Genie

Genie es un lenguaje de programación de alto nivel, imperativo, estructurado y orientado a objetos. Fue diseñado como un dialecto alternativo más simple y limpio para el compilador Vala, preservando al mismo tiempo la misma funcionalidad del lenguaje Vala, con una sintaxis que se asemeja a numerosos lenguajes modernos como **Python**, **Boo**, **D** y **Delphi**. Apareció en 2008 diseñado por Jamie McCracken y desarrollado y promovido desde el proyecto GNOME.

Tipos de Datos

Como en muchos de los lenguajes de los que deriva y se asemeja, Genie ofrece una buena variedad de tipos de datos que permiten al usuario implementar estructuras derivadas.

1. Textuales

El texto se almacena como una cadena de caracteres o un `string` y se delimita entre comillas dobles.

```
nombre:string = "Manolo"

menu:string = """
Primer plato
Segundo plato
Postre"""

caracter:char = 't'
```

Para referirnos a un solo carácter usamos comillas simples, y puede ser:

```
:char, :uchar, un solo carácter
:unichar, carácter Unicode
```

2. Números

Podemos especificar el tipo de número que utilizamos:

```
:int para números enteros
:float para números decimales
:double para decimales grandes
```

También podemos especificar los bits del entero (y sus versiones sin signo con `uint`):

3. Booleanos

Genie cuenta con el tipo de datos `bool` en su diseño. Este tipo es el encargado de almacenar las constantes `true` y `false`.

```
a:bool = true
```

4. Enumeraciones

Con enum creamos un tipo de enumeración. Se trata de una secuencia ordinal (enumerable) de valores en la que cada entrada equivale a un valor numérico (0, 1, 2, etc.). En principio empiezan desde cero, aunque también es posible asignar un valor entero específico a un indicador enum.

```
enum weekDays
    DIA_LUNES
    DIA_MARTES
    DIA_MIERCOLES
    DIA_JUEVES
    DIA_VIERNES
init
    print("%d",MiEnum.Jueves)
```

5. null

Por defecto, Genie se asegura de que todos los puntos de referencia (tanto variables como funciones) apunten a objetos reales. Esto significa que no se puede asignar arbitrariamente `null` a una variable.

Para permitir que una referencia sea null se debe seguir el tipo de dato con un interrogante, por ejemplo: `string?`, con lo que estamos afirmando que esa variable puede ser nula y así evitamos errores de código.

Podemos utilizar este modificador tanto en tipos de parámetros como en tipos de retorno de funciones, y aunque puede resultar útil en fase de depuración para la comprobación del código, se aconseja su deshabilitación posterior.

```
def permite_nulls (param : string?) : string?
    return param
```

6. Especiales

Genie cuenta con dos tipos especiales que si bien no son de un uso practico en el dia a dia, son fundamentales en la estructura del lenguaje:

- ***object***: Como buen lenguane orientado a objetos, Genie cuenta con un tipo preinstalado, `object`, que es el tipo base de todos los objetos en Genie.
- ***void:*** void es el tipo de retorno especificado cuando la función no devuelve un valor.

Creación de Nuevos Tipos

Genie permite la creación de nuevos tipos mediante el uso de dos mecanismos: las ***clases*** y las ***interfaces***.

· Clases

En Genie, las clases encapsulan atributos y procedimientos para definir nuevos tipos.

Las propiedades describen características variables de una clase. Las definimos con `prop`:

```
prop color:string = "verde"
prop altura: int = 44
```

Si las propiedades definen características de las clases, los métodos definen sus funcionalidades.

```
class Perro:Object
    prop nombre: string
    prop raza: string
    prop edad: int

    construct(nombre: string, raza: string, edad: int)
        self.nombre = nombre
        self.raza = raza
        self.edad = edad

    def ladrar()
        print("%s dice: Guau guau\n", self.nombre)
```

· Interfaces

Asi como en Java, otro lenguaje orientado a objetos, Genie también soporta el uso de interfaces, que permiten definir tipos polimórficos. Las interfaces declaran métodos sin implementar, dejando la implementación a las clases que las implementan.

```
interface Animal
    def hacer_sonido(): void

class Perro:Object, Animal
    prop raza: string

    construct(raza: string)
        self.raza = raza

    def hacer_sonido()
        print("Guau guau")

class Gato:Object, Animal
    prop color: string

    construct(color: string)
        self.color = color

    def hacer_sonido()
        print("Miau")
```

Herencia

Las clases en Genie pueden heredar de otras clases, permitiendo la reutilización de código y la creación de jerarquías de tipos. Esto soporta el polimorfismo y la creación de tipos más complejos.

```
class Animal:Object
    prop nombre: string

    construct(nombre: string)
        self.nombre = nombre

    def hacer_sonido()
        print("Sonido genérico")

class Perro:Animal
    prop raza: string

    construct(nombre: string, raza: string)
        super(nombre)
        self.raza = raza

    override def hacer_sonido()
        print("Guau guau")
```

En este ejemplo, `Perro` hereda de `Animal` y sobrescribe el método `hacer_sonido`.

Tipos Polimórficos

El polimorfismo en Genie se logra mediante la implementación de interfaces y la herencia de clases. Esto permite que diferentes clases sean tratadas de manera uniforme basándose en una interfaz común o una superclase.

Sistema de Tipos del Lenguaje

Genie utiliza **equivalencia nominal** para sus tipos, lo que significa que dos tipos son considerados equivalentes si tienen el mismo nombre y están definidos de la misma manera.

Compatibilidad

Los tipos en Genie deben ser compatibles de acuerdo a sus definiciones. Por ejemplo, una variable de tipo int no puede ser asignada a una variable de tipo string sin conversión explícita.

Genie soporta tipos primitivos, compuestos (como arrays y hashes), y tipos definidos por el usuario (como clases e interfaces).

Conversión de Tipos

En Genie, la conversión de tipos es una operación esencial que te permite transformar datos de un tipo a otro. Este lenguaje soporta tanto conversiones explícitas como implícitas entre varios tipos de datos.

1. Conversión Implícita

Genie maneja automáticamente algunas conversiones implícitas entre tipos compatibles, especialmente en operaciones aritméticas y al asignar valores a variables de tipos más generales.

2. Conversión Explícita

Para conversiones que no son manejadas automáticamente, debes realizar conversiones explícitas. En Genie, la conversión explícita de tipos se hace utilizando la sintaxis de casting, similar a otros lenguajes como C, o haciendo uso de alguno de los varios métodos incorporados en el lenguaje para convertir entre tipos, como `to_string`, `to_int`, `to_float`, etc.

De `float` a `int`:

```
var entero: int
var flotante: float = 3.14
entero = flotante.to_int()
```

De `bool` a `string`:

```
init
    var booleano: bool = true
    var cadena: string
    cadena = booleano.to_string()  // Convertir bool a string
    print("Cadena: %s\n", cadena)
```

Inferencia de Tipos

Genie admite la inferencia de tipos, lo que permite al compilador deducir automáticamente el tipo de una variable a partir del contexto de su asignación. Para esot se utiliza la palabra reservada `var` al momento de declarar la variable que sera inferida.

```
init
    var texto = "Hola, Mundo"  // `texto` es inferido como `string`
    print("Texto: %s\n", texto)
```