Instituto Tecnológico de Costa Rica

Escuela de Computación

Ingeniería en computación

Compiladores e Intérpretes

Grupo 2

Profesor: Erika Marín

Análisis Semántico y Generación de código

Josué Arrieta Salas, 2014008153

Adrián López Quesada, 2014081634

Cartago

Miércoles 22 de junio

**Índice**

[Análisis de resultados 3](#_Toc453140567)

[Casos de pruebas 4](#_Toc453140568)

[Prueba 1: Funciones 4](#_Toc453140569)

[Prueba 1.1 – Primeras pruebas con Funciones 4](#_Toc453140570)

[Prueba 1.2 – Segundas pruebas con Funciones 4](#_Toc453140571)

[Prueba 2: Expresiones 5](#_Toc453140572)

[Prueba 2.1 – Primeras pruebas con Expresiones 5](#_Toc453140573)

[Prueba 2.1 – Segundas pruebas con Expresiones 6](#_Toc453140574)

[Prueba 3: Declaración de variables 7](#_Toc453140575)

[Prueba 4: Asignaciones 9](#_Toc453140576)

[Prueba 5: Print e Input 9](#_Toc453140577)

[Prueba 6: While 10](#_Toc453140578)

[Prueba numero 7: If 11](#_Toc453140579)

[Prueba numero 8: For (Adrián) 12](#_Toc453140580)

[Prueba numero 9: Try 13](#_Toc453140581)

[Prueba numero 10: Break - Continue (Adrián) 14](#_Toc453140582)

[Prueba numero 11: OOP 14](#_Toc453140583)

[Prueba 11.1 – Estructura Básica 14](#_Toc453140584)

[Prueba 11.2 – El ; 15](#_Toc453140585)

[Gramática 17](#_Toc453140586)

[¿Cómo compilar y correr el Parser? 23](#_Toc453140587)

# Análisis de resultados

|  |  |
| --- | --- |
| **Objetivo** | **Porcentaje de éxito (100%)** |
| Programa recibe código fuente escrito en Python y analiza el archivo. | 100% |
| Se listan errores léxicos encontrados (por línea y el error). | 100%: objetivo del proyecto pasado. |
| El programa ante un error léxico se recupera de este y no despliega errores en cascada. | 100%: objetivo del proyecto pasado. |
| Se despliega lista de errores sintácticos: despliega línea, columna, y el token incorrecto. | 100% |
| Se manejan mensajes específicos para reportar errores sintácticos. | 100% |
| El programa ante un error sintáctico se recupera de este y no despliega errores en cascada. | 100%: se recupera en gran medida; sin embargo se ha de mencionar que en algunos casos es simplemente imposible recuperarse del error y continuar parseando. |
| Se parsean correctamente declaraciones de funciones. | 100% |
| Se parsean correctamente declaraciones de variables de tipo: int, float, list, string, boolean y char. | 100% |
| Se procesa correctamente la estructura de un programa funcional. | 100% |
| Se parsea correctamente la estructura de un programa orientado a objetos. | 100%: se toma en cuenta además, que debe venir al menos un atributo y un método para este tipo de archivos. |
| Se parsean correctamente todo tipo de expresiones: aritméticas y booleanas. Una llamada a función también se considera una expresión. | 100%: también se considera que las asignaciones son expresiones. |
| Se parsean correctamente todo tipo de asignaciones. También son válidas la creaciones de clases. | 100% |
| Se define la estructura de la función print e input. | 100% |
| Se parsean correctamente las estructuras de control: if, while y for. | 100% |
| En un bloque se pueden usar las sentencias break y continue. | 100% |
| Se parsea correctamente la estructura de control de errores: Try-Except-Finally. | 100% |

# ¿Cómo compilar y correr el Parser?

Debido a problemas para enviar el archivo por Gmail, se tuvieron que modificar las extensiones .jar a .jara; por lo tanto se debe eliminar esta “a” extra en los siguientes archivos:

Si se desea correr el ejecutable:

* Compilador-mypy\Ejecutable\Compilador.jara
* Compilador-mypy\Ejecutable\lib\ java-cup-11b.jara
* Compilador-mypy\Ejecutable\lib\ jflex-1.6.1.jara

Si se desea compilar el proyecto:

* Compilador-mypy\CodigoFuente\Tools\ java-cup-11b.jara
* Compilador-mypy\CodigoFuente\Tools\ jflex-1.6.1.jara

Para correr el parser se debe usar el archivo Prueba.mypy en la ubicación:

* Compilador-mypy\Ejecutable\
* Compilador-mypy\CodigoFuente\

Según se ejecute el ejecutable o se quiera recompilar el proyecto, de manera respectiva.

Notas extras: Se desarrolló el proyecto en el IDE Netbeans y se utilizaron las librerías Java Cup y JFlex para su desarrollo. Se implementó una interfaz gráfica para facilitar la ejecución del ejecutable, ya que en consola solo se puede correr desde línea de comando. La gramática quedo de manera extensa debido a problemas con encontrados con la recuperación de errores de la librería Java Cup., por lo que se intentó ser lo más específico posible.