MANUAL TÉCNICO

Proyecto final: Organizacion de lenguajes y compiladores 1

‘IGCreator’

José Carlos Soberanis Ramírez

201730246

Descripcion: El programa consta de 4 analizadores lexicos, y 4 sintacticos, que hacen una traducción del código ingresado por el usuario para generar objetos que se pueden manipular dentro del programa, siendo estos lienzos, sobre los cuales se puede pintar.

Analisis Léxico

- Un analizador léxico es la primera fase de un compilador, consistente en un programa que recibe como entrada el código fuente de otro programa (secuencia de caracteres) y produce una salida compuesta de tokens.
- Analizador léxico archivo Lienzo
  - Tokens identificados
    - Enteros[1]
    - Identificadores[2]
    - Palabras reservadas
      - LIENZOS
      - nombre
      - tipo
      - tamaño
      - Fondo
      - Red
      - Blue
      - Green
      - HEX
      - cuadro
      - dimension_x
      - dimension_y
    - Agrupadores
      - {
      - }
    - Operadores
      - :
    - Valores hexadecimales: valores hexadecimales seguidos de #
- Analizador léxico archivo Colores
  - Tokens identificados
    - Entero
    - Identificadores
    - Palabras Reservadas
      - COLORES
      - Red
      - Blue
      - Green
      - HEX
    - Agrupadores

1 Uno o mas digitos seguidos

2 Una cadena de uno o mas simbolos alfanumericos, incluyendo al simbolo “_”, que siempre inician con una letra o “_”.

        - {
    - Operadores
        - :
- Analizador léxico archivo tiempos
    - Tokens identificados
        - Enteros
        - Identificadores
        - Palabras reservadas
            - TIEMPOS
            - inicio
            - fin
            - duracion
            - id
        - Agrupadores
            - {
            - }
        - Operadores
            - :
- Analizador Léxico archivo pintar
    - Tokens identificados
        - Enteros
        - Identificadores
        - Palabras reservadas
            - if
            - else
            - while
            - VARS
            - INSTRUCCIONES
            - AND
            - OR
            - String
            - int
            - boolean
            - true
            - false
        - Operadores
            - +
            - -
            - *
            - /
            - <
            - >
            - <=
            - >=
            - ==
            - <>

Análisis sintáctico

- Es una de las partes de un compilador que transforma la entrada en un arbol de derivación. El análisis sintáctico convierte el texto de entrada en otras estructuras (conmúnmente árboles), ya que son útiles para el posterior análisis y capturan la jerarquía implicita en la entrada.

- Analizador sintactico archivo Lienzo está basado en la siguiente gramática, para crear los lienzos correspondientes.
  - Click_Aquí
- Analizador sintactico archivo Colores está basado en la siguiente gramática, para crear los colores en sus respectivos lienzos.
  - Click_Aquí
- Analizador sintactico archivo Tiempos está basado en la siguiente gramática, para crear las imagenes con sus respectivos tiempos.
  - Click_Aquí
- Analizador sintactico archivo Pintar está basado en la siguiente gramática, para pintar como corresponda.
  - Click_Aquí

Estructura de los archivos

- Analizadores
  - Colores
    - Contiene el lexer y parser para el archivo de entrada de colores.
  - Lienzos
    - Contiene el lexer y parser para el archivo de entrada de lienzos
  - Tiempos
    - Contiene el lexer y parser para el archivo de entrada de tiempos
  - Pintar
    - Contiene el lexer y parser para el archivo de entrada de pintar
  - Objetos
    - Instrucciones
      - Instruccion
        - Permite castear cualquiera de las siguientes instrucciones según su tipo
      - AsignacionInstruccion
        - Posee el id de la variable a la cual se desea asignar, y su arbol correspondiente de la expresion que se debe evaluar para asignar el valor.
      - InstruccionPintar
        - Es la traducción finalizada de las acciones a realizar sobre cada lienzo, contiene un idLienzo, idImagen, idColor, posicionX, posicionY
      - MientrasInstruccion
        - Posee un arbol correspondiente a las condiciones para ejecutar el while, y un listado de instrucciones si esta condicion se cumple.
      - PintarInstruccion
        - Posee la instrucción pintar primitiva leída del archivo de entrada, con un arbol para el idColor, idImagen, posX, posY.
      - SiInstruccion

        - Posee un arbol de expresion, correspondiente a las condiciones para ejecutar las instrucciones si esta se cumplen, o bien ejecutar las instrucciones si nos se cumple (si es que tiene).
    - InstruccionManager
        - Se encarga de hacer el analisis sobre la ejecucion de las instrucciones
- Atributo
    - Variable que almacena cualquier valor, indicandonos el valor que representa.
- Nodo
    - Nodo de un arbol de expresiones
- Variable
    - Variable que almacena un id, un tipo de dato, y el valor correspondiente que almacena dicha variable.
- Rango
    - Variable que se crea para una instrucción pintar que posee un rango en x o y.
- Token
    - Variable utilizada para el analisis lexico y para un mejor rendimiento del reporte de errores.

- Objetos
    - ColorP
        - Guarda un color con valor RGB o hexadecimal
    - GifSequenceWriter
        - Clase que crea las animaciones GIF
    - Imagen
        - Guarda un id, duracion y un Panel que representa a la respectiva imagen.
    - Lienzo
        - Variable que guarda todos los datos recolectados por todo el analisis que se ejecuta
    - Tiempo
        - Variable que almacena el listado de imagenes al realizar el analisis.
- GUI
    - Cell
        - Representa un cuadro dentro del panel para el editor grafico
    - Editor
        - Es el frame del editor de texto
    - Tab
        - Es cada uno de los TabPanel que se añaden al ir cargando los archivos de entrada.
    - PanelLienzo
        - Es un panel en el que se encuentran las herramientas para el editor grafico.
    - Pintor
        - Es el frame del editor grafico
    - RegistroArchivos
        - Clase que se encarga del manejo de los archivos, lectura/escritura.

Analisis semántico

- Verifica la compatibilidad entre un operador y sus operandos, que el flujo sea correcto, que no haya duplicidad de nombres, etc.

  Hay dos tipos, estatico, en tiempo de compilacion y dinamico en tiempo de ejecución, en nuestro caso particular lo hemos realizado dinámico, retornando siempre que se ejecute una acción invalida un valor null, que es capturado posteriormente, indicando la linea en la que se encuentra la instrucción que provocó el error.

  El analisis semántico dentro del código se realizó mediante un arbol de expresiones, utilizando el objeto nodo, que tiene 3 atributos, un valor, un tipo de retorno y un operador.

  Esto nos ayuda a que si el nodo no posee un operador significa que se trata de un nodo hoja, por ende tiene un valor asignado, todos los nodos que no sean hoja poseen un operador y no un valor.

  Al momento de evaluar el arbol en postOrden, verificamos el tipo de retorno que tiene el nodo evaluado, y si coincide con el tipo de dato de una variable se asigna, de lo contrario provoca un error semántico.

  Siendo ese básicamente su funcionamiento; evaluar los arboles y verificando que el tipo de retorno coincida con la operación que se está realizando respectivamente.

Datos desarrollo

- SO: Manjaro Linux version 20
- Ram: 8 gb
- Procesador: Intel core i7-6500u a 2.5 Ghz
- Lenguaje Utilizado: Java 11
- Librerias utilizadas/dependencias
  - Cup version 11
  - Jflex version 1.8.2
- IDE utilizado: Netbeans 11