**Fundamentos de la Programación Orientada a Objetos**

AyN: Martínez Sebastián Luciano

DNI: 45.976.797

LU: TUV000621

Docente: Ariel Alejandro Vega

Año: 2024

**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:

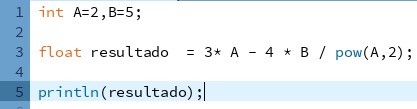
(3\*A)-(4\*B/(A^2))

6-(4\*B/4)

6-5

1

Captura de Processing

  
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

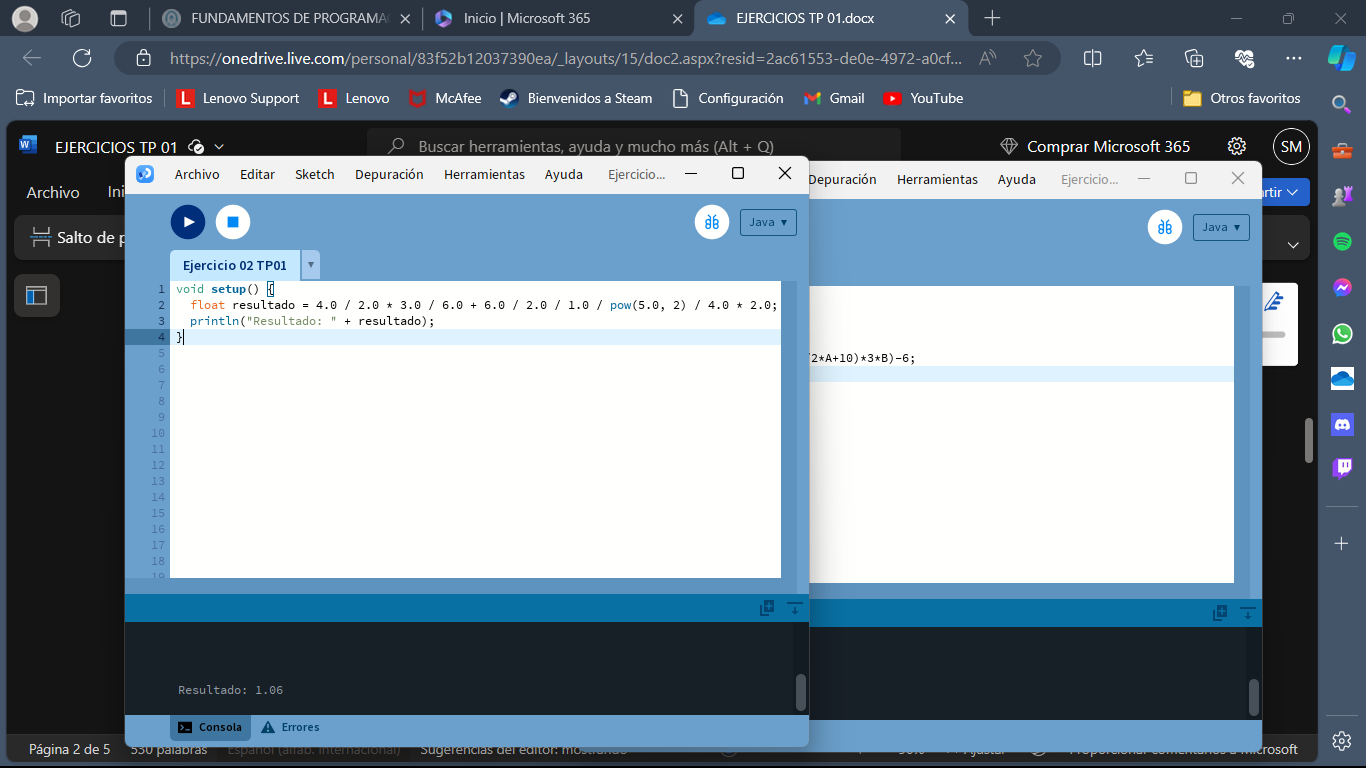
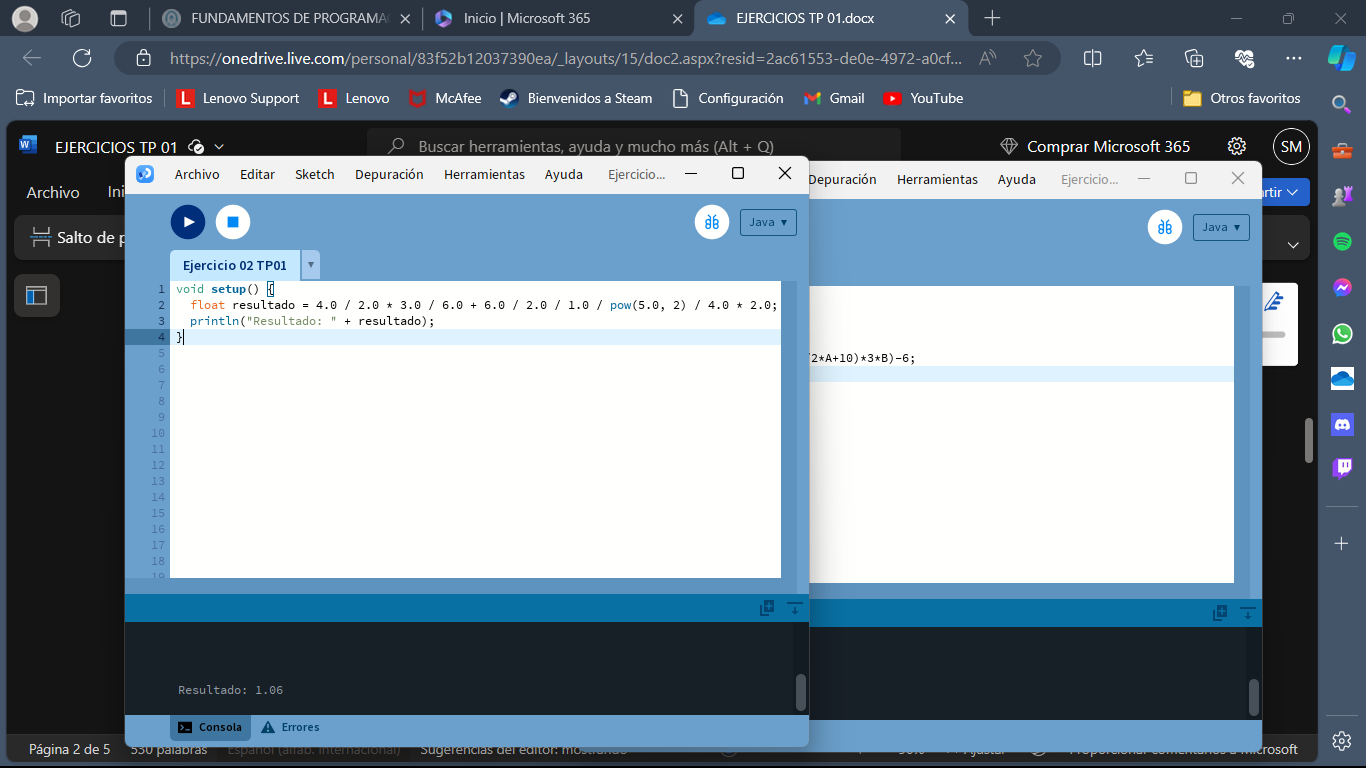
Resolución en Word:

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

1 + 6 / 2 / 1 / 25 / 8

1 + 0,06

**1,06**



Ejercicio 3: Escribir las siguientes expresiones algebraicas como expresiones algorítmicas (en su forma aritmética dentro del algoritmo). En este caso no se pide evaluarlas ni programarlas.

Ejercicio 4: Evaluar las siguientes expresiones aritméticas, para lo cual indicar en el caso de las variables, el valor indicado. **Luego escribirlas como expresiones algebraicas.**

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

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

a)𝑏2 − 4. 𝑎. 𝑐

b)3x4-5x3+x12-17

c)

d)(x2+y2)½

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

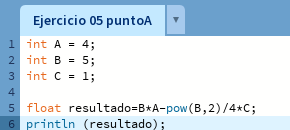
5 \* 4 – 5 ^ 2 / 4 \* 1

20 – 25 / 4 \* 1

20 – 6,25 \* 1

20 – 6,25

**13,75**

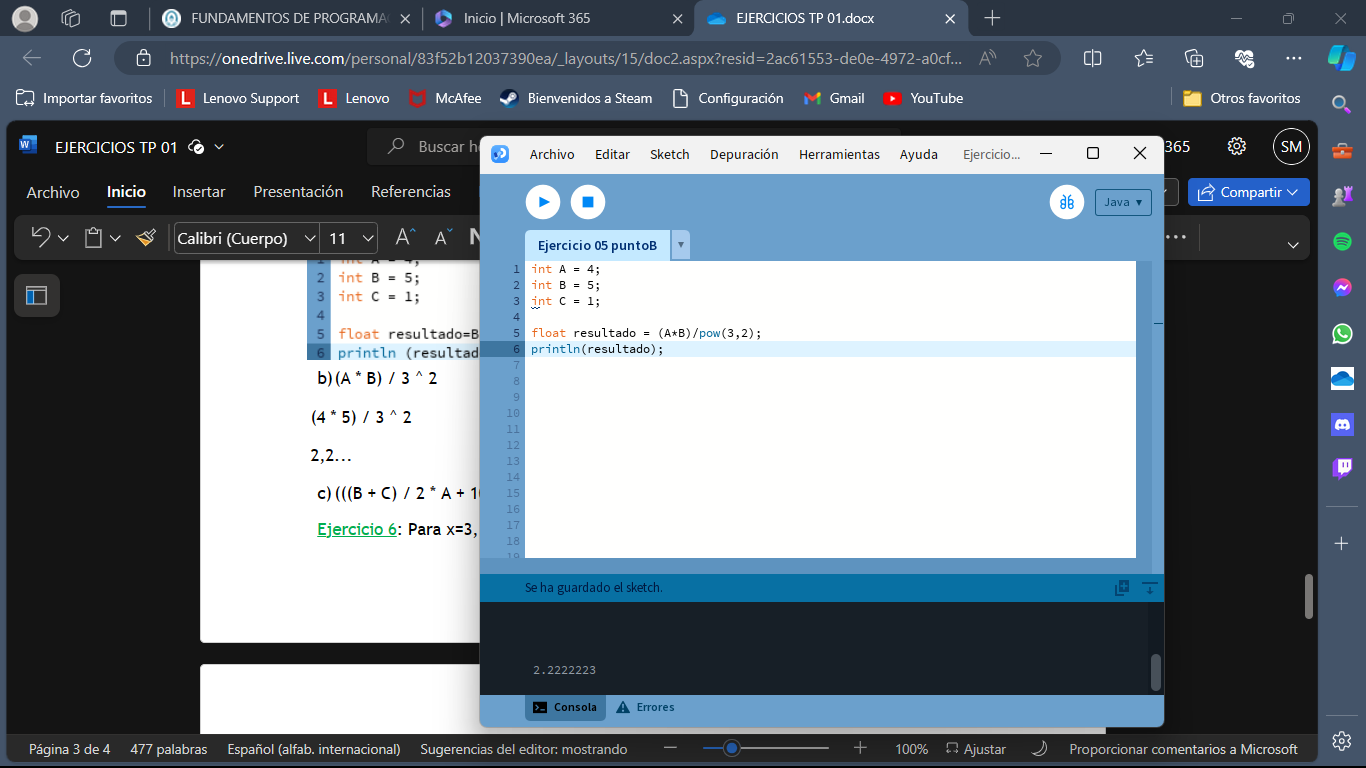
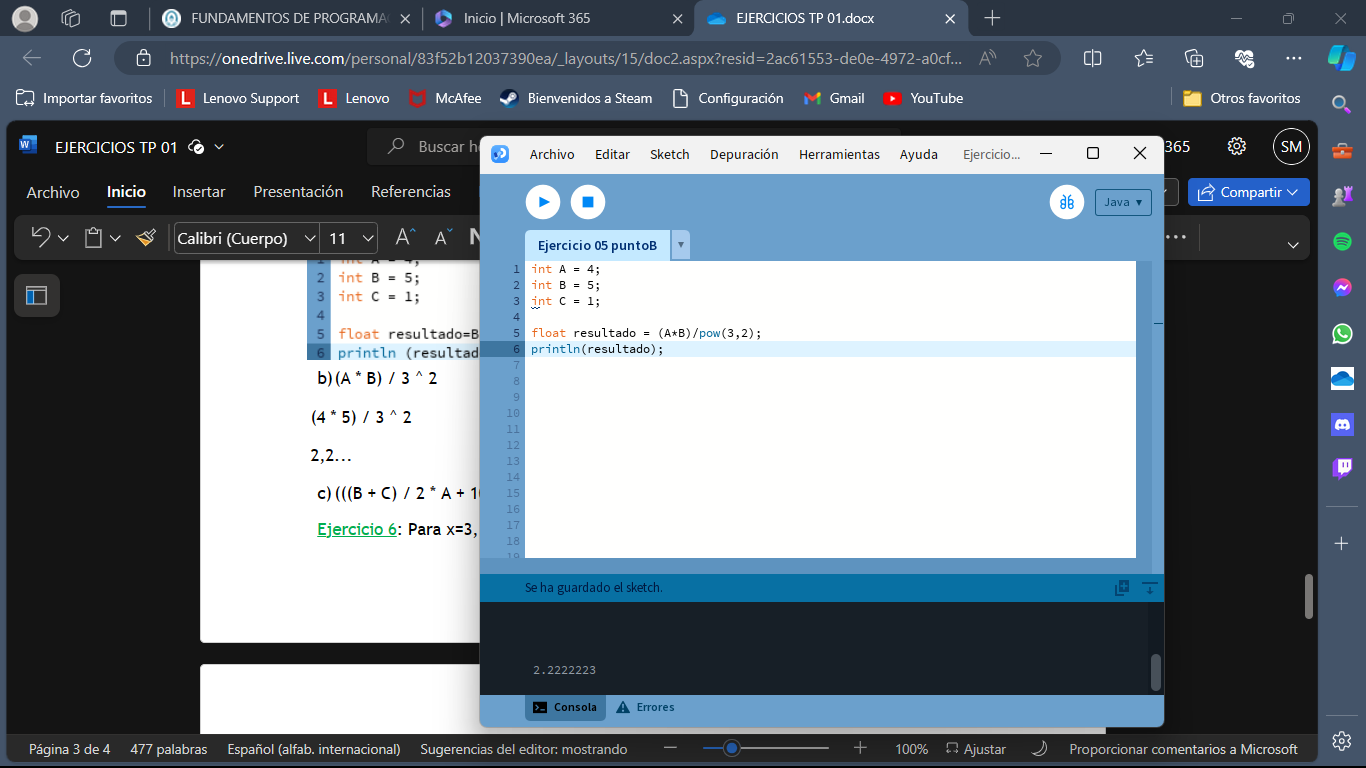




b) (A \* B) / 3 ^ 2

(4 \* 5) / 3 ^ 2

**2,2...**



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

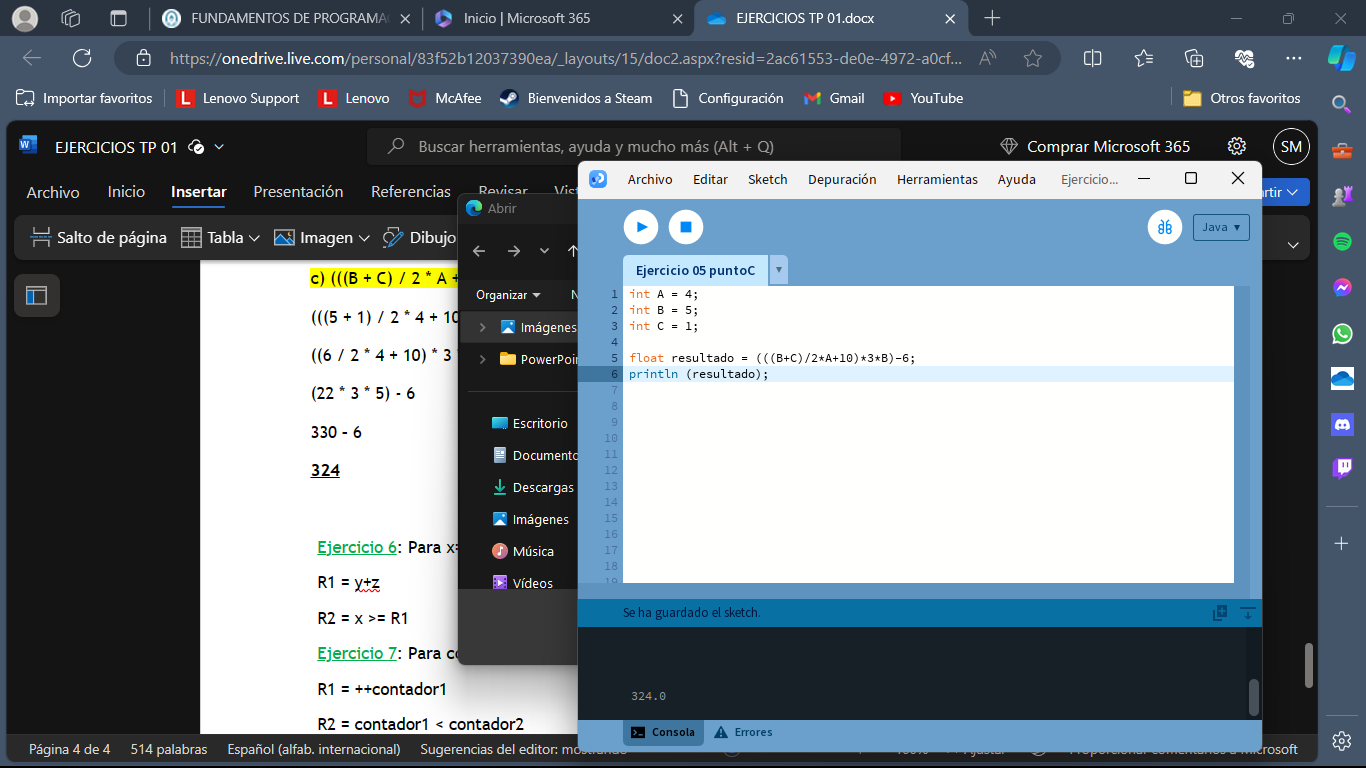
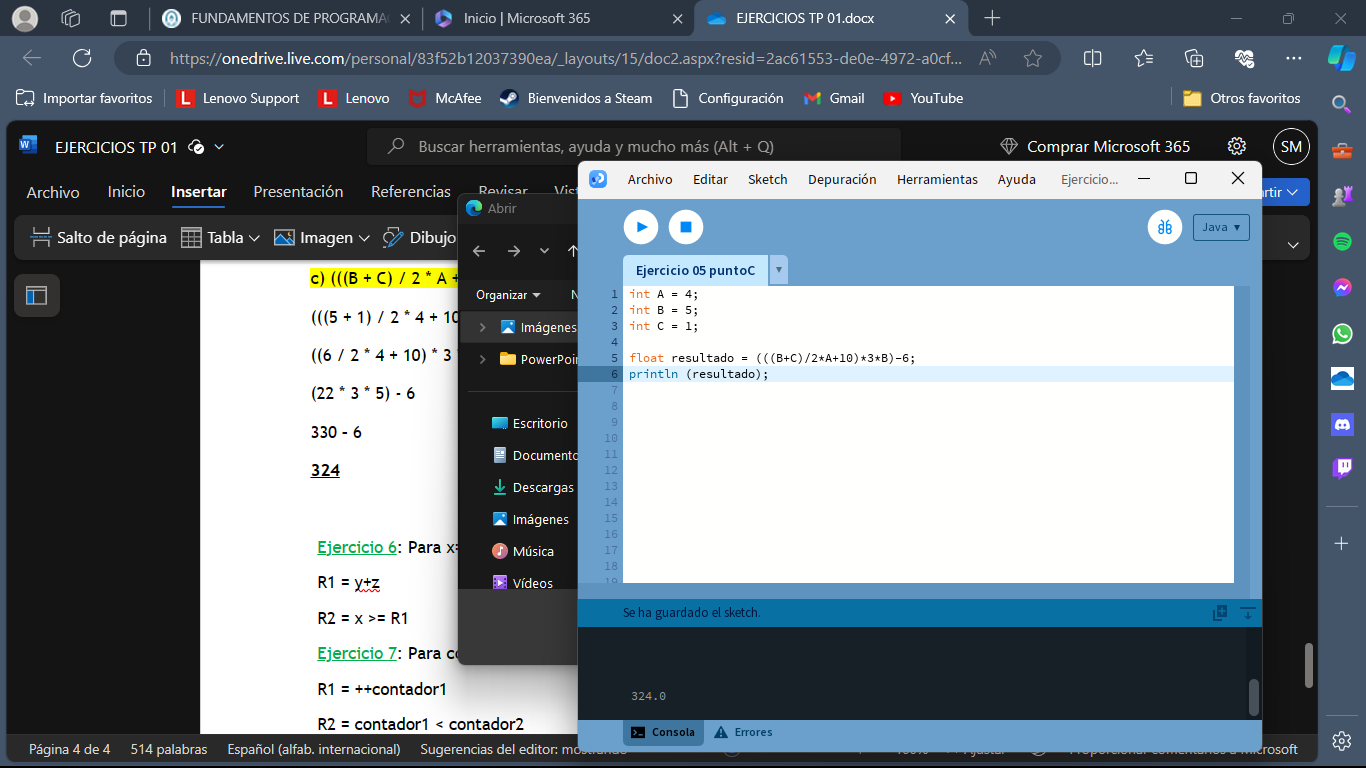
(((5 + 1) / 2 \* 4 + 10) \* 3 \* 5) - 6

((6 / 2 \* 4 + 10) \* 3 \* 5) - 6

(22 \* 3 \* 5) - 6

330 – 6

**324**



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

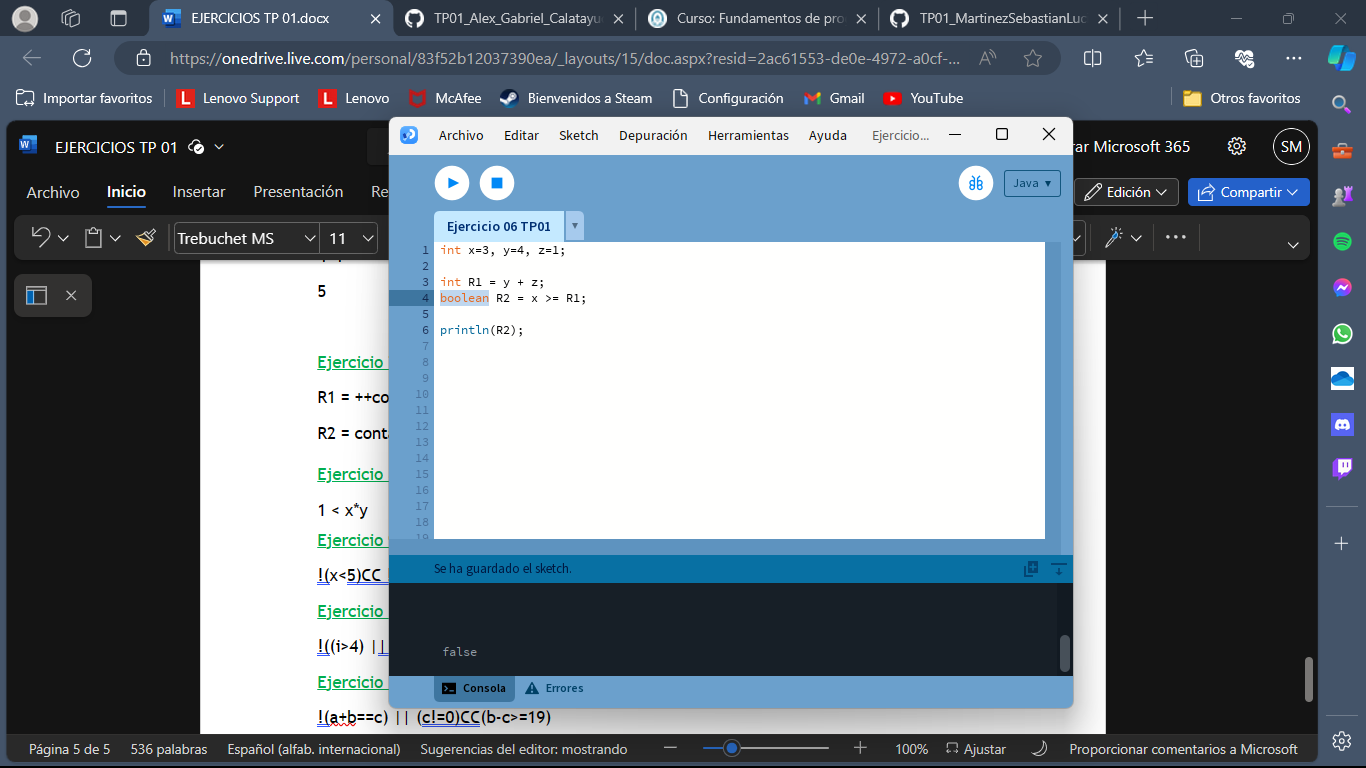
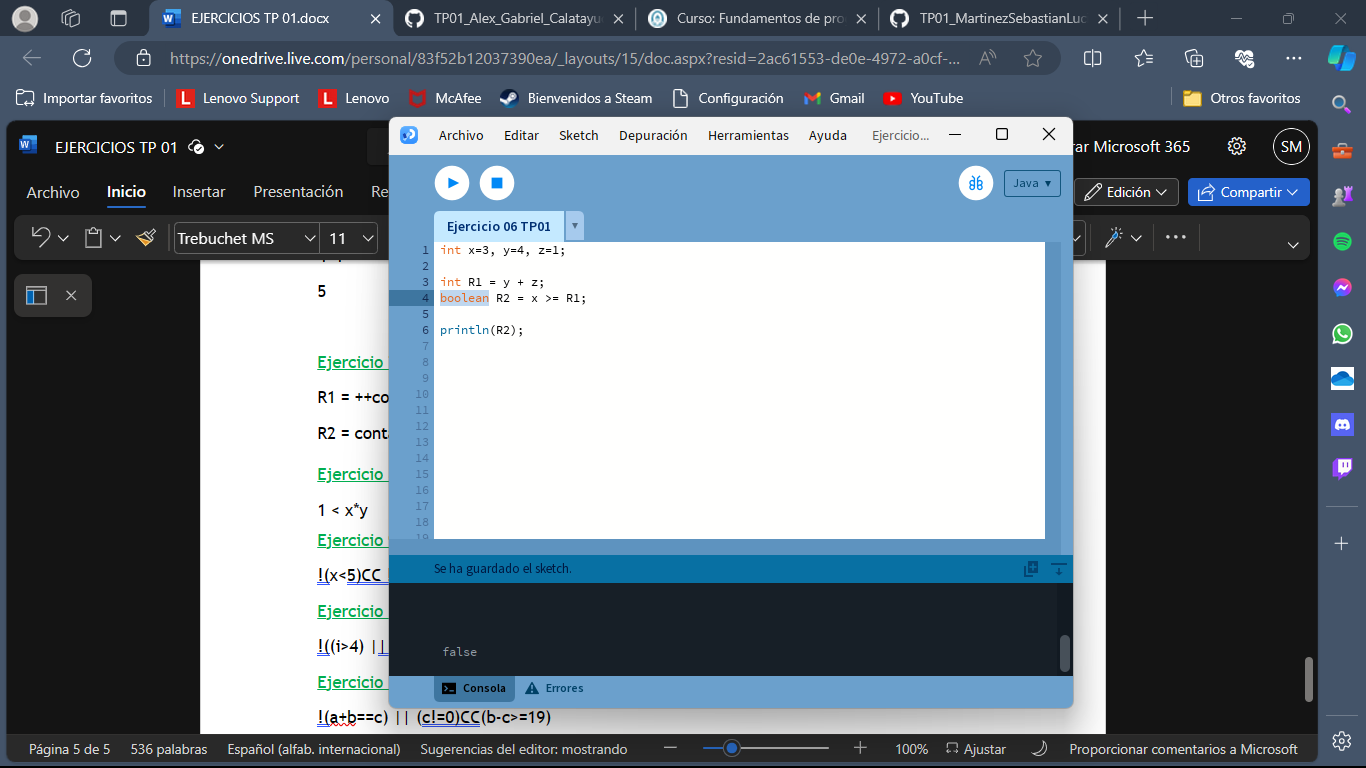
R1 = y+z

R2 = x >= R1

R1=y+z R2=x>=R1

4+1 3>=5

5 F



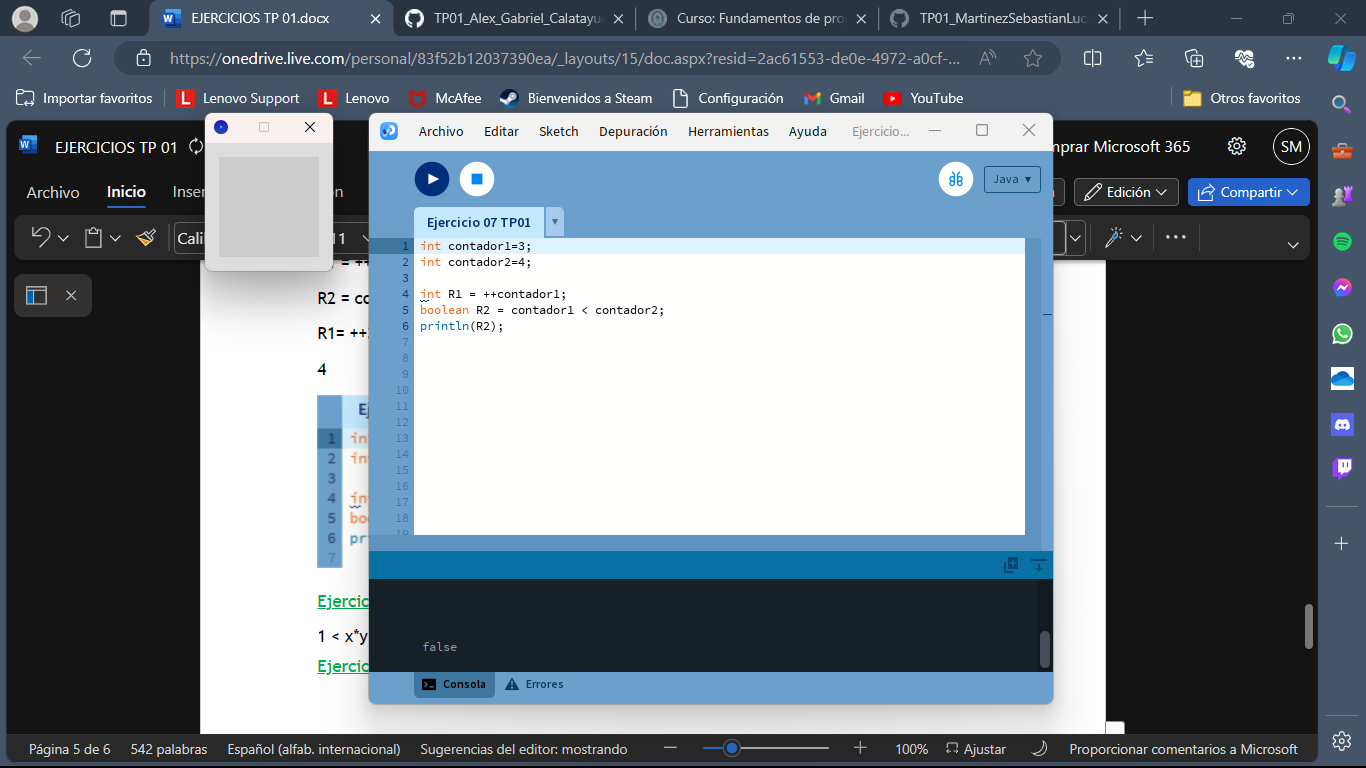
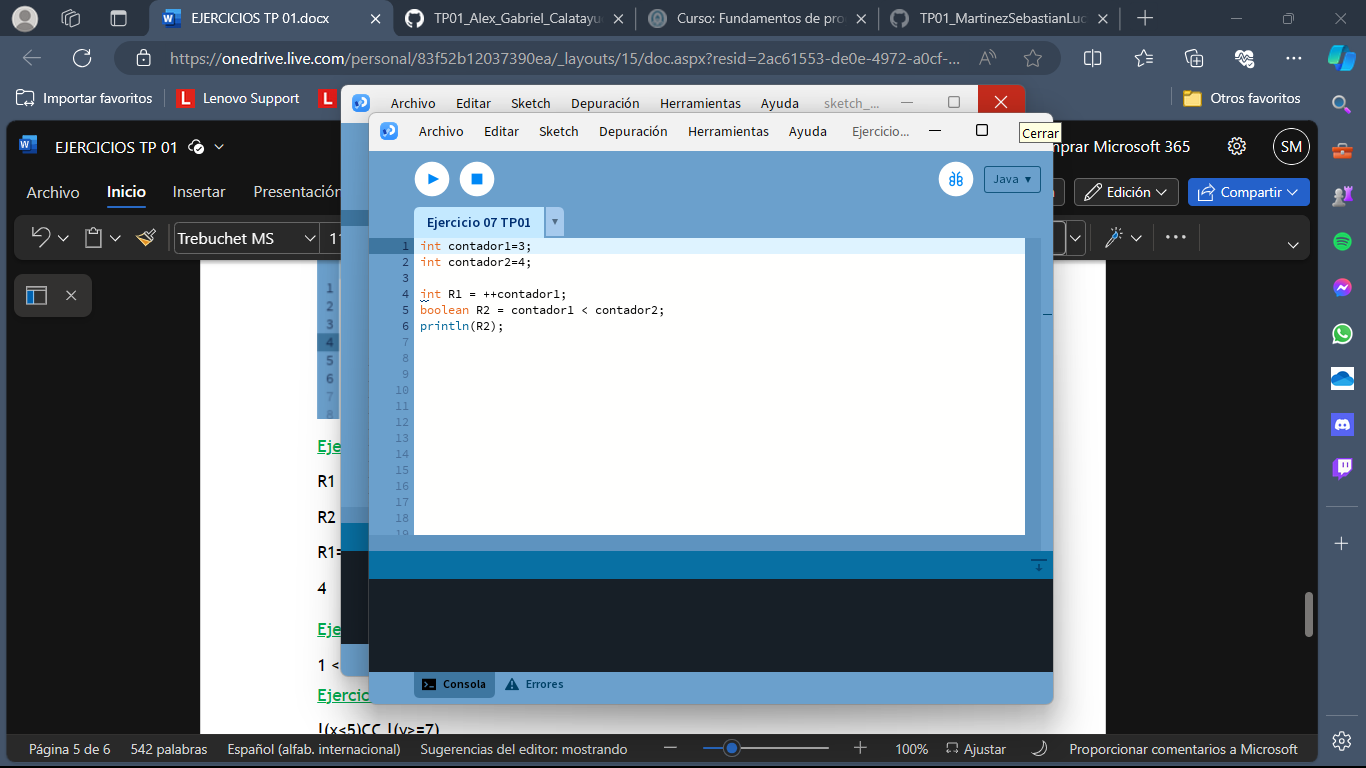
Ejercicio 7: Para contador1=3, contador3=4, evaluar el resultado de

R1 = ++contador1

R2 = contador1 < contador2

R1= ++3 R2= 3<4

4 F

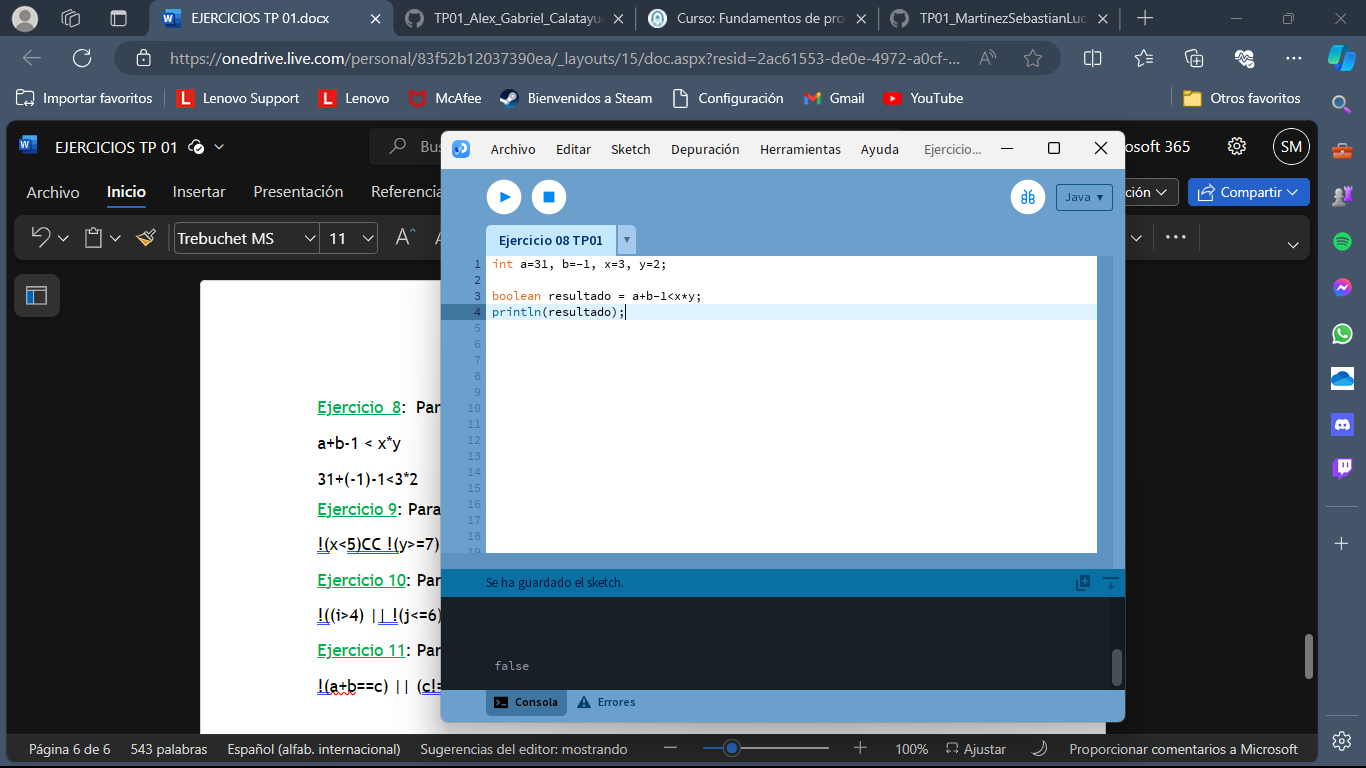
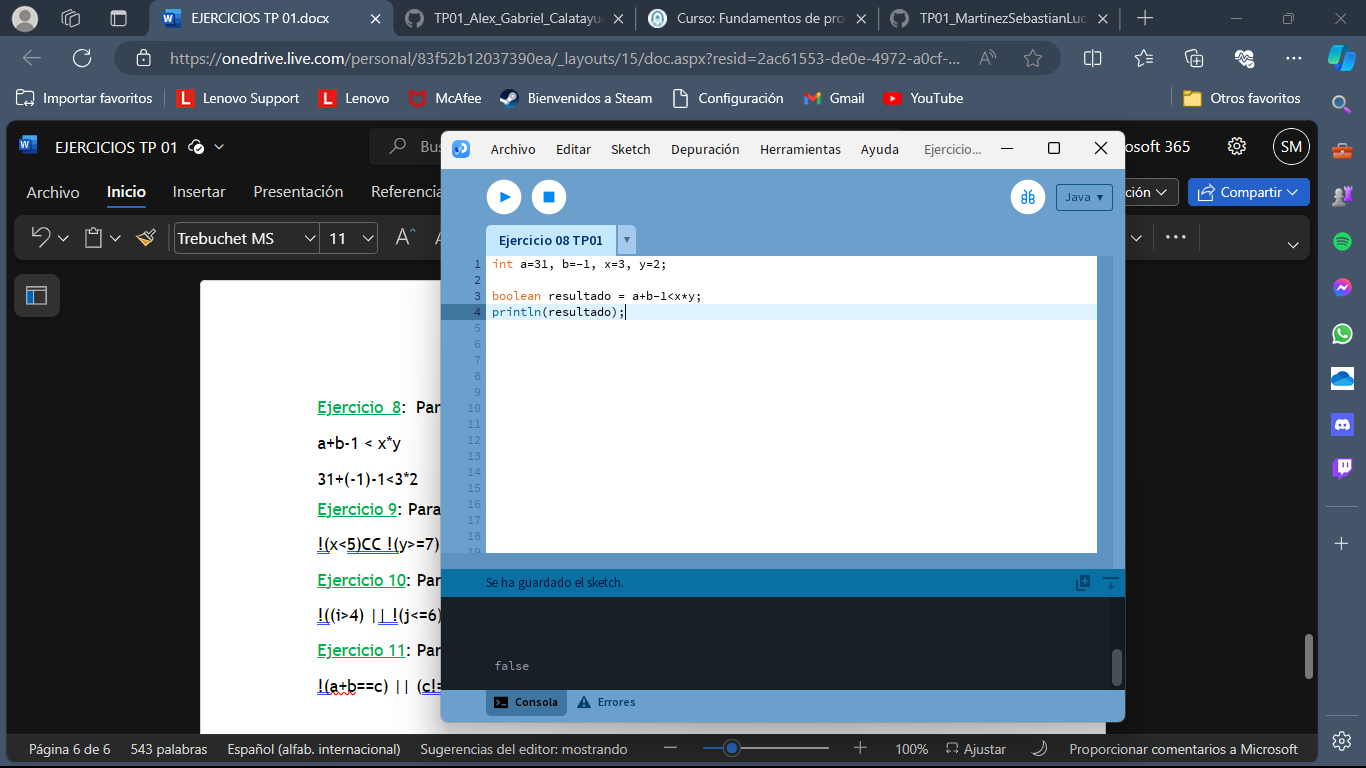


Ejercicio 8: Para a=31, b=-1; x=3, y=2, evaluar el resultado de a+b-1 < x\*y

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

29<6

F



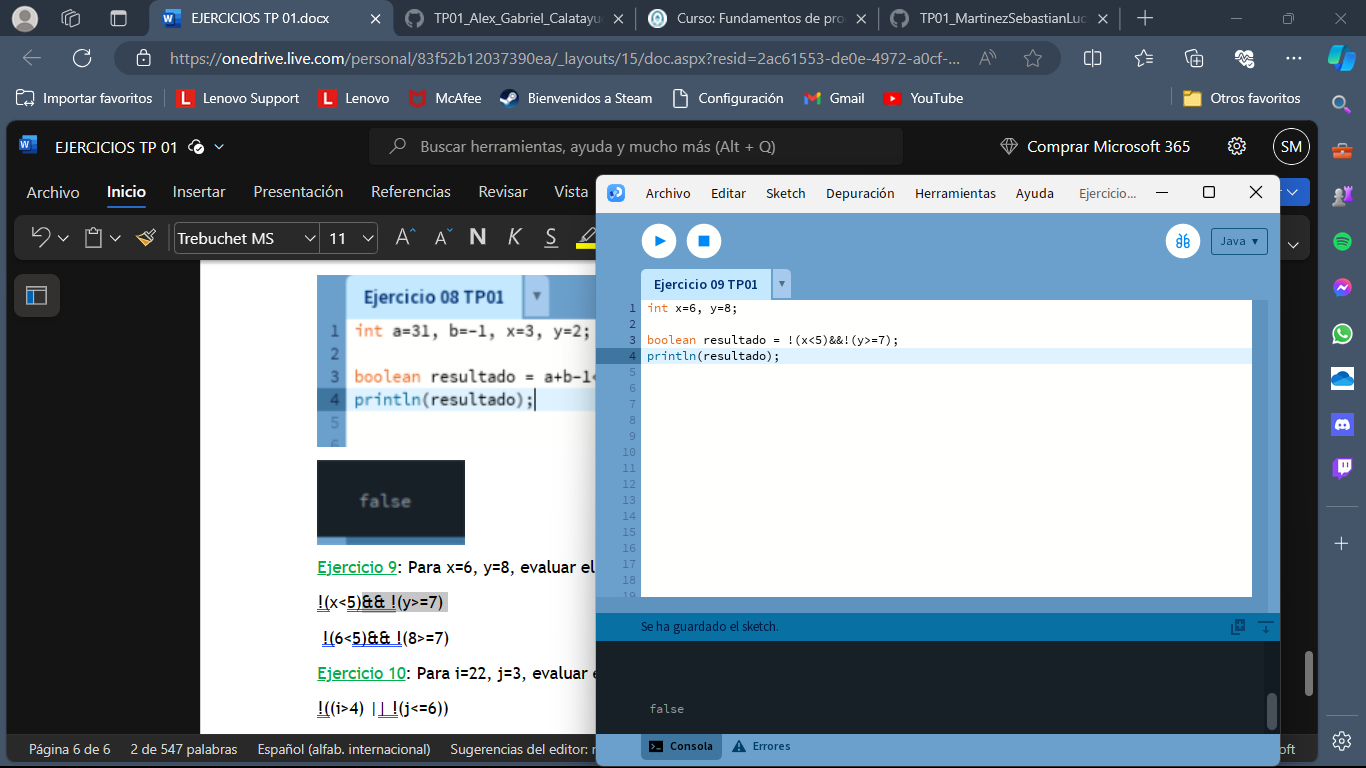
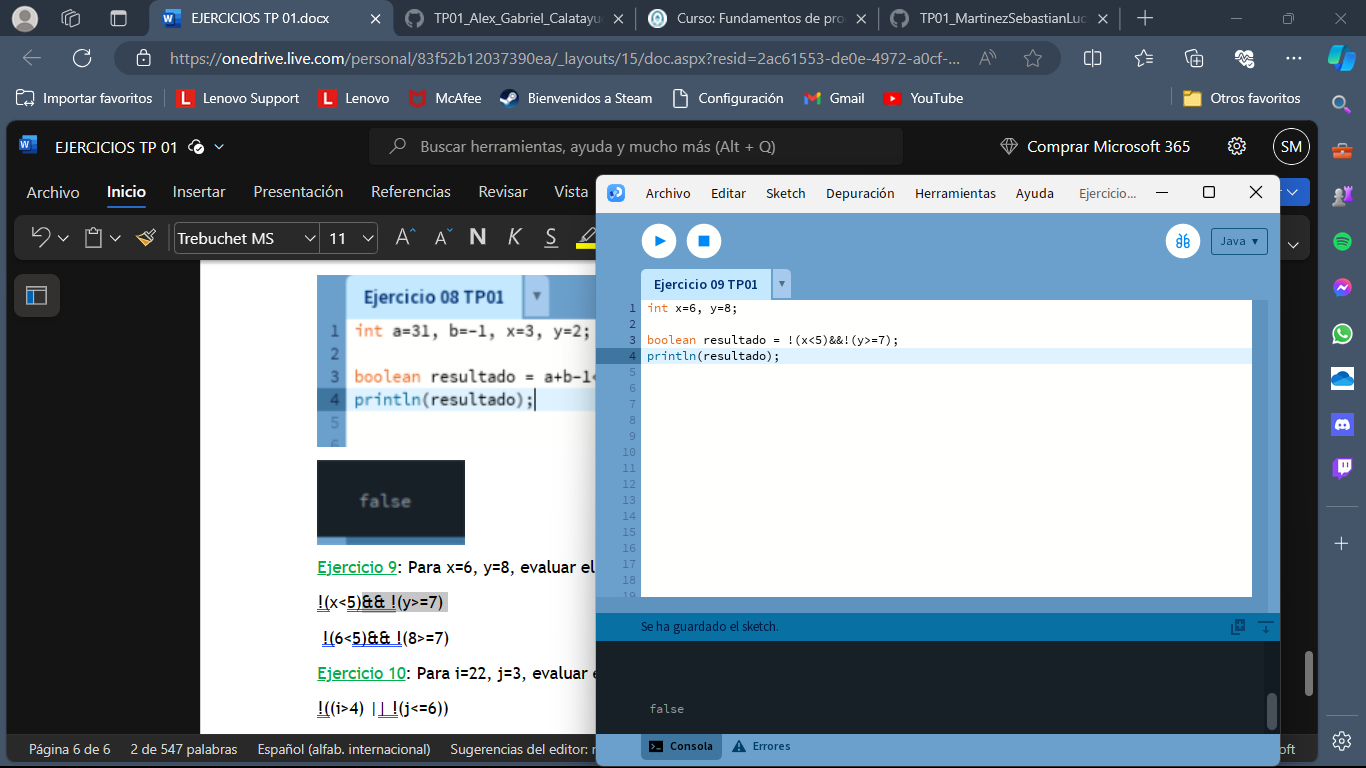
Ejercicio 9: Para x=6, y=8, evaluar el resultado de

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

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

T && F

F



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

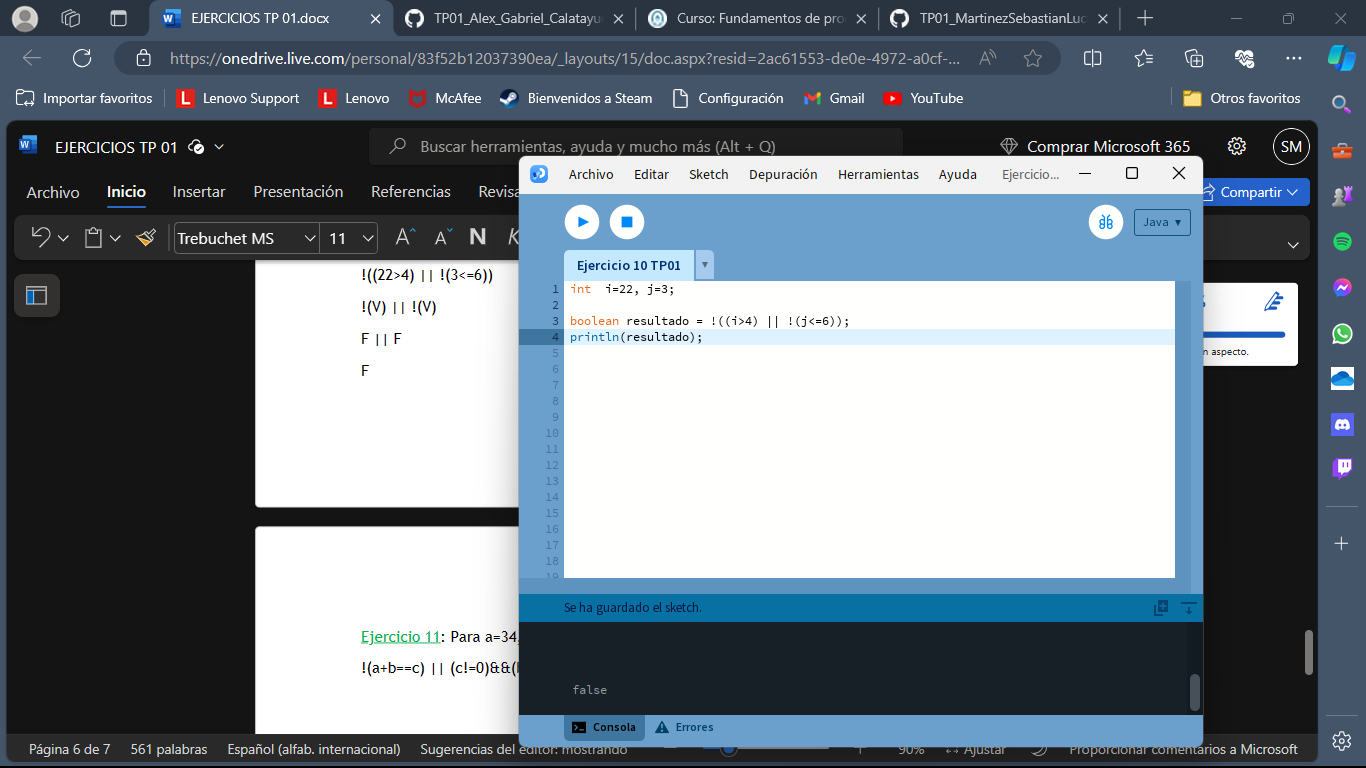
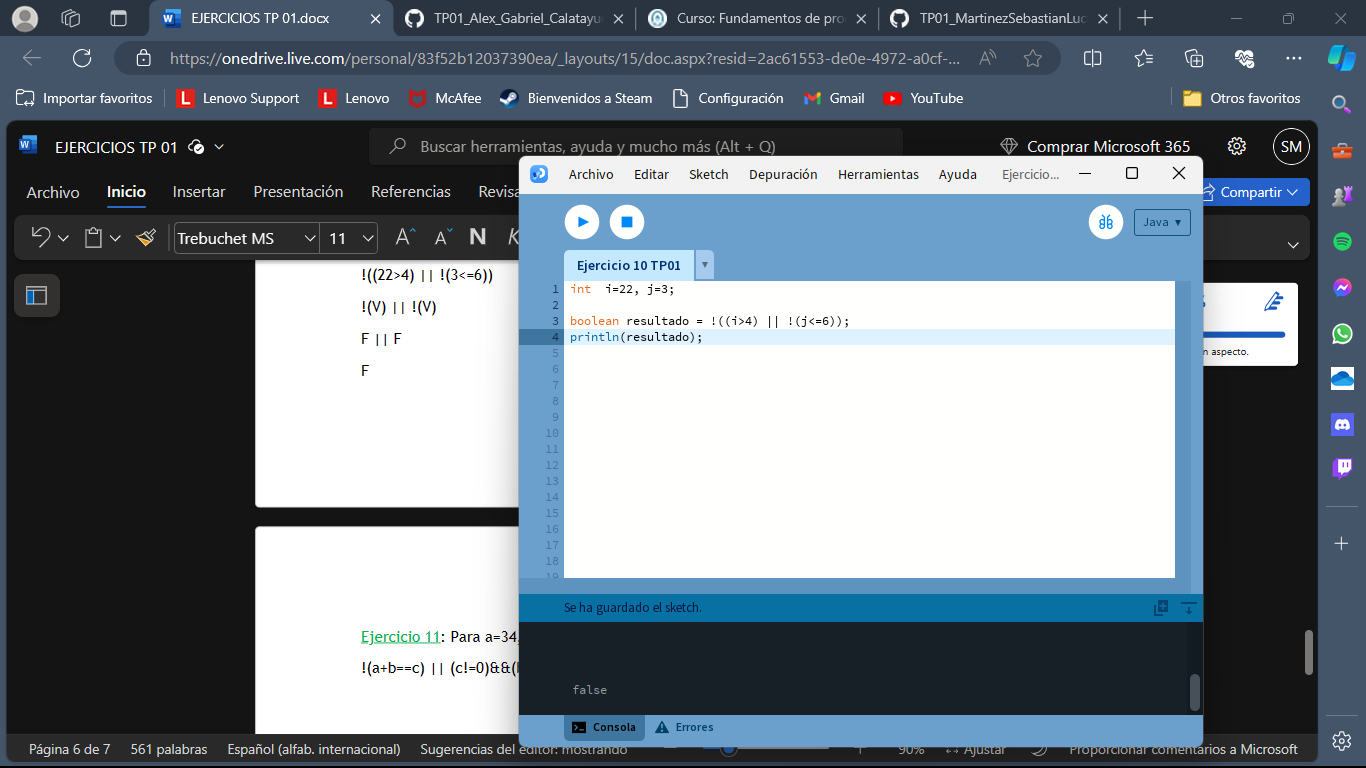
!((i>4) || !(j<=6))

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

!(V) || !(V)

F || F

F



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

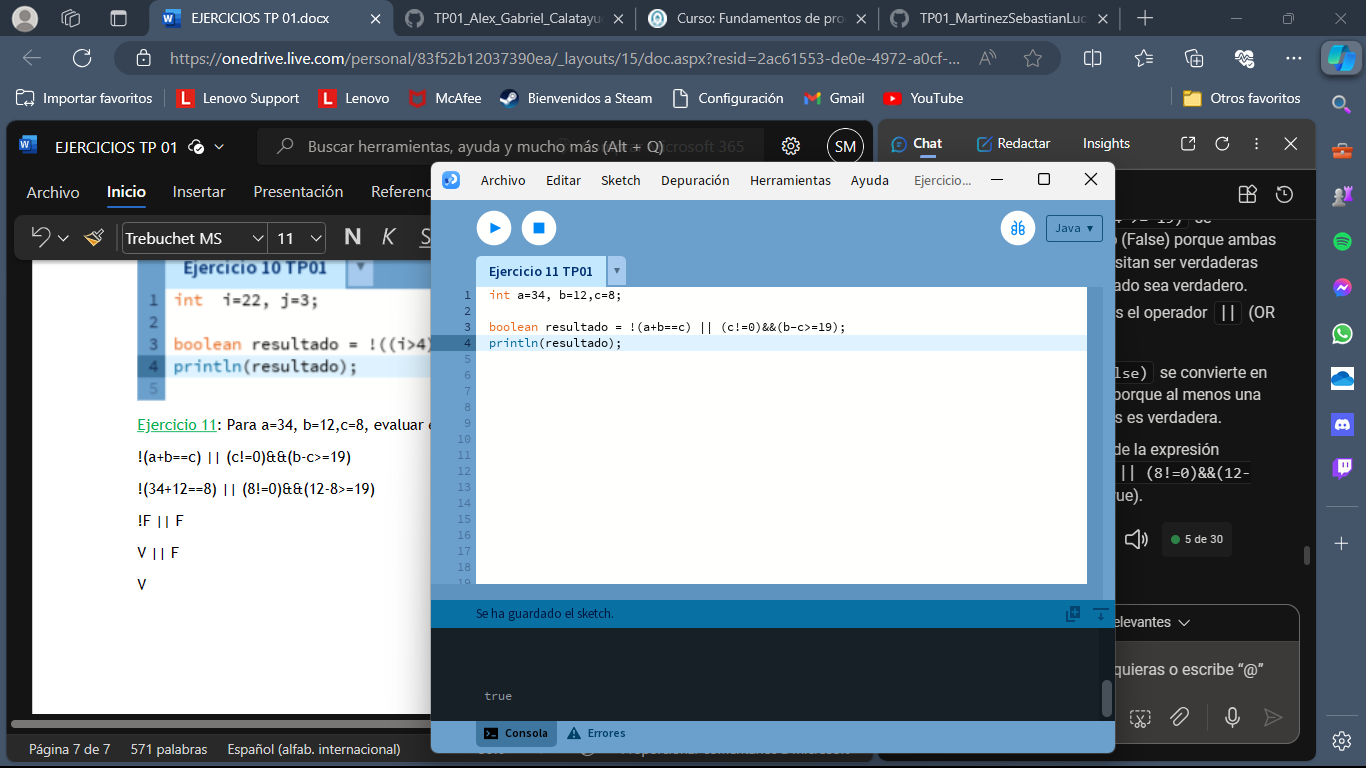
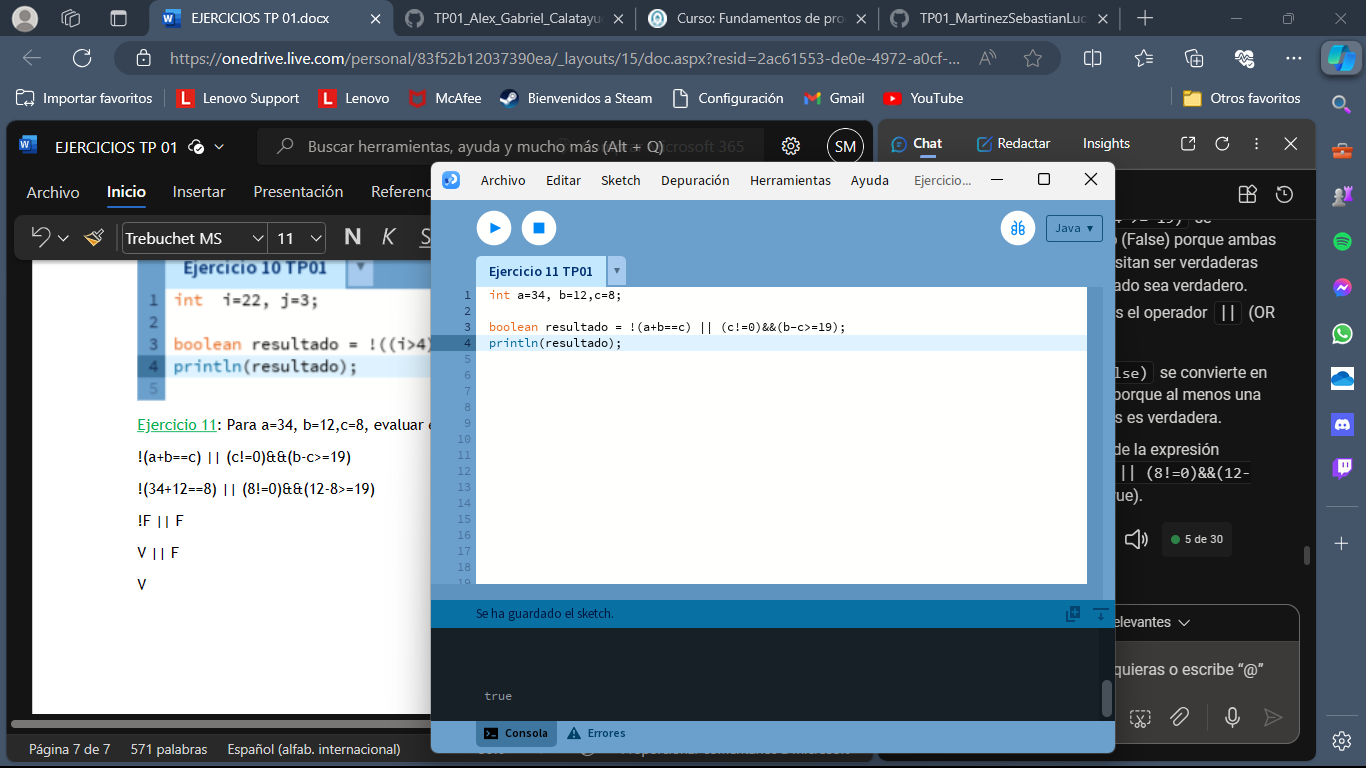
!(a+b==c) || (c!=0)&&(b-c>=19)

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

!F || F

T || F

T



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: El usuario deberá poner su nombre y posteriormente que aparezca en la pantalla el nombre del usuario escrito
* Análisis:

Datos de Entrada:

nombre // se almacenará el nombre del usuario

Datos de Salida:

saludo // se almacenará el saludo al usuario

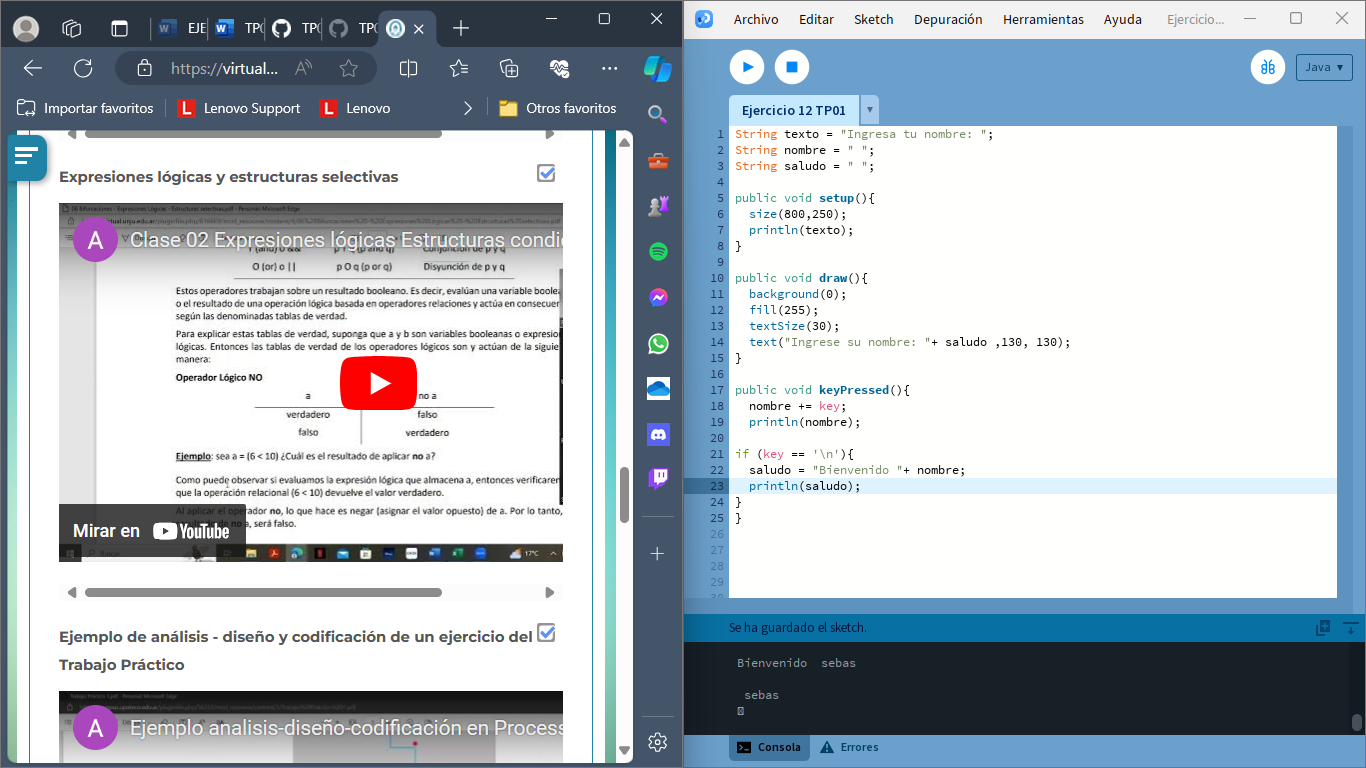
Proceso:

¿Quién debe realizar el proceso?: El programa

¿Cuál es el proceso que resuelve?: el usuario ingresa su nombre y el programa hace que aparezca en la pantalla

Fase de Diseño:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA: Programa** |
| **VARIABLES:**  nombre: string // se almacenará el nombre del usuario  saludo: string // se almacenará el saludo al usuario |
| **NOMBRE ALGORITMO:** saludar\_usuario  **PROCESO DEL ALGORITMO:**   1. *Leer nombre* 2. *Leer* saludo 3. 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

Fase de Análisis:

* Especificación del problema: Calcular el área de un rectángulo dada su base y su altura
* Análisis:

Datos de Entrada:

base // almacena la base

altura // almacena la altura

Datos de Salida:

perímetro // almacena el perímetro ya calculado del rectángulo

área // almacena el área ya calculada del rectángulo

Proceso:

¿Quién debe realizar el proceso?: Estudiante

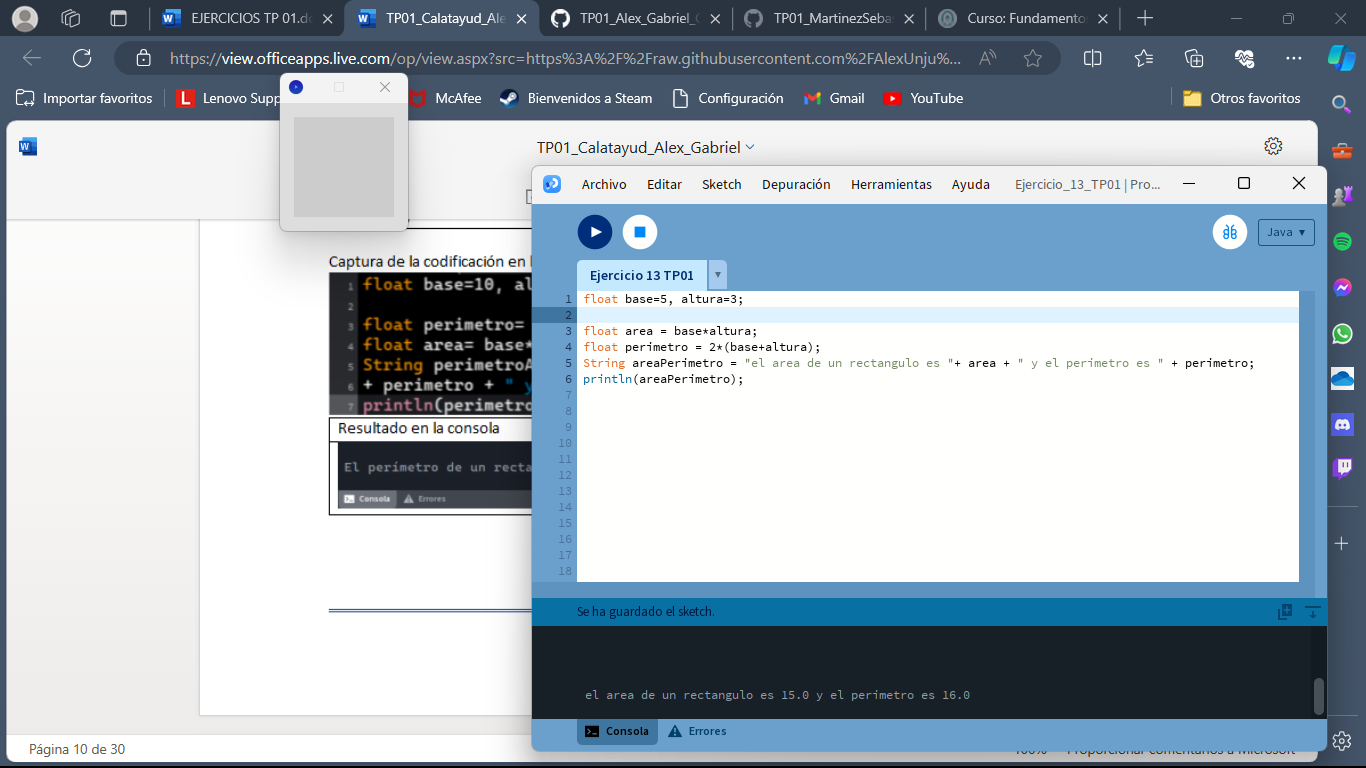
¿Cuál es el proceso que resuelve?:

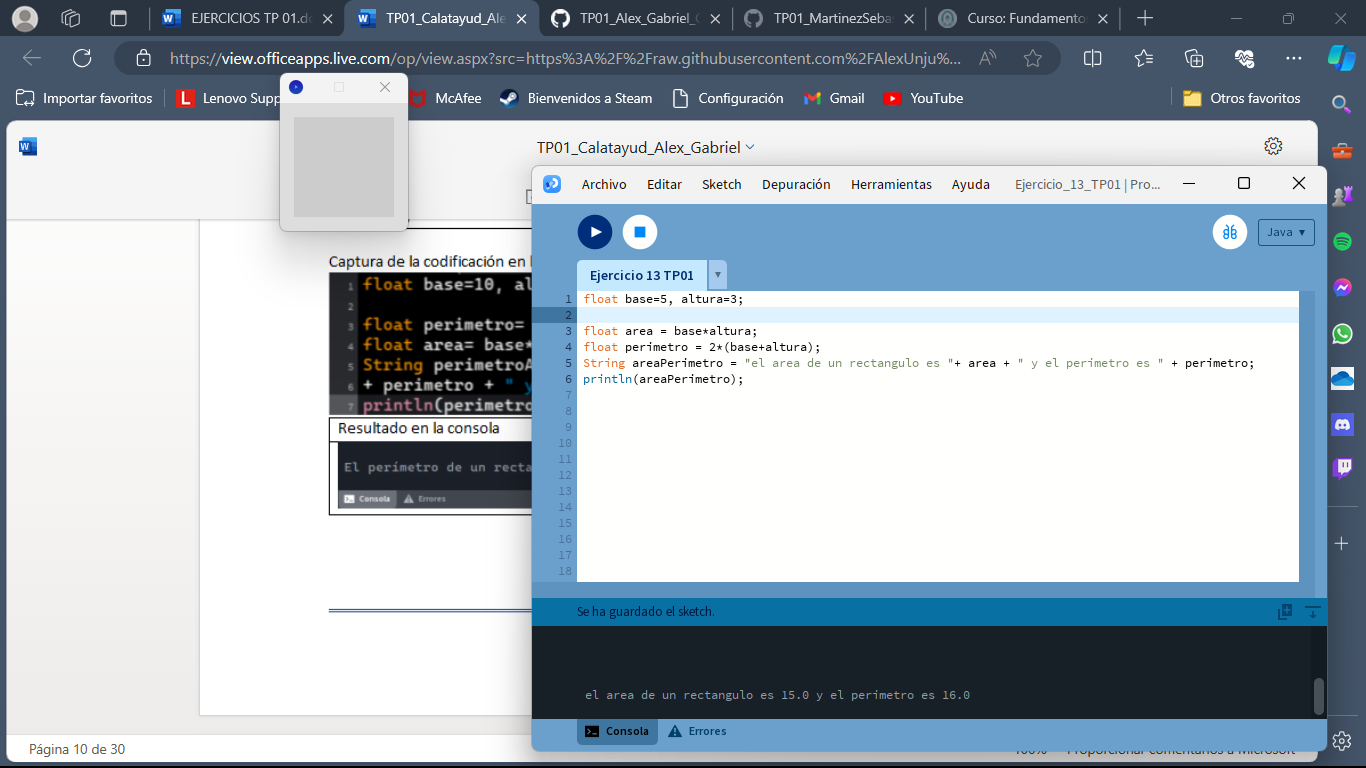
Para calcular el perímetro: 2 x base + altura, osea se debe sumar la longitud de la base del rectángulo, y luego multiplicar el resultado por 2.

Para calcular el área: base x altura, es decir multiplicar la longitud de la base del rectángulo por la altura del rectángulo.

Fase de Diseño:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Estudiante |
| **VARIABLES:**  base: Real  altura: Real  área: Real  perímetro: Real  áreaPerímetro: Real |
| **NOMBRE ALGORITMO:** calcular\_area\_perimetro  **PROCESO DEL ALGORITMO:**   1. *Leer* base 2. *Leer* altura 3. perímetro <-- 2\*(base+altura) 4. área <-- base\*altura 5. áreaPerímetro <-- “El área es “+ área + “y el perímetro es “+ perímetro 6. *Mostrar* áreaPerímetro 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



Fase de Análisis:

* Especificación del problema: Obtener la hipotenusa de un triángulo rectángulo conociendo sus catetos
* Análisis:

Datos de Entrada:

ca // almacena el cateto adyacente

co // almacena el cateto opuesto

Datos de Salida:

h // almacena la hipotenusa

Proceso:

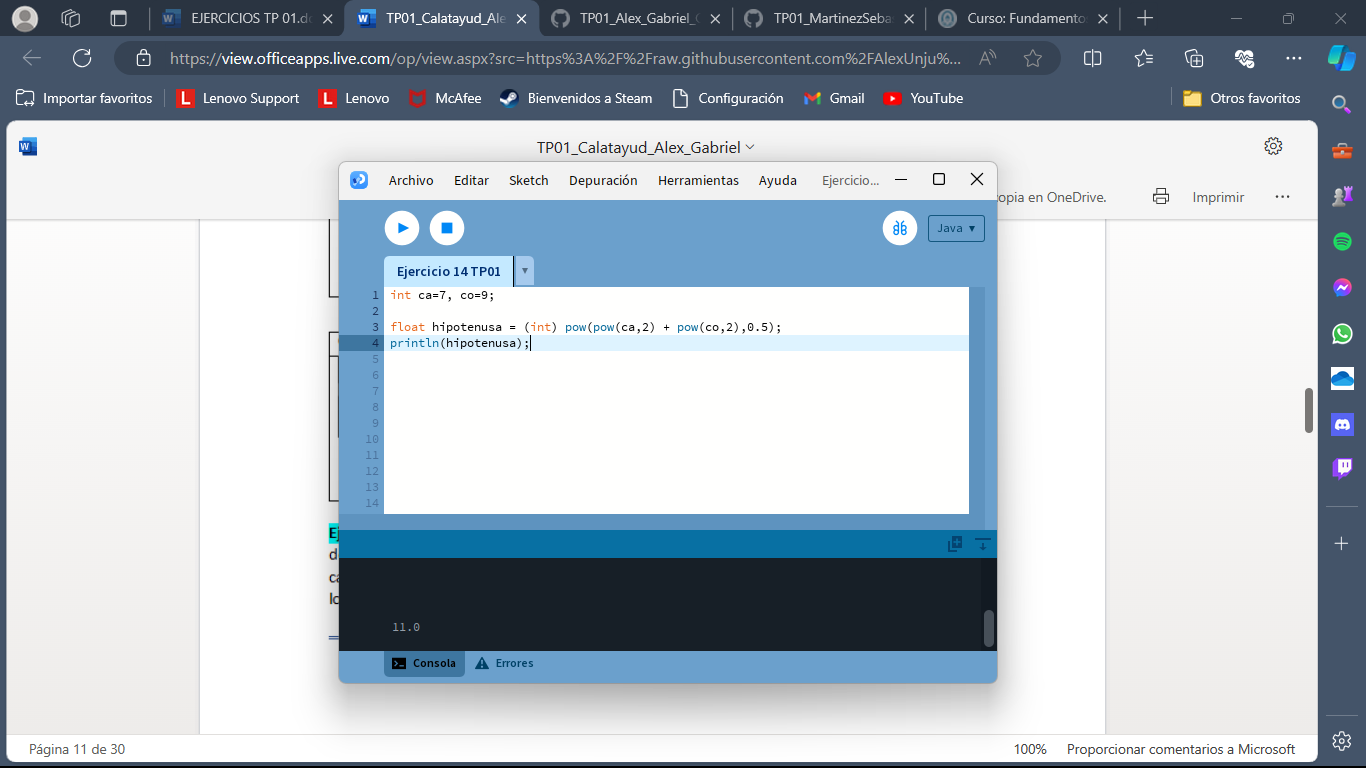
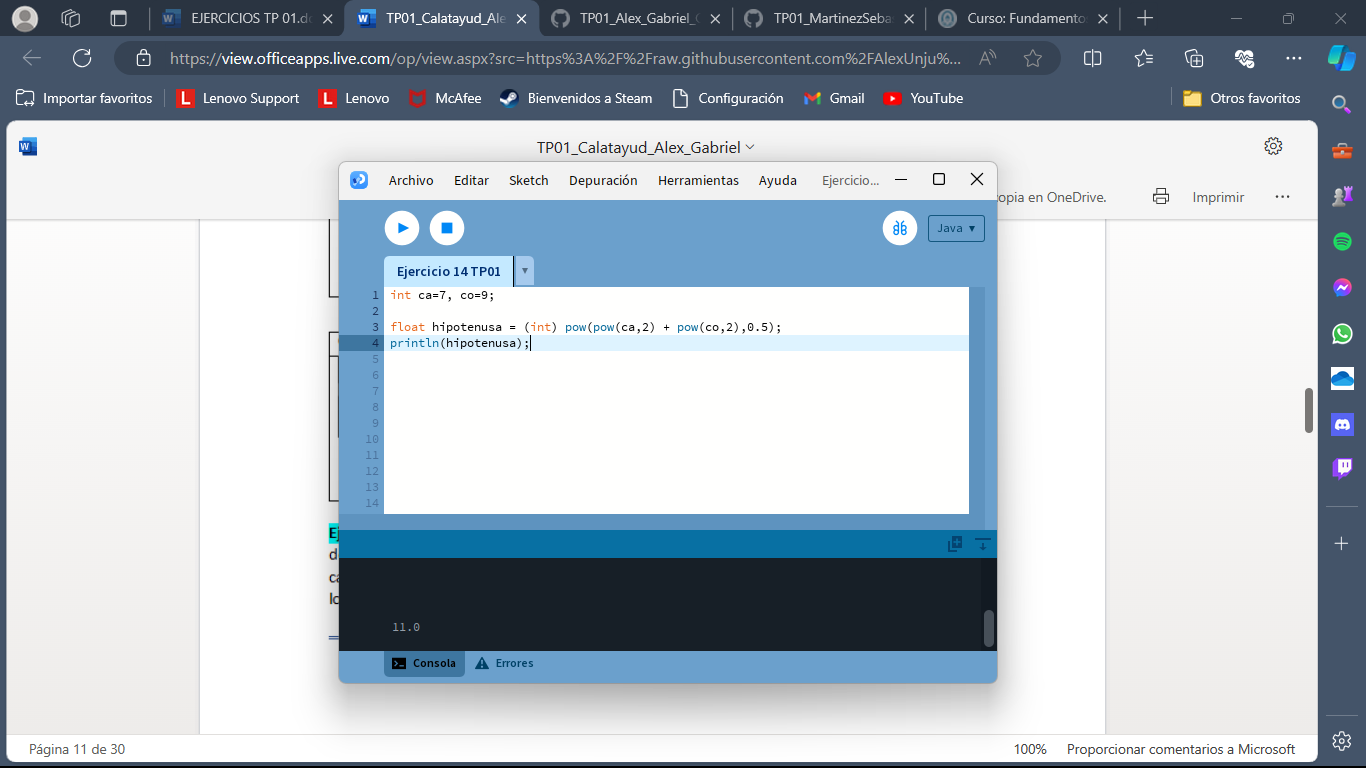
¿Quién debe realizar el proceso?: alumno

¿Cuál es el proceso que resuelve?:

Para encontrar la longitud de la hipotenusa, simplemente toma cada uno de los catetos, eleva cada uno al cuadrado, suma los resultados y luego toma la raíz cuadrada del resultado.

Fase de Diseño:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** El Alumno |
| **VARIABLES:**  ca: Entero // almacena el valor del cateto adyacente  co: Entero // almacena el valor del cateto opuesto  h: Entero // almacena el valor de la hipotenusa |
| **NOMBRE ALGORITMO:** calcular\_*hipotenusa*  **PROCESO DEL ALGORITMO:**   1. *Leer* ca 2. *Leer* co 3. h <-- (ca^2 + co^2) ^ 0,5 4. *Mostrar* h 5. *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.

Fase de Análisis:

* Especificación del problema: dado dos números permitir calcular una suma, resta, multiplicación y división
* Análisis:

Datos de Entrada:

numeroA // almacena el primer numero

numeroB // almacena un segundo numero

Datos de Salida:

sumaResuelta // almacena la suma resuelta

restaResuelta // almacena la resta resuelta

multiResuelta // almacena la multiplicación resuelta

divResuelta // almacena la división resuelta

Proceso:

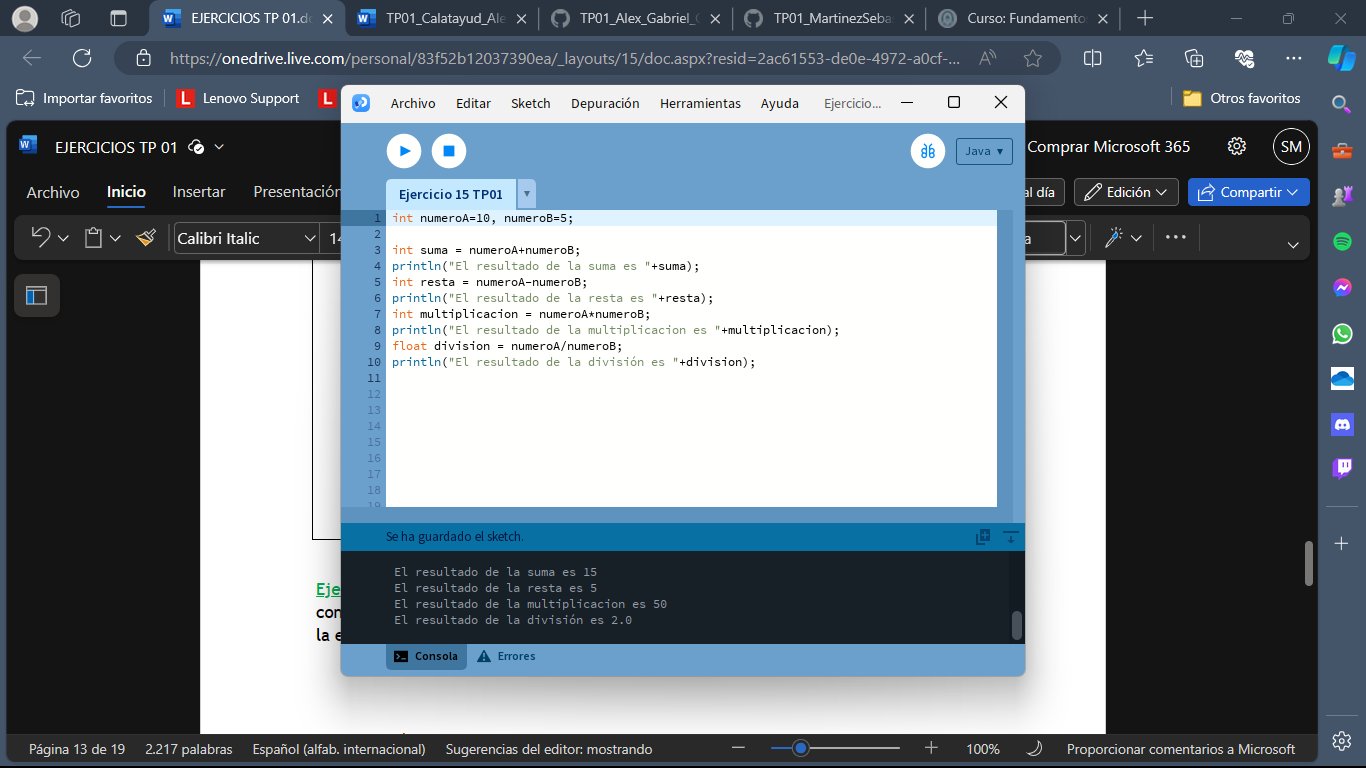
¿Quién debe realizar el proceso?: La Calculadora o Programa

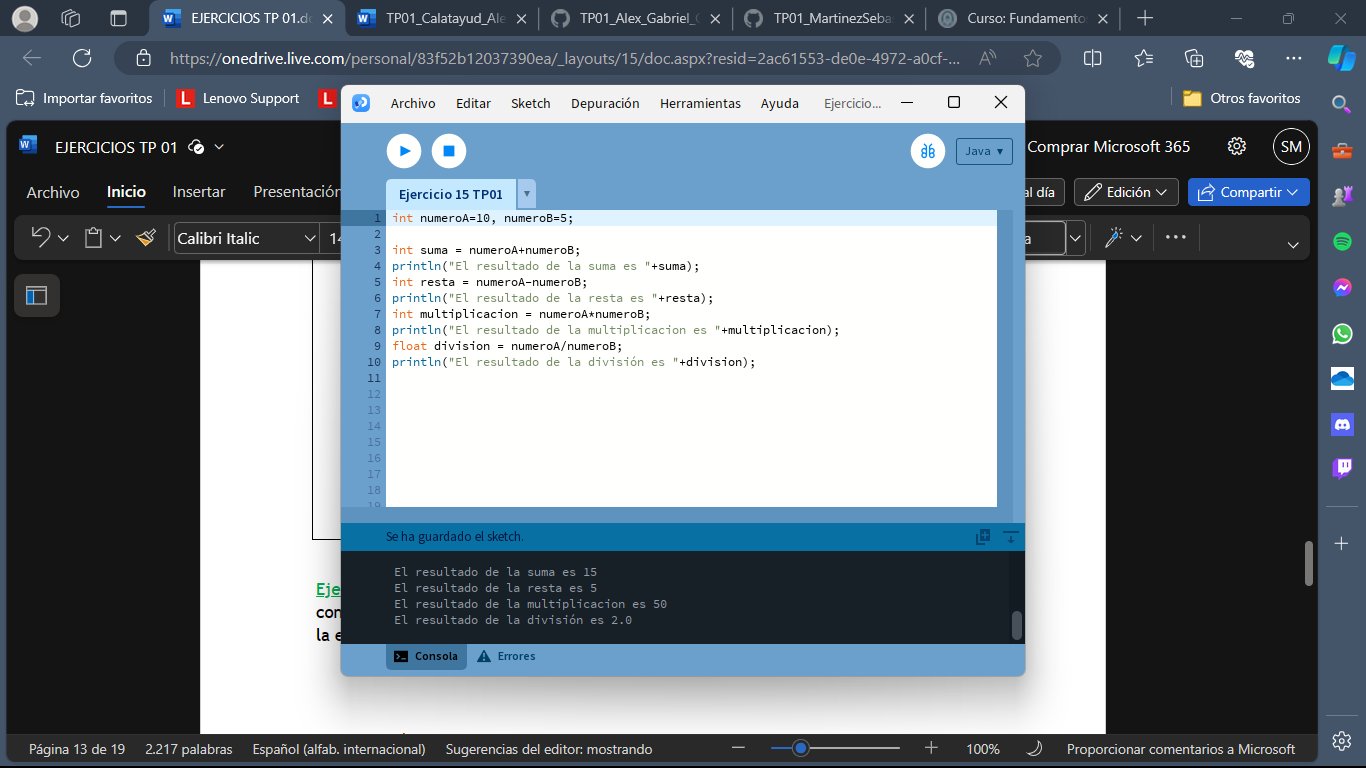
¿Cuál es el proceso que resuelve?:

Se deberá sumar, restar, multiplicar o dividir el numeroA con numeroB según lo que el usuario necesite

Fase de Diseño:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Calculadora o Programa |
| **VARIABLES:**  numeroA = Entero  numeroB = Entero  sumaResuelta = Entero  restaResuelta = Entero  multiResuelta = Entero  divResuelta = Real |
| **NOMBRE ALGORITMO:** calcular\_numeros  **PROCESO DEL ALGORITMO:**   1. *Leer* numeroA 2. *Leer* numeroB 3. sumaResuelta <-- numeroA + numeroB 4. *Mostrar* “El resultado de la suma es: “+sumaResuelta 5. restaResuelta <-- numeroA – numeroB 6. *Mostrar* “El resultado de la resta es: “-restaResuelta 7. multiResuelta <-- numeroA \* numeroB 8. *Mostrar* “El resultado de la multiplicación es: “\*multiResuelta 9. divResuelta <-- numeroA / numeroB 10. *Mostrar* “El resultado de la división es: “/divResuelta 11. *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.



Fase de Análisis:

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

Datos de Entrada:

Temperatura Fahrenheit

Datos de Salida:

Temperatura Celsius

Proceso:

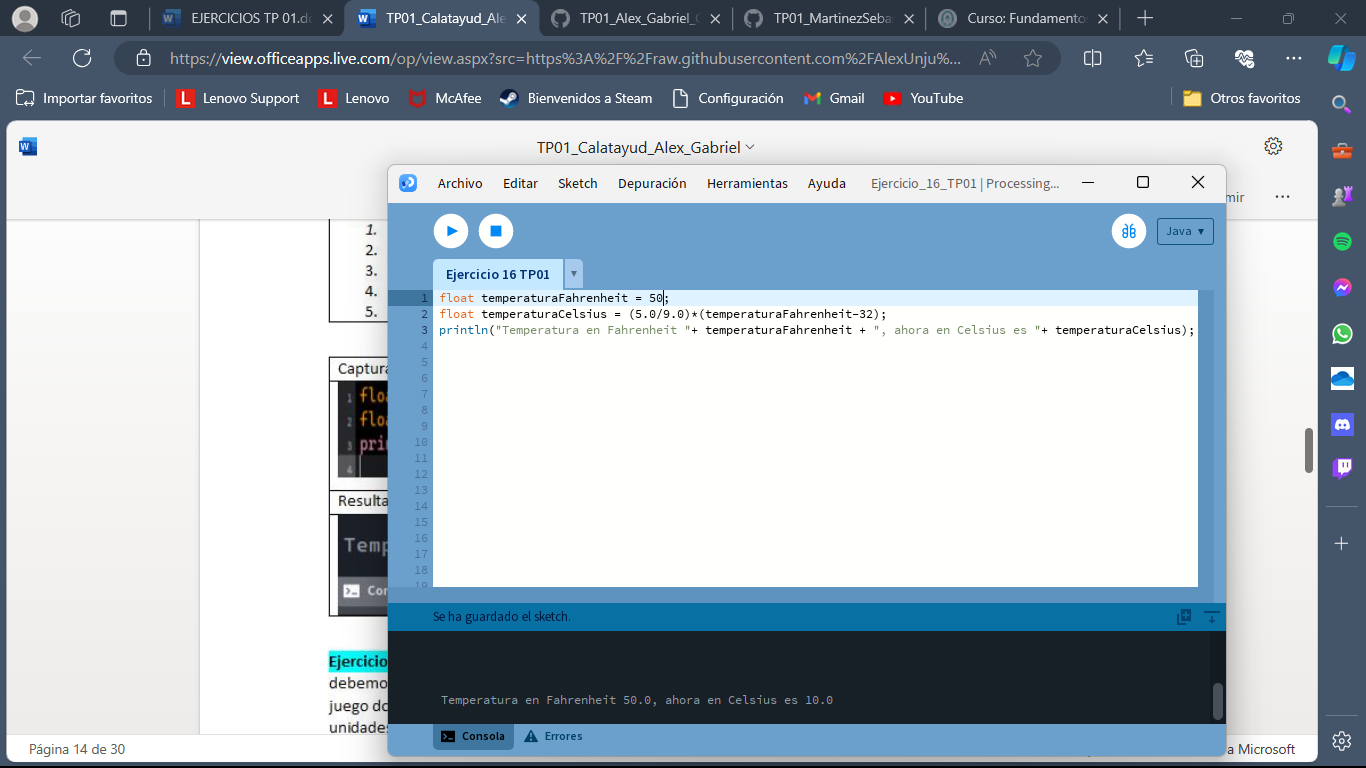
¿Quién debe realizar el proceso?: Un Alumno

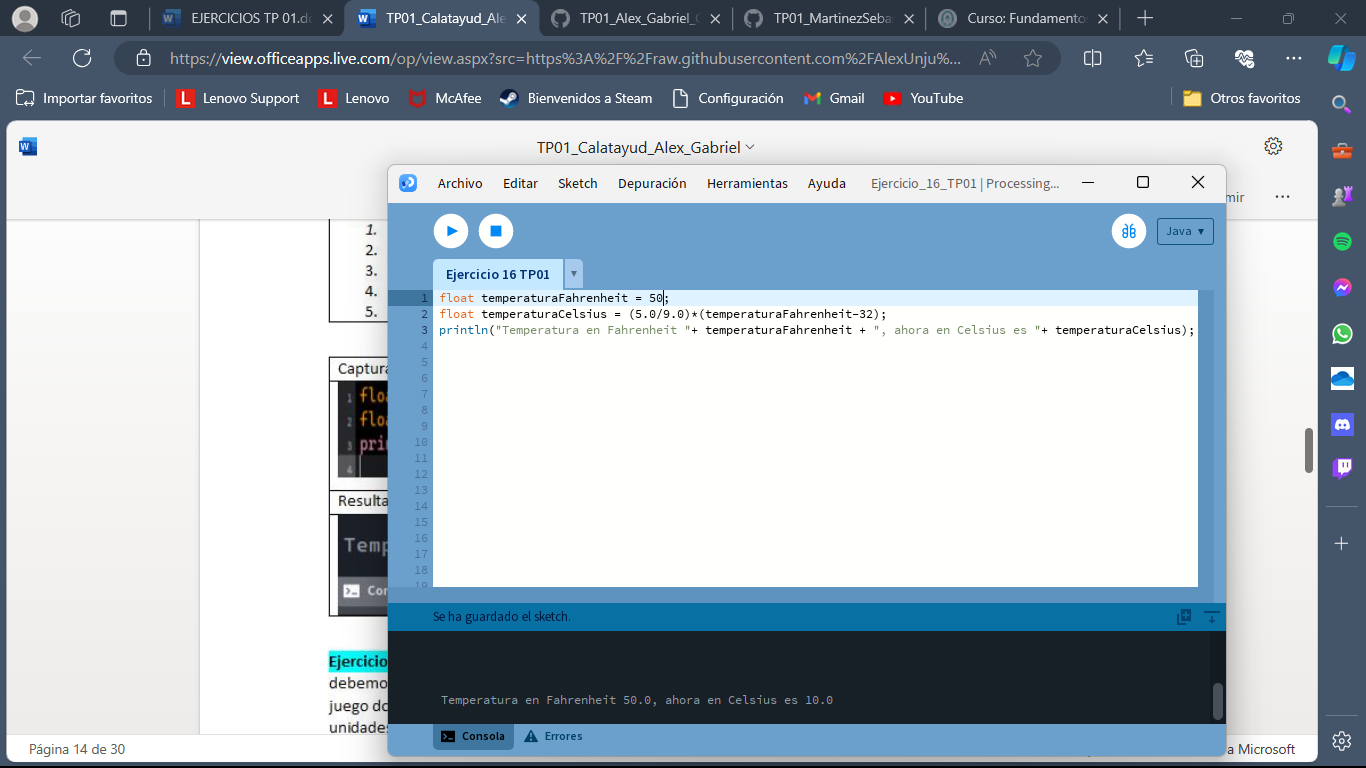
¿Cuál es el proceso que resuelve?:

Restar 32 a la temperatura Fahrenheit, luego multiplicar el resultado por 5/9 y nos daría el resultado de la temperatura Celsius

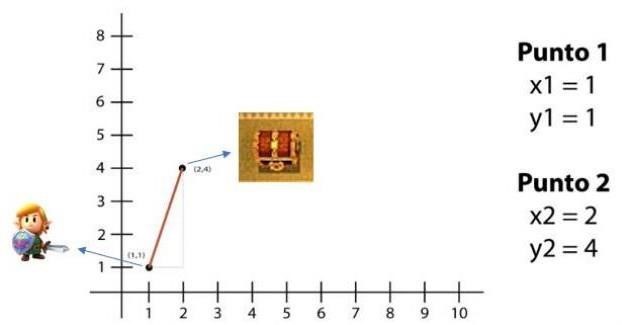
Fase de Diseño:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA: Alumno** |
| **VARIABLES:**  temperaturaFahrenheit: Real// almacena en Fahrenheit  temperaturaCelsius: Real // almacena en Celsius |
| **NOMBRE ALGORITMO:** calcular\_fahrenheit\_a\_celsius  **PROCESO DEL ALGORITMO:**   1. *Leer* temperaturaFahrenheit 2. temperaturaCelsius <-- (5.0/9.0)\*(temperaturaFahrenheit-32) 3. *Mostrar* temperaturaCelsius 4. *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.

Fase de Análisis:

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

Datos de Entrada:

coordenadas de Link

coordenadas del tesoro

Datos de Salida:

Distancia entre Link y el tesoro

Proceso:

¿Quién debe realizar el proceso?: El programa

¿Cuál es el proceso que resuelve?:

Debemos calcular la diferencia entre las coordenadas X,Y y los dos puntos de los catetos.

Fase de Diseño:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** El videojuego |
| **VARIABLES:**  x1 = Real  y1 = Real  x2 = Real  y2 = Real  coordenadasX = Real  coordenadasY = Real  distancia = Real  distanciaTesoro = Real |
| **NOMBRE ALGORITMO:** calcular\_distancia  **PROCESO DEL ALGORITMO:**   1. *Leer* x1 2. *Leer y1* 3. *Leer x2* 4. *Leer y2* 5. distanciaTesoro <-- 50 6. *c*oordenadasX *<-- x2 – x1* 7. coordenadasY *<-- y2 – y1* 8. distancia <-- ((coordenadasX)^2+(coordenadasY )^2) ^2 9. *Mostrar* “La distancia es “+ distancia 10. **si** (distancia=distanciaTesoro)**entonces** 11. *Mostrar* “¡Bonus PowerUp!” 12. *fin\_si* 13. *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.

Fase de Análisis:

* Especificación del problema: Obtener las raíces de una ecuación de segundo grado utilizando la estructura según para el análisis de la discriminante de la ecuación cuadrática.
* Análisis:

Datos de Entrada:

coeficiente a, b y c // almacena el coeficiente para las raíces

Datos de Salida:

raíces // almacena las raíces de la ecuación cuadrática

Proceso:

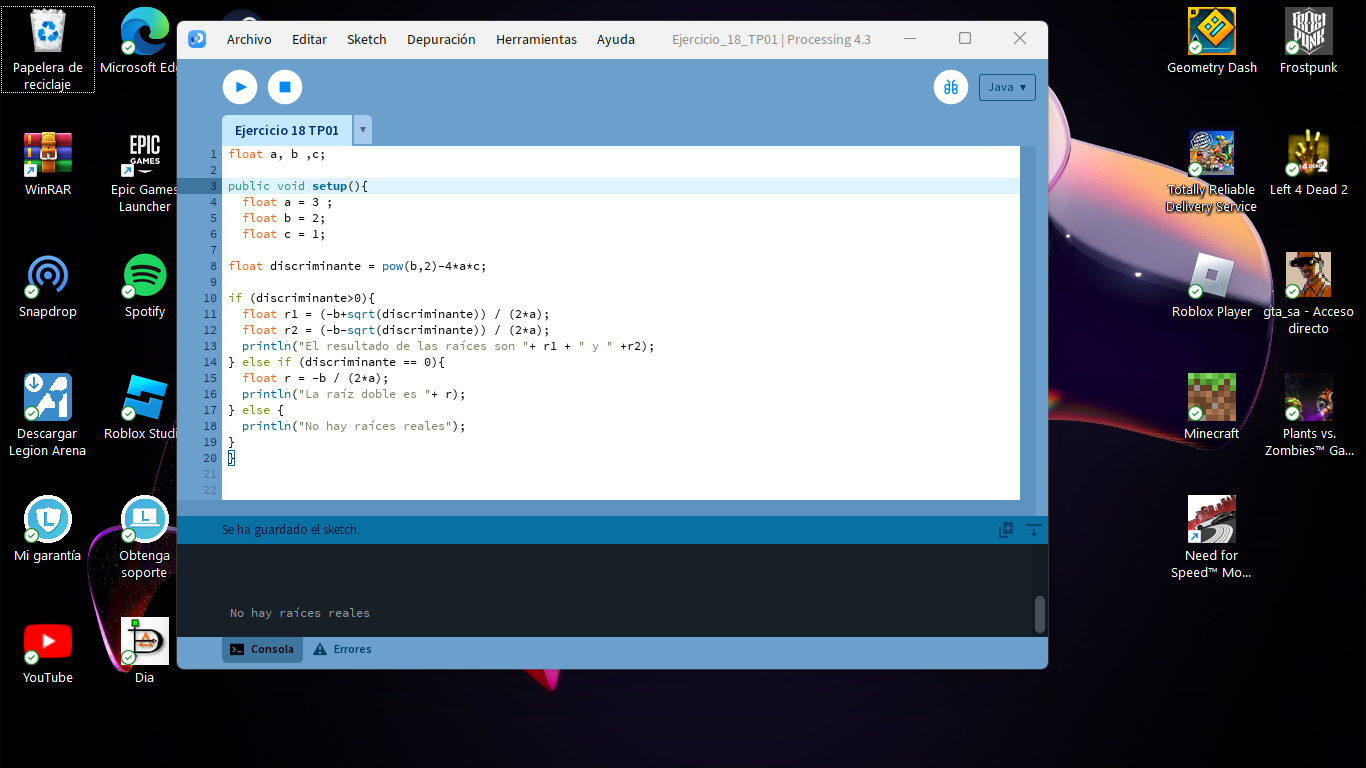
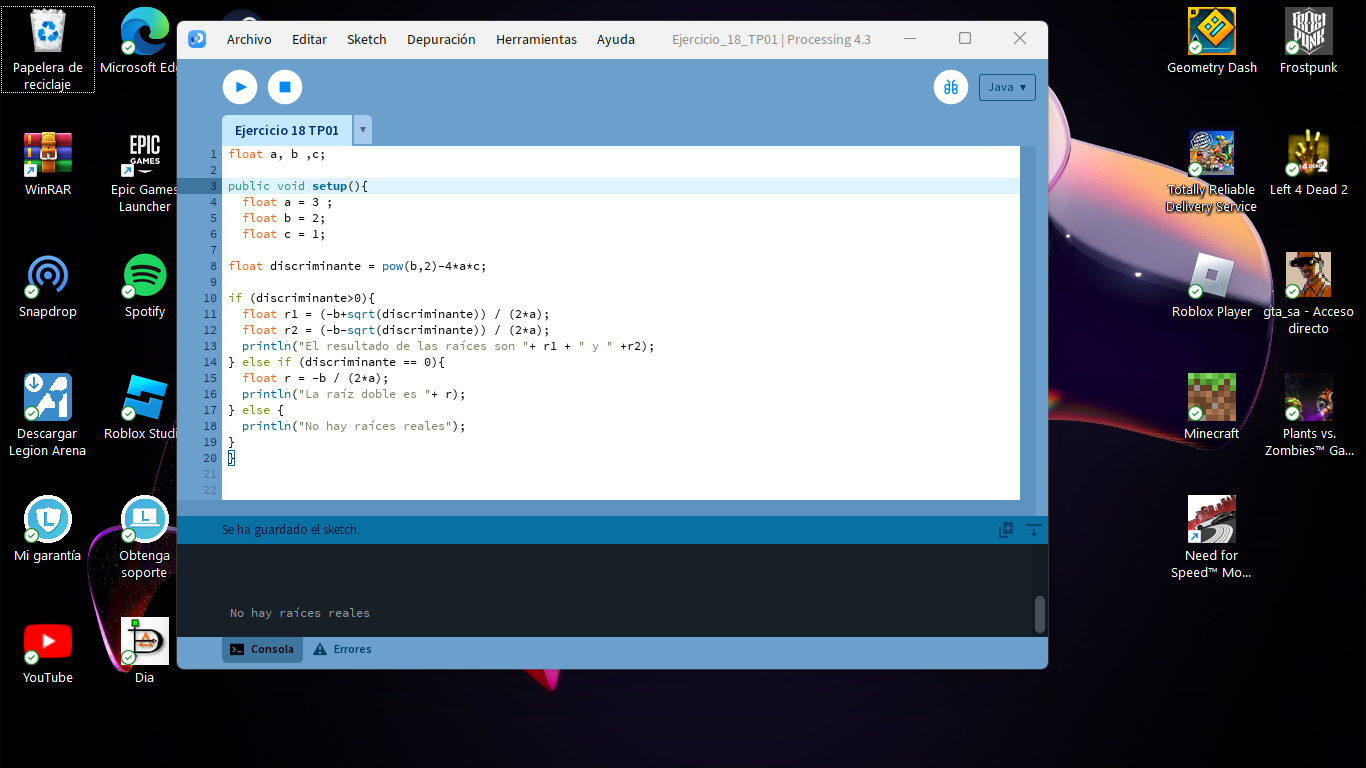
¿Quién debe realizar el proceso?: calculadora

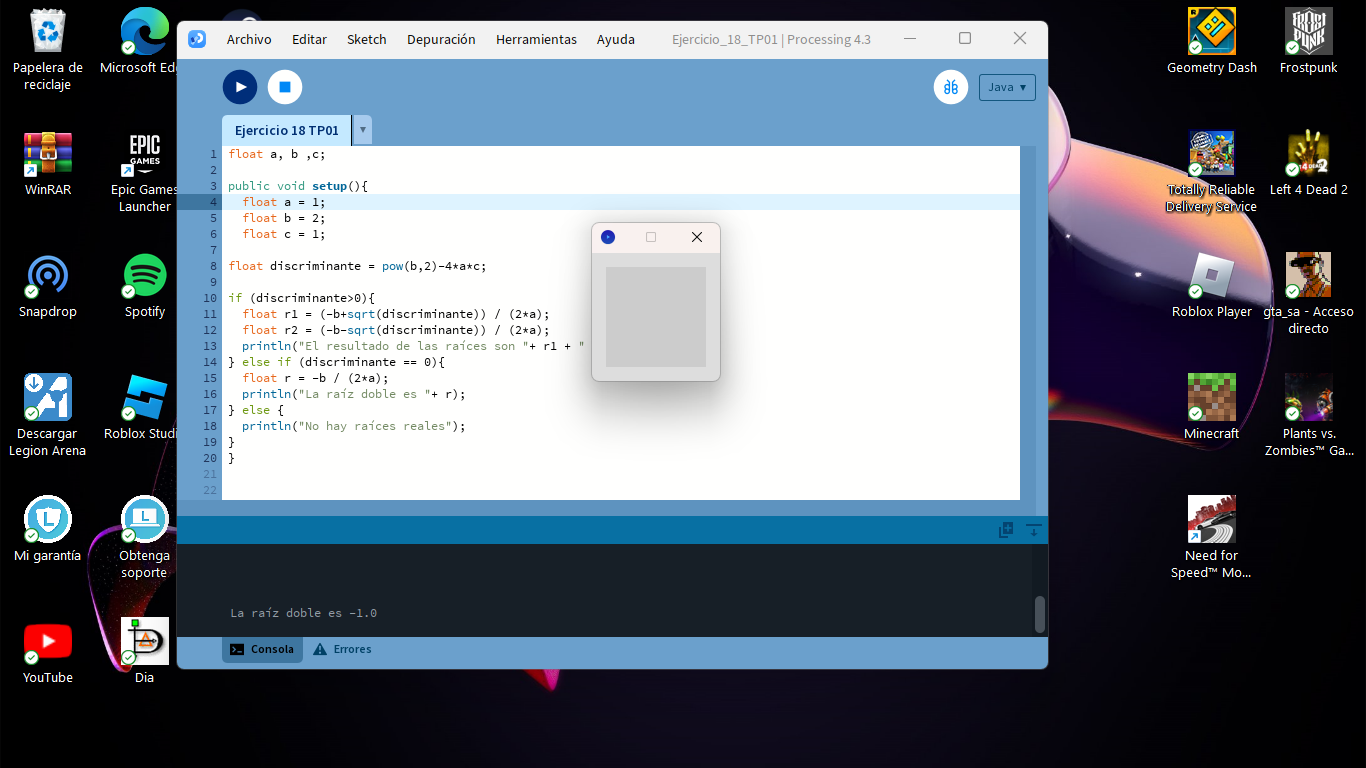
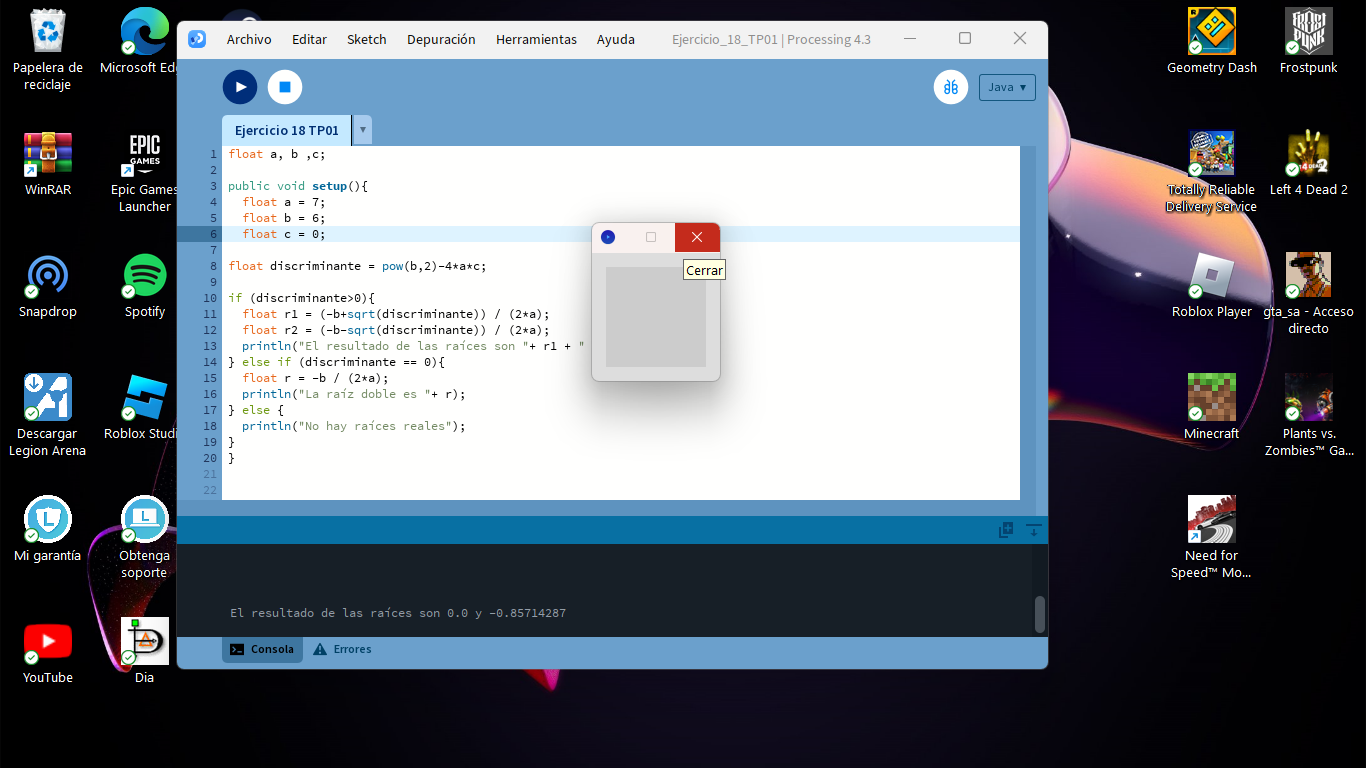
¿Cuál es el proceso que resuelve?: tiene que calcular el discriminante de la ecuación cuadrática usando la siguiente formula.

−𝑏±√𝑏2−4𝑎𝑐2𝑎

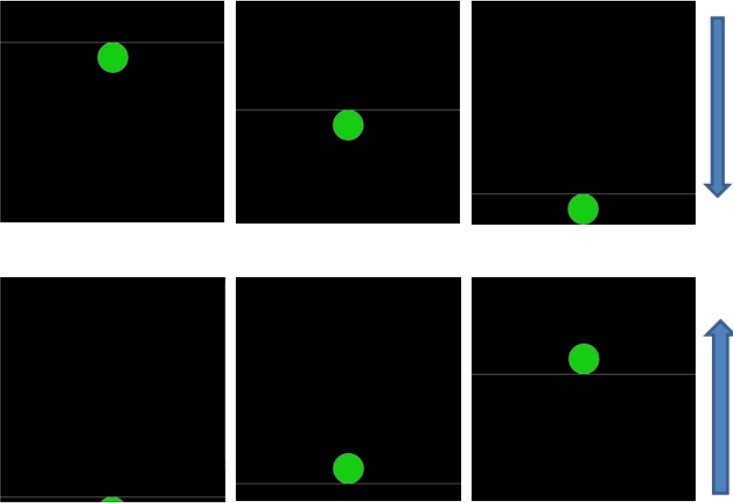
Fase de Diseño:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Calculadora |
| **VARIABLES:**  a: Real  b: Real  c: Real  discriminante: Real |
| **NOMBRE ALGORITMO:** hayar\_raices  **PROCESO DEL ALGORITMO:**   1. *Leer* a 2. *Leer* b 3. *Leer* c 4. discriminante <-- b^2-4\*a\*c 5. **si** (discriminante>0) **entonces** 6. raiz1 <-- (-b+(discriminante)) ^0.5/(2\*a) 7. raiz2 <-- (-b-(discriminante)) ^0.5/(2\*a) 8. *Mostrar* “El resultado de las raíces son: ”+ raiz1 + “ y " + raiz2 9. **si\_no si** (discriminante == 0) **entonces** 10. raiz <-- -b/(2\*a) 11. *Mostrar* “La raíz doble es: ”+ raiz 12. **si\_no** 13. *Mostrar* “no hay raíces reales” 14. *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



Fase de Análisis:

* Especificación del problema: Dibujar un círculo de 80px de ancho por 80px de alto, donde este ubicado en el medio de una línea que atraviese todo el ancho del lienzo. El circulo debe bajar y subir junto con la línea.
* Análisis:

Datos de Entrada:

línea

dir

Datos de Salida:

circulo

bucle de línea

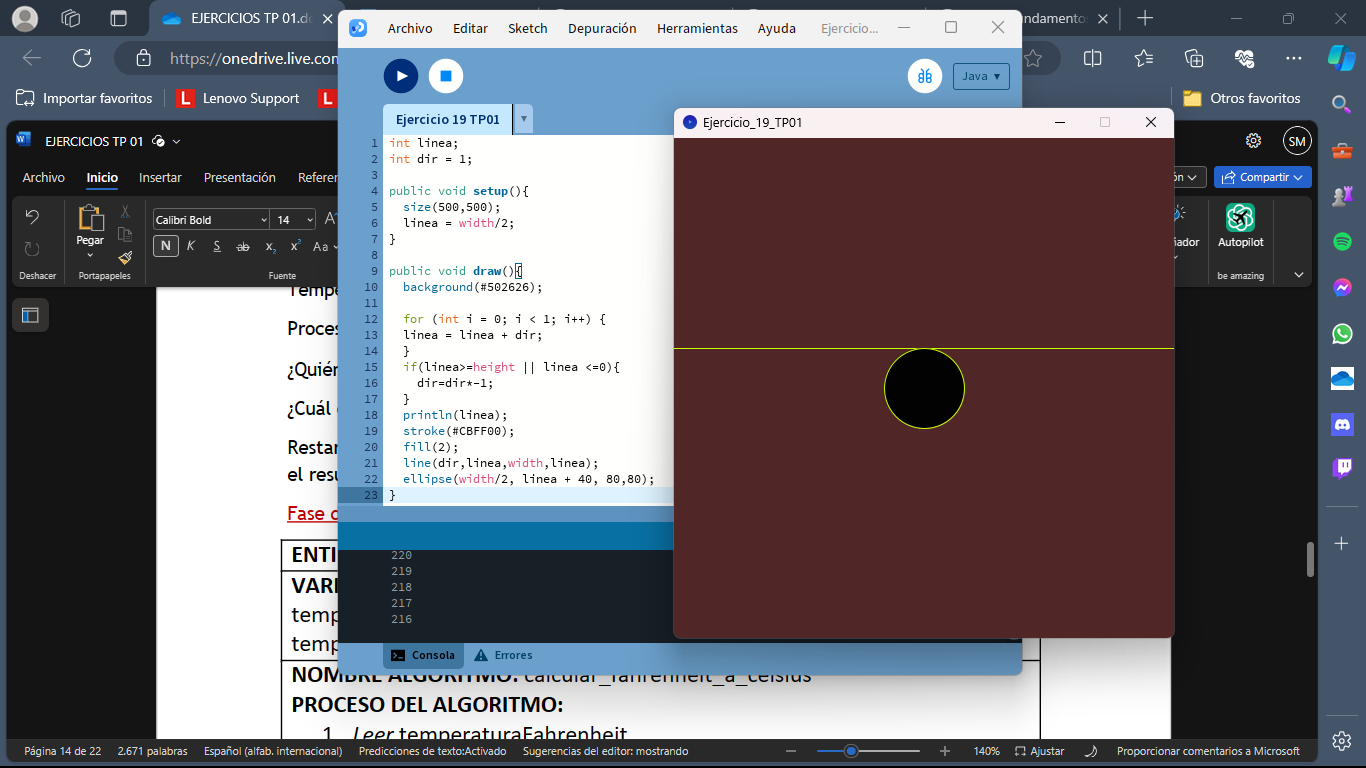
Proceso:

¿Quién debe realizar el proceso?: Processing

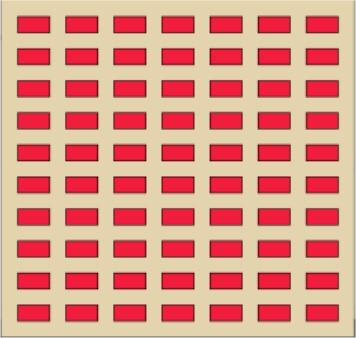
¿Cuál es el proceso que resuelve?:

Fase de Diseño:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Processing |
| **VARIABLES:**  linea: Entero // almacena el valor de la línea  dir: Entero // almacena el valor de la dirección de la linea |
| **NOMBRE ALGORITMO:** mover\_arriba\_abajo\_de\_linea  **PROCESO DEL ALGORITMO:**   1. *Leer* linea 2. *Leer* dir 3. anchoLienzo <-- 500 4. altoLienzo <-- 500 5. **para** i <-- 0 hasta **alto** incremento 1 **hacer** 6. **linea** <-- linea + dir 7. **fin\_para** 8. **si** ((linea >= anchoLienzo) O (linea <= 0)) **entonces** 9. dir ← dir \* (-1) 10. **fin\_si** 11. M*ostrar* linea 12. dibujar linea en (dir, linea, altoLienzo, linea) 13. dibujar circulo en (altoLienzo/2, linea + 40, 80, 80) 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:



Fase de Análisis:

* Especificación del problema: generar rectángulos de 40 x 20 a lo largo y ancho de un cuadro de 440 x 420
* Análisis:

Datos de Entrada:

anchoRect: Real

altoRect: Real

distEntreRect: Real

coordenadasRect: Real

Datos de Salida:

rectángulo: Real

Proceso:

¿Quién debe realizar el proceso?: Processing

¿Cuál es el proceso que resuelve?:

determinar si X es menor que el cuadrado 440 x 420, si se cumple esa condición, entonces se deben dibujar 7 rectángulos a lo ancho

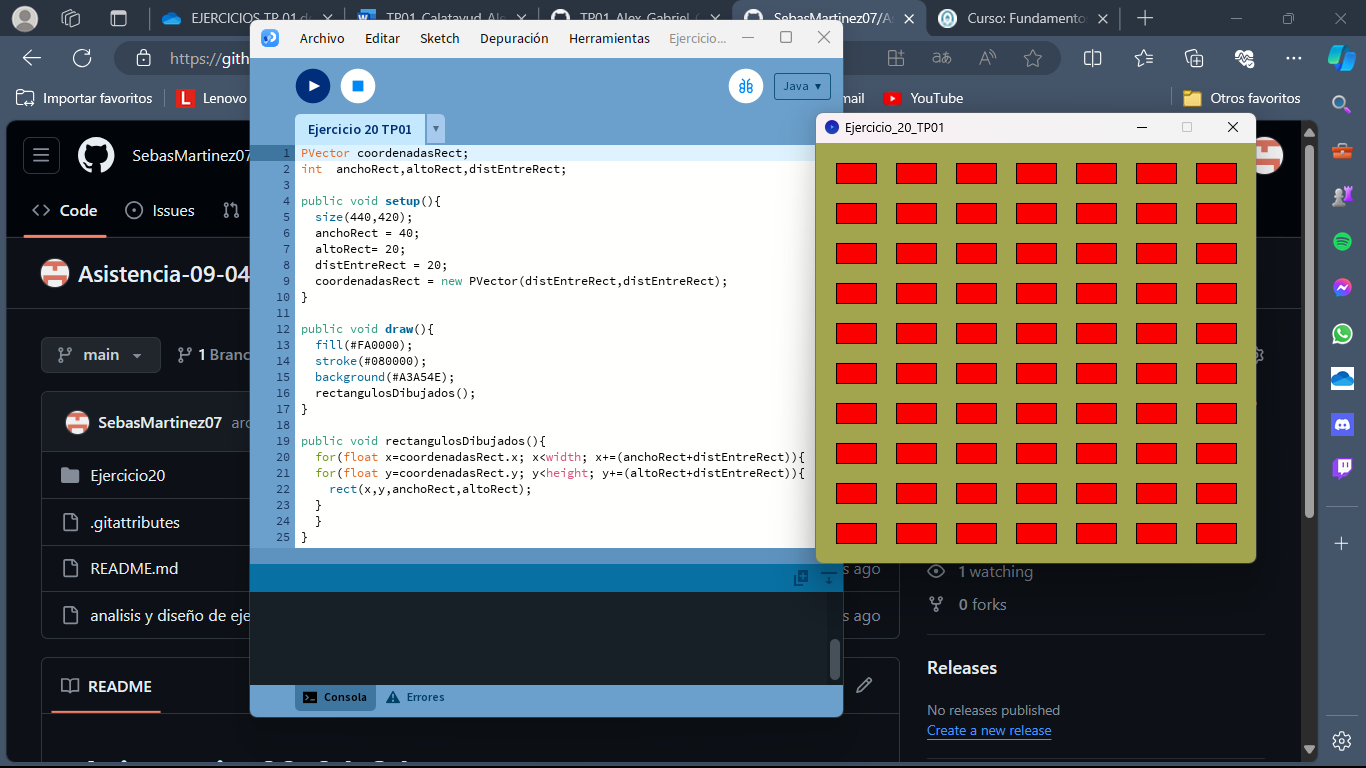
(x<ancho)

determinar si y es menor que el cuadrado 440 x 420, si se cumple esa condición, entonces se deben dibujar 10 rectángulos a lo alto

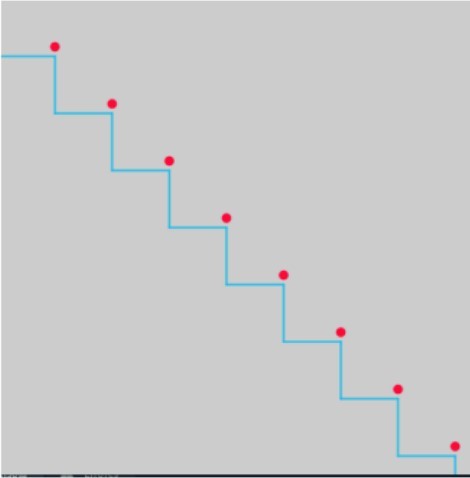
(y<alto)

Fase de Diseño:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** el cuadrado |
| **VARIABLES:**  anchoRect, altoRect: Real // almacena el alto y ancho del rectángulo  distEntreRect: Real // almacena la distancia entre cada rectángulo  coordenadasRect: Real // almacena las coordenadas donde empieza el rectángulo |
| **NOMBRE ALGORITMO:** colocar\_rectángulos  **PROCESO DEL ALGORITMO:**   1. *Leer* anchoRect 2. *Leer* altoRect 3. *Leer* distEntreRect 4. *Leer* coordenadasRect 5. **si** (x<ancho) **entonces**   x+=(anchoRect+distEntreRect)   1. **si** (y<alto)**entonces**   y+=(altoRect+distEntreRect)   1. *Mostrar* rectángulo |



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 utilizando la estructura while() en un lienzo que tenga 8 escalones de ancho alto y 8 puntos rojos en el borde de estos escalones.
* Análisis:

Datos de Entrada:

puntoA

puntoB

puntoC

puntoD

dist

Datos de Salida:

8 escalones con puntos rojos en el borde

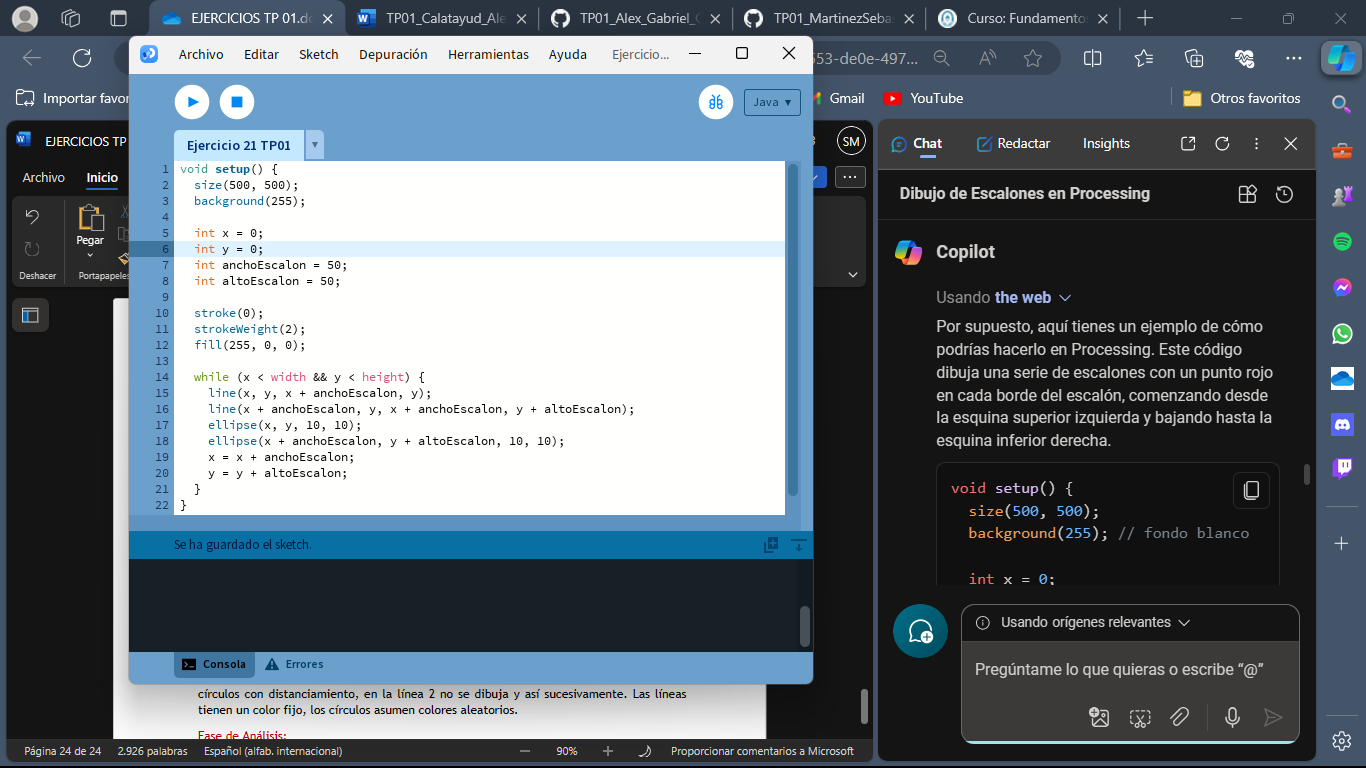
Proceso:

¿Quién debe realizar el proceso?: Processing

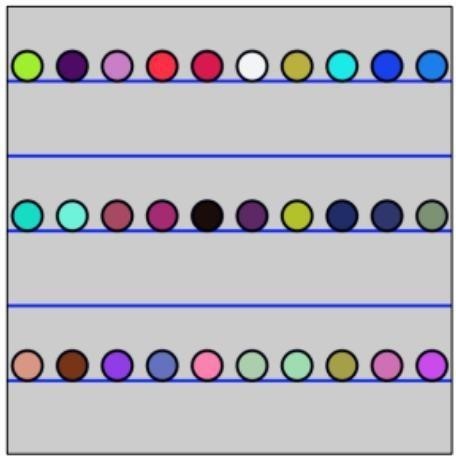
¿Cuál es el proceso que resuelve?: iterar con while para formar los 8 escalones y los puntos rojos

Fase de Diseño:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Processing |
| **VARIABLES:**  puntoA: Entero  puntoB: Entero  puntoC: Entero  puntoD: Entero  distancia: Entero |
| **NOMBRE ALGORITMO:** armar\_escalones\_puntos  **PROCESO DEL ALGORITMO:**   1. *Leer* puntoA 2. *Leer* puntoB 3. *Leer* puntoC 4. *Leer* puntoD 5. **Mientras** (puntoA.y <= anchoLienzo) **hacer** 6. dibujar línea horizontal en (puntoA.x, puntoA.y, puntoB.x, puntoB.y) 7. dibujar línea vertical en (puntoB.x, puntoB.y, puntoC.x, puntoC.y) 8. dibujar circulo en (puntoD.x, puntoD.y) 9. puntoA.x ← puntoC.x 10. puntoA.y ← puntoC.y 11. *fin\_mientras* 12. *fin* |



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: armar en processing un lienzo que tenga 5 línea verticales y la 1ra, 3ra y 5ta línea deben tener 10 círculos arriba de la línea con colores aleatorios.
* Análisis:

Datos de Entrada:

líneas

círculos

Datos de Salida:

círculos con colores aleatorios arriba de líneas

Proceso:

¿Quién debe realizar el proceso?: Processing

¿Cuál es el proceso que resuelve?: el lienzo dibuja 5 líneas verticalmente y arriba dibuja 10 círculos con colores aleatorios en orden intercalado entre cada línea, empezando en la primera línea

Fase de Diseño:

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA:** Lienzo de processing |
| **VARIABLES:**  lineaX, lineaY, circuloX, circuloY: Entero  distCirculo: Entero |
| **NOMBRE ALGORITMO:** dibujar\_lineas\_circulos  **PROCESO DEL ALGORITMO:**   1. *Leer* lineaX 2. *Leer* lineaY 3. *Leer* circuloX 4. *Leer* circuloY 5. anchoLienzo <-- 600 6. altoLienzo <-- 600 7. lineaX ← 0 8. lineaY ← 100 9. distanciaCirculo ← 30; 10. circuloY ← 75 11. **hacer** 12. circuloX ← distanciaCirculo 13. h**acer** 14. dibujar linea en (lineaX, lineaY, anchoLienzo, lineaY) 15. dibujar circulo en circuloX, circuloY, 50, 50) 16. circuloX ← circuloX + distanciaCirculo\*2 17. **fin\_hacer** 18. **mientras** (circuloX sea menor que ancholienzo) 19. LineaY ← lineaY + 100; 20. circuloY ← circuloY + 200; 21. **fin\_hacer** 22. **mientras** (lineaY sea menor que altoLienzo) 23. *fin* |

