



Compiscript



Descripción General

Este lenguaje se encuentra basado en Typescript, por lo que representa un subset del mismo, con algunas diferencias.



Instrucciones de Configuración

1. **Construir y Ejecutar el Contenedor Docker:** Desde el directorio raíz, ejecuta el siguiente comando para construir la imagen y lanzar un contenedor interactivo:

```
docker build --rm . -t csp-image && docker run --rm -ti -v "$(pwd)/program":/program csp-image
```

2. Entender el Entorno

- El directorio `program` se monta dentro del contenedor.
- Este contiene la **gramática de ANTLR de Compiscript y una versión en BNF**, un archivo `Driver.py` (punto de entrada principal) y un archivo `program.cps` (entrada de prueba con la extensión de archivos de Compiscript).

3. **Generar Archivos de Lexer y Parser:** Dentro del contenedor, compila la gramática ANTLR a Python con:

```
antlr -Dlanguage=Python3 Compiscript.g4
```

4. **Ejecutar el Analizador** Usa el driver para analizar el archivo de prueba:

```
python3 Driver.py program.cps
```

- Si el archivo es sintácticamente correcto, **no se mostrará ningún resultado**.
- Si existen errores, ANTLR los mostrará en la consola.

Características del Lenguaje

Compiscript soporta los siguientes conceptos fundamentales:

✓ Tipos de Datos

```
let a: integer = 10;
let b: string = "hola";
let c: boolean = true;
let d = null;
```



✓ Literales

```
123           // integer
"texto"       // string
true, false   // boolean
null          // nulo
```



✓ Expresiones Aritméticas y Lógicas

```
let x = 5 + 3 * 2;
let y = !(x < 10 || x > 20);
```



✓ Precedencia y Agrupamiento

```
let z = (1 + 2) * 3;
```



✓ Declaración y Asignación de Variables

```
let nombre: string;
nombre = "Compiscript";
```



✓ Constantes (const)

```
const PI: integer = 314;
```



✓ Funciones y Parámetros

```
function saludar(nombre: string): string {
  return "Hola " + nombre;
}
```



✓ Expresiones de Llamada

```
let mensaje = saludar("Mundo");
```



✓ Acceso a Propiedades (.)

```
print(dog.nombre);
```



✓ Acceso a Elementos de Arreglo ([])

```
let lista = [1, 2, 3];  
print(lista[0]);
```



✓ Arreglos

```
let notas: integer[] = [90, 85, 100];  
let matriz: integer[][] = [[1, 2], [3, 4]];
```



✓ Funciones como Closures

```
function crearContador(): integer {  
  function siguiente(): integer {  
    return 1;  
  }  
  return siguiente();  
}
```



✓ Clases y Constructores

```
class Animal {  
  let nombre: string;  
  
  function constructor(nombre: string) {  
    this.nombre = nombre;  
  }  
  
  function hablar(): string {  
    return this.nombre + " hace ruido."  
  }  
}
```



✓ Herencia

```
class Perro : Animal {  
  function hablar(): string {  
    return this.nombre + " ladra."  
  }  
}
```



✓ this

```
this.nombre = "Firulais";
```



✓ Instanciación con new

```
let perro: Perro = new Perro("Toby");
```



✓ Bloques y Ámbitos

```
{  
  let x = 42;  
  print(x);  
}
```



✓ Control de Flujo

if / else

```
if (x > 10) {  
  print("Mayor a 10");  
} else {  
  print("Menor o igual");  
}
```



0



main



compilers-2025 / compiscript / README.md

↑ Top

Preview

Code

Blame



Raw



```
while (x < 5) {  
  x = x + 1;  
}
```



do-while

```
do {  
  x = x - 1;  
} while (x > 0);
```



for

```
for (let i: integer = 0; i < 3; i = i + 1) {  
  print(i);  
}
```



foreach

```
foreach (item in lista) {  
  print(item);  
}
```



break / continue

```
foreach (n in notas) {  
    if (n < 60) continue;  
    if (n == 100) break;  
    print(n);  
}
```



✓ switch / case

```
switch (x) {  
    case 1:  
        print("uno");  
    case 2:  
        print("dos");  
    default:  
        print("otro");  
}
```



✓ try / catch

```
try {  
    let peligro = lista[100];  
} catch (err) {  
    print("Error atrapado: " + err);  
}
```



✓ return

```
function suma(a: integer, b: integer): integer {  
    return a + b;  
}
```



✓ Recursión

```
function factorial(n: integer): integer {  
    if (n <= 1) return 1;  
    return n * factorial(n - 1);  
}
```



Extensión de Archivo

Todos los archivos fuente de Compiscript deben usar la extensión:

program.cps

