2do Proyecto

Inputs:

- Editor de texto in-app: Donde se puede escribir los comandos. Al pulsar un boton se procesan y se ejecutan. Ademas, se debe poder exportar el texto.
- Lector de archivos con extension .pw. Donde se pueda importar archivos guardados y cargarlos.

Instrucciones:

- `Spawn(int x, int y)`:
  - Para que un codigo sea valido debe comenzar con Spawn.
  - Solo puede escribirse Spawn una vez.
  - '*x*' y '*y*' son las coordenadas iniciales.
  - Si cae fuera de los limites del canvas debe lanzar error en tiempo de ejecucion.
- `Color(string color)`:
  - Cambia el color del pincel.
  - Admite las siguientes colores: "*Red*", "*Blue*", "*Green*", "*Yellow*", "*Orange*", "*Purple*", "*Black*", "*White*", "*Transparent*".
  - Por defecto el color es "Transparent", que no realiza cambios en el canvas.
  - Como el canvas es blanco, "White" es un borrador.
- `Size(int k)`:
  - Modifica el tamano del pincel.
  - La entrada "*k ¿0*" representa el grosor en pixeles, de la brocha.
  - El grosor *k* debe ser un numero impar. Si la entrada es par, debe utilizarse su numero impar inmediatamente menor. Ej: si es 4 se pone 3
  - El tamano por defecto del pincel es de un pixel (*k ¿1*).
- `DrawLine(int dirX, int dirY, int distance)`:
  - Dibuja un linea en el canvas que comienza en la '*x*' y la '*y*' de Spawn y termina a una distancia = a *distance* en la direccion dada.
  - Las direcciones son:
    - (-1,-1) Diagonal Arriba Derecha
    - (-1,0) Izquierda
    - (-1,1) Diagonal Abajo Izquierda
    - (0,1) Abajo
    - (1,1) Diagonal Abajo Derecha

        - (1,0) Derecha
        - (1,-1) Diagonal Arriba Derecha
        - (0,-1) Arriba
    - Se deben pintar todos los pixeles en el camino de la linea y los adyacentes segun el ancho actual del pincel.
    - Al terminar la linea, la posicion de WallE se debe actualizar.
- `DrawCircle(int dirX, int dirY, int radius)`:
    - Dibuja un circulo de radio *radius* en la direccion establecida
    - La posicion final de WallE se vuelve el centro del circulo
    - No se dibuja el radio
- `DrawRectagle(int dirX, int dirY, int distance, int width, int height)`:
    - Dibuja un rectangulo
    - WallE se mueve en la direccion (dirX, dirY) una cantidad de pixeles = distance. La posicion donde se ubique WallE sera el centro de un rectangulo de largo *width* y *ancho* height
- `Fill()`:
    - Pinta con el color actual de la brocha todos los pixeles del color de la posicion actual que son alcanzables sin tener que caminar sobre otro color
- `var <- Expression:`
    - var tiene forma de cadena de texto y puede tener cualquiera de los 27 caracteres del espannol, caracteres numericos y el simbolo '-'.
    - var no puede comenzar por numeros ni por '-'
    - Expression es cualquier expresion aritmetica o booleana
    - Despues de una asignacion debe haber un salto de linea

Expresiones:

```
Expresiones artitmeticas:
- Pueden estar formadas por:
        - Un literal (numero entero)
        - Variable numerica
        - Operacion aritmetica entre dos o mas expresiones aritmeticas
        - Una invocacion de funcion
- Operaciones artimeticas soportadas:
        - +
        - -
        - /
        - ** (potencia)
```

```
        - % (modulo)

Expresiones booleanas:
- Son verdaderas o Falsas
- Pueden ser:
        - and (&&)
        - or (||)
        - comparadores (== , >=, <=, <, >) entre dos o mas variables numericas
o literales
- or tiene mayor precedencia que and
```

Funciones:

- `GetActualX()` y `GetActualY()`:
  - Retornan la "*x*" y la '*y*' respectivamente de la posicion actual de WallE
- `GetCanvasSize():`
  - Retorna largo y ancho del canvas. Si es de n x n retorna n.
- `GetColorCount(string color, int x1, int y1, int x2, int y2):`
  - Retorna la cantidad de casillas con color *color* que hay en el rectangulo formado por (x1, y1) esquina superior izquierda y (x2, y2) esquina inferior derecha.
- `IsBrushColor(string color):`
  - Devuelve 1 si el color de la brocha actual es color, 0 si no.
- `IsBrushSize( int size):`
  - Retorna 1 si el tamanno de la brocha actual es size, 0 si no.
- `IsCanvasColor(string color, int vertical, int horizontal):`
  - Retorna 1 si la casilla sennalada esta pintada de color *color*, 0 si no.
  - La casilla a analizar se calcula usando (X,Y) como la posicion actual de WallE y se calcula como (X + horizontal, Y + vertical).
  - Si cae fuera del canvas retorna 0.

Saltos condicionales:

```
Etiquetas:
- Cadenas de texto que marcar un lugar del codigo al que llegar a traves de
GoTo
- Por si mismas no tienen efecto al llegar su momento de ejecutarse
- Tiene la misma forma que las variables.
- Despues de una etiqueta debe haber un salto de linea
- Ejemplos: start-loop, end-loop, boniato
```

```
Saltos condicionales:
-Tienen la forma GoTo[label](condition)
-label debe ser una etiqueta anteriormente declarada en el codigo tanto antes como despues de la declaracion del salto
```

preguntar esto de arriba

```
-si label no existe en el codigo se lanza error de compilacion
-condition es una variable booleana o una comparacion entre dos variables numericas o literales.
-los saltos condicionales tienen el efecto de que, si la condicion es Verdadero el codigo continue su ejecucion en la lınea donde este label.
 -Si la condicion tiene valor Falso, entonces la linea se ignora y se continua la ejecucion en la linea siguiente.
 -Despues de un GoTo debe haber un salto de linea
```

Interfaz

- Editor de texto: Entrada de texto con numeros de linea en la parte izquierda
- Canvas: Seccion cuadriculada. Cada casilla es un pixel. Se modifica en funcion del codigo
- Entrada de dimensiones del canvas: Introducir numeros que modifiquen el ancho y el largo del canvas. Puede tener valor por defecto
- Boton de redimension del canvas: Cambia las dimensiones del canvas segun las entradas. Limpia el canvas anterior. Inicializa blancas todas las casillas
- Boton de ejecucion: Ejecuta el codigo. Si el canvas estaba modificado con anterioridad se utiliza el canvas modificado
- Boton de carga y de salva: Cargan y guardan archivos del formato de texto en formato .gw

Importante:

- Si se ejecuta un codigo valido sintatica y semanticamente pero con un error en ejecucion, debe capturarse el error, reportarlo al usuario y parar a ejecucion. El codigo valido antes de este error debe permanecer en el canvas.
- Si el codigo de entrada no es valido sintactica o semanticamente debe reportarse el tipo de error

Ejemplo de codigo:
Spawn(0, 0)

```
Color(Black)
n <- 5
k <- 3 + 3 *10*
*n* <- *k* 2
actual-x <- GetActualX()
i <- 0

loop-1
DrawLine(1, 0, 1)
i <- i + 1
is-brush-color-blue <- IsBrushColor("Blue")
Goto [loop-ends-here] (is-brush-color-blue == 1)
GoTo [loop1] (i < 10)

Color("Blue")
GoTo [loop1] (1 == 1)

loop-ends-here
```