PROGRAMACION FUNCIONAL

RODRIGUEZ MORENO MARCO ANTONIO

Aarón Hernández García

Actividad: 2.1

18/01/2024

1.3 Tipos de Datos en Programación Funcional

En programación funcional, los tipos de datos son fundamentales para definir la naturaleza y comportamiento de los datos que se manipulan. Estos tipos ayudan a garantizar la seguridad y claridad en el código. A continuación, se describen los principales tipos de datos utilizados:

1. **Tipos de Datos Primitivos:**
    - **Enteros:** Números sin parte decimal.
    - **Flotantes:** Números con parte decimal.
    - **Caracteres:** Símbolos individuales como letras o dígitos.
    - **Booleanos:** Valores de verdad (True o False).
2. **Listas:**
    - Estructuras que almacenan secuencias ordenadas de elementos del mismo tipo.
    - Ejemplo en Haskell:

```
lista = [1, 2, 3, 4]
```

3. **Tuplas:**
    - Estructuras que agrupan un número fijo de elementos, que pueden ser de diferentes tipos.
    - Ejemplo en Haskell:

```
tupla = ("Hola", 42, True)
```

4. **Tipos de Datos Algebraicos:**
    - Permiten la creación de tipos complejos mediante la combinación de otros tipos.
    - Ejemplo en Haskell:

```
data Persona = Persona String Int  -- Nombre y edad
```

5. **Funciones como Tipos de Datos:**
    - Las funciones son tratadas como ciudadanos de primera clase, lo que significa que pueden ser asignadas a variables, pasadas como argumentos y devueltas como resultados.
    - Ejemplo en Haskell:

```haskell
suma :: Int -> Int -> Int
suma x y = x + y
```

1.4 Funciones en la Programación Funcional

Las funciones son el núcleo de la programación funcional. Se utilizan para abstraer y encapsular comportamientos, facilitando la modularidad y la reutilización del código. A continuación, se detallan las características y tipos de funciones más relevantes:

1. **Funciones Puras:**
    - Siempre producen el mismo resultado para los mismos argumentos y no tienen efectos secundarios.
    - Ejemplo en Python:

```python
def suma(a, b):
    return a + b
```

2. **Funciones de Orden Superior:**
    - Funciones que pueden recibir otras funciones como argumentos o devolverlas como resultados.
    - Ejemplo en Python:

```python
def aplicar_funcion(func, x):
    return func(x)

print(aplicar_funcion(lambda x: x ** 2, 5))  # Devuelve 25
```

3. **Funciones Anónimas (Lambdas):**
    - Funciones definidas sin un nombre explícito, generalmente utilizadas para operaciones simples y de corta duración.
    - Ejemplo en Python:

```python
cuadrado = lambda x: x ** 2
print(cuadrado(4))  # Devuelve 16
```

4. **Composición de Funciones:**
    - Proceso de combinar dos o más funciones para producir una nueva función.
    - Ejemplo en Haskell:

```haskell
(.) :: (b -> c) -> (a -> b) -> a -> c
f . g = \x -> f (g x)
```

5. **Recursión:**
    - Técnica donde una función se llama a sí misma para resolver problemas que pueden ser descompuestos en subproblemas similares.
    - Ejemplo en Haskell:

```haskell
factorial 0 = 1
factorial n = n * factorial (n - 1)
```