**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:

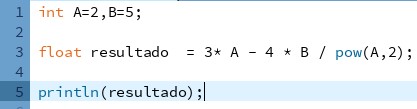
(3\*A)-(4\*B/(A^2))

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

Resolución

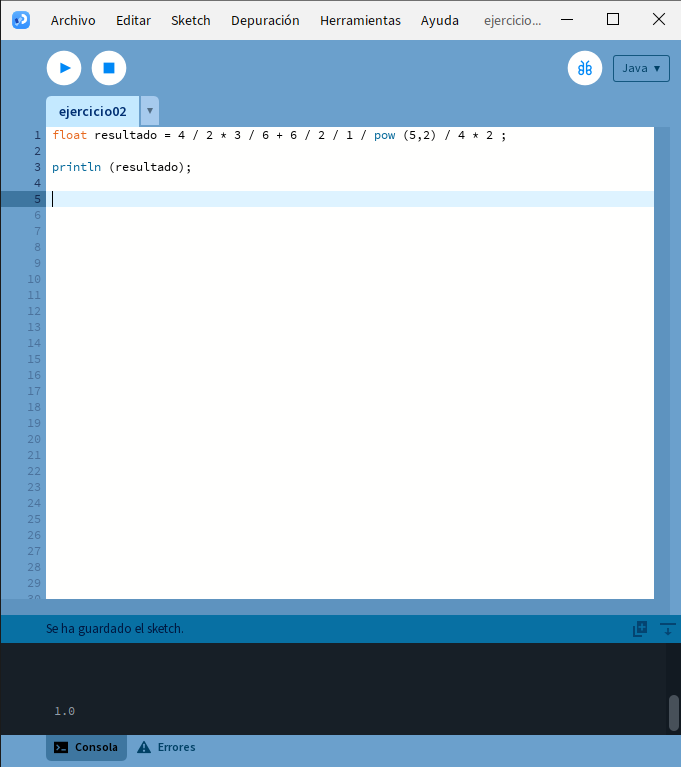
( ( 4 / ( 2 \* 3 )) / 6 ) +( ((6 / 2) / 1 ) / ((5 ^ 2) / (4 \* 2)) )

( (4 / 6) / 6 ) + (( 3 / 1 ) / ( 25 / 8 ))

(0,66 / 6) +( 3 / 3,125)

0,11 + 0,96

1,07



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.

Resolución

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

1. 3 \* X ^ 4 – 5 \* X ^ 3 + X \* 12 – 17
2. (b + d) / (c + 4)
3. (x ^ 2 + y ^ 2) ^ (1 / 2)

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

𝑏2 − 4. 𝑎. 𝑐

Resolución

A= 3, B= 4, C= 2, D= 5, X= 3, Y=1

1. ( 4 ^ 2 ) – ( ( 4 \* 3 ) \* 2 )

16 – ( 12 \* 2 )

16 – 24

-8

b) ( 3 \* ( 3 ^ 4 ) ) – ( 5 \* ( 3 ^ 3 ) ) + ( 3 \* 12 ) – 17 Ex Alg = 3.x4 – 5.x3 + 12 - 17

( 3 \* 81 ) – ( 5 \* 27 ) + 36 – 17

243 – 135 + 36 - 17

243 – 99 – 17

127

1. ( 4 + 5 ) / ( 2 + 4 ) Ex Alg =

9 / 6

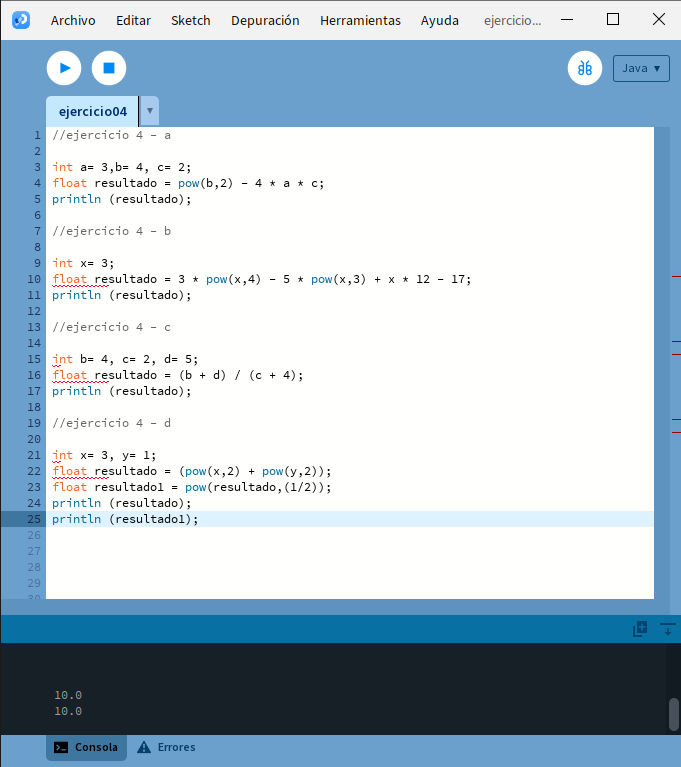
1,5

1. ( 3 ^ 2 + 1 ^ 2) ^ (1 / 2) Ex Alg = √(x2 + y2 )

( 9 + 1 ) ^ (1/2)

10 ^ (1/2)

3,16



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

Resolución

1. 5 \* 4 – 5 ^ 2 / 4 \* 1

20 – 25 / 4

20 – 6,25

13,75

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

20 / 9

2,22

1. ( ( ( 5 + 1 ) / 2 \* 4 + 10 ) \* 3 \* 5 ) – 6

( ( 6 / 2 \* 4 + 10 ) \* 15 ) -6

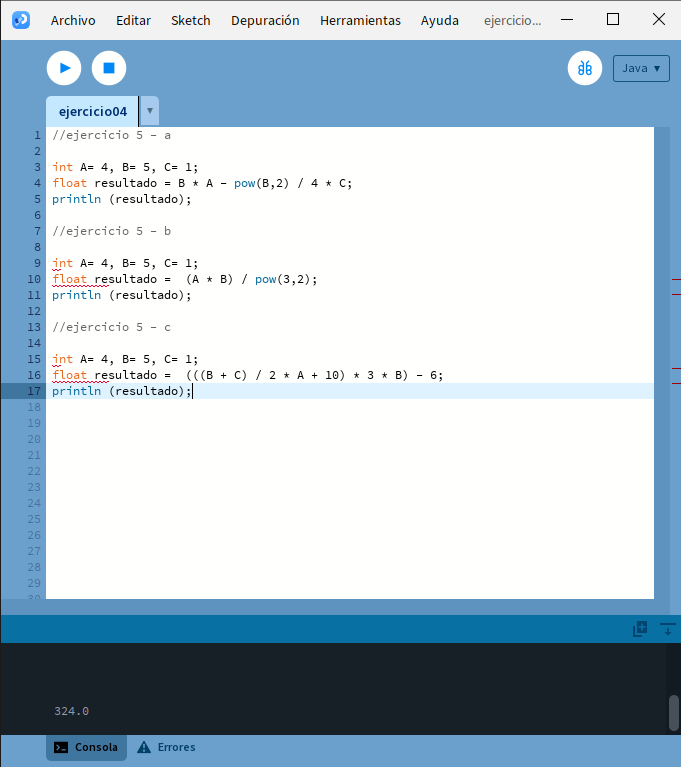
( (3 \* 4 + 10 ) \* 15 ) – 6

( ( 12 + 10 ) \* 15 ) – 6

( 22 \* 15 ) – 6

330 – 6

324



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

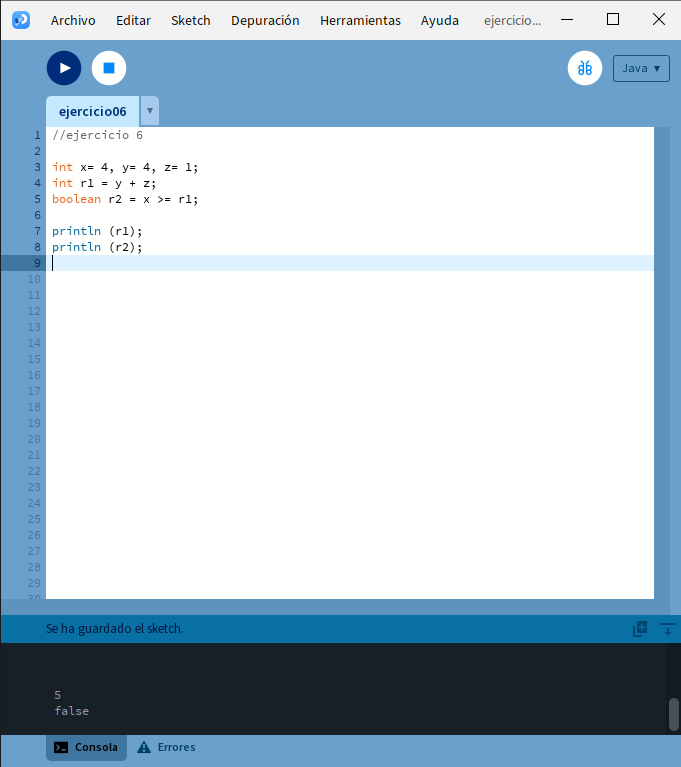
R1 = y+z

R2 = x >= R1

Resolución

R1 = 3 + 4 R2 = 3 >= R1

R1 = 7 R2 = falso



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

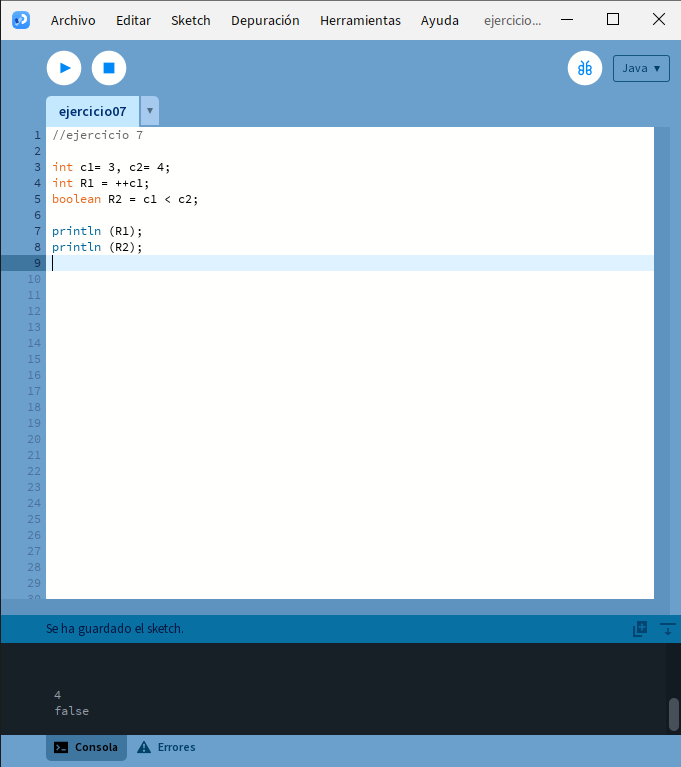
R1 = ++contador1

R2 = contador1 < contador2

Resolución

R1 = ++(3) R2= 3 < 4

R1 = 4 R2= verdadero

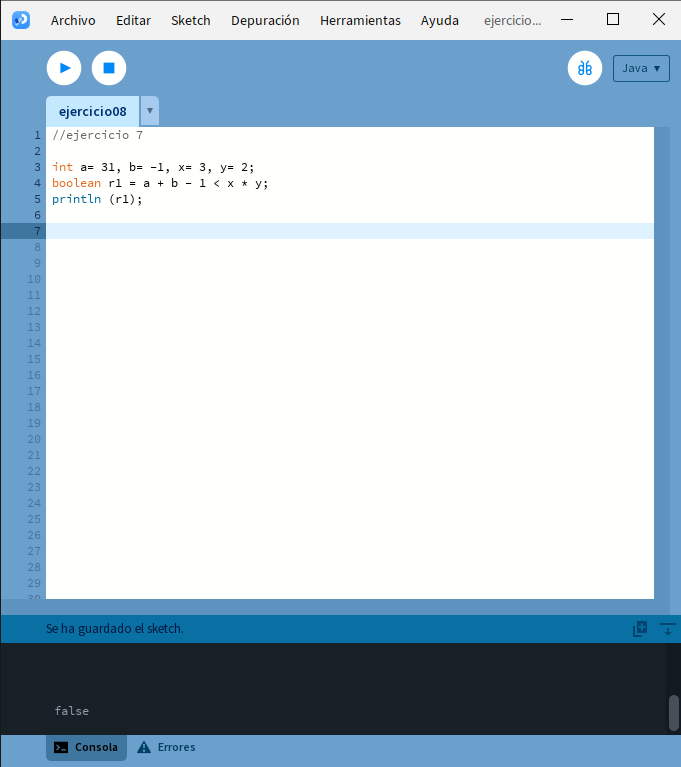


Ejercicio 8: Para a=31, b=-1; x=3, y=2, evaluar el resultado de a+b-1 < x\*y

Resolución

31 + -1 -1 < 3 \* 2

29 < 6 falso



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

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

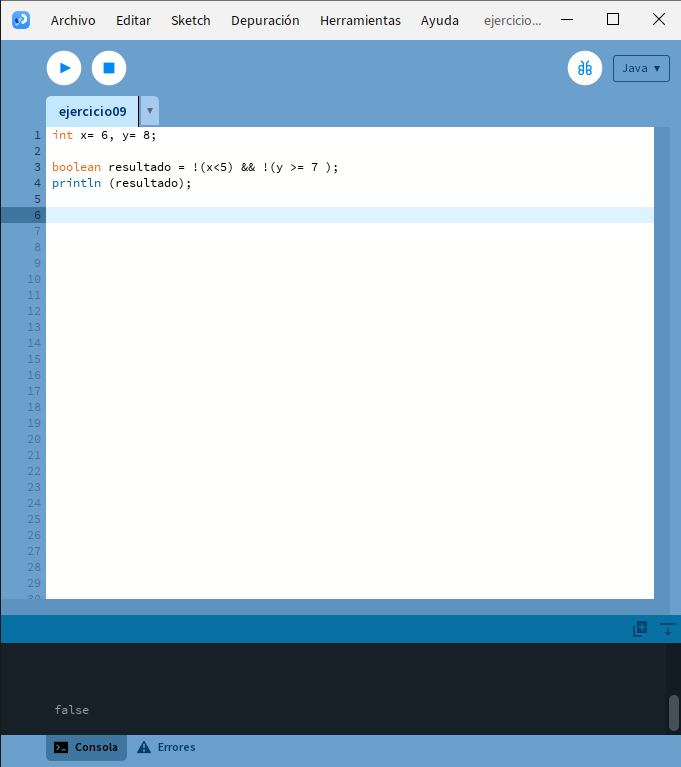
Resolución

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

!(falso) && !(verdadero)

Verdadero && Falso

Falso



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

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

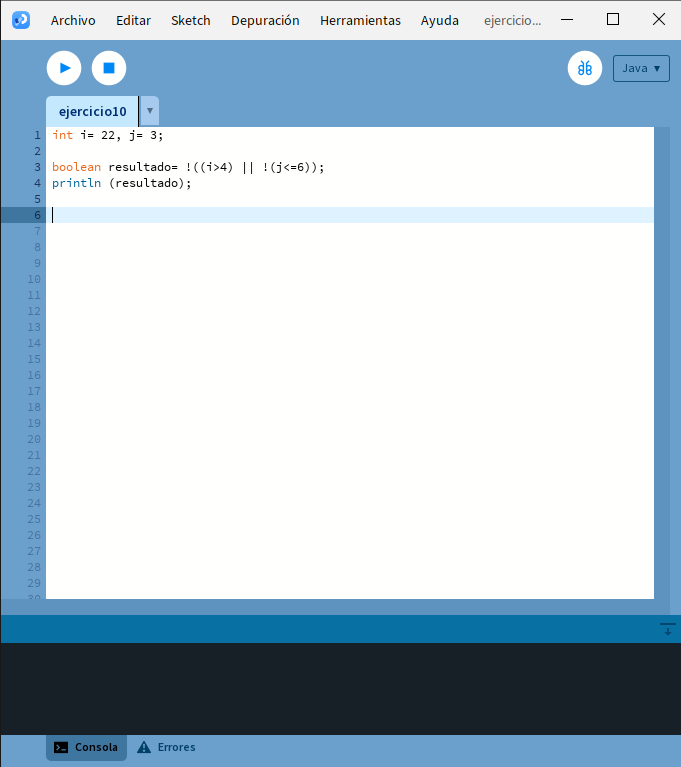
!((i>4) || !(j<=6))

!((verdadero) || !(verdadero))

!((verdadero) || (falso))

!(verdadero)

Falso



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

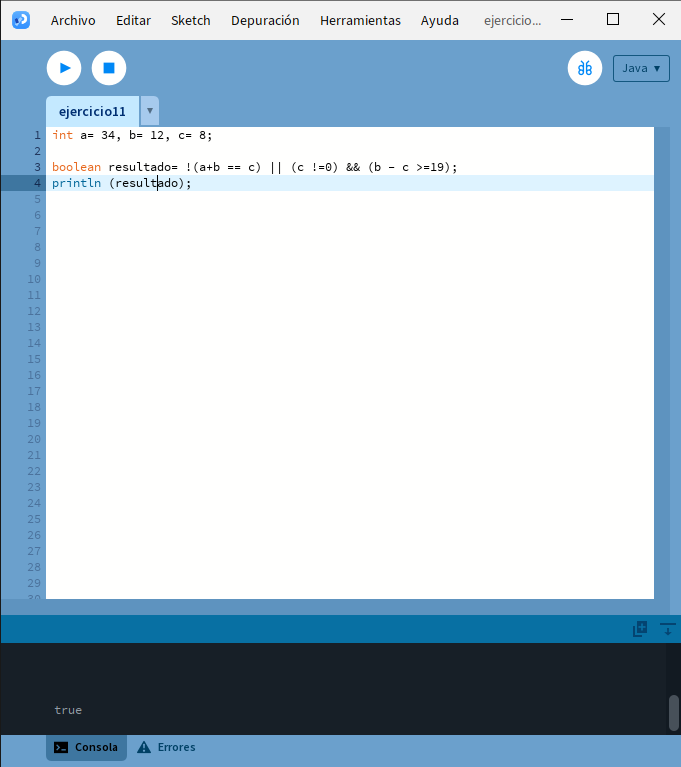
!(a+b==c) || (c!=0)CC(b-c>=19)

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

!(falso) || (verdadero) && (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.

FASE DE ANALISIS

* Definición de problema: pedir nombre por teclado y luego presentarlo en pantalla
* Datos de entrada:
  + Nombre
* Datos de salida
  + Teclado: ingrese su nombre
  + Saludo: hola “nombre” le damos la bienvenida
* Proceso
  + Solicitar por teclado el nombre
  + Guardar el nombre ingresado
  + Generar saludo con el nombre

FASE DE DISENO

|  |
| --- |
| Entidad: pantalla |
| Variables  Teclado, nombre, saludo |
| Nombre de algoritmo: hola te damos la bienvenida  Algoritmo   * Pedir por teclado el nombre * Guardar nombre ingresado * Saludar con el nombre   Fin |

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

FASE DE ANALISIS

* Definición de problema: calcular perímetro y área
* Datos de entrada:
  + Base
  + Altura
* Datos de salida
  + Perímetro
  + Área
  + Resultados
* Proceso
  + Ingresar variables para base y altura
  + Sacar el perímetro
  + Sacar el área
  + Mostrar en pantalla los resultados

FASE DE DISENO

|  |
| --- |
| Entidad: rectángulo |
| Variables  Base, altura |
| Nombre de algoritmo: área y perímetro de un rectángulo  Algoritmo  Inicio   * Ingresar valor a base y altura * Calcular perímetro * Calcular área * Mostrar los resultados   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



FASE DE ANALISIS

* Definición de problema: calcular hipotenusa
* Datos de entrada:
  + Cateto 1 (base)
  + Cateto 2 (altura)
* Datos de salida
  + hipotenusa
* Proceso
  + Introducir el valor de los catetos
  + Calcular hipotenusa
  + Mostrar el resultado

FASE DE DISENO

|  |
| --- |
| Entidad: triangulo rectangulo |
| Variables  Base, altura |
| Nombre de algoritmo: calculo de hipotenusa  Algoritmo  Inicio   * Ingresar valor de cA * Ingresar valor de Cb * Calcular hipotenusa * Mostrar el resultado por pantalla   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.

FASE DE ANALISIS

* Definición de problema: resolver ejercicios matemáticos
* Datos de entrada:
  + Ingresar valor de dos números
* Datos de salida
  + Resultado de suma
  + Resultado de resta
  + Resultado de multiplicación
  + Resultado de división
* Proceso:
  + Ingresar dos valores
  + Sumar los valores
  + Restar los valores
  + Multiplicar los valoree
  + Dividir los resultados
  + Mostrar el resultados o pedir número diferente de 0

FASE DE DISENO

|  |
| --- |
| Entidad: |
| Variables  N1, n2 |
| Nombre de algoritmo: ejercicios matematicos  Algoritmo  Inicio   * Ingresar valor n1 * Ingresar valor n2 * Sumar valores n1 y n2 * Mostrar resultado * Restar valores n1 y n2 * Mostrar resultado * Multiplicar n1 y n2 * Mostrar resultado * Dividir n1 y n2 * Mostrar resultado o pedir número distinto de 0   Fin |

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



FASE DE ANALISIS

* Definición de problema: conversión de temperatura Fahrenheit en grados Celsius
* Datos de entrada
  + Ingresar la temperatura Fahrenheit
* Datos de salida
  + Conversión de la temperatura Fahrenheit a grados Celsius
* Proceso
  + Ingresar valor de temperatura Fahrenheit
  + Convertir el valor ingresado a grados Celsius
  + Mostrar el resultado

FASE DE DISENO

|  |
| --- |
| Entidad: |
| Variable  tF, tC |
| Nombre de algoritmo: conversor  Algoritmo  Inicio   * Ingresar valor de tF * Convertir tF en tC * Mostrar el ambos valores por pantalla   Fin |

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.

FASE DE ANALISIS

* Definición del problema: raíz de una ecuación de segundo grado
* Datos de entrada
  + Ingreso de los valores de la ecuación
* Datos de salida
  + Valores de las raíces
* Proceso
  + Ingresar valores de la ecuación
  + Calcular el discriminante
  + Obtener las raíces
  + Clasificar que tipo de raíz es

FASE DE DISENO

|  |
| --- |
| Entidad: |
| Variables  A,b,c,x1,x2 |
| Nombre del algoritmo: ecuación de segundo grado  Algoritmo  Inicio   * Ingresar valor de a * Ingresar valor de b * Ingresar valor de c * Calcular discriminante * Calcular raíz x1 * Calcular raíz x2 * Mostrar en pantalla las raíces y determinar qué tipo de raíces son   Fin |

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



FASE DE ANALISIS

* Definición de problema: generar una línea de lado a lado del lienzo y que se desplace de arriba-abajo por el lienzo
* Datos de entrada
  + Línea
  + Dirección
* Datos de salida
  + Lienzo dibujado
  + Línea dibujada
  + Línea recorriendo su recorrido
* Proceso
  + Se declara variable línea
  + Se declara variable dirección
  + Diseño de línea
  + Definir movimiento de línea
  + Bucle de movimiento
  + Mostrar línea sobre el lienzo

FASE DE DISENO

|  |
| --- |
| Entidad: lienzo |
| Variables  Línea, dir |
| Nombre de algoritmo: línea con desplazamiento dentro de lienzo  Algoritmo  Inicio   * Declarar variable línea * Declarar variable dir * Comenzar la línea desde las coordenadas iniciales del lienzo * Extender la línea en el ancho del lienzo * Crear bucle de movimiento de abajo a arriba * Mostrar en pantalla   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:



**FASE DE ANALISIS**

* Definir problema: crear lienzo con rectángulos iguales en su interior
* Datos entradas:
  + Coordenadas de rectángulo: coordenadas-cartesianas
  + Ancho de rectángulo
  + Alto de rectangulo
* Datos de salida
  + Rectángulos dibujados en el lienzo
* Proceso
  + Dibujar los rectangulos

**FASE DISENO**

|  |
| --- |
| Entidad:-lienzo |
| Variables  CoordenadasRect: Coordenadas  ancho, alto, distEntreRect  anchoLienzo, altoLienzo |
| Nombre de Algoritmo:-dibujar rectangulos  Algoritmo:  Inicio   * AnchoLienzo * AltoLienzo * distanciaEntreRect * ancho * alto * dibujar un rectángulo en (x,coordenadasRect.y) condimensiones ancho y alto   Final |

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.

Resolución

**FASE DE ANALISIS**

* Definición de problema: dibujar escalones sobre el lienzo y colocar sobre cada escalon un punto rojo
* Datos de entrada: coordenadas cartesianas en 2D
* Punto A
* Punto B
* Punto C
* Punto D
* distLinea: entero
* Datos de salida:
* El dibujo de la línea horizontal
* El dibujo de la línea vertical
* El dibujo del punto rojo
* Proceso
* Dibujar una línea horizontal entre los puntos A y B, con una distancia igual a distLinea
* Dibujar una línea vertical entre los puntos B y C, con una distancia igual a distLinea
* Dibujar un punto en la siguiente posición: x= posición en x de B, y= posición en y de B – 5 unidades
* Actualizar las coordenadas de punta con las de puntoC
* Repetir desde el principio hasta que la coordenada en y de puntoA sea mayor que el alto del lienzo

**FASE DE DISEÑO**

|  |
| --- |
| Entidad: escalon |
| Variables  PuntoA, puntoB, puntoC, puntoD,: coordenadas cartesianas en 2D  distLinea: entero |
| Nombre algoritmo: dibujar escalon  Algoritmo  Inicio   * Dibujar una línea horizontal entre los puntos A y B, con distancia distLinea * Dibujar una línea vertical entre los puntos B y C< con distancia distLinea * dibujar\_circulo   Fin |
| Nombre algoritmo: dibujar\_circulo  Algoritmo  Inicio   * Dibujar un punto en la siguiente posición: x= posición en x de B, y= posición en y de B – 10 unidades   Fin |
| Nombre algoritmo: actualizar coordenadas A  Algoritmo  Inicio  puntoA,x<-puntoC,x  puntoA,y<-puntoC,y  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.

FASE DE ANALISIS

* Definir problema: dividir lienzo en franjas de igual medida y dibujar círculos con distanciamiento línea de por medio
* Datos de entrada:
  + Línea sola
  + Línea con circulo
  + Circulo
  + Distancia de circulo
* Datos de salida
  + Lienzo dibujado
* Proceso
  + Ingresar valor de línea 1
  + Ingresar valor de línea 2
  + Ingresar valor de circulo
  + Ingresar valor de distancia
  + Definir el diseño del lienzo
  + Definir el diseño de los círculos
  + Crear bule de repetición

FASE DE DISENO

|  |
| --- |
| Entidad: lienzo de líneas y círculos |
| Variables  lineaX, lineaY, circuloY, distanciaCirculo |
| Nombre de algoritmo: lienzo de líneas y círculos  Algoritmo  Inicio   * Ingresar valor de lineaX * Ingresar valor de lineaY * Ingresar valor de circuloY * Ingresar valor de distanciaCirculo * introducir las distancia entre los círculos * diseño de lienzo * diseño de las linaes * diseño de los círculos * bucle de repeticion   Final |