**TECNOLÓGICO DE COSTA RICA**

**Escuela de Ingeniería en Computación**

**Proyecto de Ingeniería de Software**

**Profesora:**

María Estrada Sánchez

**Entrega 3:**

Analizador Contextual:

Manual Técnico.

**Estudiantes:**

Christian León Guevara - 2013371982

Gabriel Ramírez Ramírez - 201020244

**Fecha de entrega:**

20-01-2019

**Período Verano**

**Cartago**

**Tabla de contenido**

[**1.** **Introducción.** 3](#_Toc535818829)

[**2.** **Ámbito del Sistema.** 3](#_Toc535818830)

[**3.** **Arquitectura del Sistema.** 3](#_Toc535818831)

[**4.** **Pruebas.** 3](#_Toc535818832)

[**5.** **Reflexión.** 3](#_Toc535818833)

[**6.** **Apéndices.** 3](#_Toc535818834)

1. **Introducción.**
2. **Visión general**

En el proyecto del Analizador Contextual se hace una reimplementación del compilador del lenguaje Triángulo que originalmente se encuentra escrito en Java. Está nueva reimplementación se hizo en una versión del lenguaje funcional ML específicamente en OCaml.

El producto es un sistema técnico conformado por varios componentes que coexisten y colaboran entre ellos. Dentro de está estructura de componentes nosotros hemos desarrollado el Analizador Contextual.

Una de las características importantes del proyecto es que se necesita interactuar con otros componentes del sistema, por esta razón se hace uso de interfaces que tendrán la definición de los ASTs para la comunicación e interacción de los componentes.

1. **Definición del problema**

Existe una implementación de un compilador para el lenguaje de programación Triangulo utilizado por el profesor Ignacio Trejos Zelaya en el curso ‘Compiladores e intérpretes’, de la carrera de Ingeniería en Computación en el TEC. El compilador es de tres pasadas y se encuentra escrito en Java.

Lo que se realizó es una reimplementación del componente del Analizador Contextual escrita en el lenguaje de programación OCaml. Dicho Analizador debe interactuar con otros componentes existentes (Analizador Sintáctico y Generador de Código) comunicándose a través de interfaces para los ASTs que se generan en las distintas fases del proceso de compilación.

1. **Justificación**

Este proyecto surge con el propósito de utilizarse con fines académicos por parte del profesor Ignacio Trejos Zelaya en el curso Compiladores e Intérpretes de la carrera Ingeniería en Computación en el TEC.

Se busca que estudiantes a futuro puedan desarrollar otros componentes del compilador y llegar a desarrollar un compilador completo para el lenguaje Triángulo.

1. **Ámbito del Sistema.**
2. **Objetivos específicos**
3. Comprender el funcionamiento de los distintos lenguajes de programación que se usan en el proceso de desarrollo del componente Analizador Contextual.
4. Analizar el funcionamiento que tiene compilador para el lenguaje Triángulo escrito en Java.
5. Definir el diseño, algoritmos y estructura para el manejo de la tabla de identificación.
6. Diseñar casos de prueba correctos e incorrectos para validar el correcto funcionamiento del componente Analizador Contextual.
7. Documentar los distintos elementos relacionados con el proyecto: pruebas, documentación técnica y manuales.
8. Realizar la integración de Analizador Sintáctico con el componente del Analizador Contextual.
9. **Objetivo General**

Construir un sistema técnico en OCaml que se acople a una estructura de componentes de un compilador y permita realizar el análisis contextual lenguaje Triángulo y se pueda integrar con el Analizador Sintáctico desarrollado en otra fase del proyecto.

1. **Criterios de Éxito**
2. Poder utilizar los lenguajes Ocaml y Triángulo de manera fácil, comprendiendo su sintaxis y estructuras.
3. Comprender correctamente como funciona el compilador de Triángulo que está desarrollado en Java.
4. Imprimir el contenido de los ASTs para ver la estructura que tienen los distintos casos de prueba desarrollados.
5. Imprimir el contenido de la tabla de identificación.
6. Recorrer el AST generado en el Analizador Sintáctico y probar las distintas combinaciones generadas.
7. Reportar errores generados en la fase del análisis contextual.
8. Demostrar mediante las pruebas de validación y verificación el correcto funcionamiento del Analizador Contextual.
9. Que la integración entre el componente Sintáctico y el Contextual sea correcto y funcione sin ningún problema dentro de la estructura del compilador.
10. **Funcionamiento**

Se muestra una tabla con los recursos utilizados y que son necesarios para el desarrollo del proyecto.

|  |  |  |
| --- | --- | --- |
| **Recurso** | **Versión** | **Explicación** |
| Ocaml | 4.02.3 | Es el lenguaje de programación utilizado para el desarrollo del componente del Analizador Contextual. |

Sitio web para la obtención del recurso: https://caml.inria.fr/download.en.html

1. **Arquitectura del Sistema.**

La arquitectura del compilador para el lenguaje Triángulo se basa en una estructura de componentes que pueden interactuar entre sí mediante el uso de interfaces. Por eso existen elementos propios del análisis contextual y otros elementos comunes con otros componentes.

Análisis Contextual:

1. IdEntry.mli: Es una interfaz que define la estructura de cada entrada de la tabla de identificación la cuál es una terna: identificador, ast y nivel.
2. IdentifacationTable.ml: Define la estructura y algoritmos necesarios para la manipulación de la tabla de identificación.
3. Checker.ml: Contiene todas las funcionalidades que debe realizar el analizador contextual.

Elementos Comunes:

1. ErrorReporter.ml: Contiene las funcionalidades para el manejo y reporte de errores.
2. IdentificationTablePrinter\_XML.ml: Contiene todas las funcionalidades que permiten el recorrido y escritura de la tabla de identificación generada el análisis contextual.
3. TreeDrawer.ml: Es un elemento compartido entre el análisis sintáctico y contextual que permite la escritura de los ASTs en formato XML.
4. Ast.mli: Interfaz que establece la estructura del AST para la comunicación entre el Analizador Sintáctico y Contextual.
5. **Pruebas.**
6. **Reflexión.**
7. **Apéndices.**