Informe Parcial

Curso: Compiladores

Integrantes	Escuela	Asignatura
Sebastian Castro Mamani, Davis Arapa Chua, Piero Delgado Chipana, Miguel Ocharan Coaquira	Carrera Profesional de Ingeniería de Software	Compiladores Semestre: V

1. Introducción

EVOLA es un lenguaje de programación diseñado para combinar la simplicidad y flexibilidad de Python con la eficiencia y el control de bajo nivel característicos de C++. Su nombre evoca evolución y versatilidad, dos conceptos clave en el diseño de esta herramienta.

La motivación detrás de EVOLA surge de la necesidad de contar con un lenguaje que no solo sea poderoso y eficiente a nivel de compilación, sino también accesible para usuarios de todos los niveles. A menudo, los lenguajes de alto rendimiento requieren una curva de aprendizaje pronunciada o una sintaxis compleja que desalienta a los programadores novatos. Por otro lado, los lenguajes simples suelen sacrificar eficiencia o control sobre el hardware.

EVOLA busca posicionarse en un punto de equilibrio entre ambos extremos: ofrece estructuras claras, modernas y concisas que facilitan la escritura de código limpio, pero sin renunciar a capacidades de compilación eficientes, control explícito sobre el flujo de ejecución y estructuras de bajo nivel cuando se necesiten.

La combinación de características provenientes de Python y C++ brinda a los desarrolladores la posibilidad de escribir código expresivo y potente sin sacrificar rendimiento ni claridad.

En este informe se presenta la especificación léxica y sintáctica de EVOLA, estableciendo las bases para el desarrollo de su compilador.

2. Especificación Léxica

A continuación, se presentan los tokens definidos para el lenguaje **EVOLA**, junto con sus respectivas expresiones regulares y descripciones:

Token	Expresion	Descripcion
var_float	`[0-9]+\.[0-9]*`	Variables de tipo decimal (números reales).
var_bool	`[0-1]`	Variables booleanas (0 o 1).
var_int	`[1-9][0-9]*`	Variables de tipo entero positivo.
literal	`"."`	Variable tipo string.
id	`[a-zA-Z]([a-zA-Z0-9]*)`	Identificadores válidos para variables y funciones.
oSuma	`+`	Operador de suma.
oResta	`-`	Operador de resta.
oMult	`*`	Operador de multiplicación.
oDiv	`/`	Operador de división.
oResiduo	`\%`	Operador módulo (residuo de división).
oOR	`|`	Operador lógico OR.
oAND	`&`	Operador lógico AND.
oMayor	`>`	Mayor que.
oMenor	`<`	Menor que.
oIgual	`=`	Operador de asignación o comparación.
oDiff	`!`	Operador de negación o diferente de.
oDot	`°`	Punto decorativo o separador personalizado.
oComa	`,`	Separador de elementos.
oSemi_coma	`;`	Fin de instrucción.
corchLeft	`[`	Corchete izquierdo.
corchRight	`]`	Corchete derecho.
parLeft	`(`	Paréntesis izquierdo.
parRight	`)`	Paréntesis derecho.
keyLeft	{	Llave izquierda.
keyRight	}	Llave derecha.
For	`for`	Palabra clave para bucle.
While	`while`	Palabra clave para bucle condicional.
Return	`return`	Instrucción para retornar un valor.
Do	`do`	Palabra clave para estructura de control.
In	`cin`	Entrada de datos.
Out	`cout`	Salida de datos.
Void	`void`	Tipo de retorno vacío.
Main	`main`	Función principal del programa.
Print	`print`	Instrucción para imprimir.
If	`if`	Condicional.
Else	`else`	Alternativa al condicional.
false	`false`	Valor booleano falso.
true	`true`	Valor booleano verdadero.
int	`int`	Tipo de dato entero.
float	`float`	Tipo de dato flotante.
bool	`bool`	Tipo de dato booleano.
string	`string`	Tipo de dato cadena.

3. Gramática

A continuación, se presenta la gramática del lenguaje **EVOLA**, definida como una gramática libre de contexto (CFG), que será utilizada en la etapa de análisis sintáctico del compilador.

Notación

- Los símbolos no terminales están escritos en minúsculas.
- Los símbolos terminales se indican entre comillas dobles o como tokens definidos.
- → indica una producción.
- | representa una alternativa.

Producciones principales

```
programa          funciones mainF

funciones         funcion funciones
                |

funcion           tipo Id ( parametros ) bloque

parametros        parametro parametros_rest
                |

parametros_rest   , parametro parametros_rest
                |

parametro         tipo Id

mainF             void main ( ) bloque

bloque            { instrucciones }

instrucciones     argumento instrucciones
                |

argumento         If
                | While
                | For
                | Return
                | Print
                | tipo Id declaracion

declaracion       asignacion
                | = exp ;
                | ;

asignacion        Id = exp ;

If                if ( exp ) bloque Else

Else              else bloque
                |

While             while ( exp ) bloque

For               for ( asignacion ; exp ; asignacion ) bloque
```

```
Return          return exp' ;

exp'            exp
              |

Print           print ( exp ) ;

tipo            int
              | float
              | bool
              | string
```

Expresiones

```
exp             E

E               C E'
E'              || C E'
              |

C               R C'
C'              & R C'
              |

R               T R'
R'              = T R'
              | < T R'
              | > T R'
              | <= T R'
              | >= T R'
              | == T R'
              | != T R'
              |

T               F T'
T'              + F T'
              | - F T'
              |

F               A F'
F'              * A F'
              | / A F'
              | % A F'
              |

A               ( L )
              | Id B
              | num
              | true
              | false

L               E L'
              |
```

```
L'                , E L'
              |

B                 ( L )
              |

Id                id
```

Descripción general

Esta gramática contempla:

- Declaración y definición de funciones, incluyendo la función principal `main`.
- Instrucciones estructurales: condicionales, ciclos, retornos y salidas por consola.
- Tipos de datos básicos (`int`, `float`, `bool`, `string`).
- Operadores lógicos, relacionales, aritméticos y agrupaciones de expresiones.
- Soporte para llamadas a funciones y paso de parámetros.