

Paradigmas de la Programación

Informe Laboratorio nro. 1

Paradigma Funcional

*Profesor: Gonzalo Martínez V.*

*Alumna: Karina Bustamante H.*

30 de junio de 2023

Tabla de contenido

[Introducción 2](#_Toc134312314)

[Descripción del problema 2](#_Toc134312315)

[Descripción del paradigma 3](#_Toc134312316)

[Análisis del problema 3](#_Toc134312317)

[Diseño de la solución 6](#_Toc134312318)

[Consideraciones de implementación 6](#_Toc134312319)

[Instrucciones de uso 7](#_Toc134312320)

[Resultados y evaluación 8](#_Toc134312321)

[Conclusiones 9](#_Toc134312322)

[Referencias 10](#_Toc134312323)

# Introducción

En el presente trabajo, se aborda el desarrollo del laboratorio Nº1 de la asignatura de Paradigmas de la Programación. En este proyecto, se nos ha encomendado la tarea de implementar un algoritmo en el lenguaje de programación Scheme, el cual se basa en la programación declarativa - funcional. El objetivo principal consiste en crear un simulador de sistema operativo que se centre específicamente en un sistema de archivos simplificado y los comandos necesarios para operar en él.

Para llevar a cabo esta implementación, hemos utilizado el lenguaje de programación Scheme a través de la aplicación DrRacket en su versión 8.8. Con esta herramienta, buscamos proporcionar una solución integral a los requerimientos funcionales planteados. Nuestra propuesta de solución permitirá al usuario realizar diversas acciones sobre este sistema de archivos ficticio, como crear, buscar, listar, entre otras.

El desarrollo de este proyecto no solo nos permitirá adquirir conocimientos prácticos sobre la programación en Scheme, sino también explorar los fundamentos de la programación declarativa - funcional y su aplicación en el contexto de un sistema operativo simplificado. A través de este trabajo, esperamos demostrar nuestra capacidad para diseñar algoritmos eficientes y crear soluciones robustas en el ámbito de la programación paradigmática.

A continuación se presenta en detalle el enunciado del problema, la metodología utilizada, la estructura del programa implementado y los resultados obtenidos.

# Descripción del paradigma

El paradigma funcional se basa en la composición de funciones que se fundamentan en principios de funciones matemáticas puras. En este enfoque, una función se considera determinista, lo que implica que siempre producirá el mismo resultado cuando se le proporcionen los mismos argumentos. Además, se enfoca en la evaluación de expresiones y la transformación de datos, evitando efectos secundarios y dependencia del estado global del programa. En su lugar, una función debe depender únicamente de sus entradas y generar una salida basada en ellas.

Podríamos establecer una analogía entre el paradigma funcional y el álgebra, donde las ecuaciones pueden ser reemplazadas por sus resultados, lo que se conoce como transparencia referencial. En lugar de utilizar bucles, se recurre a la definición de funciones recursivas, que se refieren a una función expresada en función de sí misma. Este enfoque permite resolver problemas dividiéndolos en subproblemas más pequeños.

La inmutabilidad de los datos es otra característica fundamental de la programación funcional. Esto implica que una vez que se ha creado un valor, no se puede modificar. En lugar de modificar los datos existentes, se generan nuevas estructuras de datos a partir de las existentes mediante operaciones de transformación.

Dentro de las técnicas de programación funcional, se destacan conceptos como el cálculo lambda, que utiliza la notación prefija donde el operador precede a los operandos. Asimismo, se emplean funciones de alto orden, que permiten que las funciones se utilicen como argumentos de entrada o se devuelvan como resultados. La currificación es otra técnica relevante, que consiste en transformar una función de varios argumentos en una secuencia de funciones de un solo argumento, lo que facilita la evaluación de la función paso a paso.

# 

# Descripción del problema

La descripción del problema consiste en desarrollar la simulación de un sistema de archivos utilizando el lenguaje de programación Scheme. Este proyecto tiene como objetivo crear una estructura jerárquica que permita organizar y almacenar información en un dispositivo de almacenamiento, como un disco duro.

Para lograr esto, es necesario definir un conjunto de requerimientos funcionales basados en el paradigma funcional. Estos requerimientos deben permitir llevar a cabo la simulación, brindando a los usuarios la capacidad de realizar diversas acciones dentro del sistema de archivos.

Entre las acciones que los usuarios podrán llevar a cabo se encuentran: crear y agregar unidades y usuarios, crear, modificar y eliminar archivos o carpetas según los permisos asignados, generar rutas de acceso a los archivos y carpetas, cambiar de ruta, crear usuarios y controladores de dispositivos, cambiar de unidad, e iniciar y cerrar sesión en el sistema.

En resumen, el proyecto busca desarrollar una simulación de sistema de archivos en Scheme, con el objetivo de proporcionar a los usuarios una interfaz funcional que les permita interactuar con el sistema de archivos y realizar diversas operaciones de administración y manipulación de archivos y carpetas.

# Análisis del problema

El problema principal se centra en el diseño y la implementación de algoritmos necesarios para simular un sistema de archivos eficiente y funcional basados en el paradigma funcional y utilizando la herramienta Scheme. Durante el análisis del problema, se deben tener en cuenta las siguientes consideraciones clave:

***Jerarquía de directorios***: El sistema de archivos debe permitir la creación y gestión de una estructura jerárquica de directorios. Cada directorio puede contener archivos y subdirectorios, y el sistema debe proporcionar la capacidad de navegar a través de esta estructura jerárquica.

***Creación y eliminación de archivos y directorios:*** El sistema debe permitir la creación y eliminación de archivos y directorios. Al crear un archivo, se debe especificar su nombre y ubicación en el sistema de archivos. Al eliminar un archivo o directorio, se deben eliminar todos sus contenidos y actualizar la estructura del sistema de archivos en consecuencia.

***Gestión de permisos y protección***: El sistema de archivos debe implementar un sistema de gestión de permisos que controle el acceso a los archivos y directorios. Se deben definir niveles de acceso, como lectura y escritura, y garantizar el cumplimiento de los permisos definidos.

Al abordar estos aspectos clave, nuestra simulación de un sistema de archivos en el paradigma funcional y con el software de Scheme proporcionará una herramienta funcional y útil para gestionar la organización y manipulación de archivos y directorios. En la siguiente sección, se detallará el diseño e implementación del sistema de archivos en Scheme.

# Diseño de la solución

# La simulación del sistema de archivos se diseñará e implementará utilizando el lenguaje de programación Scheme. A continuación, se detallan los principales elementos del diseño y la implementación:

1. **Definición de estructuras de datos**: Es necesario definir las estructuras de datos que representarán los archivos, directorios y permisos en el sistema de archivos. Por ejemplo, se pueden utilizar listas anidadas para representar la estructura jerárquica de directorios y archivos, y se pueden utilizar listas asociativas para almacenar información sobre los permisos de acceso.
2. **Funciones para la creación y eliminación de archivos y directorios**: Se deben implementar funciones que permitan la creación y eliminación de archivos y directorios en el sistema de archivos. Estas funciones deben tener en cuenta la estructura de directorios existente y actualizarla correctamente al realizar estas operaciones.
3. **Funciones de navegación y búsqueda**: Se deben desarrollar funciones que permitan la navegación a través de la estructura jerárquica de directorios, permitiendo al usuario moverse entre directorios, acceder a archivos y subdirectorios, y realizar búsquedas específicas en el sistema de archivos.
4. **Gestión de permisos y control de acceso**: Se deben implementar funciones que permitan la gestión de permisos y control de acceso a los archivos y directorios. Estas funciones deben verificar los permisos asignados a cada usuario y permitir o denegar el acceso según corresponda.
5. **Funciones auxiliares**: Pueden ser necesarias funciones auxiliares para realizar tareas adicionales, como la actualización de la estructura del sistema de archivos, la validación de nombres de archivos o directorios, la manipulación de rutas de acceso, entre otros.

Es importante tener en cuenta los principios del paradigma funcional al diseñar la solución, como la inmutabilidad de los datos y el enfoque en funciones puras que no tienen efectos secundarios. El diseño de la solución debe tener en cuenta estos elementos y asegurarse de que el sistema de archivos simulado cumpla con los requerimientos funcionales definidos previamente.

# Consideraciones de implementación

# Instrucciones de uso

# Resultados y evaluación

A continuación se muestra la autoevaluación de los requerimientos funcionales requeridos por el laboratorio, y el grado de logro alcanzado para cada uno de ellos.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| RF Nº | Requerimiento funcional | Grado de alcance | Tipo o cantidad de pruebas | Éxitos | Fracasos | Razones de fallo |
| 1 | TDAs | 1 |  |  |  |  |
| 2 | system - constructor | 1 |  |  |  |  |
| 3 | system - addDrive | 1 |  |  |  |  |
| 4 | system - register | 1 |  |  |  |  |
| 5 | system - login | 1 |  |  |  |  |
| 6 | system - logout | 1 |  |  |  |  |
| 7 | system - switch-drive | 1 |  |  |  |  |
| 8 | system - mkdir |  |  |  |  |  |
| 9 | system- cd |  |  |  |  |  |
| 10 | system - add-file |  |  |  |  |  |
| 11 | system - del |  |  |  |  |  |
| 12 | system - copy |  |  |  |  |  |
| 13 | system - move |  |  |  |  |  |
| 14 | system - ren |  |  |  |  |  |

**Tabla 1.-** Muestra los requerimientos funcionales y el nivel de logro.

# 

# Conclusiones

# 

# Referencias