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

*Profesores:*

*Mg. Ing. Ariel Alejandro Vega*

*Ing. Carolina Cecilia Apaza*

*Año*

Trabajo Práctico N°1

Guzman Pablo Alberto – 000393

**FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS**

Indice

**Punto 1:** Obtener el resultado de la siguiente expresión para A = 2 y B = 5 del problema: 3\*A-4\*B/A^2

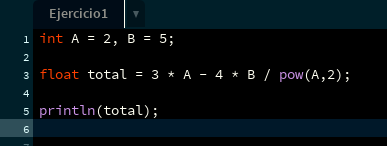
**Desarrollo del punto:**

(3\*2) – ((4\*5) / (2^2)) =

6 – (20 / 4) =

6 – 5 = 1

En Processing:



**Punto 2:** Evaluar la siguiente expresión 4/2\*3/6+6/2/1/5^2/4\*2

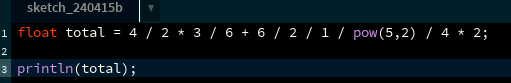
**Desarrollo del punto:**

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

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

1 + ((3 / 25) / 4) \* 2 = 1.06

En Processing:



**Punto 4:**

**Desarrollo del Punto:**

**Punto 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) 3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17 =

c) (b + d) / (c + 4) =

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

**Desarrollo del punto:**

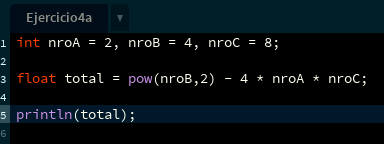
1. a = 2; b = 4; c = 8

4^2 – 4\*2\*8

16 – 64 = (-48)



En Processing:



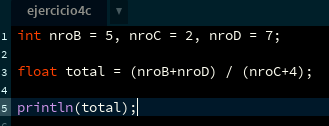
1. b = 5; c =2; d = 7

(5 + 7) / (2 + 4) =

12 / 6 = 2



En Processing:



1. X =2; y=4

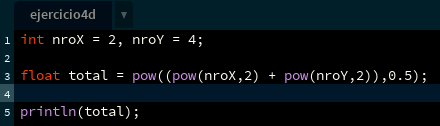
(2^2+4^2) ^ (1/2) =

(4+16) ^ (1/2) =

20 ^ (1/2) = 4,47



En Processing:



Punto 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

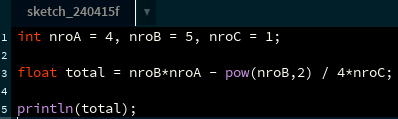
Desarrollo del Punto:

1. 5\*4 - 5^2 / 4\*1

20 – 25 / 4 =

20 – 6.25 = 13.75

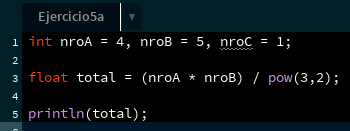
En Processing:



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

20 / 9 = 2.2

En Processing:



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

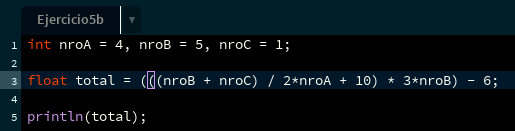
((6 / 2 \* 4 + 10) \* 15) – 6 =

((3\*4+10) \* 15) – 6 =

((12+10) \* 15) – 6 =

(22 \* 15) – 6 =

330 – 6 = 324

En Processing: 

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

R1 = y + z

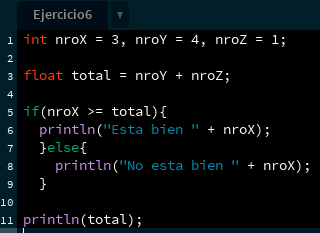
R2 = x >= R1

Desarrollo del Punto:

R1 = 4 + 1

R2 = 3 >= 5 (Falso)

En Processing:



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

a+b-1 < x\*y

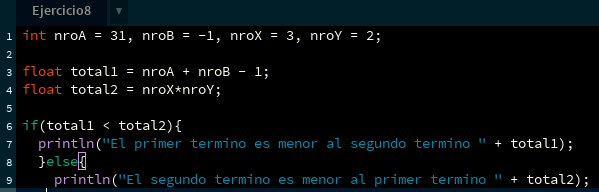
Desarrollo del Punto:

31 + (-1) – 1 < 3 \* 2

31 – 2 < 6

29 < 6 (Falso)

En Processing:



Punto 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.

Desarrollo del Punto:

Fase de Análisis

Especificación del Problema: Solicitar por teclado el nombre de un usuario y mostrarlo en pantalla con un saludo

Analisis:

Datos de Entrada:

nombre: String

Datos de Salida:

saludoFinal: String

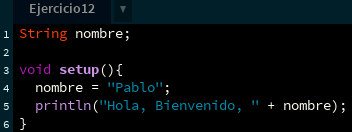
Proceso:

//Introducir el nombre del usuario

Fase de Diseño

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: ComputadoraSaludando |
| **VARIABLES:**  nombre: string  saludoFinal: String |
| **NOMBRE DEL ALGORITMO:** Saludo  **PROCESO DEL ALGORITMO**:   1. *Leer* nombre 2. *Mostrar* saludoFinal |

En Processing:



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

Desarrollo del Punto:

Fase de Análisis

Especificación del Problema: Calcular el perímetro y área de un rectángulo dada su base y altura

Analisis:

Datos de Entrada:

base, altura: int

Datos de Salida:

perímetro, área: float

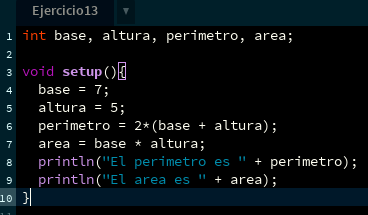
Proceso:

//Introducir el valor de la base y altura para obtener los resultados del perímetro y el area

Fase de Diseño

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Calculadora |
| **VARIABLES**:  base: int  altura: int |
| **NOMBRE DEL ALGORITMO**: Perimetro y área  **PROCESO DEL ALGORITMO**:   1. *Leer* base 2. *Leer* altura 3. perímetro <- 2(base + altura) 4. área <- base \* altura 5. *mostrar* perímetro 6. *mostrar* area |

En Processing:



Punto 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

Desarrollo del Punto:

Fase de Análisis

Especificación del Problema: Encontrar la Hipotenusa sabiendo los catetos

Analisis:

Datos de Entrada:

cateto1, cateto2: int

Datos de Salida:

hipotenusa: float

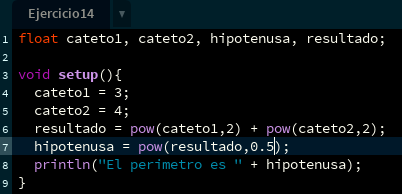
Proceso:

//Introducir el valor de los catetos para obtener la hipotenusa

Fase de Diseño

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Calculadora |
| **VARIABLES**:  cateto1: int  cateto2: int |
| **NOMBRE DEL ALGORITMO**: Teorema de Pitágoras  **PROCESO DEL ALGORITMO**:   1. *Leer* cateto1 2. *Leer* cateto2 3. hipotenusa^2 = cateto1^2 + cateto2^2 4. *mostrar* hipotenusa |

En Processing:



Punto 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

Desarrollo del Punto:

Fase de Análisis

Especificación del Problema: De dos números, hacer suma, resta, multiplicación, y división de los mismos

Analisis:

Datos de Entrada:

Numero1, numero2 = int

Datos de Salida:

suma, resta, multiplicación, division: float

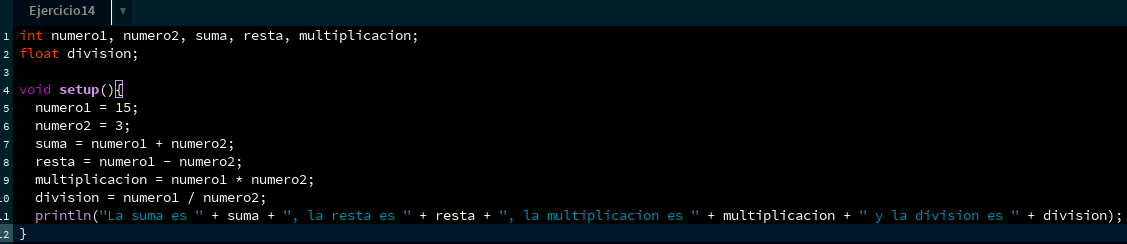
Proceso:

//Introducir el valor de dos números y calcularlos…

Fase de Diseño

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Calculadora |
| **VARIABLES**:  numero1: int  numero2: int  suma: int  resta: int  multiplicación: float  división: float |
| **NOMBRE DEL ALGORITMO**: Teorema de Pitágoras  **PROCESO DEL ALGORITMO**:   1. *Leer* numero1 2. *Leer* numero2 3. suma <- numero1 + numero2 4. resta <- numero1 - numero2 5. multiplicacion <- numero1 \* numero2 6. division <- numero1 / numero2 7. *mostrar* suma 8. *mostrar* resta 9. *mostrar* multiplicacion 10. *mostrar* division |
|  |

En Processing:



Punto 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



Desarrollo de Etapa;

Fase de Análisis

Especificación del Problema: Transformar la temperatura Fahrenheit en grados celsius

Analisis:

Datos de Entrada:

farenheit: int

Datos de Salida:

celsius: float

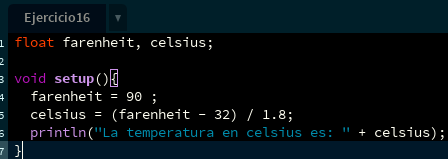
Proceso:

//Introducir el valor de la temperatura farenheit para convertirlo en celsius

Fase de Diseño

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Medidor |
| **VARIABLES**:  farenheit: int |
| **NOMBRE DEL ALGORITMO**: Temperatura  **PROCESO DEL ALGORITMO**:   1. *Leer* farenheit 2. celsius <- (farenheit-32) / 1.8 3. *mostrar* celsius |

En Processing:



Punto 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

Desarrollo de Punto:

Fase de Análisis

Especificación del Problema:

Analisis:

Datos de Entrada:

Datos de Salida:

Proceso:

//Introducir el valor de la temperatura farenheit para convertirlo en celsius

Fase de Diseño

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: |
| **VARIABLES**: |
| **NOMBRE DEL ALGORITMO**:  **PROCESO DEL ALGORITMO**: |

En Processing:

Punto 18:

Desarrollo del Punto:

Fase de Análisis

Especificación del Problema: Obtener las raíces de una ecuación de 2do grado.

Analisis:

Datos de Entrada:

numeroA: int

numeroB: int

numeroC: int

Datos de Salida:

r1 : float

r2 : float

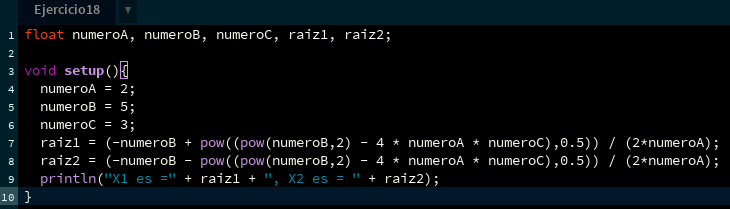
Proceso:

//Introducir los valores de 3 numeros para obtener las raices

Fase de Diseño

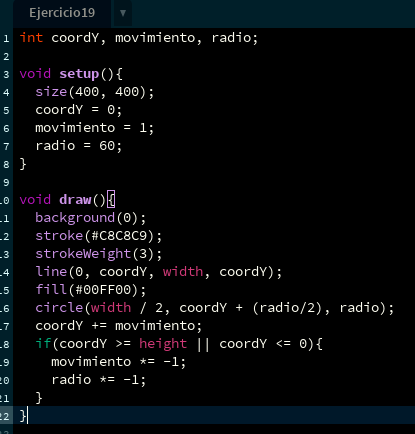
|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Calculadora |
| **VARIABLES**:  numeroA: int  numeroB: int  numeorC: int |
| **NOMBRE DEL ALGORITMO**: Temperatura  **PROCESO DEL ALGORITMO**:   1. *Leer* numeroA 2. *Leer* numeroB 3. *Leer* numeroC 4. raiz1 <- (-numeroB + (numeroB^2 – 4 \* numeroA\*numeroC) ^ (1/2)) / (2\*numeroA) 5. raiz2 <- (-numeroB - (numeroB^2 – 4 \* numeroA\*numeroC) ^ (1/2)) / (2\*numeroA) 6. *mostrar* raiz1 7. *mostrar* raiz2 |

En Processing:



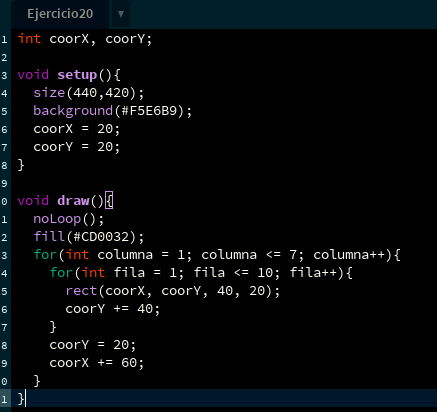
Punto 19: Declare las variables necesarias para dibujar una línea que se dibuja desde las coordenadas iniciales del lienzo y se extiende por todo el ancho. Sobre el punto medio de la línea y a una distancia de 40px (en sentido vertical desde la línea) dibuje una elipse que tenga como ancho 80px y de 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

En Processing:



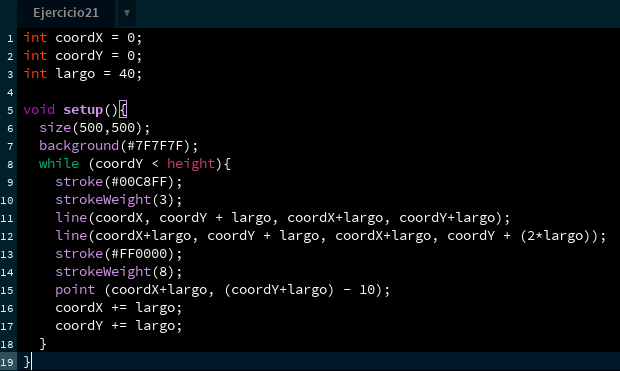
Punto 20: Dibuje en toda la extensión del lienzo de (440, 420) rectángulos de idénticas medidas (40 ancho y 20 de alto) y que mantengan una distancia de 20 pixeles entre ellos tanto horizontal como verticalmente. Utilice la estructura de control repetitiva for. El lienzo debería verse así:

En Processing:



Punto 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

En Processing:



Conclusión

Párrafos de las conclusiones

Fuentes bibliográficas

Se deben enunciar las fuentes (apuntes de la materia, páginas web, videos de youtube, libro (nombre, autores, año), etc)