AutoHouse

AutoHouse - Linguagem para Automação Residencial

Disciplina de Compiladores;
Prof: Rodrigo Evangelista
Analisador Léxico e Sintático;

Patrick Bastos;
Perses Vilhena.

Analisador Léxico

Linguagem : AutoHouse - Automação Residêncial

Linguagem de Desenvolvimento: *PHP + Python*

AutoHouse

- A linguagem AutoHouse visa ajudar na automação residencial;
- Ela permite sincronizar, ligar, desligar e programar ações do seu dia a dia em sua residência.

AutoHouse

Tabela de Tokens:

Token	Lexema	Regex	Descrição
inicio	auto	(auto)	Início do programa
fim	house	(house)	Fim do Programa
var	#a, #ab,#abc...	letra(letra)* ((#)[a-z]|[A-Z])	Variável dentro do código
se	se	(se)	Condicional
senão	senao	(senao)	Condicional
op_log	&,|,no	&,|,no (\$)|(|)|(no)	Operadores lógicos
op_ari	+,-,*,/	+,-,*,/ (\+)|(\-)|(*)|(\/)	Operadores aritméticos

AutoHouse

Token	Lexema	Regex	Descrição
att	=	= (\=)	Atribuição
valor	10,10.0,10.00,”abc”	num(num)*, num*.num*, letra(letra)* (([0-9])+(\.)?([0-9])*)|((\"){1}([A-z]|[0-9])*(\"){1})	Valores
ap	(	((\()	Abre parênteses
fp	)	) (\))	Fecha parênteses
fl	;	; (\;)	Final de linha

AutoHouse

Token	Lexema	Regex	Descrição
op_rel	==,>,<,>=,<=,!=	==,>,<,>=,<=,!= (\==)|(\<)|(\>)|(\>=)|(\<=)|(\!=)	Operadores relacionais
for	for	for (for)	Laço de repetição
af	[	[(\[)	Abre função
ff	]	] (\])	Fecha função
comment	%%abc*%%	((%%)(\s|\w|\d)*(%%))	Comenta a linha
leia	leia	(leia)	Leitura de dados
escreva	escreva	(escreva)	Impressão dos dados

AutoHouse

Palavras reservadas e declaração dos tokens

```
reserved = {
    'auto':'INICIO',
    'house':'FIM',
    'leia':'LEIA',
    'for':'FOR',
    'escreva':'ESCREVA',
    'se':'SE',
    'senao':'SENAO'
}

tokens = ['ID', 'NUMBER', 'VAR', 'OP_LOG', 'OP_ARI', 'OP_REL', 'ATT', 'VALOR', 'FL', 'AP', 'FP', 'AF', 'FF', 'COMMENT']
```

AutoHouse

Regex

```
tokens = list(reserved.values())+tokens

t_ignore = ' \t'
t_VAR = r'(\#([A-z]+))'
t_TIPO = r'(int)|(float)|(double)'
# t_SE = r'(se)'
# t_SENAO = r'(senao)'
t_OP_LOG = r'(\&)|(\|)|(no)'
t_OP_ARI = r'\+|\-|\    *|\/'
t_ATT = r'(\=)'
t_VALOR = r'(([0-9])+(\.)?([0-9])*)|((\"){1}([A-z]|[0-9]|\s)*(\"){1})'
t_FL = r'(\;)'
t_AP = r'(\()'
t_FP = r'(\))'
t_AF = r'(\[)'
t_FF = r'(\])'
t_OP_REL = r'(\=\=)|(\<)|(\>)|(\>=)|(\<=)|(\!=)'
```

AutoHouse

Gramática

```
def p_program(p):
    '''program : INICIO estrutura FIM'''
    p[0] = p[1:]

def p_estrutura(p):
    '''estrutura : inst
                 | inst estrutura'''
    p[0] = p[1:]

def p_inst(p):
    '''inst : atrib
            | entrada
            | saida
            | cond
            | repeat'''
    p[0] = p[1:]

def p_atrib(p):
    '''atrib : VAR ATT expre FL'''
    p[0] = p[1:]
```

```
def p_entrada(p):
    '''entrada : LEIA AP VAR FP FL'''
    p[0] = p[1:]

def p_saida(p):
    '''saida : ESCREVA AP VAR FP FL'''
    p[0] = p[1:]

def p_cond(p):
    '''cond : SE AP VAR FP AF estrutura FF
            | SE AP VAR FP AF estrutura FF SENAO AF estrutura FF
            | SE AP VAR OP_REL VAR FP AF estrutura FF
            | SE AP VAR OP_REL VAR FP AF estrutura FF SENAO AF estrutura FF
            | SE AP VAR OP_LOG VAR FP AF estrutura FF
            | SE AP VAR OP_LOG VAR FP AF estrutura FF SENAO AF estrutura FF'''
    p[0] = p[1:]

def p_repeat(p):
    '''repeat : FOR AP NUMBER FP AF estrutura FF
              | FOR AP VAR FP AF estrutura FF'''
    p[0] = p[1:]

def p_expre(p):
    '''expre : expre OP_ARI expre
             | AP expre FP
             | VAR
             | VALOR
             | NUMBER'''
    p[0] = p[1:]
```