Informe de proyecto HULK

Jean Carlo García Wong Diciembre,2023

1. Introducción

En este proyecto de programación llamado HULK he adquirido diversos conocimientos sobre la rama de la programación conocida como Compilación que es de vital importancia para mi desarrollo como estudiante de la carrera Ciencias de la Computación y como futuro progromador.

HULK es de una gran importancia para mi desarrollo como programador porque con él hemos podido ver cuales son los buenos hábitos que todo coder debe tener a la hora de realizar sus proyectos, me ha servido como otra materia de estudio donde he aprendido mucho sobre los algoritmos utilizados en él. Espero que les sea de utilidad este informe.

2. Requisitos para la ejecución

- .Net 7.0

3. Desarrollo

De forma general este proyecto de programación que se nombra HULK como ya habiamos mencionado anteriormente consta de dos partes, una que es una aplicación de consola llamada Visual que es donde se ejecuta la lógica del compilador y la otra llamada Parsing que es donde se implementa el lenguaje de HULK y de todas sus funcionalidades, la cual es donde se situa la mayor parte del código del mismo.

3.1. Parsing

La parte llamada Parsing consta de varias clases que hacen posible el funcionamiento del lenguaje HULK las cuales son:

- TokenValue
- BoolToken
- StringToken
- SpecialTokenClass
- Function
- RecursiveFunction
- Variable
- Lexicon
- Sintaxis
- Semantic
- PredFunction

3.1.1. TokenValue

La clase TokenValue es utilizada para almacenar el valor y el nombre de todas las variable instanceadas por el usuario, hereda de la propiedad name que tiene el nombre correspondiente a la variable y recibe la propiedad valor que tiene el valor correspondiente a la variable.

3.1.2. BoolToken

La clase BoolToken esta creada para almacenar las variables de tipo bool y tenerlas separadas del resto por las particularidades que esta tiene, hereda de la clase TokenValue y consta de dos propiedades, una llamada nombre para el nombre de dicha variable y otra llamada valor para el valor de la misma, propiedad que recibe por la herencia de TokenValue.

3.1.3. StringToken

La clase StringToken es utilizada para almacenar las cadenas usadas por el usuario en el código ingresado, heredando las propiedades name y valor de la clase TokenValue correspondientes al nombre y valor de dicha cadena respectivamente.

3.1.4. SpecialTokenClass

La clase SpecialTokenClass es utilizada para almacenar todas las variables, objetos de tipo StringToken y funciones implementadas por el usuario en la o las líneas de código que ingreso la cual contiene la propiedad name que corresponde al nombre del objeto correspondiente.

3.1.5. Function

La clase Function esta creada para almacenar las funciones que crea el usuario que hereda la propiedad name de la clase SpecialTokenClass y recibe las propiedades cuerpo para almacenar el cuerpo de la función y otra llamada parametro para almacenar los parámetros que recibe la misma.

3.1.6. RecursiveFunction

La clase RecursiveFunction es utilizada para almacenar las funciones que sean recursivas que haya implementado el usuario la cual hereda de la clase Function todas sus propiedades, name correspondiente al nombre de la función, cuerpo correspondiente al cuerpo de la función y parámetro correspondiente a los parámetros ingresados en la declaración de la función.

3.1.7. Variable

La clase Variable es utilizada para almacenar todas las variables creadas por el usuario sin importar su tipo de dato la cual hereda de la clase TokenValue heredando las propiedades nombre y valor que almacenan el nombre y el valor de la variable respectivamente y recibe la propiedad tipo la cual determina el tipo de la variable los cuales son: int, string y bool.

3.1.8. Lexicon

La clase Lexicon es donde se implementan todas las funcionalidades para realizar el análisis léxico de las líneas ingresadas por el usuario. Esta clase consta de varias propiedades en las cuales se almacenan toda el lenguaje de HULK. Esta clase cuenta con varios métodos que hacen posible que se realice el análisis léxico de las líneas de código que ingresa el usuario entre los cuales se destaca divideElements el cual parsea una línea completa y revisa todos los elementos que la conforman para ver si forman parte del lenguaje de HULK y en caso de que el usuario haya declarado alguna variable se proceda a crearla y guardarla en su correspondiente clase mediante el método checkTokens para entonces devolver una lista con la línea parseada donde cada elemento de la lista son los elementos de línea ya separados y en el orden en que el usuario los escribio. Además cuenta con dos

métodos auxiliares llamados addIntegerAsToken y BuildVar, el primero con el próposito de crear las variables tipo int y el segundo para llenar las listas que contienen las variables, los objetos de tipo StringToken y las funciones creadas por el usuario.

3.1.9. Sintaxis

La clase Sintaxis consta de dos métodos uno es BalancedParentexis el cual retorna una variable booleana que es true si los paréntesis de la línea que se esta analizando sintácticamente no están balanceados y falso si están balanceados y recibe una lista de string que representa a la línea ya parseada y el segundo método llamado syntaxisAnalysis que recibe una lista de string que representa a la línea ya parseada y una variable de tipo Lexicon que represesnta el resultado del análisis léxico de la línea que se esta analizando y se encarga de realizar todo el análisis sintáctico correspondiente a la línea recibida retornando una variable de tipo bool que es true si la línea esta escrita de forma erronea sintácticamente y da false si esta bien escrita.

3.1.10. Semantic

La clase Semantic consta de un método que se llama SemanticAnalysis que recibe una lista de string que representa a la línea ya parseada y una variable de tipo Lexicon que corresponde al resultado del análisis léxico correspondiente a la línea que se va a analizar, dicho método se encarga de llevar a cabo el análisis semántico de la línea que esta contenida en la lista y retorna true si el resultado de dicho análisis es erroneo y false si la línea fue escrita de forma correcta semánticamente.

3.1.11. PredFunction

La clase PredFunction contiene una serie de métodos que se encargan de ejecutar todas las funcionalidades del lenguaje HULK, ejecutando la línea que se acabo de analizar tanto léxica, sintáctica y semánticamente, el método que lleva a cabo esta función se denota como ExecuteLine que retorna una lista de string con lo que debe devolver la línea después de su ejecución y recibe una lista de string que representa la línea ya parseada y una variable de tipo Lexicon que corresponde al resultado del análisis léxico. También contiene los métodos let-in, if-else, print y ExecuteFunction que dichos métodos se encargan de ejecutar las funcionalidades de las instrucciones let-in, if-else, print y funciones según como las define lenguaje HULK respectivamente. Además contiene varios métodos auxiliares como son ClosePos, nextFunc, next, CloseFunc, CheckRec y entre otros que hacen posible el funcionamiento de las instrucciones principales mencioandas anteriormente de mi implementación del lenguaje HULK.

3.2. Visual

La parte llamada Visual como había dicho anteriormente consta de una aplicación de consola en la cual el usuario ingresa las líneas de código que estime necesarias para el uso del compilador. Después de haber ingresado el código se pasa a realizar el análisis léxico, sintáctico y semántico de cada línea ingresada y posteriormente se procede a la ejecución en caso de que los análisis mencionados anteriormente den resultados correctos.

4. Conclusiones

Este proyecto ha sido de gran utilidad como aprendizaje, me ayudó a mejorar mi enfoque hacia la programación y a saber la realidad de como es el día a día de un programador. Además aprendí un montón de malos hábitos que deben ser evadidos por cualquier persona que tenga conocimiento de la programación. Espero que haya sido de utilidad este informe.

Índice