**REGLAMENTO**

Crear una carpeta denominada TP01\_XXXX donde XXXX es el apellido\_nombre del estudiante. Al producto final, subirlo en su repositorio y compartir el enlace en formulario.

# Sección Expresiones aritméticas y lógicas

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:

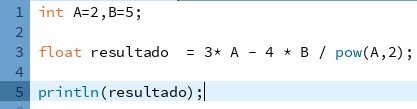
(3\*A)-(4\*B/(A^2))

6-(4\*B/4)

6-5

1

Captura de Processing



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

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

1 + ((0,12 / 4) \* 2)

1+ (0,03 \* 2)

1 + 0,06

1,06

Ejercicio 3: Escribir las siguientes expresiones algebraicas como expresiones algorítmicas (en su forma aritmética dentro del algoritmo). En este caso no se pide evaluarlas ni programarlas.

Ejercicio 4: Evaluar las siguientes expresiones aritméticas, para lo cual indicar en el caso de las variables, el valor indicado. **Luego escribirlas como expresiones algebraicas.**

a) b ^ 2 – 4 \* a \* c

(b ^ 2) – (4 \* a \* c)

16 – 48

-32

Algebraica:

16 – 48

-32

1. 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

15

Algebraica:

15

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

(b + d) / (c + 4)

12 / 10

1,2

Algebraica:

1,2

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

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

100 ^ (1/2)

10

Algebraica:

Para aclarar que indicamos con ”Luego escribirlas como expresiones algebraicas” lo aplicamos con el punto a)

𝑏2 − 4. 𝑎. 𝑐

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)

20 – 6,25

13,75

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

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

20 / 9

2,22

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

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

R1 = y+z

R2 = x >= R1

R1= 4+1 = 5

R2= 3 >= R1

Falso

Ejercicio 7: Para contador1=3, contador3=4, evaluar el resultado de

R1 = ++contador1

R2 = contador1 < contador2

R1 = 1+3

R2 = 4<4

Falso

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

a+b-1 < x\*y

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

29 < 12

Falso

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

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

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

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

Falso && Falso

Falso

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

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

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

!(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)

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

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

verdadero || verdadero && 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.

Análisis:

Datos de Entrada: nombre\_ingresado //cadena

Datos de Salida: mensaje\_saludo //cadena de texto

Proceso:¿Quien debe realizar el proceso?: El algoritmo o computadora

¿Cual es el proceso que resuelve?: Ingresar un nombre que devolverá la creación de un saludo personalizado con el nombre proporcionado y su presentación en pantalla.

Diseño:

Entidad que resuelve el problema: Algoritmo

Variables:

nombre\_ingresado: string // almacena el nombre

mensaje\_saludo: string // almacenara una cadena de caracteres

Nombre del Algoritmo: saludar\_nombre

Proceso del algoritmo:

1. inicio

2. Leer nombre\_ingresado

3. mensaje\_saludo ← “Hola, ” + nombre\_ingresado + “ ¡Bienvenido!”

4. Mostrar saludo 5. Fin

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

Análisis:

Datos de Entrada: base, altura //decimal

Datos de Salida: perimetro, area // almacena valores decimales

Proceso:

¿Quién debe realizar el proceso?: El usuario o calculadora

¿Cuál es el proceso que resuelve?: calcula el perímetro y el área de un rectángulo utilizando las fórmulas adecuadas

P=2(base +altura) y A=base ⋅ altura

Diseño:

Entidad que resuelve el problema: persona

Variables:

base: float // almacena un valor decimal

area: float // almacena un valor decimal

perimetro: float //

area: float //

perimetroArea: float // almacena un valor de calculos

Nombre del Algoritmo: perimetro\_area\_rectangulo

Proceso del algoritmo:

inicio

Leer base

Leer area

perimetro ← 2\*(base + altura)

area ← base \* altura

perimetroArea ← “el perimetro de un rectángulo es: ” + perimetro + “ y la area de un rectángulo es: ” + area

Mostrar perimetroArea

fin

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



Análisis:

Datos de Entrada: catetoA, catetoB

Datos de Salida: hipotenusa

Proceso:

¿Quien debe realizar el proceso?: La persona o calculadora

¿Cual es el proceso que resuelve?: Para calcular la longitud de la hipotenusa de un triángulo rectángulo se obtiene las longitudes de los catetos como entrada, se aplica la fórmula: h^2=a^2+b^2

h=√(a^2+b^2 )

Diseño:

Entidad que resuelve el problema: persona

Variables:

catetoA: int // almacena un valor decimal

catetoB: int // almacena un valor decimal

hipotenusa: int // almacena un valor de calculos

Nombre del Algoritmo: perimetro\_area\_rectangulo

Proceso del algoritmo:

Leer catetoA

Inicio

Leer catetoA

Leer catetoB

hipotenusa ← (a^2 + b^2 ) ^(0.5)

mostrar hipotenusa

Fin

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.

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



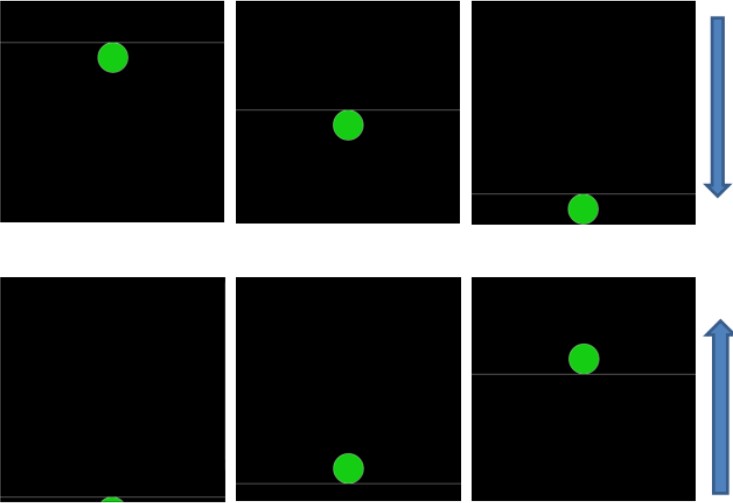
Ejercicio 17: Si queremos representar personajes o power ups (premios) en la pantalla debemos primero ubicarlos en alguna posición dentro de la pantalla. Imagine que está en un juego donde un power up desaparece porque el personaje se acerca a una distancia de x unidades, sin importar por donde se acerque. 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 lLnk con un Circulo, y al tesoro con un cuadrado. Además, mueva a Link mediante el mouse.

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.

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



Análisis:

Datos de Entrada: línea, dir

Datos de Salida: bucle de la línea y circulo

Proceso:

¿Quien debe realizar el proceso?: la computadora

¿Cual es el proceso que resuelve?:

Diseño:

Entidad que resuelve el problema: lienzo

Variables:

- linea: entero // almacena un valor entero

- dir : entero // almacena un valor enter

Nombre del Algoritmo: linea\_circulo\_en\_movimiento

Proceso del algoritmo:

1. inicio

2. Leer linea

3. Leer dir

4. anchoLienzo ← 400

5. altoLienzo ← 400

6. para i ← 0 hasta alto incremento 1 hacer

7. linea ← linea + dir

8. fin\_para

9. si ((linea >= anchoLienzo) O (linea <= 0)) entonces

10. dir ← dir \* (-1)

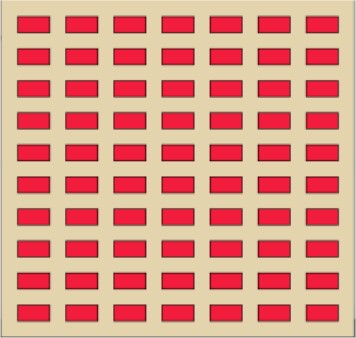
11. fin\_si

12. mostrar linea

13. dibujar linea en (dir, linea, altoLienzo, linea)

14. dibujar circulo en (altoLienzo/2, linea + 40, 80, 80)15. fin

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:



Análisis:

Datos de Entrada: Rectángulos dibujados en el lienzo según las especificaciones dadas.

Datos de Salida: Rectángulos dibujados en el lienzo según las especificaciones dadas.

Proceso:

El proceso puede ser realizado por un programa como processing.

El proceso que resuelve es a dibujar una serie de rectángulos en un lienzo de tamaño específico, manteniendo una distancia específica entre ellos tanto horizontal como verticalmente, definiendo un bucle for para dibujar los rectángulos en el lienzo

Diseño:

Entidad que resuelve el problema: lienzo

Variables:

- coordenadas Rect: float //almacena un valor de coordenadas

- ancho, alto, distancia Entre Rect: int//almacena un valor entero

- anchoLienzo, altoLienzo: int //almacenan valores enteros

Nombre del Algoritmo: rectangulos\_repetidos

Proceso del algoritmo:

1. inicio

2. anchoLienzo ← 440

3. altoLienzo ← 420

4. ancho← 40

5. alto← 20

6. distanciaRect ← 20

7. para x coordenadas Rect.x hasta ancholienzo con paso (ancho+distancia EntreRect)

8. hacer

9. para y = coordenadas Rect.y hasta altoLienzo con paso (alto+distancia EntreRect)

10. hacer

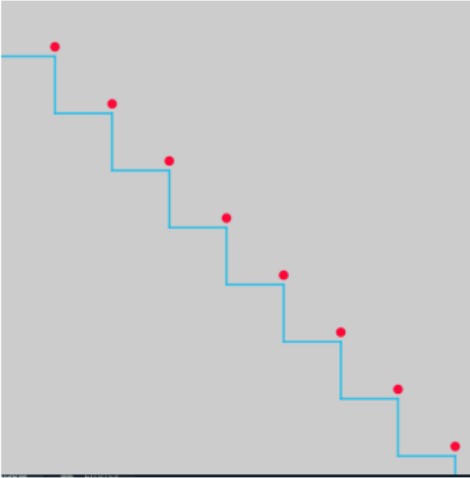
11. dibujar rectangulo en (x,y, ancho, alto)

12. fin\_para

13. fin\_para

14. fin

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.

Análisis:

Datos de Entrada: puntoA, puntoB, puntoC, puntoD, distancia

Datos de Salida: una imagen que consiste en escalones con puntos de color rojo en los bordes.

Proceso:

¿Quién debe realizar el proceso?: El programa, mediante el código en Processing.

¿Cuál es el proceso que resuelve?: El proceso consiste en iterar mediante while() para dibujar escalones y puntos rojos en los bordes

Diseño:

Entidad que resuelve el problema: programa

Variables:

- puntoA,puntoB,puntoC,puntoD: int //almacena un vector

- distancia : int //almacena un valor entero

Nombre del Algoritmo: escalones\_puntos

Proceso del algoritmo:

1. Inicio

2. anchoLienzo ← 500

3. altoLienzo ← 500

4. distancia ← 60

5. mientras (puntoA.y sea menor o igual que anchoLienzo) Hacer

6. dibujar línea horizontal en (puntoA.x, puntoA.y, puntoB.x, puntoB.y)

7. dibujar línea vertical en (puntoB.x, puntoB.y, puntoC.x, puntoC.y)

8. dibujar circulo en (puntoD.x, puntoD.y)

9. puntoA.x ← puntoC.x

10. puntoA.y ← puntoC.y

11. fin\_mientras12. Fin

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.

Análisis:

Datos de Entrada: números de líneas y círculos

Datos de Salida: círculos con colores randoms sobre líneas con un color con distanciamiento por medio

Proceso:

¿Quien debe realizar el proceso?: El lienzo se divide verticalmente en franjas de igual medida, donde se dibujan líneas en todas ellas. En cada línea de forma alternada, se dibujan círculos con colores aleatorios, los cuales están espaciados uniformemente a lo largo de la línea.

¿Cual es el proceso que resuelve?: el programa en este caso processing

Diseño:

Entidad que resuelve el problema: processing

Variables:

- distanciaCiruclo: int //almacena un valor entero

- lineaX, lineaY, circuloX, circuloX, distanciaCirculo : int //almacena un valor entero

- anchoLienzo, altoLienzo: int //almacenan valores enteros

Nombre del Algoritmo: rectangulos\_repetidos

Proceso del algoritmo:

1. inicio

2. anchoLienzo ← 600

3. altoLienzo ← 600

4. lineaX ← 0

5. lineaY ← 100

6. distanciaCirculo ← 30;

7. circuloY ← 75

8. hacer

9. circuloX ← distanciaCirculo

10. hacer

11. dibujar linea en (lineaX, lineaY, anchoLienzo, lineaY)

12. dibujar circulo en circuloX, circuloY, 50, 50)

13. circuloX ← circuloX + distanciaCirculo\*2

14. fin\_hacer

15. mientras(circuloX sea menor que ancholienzo)

16. LineaY ← lineaY + 100;

17. circuloY ← circuloY + 200;

18. fin\_hacer

19. mientras (lineaY sea menor que altoLienzo)20. fin