**TECNOLÓGICO NACIONAL DE MÉXICO**

**CAMPUS CIUDAD HIDALGO**

Un dibujo de una rueda

Descripción generada automáticamente con confianza baja

**INVESTIGACIÓN**

**MATERIA: PROGRAMACIÓN LÓGICA Y FUNCIONAL**

**UNIDAD 1: “CONCEPTOS FUNDAMENTALES”**

**ALUMNOS:**

**AXEL RUBEN PALACIOS GARCIA**

**JUAN MANUEL BAUTISTA GARCIA**

**078CA**

**DOCENTE: ERIK ESCOBAR MENDOZA**

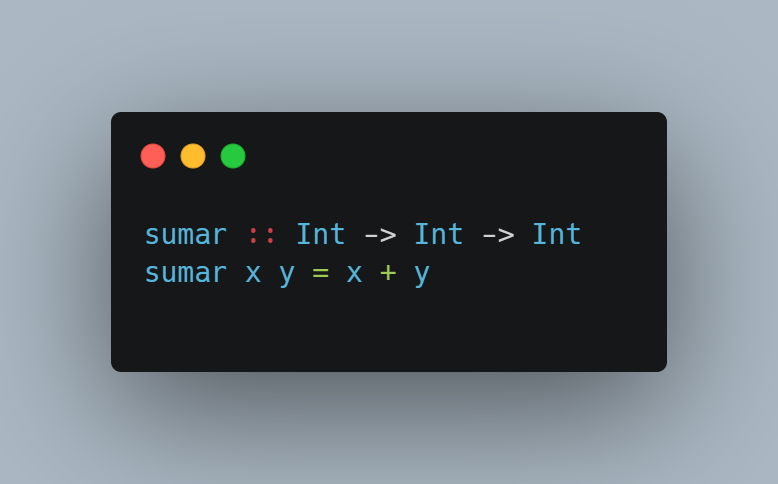
**10 DE FEBRERO DE 2023**

**CIUDAD HIDALGO, MICHOACÁN**

**Sintaxis de una función**

En Haskell, las funciones son la base de la programación funcional. Son similares a las funciones en otros lenguajes, pero con algunas diferencias clave.

En Haskell, una función se define mediante su nombre, sus argumentos y su cuerpo. Por ejemplo, aquí está una función que toma dos números y devuelve su suma:

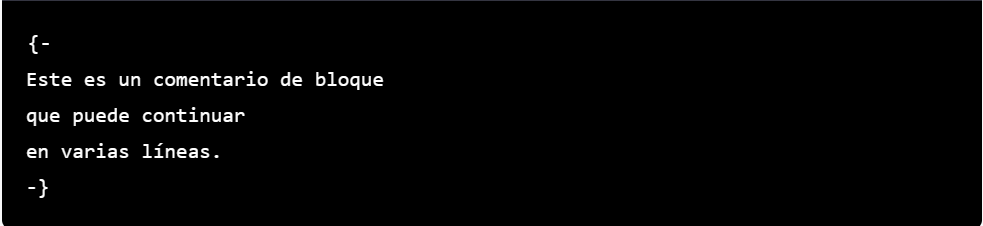


**Comentarios en Haskell**

En Haskell, existen dos tipos de comentarios: los comentarios de línea única y los comentarios de bloque.

Un comentario de línea única comienza con el carácter -- y continúa hasta el final de la línea. Por ejemplo:





**Palabras Reservadas**

* class: Define una clase, que es una estructura de datos que define un conjunto de métodos y atributos que pueden ser compartidos por diferentes tipos de datos.
* data: Define un nuevo tipo de datos en Haskell, incluyendo estructuras de datos como listas, tuplas y registros.
* default: Especifica un valor predeterminado que se utiliza cuando no se proporciona un valor para un argumento en una función.
* deriving: Permite especificar cómo se deben derivar automáticamente diferentes propiedades de un tipo de datos, como la igualdad y la representación en cadena.
* do: Marca el inicio de un bloque de acciones que se ejecutan secuencialmente en Haskell.
* else: Especifica una acción alternativa que se debe realizar si una condición dada no se cumple.
* foreign: Permite la integración de código escrito en otros lenguajes con Haskell.
* if: Permite hacer una comparación simple y ejecutar diferentes acciones dependiendo del resultado.
* import: Permite importar módulos o paquetes de Haskell en un programa para poder utilizar sus funciones y tipos de datos.
* in: Marca el final de una definición de función o una declaración de variable en Haskell.
* infix: Permite definir operadores infijos que se pueden utilizar entre dos argumentos.
* infixl: Especifica la precedencia y la asociatividad a la izquierda de un operador infijo.
* infixr: Especifica la precedencia y la asociatividad a la derecha de un operador infijo.
* instance: Permite definir una instancia de una clase en Haskell.
* let: Permite declarar una variable local en Haskell y asignarle un valor inicial.
* module: Permite definir un módulo en Haskell, que es una estructura de datos que agrupa funciones y tipos de datos relacionados.
* newtype: Permite definir un nuevo tipo de datos en Haskell que se comporta de manera similar a un tipo existente.

**Operadores**

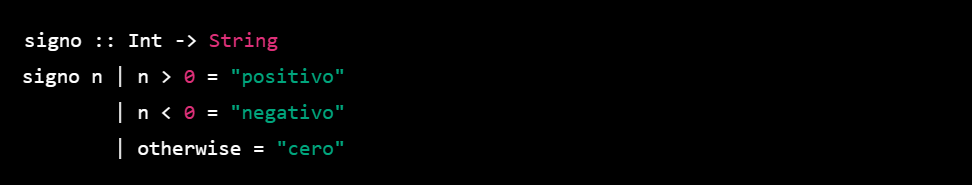
Haskell tiene una variedad de operadores que se pueden utilizar para manipular valores y realizar cálculos. Algunos de los operadores más comunes en Haskell incluyen:

* Aritméticos: +, -, \*, /, ^ (potencia)
* Lógicos: &&, ||, not
* Relacionales: ==, /=, <, <=, >, >=
* de Asignación: = ==, /=, <, <=, >, >=
* de Lista: ++ (concatenación)

**Guardas**

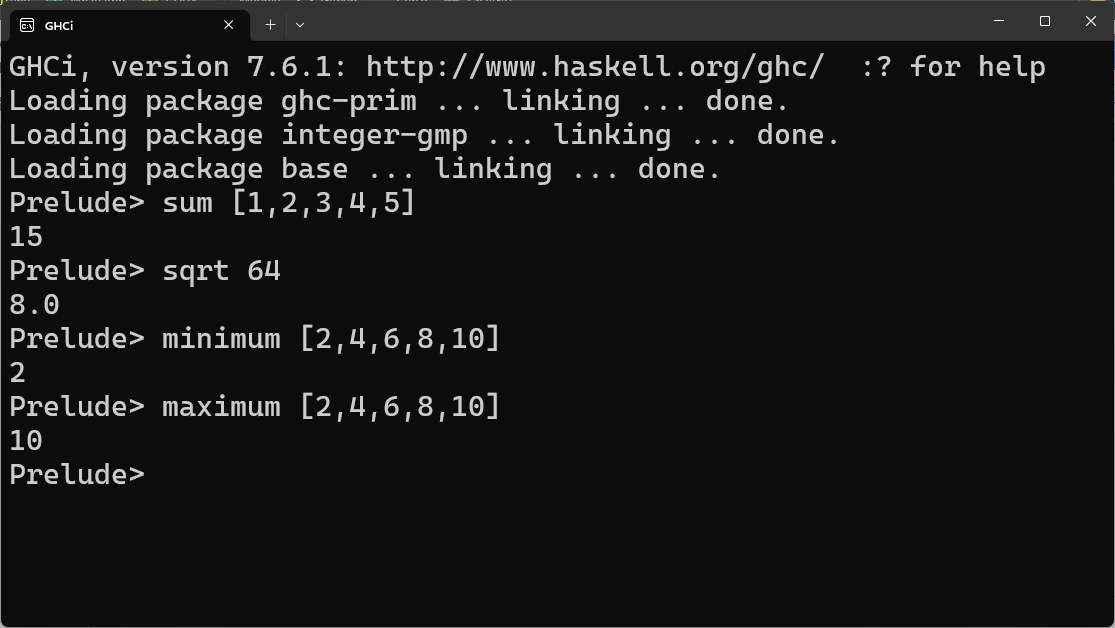
Las guardas en Haskell son una forma de controlar el flujo de una función mediante la evaluación de diferentes condiciones. Se escriben después del nombre de la función y antes de su definición, y se utilizan para controlar cuál de las definiciones de la función se utiliza en función de los valores de los argumentos.

Las guardas se definen como una lista de condiciones separadas por barras verticales (|). La primera condición que se evalúa como verdadera determina cuál de las definiciones de la función se utiliza.



**Funciones:**

Utilizamos **sum** que devuelve la suma de los números, **sqrt** que devuelve la raíz cuadrada de un número, **minimum** que devuelve el número más pequeño de una lista de números y por último **maximum** que devuelve el número más grande de una lista de números.case: Permite hacer una comparación exhaustiva de un valor y ejecutar diferentes acciones dependiendo del resultado.



**Bibliografía:**

*Resumen haskell*. (s. f.). www.cs.us.es. Recuperado 10 de febrero de 2023, de <https://www.cs.us.es/~jalonso/cursos/i1m/doc/resumen_Haskell.pdf>

(n.d.). Uso de funciones en Haskell. Retrieved February 10, 2023, from <https://antares.sip.ucm.es/~fernando/pf/temas/usoFunciones.html>

*Tutorial de Haskell | ¡Aprende como programar en Haskell!* (2020, October 14). IONOS. Retrieved February 10, 2023, from <https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/tutorial-de-haskell/>