Informe Tecnico Haskell 2

Jesús Isaac Barajas Castillo

1. Calculadora científica

Descripción del Código:

El código proporcionado es un programa en Haskell que implementa una calculadora científica simple. El usuario puede seleccionar una función matemática (seno, coseno, tangente, exponencial o logaritmo neperiano) y generar una tabla de valores de entrada y salida para esa función hasta un valor máximo especificado.

Tipos de Datos:

* String: Tipo de datos para almacenar cadenas de texto.
* Double: Tipo de datos para almacenar números de coma flotante de doble precisión.

Funciones Principales:

1. functions: Esta función toma una cadena que representa el nombre de una función matemática y un valor de entrada. Luego, devuelve el resultado de aplicar esa función al valor de entrada. Si se proporciona una cadena que no corresponde a ninguna función válida, se produce un error.
2. generateTable: Esta función genera una tabla de valores de entrada y salida para una función matemática especificada. Toma el nombre de la función y el valor máximo para la tabla como entrada, y produce la tabla en la salida estándar.
3. printRow: Esta función imprime una fila de la tabla con un valor de entrada y su correspondiente resultado de la función.
4. main: La función principal del programa. Muestra un mensaje de bienvenida, solicita al usuario que ingrese el nombre de la función y el valor máximo para la tabla, y luego genera y muestra la tabla correspondiente.

Resultados:

A continuación, se presentan los resultados de cada ejercicio, mostrando los valores de entrada y salida para cada función matemática seleccionada por el usuario.

* Ejercicio 1: Función "sin"
  + Valores de entrada: "sin" como función seleccionada por el usuario.
  + Valores de salida: Tabla de valores de entrada (números de 1 hasta el valor máximo ingresado por el usuario) y sus correspondientes resultados de la función seno.
* Ejercicio 2: Función "cos"
  + Valores de entrada: "cos" como función seleccionada por el usuario.
  + Valores de salida: Tabla de valores de entrada (números de 1 hasta el valor máximo ingresado por el usuario) y sus correspondientes resultados de la función coseno.
* Ejercicio 3: Función "tan"
  + Valores de entrada: "tan" como función seleccionada por el usuario.
  + Valores de salida: Tabla de valores de entrada (números de 1 hasta el valor máximo ingresado por el usuario) y sus correspondientes resultados de la función tangente.
* Ejercicio 4: Función "exp"
  + Valores de entrada: "exp" como función seleccionada por el usuario.
  + Valores de salida: Tabla de valores de entrada (números de 1 hasta el valor máximo ingresado por el usuario) y sus correspondientes resultados de la función exponencial.
* Ejercicio 5: Función "log"
  + Valores de entrada: "log" como función seleccionada por el usuario.
  + Valores de salida: Tabla de valores de entrada (números de 1 hasta el valor máximo ingresado por el usuario) y sus correspondientes resultados de la función logaritmo neperiano.

1. Función Booleana

Descripción del Código:

El código proporciona una implementación de una función llamada `filtrar`, que toma una función booleana y una lista y devuelve una lista filtrada según el resultado de aplicar la función booleana a cada elemento de la lista. También se proporciona un ejemplo de uso de esta función utilizando una función booleana `esPar` que filtra números pares de una lista.

Tipos de Datos:

- `a`: Tipo de dato genérico que puede ser cualquier tipo.

- `Int`: Tipo de dato que representa números enteros.

Funciones Principales:

1. `filtrar: (a -> Bool) -> [a] -> [a]`: Esta función toma una función booleana y una lista de tipo `a`, y devuelve una lista filtrada que consiste en aquellos elementos de la lista original para los cuales la función booleana devuelve `True`.

2. `esPar :: Int -> Bool`: Esta función toma un número entero y devuelve `True` si el número es par, y `False` en caso contrario.

3. `main: IO ()`: Función principal del programa. Define una lista original de números y utiliza la función `filtrar` para filtrar los números pares de la lista original.

Resultados:

- Para la lista original `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` y la función booleana `esPar`, la función `filtrar` devuelve `[2, 4, 6, 8, 10]`, que son los números pares de la lista original.

1. Calificaciones

Descripción del Código:

El código proporcionado implementa una función llamada `calificacionALetra`, que toma una lista de enteros que representan calificaciones y devuelve una lista de cadenas que representan las calificaciones convertidas a letras según un criterio establecido.

Tipos de Datos:

- `calificaciones`: Lista de enteros que representan las calificaciones numéricas.

- `calificacionesConvertidas`: Lista de cadenas que representan las calificaciones convertidas a letras.

Funciones Principales:

1. `calificacionALetra`: Función que toma una lista de enteros y devuelve una lista de cadenas que representan las calificaciones convertidas a letras según el criterio establecido.

Resultados:

- Entrada: `calificaciones = [80, 95, 60, 70, 85]`

- Salida: `["Bueno", "Excelente", "Desempeño insuficiente", "Suficiente", "Notable"]`

4) Materias Calificaciones

Descripción del Código:

El código proporcionado define un programa que permite convertir un diccionario de notas de asignaturas en un diccionario de calificaciones.

Tipos de Datos:

- `Asignatura`: Representa el nombre de una asignatura.

- `Nota`: Representa la calificación numérica obtenida en una asignatura.

- `Calificacion`: Representa la calificación en términos de texto para una asignatura.

- `DiccionarioNotas`: Tipo de dato que representa un diccionario donde se almacenan las notas asociadas a cada asignatura.

- `DiccionarioCalificaciones`: Tipo de dato que representa un diccionario donde se almacenan las calificaciones asociadas a cada asignatura.

Funciones Principales:

1. `obtenerCalificacion`: Esta función toma una nota como entrada y devuelve la calificación correspondiente según ciertos rangos predefinidos. Devuelve una cadena de texto representando la calificación.

2. `convertirCalificaciones`: Esta función toma un diccionario de notas como entrada y devuelve un diccionario de calificaciones. Convierte las asignaturas a mayúsculas y filtra solo las calificaciones aprobadas (notas mayores o iguales a 70).

3. `main`: Función principal del programa. Ejecuta un ejemplo de uso del programa, donde se define un diccionario de notas, se llama a la función `convertirCalificaciones` y se imprime el resultado.

Resultados:

- Valores de entrada: `("Matematicas", 85)`

- Salida: `"Notable"`

- Valores de entrada: `("Fisica", 65)`

- Salida: No se incluye en el diccionario de calificaciones debido a que la nota no es suficiente para aprobar.

1. Presupuesto

Descripción del Código:

El código proporcionado define un tipo de dato `Inmueble`, que representa las características de una propiedad inmobiliaria, como el año de construcción, los metros cuadrados, el número de habitaciones, si tiene garaje y la zona donde se encuentra. Luego, se definen dos funciones principales:

1. `calcularPrecio`: Esta función toma un inmueble como entrada y calcula su precio basado en su zona y antigüedad. El precio se calcula según una fórmula específica para cada zona, considerando el año de construcción y otras características del inmueble.

2. `buscarPorPresupuesto`: Esta función toma una lista de inmuebles y un presupuesto como entrada, y devuelve una lista de inmuebles cuyo precio sea igual o inferior al presupuesto dado.

Tipos de Datos:

- `Inmueble`: Representa las características de una propiedad inmobiliaria.

- `año`: Año de construcción del inmueble (tipo `Int`).

- `metros`: Metros cuadrados del inmueble (tipo `Int`).

- `habitaciones`: Número de habitaciones del inmueble (tipo `Int`).

- `garaje`: Indica si el inmueble tiene garaje (tipo `Bool`).

- `zona`: Zona donde se encuentra el inmueble (tipo `Char`).

Funciones Principales:

1. `calcularPrecio`:

- Entrada: Un inmueble (`Inmueble`).

- Salida: El precio del inmueble (tipo `Float`).

2. `buscarPorPresupuesto`:

- Entrada: Una lista de inmuebles (`[Inmueble]`) y un presupuesto (`Float`).

- Salida: Una lista de inmuebles cuyo precio sea igual o inferior al presupuesto dado (`[Inmueble]`).

Resultados:

`calcularPrecio`

- Valores de Entrada: Un inmueble con las siguientes características:

- Año: 2010

- Metros: 120

- Habitaciones: 3

- Garaje: True

- Zona: 'A'

- Salida: Precio del inmueble calculado según la fórmula correspondiente.

`buscarPorPresupuesto`\*\*

- Valores de Entrada:

- Lista de inmuebles: Una lista de varios inmuebles con diferentes características.

- Presupuesto: 100000

- Salida: Lista de inmuebles cuyo precio es igual o inferior al presupuesto dado.