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

*Ing. Carolina Cecilia Apaza*

*Año*

Trabajo Práctico

N° 1

Pérez, Ezequiel Alejandro

LU: TUV000625

**FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS**

Ejercicio 1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 3

Ejercicio 2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 3

Ejercicio 4\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 4

Ejercicio 5\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 7

Ejercicio 6\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 8

Ejercicio 7\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 9

Ejercicio 8\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 9

Ejercicio 9\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 9

Ejercicio 10\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 10

Ejercicio 11\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 10

Ejercicio 12\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 11

Ejercicio 13\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 12

Ejercicio 14\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 14

Ejercicio 15\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 15

Ejercicio 16\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 18

Ejercicio 17\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 19

Ejercicio 18\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 21

Ejercicio 19\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 23

Ejercicio 20\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 25

Ejercicio 21\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 27

Ejercicio 22\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 30

Fuentes bibliográficas\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 33

**Sección Expresiones aritméticas y lógicas**

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

3 \* A – 4 \* B / A ^ 2

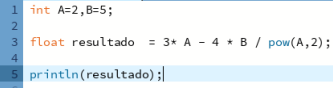
(3\*A) - (4\*B/(A^2))

6-(4\*B/4)

6-5

1

**Código:**

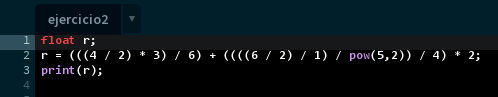


**Ejercicio 2:** Evaluar la siguiente expresión: 4 / 2 \* 3 / 6 + 6 / 2 / 1 / 5 ^ 2 / 4 \* 2

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

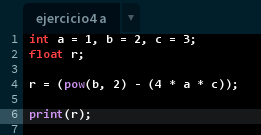
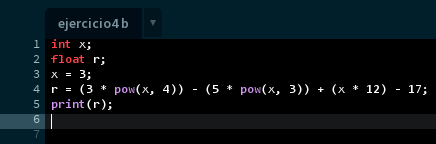
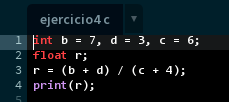
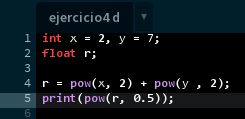
1 + 0.06 = 1.06

Código:



~~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  
     
   a = 1, b =2 , c = 3  
     
   **Aritmética:**b ^ 2 – 4 \* a \* c  
   (2 ^ 2) – (4 \* 1 \* 3)  
   4 – 12  
   -8 **Algebraica:**  
     
    b2 – 4ac  
   22 – 4 . 1 . 3  
   4 – 12  
   -8  
     
   **Código:**
2. 3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17   
     
   X = 3  
     
   **Aritmética:**(3 \* (X ^ 4)) – (5 \* (X ^ 3)) + X 12 – 17  
   (3 \* (3 ^ 4)) – (5 \* (3 ^ 3)) + (3 \* 12) – 17  
   (3 \* 81) – (5 \* 27) + (3 \* 12) – 17  
   243 – 135 + 36 – 17  
   127 **Algebraica:**3x4 – 5x3 + 12x – 17  
   3 . 34 – 5 . 33 + 12 . 3 – 17  
   3 . 81 – 5 . 27 – 12 . 3 – 17  
   243 – 135 + 36 – 17  
   127  
     
   **Código:  
     
   **
3. (b + d) / (c + 4)   
     
   b = 7, d = 3, c = 6  
     
   **Aritmética:**(b + d) / (c + 4)  
   (7 + 3) / (6 + 4)  
   10 / 10  
   1 **Algebraica:  
     
   Código:**
4. (x ^ 2 + y ^ 2) ^ (1 / 2)  
     
   x =2 , y = 7  
     
   **Aritmética:**(x ^ 2 + y ^ 2) ^ (1 / 2)  
   (2 ^ 2 + 7 ^ 2) ^ (1 / 2)  
   (4 + 49) ^ (1 / 2)  
   53 ^ (1 / 2)  
   7.2801098893  
     
    **Algebraica:**  
   **Código:**

**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

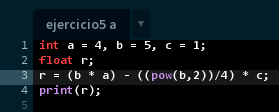
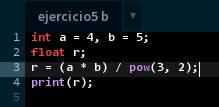
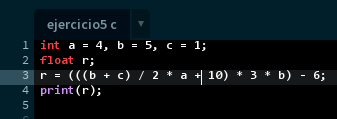
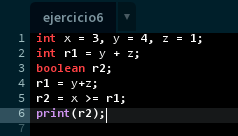
b) (A \* B) / 3 ^ 2

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

**Resolución:**  
  
a) (5 \* 4) – ((5 ^ 2) / 4) \* 1

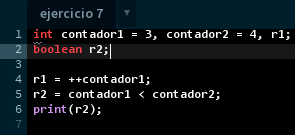
20 – (25 / 4) \* 1

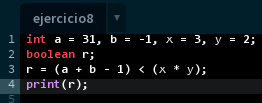
20 – 6.25

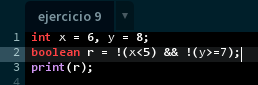
13.75  
  
  
**Código:**b) (A \* B) / 3 ^ 2  
  
(4 \* 5) / 3 ^ 2  
  
(4 \* 5) / 9  
  
20 / 9  
  
2.22  
  
**Código:**c) (((B + C) / 2 \* A + 10) \* 3 \* B) – 6  
  
(((5 + 1) / 2 \* 4 + 10) \* 3 \* 5) – 6  
((6 / 2 \* 4 + 10) \* 3 \* 5) – 6  
(3 \* 4 + 10) \* 3 \* 5) – 6  
(12 + 10) \* 3 \* 5) – 6  
(22 \* 3 \* 5) – 6  
330 – 6  
324  
  
**Código:**  
  
**Ejercicio 6**: Para x=3, y=4; z=1, evaluar el resultado de:   
  
R1 = y+z   
R2 = x >= R1  
  
**R1** = 4 + 1  
**R2** = x >= **R1** 🡪 Falso  
**Código:**  
  


**Ejercicio 7**: Para contador1=3, contador3=4, evaluar el resultado de   
  
R1 = ++contador1   
R2 = contador1 < contador2

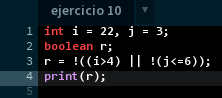
**R1** = ++contador1  
**R2** = contador1<contador2 🡪 Falso

**Código:**  
  
**Ejercicio 8**: Para a=31, b=-1; x=3, y=2, evaluar el resultado de:  
  
a+b-1 < x\*y  
  
31+(-1)-1 < 3 \* 2  
  
30 – 1 < 6  
  
29 < 6

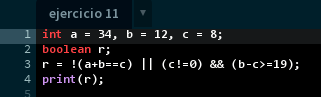
Falso  
  
**Código:**

**Ejercicio 9**: Para x=6, y=8, evaluar el resultado de:  
  
!(x<5)CC !(y>=7)  
  
!(6<5) && !(8>=7)  
  
!(False) && !(True)  
  
True && False  
  
False  
  
**Código:  
  
**

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

!(True || !(True))  
!(True || False)  
!(True)  
False  
  
**Código:**

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

!(a+b==c) || (c!=0)CC(b-c>=19)  
!(46==8) || (8!=0)CC(12 - 8>=19)  
!(False) || (True) && (4>=19)  
True || (True) && (False)  
True || False  
True  
  
**Código:**

**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.

**Fase de Análisis:**

* Especificación del problema: Pedir al usuario su nombre y mostrarlo por pantalla
* Análisis:

Datos de entrada:

nombreUsuario: **String**

Datos de salida:

darMensaje: **String**

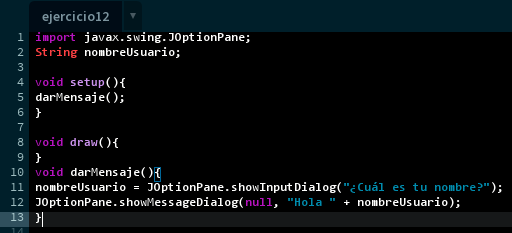
Proceso:

// se le pedirá al usuario que escriba su nombre

// con el nombre del usuario, se creará el mensaje saludándolo  
darMensaje = “Hola ” + nombreUsuario

**Fase de diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** El programador |
| **Variables:**  nombreUsuario: **String** // va a almacenar lo que el usuario ingrese |
| **Nombre del algoritmo:** darMensaje  **Proceso:**   1. **Inicio** 2. *Mostrar* “¿Cuál es tu nombre?” 3. *Leer nombreUsuario* 4. darMensaje 🡨 “Hola ” + nombreUsuario 5. *Mostrar* darMensaje 6. **fin** |

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

**Fase de Análisis:**

* Especificación del problema: Calcular el perímetro y área de un rectángulo dadas la base y la altura
* Análisis:

Datos de entrada:

base: **flotante**  
 altura: **flotante**

Datos de salida:

perimetro: **flotante**

area: **flotante**

perimetro\_area: **string**

Proceso:

// se le pedirá al usuario que ingrese el valor de la base y la altura respectivamente

// posterior a esto, se harán los siguientes cálculos:

perimetro = base \* 2 + altura \* 2

area = base \* altura

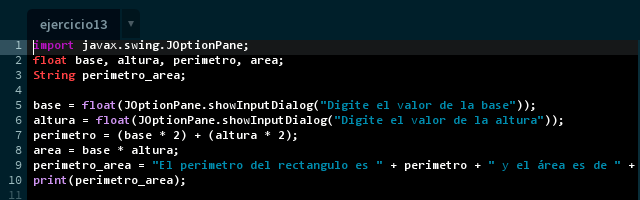
//luego, se mostrará por consola el siguiente mensaje:

perimetro\_area = "El perimetro del rectangulo es " + perimetro + " y el área es de " + area

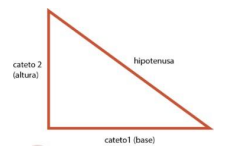
**Fase de diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** Programador |
| **Variables:**  base, altura: **flotante** // almacenaran los valores de la base y la altura respectivamente  perimetro, area: **flotante** // almacenaran los resultados  perimetro\_area: **string** // se utilizará para guardar nuestro mensaje |
| **Nombre del algoritmo:** perimetro\_area  **Proceso:**   1. **Inicio** 2. *Mostrar* “Digite el valor de la base” 3. *Leer* base 4. *Mostrar* “Digite el valor de la altura” 5. *Leer* altura 6. perimetro 🡨 base \* 2 + altura \* 2 7. area 🡨 base \* altura 8. perimetro\_area 🡨 "El perimetro del rectangulo es " + perimetro + " y el área es de " + area 9. *Mostrar* perímetro\_area 10. **Fin** |

**Código:**



**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



**Fase de Análisis:**

* Especificación del problema: Calcular la hipotenusa conociendo los catetos
* Análisis:

Datos de entrada:

cat1: **flotante**

cat2: **flotante**

Datos de salida:

hip: **flotante**

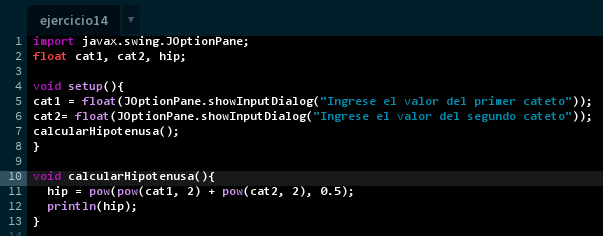
Proceso:

// se le pedirá al usuario que ingrese el valor de cada uno de los catetos, posterior a esto, se hará el siguiente calculo:

**Fase de diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** El programador |
| **Variables:**  cat1, cat2: **Flotante** // almacenaran los valores que ingrese el usuario  hip: **Flotante** // almacenara el resultado del cálculo de la hipotenusa |
| **Nombre del algoritmo:** calcularHipotenusa  **Proceso:**   1. **Inicio** 2. *Mostrar* "Ingrese el valor del primer cateto" 3. *Leer* cat1 4. *Mostrar "*Ingrese el valor del segundo cateto" 5. *Leer* cat2 6. hip 🡨 (((cat1 ^ 2) + (cat2 ^ 2)) ^ 0.5) 7. *Mostrar* hip 8. **Fin** |

**Código:**



**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.

**Fase de Análisis:**

* Especificación del problema: Realizar suma, resta, multiplicación y división de dos números ingresados
* Análisis:

Datos de entrada:

Num1: **Flotante**

Num2: **Flotante**

Datos de salida:

Suma: **Flotante**

Resta: **Flotante**

Multiplicación: **Flotante**

División: **Flotante**

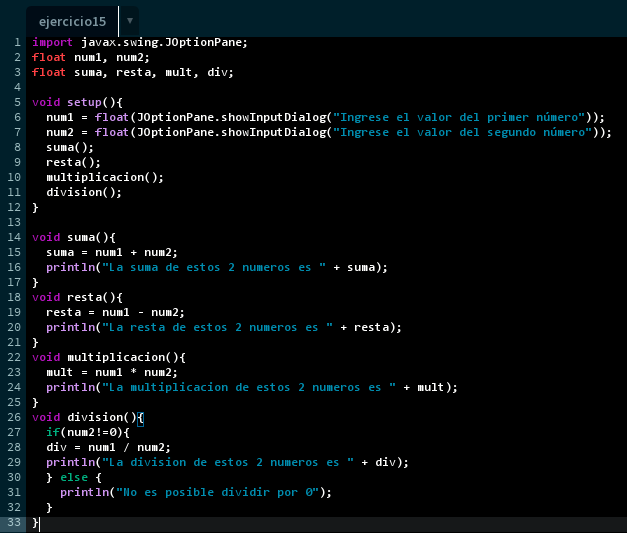
Proceso:

// se pedirá al usuario que ingrese dos números, luego de eso, se procederá a hacer las siguientes operaciones con estos números:

**Fase de diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** Programador |
| **Variables:**  num1, num2: **Flotante** // almacenara los números que ingrese el usuario  suma, resta, mult, div: **Flotante** // almacenara los resultados de las operaciones solicitadas |
| **Nombre del algoritmo:** calculadora  **Proceso:**   1. **Inicio** 2. *Mostrar* "Ingrese el valor del primer número" 3. *Leer* num1 4. *Mostrar* "Ingrese el valor del segundo número" 5. *Leer* num2 6. suma 🡨 num1 + num2 7. *Mostrar* “La suma de estos 2 números es ” + suma 8. resta 🡨 num1 – num2 9. *Mostrar* “La resta de estos 2 números es ” + resta 10. mult 🡨 num1 \* num2 11. *Mostrar* “La multiplicación de estos 2 números es ” + mult 12. **si** (num2 != 0) **entonces** 13. div 🡨 num1 / num2 14. *Mostrar* “La división de estos 2 números es ” + div 15. **si\_no** 16. *Mostrar* “No es posible dividir por 0” 17. **Fin** |

**Código:**



**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:

****

**Fase de Análisis:**

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

Datos de entrada:

tempF: **Flotante** // va a almacenar la temperatura en Fahrenheit

Datos de salida:

tempC: **Flotante** // va a almacenar el valor del resultado de la conversión

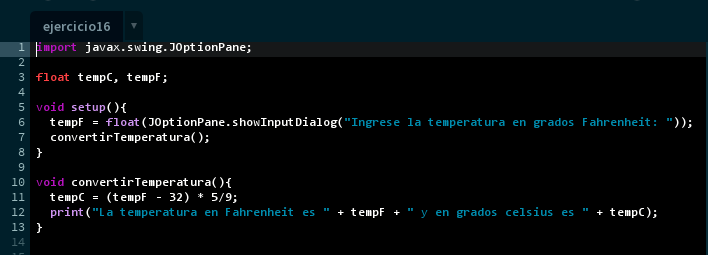
Proceso:

// se pedirá al usuario que ingrese la temperatura en Fahrenheit, posterior a esto, se hará el siguiente calculo:

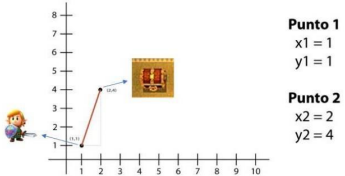
**Fase de diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** Programador |
| **Variables:**  tempF: **Flotante**  tempC: **Flotante** |
| **Nombre del algoritmo:** convertirTemperatura  **Proceso:**   1. **Inicio** 2. *Mostrar* “Ingrese la temperatura en grados Fahrenheit:” 3. *Leer* tempF 4. tempC 🡨 (tempF – 32) \* 5/9 5. *Mostrar* "La temperatura en Fahrenheit es " + tempF + " y en grados celsius es " + tempC 6. **Fin** |

**Código:**

****

**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.

**Fase de Análisis:**

* Especificación del problema: Hallar la distancia entre Link y el tesoro, mover a link mediante el mouse
* Análisis:

Datos de entrada:

link, tesoro: **PVector**

Datos de salida:

distancia: **Flotante**

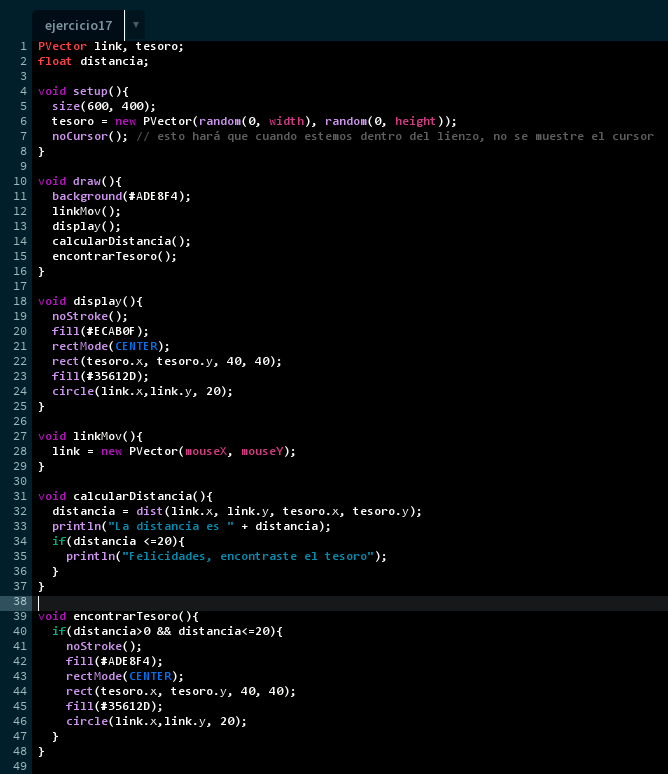
Proceso:

// se dibujará en el lienzo un circulo representando a link y un cuadrado representando al tesoro, se moverá a link con el mouse, por medio de la consola se ira mostrando la distancia que separa a link del tesoro

**Fase de diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** Programador |
| **Variables:**  link, tesoro: **PVector** // almacenaran las coordenadas de link y las coordenadas donde se dibujará el tesoro |
| **Nombre del algoritmo:** calcularDistancia  **Proceso:**   1. **Inicio** 2. altoLienzo 🡨 400 3. anchoLienzo 🡨 600 4. tesoro 🡨 PVector(random(0, anchoLienzo), random(0, altoLienzo)) 5. background🡨#ADE8F4 6. circle(link.x,link.y, 20) 7. rect(tesoro.x, tesoro.y, 40, 40) 8. link 🡨 PVector(mouseX, mouseY) 9. distancia 🡨 dist(link.x, link.y, tesoro.x, tesoro.y) 10. **si** distancia < 20 **entonces** 11. *Mostrar* “Felicidades, encontraste el tesoro” 12. **Fin\_si** 13. **Si** distancia > 0 && distancia<=20 **entonces** 14. rect(tesoro.x, tesoro.y, 40, 40); 15. circle(link.x,link.y, 20); 16. **Fin** |

**Código:**



**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.

**Fase de Análisis:**

* Especificación del problema: Obtener las raíces de una ecuación de segundo grado.
* Análisis:

Datos de entrada:

a: **Flotante**

b: **Flotante**

c: **Flotante**

Datos de salida:

raiz1: **Flotante**

raiz2: **Flotante**

raizDoble: **Flotante**

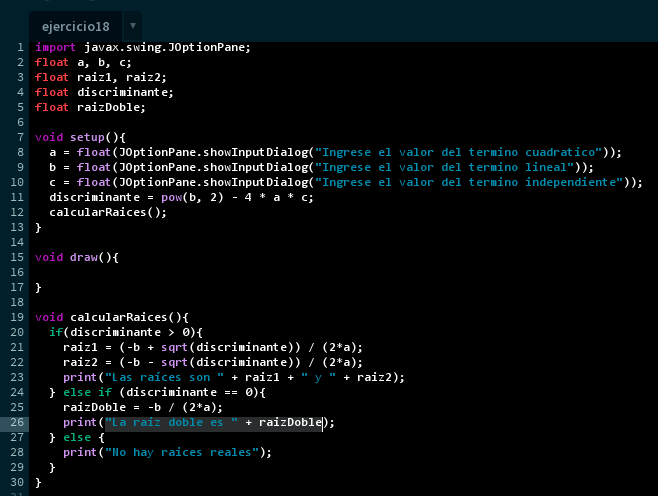
Proceso:

// se le pedirá al usuario que ingrese los valores del termino cuadrático, lineal e independiente, luego, procederemos a analizar el discriminante y según sea su resultado, sabremos si tienen 2 raíces reales, una raíz doble o si no tiene raíz dentro de los números reales.

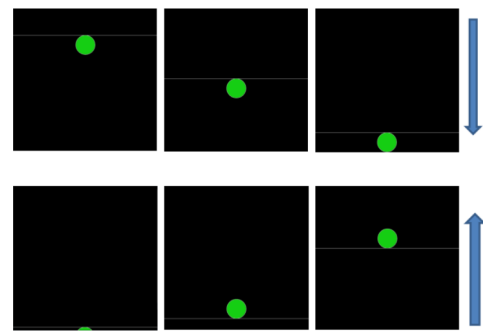
**Fase de diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** Programador |
| **Variables:**  a, b, c: **Flotante** // almacenara los valores del termino cuadrático, lineal e independiente respectivamente  raiz1, raiz2, raizDoble: **Flotante** // almacenara las raíces de la ecuación cuadrática según corresponda |
| **Nombre del algoritmo:** calcularRaices  **Proceso:**   1. **Inicio** 2. *Mostrar* "Ingrese el valor del termino cuadratico" 3. *Leer* a 4. *Mostrar* “Ingrese el valor del termino lineal" 5. *Leer b* 6. *Mostrar* "Ingrese el valor del termino independiente" 7. *Leer* c 8. discriminante 🡨 ((b ^ 2) – 4 \* a \* c) 9. **si** (discriminante > 0) **entonces** 10. raiz1 🡨 (-b + √discriminante) / (2 \* a) 11. raiz2 🡨 (-b - √discriminante) / (2 \* a) 12. *Mostrar* "Las raíces son " + raiz1 + " y " + raiz2 13. **si\_no si** (discriminante == 0) **entonces** 14. raizDoble 🡨 -b / (2 \* a) 15. *Mostrar "La raíz doble es " + raizDoble* 16. **si\_no** 17. *Mostrar* “No hay raíces reales” 18. **fin\_si** 19. **Fin** |

**Código:**

****

**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

****

**Fase de Análisis:**

* Especificación del problema: Dibujar una línea con un circulo sobre el punto medio de la misma, este círculo deberá seguir el trayecto de la línea
* Análisis:

Datos de entrada:

linea, aux: **enteros**

Datos de salida:

Línea y circulo dibujado en el lienzo

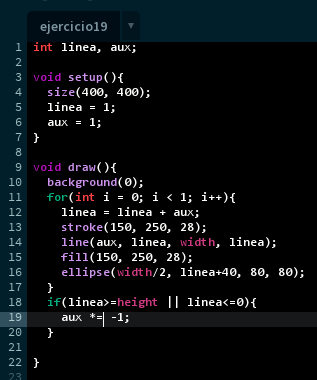
Proceso:

// se dibujará una línea que ocupara toda la extensión del lienzo, en su punto medio se dibujara un circulo, el cual seguirá la trayectoria que haga la misma

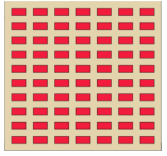
**Fase de diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** Programador |
| **Variables:**  Linea, aux: **Enteros** |
| **Nombre del algoritmo:** dibujarLineaCirculo  **Proceso:**   1. **Inicio** 2. size 🡨 (400, 400) 3. linea 🡨 1 4. *leer* linea 5. aux 🡨 1 6. *leer* aux 7. background🡨(0) 8. **para** i🡨1 **hasta** alto **incremento** 1 **hacer** 9. linea 🡨 linea + aux 10. dibujarLinea (aux, linea, ancho, linea) 11. elipse(ancho/2, linea+40, 80, 80) 12. **fin\_para** 13. **si** (linea >= altoLienzo || linea <= 0) **entonces** 14. aux \*= -1 15. **fin\_si** 16. **Fin** |

**Código:**

****

**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:

****

**Fase de Análisis:**

* Especificación del problema: Dibujar rectángulos manteniendo una distancia de 20px entre ellos.
* Análisis:

Datos de entrada:

anchoLienzo: **Entero**

altoLienzo: **Entero**

anchoRect: **Entero**

altoRect: **Entero**

posicion: **PVector**

Datos de salida:

Rectángulos dibujados en el lienzo

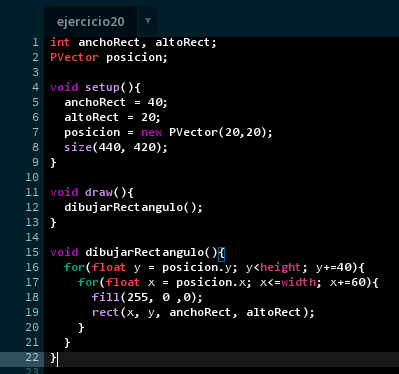
Proceso:

// se dibujarán rectángulos en un lienzo de 440 de ancho y 420 de alto, cada uno tendrá un tamaño de 40 de ancho y 20 de alto, y mantendrán una distancia de 20px tanto horizontal como verticalmente

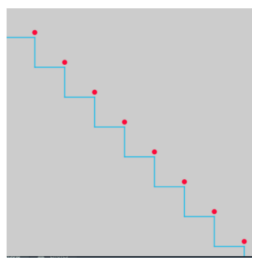
**Fase de diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** |
| **Variables:**  anchoRect: **Entero** // almacena el valor del ancho de los rectángulos  altoRect: **Entero** // almacena el valor del alto de los rectángulos  posicion: **PVector** // almacenara las coordenadas en la que comenzaran a dibujarse los rectángulos |
| **Nombre del algoritmo:** dibujarRectangulo  **Proceso:**   1. **Inicio** 2. anchoLienzo 🡨 440 3. altoLienzo 🡨 420 4. anchoRect 🡨 40 5. altoRect 🡨 20 6. posicion 🡨 (20, 20) 7. ***para*** y 🡨 posicion.y ***hasta*** altoLienzo ***incremento*** 40 ***hacer*** 8. ***para*** x 🡨 posicion.x ***hasta*** anchoLienzo ***incremento*** 60 ***hacer*** 9. dibujarRectangulo (x, y, anchoRect, altoRect) 10. ***fin\_para*** 11. ***fin\_para*** 12. **Fin** |

**Código:**



**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.

**Fase de Análisis:**

* Especificación del problema: Dibujar escalones con un pequeño círculo en cada uno de ellos
* Análisis:

Datos de entrada:

x, y, anchoEsc, altoEsc: **Enteros**

Datos de salida:

// escalones dibujados y círculos en cada uno de ellos

Proceso:

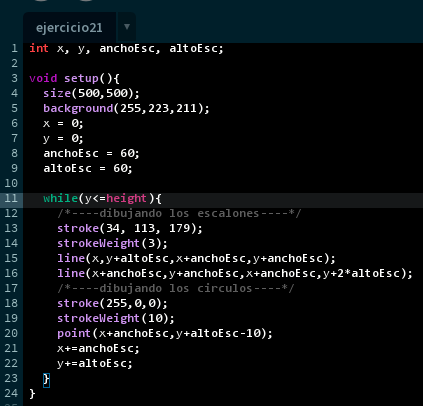
// una vez definidas tanto el alto como el ancho del escalon, con uso del while/mientras, se procederá a dibujar en el lienzo dichos escalones, que iran intercalándose a medida que vayan sumando sus valores.

// ya con los escalones dibujados, se procederá a dibujar los pequeños círculos en cada escalon, para ello, le restaremos 10 unidades al valor que contenga y en ese momento

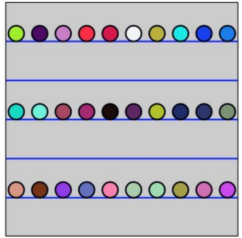
**Fase de diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** Programador |
| **Variables:**  x, y, anchoEsc, altoEsc: **Enteros** |
| **Nombre del algoritmo:** dibujarEscalonCirculo  **Proceso:**   1. **Inicio** 2. anchoLienzo 🡨 500 3. altoLienzo 🡨 500 4. background(255,223,211) 5. x 🡨 0 6. y 🡨 0 7. anchoEsc 🡨 60 8. altoEsc 🡨 60 9. **mientras** y <= altoLienzo **hacer** 10. line(x, y+altoEsc, x+anchoEsc, y+anchoEsc) 11. line(x+anchoEsc, y+anchoEsc, x+anchoEsc, y+2\*altoEsc) 12. point(x+anchoEsc, y+altoEsc-10) 13. x+=anchoEsc 14. y+=altoEsc 15. **fin\_mientras** 16. **Fin** |

**Código:**



**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.

**Fase de Análisis:**

* Especificación del problema: Dibujar líneas en el lienzo, dibujar circulos por línea de por medio.
* Análisis:

Datos de entrada:

linea, y: **Enteros**

Datos de salida:

// líneas y círculos mostrados por pantalla

Proceso:

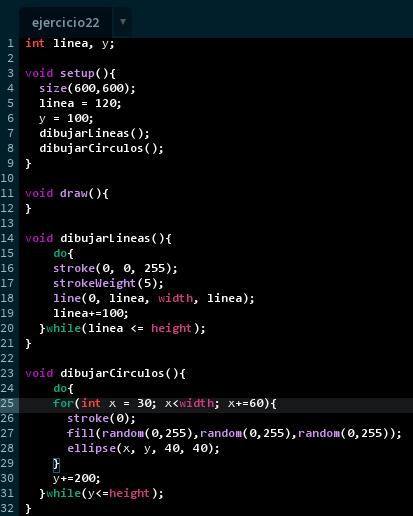
// se dibujarán líneas con una separación entre ellas

// se debe dibujar círculos manteniendo una distancia entre ellos, solo se dibujaran círculos por linea de por medio, es decir que, si ya se dibujaron círculos en la primera linea, en la segunda no se dibujaran, y se pasara a la tercera linea

**Fase de diseño:**

|  |
| --- |
| **Entidad que resuelve el problema:** Programador |
| **Variables:**  linea, y: **Enteros** |
| **Nombre del algoritmo:** dibujarLineasCirculos  **Proceso:**   1. **Inicio** 2. anchoLienzo 🡨 600 3. altoLienzo 🡨 600 4. linea 🡨 120 5. y 🡨 100 6. **hacer** 7. line(0, linea, ancho, linea) 8. linea += 100 9. **mientras** (linea <= altoLienzo) 10. **hacer** 11. **para** x🡨30 **hasta** anchoLienzo **incremento** 60 **hacer** 12. ellipse(x, y, 40, 40) 13. **fin\_para** 14. y+=200 15. **mientras** (y <= altoLienzo) 16. **Fin** |

**Código:**

****

Fuentes bibliográficas:

[documentacion processing](https://processing.org/reference)

[discriminante ecuaciones cuadraticas](https://youtu.be/ocDqH0UT8Zs?si=1bhZmMbMDbaCh21Y)

[JOptionPane Java](http://www.edu4java.com/es/java/joptionpane-showmessagedialog-showinputdialog.html)

[estructuras iterativas](https://virtual.unju.edu.ar/pluginfile.php/617801/mod_resource/content/3/08%20Estructuras%20Iterativas.pdf)

[glosario processing](https://virtual.unju.edu.ar/pluginfile.php/633255/mod_resource/content/2/Glosario%20Processing.pdf)

[bifurcacion - operadores logicos - estructuras selectivas](https://virtual.unju.edu.ar/pluginfile.php/616669/mod_resource/content/5/06%20Bifurcaciones%20-%20Expresiones%20L%C3%B3gicas%20-%20Estructuras%20selectivas.pdf)

[expresiones aritmeticas](https://virtual.unju.edu.ar/pluginfile.php/616629/mod_resource/content/5/04%20Expresiones%20aritmeticas.pdf)