MANUAL TECNICO

MinorC

TABLA DE CONTENIDO

INTRODUCCIÓN

MinorC es un IDE que funciona como el lenguaje C unicamente que esta mas corto en insturccionese, el IDE se encarga de traducir el lenguaje MinorC a el lenguaje de Augus.

Realiza una conversión de un lenguaje de alto nivel a un lenguaje de mas bajo nivel que Augus.

Tecnologías utilizadas:

- **Sistema Operativo:** Ubuntu 20 LTS
- **Lenjuage de programación:** Python3
- **GUI:** Tkinter
- **IDE:** Visual Studio Code
- **Graficas:** Graphviz

Funcionamiento:

La aplicación obtiene la entrada que está escrita en el lenguaje MinorC y este traduce la entrada a el lenguaje Augus y generar la compilación de Augus, Utilizando para ello una gramatica única para MinorC y otra gramatica para la entrada a reconocer en Augus.

Construcción:

PLY: es una herramienta de análisis que está escrita en Python. Contiene un analizado léxico y un analizador sintáctico con una gramática ascendente

Dentro del archivo que compilamos con PLY necesitamos definir nuestros tokens y nuestras producciones y nuestras palabras reservadas:

```
reservadas = {
    'main'  :   'MAIN',
    'goto'  :   'GOTO',
    'unset' :   'UNSET',
    'print' :   'PRINT',
    'read'  :   'READ',
    'exit'  :   'EXIT',
    'int'   :   'INT',
    'float' :   'FLOAT',
    'char'  :   'CHAR',
    'abs'   :   'ABS',
    'array' :   'ARRAY',
    'if'    :   'IF'
```

```
}
```

```
tokens = [
    'LABEL',

    #VARIABLES
    'TEMPORAL','PARAMETRO','RETURN','RA','PILA','PUNTEROPILA',

    #TIPOS
    'ENTERO','DECIMAL','CADENA','CARACTER',

    #SIMBOLOS
    'DOSPUNTOS','PUNTOCOMA','IGUAL','ABREPARENTESIS','CIERRAPARENTESIS','MENOS',
    'MAS','MUL','DIV','AMPERSAN','RESIDUO','NOT','AND','OR','XOR','COMPARACION',
    'DIFERENTE','MAYORIGUAL','MENORIGUAL','MAYOR','MENOR','NOTBIT','ORBIT','XORBIT',
    'SHIFTIZQ','SHIFTDER','ABRECORCHETE','CIERRACORCHETE'
]
# Tokens
t_TEMPORAL=             r'\$t[0-9]+'
t_PARAMETRO=            r'\$a[0-9]+'
t_RETURN=               r'\$v[0-9]+'
t_RA=                   r'\$ra'
t_PILA=                 r'\$s[0-9]'
t_PUNTEROPILA=          r'\$sp'
t_DOSPUNTOS=            r'\:'
t_PUNTOCOMA=            r'\;'
t_IGUAL=                r'\='
t_ABREPARENTESIS =      r'\('
t_CIERRAPARENTESIS =    r'\)'
t_MENOS =               r'\-'
t_MAS =                 r'\+'
t_MUL =                 r'\*'
t_DIV =                 r'\/'
t_RESIDUO =             r'\%'
```

Producciones

```
def p_main(t):
    'main   :  MAIN DOSPUNTOS instrucciones'
```

```
def p_instrucciones_listado(t):
    '''instrucciones    :   instrucciones   instruccion'''
```

```
def p_goto_instruccion(t):
    '''goto_instruccion     :   GOTO LABEL PUNTOCOMA'''
```

PLY nos permite crear nuestras clases abstracas mientras recorremos la gramatica, con esto nos facilita para que podamos obtener los diferentes objetos que seran traducidos en la construccion de nuestro codigo mediante las reglas semanticas.

```
def p_expresion_not_bit(t):
    '''expresion_bit            :   NOTBIT  expresion_numerica'''
    id = inc()
    t[0] = ExpresionNotBit(t[2],id,t.lexer.lineno)
```

En el ejemplo anterior vemos que creamos un objeto ExpresionNotBit cuando la gramatica reconocio la produccion.

Dicho objeto es agregado a una lista que sera nuesto arbol AST el cual recorremos para crear la logica del program

Recuperación de Errores

Para la recuperación de errores se utilizó el modo pánico que busca un token específico para poder salir de instrucción que le causo el error. Con esto se logra que el programa termine la compilación sin quedarse por el error.

```
def p_error(t):
    if t:
        error = Error("SINTACTICO","Error sintactico en: '%s'" % t.value ,t.lexer.li
neno)
        tablaerrores.agregar(error)
        parser.errok()
    else:
        error = Error("SINTACTICO","Se esperaba el simbolo ';'",-1)
        tablaerrores.agregar(error)
        parser.restart()
```

Asociación de operadores y precedencia

```
precedence = (
    ('left','COMA'),
    ('right','IGUAL','MULTIPLYAND','DIVIDEAND','MODULUSAND','PLUSIGUAL','MINUSI
GUAL','LEFTSHIFTAND','RIGHTSHIFTAND','BITWISEAND','BITWISEEXCLUSIVE','BITWISEIN
CLUSIVE'),
    ('right','TERNARIO','DOSPUNTOS'),
    ('left','ORBIT'),
    ('left','AMPERSAN'),
    ('left','OR'),
    ('left','XOR'),
    ('left','AND'),
    ('left','COMPARACION','DIFERENTE'),
    ('left','MAYORIGUAL','MENORIGUAL','MAYOR','MENOR'),
    ('left','SHIFTIZQ','SHIFTDER'),
    ('left','ABSOLUTO'),
    ('left','MAS','MENOS'),
    ('left','MUL','DIV','RESIDUO'),
    ('right','NEGATIVO'),
    ('right','MENOS','MAS','NOT','NOTBIT','INCREMENT','DECREMENT','AND'),
    ('left','ABREPARENTESIS','CIERRAPARENTESIS','ABRELLAVE','CIERRALLAVE','INCR
EMENT','DECREMENT')
    )
```