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

*Año 2024*

Trabajo Práctico N° 01

Guerra Facundo Nicolás

LU TUV000223

**FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS**

# Índice

Contenido

[Índice 2](#_Toc164172393)

[Ejercicio 1: 3](#_Toc164172394)

[Desarrollo del punto1 3](#_Toc164172395)

[Ejercicio 2: 3](#_Toc164172396)

[Desarrollo del puntO2 3](#_Toc164172397)

[Ejercicio 3: 4](#_Toc164172398)

[Ejercicio 4: 4](#_Toc164172399)

[Ejercicio 5: 5](#_Toc164172400)

[Ejercicio 6: 6](#_Toc164172401)

[Ejercicio 7: 6](#_Toc164172402)

[Ejercicio 8: 7](#_Toc164172403)

[Ejercicio 9: 7](#_Toc164172404)

[Ejercicio 10: 7](#_Toc164172405)

[Ejercicio 11: 8](#_Toc164172406)

[Ejercicio 12: 8](#_Toc164172407)

[Ejercicio 13: 9](#_Toc164172408)

[Ejercicio 14: 10](#_Toc164172409)

[Ejercicio 15: 10](#_Toc164172410)

**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

### Desarrollo del punto1

Resolución necesaria en Word:

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

6-(4\*B/4)

6-5

1

|  |  |
| --- | --- |
| CAPTURA DE PROCESSING. | RESULTADO |
|  |  |

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

### Desarrollo del puntO2

|  |  |
| --- | --- |
| CAPTURA DE PROCESSING. | RESULTADO |
|  |  |

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.

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.

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

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

1. Para a=1, b=2, c=3

|  |  |
| --- | --- |
| expresiones aritméticas | expresiones algebraicas |
|  |  |

|  |  |
| --- | --- |
| CAPTURA DE PROCESSING. | RESULTADO |
|  |  |

b) Para x=2

|  |  |
| --- | --- |
| expresiones aritméticas | expresiones algebraicas |
|  |  |
| CAPTURA DE PROCESSING. | RESULTADO |
|  |  |

c) Para b=5, c=6, d=7

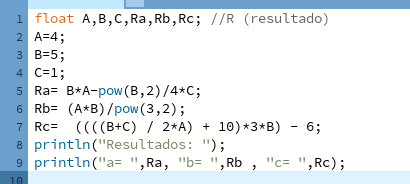
|  |  |
| --- | --- |
| expresiones aritméticas | expresiones algebraicas |
|  |  |
| CAPTURA DE PROCESSING. | RESULTADO |
|  |  |

d) Para x=2, y=3

|  |  |
| --- | --- |
| expresiones aritméticas | expresiones algebraicas |
|  | = 3.6055512 |
| CAPTURA DE PROCESSING. | RESULTADO |
|  |  |

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

Para A=4, B=5 C=1

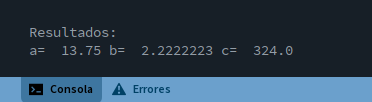
B \* A – B ^ 2 / 4 \* C

5 \* 4 – 5 ^ 2 / 4 \* 1

20 – 25 / 4

20 – 6,25 = 13,75

(A \* B) / 3 ^ 2

(4 \* 5) / 3 ^ 2

20/9

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

(((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 y R2 = x >= R1

R1= 4 + 1; R1 =5

R2= x > = R1; R2= 3 > = 5; R2=Falso

|  |  |
| --- | --- |
| CAPTURA DE PROCESSING. | RESULTADO |
|  |  |

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

R1 = ++contador1

R2 = contador1 < contador2

Ingresamos las variables contadores 1,2 y 3 →ingresamos R1 como variable entera, es igual Contador1 más 2 → Ingresamos R2 como tipo lógico y verificamos que contador 2 no tiene asignado un valor, resulta en Error.

Consideraremos contador3 = contador2

|  |  |
| --- | --- |
| CAPTURA DE PROCESSING. | RESULTADO |
|  |  |

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

Ingresamos variables tipo entero a, b, x, y, creamos un resultado tipo lógico para evaluar el problema.

a+b-1 < x\*y

31 + 1 – 1 < 3 \* 2

31 < 6

El resultado es falso

|  |  |
| --- | --- |
| CAPTURA DE PROCESSING. | RESULTADO |
|  |  |

Ejercicio 9: Para x=6, y=8, evaluar el resultado de !(x<5)CC !(y>=7).

Ingresamos datos de entrada x e y, como variables de tipo entero → Corroboramos primer y luego el segundo paréntesis, arrojan resultado de tipo lógico. → Comparamos ambas expresiones para ver que devuelve.

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

!(6<5) Compara con !(8>=7)

Expresion1 (Verdadero) ←→ Expresión 2 (Falso)

Resultado es Falso.

|  |  |
| --- | --- |
| CAPTURA DE PROCESSING. | RESULTADO |
|  |  |

Ejercicio 10: Para i=22, j=3, evaluar el resultado de !((i>4) || !(j<=6))

Ingresamos los datos de entrada, variable i y j

→ Les asignamos los valores.

→ Creamos una variable de tipo lógico para evaluar el problema

→ imprimimos por consola el resultado.

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

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

! ( Verdadero O !(Verdadero) )

! (Verdadero O Falso)

! Verdadero

Falso

|  |  |
| --- | --- |
| CAPTURA DE PROCESSING. | RESULTADO |
|  |  |

Ejercicio 11: Para a=34, b=12, c=8, evaluar el resultado de !(a+b==c) || (c!=0)CC(b-c>=19)

Ingresamos los datos de entrada → Les asignamos los valores. → Creamos una variable de tipo lógico para evaluar el problema → imprimimos por consola el resultado.

|  |  |
| --- | --- |
| CAPTURA DE PROCESSING. | RESULTADO |
|  |  |

**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

Pedir usuario que ingrese por teclado su nombre. Pedir que lo confirme. Mostrar en pantalla el saludo.

Diseño:

|  |
| --- |
| Entidad que resuelve el problema: Algoritmo |
| Variables: nombre: string |
| Proceso: |
| Inicia  Dar formato al lienzo  Pide ingresar nombre  Confirma el nombre  Muestra en pantalla el nombre. |
| Codificación |
|  |
| Muestra en pantalla |
|  |

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

Análisis: Aplicar las fórmulas matemáticas para resolver el problema.

Diseño:

|  |  |
| --- | --- |
| Entidad que resuelve el problema: Algoritmo | |
| Variables: base, altura tipo int | |
| Proceso: | |
| Inicia  Inicializamos las variables base y altura  Creamos una variable para cada respuesta: perímetro y área.  Imprimimos por consola los resultados. | |
| Codificación: | Resultado |
|  |  |

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: Aplicar el teorema de Pitágoras para resolver este ejercicio.

Crearemos los variables catetos e hipotenusa. Aplicamos su fórmula para resolver. Imprimo por consola los resultados.

Diseño:

|  |  |
| --- | --- |
| Entidad que resuelve el problema: Persona | |
| Variables: cateto1, cateto2, hipotenusa float | |
| Proceso: | |
| Inicia  Inicializamos las variables cateto1, cateto2.  Calcula la hipotenusa usando el teorema de Pitágoras.  Imprimimos por consola los resultados. | |
| Codificación: | Resultado |
|  |  |

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: Iniciar pidiendo dos variables para trabajar.→ Crear una confirmación→ Realizar un algoritmo usando ambos números.

Pedir los resultados y mostrar en consola.

Diseño:

|  |  |
| --- | --- |
| Entidad que resuelve el problema: Algoritmo | |
| Variables: float num1, num2,suma resta, multiplicación, división;  num1String, num2String String; Para luego convertirlos a números de tipo decimal.  ingresandoNum1 boolean; //Verificar el primer dato. | |
| Proceso: | |
| Declaro las variables para los números en tipo de dato Caracter y el estado de ingreso  Estado de ingreso del primer número  Dibujo la interfaz  Convierto los datos de tipo carácter a decimal.  Creo un condicional para que verifique que ingrese el primer numero  Confirmo el 2do número.  Calculo y muestro los resultados si se ingresaron ambos números.  Muestro los resultados.  Restablecer para permitir un nuevo cálculo vaciando los datos de tipo carácter. | |
| Codificación: | Resultado |
|  |  |

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

Análisis: Para realizar la conversión asignamos variables de ambas temperatura. Luego le asignaremos un valor a la de Fahrenheit para obtener el valor de la de Celsius, la cual obtendrá el nuevo valor por medio de la formula.

Diseño:

|  |  |
| --- | --- |
| Entidad que resuelve el problema: Algoritmo | |
| Variables: tempCelsius, tempFahrenheit float; | |
| Proceso: | |
| Inicia  Definir la temperatura en grados Fahrenheit  Calculamos la tempCelsius por medio de la fórmula.  Muestro el resultado | |
| Codificación: | Resultado |
|  |  |

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: Crearía dos Vectores, serían los puntos1 y punto2, representan a Link y al tesoro.

Estos serían PVectores almacenarían la posición con coordenadas.

Distancia= hipotenusa. Hipotenusa= raíz de, cateto1 = X1 – X2 mas cateto2 = Y1 – Y2.

Agregar condición que si acerca a una cierta distancia esta desaparecería. Con el condicional y usando una bandera de tipo lógica. Si no cambiara y ya no dibujara el tesoro

Posición del círculo estará definida por mouse.

Diseño:

|  |  |
| --- | --- |
| Entidad que resuelve el problema: Persona | |
| Variables: PVector linkPos, tesoroPos; Alamacenar coordenadas de link y el tesoro.  tesoroObtenido = falso ; lógico  distancia tipo decimal; | |
| Proceso: | |
| Inicia  Creamos y lee las variables.  Inicializamos las posiciones de Link y la caja de tesoro.  Dibujamos a Link en forma de circulo  Usamos condicional SI paso en VERDE la entrada igual para dibujar el tesoro, representado por un circulo por lo menos una vez.  Calculamos la distancia entre Link y el tesoro. Usando Pitágoras, donde se restan sus coordenadas en x de ambas y sus coordenadas en Y. Resultando la distancia igual a la hipotenusa.  En caso de que las distancia este muy cerca, cambiara nuestro paso a ROJO. Y dejara de dibujar el tesoro. | |
| Codificación: | Resultado |
|  |  |

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: Primero definir los coeficientes a, b, y c de una ecuación cuadrática.

Luego calcularíamos la discriminante utilizando la fórmula b^2 - 4ac.

Después, evalúa tres casos con el condicional SEGUN:

Si el discriminante es POSITIVO mayor que cero, hay dos raíces reales, por lo que calcula ambas raíces utilizando la fórmula cuadrática.

Si el discriminante es igual a cero, calculamos la raíz única.

Y si el discriminante es NEGATIVO, las raíces son imaginarias y no se pueden representar en los Reales.

Diseño:

|  |  |
| --- | --- |
| Entidad que resuelve el problema: Algoritmo | |
| Variables: Definimos los Coeficientes de la ecuación cuadrática → a, b, c  Resultado verificara la opción en el SEGÚN  Discriminante tipo decimal → Igual a la formula.  Raíces decimal → en caso de que sea la discriminante.  Una raíz única → en caso de que sea la discriminante. | |
| Proceso: | |
| Leer variables  Asigna valores a los coeficientes  Calculamos la discriminante  Si discriminante es positivo  Asigna a resultado valor de 1  Sino discriminante es 0  Asigna a un resultado 0;  Según sea el resultado:  Caso resultado sea 1 → Calcula las dos raíces con la formula → Imprime el resultado y termina;  Caso Resultado sea 0 → Calcula la única raíz con la formula → Imprime el resultado y termina;  Por defecto no positivo ni 0 → Entonces se entiende que es negativo → Imprime el resultado y termina; | |
| Codificación | Resultado |
|  |  |

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: Dibujar una línea, ubicarla en una posición, tiene que tener una longitud. Creamos Variables para su posición. x1,y1,x2,y2.

Creamos un elipse le damos una posición arriba y al centro de la línea. Definimos variables de posición de la elipse. Le damos color de relleno y fondo.

Hacemos que redibuje la línea y la elipse aumentando la posición en Y. Dando sensación de movimiento. Creamos una variable que nos definirá la velocidad de movimiento(mover), para la línea y para elipse.

Escribimos el condicional que nos dirá que si llegamos al final del lienzo superior o inferior, invierta la dirección. Dandole un valor opuesto al que tenía.

Diseño:

|  |  |
| --- | --- |
| Entidad que resuelve el problema: Algoritmo | |
| Variables: x1, y1, x2, y2 decimales ;  posElipseX, posElipseY decimales ;  moverLinea, moverElipse ambas valen 1; (dirección) | |
| Proceso: | |
| Inicia  Lee Coordenadas iniciales de la línea y del centro de la elipse  Lee Dirección de movimiento de la línea y del elipse  Asigna valores a la posición de la línea y de la elipse  Dibuja fondo negro → Dibuja Línea → Dibuja elipse  Suma en 1 valor de las posiciones en el eje Y de la Línea y del elipse  Si → valor posición de eje Y de la línea supera el máximo superior del lienzo o el inferior→ entra  Invertimos la dirección de la línea al multiplicar por -1.  Si la elipse supera la posición por debajo se coloca del otro lado de la línea.  Y invierte la dirección del movimiento de la elipse. Causa efecto de rebote.  Si no invierte su dirección hacia abajo nuevamente | |
| Codificación: | Resultado |
|  |  |

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:



Análisis: Dibujar Lienzo, Dibujar rectángulos distanciados uno de otro utilizando PARA

Datos de entrada:

Lienzo (ancho y largo)

Rectángulo (ancho y largo)

Distancia Espaciado

Datos de salida:

Dibujo terminado igual al de arriba

Diseño:

|  |  |
| --- | --- |
| Entidad que resuelve el problema: Algoritmo | |
| Variables: ancho y largo lienzo  Ancho y largo del rectángulo  Distancia espacio | |
| Proceso: | |
| Creamos la función dibujarRectangulo  Para x ← coordenadasRect.x hasta x mayor a ancho lienzo [ incremento ancho rectángulo más espacio]  Para y ← coordenadasRect.y hasta Y mayor a alto lienzo [ incrementa alto rectángulo más espacio]  Dibujar rectángulo en x , y  Rellenar color naranja el rectángulo  Dibujamos la función DibujarRectangulo | |
| Codificación: | Resultado |
|  |  |

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: Dibujar línea, creo la variables de posición de la línea, huella y luego la contrahuella.

Diseñamos el lienzo. Ejecutamos el algoritmo repetitivo mientras estemos dentro de límite del lienzo.

Diseño:

|  |  |
| --- | --- |
| Entidad que resuelve el problema: Algoritmo | |
| Variables: | |
| Proceso: | |
|  | |
| Codificación: | Resultado |
|  |  |

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

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)