Algoritmos y Estructuras de Datos
Ciclo 1, 2024

Proyecto 1
Fase 1 Intérprete de LISP básico

Joel Jaquez - 23369
Mario Rocha - 23501
Hugo Barillas - 23306

Enlace hacia el Github con los programas básicos:
https://github.com/jaq23369/Fase-1-Int-rprete-de-LISP-b-sico

Investigación de Lisp

-Historia

Lisp fue inventado por John McCarthy en 1958 mientras estaba en el Instituto Tecnológico de Massachusetts (MIT). McCarthy publicó su diseño en 1960 en un artículo de Communications of the ACM titulado "Funciones recursivas de expresiones simbólicas y su cómputo a máquina, Parte I". Lisp es el segundo lenguaje de programación de alto nivel de mayor antigüedad; apareció un año después de FORTRAN y uno antes que COBOL.

Lisp fue creado originalmente como una notación matemática práctica para los programas de computadora, basada en el cálculo lambda de Alonzo Church. Se convirtió rápidamente en el lenguaje de programación favorito en la investigación de la inteligencia artificial (AI). Como lenguajes de programación precursor, Lisp fue pionero en muchas ideas en ciencias de la computación, incluyendo las estructuras de datos de árbol, el manejo de almacenamiento automático, tipos dinámicos, y el compilador autocontenido.

El acrónimo LISP significa “LISt Processor” (‘Procesamiento de listas’). Las listas encadenadas son una de las estructuras de datos importantes de Lisp, y el código fuente de Lisp en sí mismo está compuesto de listas. Como resultado, los programas de Lisp pueden manipular código fuente de Lisp como si fueran simples datos, dando lugar a sistemas de macros que permiten a los programadores crear lenguajes de dominio específico embebidos en Lisp.

Hoy, los dialectos de Lisp más ampliamente usados son Scheme (1975), Common Lisp (1984), Emacs Lisp (1985) y Clojure (2007).

-Características

Lisp es un lenguaje de programación con características únicas y distintivas:

- ### Sintaxis homoicónica

  Lisp tiene una sintaxis basada en la notación polaca, que es inconfundible y útil. Todo el código del programa se escribe como expresiones S, o listas entre paréntesis.

- ### Manejo de memoria automático

  Lisp posee un manejo de memoria automático que libera el espacio utilizado por los objetos que dejan de ser necesitados.

- Evaluación perezosa de expresiones

  Lisp incluye un mecanismo bastante simple para utilizar evaluación perezosa de expresiones.

- Extensibilidad

  Lisp permite actualizar los programas de forma dinámica y proporciona una fácil extensibilidad.

- Depuración de alto nivel

  Lisp proporciona depuración de alto nivel.

- Programación avanzada orientada a objetos

  Lisp proporciona programación avanzada orientada a objetos.

- Sistema macro conveniente

  Lisp proporciona un sistema macro conveniente

-Usos

- **Inteligencia Artificial** (IA): Lisp se convirtió rápidamente en el lenguaje de programación favorito en la investigación de la inteligencia artificial. Su capacidad para calcular con expresiones simbólicas en lugar de números lo hace conveniente para aplicaciones de IA.
- **Procesamiento de listas**: Como su nombre indica (LISt Processor), Lisp fue diseñado para una fácil manipulación de listas de datos. Esto lo hace útil para tareas que implican el procesamiento de grandes cantidades de datos estructurados.
- **Metaprogramación**: Los programas de Lisp pueden manipular código fuente de Lisp como si fueran simples datos, dando lugar a sistemas de macros que permiten a los programadores crear lenguajes de dominio específico embebidos en Lisp.
- **Investigación académica**: Lisp ha sido ampliamente utilizado en numerosos proyectos de investigación en el MIT y otras instituciones académicas.
- **Desarrollo de software**: Lisp también se utiliza en el desarrollo de software, especialmente en áreas que requieren cálculos simbólicos y manipulación de datos

-Comparación con POO

Lisp y la Programación Orientada a Objetos (POO) son dos paradigmas de programación con características y usos distintos, pero no son mutuamente excluyentes. De hecho, Lisp ha sido influenciado por la POO y ha adoptado características de la misma.

Lisp es un lenguaje de programación multiparadigma que fue pionero en muchas ideas en ciencias de la computación, incluyendo las estructuras de datos de árbol, el manejo de almacenamiento automático, tipos dinámicos, y el compilador autocontenido. Los programas de Lisp pueden manipular código fuente de Lisp como si fueran simples datos, dando lugar

a sistemas de macros que permiten a los programadores crear lenguajes de dominio específico embebidos en Lisp.

Por otro lado, la Programación Orientada a Objetos (POO) es un paradigma de programación que parte del concepto de “objetos” como base, los cuales contienen información en forma de campos (a veces también referidos como atributos o propiedades) y código en forma de métodos. Algunas características clave de la POO son herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento.

Lisp ha sido influenciado por Smalltalk, un lenguaje de programación orientado a objetos, y ha adoptado características de la POO como clases, instancias, etc., a finales de los años 1970. Por ejemplo, Common Lisp Object System (CLOS) es una extensión de Common Lisp que agrega capacidades de programación orientada a objetos. Además, existen dialectos de Lisp como Clojure que incorporan características de la POO.

Investigación de Java Collections Framework

La jerarquía del Java Collections Framework se organiza principalmente en torno a interfaces que representan diferentes tipos de colecciones y las implementaciones concretas que proporcionan funcionalidades específicas. Aquí hay una descripción general:

Interfaces Principales:

1. **Collection:**

- La interfaz base que extiende `Iterable`. Define operaciones comunes a todas las colecciones, como añadir, eliminar, consultar tamaño y obtener un iterador.

2. **List:**

- Extiende `Collection` y representa una secuencia ordenada de elementos. Permite elementos duplicados y proporciona acceso por índice. Ejemplos de implementaciones son `ArrayList`, `LinkedList`.

3. **Set:**

- Extiende `Collection` y representa una colección que no permite elementos duplicados. Ejemplos de implementaciones son `HashSet`, `TreeSet`.

4. **Queue:**

- Extiende `Collection` y representa una colección diseñada para manejar elementos en un orden específico para operaciones de encolar y desencolar. Ejemplos de implementaciones son `LinkedList`, `PriorityQueue`.

5. **Map:**

- No extiende `Collection`, pero es parte del framework. Representa una colección de pares clave-valor, donde cada clave es única. Ejemplos de implementaciones son `HashMap`, `TreeMap`.

Implementaciones Concretas:

1. **ArrayList:**

- Implementa la interfaz `List`. Almacena elementos en un arreglo dinámico, permitiendo acceso rápido a través de índices.

2. **LinkedList**:

- Implementa las interfaces `List` y `Queue`. Almacena elementos en nodos enlazados, proporcionando inserciones y eliminaciones eficientes en cualquier posición.

3. **HashSet:**

- Implementa la interfaz `Set`. Almacena elementos sin duplicados utilizando una tabla hash para acceso rápido.

4. **TreeSet:**

- Implementa la interfaz `Set`. Almacena elementos en un árbol, manteniendo un orden natural o especificado por un comparador.

5. **HashMap:**

- Implementa la interfaz `Map`. Almacena pares clave-valor utilizando una tabla hash para búsquedas rápidas.

6. **TreeMap:**

- Implementa la interfaz `Map`. Almacena pares clave-valor en un árbol, manteniendo un orden natural o especificado por un comparador.

Estas implementaciones proporcionan diferentes características y rendimientos, lo que permite a los desarrolladores elegir la estructura de datos más apropiada según sus necesidades específicas. La jerarquía del framework facilita la estandarización de operaciones y la reutilización de código a través de interfaces comunes.

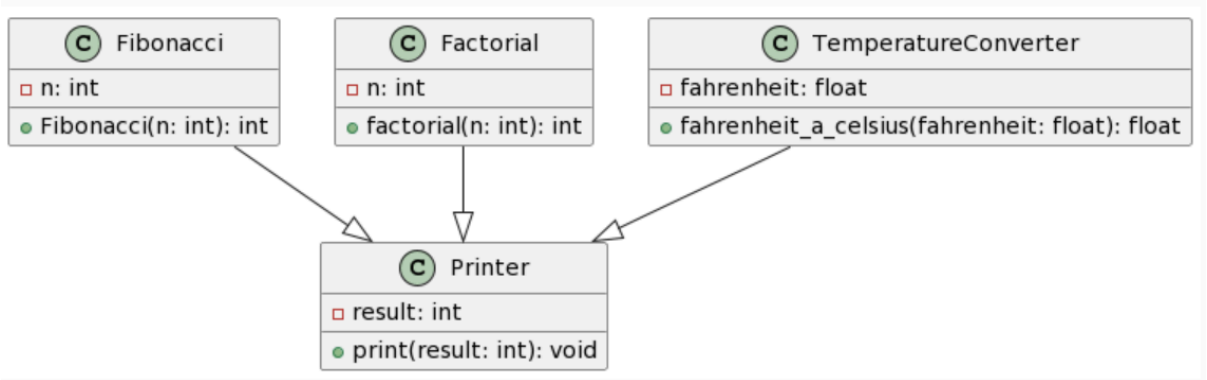
-usos en el proyecto

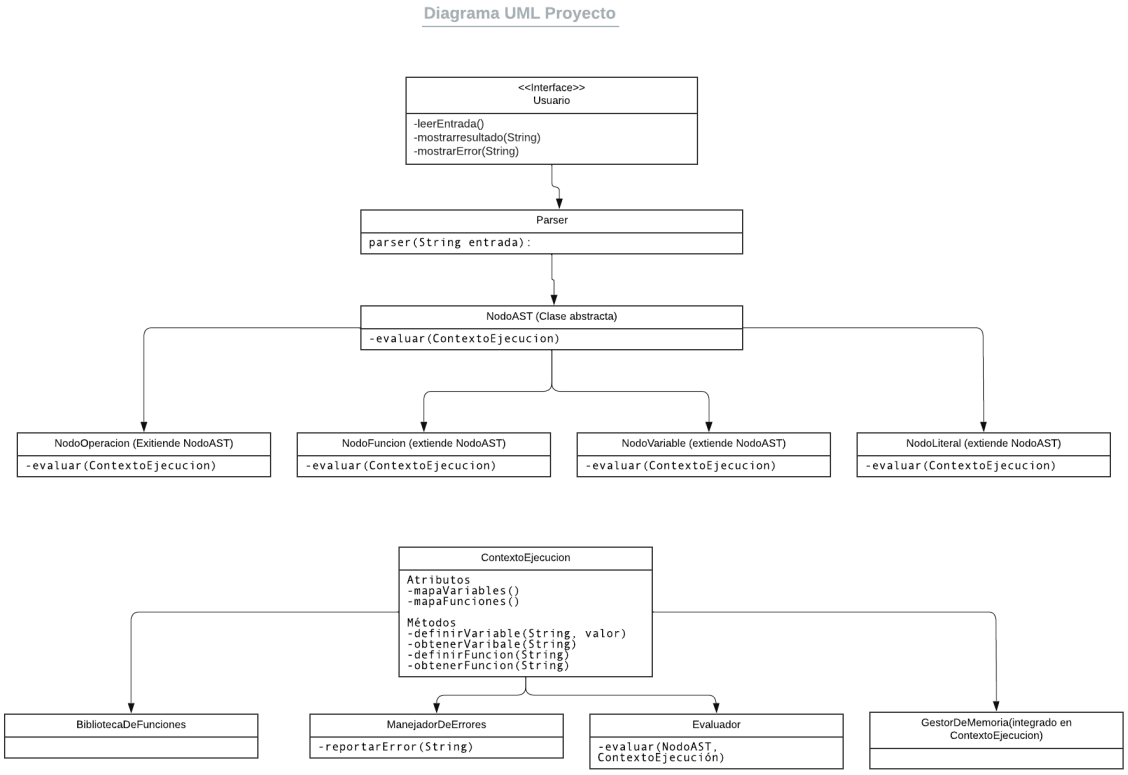
Ambiente de trabajo Lisp

Programa en Lisp

Lisp para la producción del término n de la serie de Fibonacci

Propio Lisp - Diagrama UML





Referencias

Elaboración del LISP: El lenguaje de programación para I. (2023, junio 4). Inteligencia Artificial. https://inteligenciaartificial.science/elaboracion-del-lisp-el-lenguaje-de-programacion-para-i/

Velasco, J. J. (2011, octubre 25). Historia de la tecnología: Lisp. Hipertextual. https://hipertextual.com/2011/10/historia-de-la-tecnologia-lisp

Reclu IT. (2020, octubre 13). Recluit.com. Recuperado el 4 de febrero de 2024, de https://recluit.com/que-es-lisp/

Academia Lab. (2024). Lisp (lenguaje de programación). Enciclopedia. Revisado el 4 de febrero del 2024. https://academia-lab.com/enciclopedia/lisp-lenguaje-de-programacion/
LISP - Descripción general. (2020, diciembre 10). Stack. https://isolution.pro/es/t/lisp/lisp-overview/lisp-descripcion-general
(1997, noviembre 20). Alonso Pardo. Recuperado el 4 de febrero de 2024, de https://www.grupocole.org/cole/library/html/Alo94/node152.html
recluit.com.(13 de octubre 2020). ¿Qué es LISP?https://recluit.com/que-es-lisp/#:~:text=LISP%2C%20acr%C3%B3nimo%20de%20list%20processing,de%20al%20menos%20un%20objeto.