Fundamentos de programación orientada a objetos

Azul Argentina Brito
TUDIVJ000105
44773907

2-

Aritmética
4 / 2 * 3 / 6 + 6 / 2 / 1 / 5 ^ 2 / 4 * 2 (((4/2) * 3) /6) + ((((6/2) / 1) / (5 ^ 2)) / 4) * 2 1.0 + 0.06 1.06

4a--

Aritmetica	
b ^ 2 – 4 * a * c (3 ^ 2) - (4 * 2 * 4) 9-32 -23	

4b-

Aritmetica
3 * X ^ 4 – 5 * X ^ 3 + X 12 – 17 (3*(5^4)) - (5 * (5^3)) + (5*12) - 17 1875 – 625 + 60 – 17 1293

4c-

Aritmetica
(b + d) / (c + 4) (3+82) / (5+4) 85/9 9.4…

4d-

Aritmetica
(x ^ 2 + y ^ 2) ^ (1 / 2) (22^ 2 +15^ 2)^ 0.5 709^ 0.5 26.627054

5a-

Aritmetica
B * A – B ^ 2 / 4 * C 5 * 4 – ((5 ^ 2) / 4) * 1 20 - 6.25 13.75

b-

Aritmetica
(A * B) / 3 ^ 2 (4 * 5) / 3 ^ 2 2.2 ...

c-

Aritmetica
(((B + C) / 2 * A + 10) * 3 * B) – 6 (((5 + 1) / 2 * 4 + 10) * 3 * 5) - 6 ((6 / 2 * 4 + 10) * 3 * 5) - 6 ((3 * 4 + 10) * 3 * 5) - 6 (22 * 3 * 5) - 6 330 – 6 324

6-

R1 = y+z

R2 = x >= R1

R1= 4+1 = 5

R2= 3 >= R1

Falso

7-

R1 = ++contador1

R2 = contador1 < contador2

R2= 4 < 4

R2= falso

8-

a+b-1 < x*y

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

29 < 12

falso

9-
! (x<5) &&!(y>=7)
!(x<5) && !(y>=7)
!(6<5) && !(8>=7)
falso && falso
falso

10-
!((i>4) || !(j<=6))
!((i>4) || !(j<=6))
!((22>4) || !(3<=6))
!(verdadero || falso)
!(verdadero)
Falso

11-
!(a+b==c) || (c!=0) && (b-c>=19)
!(34+12==8) || (8!=0)&&(12-8>=19)
!(46==8) || (8!=0)&&(4>=19)
verdadero

12-Analisis
Datos de entrada: nombre// cadena de texto
Datos de salida: mostarSaludo// boolean

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: Algoritmo
Variables: nombre: string//almacena el nombre mostrarSaludo: boolean//
Nombre del algoritmo: mostrarSaludo_nombre
1. inicio 2. leer nombre

3. mostrarSaludo <= verdadero
4. msostrarSaludo “¡Hola, + nombre + ! Bienvenidi a Processing -3-"
5. fin

13-
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 correspondientes.

Dsieño:

Entidad que resuelve el problema: persona
Variables: • base: float // almacena un valor decimal • altura: float // almacena un valor decimal • perimetro: float // almacena valor de calculo • area: float // almacena valor de calculo
nombre algoritmo: perimetro_area
1. inicio 2. leer base 3. leer altura 4. leer perimetro <= formula del perimetro, 2*(base +altura) 5. leer area <= formula del area, base * altura 6. mostrar perimetro <="el perimetro del rectangulo es: "+ perimetro 7. mostrar area <= “el area del rectangulo es: " + area 8. fin

14-
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

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 algoritmo: calculo_hipotenusa
Proceso del algoritmo: 1. Inicio 2. Leer catetoA 3. Leer catetoB 4. hipotenusa <= (a^2 + b^2) ^(0.5) 5. mostrar hipotenusa 6. Fin

15-
Datos de Entrada: num1, num2
Datos de Salida: suma
Proceso:
¿Quien debe realizar el proceso?: La persona o calculadora
¿Cual es el proceso que resuelve?: Para calcular la longitud de la hipotenusa(suma) de un triángulo rectángulo se obtiene las longitudes de los catetos como entrada(num1, num2), se aplica la fórmula

Diseño

Entidad que resuelve el problema: persona
Variables: num1: int // almacena un valor decimal num2: int // almacena un valor decimal suma: int // almacena un valor de suma resta: int// almacena valor de resta multiplicación: int// almacena valor de la multiplicacion division. int// almacena valor de la division

nombre algoritmo: operaciones_basicas

proceso algoritmo:

1. inicio
2. leer num1
3. leer valor2
4. suma<= num1+num2
5. mostrar "El resultado de la suma es: " + suma
6. resta <= num1-num2
7. mostrar "el resultado de la resta es: " + resta
8. multiplicacion <= num1 * num2
9. mostrar resultado multiplicacion
10. division <= num1/num2
11. si la division no es cero entonces mostrar "El resultado de la división es: " + division
12. si no, entonces mostrar "La división por cero no está definida."
13. fin

16-

Datos de Entrada: Temperatura en grados Fahrenheit
Datos de Salida: Temperatura en grados Celsius

Proceso:
¿Quién debe realizar el proceso?: Puede ser realizado por un programa informático o una calculadora
¿Cuál es el proceso que resuelve?: El proceso consiste en convertir una temperatura dada en grados Fahrenheit a grados Celsius utilizando la fórmula de conversión correspondiente

Diseño:

Entidad que resuelve el problema: persona
Variables: ● temperaturaF: float // almacena un valor decimal ● temperaturaC: float // almacena un valor decimal

nombre algoritmo: conversor_Fahrenheit_Celcius
Proceso del algoritmo: 1. inicio 2. Leer temperaturaF 3. temperaturaC <= (5.0 / 9.0) * (temperaturaF – 32) 4. mostrar "Temperatura en Celsius: " + temperaturaC

5. fin

17-
Análisis:
Datos de Entrada: Coordenadas de Link, Coordenadas del tesoro
Datos de Salida: Distancia entre Link y tesoro.

Proceso:
¿Quién debe realizar el proceso?: El programa informático o una calculadora que pueda realizar cálculos matemáticos.
¿Cuál es el proceso que resuelve?: Calculamos las diferencias en las coordenadas x;y entre los dos puntos que nos darán los catetos formados por los puntos

Diseño:

Entidad que resuelve el problema: persona

variables
- x1: float // almacena un valor decimal
- y1: float // almacena un valor decimal
- x2: float//almacena valor decimal
- y2: float//almacena valor decimal
- coordenadas//almacena resultado
- distanciaTesoro:float//almacena un valor
- distancia:float//almacena un valor

nombre algoritmo: distancia_tesoro_link

1. inicio
2. leer x1
3. leer y1
4. leer x2
5. leer y2
6. distanciaTesoro=15
7. coordenadas = "X1: " + posicion del mouse en eje x + ", Y1: " + posición de mouse en eje y
8. coordenadaX = x2 - x1
9. cordenadaY = y2 - y1
10. distancia <=((coordenadaX)^2 + (coordenadaY)^2)^2
11. si la distancia es menor o igual a la distancia del tesoro entonces mostrar "PowerUp"
12. fin

18-
Analisis:
Datos de Entrada: Coeficientes de la ecuación cuadrática: a, b y c.
Datos de Salida: Raíces de la ecuación cuadrática.

Proceso:
¿Quién debe realizar el proceso?: El programa informático o una calculadora que pueda realizar cálculos matemáticos.
¿Cuál es el proceso que resuelve?: Calcular el discriminante de la ecuación cuadrática utilizando la fórmula

Diseño:

Entidad que resuelve el problema: persona
variables: ● a: float//almacena valor decimal ● b:float//almacena valor decimal ● c:floa//talmacena valor decimal ● discriminante: float//almacena valor decimal

nombre algoritmo:raices
1. inicio 2. leer a 3. leer b 4. leer 5. discriminante <= b * b - 4 * a * c 6. si discriminante es mayor que 0 7. x1 <= (-b + (discriminante))^0.5 /(2*a) 8. x2<= (-b - (discriminante))^0.5 /(2*a) 9. mostrar "Las raíces son: " + x1 + " y " + x2 10. sino, si el discriminante es igual a cero 11. x <= -b / (2* a) 12. mostrar "La raíz repetida es: " + x 13. sino mostrar "Las raíces son complejas y no se pueden representar en los números reales."

19-
Analisis:
Datos de Entrada: posY, dirY
Datos de Salida: bucle de la línea y circulo

Proceso:
¿Quien debe realizar el proceso?: la computadora
Diseño

Entidad que resuelve el problema: lienzo
Variables: ● posY:float//almacena valor ● dirY:float//almacena valor

nombre algoritmo: bucle_linea_cirulo
proceso 1. inicio 2. lee posY 3. lee dirY 4. ancho lienzo<=500 5. alto lienzo<=500 6. se incrementa posY y se asigna a dirY al mismo tiempo 7. si posY es mayor a la altura o es menor a 0 entonces 8. se multiplica dirY por menos uno para que se invierta 9. dibujar linea <=rect(0, posY, 500, 2) 10. dibujar elipse <=ellipse(250, posY + 40, 80, 80) 11. fin

20_Analisis:
Datos de entrada: Rectángulos dibujados segun las especificaciones dadas
Datos de salida:Rectángulos dibujados segun las especificaciones dadas
Proceso:
¿Quien debe realizar el proceso?: El proceso puede ser realizado por un programa como processing.
¿Cual es el proceso que resuelve?: 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: ● PVector coordenadasRect// almacena la posicion de los rectangulos ● int ancho, alto, distRects// almacena valor de entero

Nombre del algoritmo: rectangulos
Proceso: 1. inicio 2. medidas del lienzo, ancho 440, alto 420 3. distRects 20 4. ancho 40 5. alto 20 6. dibujarRectangulos (en x,y, ancho, alto) 7. para x coordenadasRect.x hasta anchoLienzo con paso (ancho+distRects) 8. para y coordenadasRect.y hasta altoLienzo con paso (alto+distRects)

21-
analisis:
Datos de Entrada: puntoA, puntoB, puntoC, puntoD y distLinea
Datos de Salida: una imagen que consiste en escalones con puntos de color rosa 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,B,C,D:PVector// almacena valor de un vector ● distLinea : int //almacena un valor entero

nombre algoritmo: escalones_puntos
proceso 1. inicio 2. ancho lienzo<=500 3. alto lienzo<=500 4. distLinea<=60 5. mientras (puntoA sea menor que la altura del lienzo) 6. dibujar escalon <=dibujar línea horizontal en (puntoA.x, puntoA.y, puntoB.x, puntoB.y)

7. dibujar punto <= (puntoB.x, puntoB.y-10)
8. actualizar las coordenadas de puntoA
9. puntoA.x<= puntoC.x;
10. puntoA.y <= puntoC.y;
11. fin

22-
analisis:
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. ● distanciaCirculo:int//almacena valor entero ● circuloY: int//valor entero ● dibujarCirculos: boolean//para verificar si es verdadero o falso

nombre algoritmo: lineas_circulos
1. inicio 2. ancho lienzo:600 3. alto lienzo:600 4. distanciaCirculo<=30 5. circuloY<=80 6. dibujarCirculos <= true 7. bucle *para* repetir 5 cada iteración del bucle representa una fila de líneas y círculos 8. si dibujarCirculo es verdadero, se dibuja una elipse<=circuloX, circuloY, 50, 50 9. aumenta la coordenada X de circuloX en una cantidad equivalente a distanciaCirculo * 2 10. aumenta la coordenada Y de circuloY en 100 píxeles 11. dibujarCircuclo <= no dibujarCirculo

12. fin