

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Estructuras De Datos
Sección A
Catedrático: ING. KEVIN ADIEL LAJPOP AJPACAJA
Tutor académico: MOISES GONZALEZ FUENTES



MANUAL DE USUARIO

Pseudo-Parser

Juan Pablo García Ceballos

201901598

Lugar y Fecha: Guatemala, Sacatepéquez 17/08/2022

Índice

Descripción del proyecto.....	3
Requisitos del sistema.....	3
Interfaz Grafica.....	4
Interfaz de Inicio.....	4
ARCHIVO	4
ANALIZAR	5
Código en Python	5
Código en Golang	6
Reportes Errores	7
Reportes Diagrama.....	7

Descripción del proyecto

Este programa es una solución de software, Pseudo-Parser para que el nuevo personal que no conoce los lenguajes de Python y Golang, para que pueda aplicar sus conocimientos en pseudocódigo y utilizando una aplicación, traducir este código y ver cómo se comportan las diferentes sintaxis de cada uno de los lenguajes ya que para cada uno existen diferentes características, utilizar, en el que muestra los resultados y reportes, tanto de errores como el diagrama de flujo y Árbol Sintáctico.

Requisitos del sistema

Mínimos:

CPU	Pentium 2
RAM	1 GB
Internet	Si
Sistema operativo	Windows vista SP2
Extensión de los archivos	".olc"

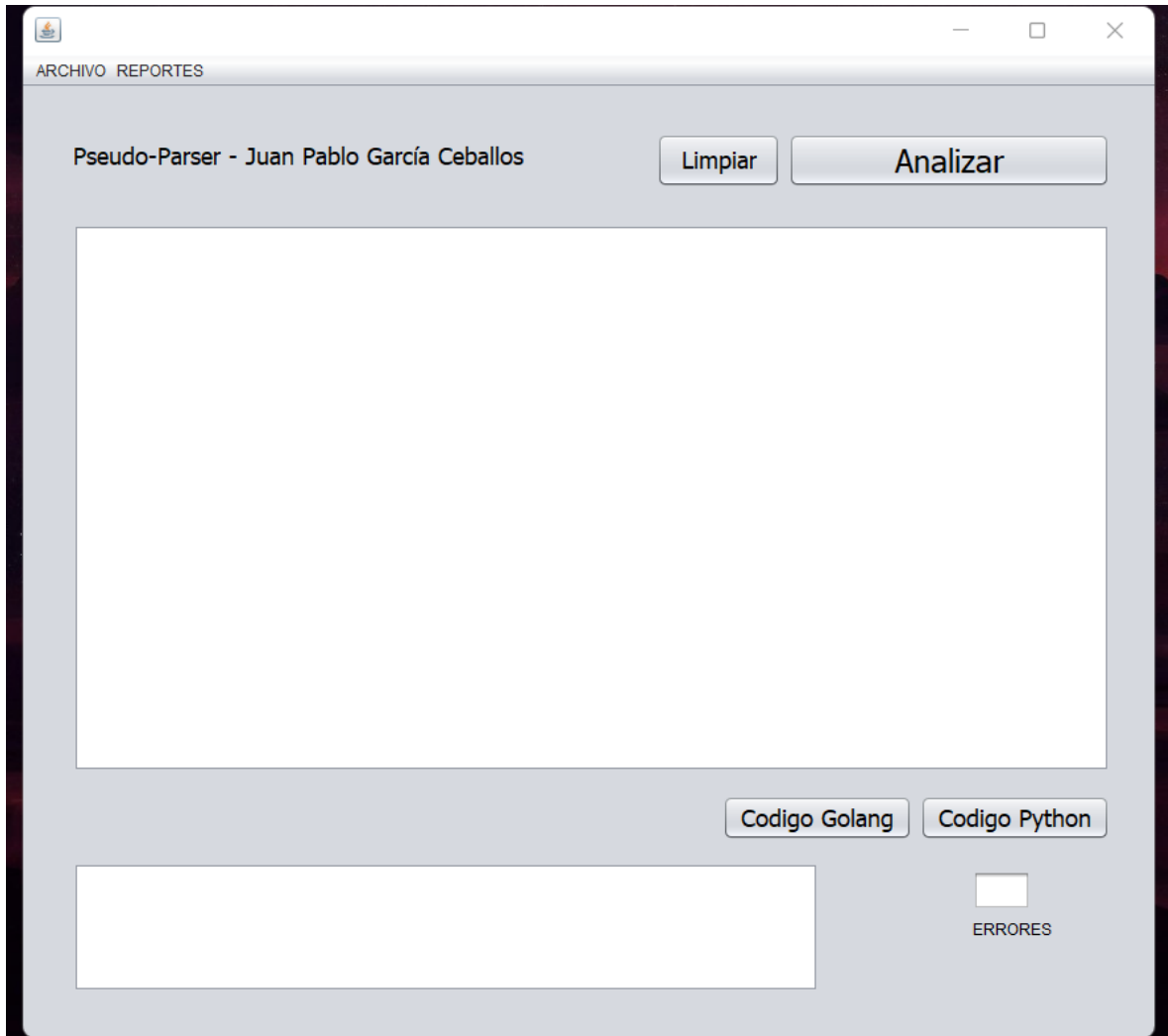
Recomendados:

CPU	Intel Core 2
RAM	4 GB
Internet	Si
Sistema Operativo	Windows 7
Extensión los archivos	".olc"

Interfaz Grafica

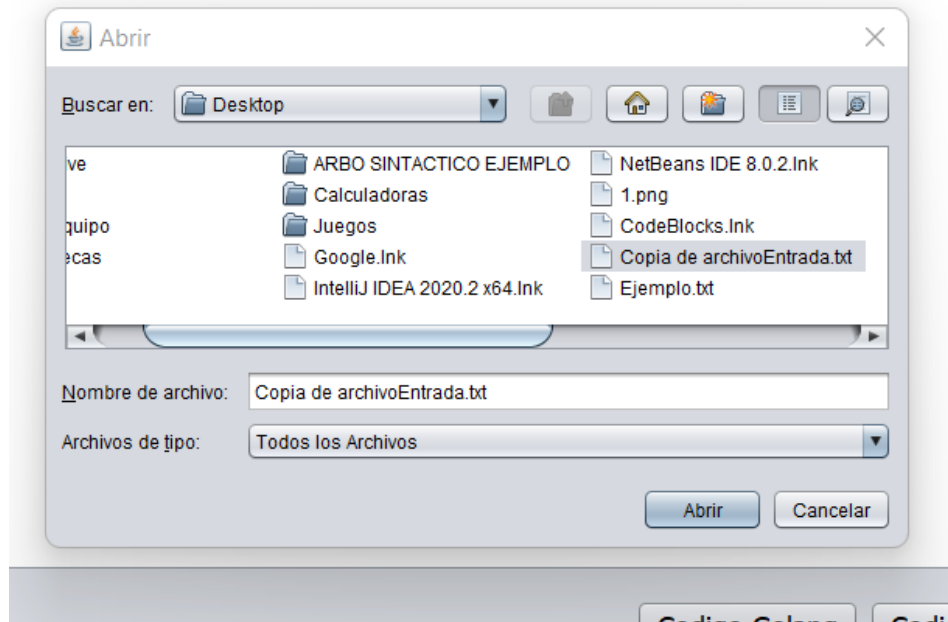
En esta sección se dará una explicación de las herramientas de este programa.

Interfaz de Inicio



BOTONES:

ARCHIVO = Da 2 opciones de abrir un archivo con extensión .olc para su posterior análisis y otra opción de guardar el contenido del área de texto en un archivo.



ANALIZAR: Ya cargado el archivo o escrito el código en el área de texto se puede analizar y ver que tipo de errores existe, léxicos o sintácticos.



Código en Python: Muestra una ventana emergente con el código traducido en Python y genera un archivo .py con el código de este.

```
Codigo en Python
if __name__ == '__main__':
    main()
    _variable1_=5
    _curso1_="olc"
    _curso2_="olc"
    _curso3_="olc"
    _pi1_=3
    _pi2_=3.1
    _pi3_=3.14
    _pi4_=3.141
    _anio1_=1
    _anio2_=9
    _anio3_=4
    _anio4_=5
    _encabezado1_="universidad san carlos de guatemala.-"
    _encabezado2_="escuela de ciencias y sistemas
segundo semestre
"
    _flag1_=True
    _flag2_=False
    _name1_="f"
    _name2_="e"
    _name3_="r"
    _name4_=_name6_="n"
    _name5_="a"
    _operaciones1basica_int=1*(1)
    _operaciones1basica2_=_operaciones1basica_+_operaciones1basica_
    _operaciones1intermedia_int=15*(9*8)+2008*3+9
    _operaciones1avanzadas1_int=((15+9)*9-2008*3+9)
    _operaciones1avanzadas2_int=30*(22*2*2)*(2)
    _operaciones1avanzadas3_int=(30*(2)*(2))
    _operaciones1avanzadas4_int=(30*(10+9+4*2-1))*(2)
    _operaciones1avanzadas5_int=(8*8)*(1+5+6)
    _operaciones1rela3_int=_operaciones1basica_> 8
    _operacionrela3_int=_operaciones1basica_<= 8
    _operacionrela3_int=_operaciones1basica_<= 8
    _operacionrela4_int=_operaciones1basica_== 8
    _operacionrela5_int=_operaciones1basica_== _operaciones1basica_
    _operacionrela6_int=_operaciones1basica_== _operaciones1basica_+1
    _operacionrela7_int=_operaciones1basica_== (_operaciones1basica_)*(8+5)
    _v2_/_v3_="estas cadenas deben ser diferentes"
    _curso1_+_curso2_+_curso3_="organizacion de lenguajes y compiladores 1"
    _curso1_+_curso2_+_curso3_+_encabezado1_
    print(_encabezado2_)
    print(...)
    print(_anio1_)
    print(_anio2_)
```

Código en Golang: Muestra una ventana emergente con el código traducido en Golang y genera un archivo .go con el código de este.

```
Codigo en Golang
Package main
import(
    "math"
    "fmt"
)
var _variable1_int=5
var _curso1_int="olc"
var _curso2_int="olc"
var _curso3_int="olc"
var _pi1_int=3
var _pi2_int=3.1
var _pi3_int=3.14
var _pi4_int=3.141
var _anio1_int=1
var _anio2_int=9
var _anio3_int=4
var _anio4_int=5
var _encabezado1_int="universidad san carlos de guatemala.-"
var _encabezado2_int="escuela de ciencias y sistemas
segundo semestre
"
var _flag1_int=null
var _flag2_int=null
var _name1_int="f"
var _name2_int="e"
var _name3_int="r"
var _name4_+_name6_int="n"
var _name5_int="a"
var _operaciones1basica_int=1*(1)
var _operaciones1basica2_int=_operaciones1basica_+_operaciones1basica_
var _operaciones1intermedia_int=15*(9*8)+2008*3+9
var _operaciones1avanzadas1_int=((15+9)*9-2008*3+9)
var _operaciones1avanzadas2_int=math.Pow(float64(30),float64(22*2*2))*(2)
var _operaciones1avanzadas3_int=(math.Pow(float64(30),float64(2)))*(2)
var _operaciones1avanzadas4_int=(math.Pow(float64(30),float64(10+9+4*2-1))*(2)
var _operaciones1avanzadas5_int=(math.Pow(float64(8),float64(1+5+6)))
var _operacionrela3_int=_operaciones1basica_> 8
var _operacionrela3_int=_operaciones1basica_<= 8
var _operacionrela3_int=_operaciones1basica_<= 8
var _operacionrela4_int=_operaciones1basica_== 8
var _operacionrela5_int=_operaciones1basica_== _operaciones1basica_
var _operacionrela6_int=_operaciones1basica_== _operaciones1basica_+1
var _operacionrela7_int=_operaciones1basica_== (_operaciones1basica_)*(8+5)
_v2_/_v3_="estas cadenas deben ser diferentes"
_curs01_+_curs02_+_curs03_="organizacion de lenguajes y compiladores 1"
_curs01_+_curs02_+_curs03_+_fmtPrint(_encabezado1_)
fmtPrint(_encabezado2_)
fmtPrint(...)
fmtPrint(_anio1_)
fmtPrint(_anio2_)
fmtPrint(_anio3_)
```

Reportes Errores: Muestra el archivo html con todos los errores léxicos y sintácticos del análisis realizado.



Juan Pablo Garcia ceballos

Carne: 201901598

Organización de Lenguajes y Compiladores 1

Proyecto 1

TABLA DE ERRORES

No.	TIPO DE ERROR	LEXEMA	DESCRIPCION
-----	---------------	--------	-------------

Reportes Diagrama: Muestra el árbol sintáctico creado con el análisis sintáctico

