Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas

Moviecats

—

Nombre: albertt wosveli itzep raymundo

carnet: 201908658

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 intérprete de un nuevo lenguaje de programación.

.

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.

Descripción de Interface

Funcionalidades dentro de la interfaz:

- Carga Masiva: el sistema es capaz de cargar información de tipo Json
- Ver Estructuras : el sistema le mostrará el estatus general de las estructuras.

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.

Estructuras de control:

If:

Ejecuta un bloque de sentencias si una condición especificada es evaluada como verdadera. Si la condición es evaluada como falsa, otro bloque de sentencias puede ser ejecutado.

```
if 3 < 4 {
// Sentencias
} else if 2 < 5 {
// Sentencias
} else {
// Sentencias
}
if true { // Sentencias }
if false {// Sentencias } else { // Sentencias }
if false {// Sentencias } else if true { // Sentencias }
```

Switch:

Evalúa una expresión, comparando el valor de esa expresión con un case, y ejecuta declaraciones asociadas a ese case, así como las declaraciones en los case que siguen. Si ocurre una coincidencia, el programa ejecuta las declaraciones asociadas correspondientes. Si la expresión coincide con múltiples entradas, la primera será la seleccionada. Si no se encuentra una cláusula de case coincidente, el programa busca la cláusula default opcional, y si se encuentra, transfiere el control a ese bloque, ejecutando las declaraciones asociadas. Si no se encuentra un default el programa continúa la ejecución en la instrucción siguiente al final del switch. Por convención, el default es la última cláusula. La declaración break es opcional, puede o no estar entre las sentencias de cada bloque.

```
let numero = 2
switch numero {
  case 1:
    print("Uno")
  case 2:
    print("Dos")
  case 3:
    print("Tres")
  default:
    print("Invalid day")
}
/* Salida esperada:
Dos
*/
```

While:

Crea un bucle que ejecuta un bloque de sentencias especificadas mientras cierta condición se evalúe como verdadera (true). Dicha condición es evaluada antes de ejecutar el bloque de sentencias y al final de cada iteración

```
while num != 0 {
    num -= 1
    print(num)
}
/* Salida esperada:
9
8
7
6
5
4
3
2
1
0
*/
```

For:

Un bucle for en el lenguaje T-Swift se comportará como un for moderno, que recorrerá alguna estructura compuesta, en este caso sobre: cadenas, arreglos unidimensionales y rangos. La variable que recorre los valores se comportará como una constante, por lo tanto no se podrán modificar su valor en el bloque de sentencias, su valor únicamente cambiará con respecto a las iteraciones.

```
let letras = ["O", "L", "C", "2"]
// for que recorre un arreglo unidimensional
for letra in letras {
    print(letra)
    letra = "cadena" //error no es posible asignar algo a letra
}
/*Salida esperada:
1
2
3
4
5
h
o
l
a
O
L
C
2
*/
```

.

Guard:

Esta sentencia perteneciente al lenguaje T-Swift su finalidad es transferir el control del programa fuera del ámbito actual cuando no se cumple cierta condición, a diferencia de la sentencia if guard evalúa la expresión y si es falsa ejecuta el bloque else. Guard está pensado para ser utilizado exclusivamente para este fin, por lo tanto requiere que el bloque de sentencias termine con una sentencia de tipo break, return o continue, en caso contrario será considerado un error.

```
var i = 2

while (i <= 10) {
  // la sentencia guard verifica si i es impar
  guard i % 2 == 0 else {
     i = i + 1
    continue
  }
  print(i)
  i = i + 1
}
/* Salida
2
4
6
8
10
*/
```

Sentencias de retorno:

podemos utilizar dentro del lenguaje las sentencias break, continue, return

También podemos usar estructuras como vectores:

```
//vector con valores
var vec1: [Int] = [10,20,30,40,50]
//vector vacío
var vec2: [Float] = []
//vector vacío
var vec3: [String] = [ ];

//imprime 0
print(vec2.count)

//inserta 100 al final
vec1.append(100) //[10,20,30,40,50,100]

//inserciones en vacíos
vec2.append(1.0) // [1.0]
vec3.append("cadena") // ["cadena"]
```

Matriz

```
var matrix : [[[Int]]] = [[[Int]]] (repeating: [[Int]] (repeating: [Int]
(repeating: 0, count:2), count:3), count:4)

/*
esta matriz sería:
[
  [
   [0, 0], [0, 0], [0, 0]
  ],
  [
   [0, 0], [0, 0], [0, 0]
  ],
  [
   [0, 0], [0, 0], [0, 0]
  ],
  [
   [0, 0], [0, 0], [0, 0]
  ]
]
```

Struct

```
struct Persona{
    var Nombre: String
    var edad = 0
}
// struct con funciones
struct Avion {

  var pasajeros = 0
  var velocidad = 100
```

Funciones:

```
func func() {
  return;
}

// función inválida:
// ya se ha declarado una función llamada  func previamente

func func () -> String {
   return "valor";
}

// función inválida
// nombre inválido

func if() -> Float {
   return 21.0;
}

func valor() -> String {
   // error: valor de retorno incompatible con el tipo de retorno
  return 10;
}

func invalida()  {
  // error no se esperan valor de regreso
  return 1000;
}
```

```swift
func suma( num1 x : Int, num2 y: Int) -> Int {
    return x + y
}
//funcion resta
// Nombres externos: ninguno
// Nombres internos: x, y
func resta(_ x : Int, _ y: Int) -> Int {
    return x - y
}

//función mul
// Nombres externos: x, y
// Nombres internos: x, y
func mul(x: Int, y: Int) -> Int {
    return x * y
}

//funciones por referencia

// duplica el valor ingresado
func duplicar(_  x: inout Int){
    x += x
}


// duplica los valores de un array
func duplicarA (_  array: inout [Int] ) {
    var i = 0
```