Universidad Politécnica de Aguascalientes.
Ingeniería en Sistemas Computacionales.

Estructuras de Datos

ISC04A

"Calculadora AINH"

Alumno: Ángel Issac Núñez Hernández, UP230534.
Docente: Juan Carlos Herrera Hernández.

Fecha: 17/03/2025

Calculadora AINH

Este documento describe la arquitectura y la lógica detrás de la calculadora desarrollada por Ángel Issac Núñez Hernández. La calculadora se divide en varios módulos, cada uno con una responsabilidad específica en el proceso de calcular una expresión matemática ingresada por el usuario.

1. MCR (Interfaz Gráfica e Interacción con el Usuario)

El módulo MCR actúa como la interfaz principal de la calculadora. Utiliza la biblioteca tkinter para crear una ventana interactiva con botones para números, operadores y funciones. Su función principal es capturar la entrada del usuario a través de estos botones y mostrar los resultados. Cuando el usuario presiona el botón de igual (=), la expresión matemática completa que se muestra en la pantalla se envía al siguiente módulo, Tails, para su procesamiento inicial.

Puntos a destacar:

- La calculadora se divide en varias secciones, como se ve en la Imagen 1.
    - Numbers: Números del 0 al 9
    - Math: Operadores básicos matemáticos.
    - Functions: Funciones trigonométricas y logarítmicas.
    - Specials: Son elementos especiales para la calculadora.
    - Utilities: Son utilidades para el cálculo.
    - No tocar: Esta sección son “Easter eggs”.

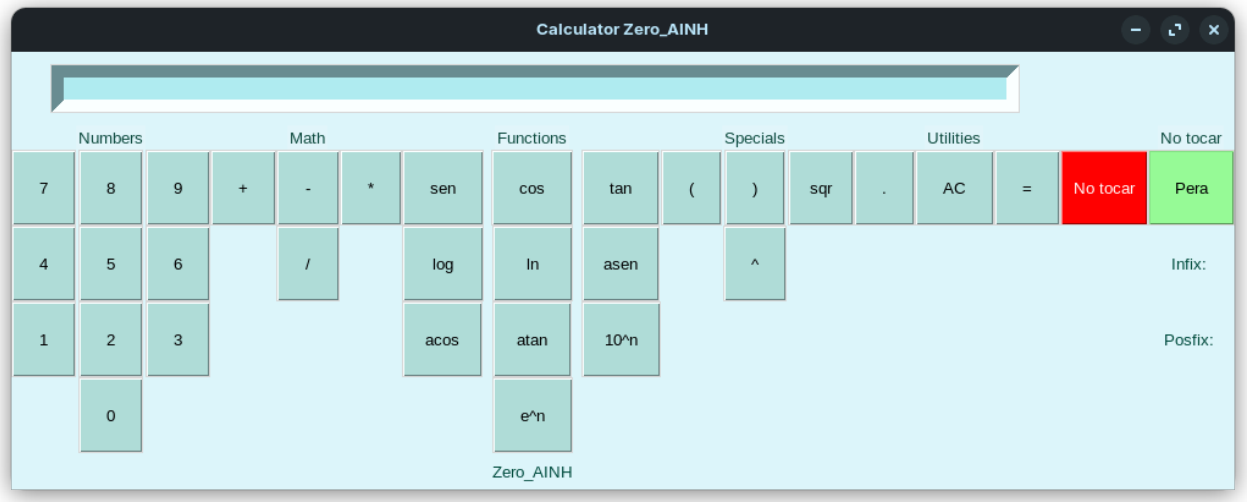

Imagen 1: Interfaz de la calculadora.

- El nombre hace referencia a Mecatrónica, el MCR en lógica de control se usa el término *”Master Control Rele”* el cual es el encargado que todo esté en sincronia.

2. Tails (Procesamiento Inicial de la Entrada y Gestión de la Cola)

El módulo Tails juega un papel crucial en la preparación de la entrada del usuario para las etapas posteriores de cálculo. Una vez que recibe la cadena de la expresión desde MCR, su principal objetivo es **tokenizar** esta cadena. La tokenización implica dividir la expresión en unidades más pequeñas y significativas, como números individuales, operadores matemáticos (+, -, *, /), funciones (sen, cos, log, etc.), paréntesis y otros símbolos.
Para gestionar estos tokens de manera ordenada, Tails implementa una estructura de datos de **cola**. Una cola es una estructura lineal que sigue el principio de "primero en entrar, primero en salir" (FIFO). Esto asegura que los tokens se procesen en el mismo orden en que fueron ingresados por el usuario.
La lógica principal de Tails reside en iterar a través de cada carácter de la expresión ingresada y decide cómo agruparlos en tokens.

- **Números:** Los dígitos consecutivos se combinan para formar números. Esto permite el ingreso de números de varias cifras y decimales.
- **Operadores Estándar:** Los operadores como +, *, / y ^ se identifican y se encolan como tokens individuales.
- **Funciones Especiales:** Las funciones como "sen", "cos", "tan", "log", "ln", "asen", "acos", "atan", "sqr" y "e" se reconocen como palabras completas y se encolan como un solo token.
- **Paréntesis:** Los paréntesis de apertura "(" y cierre ")" se identifican y se encolan para mantener la estructura de la expresión.

Puntos clave:

- El método examina cada carácter de la entrada. Mantiene un registro del último carácter procesado para tomar decisiones sobre cómo formar los tokens.
    - **Manejo de la Resta ('-'):** Tails distingue entre el operador de resta y un signo negativo que forma parte de un número. Si el carácter '-' aparece al principio de la expresión o inmediatamente después de un operador, se considera como parte de un número negativo y se adjunta al número que se está construyendo. Si aparece después de un número, se considera el operador de resta y se encola como un token separado. También se considera el caso especial de la notación científica con "e-".

    - **Manejo de la 'e' para Notación Exponencial:** Tails reconoce la letra 'e' como parte de la notación exponencial. Si se encuentra una 'e', se marca un estado interno. Si el siguiente carácter es un '-', se considera "e-" como parte de un número. Si la 'e' es seguida por '^', se reconoce como la función especial "e^" y se encola como un solo token.
    - Una vez que la expresión se ha tokenizado por completo, todos los tokens se encuentran en la cola de Tails, listos para ser procesados por el siguiente módulo.
- El nombre Tails hace referencia al personaje de Sega llamado *“Tails”* de la saga de videojuegos Sonic, este personaje se eligió, ya que trabaja de manera secuencial y además tiene varias colas.

3. Infix (Representación Conceptual de la Expresión)

Después de que Tails procesa la entrada, los tokens en la cola representan conceptualmente la expresión matemática en **notación infija**. La notación infija es la forma tradicional en que escribimos las expresiones matemáticas, con los operadores entre los operandos (por ejemplo, 2 + 3).

4. Infix2posfix (Conversión de Notación Infija a Postfija)

El módulo Infix2posfix toma la secuencia de tokens en notación infija (proveniente de Tails) y la convierte a **notación postfija**. En la notación postfija, los operadores aparecen después de sus operandos (por ejemplo, 2 3 +). Esta notación es más fácil de evaluar por una computadora utilizando una pila.
Infix2posfix utiliza una **pila** (proporcionada por el módulo Duracell) para almacenar temporalmente los operadores durante el proceso de conversión. También se define una **jerarquía de operadores** para determinar el orden en que deben aplicarse las operaciones. Por ejemplo, la multiplicación y la división tienen mayor precedencia que la suma y la resta.
Puntos clave:

- El algoritmo de conversión itera a través de los tokens de la expresión infija.
    - **Operandos:** Si el token es un número, se añade directamente a la secuencia de salida (la expresión postfija).
    - **Paréntesis de Apertura '(':** Se ingresa a la pila de operadores.

    - **Paréntesis de Cierre ')':** Se saca de operadores de la pila y se añaden a la secuencia de salida hasta que se encuentra un paréntesis de apertura en la cima de la pila. El paréntesis de apertura se saca, pero no se añade a la salida. Esto asegura que las operaciones dentro de los paréntesis se realicen primero.
    - **Operadores:** Si el token es un operador, se compara su precedencia con la precedencia del operador en la cima de la pila. Mientras la pila no esté vacía, el operador en la cima tenga una precedencia mayor o igual al operador actual (y no sea un paréntesis de apertura), se saca de la pila y se añade a la secuencia de salida. Luego, el operador actual se ingresa a la pila.
- **Manejo Específico de la Resta y 'e' en Infix2posfix:**
    - **'e^':** La función especial "e^" se trata como un operador con una precedencia específica (en este caso, alta, similar a otras funciones matemáticas). Cuando se encuentra, se ingresa a la pila de operadores y se saca en el momento adecuado según su precedencia.
- Al final del proceso de conversión, cualquier operador restante en la pila se saca y se añade a la secuencia de salida, formando la expresión completa en notación postfija.

5. Posfix (Representación de la Expresión en Notación Postfija)

Este módulo representa conceptualmente la expresión matemática ya convertida a notación postfija. En esta forma, el orden de las operaciones está explícitamente definido por la posición de los operadores.

6. Posfix2Value (Evaluación de la Expresión Postfija)

El módulo Posfix2Value toma la expresión en notación postfija y la evalúa para obtener el resultado final. Utiliza **pila** (también proporcionada por Duracell) para almacenar los operandos durante la evaluación.
El algoritmo de evaluación itera a través de los tokens de la expresión postfija.

- **Operandos:** Si el token es un número, se ingresa a la pila de operandos.
- **Operadores:** Si el token es un operador, se sacan de la pila el número necesario de operandos (uno para operadores unarios como "sen" o "log", y dos para operadores binarios como +, -, *, /). Se realiza la operación especificada y el resultado se ingresa de nuevo a la pila de operandos.

Después de procesar todos los tokens, el resultado final de la expresión se encontrará en la cima de la pila de operandos.

7. Duracell (Implementación de la Pila)

El módulo Duracell proporciona la implementación de la estructura de datos **pila** que se utiliza tanto en Infix2posfix como en Posfix2Value. Una pila es una estructura de datos lineal que sigue el principio de "último en entrar, primero en salir" (LIFO). Permite añadir elementos a la cima (push) y remover elementos de la cima (pop). También proporciona una forma de ver el elemento en la cima (top) sin removerlo y de verificar si la pila está vacía (isEmpty).

8. Extras

Esta sección describe funcionalidades adicionales y elementos especiales incluidos en la calculadora "*Easter eggs"*.

- **"Preso":** Es el primer "easter egg", si la función `infix2pos` del módulo `Infix2posfix` devuelve la cadena "Preso ", se muestra un mensaje específico en la pantalla: "Preso, de la cárcel de tus besos, de tu forma de hacer eso a lo que llamas amor". Este mensaje es un fragmento de la canción "*Preso"* de Jose Jose.
- **Botón "No tocar":** Este botón representa otro "easter egg" menos oculto dentro de la calculadora. Al hacer clic en él, se activa una secuencia mostrando la letra de la canción "*El triste"* de Jose Jose que se muestran en la pantalla con un retardo de tiempo entre cada uno. Finalmente, después de mostrar el último mensaje, la aplicación se cierra automáticamente. Esta funcionalidad se implementa utilizando el método `.after()` de `tkinter`. Se utiliza una variable de control llamada `self.time` para asegurar que esta secuencia se ejecute completa sin interrupciones.
- **Botón "Pera":** Similar al botón "No tocar", el botón "Pera" es otro "easter egg". Al hacer clic en él, se inicia una secuencia con la letra de la canción "*La nave del olvido"* de Jose Jose, mostrada con retardos de tiempo. Al igual que con el botón anterior, al final de la secuencia, la aplicación se cierra. También utiliza el método `.after()` para la implementación de los retardos y la secuencia de mensajes. La variable `self.time` también se utiliza aquí.

- **.after():** El método `.after()` es una función proporcionada por la biblioteca `tkinter` que permite ejecutar una función o un fragmento de código después de un cierto período de tiempo especificado en milisegundos. En el contexto de los botones "No tocar" y "Pera", se utiliza para crear la secuencia de eventos. La función `.after()` toma dos argumentos principales: el tiempo de retardo (en milisegundos) y la función (o una función anónima `lambda`) que se debe ejecutar después de ese tiempo. En estos casos, se utiliza para cambiar el texto mostrado en la pantalla de la calculadora después de intervalos definidos, creando así la narrativa secuencial antes de que la aplicación se cierre.

#AINH