

**Tecnicatura Universitaria en Diseño Integral de Videojuegos**

**FPOO**

**Trabajo Práctico N°1**

**Alumno:** Mateo Nicolás Goitea

**DNI:** 45977040

**LU:** TUV000720

**Profesor:** Ariel Alejandro Vega

**Año:** 2024

**Sección Expresiones aritméticas y lógicas**

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

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

**DESARROLLO:**

(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

**DESARROLLO:**

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

(2\* 3 / 6) + (6 / 2 / 1 / 5 ^ (0,5 \* 2))

(6 / 6) + (6 / 2 / 1 / 5 ^ 1)

1 + (6 / 2 / 1 / 5)

1 + (3 / 1 / 5)

1 + (3 / 5)

1 + (0,6)

1,6

**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

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

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

d) (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

**DESARROLLO:**

**a.** b² - 4.a.c

**b.** 3x⁴ - 5x³ + x12 - 17

**c.** b+d

c+4

**d.** (x²+ y²)½

**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

**DESARROLLO:**

**a.**

(B \* A) – (B ^ (2 / 4 \* C))

(5 \* 4) – (5 ^ (2 / 4 \* 1))

20 – (5 ^ (0,5))

20 – 2,23

17,77

**b.**

(A \* B) / 3 ^ 2

(4 \* 5) / 3 ^ 2

20 / 9

2,22…

**c.**

(((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

((12 + 10) \* 3 \* 5) – 6

(22 \* 3 \* 5) – 6

(66 \* 5) – 6

330– 6

324

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

R1 = y+z

R2 = x >= R1

**DESARROLLO:**

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

R2 = contador1 < contador2

**DESARROLLO:**

R1 = ++contador1 = 3

R2 = contador1 < contador2

R2 = 4 < 4 Falso

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

a+b-1 < x\*y

**DESARROLLO:**

a + b - 1 < x\*y

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

31 - 1 - 1 < 3\*2

31 - 1 - 1 < 3\*2

29 < 6 Falso

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

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

**DESARROLLO:**

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

!(6<5)&& !(8>=7) Falso

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

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

**DESARROLLO:**

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

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

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

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

**DESARROLLO:**

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

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

!(46==8) || (8!=0)&&(4>=19) Verdadero

**Sección Análisis – Diseño y Codificación de algoritmos – Aplicación de estructuras de**

**Control**

**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.

**DESARROLLO:**

**Fase de Análisis:**

**Definición del problema:** Ingresar un nombre y mostrar un saludo con el nombre ingresado

**Análisis del problema:**

* Datos de entrada: nombre: String
* Datos de salida: mensaje : String
* Proceso:

//añadir nombre ingresado al mensaje

Mensaje🡨mensaje+nombre

**Fase de Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** PersonaIngresandoNombre |
| **VARIABLES:** mensaje, nombre: String |
| **NOMBRE ALGORITMO:** ejercicio12  **ALGORITMO:**  Inicio  Leer nombre  mensaje🡨”Hola, “+nombre  Mostrar mensaje  Fin |

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

**DESARROLLO:**

**Fase de Análisis:**

**Definición del problema:** calcular perímetro y área de un rectángulo con los datos ingresados.

**Análisis del problema:**

* Datos de entrada: base, altura: Entero
* Datos de salida: perímetro, área: Entero
* Proceso:

//calcular perímetro

Perímetro🡨base\*2+altura\*2

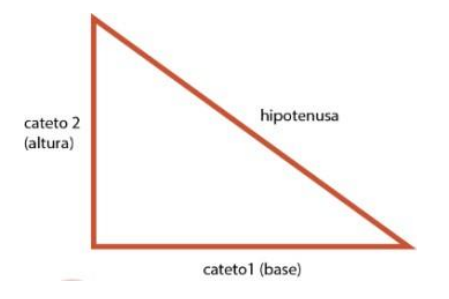
//calcular área

Área🡨base\*altura

**Fase de Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** PersonaIngresandoDatos |
| **VARIABLES:** base, altura, perímetro, área: int |
| **NOMBRE ALGORITMO:** ejercicio13  **ALGORITMO:**  Inicio  Leer base  Leer altura  perímetro🡨base\*2+altura\*2  área🡨base\*altura  Mostrar perímetro  Mostrar área  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



**DESARROLLO:**

**Fase de Análisis:**

**Definición del problema:** encontrar la hipotenusa de un triángulo conociendo los catetos

**Análisis del problema:**

* Datos de entrada: cateto1, cateto2: Entero
* Datos de salida: hipotenusa: Real
* Proceso:

//cálculo de la hipotenusa

Hipotenusa🡨√(cateto1)²+(cateto2)²

**Fase de Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** PersonaIngresandoDatos |
| **VARIABLES:** cateto1, cateto2: int  hipotenusa: float |
| **NOMBRE ALGORITMO:** ejercicio14  **ALGORITMO:**  Inicio  Leer cateto1  Leer cateto2  hipotenusa🡨 sqrt((pow(cateto1,2))+(pow(cateto2,2)),2)  Mostrar hipotenusa  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.

**DESARROLLO:**

**Fase de Análisis:**

**Definición del problema:** dados dos números, calcular su suma, resta, multiplicación y división

**Análisis del problema:**

* Datos de entrada: numero1, numero2: Entero
* Datos de salida: suma, resta, multiplicación: Entero

División: Real

* Proceso:

//calculo suma

Suma🡨numero1+numero2

//calculo resta

Resta🡨numero1-numero2

//calculo multiplicación

Multiplicación🡨numero1\*numero2

//calculo división

División🡨numero1/numero2

**Fase de Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** PersonaIngresandoDatos |
| **VARIABLES:** numero1, numero2, suma, resta, multiplicación:int  división:float |
| **NOMBRE ALGORITMO:**  **ALGORITMO:**  Inicio  Leer numero1  Leer numero2  suma🡨numero1+numero2  resta🡨numero1-numero2  multiplicación🡨numero1\*numero2  división🡨numero1/numero2  Mostrar suma  Mostrar resta  Mostrar multiplicación  Mostrar división  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



**DESARROLLO:**

**Fase de Análisis:**

**Definición del problema:** convertir una temperatura Farenheit en Celsius

**Análisis del problema:**

* Datos de entrada: temperatura farenheit: Entero
* Datos de salida: temperatura Celsius: Real
* Proceso:

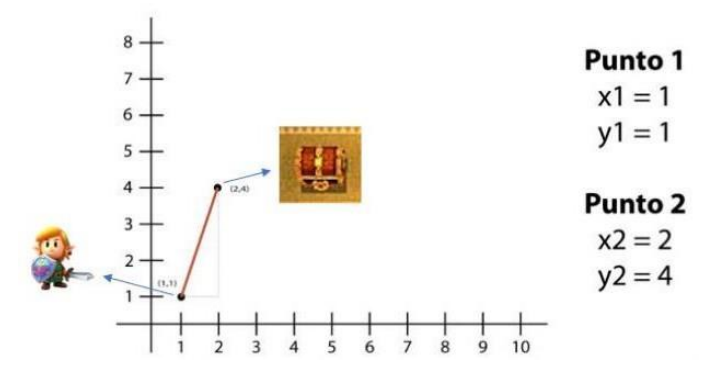
//cálculo temperatura farenheit a Celsius

TemperaturaCelsius🡨(temperaturaFarenheit-32)/1.8

**Fase de Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** PersonaIngresandoFarenheit |
| **VARIABLES:** tempFarenheit: int  tempCelsius: float |
| **NOMBRE ALGORITMO:** ejercicio16  **ALGORITMO:**  Inicio  Leer tempFarenheit  tempCelsius🡨(tempFarenheit-32)/1.8  Mostrar tempCelsius  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 Círculo, y al tesoro con un cuadrado. Además, mueva a Link mediante el mouse.

**DESARROLLO:**

**Fase de Análisis:**

**Definición del problema:** representar un PowerUp y a Link en un lienzo. Cuando Link se acerque a cierta distancia, el PowerUp desaparece. Link se moverá mediante el mouse.

**Análisis del problema:**

* Datos de entrada: x, powerX, powerY, linkX, linkY, cateto1, cateto2: Entero
* Datos de salida: distancia: Real
* Proceso:

//cálculo catetos

Cateto1🡨powerX-linkX

Cateto2🡨powery-linkY

//cálculo distancia

Distancia🡨√(cateto1)²+(cateto2)²

//condición para el PowerUp

Si distancia<x desaparecer PowerUp

**Fase de Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** PersonaControlandoLink |
| **VARIABLES:** powerX, powerY, linkX, linkY, anchoLienzo, altoLienzo, cateto1, cateto2: int  distancia: float |
| **NOMBRE ALGORITMO:** ejercicio17  **ALGORITMO:**  Inicio  ancholienzo🡨100  altolienzo🡨100  powerX🡨rand(0,100)  powerY🡨rand(0,100)  Dibujar rectángulo en (powerX, powerY) con ancho y alto (5, 5)  linkX🡨mouseX  linkY🡨mouseY  Mientras linkX y linkY >= 0 hacer  cateto1🡨powerX-linkX  cateto2🡨powerY-linkY  distancia🡨 sqrt((pow(cateto1,2))+(pow(cateto2,2)),2)  Si distancia<2 borrar rectángulo  Si no dibujar rectángulo en (powerX, powerY) con ancho y alto (5, 5)  Fin Mientras  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.

**DESARROLLO:**

**Fase de Análisis:**

**Definición del problema:** obtener las raíces de una ecuación de segundo grado. Utilizar el análisis de la discriminante de la ecuación cuadrática.

**Análisis del problema:**

* Datos de entrada: valor 1, valor2, valor3: Entero
* Datos de salida: x1, x2: Real
* Proceso:

//realizar el análisis de la discriminante

b²-4\*a\*c

//si la discriminante>0 calcular las 2 x

(-b±√b²-4\*a\*c)/2\*a

//si la discriminante=0 calcular una x e igualar las raíces

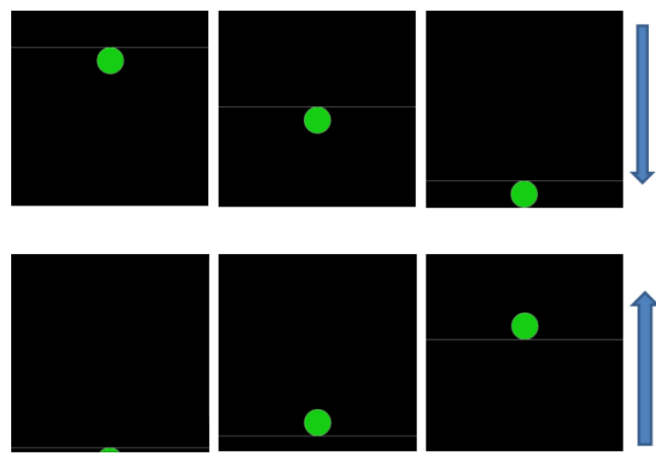
x1=x2

//si la discriminante<0 mostrar mensaje de que no hay raíces

**Fase de Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** PersonaIngresandoValores |
| **VARIABLES:** valor1, valor2, valor3: int  discriminante, x1, x2: float |
| **NOMBRE ALGORITMO:** ejercicio18  **ALGORITMO:**  Inicio  Leer valor1  Leer valor2  Leer valor3  discriminante🡨pow(valor2, 2)-(4\*valor1\*valor3)  Si discriminante>0 hacer  x1🡨(-(valor2)+(sqrt((pow(valor2, 2))-(4\*valor1\*valor3)), 2))/(2\*valor1)  x2🡨(-(valor2)-(sqrt((pow(valor2, 2))-(4\*valor1\*valor3)), 2))/(2\*valor1)  Mostrar x1  Mostrar x2  Sino Si discriminante=0 hacer  x1🡨(-(valor2)+(sqrt((pow(valor2, 2))-(4\*valor1\*valor3)), 2))/(2\*valor1)  x2🡨x1  Mostrar x1  Mostrar x2  Sino Si discriminante<0 hacer  Fin Si  Mostrar mensaje no hay raíces  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



**DESARROLLO:**

**Fase de Análisis:**

**Definición del problema:** realizar el gráfico indicado

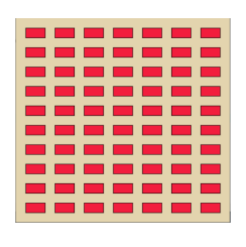
**Análisis del problema:**

* Datos de entrada: anchoLienzo, altoLienzo, x, y: Entero
* Datos de salida: gráfico realizado
* Proceso: dibujar el gráfico

**Fase de Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Lienzo |
| **VARIABLES:** anchoLienzo, altoLienzo, x, y: int |
| **NOMBRE ALGORITMO:** ejercicio19  **ALGORITMO:**  Inicio  anchoLienzo🡨800  altoLienzo🡨800  x🡨1  Mientras x=1 hacer  Para y🡨altolienzo hasta 0 con paso -1 hacer  Dibujar línea en (0, y, anchoLienzo, y)  Dibujar círculo en (anchoLienzo/2, y-40) con dimensiones (80, 80)  Fin Para  Si y=0 hacer  Para y🡨0 hasta altoLienzo con paso 1 hacer  Dibujar línea en (0, y, anchoLienzo, y)  Dibujar círculo en (anchoLienzo/2, y40) con dimensiones (80, 80)  Fin Para  Fin Si  Fin Mientras  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:



**DESARROLLO:**

**Fase de Análisis:**

**Definición del problema:** Dibujar en Processing la imagen indicada en el punto 20 del práctico.

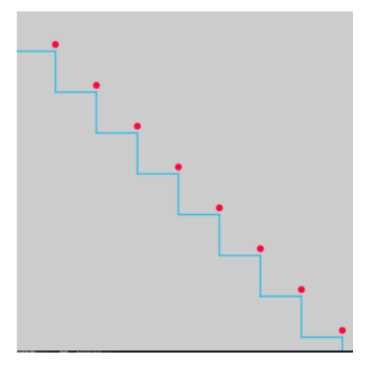
**Análisis del problema:**

* Datos de entrada: anchoLienzo, altoLienzo, anchoRect, altoRect, distanciaEntreRectangulos: Entero
* Datos de salida: Los rectángulos dibujados
* Proceso: Dibujar los cuadros

**Fase de Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Lienzo |
| **VARIABLES:** coordenadasRect: Coordenadas  ancho, alto, distEntreRect: int  anchoLienzo, altoLienzo: int |
| **NOMBRE ALGORITMO:** ejercicio20  **ALGORITMO:**  Inicio  anchoLienzo🡨440  altoLienzo🡨420  distanciaEntreRect🡨20  ancho🡨40  alto🡨20  Para x🡨coordenadasRect.x hasta anchoLienzo con paso (ancho+distEntreRect) hacer  Para y🡨coordenadasRect.y hasta altoLienzo con paso (alto+distEntreRect) hacer  Dibujar un rectángulo en (x, y) con dimensiones ancho y alto  Fin Para  Fin Para  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.

**DESARROLLO:**

**Fase de Análisis:**

**Definición del problema:**

**Análisis del problema:**

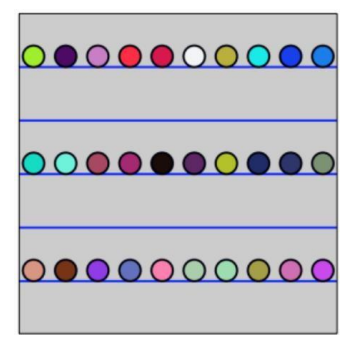
* Datos de entrada:
* Datos de salida:
* Proceso:

**Fase de Diseño:**

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



**DESARROLLO:**

**Fase de Análisis:**

**Definición del problema:** realizar el gráfico indicado.

**Análisis del problema:**

* Datos de entrada: anchoLienzo, altoLienzo, y, x: Entero
* Datos de salida: gráfico realizado
* Proceso: dibujar el gráfico

**Fase de Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Lienzo |
| **VARIABLES:** anchoLienzo, altoLienzo, y, x: int |
| **NOMBRE ALGORITMO:** ejercicio22  **ALGORITMO:**  Inicio  anchoLienzo🡨600  altoLienzo🡨600  y🡨0  Mientras y!=600 hacer  Dibujar línea en (0, y, anchoLienzo, y)  Si mod(y,200)!=0 hacer  Para x🡨0 hasta anchoLienzo con paso (ancho+dist) hacer  Dibujar un circulo en (, y+10) con dimensiones (alto, ancho)  Fin Para  Fin Si  y🡨y+100  Fin Mientras  Fin |