|  |  |  |
| --- | --- | --- |
| 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*

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

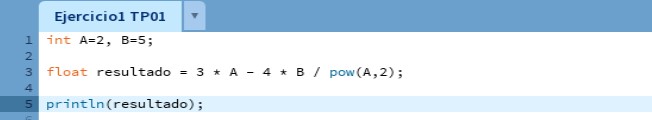
3\* A - 4 \* B / A ^ 2

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

6 - (4\*B / 4)

6 – 5

1



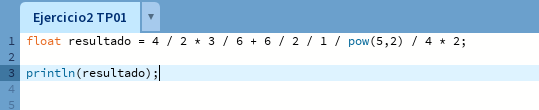
**Ejercicio 2**: Evaluar la siguiente expresión 4 / 2 \* 3 / 6 + 6 / 2 / 1 / 5 ^ 2 / 4 \* 2

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

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

1 + 0,06

1,06



**Ejercicio4**: Evaluar las siguientes expresiones aritméticas, para lo cual indicar en el caso

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

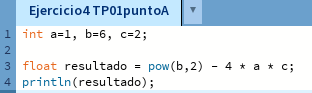
a) b ^ 2 – 4 \* a \* c Expresión algebraica: *b2 – 4.a.c*

Valores: a=1, b=6, c=2

(b ^ 2) – (4 \* a \* c)

36 – 8

28



b) 3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17 Expresión algebraica: *3.X4 - 5.X3 + X12 – 17*

Valores: X=2, X12=10

(3 \* (X ^ 4)) – (5 \* (X ^ 3)) + X12 – 17

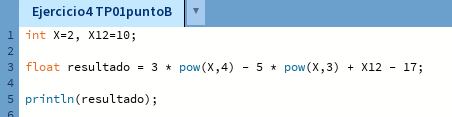
(3 \* 16) - (5 \* 8) + X12 – 17

48 – 40 + X12 – 17

8 + X12 – 17

18 – 17

1



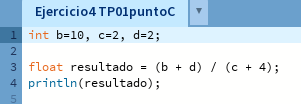
c) (b + d) / (c + 4) Expresión algebraica:

Valores: b=10, c=2, d=2

(b + d) / (c+4)

12 / 6

2



d) (x ^ 2 + y ^ 2) ^ (1 / 2) Expresión algebraica:

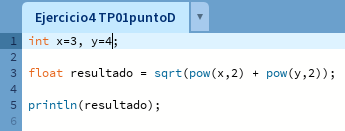
Valores: x=3, y=4

(x^2 + y^2) ^ (½)

(9 + 16) ^ (½)

25 ^ (½)

5



**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 ^ 2) / 4 \* C

20 – 25 / 4 \* C

20 – 6,25

13,75

b) (A \* B) / 3 ^ 2

20 / 9

2,22

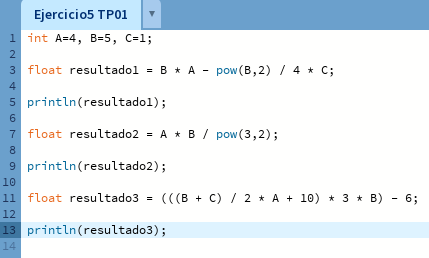
c) (((B + C) / 2 \* A + 10) \* 3 \* B) – 6

((6 / 2 \* A + 10) \* 3 \* B) - 6

(22 \* 3 \* B) -6

330 – 6

324



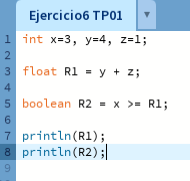
**Ejercicio 6:** Para x=3, y=4; z=1, evaluar el resultado de

R1 = y + z

R1 = 5

R2 = x >= R1

R2 = 3 >= 5



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

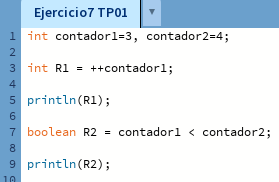
R1 = ++contador1

R1 = 3 + 1

R1 = 4

R2 = contador1 < contador2

R2 = 4 < 4



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

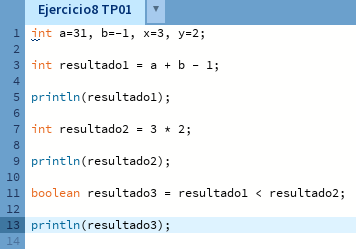
a+b-1 < x\*y

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

31 – 1 –1 < 6

29 < 6

falso



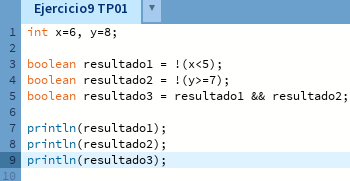
**Ejercicio 9:** Para x=6, y=8, evaluar el resultado de

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

**no** (x<5) **y no** (y>=7)

verdadero **y** falso

falso



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

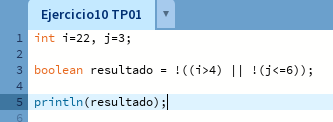
! ((i>4) ||! (j<=6))

**no** ((i>4) **o no** (j<=6))

**no** (verdadero **o** verdadero)

**no** (verdadero)

falso



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

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

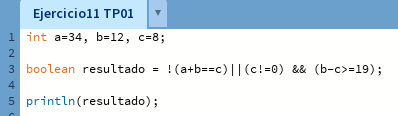
**no** (46 ==c) **o** (verdadero) **y** (4>=19)

**no (**(falso **o** verdadero) **y** falso)

**no** (verdadero **y** falso)

**no** 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.

**Definición del Problema:** pedir por teclado un nombre y después mostrar en pantalla el saludo más el nombre

**Análisis:**

* Datos de entrada:

nombre : string

saludo : string

* Datos de salida:

saludo y nombre : string

* Proceso:

mensaje = saludo + nombre

**Diseño del Algoritmo:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Presentación |
| **VARIABLES:**  nombre, saludo: string // almacena el nombre y el saludo en cadena de caracteres  mensaje: string //almacena el mensaje de respuesta |
| **NOMBRE ALGORITMO: s**aludo\_nombre  **PROCESO DEL ALGORITMO:**   1. *Leer* nombre 2. *Leer* saludo 3. mensaje ← saludo + nombre 4. *Mostrar* mensaje 5. Fin |

**Ejercicio 13:** Será común resolver problemas utilizando variables. Calcule el perímetro y

área de un rectángulo dada su base y su altura.

**Definición del Problema:** calcular el perímetro de un rectángulo.

**Análisis:**

* Datos de entrada:

base: entero

altura: entero

* Datos de salida:

perímetro: entero

* Proceso:

perímetro = base\*2 + altura\*2

**Diseño del Algoritmo:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Calculadora |
| **VARIABLES:**  base, altura: entero // datos de base y altura del rectángulo  perímetro: entero //resultado perímetro del rectángulo |
| **NOMBRE ALGORITMO:** calcular\_perimetro  **PROCESO DEL ALGORITMO:**   1. *Leer* base 2. *Leer* altura 3. perímetro ← base\*2 + altura\*2 4. *Mostrar* perímetro 5. 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

**Definición del Problema:** calcular la hipotenusa de un triángulo rectángulo.

**Análisis:**

* Datos de entrada:

cateto1: entero

cateto2: entero

* Datos de salida:

hipotenusa: entero

* Proceso:

hipotenusa = (cateto1^2 + cateto2^2) ^(½)

**Diseño del Algoritmo:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Pitágoras |
| **VARIABLES:**  cateto1, cateto2: entero  hipotenusa: entero |
| **NOMBRE ALGORITMO:** calcular\_hipotenusa  **PROCESO DEL ALGORITMO:**   1. *Leer* cateto1 2. *Leer* cateto2 3. hipotenusa ← (cateto1^2 + cateto2^2) ^(½) 4. *Mostrar* hipotenusa 5. 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.

**Definición del Problema:** calcular distintas operaciones con dos números.

**Análisis:**

* Datos de entrada:

numA, numB: Real // los números que se van a calcular

opción: string // almacena las opciones de distintas operaciones.

* Datos de salida:

resultado: Real // resultado de la operación que puede ser decimal

* Proceso:

// se asigna a opción la operación a realizar mediante texto

// si se cumple lo anterior se muestra del resultado de la operacion

// caso contrario el mensaje será

mensaje = “error”

**Diseño del Algoritmo:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Calculadora |
| **VARIABLES:**  numA, numB: Real  opción: string  resultado: Real |
| **NOMBRE ALGORITMO:** calcular\_operacion  **PROCESO DEL ALGORITMO:**   1. *Leer* numA 2. *Leer* numB 3. *Leer* opción 4. **según\_sea** (opción) hacer 5. **caso** suma: 6. resultado = numA + numB 7. mensaje ← resultado 8. sentencia de ruptura 9. **caso** resta: 10. resultado = numA - numB 11. mensaje ← resultado 12. sentencia de ruptura 13. **caso** multiplicacion: 14. resultado = numA \* numB 15. mensaje ← resultado 16. sentencia de ruptura 17. **caso** division: 18. resultado = numA / numB 19. mensaje ← resultado 20. sentencia de ruptura 21. **otros:** 22. mensaje ← “error” 23. sentencia de ruptura 24. **fin\_segun** 25. *Escribir* mensaje |

**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

temperaturaCelsius= (temperaturaFahrenheit – 32) /1,8

**Definición del Problema:** Convertir temperatura Fahrenheit a Celsius

**Análisis:**

* Datos de entrada:

temperaturaF: Real // almacena la temperatura Fahrenheit

* Datos de salida:

temperaturaC: Real

* Proceso:

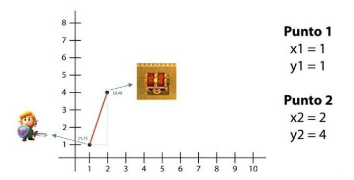
temperaturaC = (temperaturaF - 32) /1.8

**Diseño del Algoritmo:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Conversor |
| **VARIABLES:**  temperaturaF: Real |
| **NOMBRE ALGORITMO:** convertir\_temperatura  **PROCESO DEL ALGORITMO:**   1. *Leer* temperaturaF 2. temperaturaC ← (temperaturaF – 32) /1.8 3. *Mostrar* temperaturaC 4. 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 Link con un Circulo, y al tesoro con un cuadrado. Además, mueva a Link mediante el mouse.



**Definición del Problema:** Necesitamos obtener la distancia entre Link y el Cofre y que además interactúen

**Análisis:**

* Datos de entrada:

coordenadas de Link y el Cofre

x1, x2, y1, y2: entero //almacena los valores dados de la posición de Link y el Cofre

Green: Color // almacena un dato de color

mensaje: string // el mensaje una vez logrado el objetivo

* Datos de salida:

cateto1, cateto2: entero // los datos de los catetos del triángulo rectángulo

distancia: Real // la distancia entre los Link y el Cofre

* Proceso:

cateto1 = y2 – y1

cateto2 = x2 - x1

distancia = (cateto1^2 + cateto2^2) ^ (½)

**Diseño del Algoritmo:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Coordenadas |
| **VARIABLES:**  Link, Cofre: coordenadas //almacena las coordenadas en **x** e **y** para Link y el Cofre  x1, x2, y1, y2: entero //almacena los valores para calcular los catetos  Green: color // color del cofre  cateto1, cateto2: entero //los catetos del triangulo  distancia: Real //la hipotenusa del triangulo  mensaje: string |
| **NOMBRE ALGORITMO:** Calcular\_distancia\_entre\_Link\_cofre  **PROCESO DEL ALGORITMO:**   1. *Leer* x1 2. *Leer* x2 3. *Leer* y1 4. *Leer* y2 5. cateto1 <-- y2 - y1 6. cateto2 <-- x2 - x1 7. distancia <-- (cateto1^2 + cateto2^2) ^(½) 8. *Leer* color 9. *Mostrar* distancia 10. Dibujar un circulo en (Link.x,Link.y) con radio 20 11. Dibujar rectangulo en (Cofre.x,Cofre.y) con dimensiones 20 de ancho y 20 de alto 12. Link.x <-- mouseX 13. Link.y <-- mouseY 14. **si** mouseX + mouseY == Cofre.x + Cofre.y **entonces** 15. Green = 0 16. *Leer* mensaje <-- “Clear” 17. *Mostrar* mensaje 18. **fin\_si** 19. 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.

**Definición del Problema:** Necesitamos obtener las raíces de una ecuación de segundo grado

**Análisis:**

* Datos de entrada:

x1 =

* Datos de salida:

* Proceso:

**Diseño del Algoritmo:**

**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

**Definición del Problema:** Dibujar una elipse sobre una línea que baje y suba

**Análisis:**

* Datos de entrada:

cero, distancia: Coordenadas

anchoLienzo, altoLienzo: entero

stroke: entero

* Datos de salida:

línea

elipse

* Proceso:

// se inicia una línea con las coordenadas iniciales en 0 tanto en **x** como en **y,** termina en el ancho del lienzo

// se dibuja una elipse con una distancia de 40 ubicado en la mitad del lienzo

// se van incrementando los valores de las coordenadas **y** en la línea y en la elipse

//se coloca una condición que cuando las coordenadas lleguen a la altura del lienzo los valores vayan disminuyendo

**Diseño del Algoritmo:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Lienzo |
| **VARIABLES:**  cero, distancia: Coordenadas  anchoLienzo, altoLienzo: enteros  stroke: entero // color del borde dibujo  background: entero // color del lienzo |
| **NOMBRE ALGORITMO:** Dibujar\_linea\_elipse  **PROCESO DEL ALGORITMO:**   1. anchoLienzo <--- 500 2. altoLienzo <--- 500 3. stroke <--- 250 4. background <--- 0 5. dibujar una línea en (cero.x,cero.y,anchoLienzo,cero.y) 6. cero.y= cero.y + distancia.x 7. dibujar una elipse en (anchoLienzo/2, distancia.y,80,80) 8. **si** cero.y == altoLienzo **entonces** 9. distancia.y = altoLienzo – 40 10. distancia.x=distancia.x-1 11. **fin\_si** 12. **si** cero.y<0 **entonces** 13. distancia.y= 40 14. distancia.x = distancia.x+1 15. **fin\_si** 16. 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:

**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.

**Definición del Problema:** Dibujar escalones y sobre ellos un punto.

**Análisis:**

* Datos de entrada:

anchoEscalon, altoEscalon: coordenadas

anchoLienzo, altoLienzo: entero

strokeWeigth: entero // grosor del trazo

stroke: entero // color de los bordes

x, y: entero

distancia: entero // la distancia que hay entre los extremos de los escalones

* Datos de salida:

línea

punto

* Proceso:

// se dibujan lineas y puntos en determinadas coordenadas mientras se cumpla la condición

**Diseño del Algoritmo:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Lienzo |
| **VARIABLES:**  anchoEscalon: entero  altoEscalon: entero  anchoLienzo: entero  altoLienzo: entero  x: entero  y: entero  distancia: entero |
| **NOMBRE ALGORITMO:** Dibujar\_puntos\_lineas  **PROCESO DEL ALGORITMO:**   1. *Leer* x 2. *Leer* y 3. anchoLienzo<--500 4. altoLienzo<--500 5. *Leer* distancia 6. **mientras** (anchoEscalon.x <-- anchoLienzo) **hacer** 7. Dibujar linea en (anchoEscalon.x,anchoEscalon.y,anchoEscalon.y,anchoEscalon.y) 8. anchoEscalon.x += distancia 9. anchoEscalon.y += distancia 10. Dibujar linea en (altoEscalon.x,altoEscalon.x,altoEscalon.x,altoEscalon.y) 11. altoEscalon.x += distancia 12. altoEscalon.y += distancia 13. **fin\_mientras** 14. **mientras** (x<anchoLienzo) **hacer** 15. Dibujar punto en (x,y) 16. x +=distancia 17. y +=distancia 18. **fin\_mientras** 19. 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.

Conclusión

Se pudo entender las distintas maneras de lograr algo a través de las estructuras de códigos

que se proporciona en cada material de estudio que te hacen ver que lo que en realidad algo sencillo está formado por un sin fin de decisiones y posibilidades. Las condicionales que sirven para evaluar si algo es verdadero o falso te permite determinar el valor de algo y cambiar ese valor si así se lo precisa.

Párrafos de las conclusiones

Fuentes bibliográficas

libro Processing: Un lenguaje al alcance de todos Autor: Ignacio Buioli y Jaime Perez Marin Año:2013 version 02

apuntes materiales pdf y libros en aula virtual

https://virtual.unju.edu.ar/mod/resource/view.php?id=299570

https://youtu.be/\_YblzDgoAus

https://youtu.be/EAclc8NcnOo

https://youtu.be/bMJhygO43cA

https://youtu.be/LUfoF-tG8xw

https://youtu.be/Nr3NdAxjqsE

[Reference / Processing.org](https://processing.org/reference/)

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