**TRABAJO PRACTICO N1**

**MATERIA: FUNDAMENTO DE LA PROGRAMACION ORIENTADA A OBJETOS**

**N/A: Chimale, Luciano Karim**

**DNI: 44.281.397**

**LU: TUV000742**

**PROFESOR: Mg. Ing. Ariel Alejandro Vega**

**AÑO: 2024**

**DESARROLLO:**

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

**3\* A – 4\* B / A ^ 2**

Resolución necesaria en Word:

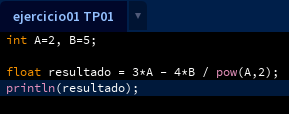
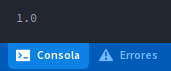
(3\*A)-(4\*B/(A^2))

6-(4\*B/4)

6-5

1

Captura de Processing: Resultado:



**Ejercicio 02:** Evaluar la siguiente expresión:

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

|  |  |
| --- | --- |
| **Aritmética** | **Algebraica** |
| **4 / 2 \* 3 / 6 + 6 / 2 / 1 / 5 ^ 2 / 4 \* 2**  (((4/2) \* 3) /6 ) + ((((6/2) / 1) / (5 ^ 2)) / 4) \* 2  1.0 + 0.06  1.06 |  |

|  |  |
| --- | --- |
| **Captura Processing** | **Resultado** |
|  |  |

**Ejercicio 03:** 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**  a= 2, b=4, c=1

|  |  |
| --- | --- |
| **Aritméticas** | **Algebraica** |
| **b ^ 2 – 4 \* a \* c**  (4 ^ 2) - (4 \* 2 \* 1)  16 – 8  8 |  |

|  |  |
| --- | --- |
| **Captura Processing** | **Resultado** |
|  | C:\Users\Luciano\AppData\Local\Microsoft\Windows\INetCache\Content.Word\resultado.png |

1. **3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17** x=5

|  |  |
| --- | --- |
| **Aritmética** | **Algebraica** |
| **3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17**  (3\*(5^4)) - (5 \* (5^3)) + (5\*12) – 17  1875 – 625 + 60 – 17  1293 |  |

|  |  |
| --- | --- |
| **Captura Processing** | **Resultado** |
| **C:\Users\Luciano\AppData\Local\Microsoft\Windows\INetCache\Content.Word\captura processing ejer 3.png** | C:\Users\Luciano\AppData\Local\Microsoft\Windows\INetCache\Content.Word\resultado.png |

1. **(b + d) / (c + 4)** b=2, c=1, d=4

|  |  |
| --- | --- |
| **Aritmética** | **Algebraica** |
| **(b + d) / (c + 4)**  (2 + 4) / (1 + 4)  1.2 |  |

|  |  |
| --- | --- |
| **Captura Processing** | **Resultado** |
|  | C:\Users\Luciano\AppData\Local\Microsoft\Windows\INetCache\Content.Word\captura processing resultado ejer 3.png |

1. **(x ^ 2 + y ^ 2) ^ (1 / 2)** x=2 y=3

|  |  |
| --- | --- |
| **Aritmética** | **Algebraica** |
| **(x ^ 2 + y ^ 2) ^ (1 / 2)**  (2^2 + 3^2) ^ (1 / 2)  13 ^ (1 / 2)  3.605512 |  |

|  |  |
| --- | --- |
| **Captura Processing** | **Resultado** |
|  | C:\Users\Luciano\AppData\Local\Microsoft\Windows\INetCache\Content.Word\captura resultado ejer 3.png |

**Ejercicio 04:** 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**

|  |  |
| --- | --- |
| **Aritmética** | **Algebraica** |
| **B \* A – B ^ 2 / 4 \* C**  5 \* 4 – ((5 ^ 2) / 4) \* 1  20 - 6.25  13.75 |  |

|  |  |
| --- | --- |
| **Captura Processing** | **Resultado** |
|  |  |

1. **(A \* B) / 3 ^ 2**

|  |  |
| --- | --- |
| **Aritmética** | **Algebraica** |
| **(A \* B) / 3 ^ 2**  (4 \* 5) / 3 ^ 2  2.2 ... |  |

|  |  |
| --- | --- |
| **Captura Processing** | **Resultado** |
|  |  |

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

|  |  |
| --- | --- |
| **Aritmética** | **Algebraica** |
| **(((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  (22 \* 3 \* 5) - 6  330 – 6  324 |  |

|  |  |
| --- | --- |
| **Captura Processing** | **Resultado** |
|  | C:\Users\Luciano\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Captura de pantalla 2024-04-14 145327.png |

**Ejercicio 05:** Para x=3, y=4, z=1, evaluar el resultado de:

R1= y+z

R2= x >=R1

|  |
| --- |
| R1= 4+1 = 5  R2= 3 >= R1  falso |

|  |  |
| --- | --- |
| **Captura Processing** | **Resultado** |
|  | C:\Users\Luciano\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Captura de pantalla 2024-04-14 150326.png |

**Ejercicio 06:** Para contador1=3, contador2=4, evaluar el resultado de R1=++contador1

R2=contador1 < contador2

|  |
| --- |
| R2= 4 < 4  R2= falso |

|  |  |
| --- | --- |
| **Captura Processing** | **Resultado** |
|  |  |

**Ejercicio 07:** Para a=31, b=-1; x=3, y=2, evaluar el resultado de a+b-1 < x\*y

|  |
| --- |
| a+b-1 < x\*y  31+(-1)-1 < 3\*2  29 < 12  Falso |

|  |  |
| --- | --- |
| **Captura Processing** | **Resultado** |
|  | C:\Users\Luciano\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Captura de pantalla 2024-04-14 151214.png |

**Ejercicio 08:** Para x=6, y=8, evaluar el resultado de ! (x<5) &&!(y>=7)

|  |
| --- |
| !(x<5) && !(y>=7)  !(6<5) && !(8>=7)  falso && falso  falso |

|  |  |
| --- | --- |
| **Captura Processing** | **Resultado** |
|  | C:\Users\Luciano\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Captura de pantalla 2024-04-14 151607.png |

**Ejercicio 09:** Para i=22, j=3, evaluar el resultado de !((i>4) || !(j<=6))

|  |
| --- |
| !((i>4) || !(j<=6))  !((22>4) || !(3<=6))  !(verdadero || falso)  !(verdadero)  Falso |

|  |  |
| --- | --- |
| **Captura Processing** | **Resultado** |
|  |  |

**Ejercicio 10:** Para a=34, b=12, c=8, evaluar el resultado de !(a+b==c) || (c!=0) && (b-c>=19)

|  |
| --- |
| !(a+b==c) || (c!=0) && (b-c>=19)  !(34+12==8) || (8!=0)&&(12-8>=19)  !(46==8) || (8!=0)&&(4>=19)  verdadero || verdadero && falso  verdadero|| falso  verdadero |

|  |  |
| --- | --- |
| **Captura Processing** | **Resultado** |
|  | C:\Users\Luciano\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Captura de pantalla 2024-04-14 152358.png |

**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 11:** Un problema sencillo. Deberia pedir por teclado al usuario un nombre y posteriormente realizara la presentación en pantalla de un saludo con el nombre indicado.

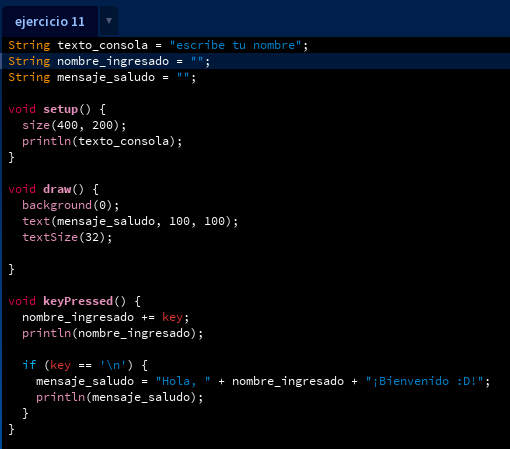
**Analisis:**

|  |
| --- |
| Datos de Entrada: nombre\_ingresado //cadena  Datos de Salidas: mensaje\_saludo //cadena de texto |
| Proceso:  **¿Quién debe realizar el proceso?:** El algoritmo o computadora  **¿Cuál es el proceso que resuelve?:** Ingresar un nombre que devolverá la creación de un saludo personalizado con el nombre proporcionado y su presentación en pantalla. |

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** Algoritmo |
| **Variables:**  nombre\_ingresado: string // almacena el nombre  mensaje\_saludo: string // almacenara una cadena de caracteres |
| **Nombre del Algoritmo:** saludar\_nombre |
| **Proceso del algoritmo:**   1. *Inicio* 2. *Leer* nombre\_ingresado 3. mensaje\_saludo *←* “Hola, ” + nombre\_ingresado + “¡Bienvenido!” 4. *mostrar* saludo 5. *fin* |

Captura de la codificación en lenguaje processing:



|  |  |  |
| --- | --- | --- |
| Mensaje que aparece al iniciar el programa. | Al escribir, el programa va escuchando cada letra presionada y finalmente al presionar la tecla enter, lanza el mensaje del saludo. | En la ventana del dibujo se muestra lo siguiente: |
|  | C:\Users\Luciano\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Captura de pantalla 2024-04-14 161830.png |  |

**Ejercicio 12:** Sera común resolver problemas utilizando variables. Calcule el perímetro y área de un rectángulo dada su base y su altura.

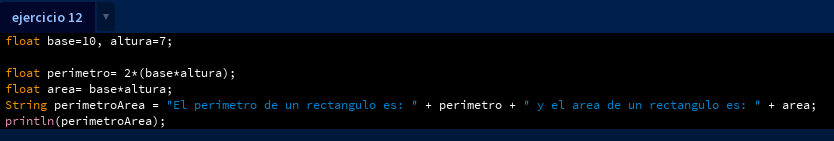
**Análisis:**

|  |
| --- |
| **Datos de Entrada:** base, altura // decimal  **Datos de Salida:** perímetro, área // almacena valores decimales  **Proceso:**  **¿Quién debe realizar el proceso?:** El usuario o calculadora.  **¿Cuál es el proceso que resuelve?:** Calcula el perímetro y el área de un rectángulo utilizando las fórmulas adecuadas. |

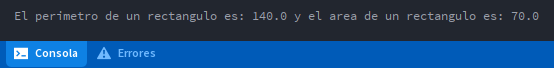
**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** persona |
| **Variables:**   * **base: float** // almacena un valor decimal * **área: float** // almacena un valor decimal * **perímetro: float** * **área: float** * **perimetroArea: float** // almacena un valor de calculos |
| **Nombre del Algoritmo:** perímetro\_area\_rectangulo |
| **Proceso del algoritmo:**   1. *inicio* 2. *Leer* base 3. *Leer* área 4. Perímetro *←* 2\*(base + altura) 5. Área *←* base \* altura 6. perimetroArea *←* “el perímetro de un rectandulo es: ” + perímetro + “ y la área de un rectángulo es: “ + área 7. *Mostrar* perimetroArea 8. *fin* |

Captura de la codificación en lenguaje processing:



Resultado de la consola:



**Ejercicio 13:** 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 conocido sus catetos.

**Analisis:**

|  |
| --- |
| **Datos de Entrada:** catetoA, catetoB  **Datos de Salida:** hipotenusa  **Proceso:**  **¿Quién debe realizar el proceso?:** La persona o calculadora.  **¿Cuál es el proceso que resuelve?:** Para calcular la longitud de la hipotenusa de un triángulo rectángulo se obtiene las longitudes de los catetos como entrada, se aplica la fórmula: |

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** persona |
| **Variables:**   * **catetoA: int** // almacena un valor decimal * **catetoB: int** // almacena un valor decimal * **hipotenusa: int** // almacena un valor de cálculos |
| **Nombre del Algoritmo:** perímetro\_area\_rectangulo |
| **Proceso del algoritmo:**   1. *inicio* 2. *Leer* catetoA 3. *Leer* catetoB 4. Hipotenusa ← (a^2 + b^2 ) ^(0.5) 5. *Mostrar* hipotenusa 6. *fin* |

|  |  |
| --- | --- |
| **Captura Processing** | **Resultado** |
|  | C:\Users\Luciano\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Captura de pantalla 2024-04-14 172001.png |

**Ejercicio 14:** 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.

**Analisis:**

|  |
| --- |
| **Datos de Entreda:** num1, num2  **Datos de Salida:** suma, resta, multiplicación, división  **Proceso:**  **¿Quién debe realizar el proceso?** La persona o calculadora  **¿Cuál es el proceso que resuelve?:** |

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** persona |
| **Variables:**   * **num1: int** // almacena un valor entero * **num2: int** // almacena un valor entero * **suma: int** // almacena un valor de una suma * **resta: int** // almacena un valor de una resta * **multiplicación: int** //almacena un valor de una multiplicación * **división: int** // almacena un valor de una división |
| **Nombre del Algoritmo:** calculadora\_basica |
| **Proceso del algoritmo:**   1. *inicio* 2. *Leer* num1 3. *Leer* num2 4. suma *←* num1 + num2 5. *mostrar ←* “el resultado de la suma es: “ + suma 6. resta← num1 – num2 7. *mostrar ←* “el resultado de la resta es: “ + resta 8. multiplicación ← num1 \* num2 9. *mostrar ←* “el resultado de la multiplicación es: “ + multiplicación 10. división ← num1 / num2 11. Si (num2 !=0) entonces 12. *mostrar ←* “el resultado de la división es: “ + división 13. *si\_no* 14. *mostrar* ← “la división por cero no está definida.” 15. *Fin* |

|  |
| --- |
| **Captura Processing** |
|  |

|  |
| --- |
| **Resultado** |
|  |

**Ejercicio 15:** 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.

**Analisis:**

|  |
| --- |
| **Datos de Entrada:** Temperatura en grados Fahrenheit  **Datos de Salida:** Temperatura en grados Celsius  **Proceso:**  **¿Quién debe realizar el proceso?** Puede ser realizado por un programa informático o una calculadora.  **¿Cuál es el proceso que resuelve?** El proceso consiste en convertir una temperatura dada en grados Fahrenheit a grados Celsius utilizando la formula de conversión correspondiente.  Esto implica restar 32 a la temperatura en Fahrenheit, multiplicar el resultado por 5/9 y obtener así la temperatura en Celsius. |

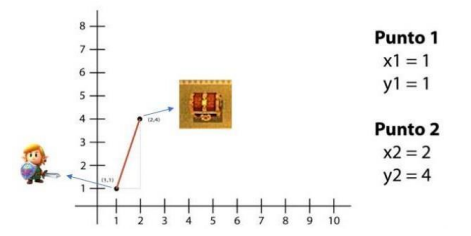
**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** persona |
| **Variables:**   * **temperaturaFahrenheit: flot** // almacena un valor decimal * **temperaturaCelsius: float** // almacena un valor decimal |

|  |
| --- |
| **Nombre del Algoritmo:** conversor\_de\_temperatura\_celsius |
| **Proceso del algoritmo:**   1. *inicio* 2. *Leer* temperaturaFahrenheit 3. temperaturaCelsius *←* (5.0 / 9.0) \* (temperaturaFahrenheit – 32) 4. *Mostrar* temperaturaCelsius 5. *fin* |

|  |
| --- |
| **Captura Processing** |
|  |
| **Resultado** |
|  |

**Ejercicio 16:** 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.

**Analisis:**

|  |
| --- |
| **Datos de Entrada:** coordenadas de Link, coordenadas del tesoro  **Datos de Salida:** distancia entre Link y tesoro  **Proceso:**  **¿Quién debe realizar el proceso?:** El programa informático o una calculadora que pueda realizar cálculos matemáticos.  **¿Cuál es el proceso que resuelve?:** Calculamos las diferencias en las coordenadas **x ; y** entre los dos puntos que nos darán los catetos formados por los puntos. |

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** persona |
| **Variables:**   * **x1: float** // almacena un valor decimal * **y1: float** // almacena un valor decimal * **x2: float** // almacena un valor decimal * **y2: float** // almacena un valor decimal * **coordenadaX: float** // almacena el resultado de un calculo * **coordenadaY: float** // almacena el resultado de un calculo * **distancia: float** // almacena el resultado de un calculo * **distanciaTesoro: float** // almacena un valor |
| **Nombre del Algoritmo:** distancia\_puntos |
| **Proceso del algoritmo:**   1. *inicio* 2. *Leer x1* 3. *Leer y1* 4. *Leer x2* 5. *Leer y2* 6. distanciaTesoro ← 50 7. coordenadaX ← x2 - x1 8. coordenadaY ← y2 – y1 9. distancia ← ((coordenadaX)^2 + (coordenadaY)^2)^2 10. *Mostrar* “la distancia es de: ” + distancia 11. **si** (distancia = distanciaTesoro) **entonces** 12. *Mostrar* “¡PowerUp activado!” 13. *fin\_si* 14. *fin* |

|  |
| --- |
| **Captura Processing** |
|  |
|  |

|  |  |
| --- | --- |
| Al iniciar el programa, lanza el primer mensaje sobre la distancia entre Link y el cofre. | Al mover con el mouse el personaje(Link) en la posición del cofre, lanzara ora mensaje sobre un powerUp |
|  |  |

|  |  |
| --- | --- |
| Posicion al iniciar el programa | Posición en la que lanzará el mensaje de powerUp |
|  |  |

**Ejercicio 17:** 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.

**Analisis:**

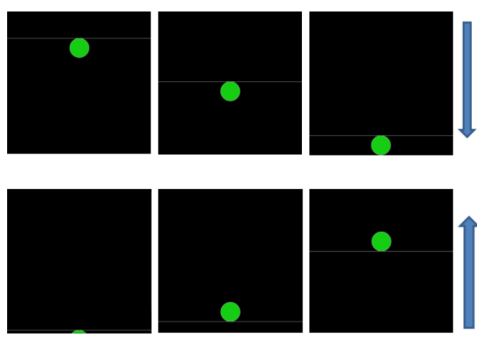
|  |
| --- |
| Datos de Entrada: Coeficientes de la ecuación cuadrática: a, b y c.  Datos de Salida: Raíces de la ecuación cuadrática.  Proceso:  **¿Quién debe realizar el proceso?: El** programa informático o una calculadora que pueda realizar cálculos matemáticos.  **¿Cuál es el proceso que resuelve?:** Calcular el discriminante de la ecuación cuadrática utilizando la fórmula |

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** persona |
| **Variables:**   * **a: float** // almacena un valor decimal * **b: float** // almacena un valor decimal * **c: float** // almacena un valor decimal * **discriminante: float** // almacena el valor de calculos |
| **Nombre del Algoritmo:** encontrar\_raices |
| **Proceso del algoritmo:**   1. *inicio* 2. *Leer* a 3. *Leer* b 4. *Leer* c 5. discriminante ← b^2 – 4\*a\*c 6. **si** (discriminante > 0) **entonces** 7. raiz1 ← (-b + (discriminante))^0.5 /(2\*a) 8. raiz2 ← (-b - (discriminante))^0.5 /(2\*a) 9. *mostrar* “las raíces son: ” + raiz1 + “ y ” + raiz2 10. **si\_no si** (discriminante == 0) **entonces** 11. raiz ← -b / (2\*a*)* 12. *Mostrar* “la raíz doble es: “ + raiz 13. **si\_no** 14. *Mostrar* “no hay raíces reales” 15. fin |

|  |  |
| --- | --- |
|  | Valores con a=1, b=0, c=-1  Valores con a=1, b=-2, c=1  Valores a=1, b=1, c=1 |

**Ejercicio 18:** 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:** línea, dir  **Datos de Salida:** bucle de la línea y circulo  **Proceso:**  **¿Quién debe realizar el proceso?: La computadora**  **¿Cuál es el proceso que resuelve?** |

**Diseño:**

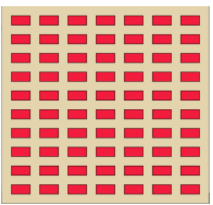
|  |
| --- |
| **Entidad que resuelve el problema:** lienzo |
| **Variables:**   * **línea: entero** // almacena un valor entero * **dir: entero** // almacena un valor entero |

|  |
| --- |
| **Nombre del Algoritmo:** linea\_ circulo\_en\_movimiento |
| **Proceso del algoritmo:**   1. *inicio* 2. *Leer* linea 3. *Leer* dir 4. anchoLienzo ← 400 5. altoLienzo ← 400 6. **para** i ← 0 hasta **alto** incremento 1 **hacer** 7. linea ← linea + dir 8. **fin\_para** 9. si ((linea >= anchoLienzo) O (linea <= 0)) **entonces** 10. dir ← dir \* (-1) 11. **fin\_si** 12. *mostrar* linea 13. dibujar linea en (dir, linea, altoLienzo, linea) 14. dibujar circulo en (altoLienzo/2, linea + 40, 80, 80) 15. *fin* |

|  |
| --- |
| **Captura Processing** |
|  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **C:\Users\Luciano\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Captura de pantalla 2024-04-15 011759.png** | **C:\Users\Luciano\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Captura de pantalla 2024-04-15 011707.png** | **C:\Users\Luciano\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Captura de pantalla 2024-04-15 011724.png** |

**Ejercicio 19:** 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:**

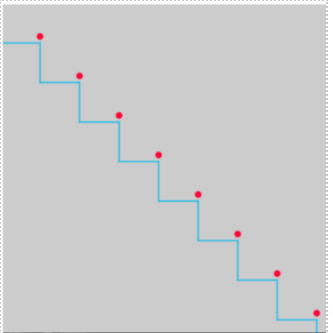
|  |
| --- |
| **Datos de Entrada:** Rectángulos dibujados en el lienzo según las especificaciones dadas.  **Datos de Salida:** Rectángulos dibujados en el lienzo según las especificaciones dadas.  **Proceso:**  **¿Quién debe realizar el proceso?:** El proceso puede ser realizado por un programa como processing.  **¿Cuál es el proceso que resuelve?:** dibujar una serie de rectángulos en un lienzo de tamaño específico, manteniendo una distancia específica entre ellos tanto horizontal como verticalmente, definiendo un bucle ***for*** para dibujar los rectángulos en el lienzo. |

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** lienzo |
| **Variables:**   * **coordenadasRect: float** // almacena un valor de coordenadas * **acho, alto, distanciaEntreRect: int** // almacena un valor entero * **anchoLienzo, altoLienzo: int** //almacenan valores enteros |
| **Nombre del Algoritmo:** rectángulos\_repetidos |
| **Proceso del algoritmo:**   1. *inicio* 2. anchoLienzo ← 440 3. altoLienzo ← 420 4. ancho ← 40 5. alto ← 20 6. distanciaRect ← 20 7. ***para*** x ← coordenadasRect.x **hasta** anchoLienzo **con paso** (ancho+distanciaEntreRect) 8. ***hacer*** 9. ***para*** y ← coordenadasRect.y **hasta** altoLienzo **con paso** (alto+distanciaEntreRect) 10. ***hacer*** 11. dibujar rectangulo en (x,y,ancho,alto) 12. *fin\_para* 13. *fin\_para* 14. *fin* |

|  |
| --- |
| **Captura Processing** |
|  |
| **Resultado** |
| **C:\Users\Luciano\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Captura de pantalla 2024-04-15 013829.png** |

**Ejercicio 20:** 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:**

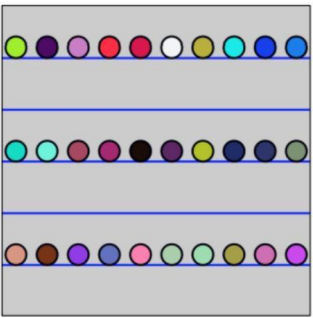
|  |
| --- |
| **Datos de Entrada:** puntoA, puntoB, puntoC, puntoD, distancia  **Datos de Salida:** Una imagen que consiste en escalones con puntos de color rojo en los bordes  **Proceso:**  **¿Quién debe realizar el proceso?:** El programa, mediante el código en Processing  **¿Cuál es el proceso que resuelve?:** El proceso consiste en iterar mediante while() para dibujar escalones y puntos rojos en los bordes |

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** programa |
| **Variables:**   * **puntoA, puntoB, puntoC, puntoD: int** // almacena un vector * **distancia: int** // almacena un valor entero |
| **Nombre del Algoritmo:** escalones\_puntos |
| **Proceso del algoritmo:**   1. *Inicio* 2. anchoLienzo ← 500 3. altoLienzo ← 500 4. distancia ← 60 5. ***mientras*** (puntoA.y sea menor o igual que anchoLienzo) **Hacer** 6. ***dibujar*** línea horizontal en (puntoA.x, puntoA.y, puntoB.x, puntoB.y) 7. ***dibujar*** línea vertical en (puntoB.x, puntoB.y, puntoC.x, puntoC.y) 8. dibujar circulo en (puntoD.x, puntoD.y) 9. puntoA.x ← puntoC.x 10. puntoA.y ← puntoC.y 11. ***fin\_mientras*** 12. *Fin* |

|  |
| --- |
| **Captura Processing** |
|  |
| **Resultado** |
|  |

**Ejercicio 21:** 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:** números de líneas y círculos

**Datos de Salida:** círculos con colores randoms sobre líneas con un color con distanciamiento por medio

**Proceso:**

**¿Quien debe realizar el proceso?:** El lienzo se divide verticalmente en franjas de igual medida, donde se dibujan líneas en todas ellas. En cada línea de forma alternada, se dibujan círculos con colores aleatorios, los cuales están espaciados uniformemente a lo largo de la línea.

**¿Cual es el proceso que resuelve?:** el programa en este caso processing

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** processing |
| **Variables:**   * **distanciaCirculo: int** // almacena un valor entero * **lineaX, lineaY, circuloX, circuloY, distanciaCirculo: int** // almacena un valor entero * **anchoLienzo, altoLienzo, int** // almacenan valores enteros |
| **Nombre del Algoritmo:** rectángulos\_repetidos |
| **Proceso del algoritmo:**   1. *inicio* 2. anchoLienzo ← 600 3. altoLienzo ← 600 4. lineaX ← 0 5. lineaY ← 100 6. distanciaCirculo ← 30; 7. circuloY ← 75 8. ***hacer*** 9. circuloX ← distanciaCirculo 10. ***hacer*** 11. dibujar linea en (lineaX, lineaY, anchoLienzo, lineaY) 12. dibujar circulo en circuloX, circuloY, 50, 50) 13. circuloX ← circuloX + distanciaCirculo\*2 14. ***fin\_hacer*** 15. **mientras**(circuloX sea menor que ancholienzo) 16. LineaY ← lineaY + 100; 17. circuloY ← circuloY + 200; 18. ***fin\_hacer*** 19. **mientras** (lineaY sea menor que altoLienzo) 20. *fin* |

|  |
| --- |
| **Captura Processing** |
|  |
| **Resultado** |
|  |