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

Trabajo Práctico/Actividad

N° 1

Apellido y Nombre:

Sanchez, Santiago Carlos Ezequiel

LU: TUV000774

*Profesores:*

*Mg. Ing. Ariel Alejandro Vega*

*Primer Año*

**Desarrollo:**

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

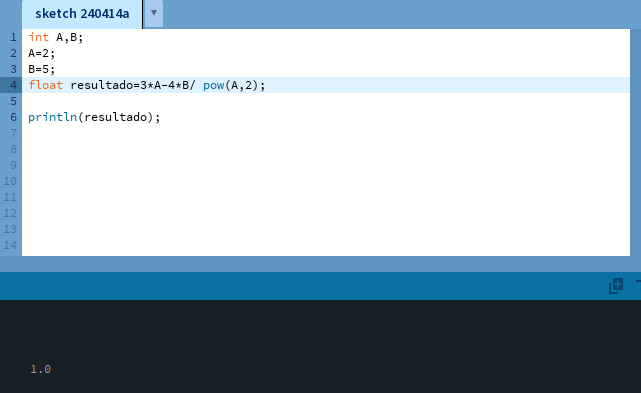
3\* A - 4 \* B / A ^ 2

Resolución necesaria en Word:

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

6-(4\*B/4)

6-5

1

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

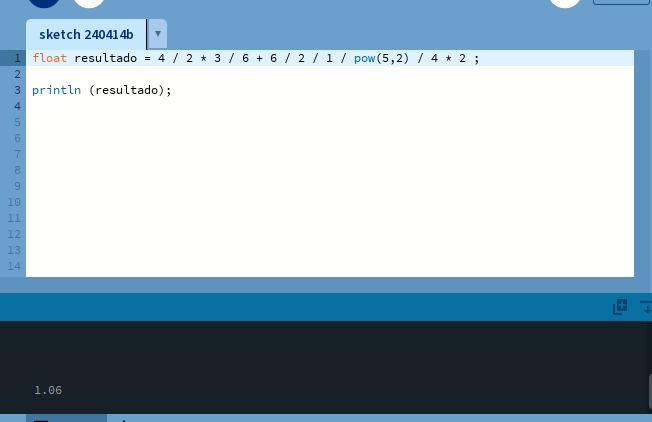
(4 / 2 \* 3 / 6) + (6 / 2 / 1 / (5 ^ 2) / 4 \* 2)

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

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

1 + 0,06

1,06

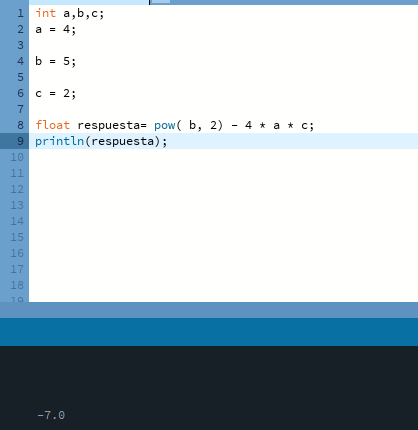


Ejercicio 4: Evaluar las siguientes expresiones aritméticas, para lo cual indicar en el caso

de las variables, el valor indicado. Luego escribirlas como expresiones algebraicas.

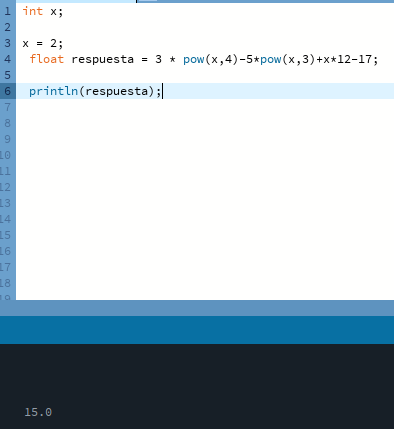
a) b ^ 2 – 4 \* a \* c a = 4 b = 5 c = 2

|  |  |
| --- | --- |
| Expresión aritmética | Expresión algebraica |
| (b ^ 2) – (4 \* a \* c)  25 – (4 \* a \* c)  25 – 32  -7 |  |



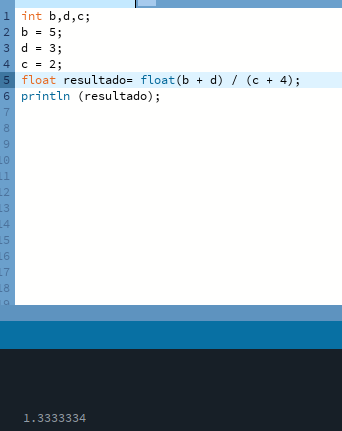
b) 3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17 x = 2

|  |  |
| --- | --- |
| Expresión aritmética | Expresión algebraica |
| (3 \* (x ^ 4)) – (5 \* (x ^ 3)) +( x \* 12) - 17  (3 \* 16) - (5 \* 8) + (x \* 12) – 17  48 - 40 + 24 – 17  15 |  |



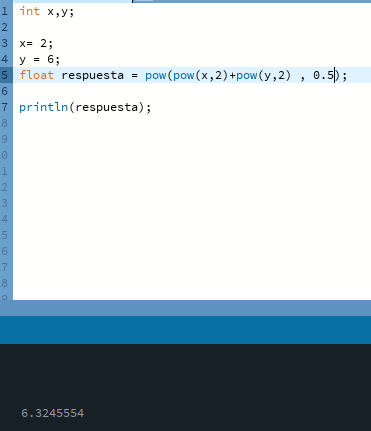
c) (b + d) / (c + 4) b = 5 d = 3 c = 2

|  |  |
| --- | --- |
| Expresión aritmética | Expresión algebraica |
| (b + d) / (c + 4)  8 / 6  1,333333 |  |



d) (x ^ 2 + y ^ 2) ^ (1 / 2) x = 2 y = 6

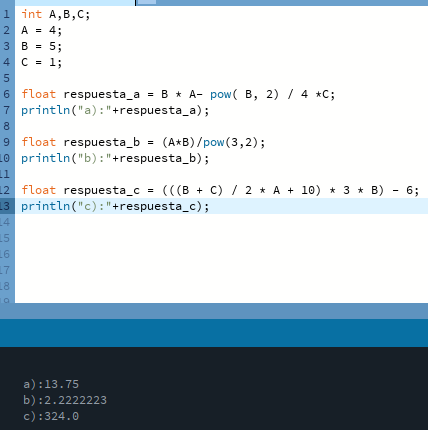
|  |  |
| --- | --- |
| Expresión aritmética | Expresión algebraica |
| ((x ^ 2) + (y ^ 2)) ^ (1 / 2)  (4 + 36) ^ (1 / 2)  40 ^ (1 / 2)  40 ^ 0,5  6,324555 |  |



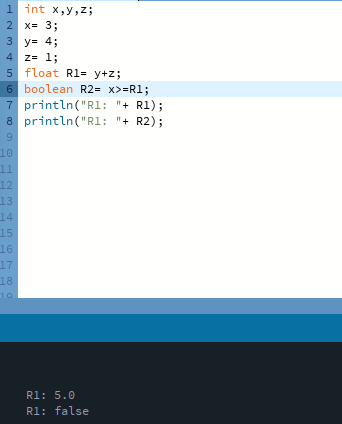
Ejercicio 5: Si el valor de A es 4, el valor de B es 5 y el valor de C es 1, evaluar las siguientes

expresiones:

|  |  |  |
| --- | --- | --- |
| a) B \* A – B ^ 2 / 4 \* C | b) (A \* B) / 3 ^ 2 | c) (((B + C) / 2 \* A + 10) \* 3 \* B) – 6 |
| (B \* A) – (((B ^ 2) / 4) \* C)  20 – ((25 / 4) \* C)  20 - (6,25 \* C)  20 - 6,25  13,75 | ((A\*B)) / (3 ^ 2)  20 / (3 ^ 2)  20 / 9  2,22222222 | (((B + C) / 2 \* A + 10) \* 3 \* B) – 6  ((6 / 2 \* A + 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= 4+1

R1= 5

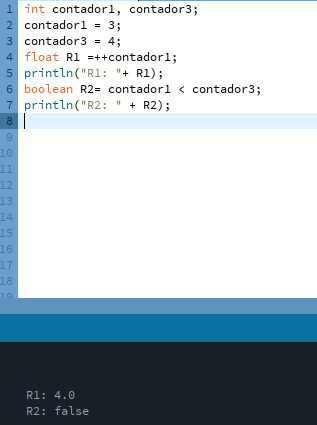
R2 = x >= R1

R2= 3 >= 5

Falso

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

R1 = ++contador1

R1 = (1 + contador1) //contador1 se le suma 1 y ahora vale 4 y R1 vale contador1 que es 4

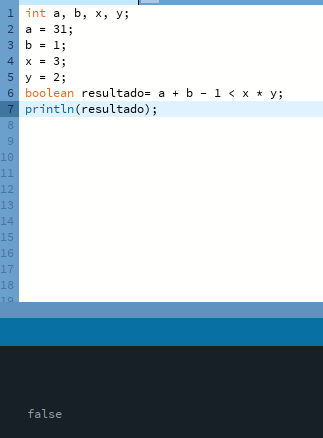
R1 = 4

R2 = contador1 < contador3

R2 = 4 < 4

R2 = Falso

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

a+b-1 < x\*y

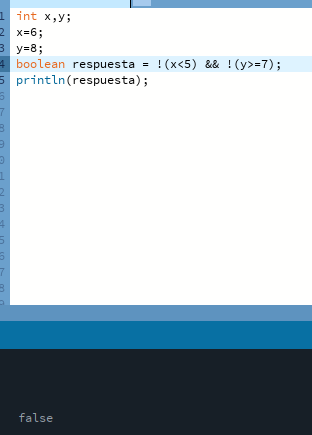
31 + 1 – 1 < 3 \* 2

31 + 1 – 1 < 6

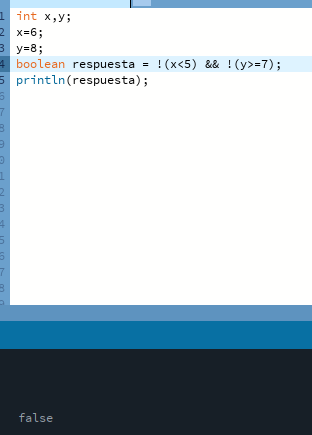
31 < 6

Falso

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

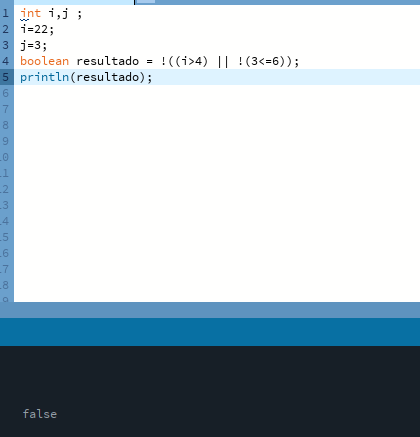
! (x<5) CC! (y>=7)

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

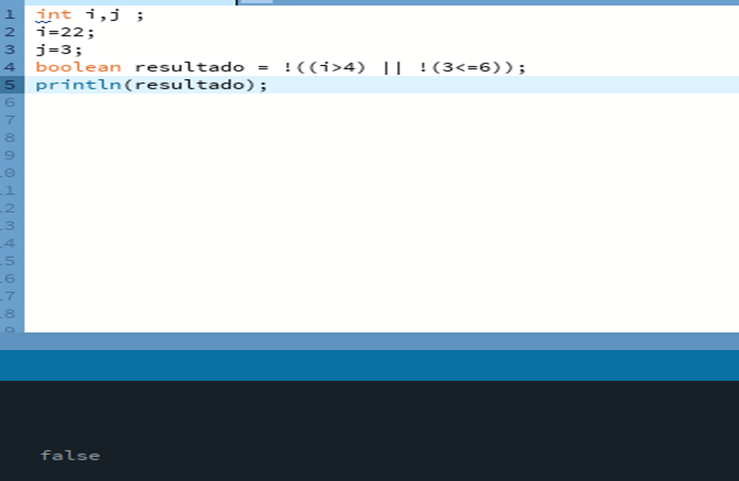
! (falso) CC! (verdadero)

Verdadero CC falso

Falso

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

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

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

! (¡verdadero ||! verdadero)

! (verdadero || falso)

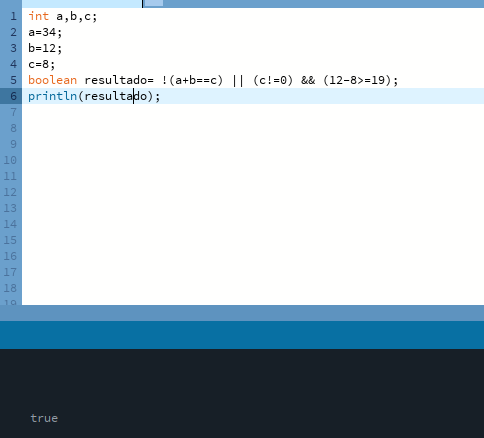
! verdadero

falso

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

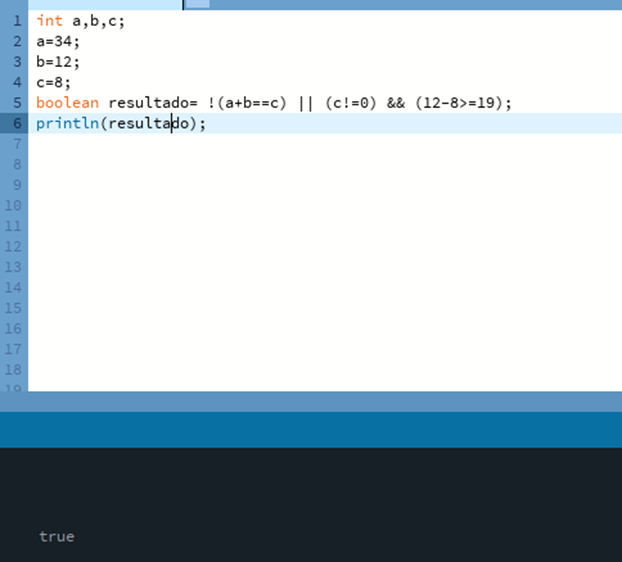
! (aba==c) || (¡c! =0) CC(b-c>=19)

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

! falso || verdadero CC falso

Verdadero || verdadero CC 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 del usuario

**Datos de Salida:**

saludo por pantalla

**Proceso: ¿Quién lo realiza?**

El algoritmo

**¿Qué proceso realiza?**

 El usuario ingresa su nombre, se lee el nombre y se devuelve el nombre junto con un saludo que se muestra en la pantalla

|  |
| --- |
| Entidad que resuelve el problema: algoritmo |
| Variables:  Usuario: String //el nombre del usuario  Saludo1: String //la primera parte del saludo que se realizara  Saludo2: String// segunda parte del saludo  Saludo3: String //donde se guardarán los saludos y el nombre del usuario |
| Nombre del algoritmo: saludo\_al\_usuario |
| 1 inicio  2 saludo1🡨 Hola  3 saludo2🡨 siéntete bienvenido  4 *leer* Nusuario  5 saludo3🡨 saludo1 + Nusuario + saludo2  6 *mostrar* saludo3 |

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 y altura

**Datos de Salida:**

Perímetro y área

**Proceso: ¿Quién lo realiza?**

El usuario

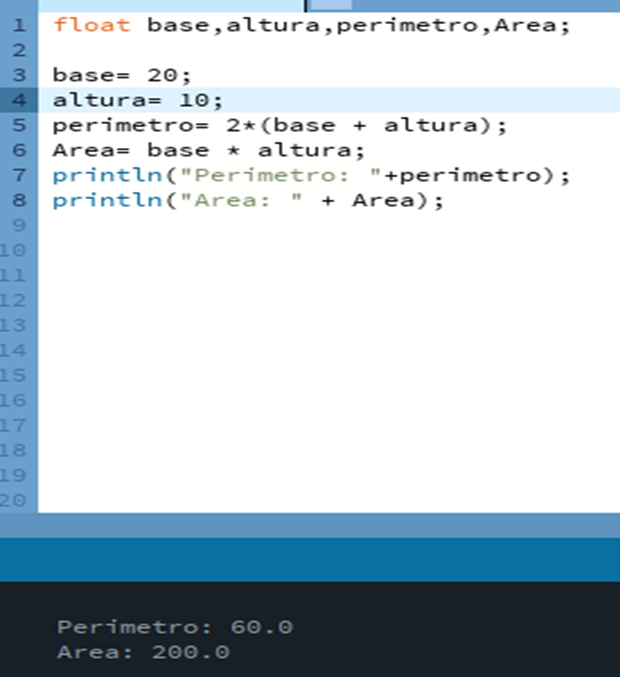
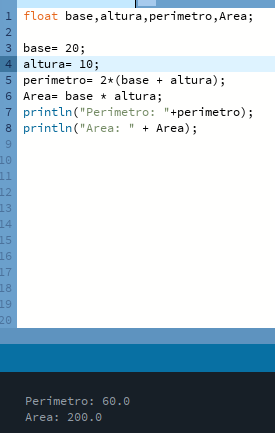
**¿Qué proceso realiza?**

Genera el perímetro multiplicando por 2 la suma de l base y la altura

Y luego calcula el área multiplicando base por altura

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: persona** |
| **Variables:**   * **base: float //** almacena un valor decimal * **altura: float** // almacena un valor decimal * **perímetro: float //** * **área: float //** |
| **Nombre del Algoritmo:** **Area\_y\_perimetro\_de\_un\_rectangulo** |
| **Proceso del algoritmo:**   1. *inicio* 2. *Leer base* 3. *Leer área* 4. *perímetro ← 2\*(base + altura)* 5. *área ← base \* altura* 6. *Mostrar perímetro* 7. *Mostrar área* 8. *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:**

cateto1 y cateto2

**Datos de Salida:**

hipotenusa

**Proceso: ¿Quién lo realiza?**

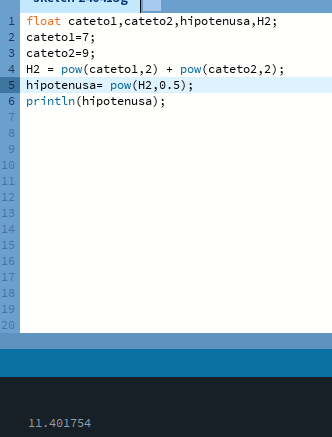
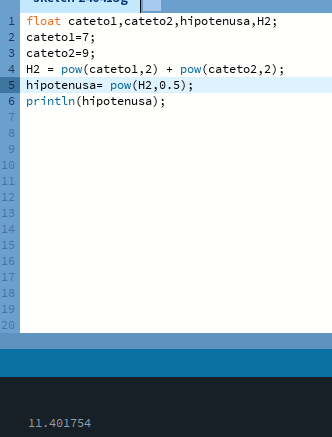
El usuario

**¿Qué proceso realiza?**

Aplica el teorema de Pitágoras elevando los dos catetos al cuadrado luego los suma y el al resultado se le aplica la raíz cuadrada

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: algoritmo** |
| **Variables:**   * **cateto1: float** * **cateto2: float** * **hipotenusa: float //el resultado final** * **H2: float //donde se guardará el resultado de la operación** |
| **Nombre del Algoritmo:** **Hipotenusa\_de\_un\_triangulo** |
| **Proceso del algoritmo:**   1. *inicio* 2. *Leer cateto1* 3. *Leer cateto2* 4. *H2🡨 cateto1 ^ 2 + cateto2 ^ 2* 5. *Hipotenusa🡨 h2^ (1/2)* 6. *Mostrar hipotenusa* 7. *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.

**Análisis:**

**Datos de Entrada:**

Num1y num2

**Datos de Salida:**

Suma, resta, multiplicación y división

**Proceso: ¿Quién lo realiza?**

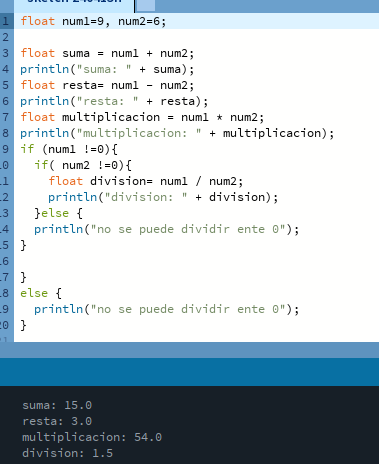
El usuario

**¿Qué proceso realiza?**

Utiliza los 2 umeros ingresados y hace las operaciones, pero si un numero es 0 la división no se realizará

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: programa** |
| **Variables:**   * **num1: entero // número que se va a sumar, restar, dividir, multiplicar** * **num2: entero // número que se va a sumar, restar, dividir, multiplicar** * **multiplicación: entero** * **división: float** * **suma: entero** * **resta: entero** |
| **Nombre del Algoritmo:** **operaciones\_conjuntas** |
| **Proceso del algoritmo:**   1. *inicio* 2. *Leer num1* 3. *Leer num2* 4. *Suma🡨 num1 + num2* 5. *Mostrar suma* 6. *Resta🡨 num1 - num2* 7. *Mostrar resta* 8. *Multiplicación🡨 num1\*num2* 9. *Mostar multiplicación* 10. *Si num1, num2 !=0* 11. *División🡨num1/num2* 12. *Mostrar División* 13. *Else* 14. *Mostar “no es posible dividir entre 0”* 15. *Fin si* 16. *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 ****

**Análisis:**

**Datos de Entrada:**

temperatura en Fahrenheit

**Datos de Salida:**

Temperatura en Celsius

**Proceso: ¿Quién lo realiza?**

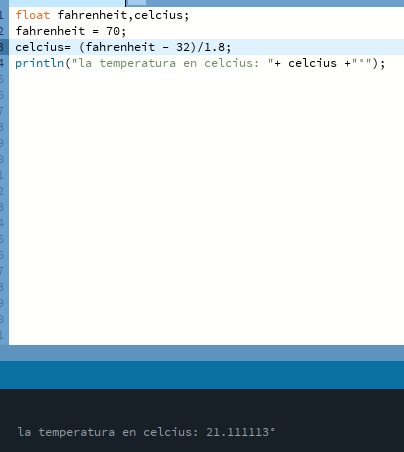
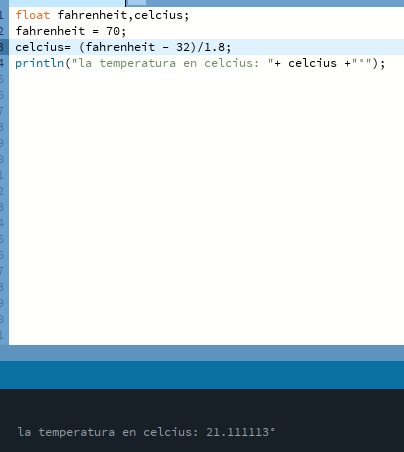
El programa

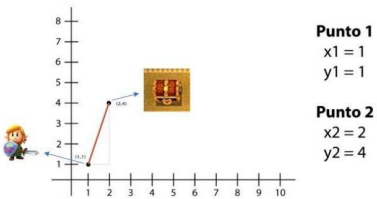
**¿Qué proceso realiza?**

Utiliza la temperatura en Fahrenheit le resta 32 y luego lo divide en 1.8

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: usuario** |
| **Variables:**   * **Fahrenheit: float// temperatura en Fahrenheit** * **Celsius: float // temperatura en Celsius** |
| **Nombre del Algoritmo:** **de\_Fahrenheit\_a\_celcius** |
| **Proceso del algoritmo:**   1. *Inicio* 2. *Leer Fahrenheit* 3. *Celsius🡨 (Fahrenheit – 32) /1.8* 4. *Mostrar Celsius* 5. *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.

**Análisis:**

**Datos de Entrada:**

Posición de link (x,y) posición del cofre (x,y)

**Datos de Salida:**

Distancia entre link y el tesoro

**Proceso: ¿Quién lo realiza?**

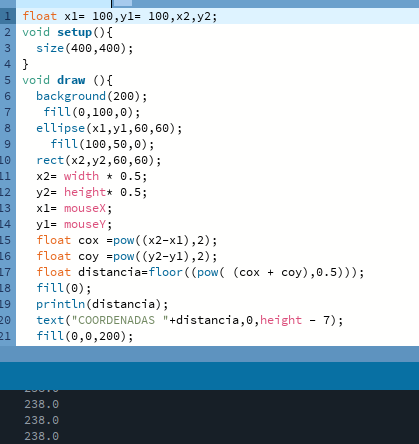
El programa

**¿Qué proceso realiza?**

Resta los x y los y con quien corresponda luego los eleva al cuadrado los vuelve a restar y al resultado le aplica la raíz obteniendo la distancia que separa a link del tesoro

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: programa** |
| **Variables:**   * **X1: float// posición en x de link** * **Y1: float// posición en y de link** * **X2float// posición en x del cofre** * **Y2: float//posición en y del cofre** * **Cox: float//resta de las x** * **Coy: float//resta de las y** * **Distancia: float/distancia entre link y el cofre** |
| **Nombre del Algoritmo:** **de\_Fahrenheit\_a\_celcius** |
| **Proceso del algoritmo:**   1. *Inicio* 2. *Leer x1* 3. *Leer y1* 4. *Leer x2* 5. *Leer y2* 6. *X1🡨100* 7. *Y1🡨100* 8. *X2🡨60* 9. *Y2🡨60* 10. *Cox🡨(x1-x2) ^2* 11. *Coy🡨 (y1-y2) ^2* 12. *Distancia🡨(Cox+coy) ^0.5* 13. *Escribir “coordenadas” + distancia* 14. *Si distancia <50* 15. *Escribir “GANASTE UN PREMIO”* 16. *Fin si* 17. *Fin* |





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.

**Análisis:**

**Datos de Entrada:**

Números a y c

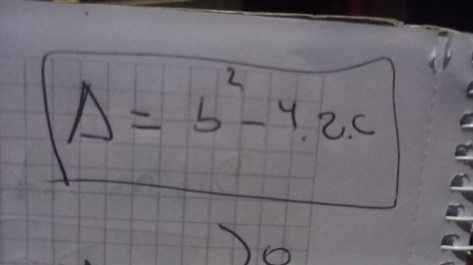
**Datos de Salida:**

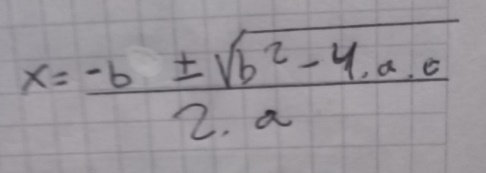
Valor del discriminante y valor de las raíces, a que números pertenecen y si son raíces iguales o no

**Proceso: ¿Quién lo realiza?**

El programa

**¿Qué proceso realiza?**

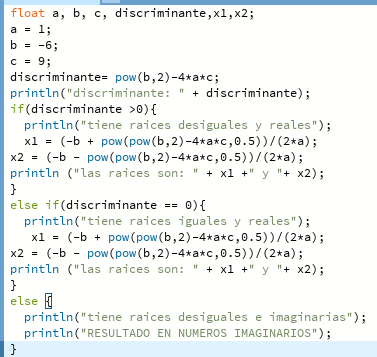
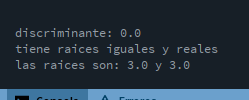
Primero calcula el discriminante con la siguiente formula, pero en ecuaciones de algoritmo

Luego verifica si el discriminante es mayor, menor o igual a cero, en el caso de que sea menor a cero se dará un mensaje que el resultado está en números imaginarios, caso contrario se realiza la ecuación para obtener las raíces

Y finalmente muestra las x (raíces)

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: programa** |
| **Variables:**   * **A: float** * **B: float** * **C: float** * **X1: float** * **X2: float** * **Discriminante: float** |
| **Nombre del Algoritmo:** **ecuación\_2\_grado** |
| **Proceso del algoritmo:**   1. *Inicio* 2. *Leer a* 3. *Leer b* 4. *Leer c* 5. *Discriminante🡨 b^2 + 4 \*a \*c* 6. *Mostrar “discrimi8nante: “+ discriminante* 7. *Si discriminante > 0* 8. *Mostrar “tiene raíces desiguales y reales”* 9. *X1🡨 (-b +( b^2 – 4\*a\*c) ^2) /(2\*a)* 10. *X2 🡨 (-b +( b^2 – 4\*a\*c) ^2) /(2\*a)* 11. *Mostrar “sus raíces son: “+ X1 +” y “+ x2* 12. *Pero si discriminante = 0* 13. *Mostrar “tiene raíces iguales y reales”* 14. *X1🡨 (-b +( b^2 – 4\*a\*c) ^2) /(2\*a)* 15. *X2 🡨 (-b +( b^2 – 4\*a\*c) ^2) /(2\*a)* 16. *Mostrar “sus raíces son: “+ X1 +” y “+ x2* 17. *Si no* 18. *Mostrar “resultado en números imaginarios”* 19. *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

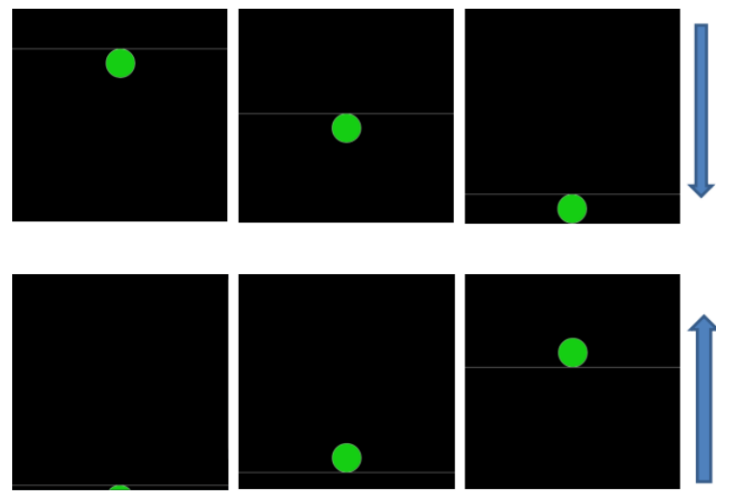
la elipse. Mantenga en cero el valor para background(). Cuando la línea supere la posición

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



**Análisis:**

**Datos de Entrada:**

Longitud de línea, alto y ancho de circulo, color del fondo

**Datos de Salida:**

Línea horizontal y circulo en bucle

**Proceso: ¿Quién lo realiza?**

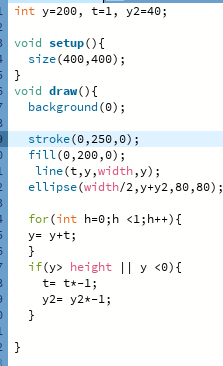
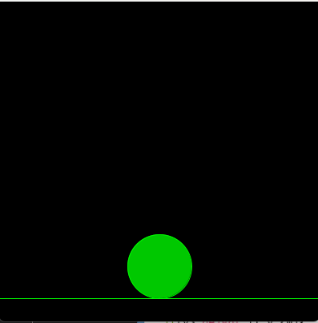
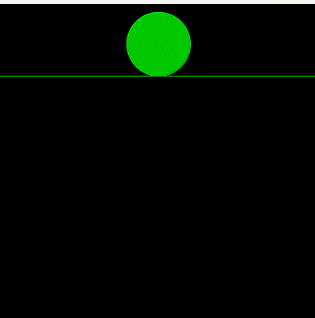
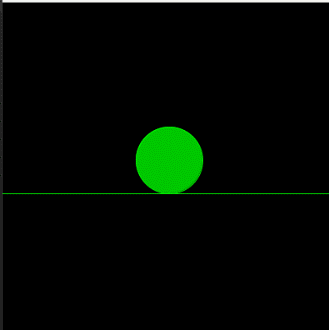
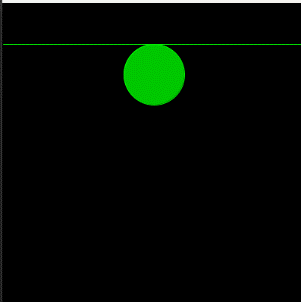
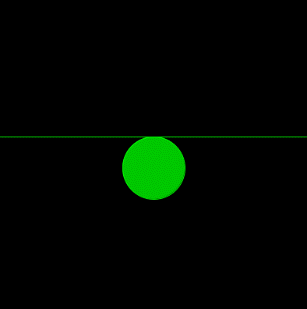
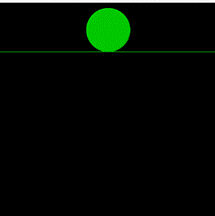
El programa

**¿Qué proceso realiza?**

Utiliza el valor en y tanto de la línea como del circulo y les va aumentando en uno hasta llegar a un extremo de la pantalla y luego cambia sumar a restar para volver al inicio e iniciar de nuevo

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: usuario** |
| **Variables:**   * **Y: entero// posición de la línea y el circulo** * **T: entero //almacena el número uno donde se va a sumar 1 a la variable y** * **Y2: entero// almacena 40 que se le va a sumara y en el circulo para poder cambiar de lado de la línea** |
| **Nombre del Algoritmo:** **de\_Fahrenheit\_a\_celcius** |
| **Proceso del algoritmo:**   1. *Inicio* 2. *Leer y* 3. *Leer t* 4. *Leer y2* 5. *Y🡨200* 6. *T🡨1* 7. *Y2🡨40* 8. *Crear pantalla de 400x, 400y* 9. *Definir fondo en 0 negro* 10. *Definir color de centro verde* 11. *Definir color de delineado en verde* 12. *Crear línea posición x1 en T posición y1 en Y posición x2 en width definir y2 en y* 13. *Crear circulo con alto y ancho de 80 en posición en x en height/2 posición y en Y-y2* 14. *For h=0; h<1; h +1* 15. *Y= y + t* 16. *Fin for* 17. *Si y>height o y <0* 18. *T = t\* -1* 19. *Y2 = y2\*-1* 20. *Fin* |



Ejercicio 20

**Definición del Problema:** Dibujar rectángulos en todo el lienzo

**Análisis:**

**Datos de entrada:**

DistEntreRect: Entero

Ancho: Entero

Alto: Entero

AltoDeLienzo: Entero

AnchoDeLienzo: Entero

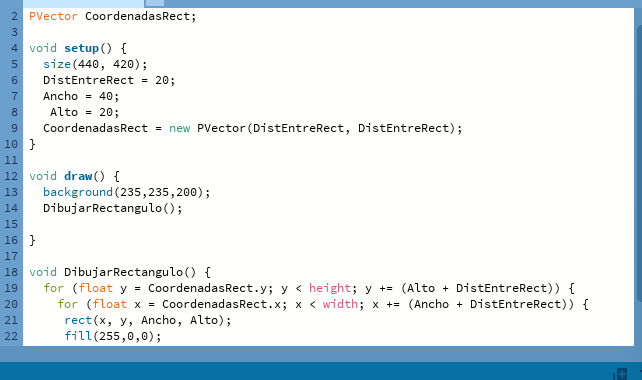
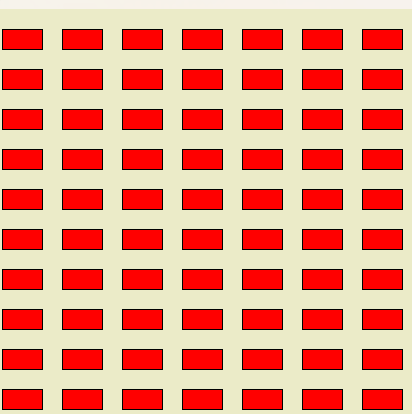
**Datos de Salida:**

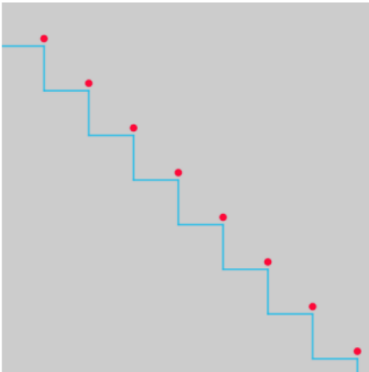
Los rectángulos dibujados en el lienzo

**Proceso:**

Establecer las medidas del lienzo

|  |
| --- |
| **Entidad que resuelve el problema: Rectangulos** |
| **VARIABLES:**  DistEntreRect:  Alto:  Ancho: |
| **Nombre del Algoritmo:** Dibujar\_Rectangulos  **Proceso del Algoritmo:**  Inicio  Ancho del lienzo🡨440,420  *DistEntreRect 🡨20*  *Ancho🡨40*  *Alto🡨20*  *CoordenadasRect🡨(DistEntreRect, DistEntreRect,)*  *DibujarRectangulo*  *Para y= CoordenadasRect. Y < heigth; y+= (Alto+ DistEntreRect)*  *Para x= CoordenadasRect. x < width; x+= (Alto+ DistEntreRect)*  *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**:

alto y ancho de la pantalla, puntoA, puntoB, puntoC, puntoD, distancia

**Datos de salida**:

Puntitos rojos encima de escalones celestes

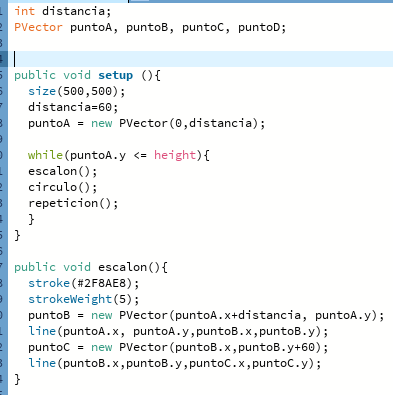
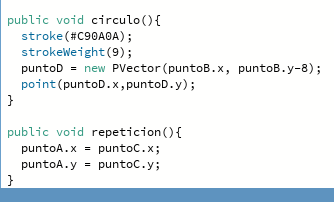
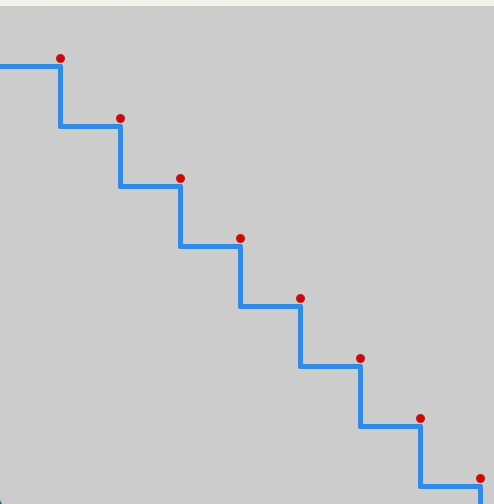
**Proceso:** ¿Quién realiza el proceso?

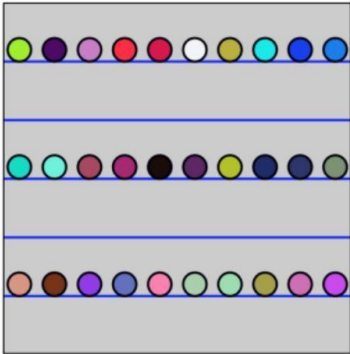
El programa  
¿Qué proceso realiza?

Dibuja rectángulos en el lienzo con las medidas que se especifican y por medio de un bucle se realizan los demás escalones y puntos

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el algoritmo**: Programa |
| Variables:   * puntoA, puntoB, puntoC, puntoD: int //almacena un vector * distancia: int //almacena un valor entero |
| Nombre del algoritmo: puntos\_y\_escalones |
| 1. *Inicio* 2. *Crear pantalla de 500 por 500* 3. *distancia ← 60* 4. *mientras (puntoA.y sea menor o igual que anchoLienzo) Hacer* 5. *dibujar línea horizontal en (puntoA.x, puntoA.y, puntoB.x, puntoB.y)* 6. *dibujar línea vertical en (puntoB.x, puntoB.y, puntoC.x, puntoC.y)* 7. *dibujar circulo en (puntoD.x, puntoD.y)* 8. *puntoA.x ← puntoC.x* 9. *puntoA.y ← puntoC.y* 10. *fin\_mientras* 11. *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:**

Líneas, cuantas líneas de por medio dejar para hacer los círculos, tamaño de la pantalla

**Datos de Salida:**

Líneas y círculos ordenados como se especifica

**Proceso: ¿Quién lo realiza?**

El programa

**¿Qué proceso realiza?**

Divide la pantalla en seis líneas y dibuja círculos de colores al azar dejando una línea de por medio

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: programa** |
| **Variables:**   * **rectaX :entero** * **rectaY: entero** * **circX :entero** * **circY: entero** * **distanciacirc: entero** |
| **Nombre del Algoritmo:** **círculos\_y\_lineas** |
| **Proceso del algoritmo:**   1. *Inicio* 2. *anchoLienzo ← 600* 3. *altoLienzo ← 600* 4. *rectaX ← 0* 5. *rectaY ← 100* 6. *distCirc ← 30;* 7. *circY ← 75* 8. *hacer* 9. *circX ← distCirc* 10. *hacer* 11. *dibujar linea en (rectaX, rectaY, anchoLienzo, rectaY)* 12. *dibujar circulo en (circX, circY, 50)* 13. *circX ← circX + distCirc\*2* 14. *fin\_hacer* 15. *mientras (circuloX < ancholienzo)* 16. *rectaY ← rectaY + 100;* 17. *circY ← circY + 200;* 18. *fin\_hacer* 19. *mientras (lineaY < altoLienzo)* 20. *Fin* |

