# TRABAJO PRÁCTICO N°1

# Fundamentos de la programación orientada a objetos

Nombre: Riveros Luciano Martín

DNI: 44515045

LU: TUV000729

## 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

Resolución necesaria en Word:

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

6-(4*B/4)

6-5

1

Captura de Processing

```
1  int A=2,B=5;
2
3  float resultado  = 3* A - 4 * B / pow(A,2);
4
5  println(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

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.** 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)

$$b^2 - 4.a.c$$

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

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

R1 = y+z

R2 = x >= R1

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

R1 = ++contador1

R2 = contador1 < contador2

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

a+b-1 < x*y

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

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

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

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

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

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

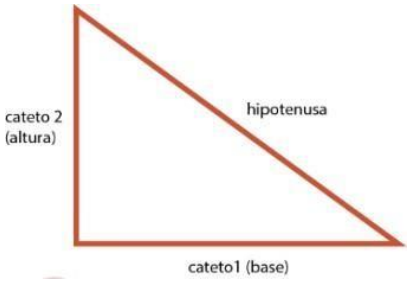
## Sección Análisis - Diseño y Codificación de algoritmos - Aplicación de estructuras de control

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.

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

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

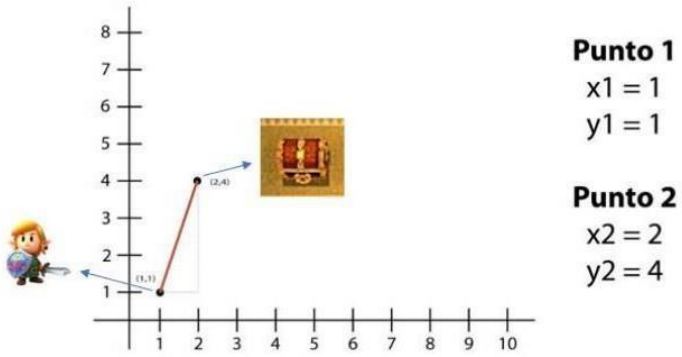



Ejercicio 15: Si viste algo de los apuntes y vídeos, esto debería ser muy fácil de resolver. Dados dos números permita calcular la suma, resta, multiplicación y división de estos. Considere que cada una de estas operaciones es un algoritmo cuando realice el diseño. Obviamente muestre los resultados.

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**

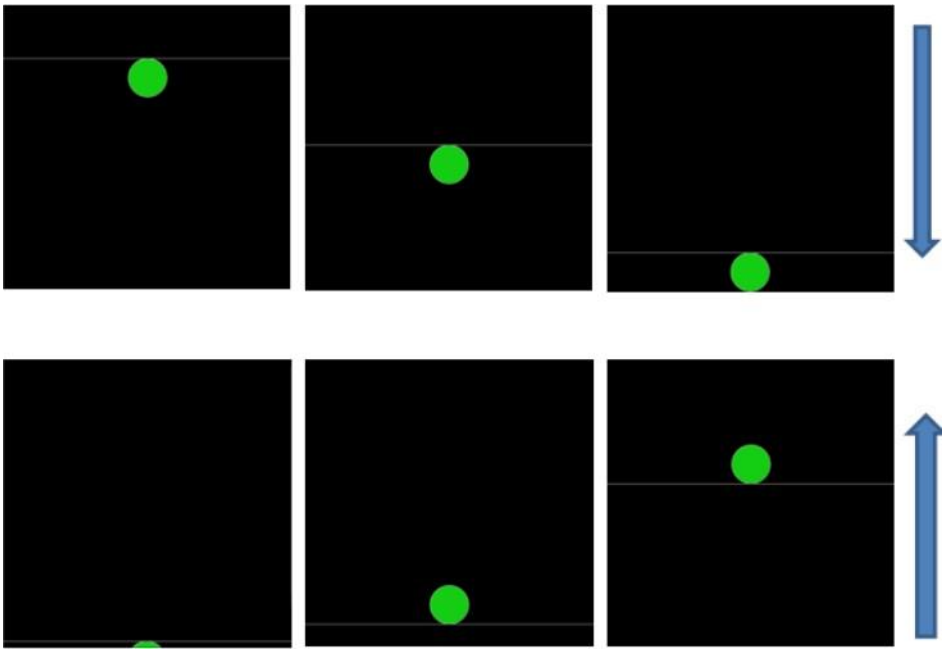
Ejercicio 17: Si queremos representar personajes o power ups (premios) en la pantalla debemos primero ubicarlos en alguna posición dentro de la pantalla. Imagine que está en un juego donde un power up desaparece porque el personaje se acerca a una distancia de x unidades, sin importar por donde se acerque. 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



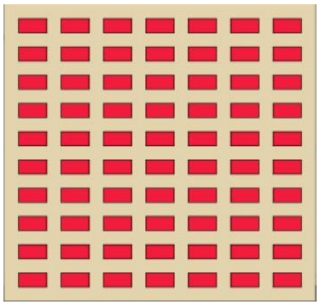
Para esto debe calcular el tamaño de los catetos y luego aplicar el teorema. Halle la distancia entre ambos objetos. Cuando programe, represente a lLnk con un Circulo, y al tesoro con un cuadrado. Además, mueva a Link mediante el mouse.

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.

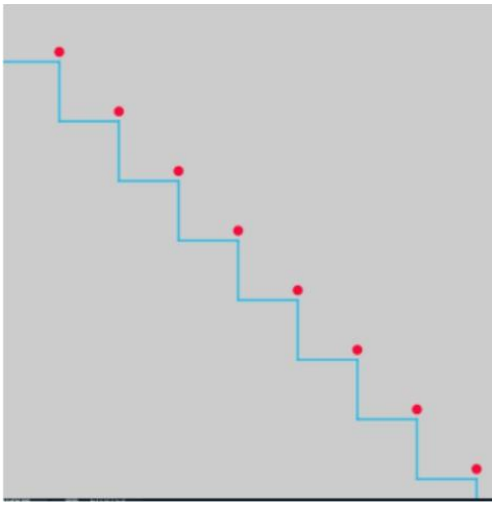
Ejercicio 19: Declare las variables necesarias para dibujar una línea que se dibuja desde las coordenadas iniciales del lienzo y se extiende por todo el ancho. Sobre el punto medio de la línea y a una distancia de 40px (en sentido vertical desde la línea) dibuje una elipse que tenga como ancho 80px y 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



Ejercicio 20: Dibuje en toda la extensión del lienzo de (440, 420) rectángulos de idénticas medidas (40 ancho y 20 de alto) y que mantengan una distancia de 20 pixeles entre ellos tanto horizontal como verticalmente. Utilice la estructura de control repetitiva for. El lienzo debería verse así:

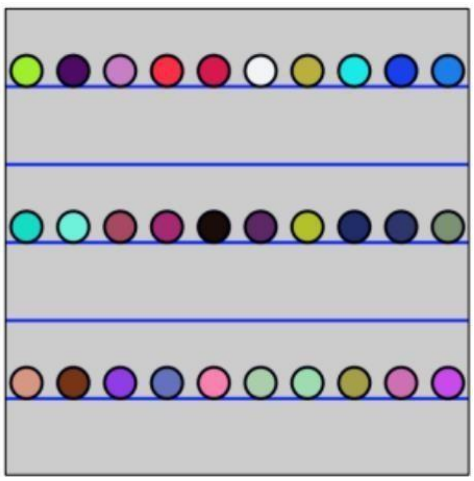


Ejercicio 21: Utilizando la estructura de control repetitiva while() dibuje la siguiente imagen utilizando líneas que forman escalones y sobre cada borde de escalón se dibuje un punto de color rojo



El tamaño del lienzo es size(500,500). La estructura while() se ejecuta dentro de la función setup(). La condición es que solo se dibuje dentro del lienzo. Utilice variables que puedan ayudar a la construcción del dibujo, por ej: x, y, anchoEscalon, altoEscalon, etc.

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***

Ejercicio 1

Resolución necesaria en Word:

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

6-(4*B/4)

6-5

1

ejercicio 1

```
1  int A=2,B=5;
2
3  float resultado = 3* A - 4 * B / pow(A,2);
4
5  println(resultado);
```

Ejercicio 2

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

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

ejercicio 2

```
1  float resultado = (( (4/2) * 3) /6) + ((((6/2) / 1)/ pow(5,2)) / 4)*2;
2  println("el resultado es " + resultado);
3
```

## Ejercicio 4

a. a) b ^ 2 – 4 * a * c=

(6^ 2)- (4 * 4 * 2) = **4**

ejercicio 4a

```
1  void setup() {
2    int A = 4;
3    int B = 6;
4    int C = 2;
5
6    int resultado = B*B - 4*A*C;
7    println(resultado);
8
9
0  }
```

| algrebraica |
| --- |
| $b^2 - 4ac$ |
| $6^2 - 4 \cdot 4 \cdot 2$ |
| $36 - 32 \ = 4$ |

b.

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

*(3*(3^4)) - (5 * (3^3)) + (3*12) – 17=* ***127***

ejercicio 4b

```
int X = 3;

float resultado = 3 * pow(X, 4) - 5 * pow(X, 3) + X * 12 - 17;
  println(resultado);
```

| Algebraica |
| --- |
| $3x^4 - 5x^3 + x^{12} - 17=$ |
| $3 \cdot 3^4 - 5 \cdot 3^3 + 3 \cdot 12 - 17=$ |
| $127$ |

c.

**(b + d) / (c + 4)** b=3, c=1, d=2

ejercicio 4 c

```
int b = 3;
int d = 1;
int c = 2;

float resultado = (b + d) / float(c + 4);
  println(resultado);
```

| Algebraica |
| --- |
| $\frac{(b+c)}{(c+4)}$ |
| $\frac{(3+1)}{(2+4)} = 0.66$ |

d.

(x ^ 2 + y ^ 2) ^ (1 / 2)  x=2  y=4

```
ejercicio 4 d
1  int x = 2;
2  int y = 4;
3
4  float resultado = pow(pow(x, 2) + pow(y, 2), 0.5);
5  println(resultado);
6
```

| Algebraica |
| --- |
| $(x^2 + y^2)^{(\frac{1}{2})}$ |
| $(2^2 + 4^2)^{(\frac{1}{2})}$ |
| $20^{(\frac{1}{2})} = 4{,}47$ |

## Ejercicio 5

a. B * A – B ^ 2 / 4 * C=

5 * 4 – ((5 ^ 2) / 4) * 1=

20- 6.25=13.75

```
ejercicio 5 a
1  void setup() {
2    int A = 4;
3    int B = 5;
4    int C = 1;
5
6    float resultado = B * A - (pow(B, 2) / 4) * C;
7    println(resultado);
8
9  }
```

b. (A * B) / 3 ^ 2

(A * B) / 3 ^ 2=

(4 * 5) / 3 ^ 2= 2.2

```
ejercicio 5 b
1  void setup() {
2    int A = 4;
3    int B = 5;
4
5    float resultado = (A * B) / pow(3, 2);
6    println(resultado);
7
8  }
```

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=

(22 * 3 * 5) – 6=

330 – 6 =324

ejercicio 5 c

```
void setup() {
  int A = 4;
  int B = 5;
  int C = 1;

  float resultado = (((B + C) / 2.0 * A + 10) * 3 * B) - 6;
  println(resultado);

}
```

## Ejercicio 6

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

R1 = y+z

R2 = x >= R1

ejercicio 6

```
void setup() {
  int x = 3;
  int y = 4;
  int z = 1;

  int R1 = y + z;
  boolean R2 = x >= R1;

  println("R1 = " + R1);
  println("R2 = " + R2);

}
```

R1= 4+1 = 5

R2= 3 >= R1

Falso

## Ejercicio 7

Para contador1=3, contador3=4, evaluar el resultado de

R1 = ++contador1

R2 = contador1 < contador2

ejercicio 7

```
void setup() {
  int contador1 = 3;
  int contador2 = 4;

  int R1 = ++contador1;
  boolean R2 = contador1 < contador2;

  println("R2 = " + R2);

}
```

R2= 4 < 4

R2= falso

## Ejercicio 8

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

a+b-1 < x*y

ejercicio 8

```
void setup() {
  int a = 31;
  int b = -1;
  int x = 3;
  int y = 2;

  boolean resultado = a + b - 1 < x * y;
  println(resultado);

}
```

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

29 < 12

falso

## Ejercicio 9

Para x=6, y=8, evaluar el resultado de

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

ejercicio 9

```
void setup() {
  int x = 6;
  int y = 8;

  boolean resultado = !(x < 5) && !(y >= 7);
  println(resultado);


}
```

Falso

## Ejercicio 10

Para i=22, j=3, evaluar el resultado de

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

ejercicio 10

```
void setup() {
  int i = 22;
  int j = 3;

  boolean resultado = !((i > 4) || !(j <= 6));
  println(resultado);

}
```

Falso

## Ejercicio 11

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

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

ejercicio 11

```
void setup() {
  int a = 34;
  int b = 12;
  int c = 8;

  boolean resultado = !(a + b == c) || (c != 0) && (b - c >= 19);
  println(resultado);

}
```

Verdadero

## Ejercicio 12

ejercicio 12

```
String texto_consola = "escribe tu nombre:";
String nombre_ingresado = "";
String mensaje_saludo = "";

void setup() {
  size(400, 200);
  println(texto_consola);
}

void draw() {
  background(0);
  text(mensaje_saludo, 100, 100);
  textSize(32);

}

void keyPressed() {
  nombre_ingresado += key;
  println(nombre_ingresado);

  if (key == '\n') {
    mensaje_saludo = "Hola, " + nombre_ingresado;
    println(mensaje_saludo);
  }
}
```

## Análisis:

Datos de Entrada: nombre_ingresado //cadena

Datos de Salida: mensaje_saludo //cadena de texto

Proceso: 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: ejercicio_12

Proceso del algoritmo:

1. *inicio*
2. *Leer nombre_ingresado*
3. *mensaje_saludo ← "Hola, " + nombre_ingresado + " ¡Bienvenido!"*
4. *Mostrar saludo*
5. *fin*

## Ejercicio 13

ejercicio 13

```
void setup() {
  float base = 10;
  float altura = 5;

  float perimetro = 2 * (base + altura);
  float area = base * altura;

  println(perimetro);
  println(area);
}
```

Análisis

| |
|---|
| Datos de Entrada: base, altura //decimal<br><br>Datos de Salida: perímetro, área // almacena valores decimales<br><br>Proceso: calcula el perímetro y el área de un rectángulo utilizando las fórmulas.<br><br>P=2(base+altura)  y A= base.altura |

Diseño

| |
|---|
| Entidad que resuelve el problema: persona |
| Variables:<br>- base: float // almacena un valor decimal<br>- area: float // almacena un valor decimal<br>- perimetro: float //<br>- area: float //<br>- perimetroArea:  float // almacena un valor de calculos |
| Nombre del Algoritmo: ejercicio_13 |
| Proceso del algoritmo:<br>1. *inicio*<br>2. *Leer base*<br>3. *Leer área* |

4. *perímetro ← 2*(base + altura)*
5. *área ← base * altura*
6. *Mostrar perímetro*
7. *Mostrar área*
8. *fin*

## Ejercicio 14

ejercicio 14

```
void setup() {

  float catetoA = 3;
  float catetoB = 4;

  float hipotenusa = sqrt(pow(catetoA, 2) + pow(catetoB, 2));

  println(hipotenusa);
}
```

Análisis:

Datos de Entrada: catetoA, catetoB

Datos de Salida: hipotenusa

Proceso: se aplica la fórmula: $h^2 = a^2 + b^2$

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 calculos

Nombre del Algoritmo: perimetro_area_rectangulo

Proceso del algoritmo:

*Leer catetoA*

1. *Inicio*
2. *Leer catetoA*
3. *Leer catetoB*
4. *hipotenusa ← \sqrt((a^(2)+b^(2)))*
5. *mostrar hipotenusa*
6. *Fin*

## Ejercicio 15

ejercicio 15

```
int num1=37, num2=12;

int suma = num1 + num2;
println("el resultado de la suma es: " + suma);
int resta = num1 - num2;
println("el resultado de la resta es: " + resta);
int multiplicacion = num1 * num2;
println("el resultado de la multiplicación es: " + multiplicacion);
float division = num1 / num2;
if (num2 != 0) {
    println("El resultado de la división es: " + division);
  }
```

Análisis:

Datos de Entrada: num1, num2

Datos de Salida: suma, resta, multiplicacion, division

Proceso: Resolver las operaciones pedidas.

## Diseño:

Entidad que resuelve el problema: persona

Variables:

- num1: int // almacena un valor entero
- num2: int // almacena un valor entero

Nombre del Algoritmo: ejercicio_15

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. *division ← num1 / num2*
11. *mostrar ← "el resultado de la división es: " + división*
12. *mostrar ← "la division por cero no está definida."*
13. *fin*

- suma: int // almacena un valor de una suma
- resta: int // almacena un valor de una resta
- multiplicacion: int // almacena un valor de una multiplicación
- division: int // almacena un valor de una división

## Ejercicio 16

ejercicio 16

```
void setup() {

  float temperaturaFahrenheit = 68;

  float temperaturaCelsius = (temperaturaFahrenheit - 32) / 1.8;

  println("La temperatura en Celsius es: " + temperaturaCelsius + " grados Celsius");
}
```

Análisis:

Datos de Entrada: Temperatura en grados Fahrenheit

Datos de Salida: Temperatura en grados Celsius

Proceso: Pasar mediante la formula de grados farenheit a grados celcius

## Diseño:

| |
|---|
| Entidad que resuelve el problema: persona |
| Variables:<br>-temperaturaFahrenheit: float // almacena un valor decimal<br>-temperaturaCelsius: float // almacena un valor decimal |
| Nombre del algoritmo: ejercicio_16 |
| Proceso del algoritmo:<br>1. *inicio*<br>2. *Leer temperaturaFahrenheit*<br>3. *temperaturaCelsius ← (9 / 5) * (temperaturaFahrenheit – 32)*<br>4. *mostrar temperaturaCelsius*<br>5. fin |

## Ejercicio 17

ejercicio 17

```
float x1 = 100;
float y1 = 100;
float x2 = 200;
float y2 = 400;
float distanciaTesoro = 50;

void setup() {
  size(800, 600);
}

void draw() {
  background(135, 206, 250);


  float coordenadaX = x2 - x1;
  float coordenadaY = y2 - y1;


  float distancia = sqrt(pow(coordenadaX, 2) + pow(coordenadaY, 2));
  String textoDistancia = "La distancia es de: " + distancia;
  println(textoDistancia);


  if (distancia <= distanciaTesoro) {
    println("¡Power-Up!");
  }
  fill(255, 215, 0);
  rectMode(CENTER);
  rect(x2, y2, 40, 40);


  fill(0, 191, 255);
  ellipse(x1, y1, 30, 30);


  String coordenadas = "X1: " + mouseX + ", Y1: " + mouseY;
  fill(0);
  textSize(20);
  textAlign(RIGHT, TOP);
  text(coordenadas, width, 0);
}

void mouseMoved() {

  x1 = mouseX;
  y1 = mouseY;
}
```

Análisis:

| |
|---|
| Datos de Entrada: Coordenadas de Link, Coordenadas del tesoro<br>Datos de Salida: Distancia entre Link y tesoro.<br>Proceso: Calcular 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:<br>- x1: float // decimal<br>- y1: float // decimal<br>- x2: float // decimal<br>- y2: float // decimal<br>- coordenadaX: float // almacena el resultado de un calculo<br>- coordenadaY: float //almacena el resultado de un calculo<br>- textoDistancia: float // almacena el resultado de un calculo<br>- distanciaTesoro: float // almacena un valor |

| Nombre del Algoritmo: Ejercicio_17 |
|---|
| **Proceso del algoritmo:**<br>1. *inicio*<br>2. *Leer x1*<br>3. *Leer y1*<br>4. *Leer x2*<br>5. *Leer y2*<br>6. *distanciaTesoro ← 50*<br>7. *anchoLienzo*<br>8. *altoLienzo*<br>9. *coordenadaX ← x2 - x1*<br>10. *coordenadaY ← y2 – y1*<br>11. *distancia←sqrt(pow(coordenadaX, 2) + pow(coordenadaY, 2))*<br>12. *mostrar "la distancia es de: " + distancia*<br>13. *leer textoDistancia*<br>14. ***si** (distancia <= distanciaTesoro) **entonces***<br>15. ***mostrar** "¡PowerUp!"*<br>16. *fin* |

## Ejercicio 18

ejercicio 18

```
void setup() {

  float a = 1;
  float b = -3;
  float c = 2;

  float discriminante = b * b - 4 * a * c;

  if (discriminante > 0) {

    float x1 = (-b + sqrt(discriminante)) / (2 * a);
    float x2 = (-b - sqrt(discriminante)) / (2 * a);
    println("Las raíces son: " + x1 + " y " + x2);
  } else if (discriminante == 0) {

    float x = -b / (2 * a);
    println("La raíz doble es: " + x);
  } else {

    println("Las raíces son complejas");
  }
}
```

## Análisis:

| |
|---|
| Datos de Entrada: Coeficientes de la ecuación cuadrática: a, b y c.<br><br>Datos de Salida: Raíces de la ecuación cuadrática.<br><br>Proceso: Calcular el discriminante de la ecuación cuadrática utilizando la fórmula |

## Diseño:

| |
|---|
| Entidad que resuelve el problema: persona |
| Variables:<br><br>-a : float // almacena un valor decimal<br><br>-b : float // almacena un valor decimal<br><br>-c : float // almacena un valor decimal<br><br>-discriminante: float //almacena el valor de calculos |

**Nombre del Algoritmo:** ejercicio_18

**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

## Ejercicio 19

### Análisis

| |
|---|
| Datos de entrada: lineaY, velocidadY, direccionAbajo<br><br>Datos de Salida: bucle de la línea y circulo<br><br>Proceso: se crea un bucle donde la línea empuja al circulo |

### Diseño

| |
|---|
| Entidad que resuelve el problema: lienzo |
| Variables:<br><br>-lineaY: entero<br><br>-velocidadY: entero<br><br>-direccionAbajo:<br><br>-elipseY: decimal<br><br>-elipseX: decimal |
| Nombre del algoritmo: ejercicio_19 |
| Proceso:<br><br>1. inicio<br>2. Leer lineaY<br>3. Leer velocidadY<br>4. anchoLienzo ←400<br>5. altoLienzo←400<br>6. si direccionAbajo entonces<br>7. lineaY += velocidadY<br>8. si_no<br>9. lineaY-= velocidadY<br>10. leer elipseX ← anchoElipse / 2 |

ejercicio 19

```
int lineaY;
int velocidadY = 1;
boolean direccionAbajo = true;

void setup() {
  size(400, 400);
  lineaY = height / 2;
}

void draw() {
  if (direccionAbajo) {
    lineaY += velocidadY;
  } else {
    lineaY -= velocidadY;
  }


  background(0);
  stroke(255);
  line(0, lineaY, width, lineaY);


  float elipseX = width / 2;
  float elipseY;
  if (direccionAbajo) {
    elipseY = lineaY + 40;
  } else {
    elipseY = lineaY - 40;
  }
  ellipse(elipseX, elipseY, 80, 80);


  if (lineaY >= height || lineaY <= 0) {
    direccionAbajo = !direccionAbajo;
  }
}
```

11. leer elipseY
12. si direccionAbajo entonces
13. elipseY= lineaY  + 40;
14. si_no
15. elipseY= lineaY – 40
16. dibujar elipse
17. si lineaY >= alto || líneaY <= 0 entonces
18. direccionAbajo = !direccionAbajo;
19. fin
20. fin

## Ejercicio 20

Análisis

Datos de Entrada: Rectángulos dibujados en el lienzo según las indicaciones dadas.

Datos de Salida: Rectángulos dibujados en el lienzo según las indicaciones dadas.

Proceso: dibujar una serie de rectángulos en un lienzo de tamaño especifico, manteniendo una distancia especifica entre ellos tanto vertical como horizontalmente, definiendo un bucle for para dibujarlos.

Diseño

Entidad que resuelve el problema: Lienzo

Variables:

- coordenadasRect: float//almacena un valor de coordenadas
- distRect, anchoRect, altoRect: int // almacena un valor entero
- anchoLienzo, altoLienzo: int // almacena valores enteros

Nombre de Algoritmo: ejecicio_20

Proceso del algoritmo:

1. inicio
2. anchoLienzo
3. altoLienzo
4. distRect <- 20
5. anchoRect <- 40
6. altoRect <- 20
7. para x coordenadasRect.x hasta anchoLienzo con paso (anchoRect+distRect)
8. hacer
9. para y = coordenadasRect.y hasta altoLienzo con paso (altoRect+distRect)
10. hacer
11. dibujar rectángulo en (x,y,ancho,alto)
12. fin_para
13. fin_para
14. fin

Ejercicio20

```
PVector coordenadasRect;
int altoRect, anchoRect, distRect;

void setup() {
  size(440, 420);
  distRect = 20;
  anchoRect = 40;
  altoRect = 20;
  coordenadasRect = new PVector(distRect, distRect);
}

void draw() {
  background(#FFFF99);
  fill(#C11010);
  stroke(#FCF32E);
  dibujarRec();
}

void dibujarRec() {
  for (float x = coordenadasRect.x; x < width; x += (anchoRect + distRect)) {
    for (float y = coordenadasRect.y; y < height; y += (altoRect + distRect)) {
      rect(x, y, anchoRect, altoRect);
    }
  }
}
```

## Ejercicio 21

ejercicio 21

```
int distancia;
PVector puntoA, puntoB, puntoC, puntoD;


public void setup (){
  size(500,500);
  distancia=60;
  puntoA = new PVector(0,distancia);

  while(puntoA.y <= height){
  escalon();
  circulo();
  repeticion();
  }
}

public void escalon(){
  stroke(0);
  strokeWeight(5);
  puntoB = new PVector(puntoA.x+distancia, puntoA.y);
  line(puntoA.x, puntoA.y,puntoB.x,puntoB.y);
  puntoC = new PVector(puntoB.x,puntoB.y+60);
  line(puntoB.x,puntoB.y,puntoC.x,puntoC.y);
}

public void circulo(){
  stroke(#FF0000);
  strokeWeight(9);
  puntoD = new PVector(puntoB.x, puntoB.y-8);
  point(puntoD.x,puntoD.y);
}

public void repeticion(){
  puntoA.x = puntoC.x;
  puntoA.y = puntoC.y;
}
```

## Análisis:

| |
|---|
| Datos de Entrada: puntoA, puntoB, puntoC, puntoD, distancia<br>Datos de Salida: una imagen que consiste en escalones con puntos de color rojo en los bordes.<br>Proceso: El proceso consiste en iterar mediante while() para dibujar escalones y puntos rojos en los bordes |

## Diseño:

| |
|---|
| Entidad que resuelve el problema: programa |
| Variables:<br><br>-puntoA,puntoB,puntoC,puntoD: int //almacena un vector<br><br>-distancia : int //almacena un valor entero |

| Nombre del Algoritmo: ejercicio_21 |
| --- |
| Proceso del algoritmo:<br>1. *Inicio*<br>2. *anchoLienzo ← 500*<br>3. *altoLienzo ← 500*<br>4. *distancia ← 60*<br>5. *mientras (puntoA.y sea menor o igual que anchoLienzo) Hacer*<br>6. *dibujar línea horizontal en (puntoA.x, puntoA.y, puntoB.x, puntoB.y)*<br>7. *dibujar línea vertical en (puntoB.x, puntoB.y, puntoC.x, puntoC.y)*<br>8. *dibujar circulo en (puntoD.x, puntoD.y)*<br>9. *puntoA.x ← puntoC.x*<br>10. *puntoA.y ← puntoC.y*<br>11. *fin_mientras*<br>12. *fin* |

## Ejercicio 22

ejercicio 22

```
void setup() {
  size(600, 600);
  int lineaX = 0;
  int lineaY = 100;
  int circuloY = 75;
  int distanciaCirculo = 30;

  do {
    int circuloX = distanciaCirculo;

    do {
      stroke(#00008b);
      line(lineaX, lineaY, width, lineaY);
      fill(random(255), random(255), random(255));
      stroke(0);
      strokeWeight(2);
      ellipse(circuloX, circuloY, 50, 50);
      circuloX += distanciaCirculo * 2;
    } while (circuloX < width);

    lineaY += 100;
    circuloY += 200;
  } while (lineaY < height);
}
```

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: 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.

Diseño:

Entidad que resuelve el problema: processing

Variables:

- distanciaCiruclo: int //almacena un valor entero
- lineaX, lineaY, circuloX, circuloX, distanciaCirculo : int //almacena un valor entero
- anchoLienzo, altoLienzo: int //almacenan valores enteros

Nombre del Algoritmo: rectangulos_repetidos

Proceso del algoritmo:

1. *inicio*
2. *anchoLienzo ← 600*
3. *altoLienzo ← 600*
4. *lineaX ← 0*
5. *lineaY ← 100*
6. *circuloY ← 75*
7. *distanciaCirculo ← 30;*
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*