Informe Tecnico Haskell

Jesús Isaac Barajas Castillo

1. Descuento e IVA Haskell

Descripción del Código:

El código implementa un sistema para calcular el precio final de una cesta de la compra, aplicando descuentos o impuestos al precio de los productos.

Tipos de Datos:

1. `Price`: Representa un precio, definido como un valor de tipo `Float`.

2. `Percentage`: Representa un porcentaje, también definido como un valor de tipo `Float`.

3. `Product`: Representa un producto en la cesta, definido como una cadena de caracteres (`String`).

4. `Basket`: Representa la cesta de la compra, que es una lista de tuplas donde cada tupla contiene el nombre del producto y su precio.

Funciones Principales:

1. `applyDiscount`: Toma un precio y un porcentaje de descuento, y devuelve el precio con el descuento aplicado.

2. `applyVAT`: Toma un precio y un porcentaje de impuesto (IVA), y devuelve el precio con el impuesto aplicado.

3. `applyDiscountOrVAT`: Toma una cesta de la compra y una función para aplicar descuento o IVA, junto con el porcentaje correspondiente, y devuelve el precio final de la cesta con el descuento o el impuesto aplicado.

Resultados:

- Precio y producto: ("Leche", 1.5), ("Pan", 2.0), ("Huevos", 3.0)

- Precio final con descuento: 5.85

- Precio final con IVA: 6.82

1. Función llama a otra función

Tipos de Datos:

-`a`: Tipo genérico para el argumento de la función `aplicarFuncion`.

- `b`: Tipo genérico para el resultado de aplicar la función `aplicarFuncion`.

- `duplicar`: Función que toma un valor entero (`Int`) y devuelve otro valor entero (`Int`).

- `listaOriginal`: Lista de enteros (`[Int]`) que se utiliza como entrada en la función `aplicarFuncion`.

- `listaDuplicada`: Lista de enteros (`[Int]`) que representa la salida después de aplicar la función `aplicarFuncion`.

Funciones Principales:

- `aplicarFuncion`: Función que toma una función (`f`) que transforma un tipo `a` en un tipo `b`, y una lista de elementos de tipo `a`, y devuelve una lista de elementos de tipo `b` aplicando la función `f` a cada elemento de la lista.

- `duplicar`: Función que duplica un número entero.

- `main`: Función principal del programa, que imprime la lista original y la lista duplicada.

Resultados:

Entrada:

- `duplicar`: Función que duplica un número entero.

- `listaOriginal`: `[1, 2, 3, 4, 5]` - Lista de enteros.

- Salida: `listaDuplicada`: `[2, 4, 6, 8, 10]` - Lista de enteros duplicados.

1. Diccionario

Tipos de Datos:

- `frase`: String. Representa la frase de entrada.

- `diccionario`: `Map.Map String Int`. Es un diccionario donde las claves son palabras encontradas en la frase y los valores son las longitudes de esas palabras.

Funciones Principales:

- `contarLongitudes`: Esta función toma una cadena de texto (`frase`) como entrada y devuelve un diccionario donde las claves son las palabras en la frase y los valores son las longitudes de esas palabras.

Resultados:

Para la frase de entrada "Hola mundo, este es un ejemplo", la función `contarLongitudes` genera el siguiente diccionario:

fromList [("Hola",4),("mundo,",6),("este",4),("es",2),("un",2),("ejemplo",7)]

Este resultado indica que:

- La palabra "Hola" tiene 4 caracteres.

- La palabra "mundo," tiene 6 caracteres (incluyendo la coma).

- La palabra "este" tiene 4 caracteres.

- La palabra "es" tiene 2 caracteres.

- La palabra "un" tiene 2 caracteres.

- La palabra "ejemplo" tiene 7 caracteres.

4) Calificaciones

Tipos de Datos:

- `calificar`: Función que toma un entero (nota) y devuelve una cadena de caracteres (calificación).

- `procesarNotas`: Función que toma una lista de tuplas de cadenas de caracteres y enteros (nombre de la asignatura y nota) y devuelve una lista de tuplas de cadenas de caracteres (nombre de la asignatura en mayúsculas) y cadenas de caracteres (calificación).

- `notasAlumno`: Lista de tuplas de cadenas de caracteres y enteros (nombre de la asignatura y nota).

Funciones Principales:

1. `calificar`: Toma una nota y devuelve la calificación correspondiente según ciertas condiciones.

2. `procesarNotas`: Procesa la lista de notas del alumno, convirtiendo cada asignatura a mayúsculas y asignando una calificación a cada nota.

3. `main`: Función principal que ejecuta un ejemplo de uso del programa, imprimiendo las calificaciones de las asignaturas del alumno.

Resultados: -Entrada: notasAlumno = [("Matematicas", 90), ("Ciencias", 78), ("Historia", 60)]

-Salida: [("MATEMATICAS","Notable"),("CIENCIAS","Bueno"),("HISTORIA","Desempeño insuficiente")]

1. Vectores

Tipos de Datos: `Vector`: Este tipo de dato representa un vector en el plano cartesiano. Está definido como una tupla de dos valores de tipo `Double`, donde el primer valor representa la coordenada en el eje x y el segundo valor representa la coordenada en el eje y.

Funciones Principales: `moduloVector: Vector -> Double`: Esta función toma un vector como entrada y devuelve un valor de tipo `Double`, que representa el módulo del vector.

- `main: IO ()`: Esta es la función principal del programa. Es de tipo `IO ()`, lo que significa que realiza operaciones de entrada/salida. En esta función, se define un ejemplo de uso donde se crea un vector con coordenadas, se calcula su módulo utilizando la función `moduloVector`, y se imprime el resultado en la consola utilizando `printf`.

Resultados:

- Entrada: Vector: (3.0, 4.0)

- Salida: Módulo del vector (3.0, 4.0) es: 5.00

1. Valores Atípicos

Tipos de Datos:

- `outliers`: Esta función toma una lista de números de tipo `Double` como entrada y devuelve otra lista de números `Double`, que son los valores atípicos (outliers) de la muestra de entrada.

- `main`: Esta función es de tipo `IO ()`, lo que significa que es una acción de entrada/salida que no devuelve ningún valor útil. Se utiliza para ejecutar el código principal del programa.

Funciones Principales:

- `outliers`: Calcula los valores atípicos en una muestra de números utilizando la media (`mean`) y la desviación estándar (`stdDev`) de la muestra. Utiliza la función `zScore` para calcular el puntaje Z de cada elemento en la lista.

- `main`: Define una lista de muestra `sample`, calcula los valores atípicos utilizando la función `outliers` y luego imprime estos valores atípicos en la consola.

Resultados:

- Valores de Entrada (Sample):

sample = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1000]

- Valores de Salida (Atípicos): atipicos = [1000.0]