2do proyecto de programación

Liz Cartaya Salabarría

C112

Proyecto realizado en C# en Visual Studio usando Windows Form.

Pixel Walle es un programa que es capaz de recibir una serie de comandos del lenguaje Pixel_Walle-E y transformarlas en un dibujo en un canvas cuadriculado.

Todas las instrucciones del lenguaje están en el Readme del proyecto.

El proyecto está divido en varios archivos.

En Form1.cs se encuentra todo lo relacionado con la parte visual de este proyecto. En la visualización encontraremos un canvas donde se podrán apreciar los dibujos; un cuadro de texto donde se ponen las instrucciones o comandos; un cuadro de texto donde se va mostrando la línea que se va ejecutando; un cuadro de texto en el que se puede modificar el tamaño del canvas; pequeños cuadros de texto donde se modifican los tiempos de dibujo y ejecución; cinco botones:

Run(ejecutar)

Save(guardar los comandos en un archivo de tipo .pw)

Upload(cargar archivos antes guardados)

Reset(Reiniciar la aplicación)

Close(cerrar la aplicación).

Además cuenta con la opción de Debug por defecto sin marcar(Falso) con la que se puede ver los tokens y la forma en que se dividen los tokens en un árbol de sintaxis abstracta.

En Token.cs se encuentra un Enum con los tipos de tokens que acepta el programa y en este archivo se definen las propiedades que posee un token.

En Lexer.cs se encuentra la clase Scanner donde se clasifican los token según su tipo y se van agregando a una lista.

En AST.cs es donde se trabaja el concepto de árbol de sintaxis abstracta. Es aquí donde se crean los distintos tipos de nodos según el tipo de estructura. Cada uno posee un token de inicio y de fin, así como una lista de Childrens. En este archivo creé inicialmente un ASTNode (genérico) y los demás heredan de él.

En Parser.cs en función de la estructura sintáctica se crea el tipo de AST correspondiente.

En Class1.cs se encuentra la clase Interprete, la cual se encarga de interpretar el texto. Se almacena cada línea del texto de entrada, se llama al Scanner para que los separe en token y al parser para que convierta los tokens en un Árbol de Sintaxis Abstracta (AST). Además, se encuentra el logger el cual se encarga de registrar mensajes y errores.

Executer.cs es donde se encuentra la clase executer que es la encargada de ver a qué método llamar y que argumentos pasarle.

En Context.cs es donde se definen todos los tipos de comandos que acepta mi programa.