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

Trabajo Práctico

N°1: Algoritmos y

Estructuras de Control

Tactaca, Cecilia Yazmin – LU TUV000764

*Profesores:*

*Mg. Ing. Ariel Alejandro Vega*

*Año 2024*

**FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS**

Índice

[Sección Expresiones aritméticas y lógicas 4](#_Toc164206409)

[Ejercicio 1: 4](#_Toc164206410)

[Desarrollo del ejercicio 1: 4](#_Toc164206411)

[Ejercicio 2: 4](#_Toc164206412)

[Desarrollo del ejercicio 2: 4](#_Toc164206413)

[Ejercicio 4: 5](#_Toc164206414)

[Desarrollo del ejercicio 4: 5](#_Toc164206415)

[Ejercicio 5: 7](#_Toc164206416)

[Desarrollo del ejercicio 5: 7](#_Toc164206417)

[Ejercicio 6: 8](#_Toc164206418)

[Desarrollo del ejercicio 6: 8](#_Toc164206419)

[Ejercicio 7: 9](#_Toc164206420)

[Desarrollo del ejercicio 7: 9](#_Toc164206421)

[Ejercicio 8: 9](#_Toc164206422)

[Desarrollo del ejercicio 8: 9](#_Toc164206423)

[Ejercicio 9: 10](#_Toc164206424)

[Desarrollo del ejercicio 9: 10](#_Toc164206425)

[Ejercicio 10: 10](#_Toc164206426)

[Desarrollo del ejercicio 10: 10](#_Toc164206427)

[Ejercicio 11: 11](#_Toc164206428)

[Desarrollo del ejercicio 11: 11](#_Toc164206429)

[Sección Análisis – Diseño y Codificación de algoritmos – Aplicación de estructuras de control 11](#_Toc164206430)

[Ejercicio 12: 11](#_Toc164206431)

[Desarrollo del ejercicio 12: 11](#_Toc164206432)

[Ejercicio 13: 12](#_Toc164206433)

[Desarrollo del ejercicio 13: 12](#_Toc164206434)

[Ejercicio 14: 14](#_Toc164206435)

[Desarrollo del ejercicio 14: 14](#_Toc164206436)

[Ejercicio 15: 15](#_Toc164206437)

[Desarrollo del ejercicio 15: 15](#_Toc164206438)

[Ejercicio 16: 17](#_Toc164206439)

[Desarrollo del ejercicio 16: 17](#_Toc164206440)

[Ejercicio 17: 18](#_Toc164206441)

[Desarrollo del ejercicio 17: 18](#_Toc164206442)

[Ejercicio 18: 20](#_Toc164206443)

[Desarrollo del ejercicio 18: 20](#_Toc164206444)

[Ejercicio 19: 22](#_Toc164206445)

[Desarrollo del ejercicio 19: 23](#_Toc164206446)

[Ejercicio 20: 24](#_Toc164206447)

[Desarrollo del punto 20 24](#_Toc164206448)

[Ejercicio 21: 26](#_Toc164206449)

[Desarrollo del ejercicio 21: 27](#_Toc164206450)

[Ejercicio 22: 28](#_Toc164206451)

[Desarrollo del ejercicio 22: 29](#_Toc164206452)

[Fuentes bibliográficas 30](#_Toc164206453)

# 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

### Desarrollo del ejercicio 1:

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

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

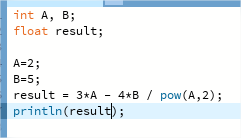
(3\*2)-(4\*5/(2^2))

6-(4\*5/4)

6-5

1

Captura de Processing



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

### Desarrollo del ejercicio 2:

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

(((4/2)\*3)/6)+((((6/2)/1)/25) / 4) \* 2)

((2\*3)/6)+(((3/1)/25)/4)\*2)

(6/6)+((3/25)/4)\*2)

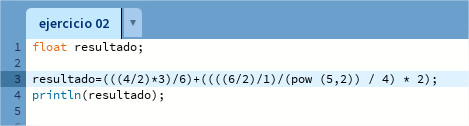
1+((0.12/4)\*2)

1+(0.03\*2)

1+0.06

1.06

Captura de Processing



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)

### Desarrollo del ejercicio 4:

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

b ^ 2 – ((4 \* a) \* c)

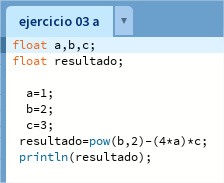
2 ^ 2 – ((4 \* 1 )\* 3)

4-((4\*1)\*3)

4-(4\*3)

4-12

-8



1. x=2

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

(3 \* (2 ^ 4)) – (5 \*( 2 ^ 3 ))+ (2 \* 12) – 17

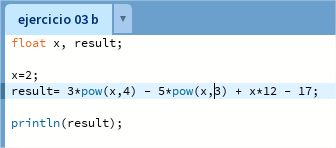
(3 \* 16) - (5\*8) + 24 -17

48 – 40 + 24 – 17

8 + 24 – 17

32 – 17

15



1. b=7, c=4 y d=9

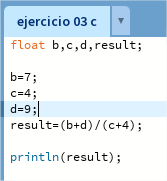
(b + d) / (c + 4)

(7+9)/(4+4)

16/(4+4)

16/8

2



1. x=3 e y=4

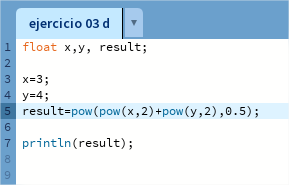
(x ^ 2 + y ^ 2) ^ (1 / 2)

(3 ^ 2 + 4 ^ 2) ^ (1 / 2)

(9 + 16) ^ (1 / 2)

(25) ^ (1 / 2)

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:

1. B \* A – B ^ 2 / 4 \* C
2. (A \* B) / 3 ^ 2
3. (((B + C) / 2 \* A + 10) \* 3 \* B) – 6

### Desarrollo del ejercicio 5:

1. 5 \* 4 – 5 ^ 2 / 4 \* 1

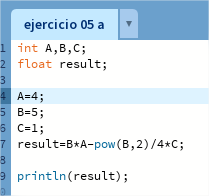
5\*4 – 25/4\*1

20 – 25/4\*1

20 – 25/4

20 – 6.25

13.75



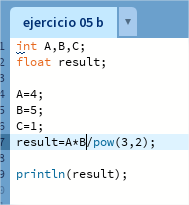
1. (A \* B) / 3 ^ 2

(4\*5) / 3 ^ 2

(4\*5) / 9

20/9

2.2222



1. (((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

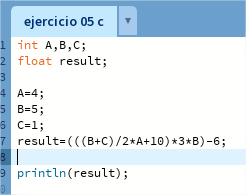
((12+10)\*3\*5)-6

(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

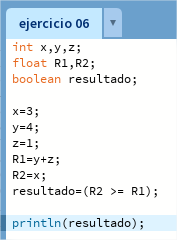
R2 = x >= R1

### Desarrollo del ejercicio 6:

R1=4+1=5

R2=3

R2>=R1 falso



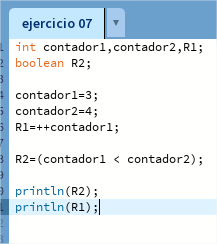
Ejercicio 7: Para contador1=3, contador2=4, evaluar el resultado de

R1 = ++contador1

R2 = contador1 < contador2

### Desarrollo del ejercicio 7:

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 del ejercicio 8:

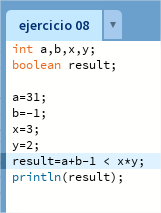
31+(-1)-1 < 3\*2

31+(-1)-1 < 6

30-1 < 6

29 < 6

Falso



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

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

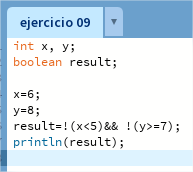
### Desarrollo del ejercicio 9:

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

¡falso && ¡verdadero

verdadero && falso

falso



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

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

### Desarrollo del ejercicio 10:

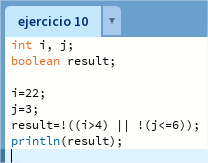
!((22>4) || !(3<=6))

!((verdadero) || !(verdadero))

!(verdadero || falso)

!(verdadero)

Falso



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

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

### Desarrollo del ejercicio 11:

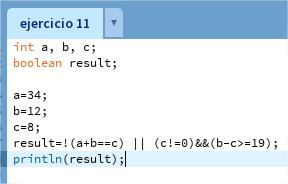
!(34+12==8) || (8!=0)&&(12-8>=19)

!(46==8) || (8!=0)&&(20>=19)

!(falso) || (verdadero)&&(verdadero)

verdadero || (verdadero)&&(verdadero)

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 del ejercicio 12:

#### Fase de análisis

* Especificación del problema: Mostrar un saludo con un nombre ingresado por teclado.
* Análisis:

Datos de entrada:

Nombre.

Datos de salida:

Saludo con nombre.

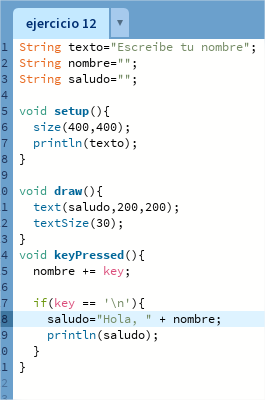
Proceso:

¿Quién realiza el proceso? Processing

¿Cuál es el proceso que se realiza? Se pide que se ingrese un nombre, una vez ingresado el nombre por el teclado se emite en pantalla un saludo con el nombre.

#### Fase de diseño

|  |
| --- |
| Nombre de la entidad que resuelve el problema: Processing |
| Variables:  nombre: string // almacenan el nombre ingresado.  saludo: string // almacenan el saludo y nombre. |
| Nombre del algoritmo: hola\_nombre  Proceso del algoritmo   1. *inicio* 2. *Leer* nombre 3. *mensaje\_*saludo *← “Hola, ”*  + nombre 4. *Mostrar* saludo 5. *fin* |



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

### Desarrollo del ejercicio 13:

#### Fase de análisis

* Especificación del problema: Calculo de perímetro y área de un rectángulo.
* Análisis:

Datos de entrada:

Base del rectángulo.

Altura de rectángulo.

Datos de salida:

Perímetro del rectángulo.

Área del rectángulo.

Proceso:

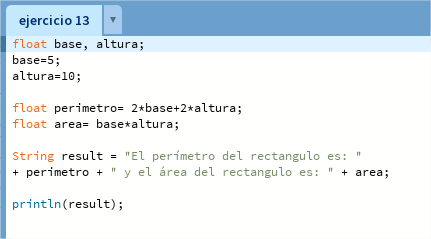
¿Quién realiza el proceso? Formula de perímetro y área de figuras geométricas.

¿Cuál es el proceso que se realiza? Se ingresan los datos de base y altura del rectángulo, luego se calcula el perímetro con la siguiente formula:

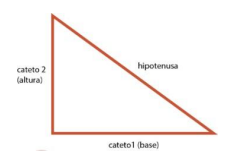
Para el calculo del área se usa la siguiente formula:

#### Fase de diseño

|  |
| --- |
| Nombre de la entidad que resuelve el problema: Fórmula matemática. |
| Variables:  base, altura: real // almacenan las medidas del rectángulo.  Perímetro, área: real // almacenan los resultados del perímetro y área del rectángulo. |
| Nombre del algoritmo: perímetro\_area  Proceso del algoritmo   1. *inicio* 2. *Leer* base 3. *Leer* altura 4. perimetro ← 2\*base + 2\*altura 5. area ← base \* altura 6. *Mostrar* “El perímetro del rectángulo es: “ + perímetro + “ y el área del rectángulo es: “ + area 7. *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 del ejercicio 14:

#### Fase de análisis

* Especificación del problema: Calcular hipotenusa de un triángulo rectángulo.
* Análisis:

Datos de entrada:

Cateto 1

Cateto 2

Datos de salida:

Hipotenusa

Proceso:

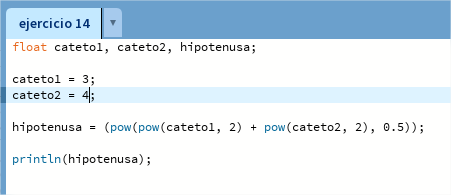
¿Quién realiza el proceso? Teorema de Pitágoras

¿Cuál es el proceso que se realiza? Al ser un triangulo rectángulo, uno de sus ángulos es de 90º, por ello se puede aplicar el teorema de Pitágoras para calcular sus catetos e hipotenusa.

Se ingresan las medidas de los catetos y mediante el teorema de Pitágoras se calcula la hipotenusa del triángulo rectángulo.

#### Fase de diseño

|  |
| --- |
| Nombre de la entidad que resuelve el problema: Teorema de Pitágoras |
| Variables:  cateto1, cateto2: real // almacenan las medidas de los catetos del triángulo.  hipotenusa: real // almacenan la hipotenusa calculada. |
| Nombre del algoritmo: calculo\_hipotenusa  Proceso del algoritmo   1. *Inicio* 2. *Leer* cateto1 3. *Leer* cateto2 4. hipotenusa *←* (a^2 + b^2) ^ (0.5) 5. *Mostrar* hipotenusa 6. *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 del ejercicio 15:

#### Fase de análisis

* Especificación del problema: Crear una calculadora que sume, reste, multiplique y divida dos números.
* Análisis:

Datos de entrada:

Número 1.

Número 2.

Datos de salida:

Resultado de la operación requerida.

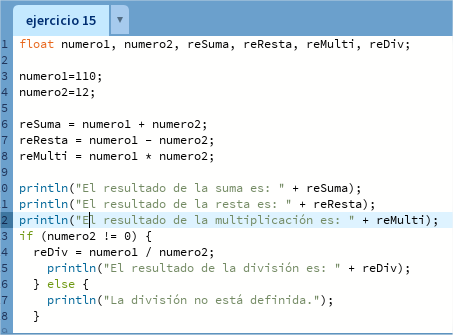
Proceso:

¿Quién realiza el proceso? Calculadora.

¿Cuál es el proceso que se realiza? Se ingresan dos números y sea suman, restan, multiplican y dividen entre ellos, y se muestra el resultado.

#### Fase de diseño

|  |
| --- |
| Nombre de la entidad que resuelve el problema: Calculadora. |
| Variables:  numero1, numero2: reales // almacenan los numero ingresados.  reSuma, reResta, reMulti, reDiv : real // almacenan el resultado de las operaciones |
| Nombre del algoritmo: suma  Proceso del algoritmo   1. *inicio* 2. *Leer* numero1 3. *Leer* numero2 4. reSuma ← numero1 + numero2 5. *Mostrar ← “El resultado de la suma es: “* + reSuma 6. *Fin* |
| Nombre del algoritmo: resta  Proceso del algoritmo   1. *inicio* 2. *Leer* numero1 3. *Leer* numero2 4. reResta ← numero1 – numero2 5. *Mostrar ← “El resultado de la resta es: “* + reResta 6. *Fin* |
| Nombre del algoritmo: multiplicacion  Proceso del algoritmo   1. *inicio* 2. *Leer* numero1 3. *Leer* numero2 4. reMulti ← numero1 \* numero2 5. *Mostrar ← “El resultado de la multiplicación es: “* + reMulti 6. *Fin* |
| Nombre del algoritmo: division  Proceso del algoritmo   1. *inicio* 2. *Leer* numero1 3. *Leer* numero2 4. *Si* (numero2 !=0) *entonces* 5. reDiv ← numero1 / numero2 6. *Mostrar ← “El resultado de la división es: “* + reDiv 7. *Si\_no* 8. *Mostrar ← “La división no está definida.”* 9. *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 del ejercicio 16:

#### Fase de análisis

* Especificación del problema: Convertir temperatura en grados Fahrenheit a grados Celsius.
* Análisis:

Datos de entrada:

Temperatura en grados Fahrenheit.

Datos de salida:

Temperatura en grados Celsius.

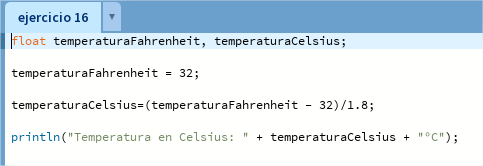
Proceso:

¿Quién realiza el proceso? Ecuación de conversión de unidades de temperatura.

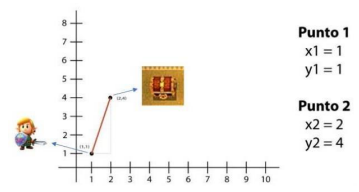
¿Cuál es el proceso que se realiza? Se ingresa la temperatura en grados Fahrenheit y mediante la ecuación de conversión de unidades se calcula la temperatura en grados Celsius.

#### Fase de diseño

|  |
| --- |
| Nombre de la entidad que resuelve el problema: Ecuación de conversión de unidades. |
| Variables:  temperaturaFahrenheit // almacenan la temperatura en grados Fahrenheit.  temperaturaCelsius // almacenan la temperatura en grados Celsius. |
| Nombre del algoritmo: temperatura  Proceso del algoritmo   1. *inicio* 2. *Leer* temperaturaFahrenheit 3. temperaturaCelsius ← (temperaturaFahrenheit – 32)/1.8 4. *Mostrar* “Temperatura en Celsius: “ + temperaturaCelsius + “ºC” 5. 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.

### Desarrollo del ejercicio 17:

#### Fase de análisis

* Especificación del problema: Calcular la distancia entre Link y el tesoro.
* Análisis:

Datos de entrada:

Posición de Link, dada por el movimiento del mouse.

Posición del tesoro.

Datos de salida:

Distancia entre link y el tesoro.

Mensaje de “Power up DESACTIVADO”.

Proceso:

¿Quién realiza el proceso? Processing

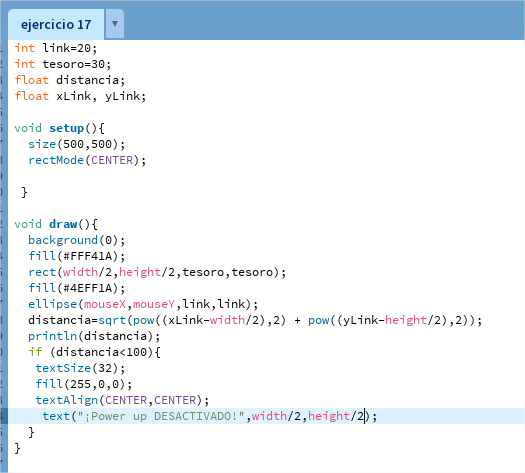
¿Cuál es el proceso que se realiza? Se proyectan unos ejes cartesianos en el plano donde están Link y el tesoro. Con las posiciones de Link y el tesoro se puede imaginar un triangulo rectángulo, por lo tanto, primero se calculan los catetos de este triangulo:

y luego con el teorema de Pitágoras se calcula la distancia entre ellos.

La posición del tesoro estará fija en centro del lienzo, mientras que Link se moverá con el mouse.

#### Fase de diseño

|  |
| --- |
| Nombre de la entidad que resuelve el problema: Processing |
| Variables:  anchoLienzo y altoLienzo // almacenan los datos del comando size.  link: entero // almacena el diámetro del círculo que representa a Link.  tesoro: entero // almacena el lado del cuadrado que representa el tesoro.  xLink: real // almacena la posición en x de Link.  yLink: real // almacena la posición en y de Link.  distancia: real // almacena la distancia entre Link y el tesoro. |
| Nombre del algoritmo: Link\_tesoro  Proceso del algoritmo   1. Inicio 2. link <- 10 3. tesoro <- 30 4. anchoLienzo <- 500 5. altoLienzo <- 500 6. xTesoro <- anchoLienzo/2 7. yTesoro <- altoLienzo/2 8. Dibujar rectángulo (xTesoro,yTesoro,tesoro,tesoro) 9. Dibujar elipse(mouseX,mouseY,link,link) 10. xLink <- mouseX 11. yLink <- mouseY 12. Distancia <- sqrt(pow((xLink-xTesoro),2) + pow((yLink-yTesoro),2)) 13. Si (distancia<100) Hacer 14. Mostrar “Power up DESACTIVADO” 15. 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 del ejercicio 18:

#### Fase de análisis

* Especificación del problema: Calcular las raíces de una ecuación cuadrática.
* Análisis:

Datos de entrada:

Coeficientes de la ecuación cuadrática.

Datos de salida:

Raíces de la ecuación.

Proceso:

¿Quién realiza el proceso? Ecuación de determinantes y calculo de raíces.

¿Cuál es el proceso que se realiza? Se ingresa los coeficientes de la ecuación cuadrática de la cual se quieren conocer las raíces. Primero se determina si hay raíces reales, imaginarias, o si son dos distintas o existe una sola raíz, para ello usamos el discriminante:

Si es mayor a cero la ecuación tiene dos raíces reales distintas.

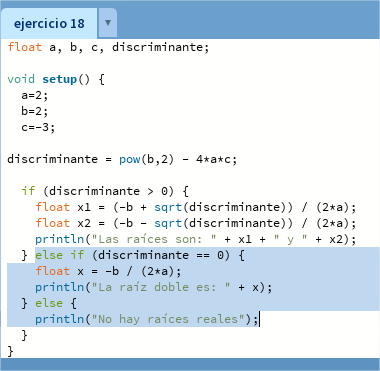
Si es igual a cero la ecuación tendrá dos raíces reales iguales.

Si es menor a cero no tendrá raíces reales.

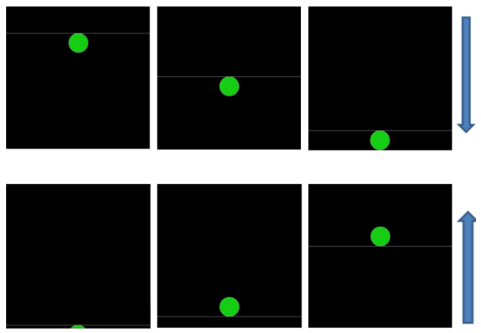
Luego de determinar el tipo de raíces, se calculan con la ecuación:

#### Fase de diseño

|  |
| --- |
| Nombre de la entidad que resuelve el problema: Ecuación de raíces. |
| Variables:  a,b,c: real // almacenan los coeficientes de la ecuación cuadrática.  discriminante: real // almacenan el discriminante.  x1, x2: reales // almacenan las raíces de la ecuación. |
| Nombre del algoritmo: raíces\_cuadraticas  Proceso del algoritmo   1. inicio 2. a <-2 3. a <-2 4. c <-3 5. discriminante <- b ^ 2 – 4 \* a \* c 6. *Según* discriminante *Hacer* 7. discriminante>0: 8. x1 <- (-b + sqrt(discriminante)) / (2\*a) 9. x2 <- (-b - sqrt(discriminante)) / (2\*a) 10. *Mostrar* "Las raíces son: " + x1 + " y " + x2 11. discriminante == 0: 12. x <- -b / (2\*a); 13. *Mostrar* "La raíz doble es: " + x; 14. discriminante<0: 15. *Mostrar* "No hay raíces reales" 16. *Fin Según* 17. *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 del ejercicio 19:

#### Fase de análisis

* Especificación del problema: Dibujar una línea y un círculo que se muevan verticalmente por el lienzo.
* Análisis:

Datos de entrada:

Tamaño del lienzo: alto y ancho en pixeles.

Tamaño del círculo: ancho y alto en pixeles del círculo.

Posición del circulo con respecto a la línea.

Datos de salida:

Lienzo con fondo negro, en él se observará una línea y un circulo que se mueven hacia abajo y arriba continuamente.

Proceso:

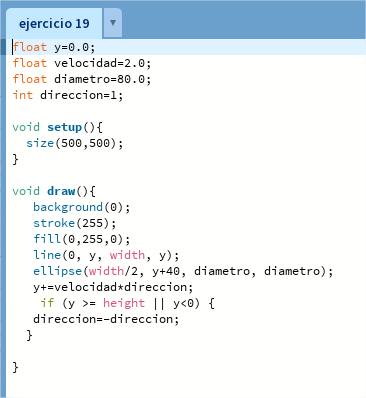
¿Quién realiza el proceso? Processing

¿Cuál es el proceso que se realiza? Se ingresan las medidas del lienzo, también se define el diámetro del círculo, y una variable y que almacenara las coordenas en y que iran cambiando al ser modificadas por la variable velocidad.

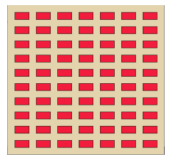
Se dibujan la línea y el circulo, con la variable “y” en el sitio de coordenada y para que vaya variando y provoque el movimiento. Cuando “y” llegue al valor igual al alto del lienzo cambia de dirección, y vuelve a cambiar de dirección cuando llegue al valor cero.

#### Fase de diseño

|  |
| --- |
| Nombre de la entidad que resuelve el problema: Processing |
| Variables:  anchoLienzo y altoLienzo // almacenan el tamaño del lienzo.  diámetro: entero // almacena el diámetro del círculo.  velocidad: real // almacena un valor que modifica y cada que se ejecute void draw.  dirección: real // almacenará el valor que hará cambiar de dirección el movimiento.  y: real // almacena la coordenada en y. |
| Nombre del algoritmo: circulo\_movil  Proceso del algoritmo   1. inicio 2. y <- 0.0 3. diámetro <- 80.0 4. velocidad <- 2.0 5. direccion <- 1 6. anchoLienzo <-300 7. altoLienzo<-300 8. dibujar línea (0, y, anchoLienzo, y); 9. dibujar elipse (anchoLienzo/2, y+40, diametro, diametro) 10. y <- y+velocidad\*direccion 11. Si (y >= altoLienzo || y<0) 12. dirección <- -direccion 13. Fin si 14. 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 del punto 20

#### Fase de análisis

* Especificación del problema: Dibujar rectángulos rojos en un lienzo amarillo.
* Análisis:

Datos de entrada:

Tamaño del lienzo: alto y ancho en pixeles.

Tamaño de los rectángulos: ancho y alto en pixeles, de cada rectángulo.

Distancia entre los rectángulos.

Distancia entre los rectángulos y los límites del lienzo.

Datos de salida:

Lienzo amarillo con rectángulos rojos distribuidos como en la figura 1.

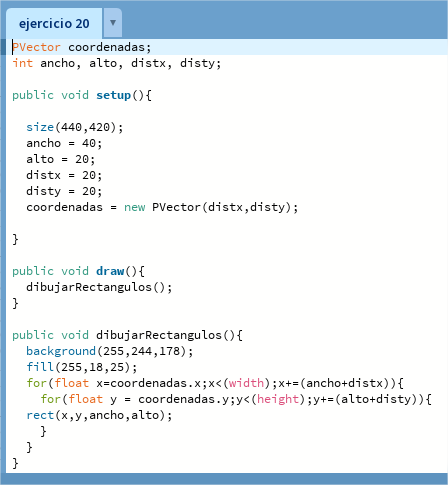
Proceso:

¿Quién realiza el proceso? Processing

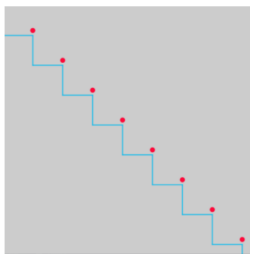
¿Cuál es el proceso que se realiza? Se ingresan las medidas del lienzo y las medidas de los rectángulos que desean dibujarse, además se especifica su color. Posteriormente se calcula en que coordenadas deben dibujarse cada rectángulo con el uso de la estructura de control repetitiva for.

#### Fase de diseño

|  |
| --- |
| Nombre de la entidad que resuelve el problema: Processing |
| Variables:  alto, ancho: entero // almacenan las medidas del rectángulo.  anchoLienzo y altoLienzo // almacenan en el comando size.  distx, disty: entero // almacenan las distacias que hay entre cada rectángulo y los límites del lienzo.  coordenadas: vector // almacena las coordenadas donde se dibuja el rectángulo. |
| Nombre del algoritmo: ejercicio\_20  Proceso del algoritmo   1. inicio 2. anchoLienzo <-440 3. altoLienzo<-420 4. ancho<-40 5. alto<-20 6. distx<-20 7. disty<-20 8. coordenadas<-(distx, disty) 9. Para (x= cooredenadas.x) hasta (anchoLienzo) con paso (ancho+distx) Hacer 10. Para (y =coordenadas.y) hasta (altoLienzo) con paso (alto+disty) Hacer 11. Dibujar rectángulo en (x,y,ancho,alto) 12. Fin-para 13. Fin-para 14. 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 del ejercicio 21:

#### Fase de análisis

* Especificación del problema: Dibujar escalones sobre un lienzo y colocar un punto rojo sobre cada escalón, como en la figura anterior.
* Análisis:

Datos de entrada:

PuntoA, PuntoB, PuntoC, puntos para definir las líneas.

PuntoD, punto para definir el punto rojo.

Distancia de la línea.

Datos de salida:

Imagen del lienzo con escalones y puntos rojos

Proceso:

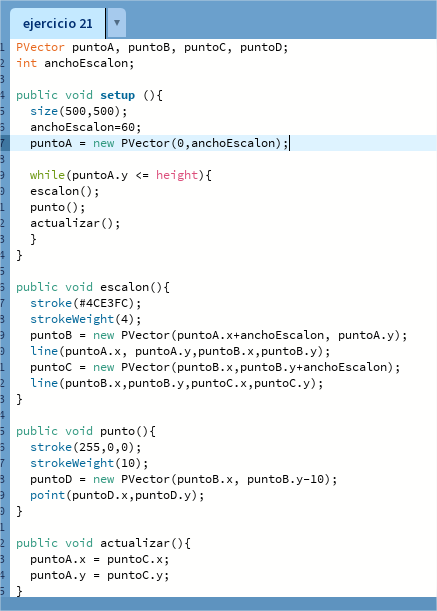
¿Quién realiza el proceso? Processing

¿Cuál es el proceso que se realiza?

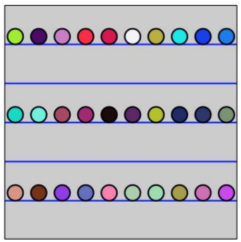
* Se dibuja una línea horizontal entre el puntoA y el puntoB.
* Luego la línea vertical entre el puntoB y puntoC.
* Después se dibuja el punto rojo en la posición x del puntoB y en la posición y corrida 10 unidades de la posición y del puntoB. Se repite el proceso para llenar el lienzo.
* Se actualiza las coordenadas del puntoA al puntoC.
* Se repite para que dibuje el resto de escalones.

#### Fase de diseño

|  |
| --- |
| Nombre de la entidad que resuelve el problema: Escalón. |
| Variables:  PuntoA, PuntoB, PuntoC, punto D // coordenadas cartesianas en 2D.  distLinea: entero// tamaño escalon. |
| Nombre del algoritmo: dibujar\_escalon\_punto  Proceso del algoritmo   1. *Inicio* 2. anchoLienzo <-500 3. altoLienzo <-500 4. anchoEscalon <- 60 5. puntoA <- (0, anchoEscalon) 6. Mientras (puntoA.y <= altoLienzo) Hacer 7. escalon () 8. punto () 9. actualizar () 10. Fin Mientras 11. Fin |
| Nombre del algoritmo: escalon  Proceso del algoritmo   1. *Inicio* 2. puntoB <- (puntoA.x+anchoEscalon, puntoA.y) 3. Dibujar línea (puntoA.x, puntoA.y,puntoB.x,puntoB.y) 4. puntoC <- (puntoB.x,puntoB.y+anchoEscalon) 5. Dibujar línea (puntoB.x,puntoB.y,puntoC.x,puntoC.y) 6. Fin |
| Nombre del algoritmo: punto  Proceso del algoritmo   1. *Inicio* 2. puntoD <- (puntoB.x, puntoB.y-10) 3. Dibujar punto (puntoD.x,puntoD.y) 4. Fin |
| Nombre del algoritmo: actualizar  Proceso del algoritmo   1. *Inicio* 2. puntoA.x <- puntoC.x; 3. puntoA.y <- puntoC.y 4. 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 del ejercicio 22:

#### Fase de análisis

* Especificación del problema: Dibujar círculos en un lienzo de acuerdo a la distribución mostrada en la imagen.
* Análisis:

Datos de entrada:

Tamaño del lienzo: alto y ancho en pixeles.

Color de la línea.

Distancia entre las divisiones.

Círculos de colores aleatorios

Datos de salida:

Lienzo dividido por líneas y círculos en algunas de ellas, como se muestra en la imagen.

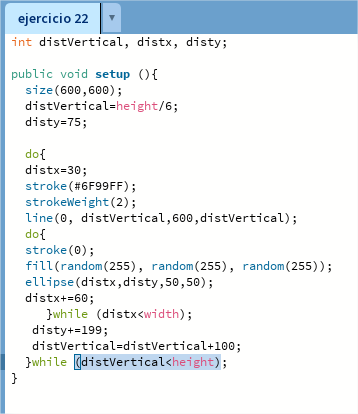
Proceso:

¿Quién realiza el proceso? Processing

¿Cuál es el proceso que se realiza? Se dibuja el lienzo y se calculan las coordenadas de la primera línea. Con la estructura do-while se irán dibujando el resto de líneas, en cada repetición se cambia la coordenada y de las líneas, dentro de esta estructura también se cambian las coordenadas y de los círculos para que se dibujen de manera alternada sobre las líneas. Se coloca otra estructura do-while dentro de la anterior, en esta se dibujará un círculo, y se ira cambiando la coordenada x del circulo para que dibuje los demás, también dentro de esta se especificara que el color de los círculos será random.

Fase de diseño

|  |
| --- |
| Nombre de la entidad que resuelve el problema: Processing |
| Variables:  anchoLienzo y altoLienzo // almacenan en el comando size.  distVertical: entero // almacena la distancia y a la que se dibujaran las líneas.  distx, disty: entero // coordenadas de los circulos |
| Nombre del algoritmo: lineas\_circulos  Proceso del algoritmo   1. *Inicio* 2. distVertical <- altoLienzo/6 3. disty <- 75 4. *Hacer* 5. distx <- 30 6. *Dibujar* línea (0, distVertical, 600, distVertical) 7. *Hacer* 8. *Dibujar* elipse(distx,disty,50,50) 9. distx <- ditsx+60 10. *Mientras* (distx<anchoLienzo) 11. disty <- disty+199 12. distVertical <- distVertical+100 13. *Mientras* (distVertical<altoLienzo) 14. *Fin* |



# Fuentes bibliográficas

* Mg. Ing. Vega, A. DISEÑO DE ALGORITMOS: Expresiones aritméticas. Facultad de Ingeniería. Universidad Nacional de Jujuy.
* Mg. Ing. Vega, A. DISEÑO DE ALGORITMOS: Bifurcaciones. Expresiones Logicas. Estructuras selectivas. Facultad de Ingeniería. Universidad Nacional de Jujuy.
* Mg. Ing. Vega, A. DISEÑO DE ALGORITMOS: Estructuras Selectivas. Facultad de Ingeniería. Universidad Nacional de Jujuy.
* Mg. Ing. Vega, A. Programando con Processing. Facultad de Ingeniería. Universidad Nacional de Jujuy.
* <https://www.youtube.com/watch?v=EAclc8NcnOo&t=1s>
* <https://www.youtube.com/watch?v=LUfoF-tG8xw>
* <https://www.youtube.com/watch?v=Nr3NdAxjqsE>
* <https://www.youtube.com/watch?v=_YblzDgoAus>
* <https://www.youtube.com/watch?v=bMJhygO43cA>
* <https://www.youtube.com/watch?v=HF1P1LHIS1s>
* <https://www.youtube.com/watch?v=dysxlptjAx0>
* <https://www.youtube.com/watch?v=4RQV0gMl8y8>
* <https://www.youtube.com/watch?v=kApd8u4l0tU>
* <https://www.youtube.com/watch?v=r-Olf82ViNE>