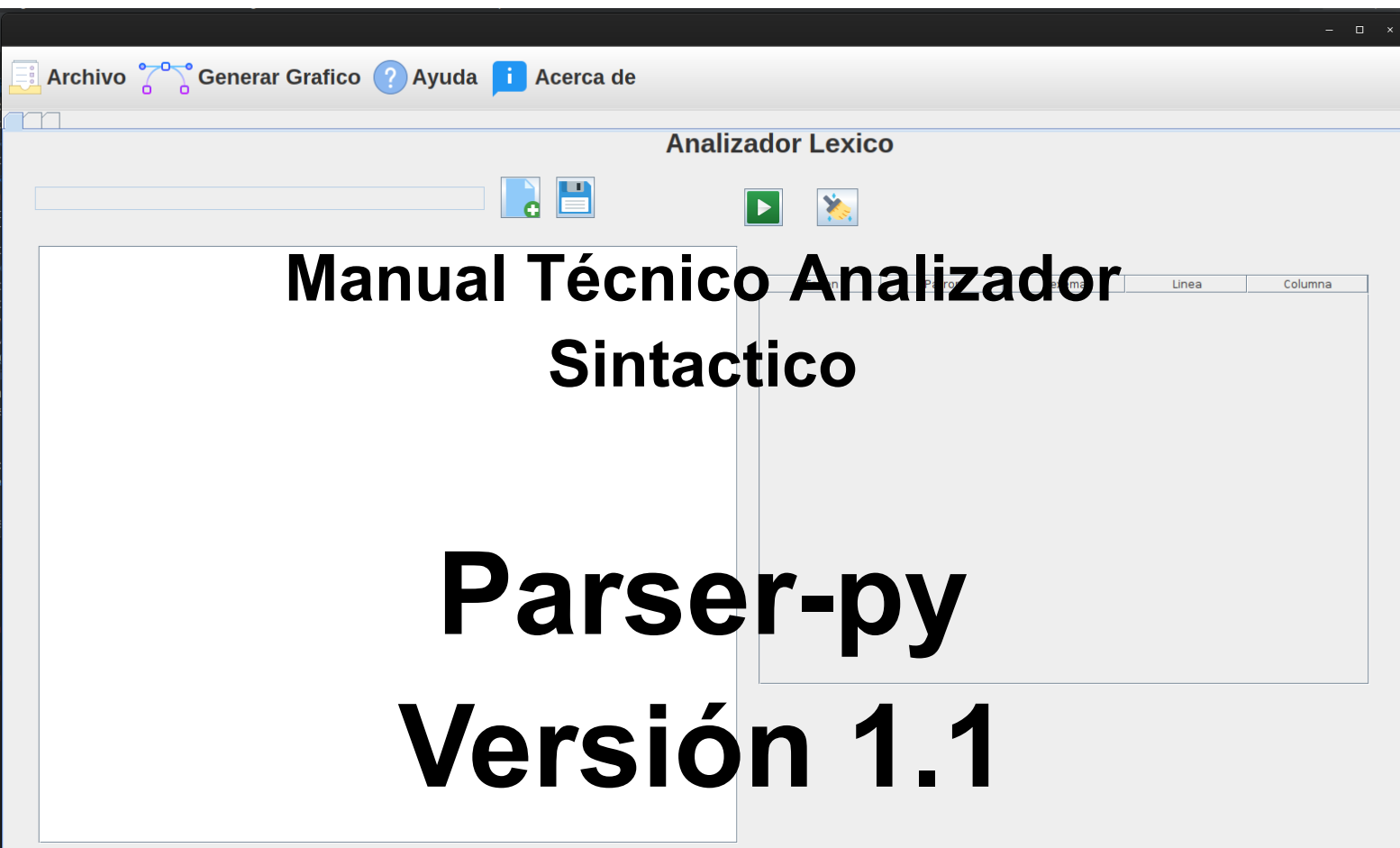


UNIVERSIDAD SAN CARLOS DE GUATEMALA
CENTRO UNIVERSITARIO DE OCCIDENTE
DIVISIÓN DE CIENCIAS DE LA INGENIERÍA
Ingeniería en Ciencias y Sistemas.
Ing. Christian Quiroa
Auxiliar: Julio Fernando Ixcoy



Byron Fernando Torres Ajxup

201731523



Índice

Introducción	3
Requisitos	3
Diseño	4
El analizador utiliza las siguientes estructuras de datos	4
Implementación	4
Herramientas usadas para el desarrollo	5
Descarga y ejecución.	5
Diagramas UML	6

Introducción

Este manual técnico describe el analizador sintáctico realizado por [tu nombre]. El analizador sintáctico es una herramienta que se utiliza para analizar la estructura sintáctica de un lenguaje. En este caso, el analizador sintáctico está diseñado para analizar el lenguaje de programación Python.

Requisitos

El analizador sintáctico tiene los siguientes requisitos:

- Soporta el lenguaje de programación Python.
- Realiza un análisis sintáctico descendente.
- Reconoce los siguientes tipos de expresiones:
 - Variables
 - Operadores
 - Funciones
 - Condicionales
 - Bucles

Diseño

El analizador sintáctico está diseñado utilizando un enfoque propio, partiendo de la realización utilizando la lógica propia sin enfoques predefinidos puesto que ya se tenía el análisis léxico, el sintáctico solo se encargó de llamar token por token para analizar si estaban en el orden adecuado, utilice una clase para el análisis y dividir en varios métodos para una mejor lectura de código, cada acción de bloque realizaba operaciones diferentes por ejemplo para verificar la estructura de asignación, ciclo for, sentencia if, etc.

El analizador utiliza las siguientes estructuras de datos

- Pila: Para almacenar los nodos del árbol sintáctico que aún no se han analizado (ArrayList).
- Tabla de símbolos: Para almacenar las variables y funciones definidas en el programa (HashMap).

Implementación

Implementación

El analizador sintáctico está implementado utilizando el lenguaje de programación Java. El analizador se divide en una sola clase llamada Sintaxis.

La clase Sintaxis implementa el analizador sintáctico descendente. La clase utiliza las siguientes estructuras de datos:

ArrayList: Para almacenar los tokens en el analizador léxico.

HashMap: Para almacenar las variables y funciones definidas en el programa.

La clase Sintaxis tiene varios métodos que se encargan de verificar la posición de cada token para el análisis sintáctico.

A continuación, se describen los métodos más importantes de la clase Sintaxis:

****public void analiza():** Este método es el punto de entrada del analizador sintáctico. El método analiza el primer token del ArrayList y llama a los métodos apropiados para analizar el resto del programa.

****public void idAnalyzer():** Este método analiza una expresión. El método utiliza la tabla de símbolos para verificar que las variables y funciones utilizadas en la expresión sean válidas.

****public void opAsignacion():** Este método analiza una asignación. El método utiliza la tabla de símbolos para asignar el valor de la expresión a la variable.

****public void sentenciIF():** Este método analiza una instrucción condicional. El método utiliza la tabla de símbolos para verificar que las variables utilizadas en la condición sean válidas.

****public void bucleFor():** Este método analiza una instrucción de bucle. El método utiliza la tabla de símbolos para verificar que las variables utilizadas en el bucle sean válidas.

Entre estas hay más métodos que analizan la posición del token en durante el análisis.

Pruebas

El analizador sintáctico se ha probado con una serie de casos de prueba. Los casos de prueba incluyen expresiones simples y complejas, así como funciones y condicionales.

Excepciones y errores

El analizador sintáctico puede generar las siguientes excepciones y errores:

Error de sintaxis: Se genera cuando el analizador encuentra un error en la sintaxis del programa.

Error Léxico: Se genera cuando el analizador no reconoce los tokens previamente personalizados usando la librería regex de java.

Imagen de un error de sintaxis en la línea y columna provocado

Archivo Generar Grafico Ayuda Acerca de

Analizador Lexico-Sintactico

Position: 73, 22

```
43 comparacion = "x es igual a y"
44
45 #ciclos
46 for i in range(10, 15):
47     print(f"Número {i}")
48
49 contador = 10
50 while contador < 15:
51     print(f"Contador {contador}")
52     contador += 1
53
54 # funciones
55 def multiplicar(a, b):
56     return a * b
57
58 def dividir(a, b):
59     if b != 0:
60         return a / b
61     else:
62         return "No se puede dividir entre cero"
63
64 def longitud_texto(texto):
65     return len(texto)
66
67 def es_par(numero):
68     return numero % 2 == 0
69
70 def convertir_a_lista(*args):
71     for arg in args:
72         print(arg)
73     return list(args)
```

Tokens Reconocidos

Tokens	Patron	Lexema	Linea	Columna
ID	[a-zA-Z_][a-zA-Z0-9_...]	variable1	1	10
OP_AS	=	=	1	12
ENTERO	?[d+]	100	1	16
ID	[a-zA-Z_][a-zA-Z0-9_...]	texto	2	6
OP_AS	=	=	2	8
CONS	"[^"]*" '[^']*'	"Hola Mundo"	2	21
ID	[a-zA-Z_][a-zA-Z0-9_...]	booleano	3	9
OP_AS	=	=	3	11
OP_AS	=	=	3	16
ID	[a-zA-Z_][a-zA-Z0-9_...]	lista_vacia	4	12
OP_AS	=	=	4	14
ID	[a-zA-Z_][a-zA-Z0-9_...]	[4	16
ID	[a-zA-Z_][a-zA-Z0-9_...]	[4	17
ID	[a-zA-Z_][a-zA-Z0-9_...]	diccionario	5	12
OP_AS	=	=	5	14
CONS	{	{	5	23
OT	:	:	5	24
CONS	"[^"]*" '[^']*'	"valor"	5	32
OT	}	}	5	33
COM	#	#	7	10
KEY_WD	reservadas	if	8	3
ID	[a-zA-Z_][a-zA-Z0-9_...]	variable1	8	13
OP_COMP	>=<=> < > !=	>	8	15
ENTERO	?[d+]	50	8	18
OT	:	:	8	19
ID	[a-zA-Z_][a-zA-Z0-9_...]	resultado	9	14
OP_AS	=	=	9	16
CONS	"[^"]*" '[^']*'	"mayor"	9	24
KEY_WD	reservadas	else	10	5
OT	:	:	10	6
ID	[a-zA-Z_][a-zA-Z0-9_...]	resultado	11	14
OP_AS	=	=	11	16
CONS	"[^"]*" '[^']*'	"menor"	11	24
KEY_WD	reservadas	if	13	3
ID	[a-zA-Z_][a-zA-Z0-9_...]	texto	13	6

Mensaje

Error de sintaxis en línea: 47 y columna: 24

Aceptar

Ejemplo de tabla de símbolos

Archivo Generar Grafico Ayuda Acerca de

Analizador Lexico-Sintactico

Position: 73, 22

```
43 comparacion = "x es igual a y"
44
45 #ciclos
46 for i in range(10, 15):
47     print(f"Número {i}")
48
49 contador = 10
50 while contador < 15:
51     print(f"Contador {contador}")
52     contador += 1
53
54 # funciones
55 def multiplicar(a, b):
56     return a * b
57
58 def dividir(a, b):
59     if b != 0:
60         return a / b
61     else:
62         return "No se puede dividir entre cero"
63
64 def longitud_texto(texto):
65     return len(texto)
66
67 def es_par(numero):
68     return numero % 2 == 0
69
70 def convertir_a_lista(*args):
71     for arg in args:
72         print(arg)
73     return list(args)
```

Tokens Reconocidos

Tokens	Patron	Lexema	Linea	Columna
ID	[a-zA-Z_][a-zA-Z0-9_...]	variable1	1	10
OP_AS	=	=	1	12
ENTERO	?[d+]	100	1	16
ID	[a-zA-Z_][a-zA-Z0-9_...]	texto	2	6
OP_AS	=	=	2	8
CONS	"[^"]*" '[^']*'	"Hola Mundo"	2	21
ID	[a-zA-Z_][a-zA-Z0-9_...]	booleano	3	9
OP_AS	=	=	3	11
OP_AS	=	=	3	16
ID	[a-zA-Z_][a-zA-Z0-9_...]	lista_vacia	4	12
OP_AS	=	=	4	14
ID	[a-zA-Z_][a-zA-Z0-9_...]	[4	16
ID	[a-zA-Z_][a-zA-Z0-9_...]	[4	17
ID	[a-zA-Z_][a-zA-Z0-9_...]	diccionario	5	12
OP_AS	=	=	5	14
CONS	{	{	5	16
OT	:	:	5	23
CONS	"[^"]*" '[^']*'	"clave"	5	24
OT	}	}	5	32
COM	#	#	7	10
KEY_WD	reservadas	if	8	3
ID	[a-zA-Z_][a-zA-Z0-9_...]	variable1	8	13
OP_COMP	>=<=> < > !=	>	8	15
ENTERO	?[d+]	50	8	18
OT	:	:	8	19
ID	[a-zA-Z_][a-zA-Z0-9_...]	resultado	9	14
OP_AS	=	=	9	16
CONS	"[^"]*" '[^']*'	"mayor"	9	24
KEY_WD	reservadas	else	10	5
OT	:	:	10	6
ID	[a-zA-Z_][a-zA-Z0-9_...]	resultado	11	14
OP_AS	=	=	11	16
CONS	"[^"]*" '[^']*'	"menor"	11	24
KEY_WD	reservadas	if	13	3
ID	[a-zA-Z_][a-zA-Z0-9_...]	texto	13	6

Mensaje

TABLA DE SIMBOLOS

variable1, Tipo: ENTERO, Valor: 100

estado, Tipo: CONS, Valor: "Inactivo"

resultado, Tipo: CONS, Valor: "menor"

saludo, Tipo: BOOL, Valor: False

clave_estado, Tipo: CONS, Valor: "Clave no existe"

lista_estado, Tipo: CONS, Valor: "Lista no está vacia"

diccionario, Tipo: OT, Valor: {"clave": "valor"}

texto, Tipo: CONS, Valor: "Hola Mundo"

lista_vacia, Tipo: OT, Valor: []

booleano, Tipo: BOOL, Valor: True

contador, Tipo: ENTERO, Valor: 1

x, Tipo: ENTERO, Valor: 15

y, Tipo: ENTERO, Valor: 20

comparacion, Tipo: CONS, Valor: "x es igual a y"

Aceptar

