|  |  |
| --- | --- |
|  |  |
|  | A&D Productions  Guatemala 28 de Abril, 2021  Manual Tecnico |
|  | El sistema desarrollado es una herramienta que permite al usuario realizar cálculos complejos mediante expresiones regulares, con mecanismos de evaluación, lo que ayudará al usuario final a que tenga una mejor comprensión de las ecuaciones matemáticas.  El objetivo de este manual técnico es informar a los usuarios finales de la funcionalidad del software. |
| "Hay dos formas de escribir programas sin errores; sólo la tercera funciona" | |



|  |  |  |
| --- | --- | --- |
| Los requerimientos minimos de la PC son los siguientes:  **Sistema Operativo:** Windows 7, Windows XP, Windows Vista (Windows XP Professional SP3/Vista SP1/Windows 7 Professional)  **Procesador:** Intel Pentium III 800 MHz (800MHz Intel Pentium III o su equivalente)  **RAM:** 512 MB  **Disco Duro:** 100mbs  **Tarjeta Gráfica:** N/A  **Resolución de Pantalla:** 1024 x 728 |  | **¿Cuáles son los requerimientos  del programa?** |

|  |  |
| --- | --- |
| Se desarrollo y un intérprete que recibe diversas  expresiones aritméticas, y este presentará como output el resultado de dicha expresión ingresada.  **¿Qué tecnologías se necesitan para  su funcionamiento y desarrollo?**  **Jlex**: es una herramienta desarrollada en Java que toma  como entrada un archivo “entrada”, con este crea un archivo fuente java entrada. lex. java correspondiente al analizador léxico. En otras palabras, Jlex es un generador de analizadores léxicos.  **CUP**: es un parser-generador. Es un analizador sintáctico que construye un parser para gramáticas tipo LALR(1), con código de producción y asociación de fragmentos de código JAVA. Cuando una producción en particular es reconocida, se genera un archivo fuente Java, parser. | **¿Cuál es el objetivo del software y cómo funciona?** |

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | Primero que nada, debemos tener instalado lo siguiente:   * Java Development Kit (JDK) * Netbeans 8.2   Se necesita la descarga de JLex el cual puede hacerse desde la siguiente página: <https://www.cs.princeton.edu/~appel/modern/java/JLex/>  Y también se debe descargar CUP, para ello se ingresa a: <http://www2.cs.tum.edu/projects/cup/>  Se descomprimen los archivos descargados, para ello se recomienda  WinRar, y obtendremos los siguientes archivos:    Dichas librerías deben estar instaladas en NetBeans, para poder asegurar que las librerías estén correctamente instaladas se puede ver de dos maneras.  La primera opción es ver las propiedades del programa > Libraries, si las librerías no se encuentra ahí, simplemente se agregan dando click sobre “Add JAR/Folder” e incluìmos ambos archivos. Como referencia:    Estas librerías las discutiremos mas adelanto pues nos ayudarán a que el software funcione de una forma optima. | **¿Cuáles son los prerrequisitos para el funcionamiento del software?** | |

|  |  |
| --- | --- |
| Dentro de la carpeta “Compilador” encontraremos nuestro Generador Lexico, el nombre de este  es “LexicalGrammarGenerator.java”, el cual nos ayudará a generar los archivos necesarios con la  lógica para el correcto funcionamiento del programa.  Para ello, el único cambio importante que se debe hacer es la ruta en donde se generaran los archivos. Por ejemplo:  C:/Users/UserName/Documents/NetBeansProjects/ CalculadoraCompiladores/src/Compilador/Lexer.flex";  Recordemos que todas las rutas utilizadas en esta sección deben actualizarse acorde a donde este ubicado el programa.  Antes de seguir avanzando, tomemos en cuenta que hay 2 archivos  que dan la funcionabilidad y lógica en el programa, los cuales son: Lexer.flex y Sintax.cup.  En el archivo “Lexer.flex” incluiremos todo el código que le indicará a Jlex lo que debe hacer. Como referencia, lo que debe llevar dicho archivo es:  En el archivo “Sintax.cup” incluiremos todo el código que le indicará a Cup lo que debe hacer. Como referencia, lo que debe llevar dicho archivo es: | **¿Cómo se realiza la configuración del software?** |
| El programa se ejecutará de una forma “individual”:    Luego de ejecutar el programa, se generarán 3 archivos:  Lexer.java Sintax.java sym.java  En este caso, la clase sym.java, sirve como puente entre la clase Lexer.java y Syntax.java. Por ejemplo, cuando el analizador léxico reconoce un número entero, instancia un objeto de la clase Symbol e indica que es de tipo número entero por medio de la constante que se genera dentro de la clase sym.java y esta constante se genera porque en el archivo de entrada para Cup se indicó que existe un terminal llamado “entero”. Entonces tanto el analizador léxico como el sintáctico hacen referencia a los tokens de tipo número entero con la misma constante. Básicamente eso es sym.java, una clase con muchas constantes estáticas a las que acceden ambos analizadores para poder integrarse y ejecutar sus tareas exitosamente. | **¿Qué genera el “LexicalGrammarGenerator” y como utilizarlo?** |

|  |  |
| --- | --- |
| El código se fue comentando conforme los metodos y cual es la función de cada una. Esta será unicamente una guía de como es el funcionamiento de las mismas.        Para esta sección se utilizaron de referencia dos páginas: <https://stackoverflow.com/questions/39034150/get-all-not-matching-characters-from-a-regex> <https://regex101.com/r/bicrhA/1>    La sección de **Infix a Postfix** es un poco mas compleja, se hace de la siguiente forma:    En donde luego se realizara un POP a todos los  operadores que llevamos ingresados en la STACK.    Aquí se utilizo de referencia la siguiente página: <https://tinyurl.com/yz3qjymj> (Misma página que en la captura, solo se acorto el link)  Para generar el **arbol sintactico** se utilizaron varios  imports que nos ayudaron a poder graficar el arbol se divide en dos secciones, la primera en donde se genera el arbol y en la otra es donde se grafica el arbol:    Se utilizaron los siguientes imports para el graficado:    Y para el graficado del arbol como tal se usa lo siguiente: | **¿Qué debemos saber del código fuente?** |

|  |  |
| --- | --- |
| Para el desarrollo de este Software utilizamos la  metodología SCRUM, la cual nos ayudo a tener un  mejor orden, definir fechas limites y tambien el poder definir tareas para completar el proyecto con éxito.  En este mismo archivo también incluímos la sección de  pruebas o metacognición, en donde fuimos detallando la información de las pruebas que ibamos realizando, sus respectivas fechas, el status, y cual fue el resultado con información de que se implemento para arreglar dicho problema si hubo alguno.   Para ver el archivo excel, por favor descarguelo: | **Anexos** |

Como último punto, el proyecto fue desarrollado en conjunto utilizando GitHUB. Todo cambio luego de ser probado se iba implementando en el proyecto Main. La razón de hacerlo de esta forma era para que se tuviera mejor visibilidad de los cambios y para más facilidad para el manejo de este. El link que contiene el código fuente es:

[**https://github.com/ArmandoS98/Proyecto-2---rbol-sint-ctico-y-postfix**](https://github.com/ArmandoS98/Proyecto-2---rbol-sint-ctico-y-postfix)