Text

Description automatically generated with medium confidence

**Instituto Tecnológico de Costa Rica**

**Campus Tecnológico Cartago**

**Escuela de Ingeniería en Computación**

**IC 5701**

**Grupo 60**

**Proyecto II**

**Profesor:** Allan Dávila Rodríguez

Estudiantes:

Javier Enrique Vásquez Monsalve / 2020215189

David Keanu Achoy Yakimova / 2020053336

**I semestre, 2023**

**Fecha de entrega:** 24 / 05 /2023

Contenido

[Manual de usuario 2](#_Toc136293663)

[Pruebas de funcionalidad 4](#_Toc136293664)

[Descripción del problema 6](#_Toc136293665)

[Diseño del programa 7](#_Toc136293666)

[Librerías utilizadas 9](#_Toc136293667)

[Análisis de resultados 10](#_Toc136293668)

[Bitácora 11](#_Toc136293669)

# Manual de usuario

**Nueva implementación**: botón “Borrar” de la sección de carga de archivos. **Funcionalidad:** Borra el texto o el archivo que se haya escrito o cargado en la casilla de texto de la sección de abrir archivo.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**Ejemplo:**

**Nueva implementación:** Generador de código de 3 direcciones.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**Funcionalidad:** Después de hacer el análisis sintáctico, presionar el botón “Código intermedio e imprime el código intermedio de dicho código. También cuenta con el botón de borrado que realiza la misma función que el botón que se explicó anteriormente, solo que esta vez borra la casilla de texto de la sección de código intermedio.

Ejemplo: Esto solo es posible después de realizar el análisis sintáctico

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

# Pruebas de funcionalidad

Ejemplo 1:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ejemplo 2:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteEjemplo 3:

# Descripción del problema

El problema por resolver en este proyecto consiste en la implementación de un Analizador Semántico y un Generador de Código Intermedio para un lenguaje de programación que se describe mediante una gramática EBNF y que tiene un Lexer y Parser generados previamente en el Proyecto I. El lenguaje está compuesto por una secuencia de declaraciones de funciones que contienen diferentes expresiones, y todo programa debe tener exactamente un método main.

El objetivo del proyecto es tomar un archivo fuente del lenguaje y realizar las siguientes tareas:

a) preservar y corregir los alcances del proyecto anterior,

b) verificar si el archivo fuente puede ser generado por la gramática del lenguaje, teniendo en cuenta la gramática, sintaxis y semántica (tipado explícito y fuerte),

c) reportar los errores léxicos, sintácticos y semánticos encontrados, utilizando la técnica de recuperación en modo pánico, y

d) escribir en un archivo el código intermedio correspondiente al archivo fuente con el mismo sentido semántico.1

Para lograr esto, se utilizará la gramática EBNF descrita en el Proyecto I como base para el Proyecto II, y se utilizarán los archivos JFlex y CUP generados en el Proyecto I. Se implementará el Analizador Semántico utilizando la herramienta CUP, lo que incluirá la comprobación semántica y la recuperación de errores. Además, se desarrollará un Generador de Código Intermedio que escriba en un archivo el código intermedio correspondiente al archivo fuente.

# Diseño del programa

Para realizar la generación de código intermedio (código de 3 direcciones) se implementó el algoritmo visto en clase.

Texto, Carta

Descripción generada automáticamente

Para esto se utilizaron 2 variables y una función que nos permite imprimir el código intermedio generado:

Variables:

Imagen que contiene Gráfico

Descripción generada automáticamente

Función de impresión:

La primera función nos permite imprimir el código en consola, esto nos ayudó para realizar pruebas. El segundo condigo se utilizó para poder imprimir el código generado en la interfaz.

Texto

Descripción generada automáticamente

Parar realizar la validación semántica se utilizó el siguiente algoritmo, se utilizó la tabla de símbolos como apoyo para poder realizar el respectivo análisis semantico:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

# Librerías utilizadas

* **java.awt.Color**
* **java.io.BufferedReader**
* **java.io.File**
* **java.io.FileNotFoundException**
* **java.io.FileReader**
* **java.io.IOException**
* **java.io.PrintWriter**
* **java.io.Reader**
* **java.io.StringReader**
* **java.nio.file.Files**
* **java.nio.file.Path**
* **java.nio.file.Paths**
* **java.util.ArrayList**
* **java.util.logging.Level**
* **java.util.logging.Logger**
* **java\_cup.runtime.Symbol**
* **javax.swing.JFileChooser**
* **Jflex.jar**
* **java-cup.11a.jar**
* **java\_cup.jarS**

# Análisis de resultados

|  |  |  |
| --- | --- | --- |
| Objetivo | Avance | Razón |
| Preservar y corregir los alcances del Proyecto I | 100% | Se agregó la tabla de símbolos |
| Indicar si el archivo fuente puede o no ser generado por la gramática. | 5% | No se logró implementar |
| Reportar los errores léxicos, sintácticos y semánticos encontrados. Recuperación en Modo Pánico. | 50% | Se implementó a medias |
| Escribir en un archivo el código intermedio para el archivo fuente con el mismo sentido semántico. | 40% | Se implementó a medias |

# Bitácora

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Imagen que contiene Aplicación

Descripción generada automáticamente

Forma

Descripción generada automáticamente