Grupo: 10

Fecha de entrega: 7 de junio de 2024

Curso : Brasburg, Cano, Raik

Integrantes:

Padrón	**Apellido y nombre**
110845	Cardoso Landaburu Juan Gabriel
110598	Almanza Lucia
110736	López Facundo Gabriel
111019	Fernández Ignacio Agustin

Introducción:

El presente informe tiene como objetivo presentar las hipótesis y las decisiones tomadas en el trabajo práctico número dos de la asignatura Paradigmas de la programación. El proyecto consistió en la implementación de un intérprete de Cálculo Lambda no tipado, aplicando los conceptos vistos en las clases de programación funcional. El desarrollo de este intérprete tiene como finalidad analizar el problema planteado correctamente, demostrar el conocimiento de los conceptos del cálculo lambda y seguir las buenas prácticas de programación.

Hipótesis:

A partir de las consignas proporcionadas hemos formulado las siguientes hipótesis para la implementación del programa:

- Los paréntesis sólo serán para aplicaciones.
- El input del usuario siempre será correcto.?
- El programa termina ejecucion para el input “exit”

Diseño:

Para implementar nuestro intérprete, dividimos el proyecto principalmente en tres componentes principales:

Lexer: Recibe como parámetro un ***String*** que representa una expresión lambda. Para poder interpretar correctamente esto, tenemos un *Trait* llamado Token que tiene asignado los distintos símbolos característicos del cálculo lambda y de esta forma permite convertir el input en una expresión lambda .

Parser: Posteriormente a que el lexer transforme el string en un *List* de Tokens, el Parser se encarga de recibir esta lista y transformarla en un *Árbol de Sintaxis.* Así mismo, el Parser debe poder recibir un *AST* y devolver la expresión lambda a modo de *String* es por eso que se encuentra implementado el método ***unparser.*** Volvimos a definir un *Sealed Trait* llamado AST para definir las expresiones recibidas. las cuales pueden ser *Var, Abstr y App.*

Reductor: toma una expresión lambda representada como un árbol de sintaxis abstracta (AST) y aplica las reglas de reducción correspondientes según la estrategia seleccionada por el usuario. Para ello aplica los métodos reducción β y sustitución α.

Conclusiones:

Concluimos que logramos implementar un intérprete de Cálculo Lambda no tipado que cumple con todas las funcionalidades mínimas requeridas. El intérprete puede interpretar expresiones lambda, analizar su estructura sintáctica y reducirlas utilizando dos estrategias distintas: Call-by-name y Call-by-value. También es capaz de identificar y devolver las variables libres de una expresión lambda. La implementación se realizó utilizando exclusivamente el paradigma de programación funcional, lo cual permitió aplicar y reforzar los conceptos aprendidos en clase y cumplir con los objetivos mencionados.