MANUAL TÉCNICO

T-SWIFT

albertt wosveli itzep raymundo

8/09/2023

Universidad San Carlos de Guatemala
Facultad de ingeniería
Escuela de ciencias y sistemas

ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 2

INTRODUCCIÓN

Una de las líneas más complejas de la rama de las ciencias computacionales es el análisis de lenguajes, compiladores e intérpretes, su dificultad resulta en el proceso de abstracción que poseen estas herramientas, en este curso algunas de estas etapas de creación de compiladores e intérpretes las usamos para crear este programa en donde creamos un interprete de un nuevo lenguaje de programación.

Requisitos del Sistema

- Terminal
- Sistema Operativo Libre
- Repositorio clonado de Github
- Navegador A elección
- Herramientas para pruebas de APIS
- Compilador de golang instalado

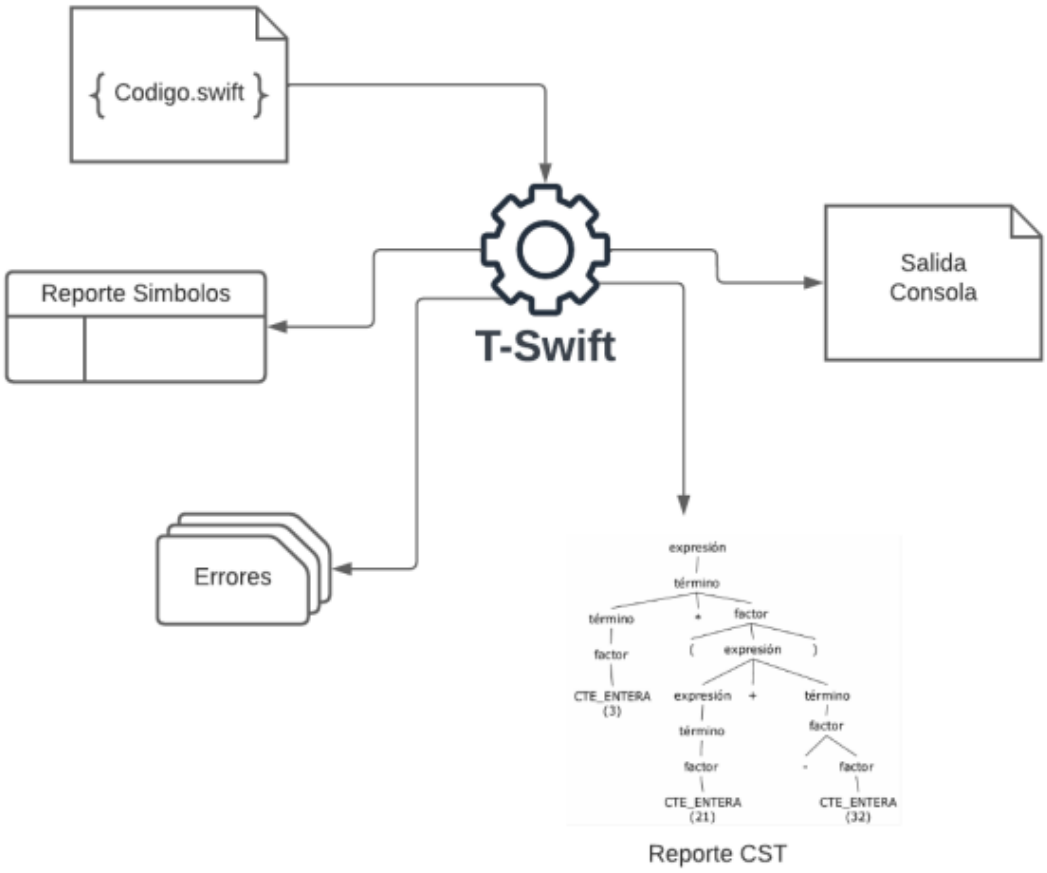
T-Swift

T-Swift IDE es un entorno de desarrollo que provee las herramientas para la escritura de programas en lenguaje T-Swift. Este IDE nos da la posibilidad de visualizar tanto la salida en consola de la ejecución del archivo fuente como los diversos reportes de la aplicación que se explican más adelante. La Interfaz gráfica podrá ser desarrollada con la arquitectura y el framework a elección del estudiante, siempre y cuando se utilice el lenguaje y las herramientas indicadas por los tutores para el procesamiento y reconocimiento del lenguaje T-Swift.

Arquitectura General del proyecto

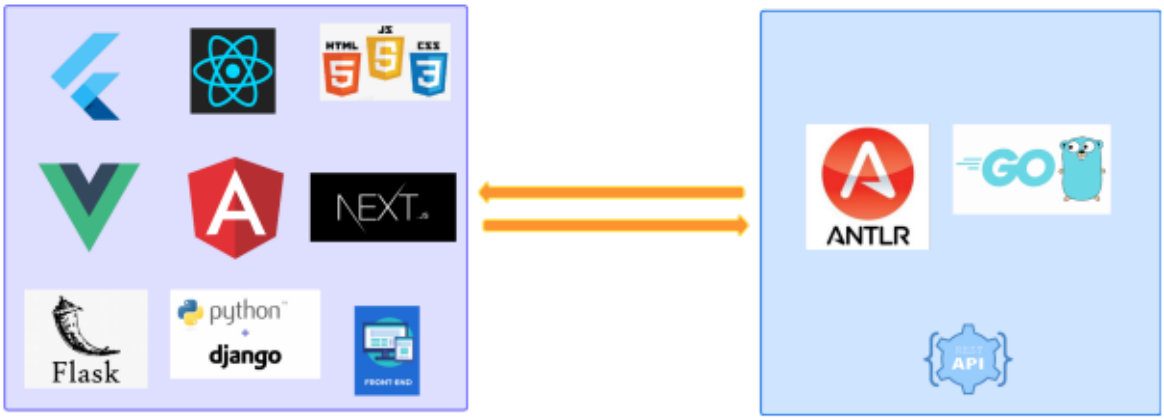
Para una mayor facilidad de empleo de esta herramienta, será realizado por medio de un sistema web en utilizando algunas herramientas sencillas para que sea fácil de utilizar.

Estructura cliente servidor



Flujo de ejecución del proyecto

Desde el lado del servidor tendremos un sistema en donde realizaremos el análisis de la entrada enviada, y retornando un json estas pueden ser algunas de las herramientas que podemos utilizar para el sistema.



Para este sistema utilizamos únicamente el sistema de despliegue Vite y node puro ya que no necesitamos demasiados recursos, la ligereza obtenida con un sistema de despliegue nos brinda un poco de minimalismo a la hora de manejar los elementos del despliegue.

Generalidades del lenguaje

El lenguaje T-Swift está inspirado en la sIntaxis del lenguaje Swift, por lo tanto se conforma por un subconjunto de instrucciones de este, pero con la diferencia de que T-Swift tendrá una sIntaxis más reducida pero sin perder las funcionalidades que caracterizan al lenguaje original.

Cuando se haga referencia a una ‘expresión’ , se hará referencia a cualquier sentencia u operación que devuelve un valor. Por ejemplo:

- Una operación aritmética, comparativa o lógica
- Acceso a un variable
- Acceso a un elemento de una estructura
- Una llamada a una función

T-Swift por su naturaleza carece de una función main, por ello para el inicio de ejecución del programa, el Intérprete deberá de ejecutar las órdenes conforme estas sean declaradas en el archivo de entrada.

Identificadores

Un identificador será utilizado para dar un nombre a variables y métodos. Un identificador está compuesto básicamente por una combinación de letras, dígitos, o guión bajo. Ejemplo: IdValido, id_Valido, i1d_valido5.

Case sensitive

El lenguaje T-Swift es case sensitive, esto quiere decir que diferenciará entre mayúsculas con minúsculas, por ejemplo, el identificador variable hace referencia a una variable específica y el identificador VariabLe hace referencia a otra variable. Las palabras reservadas también son case sensitive por ejemplo la palabra if no será la misma que IF.

Comentarios

Un comentario es un componente léxico del lenguaje que no es tomado en cuenta para el análisis sintáctico de la entrada. Existirán dos tipos de comentarios:

- Los comentarios de una línea que serán delimitados al inicio con el símbolo de // y al final como un carácter de finalización de línea.
- Los comentarios con múltiples líneas que empezarán con los símbolos /* y terminarán con los símbolos *

Tipos de datos primitivos

Se utilizan los siguientes tipos de datos primitivos:

Tipo primitivo	Definición	Rango (teorico)	Valor por defecto
`Int`	Acepta valores números enteros	-2,147,483,648 a +2,147,483,647	`nil`
`Float`	Acepta valores numéricos de punto flotante	1.2E-38 a 3.4E+38 (6 dígitos de precisión)	`nil`
`String`	Acepta cadenas de caracteres	[0, 65535] caracteres (acotado por conveniencia)	`nil`
`Bool`	Acepta valores lógicos de verdadero y falso	`true false`	`nil`
`Character`	Acepta un solo caracter ASCII	[0, 65535] caracteres	`nil`

Tipos de datos Compuestos

Cuando hablamos de tipos compuestos nos vamos a referir a ellos como no primitivos, en estos tipos vamos a encontrar las estructuras básicas del lenguaje T-Swift

- Vector
- Matriz
- Struct

Valor Nulo

Será un conjunto de sentencias delimitado por llaves "{ }", cuando se haga referencia a esto querrá decir que se está definiendo un ámbito local con todo y sus posibles instrucciones y que tiene acceso a las variables del ámbito global, además las variables declaradas en dicho ámbito únicamente podrán ser utilizadas en este ámbito o en posibles ámbitos anidados

Bloques de secuencia

Será un conjunto de sentencias delimitado por llaves "{ }", cuando se haga referencia a esto querrá decir que se está definiendo un ámbito local con todo y sus posibles instrucciones y que tiene acceso a las variables del ámbito global, además las variables declaradas en dicho ámbito únicamente podrán ser utilizadas en este ámbito o en posibles ámbitos anidados

```
// <sentencias de control>
{
// sentencias
}

var i = 10 // variable global es accesible desde este ámbito
if i == 10
{
    var j:Int = 10 + i // i es accesible desde este ámbito
    if  i == 10
    {
        var k: Int = j + 1    // i y j son accesibles desde este ámbito
    }
    j = k // error k ya no es accesible en este ámbito
}
```

Signos de agrupación

Para dar orden y jerarquía a ciertas operaciones aritméticas se utilizarán los signos de agrupación. Para los signos de agrupación se utilizarán los paréntesis ()

Ejemplo: 3 - (1 + 3) * 32 / 90

Variables

Una variable es un elemento de datos cuyo valor puede cambiar durante el curso de la ejecución de un programa siempre y cuando sea el mismo tipo de dato. Una variable cuenta con un nombre y un valor, los nombres de variables no pueden coincidir con una palabra reservada. Para poder utilizar una variable se tiene que definir previamente, la declaración nos permite crear una variable y asignarle un valor o sin valor. Además la definición de tipo durante la declaración puede ser implícita o explícita, es explícita cuando se indica el tipo del cual será la variable e implícita cuando esta toma el tipo del valor al cual se está asignando

```
// declaración con tipo y valor
var <identificador> : <Tipo> = <Expresión>
// declaración con valor
var <identificador> = <Expresión>
// declaración con tipo y sin valor
var <identificador> : <Tipo> ?
```

Constantes

```
// declaración de constantes

//Incorrecto, la constante debe tener un valor asignado
let valor: String?

// correcto, declaración de una constante tipo Int con valor
Let valor1 = 10

let valor1:Int = 10.01 // Error: no se puede asignar un Float a un Int

let valor2:Float = 10.2 // correcto

let valor3 = "esto es una variable";  //correcto constante tipo String

let valor4:Bool = true   //correcto

let .58 = 4; // debe ser error porque .58 no es un nombre válido

let if = "10" // debe ser un error porque "if" es una palabra reservada

// error valor1 al ser una constante no puede cambiar su valor
valor1 = 200
```

Operadores Aritméticos

Los operadores aritméticos toman valores numéricos de expresiones y retornan un valor numérico único de un determinado tipo. Los operadores aritméticos estándar son adición o suma +, sustracción o resta -, multiplicación *, y división /, adicionalmente vamos a trabajar el módulo %.

Operandos	Tipo resultante	Ejemplo
`Int + Int` `Int + Float`	`Int` `Float`	`1 + 1 = 2` `1 + 1.0 = 2.0`
`Float + Float` `Float + Int`	`Float` `Float`	`1.0 + 13.0 = 14.0` `1.0 + 1 = 2.0`
`String + String`	`String`	`"ho" + "la" = "hola"`

Operandos	Tipo resultante	Ejemplo
`Int - Int` `Int - Float`	`Int` `Float`	`1 - 1 = 0` `1 - 1.0 = 0.0`
`Float - Float` `Float - Int`	`Float` `Float`	`1.0 - 13.0 = -12.0` `1.0 - 1 = 0.0`

Operandos	Tipo resultante	Ejemplo
`Int * Int` `Int * Float`	`Int` `Float`	`1 * 10 = 10` `1 * 1.0 = 1.0`
`Float * Float` `Float * Int`	`Float` `Float`	`1.0 * 13.0 = 13.0` `1.0 * 1 = 1.0`

Operandos	Tipo resultante	Ejemplo
`Int / Int` `Int / Float`	`Int` `Float`	`10 / 3 = 3` `1 / 3.0 = 0.3333`
`Float / Float` `Float / Int`	`Float` `Float`	`13.0 / 13.0 = 1.0` `1.0 / 1 = 1.0`

Operandos	Tipo resultante	Ejemplo
`Int % Int`	`Int`	`10 % 3 = 1`

Operadores Asignación

```
var var1:Int = 10
var var2:Float = 0.0

var1 += 10 //var1 tendrá el valor de 20

var1 += 10.0 // error, no puede asignar un valor de tipo Float a un Int

var2 += 10 // var2 tendrá el valor de 10.0

var2 += 10.0 //var tendrá el valor de 20.0

var str:String = "cad"

str += "cad" //str tendrá el valor de "cadcad"

str += 10 //operación inválida String + Int
```

Vista general del lenguaje

```
//struct con atributo sin valor por defecto
// y con un atributo con valor por defecto
struct Persona{
    var Nombre: String
    var edad = 0
}
// struct con funciones
struct Avion {

  var pasajeros = 0
  var velocidad = 100
```

```
  var nombre: String
  var piloto: Persona
  // metodo dentro de struct
  mutating func frenar(){
    print("Frenando")
    //al ser mutable sí afecta al struct
    self.velocidad = 0
  }
  // funcion inmutable
  func mostrarVelocidad(){
    print("Velocidad",self.velocidad)
  }

}

//  creación de una instancia
var avioneta = Avion( nombre: "78496", piloto: Persona(Nombre: "Joel",
edad: 43 )  )
// acceso a un atributo
print(avioneta.pasajeros)
// modificion de un atributo
avioneta.pasajeros = 5

print("Velocidad", avioneta.pasajeros)
// llamada de la funcion
avioneta.mostrarVelocidad()
// copia de structs por valor
var avioneta2 = avioneta
avioneta2.pasajeros = 0
//imprime: avioneta.pasajeros: 5
print("avioneta.pasajeros:",avioneta.pasajeros)
//avioneta2.pasajeros 0
print("avioneta2.pasajeros:",avioneta2.pasajeros)

print("avioneta.piloto.Nombre:",avioneta2.piloto.Nombre )

struct Fruta{
    let nombre: String = "pera"
    var precio: Int
    }

// solo se puede definir precio en el constructor
// si se llega a definir nombre será un error
var pera = Fruta(precio: 10)
```