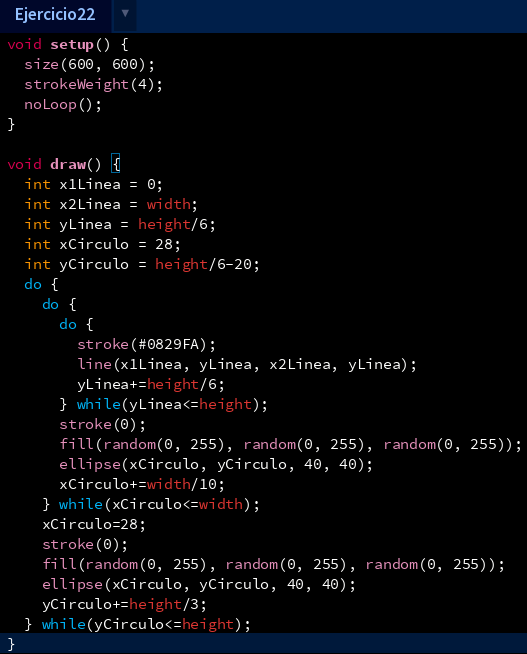
|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** El usuario |
| **VARIABLES**  xRectangulo, yRectangulo: Entero |
| **NOMBRE ALGORITMO:** dibujar\_rectangulos **PROCESO DEL ALGORITMO**  **para** xRectangulo 🡨 20 **hasta** xRectangulo < 440 **incremento** 60  **hacer**  **para** yRectangulo 🡨 20 **hasta** yRectangulo < 420 **incremento**  40 **hacer**  *Leer* xRectangulo  Leer yRectangulo  rectangulo(xRectangulo, yRectangulo, 40, 20)  **fin\_para  fin\_para** |

**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.  
  
**Definición del problema:** Dibujar escalones y sobre cada borde dibujar un punto rojo  
  
**Análisis:** Datos de Entrada: x1Escalon1, x2Escalon1, yEscalon1, xEscalon2,  
 y1Escalon2, y2Escalon2, xPunto, yPunto: Entero  
 Datos de Salida: x1Escalon1, x2Escalon1, yEscalon1, xEscalon2,  
 y1Escalon2, y2Escalon2, xPunto, yPunto  
 Proceso: ¿Quién debe realizar el proceso? El usuario  
 ¿Cuál es el proceso que realiza? Declara las variables y  
 les da el valor necesario para indicarle al programa que  
 dibuje los escalones y los puntos mediante la iteración  
 “mientras”  
  
 **Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** El usuario |
| **VARIABLES**  x1Escalon1, x2Escalon1, yEscalon1, xEscalon2, y1Escalon2,  y2Escalon2, xPunto, yPunto |
| **NOMBRE ALGORITMO:** dibujar\_figuras **PROCESO DEL ALGORITMO**  x1Escalon1 🡨 0  x2Escalon1 🡨 60  yEscalon1 🡨 60  xEscalon2 🡨 60  y1Escalon2 🡨 60  y2Escalon2 🡨 120  *Leer* xEscalon2  xPunto 🡨 xEscalon2  *Leer* y1Escalon2  yPunto 🡨 y1Escalon2 -10  *Leer* x1Escalon1  *Leer* x2Escalon1  *Leer* yEscalon1  *Leer* y2Escalon2  *Leer* xPunto  *Leer* yPunto  **mientras** (x2Escalon1<=*Ancho*) **hacer**  **mientras** (y2Escalon2<=*Alto*) **hacer**  **mientras** (xPunto<=*Ancho*) **hacer** punto(xPunto, yPunto)  xPunto 🡨 xPunto + 60  yPunto 🡨 yPunto + 60  **fin\_mientras** linea(xEscalon2, y1Escalon2, xEscalon2, y2Escalon2)  xEscalon2 🡨 xEscalon2 + 60  y1Escalon2 🡨 y1Escalon2 + 60  y2Escalon2 🡨 y2Escalon2 + 60  **fin\_mientras**  linea(x1Escalon1, yEscalon1, x2Escalon1, yEscalon1)  x1Escalon1 🡨 x1Escalon1 + 60  yEscalon1 🡨 yEscalon1 + 60  x2Escalon1 🡨 x2Escalon1 + 60  **fin\_mientras**  linea(xEscalon2, y1Escalon2, xEscalon2, y2Escalon2) |

**Codificación:**  
  
  
  
**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 franjas que dividan el lienzo en igual medida, y dibujar círculos sobre cada línea de por medio  
  
**Análisis:** Datos de Entrada: x1Linea, x2Linea, yLinea, xCirculo, yCirculo:   
 Entero  
 Datos de Salida: yLinea, xCirculo, yCirculo  
 Proceso: ¿Quién debe realizar el proceso? El usuario  
 ¿Cuál es el proceso que realiza? Declara las variables y  
 les da el valor necesario para indicarle al programa que  
 dibuje las franjas y los círculos mediante la iteración  
 “hacer-mientras”  
  
**Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** El usuario |
| **VARIABLES**  x1Linea, x2Linea, yLinea, xCirculo, yCirculo: Entero |
| **NOMBRE ALGORITMO:** dibujar\_figuras **PROCESO DEL ALGORITMO**  x1Linea 🡨 0  x2Linea 🡨 *Ancho*  yLinea 🡨 *Alto* / 6  xCirculo 🡨 28  yCirculo 🡨 *Alto* / 6 – 20  *Leer* x1Linea  *Leer* x2Linea  *Leer* yLinea  *Leer* xCirculo  *Leer* yCirculo  **hacer  hacer  hacer**  linea(x1Linea, yLinea, x2Linea, yLinea  yLinea 🡨 yLinea + *Altura* / 6  **mientras** (yLinea <= *Altura*)  elipse(xCirculo, yCirculo, 40, 40)  xCirculo 🡨 xCirculo + *Ancho* / 10  **mientras** (xCirculo <= *Ancho*)  xCirculo 🡨 28  elipse(xCirculo, yCirculo, 40, 40)  yCirculo 🡨 yCirculo + *Alto* / 3  **mientras** (yCirculo <= *Alto*) |

**Codificación:  
  
**