

Gabriel Ribeiro Bernardi – 11821BCC036

Guilherme Soares Correa – 11821BCC026

Pedro Henrique Gonçalves Teixeira – 11821BCC008

1ª Etapa do Projeto

Especificação da linguagem:

Definição da gramática livre de contexto (GLC) com as estruturas da linguagem especificada

Gramática livre de contexto (GLC)

$S = \langle S \rangle$

$T = \{ID, FUNCTION, \{ \{, \} \}, :, INT, CHAR, FLOAT, \{ \{, \} \}, SE, ENTAO, SENAO, (,), ENQUANTO, FACA, REPITA, ATE, =, <, >, <>, ==, <=, >=, +, -, *, /, ^, CONSTINT, CONSTFLOAT, CONSTCHAR, RELOP \}$

$N = \{ \langle S \rangle, \langle bloco \rangle, \langle declaracaoVariavel \rangle, \langle sequenciaComandos \rangle, \langle ComandoAtribuicao \rangle, \langle ComandoRepeticao \rangle, \langle comandoCondicional \rangle, \langle lista_ids \rangle, \langle tipo \rangle, \langle cond \rangle, \langle expressao \rangle, \langle Condition \rangle, \langle ConditionOp \rangle \}$

P :

$S \rightarrow FUNCTION ID() \text{ bloco}$

$sequenciaComandos \rightarrow ComandoAtribuicao \mid comandoCondicional \mid ComandoRepeticao$

$bloco \rightarrow \{ declaracaoVariavel \text{ sequenciaComandos } \}$

$declaracaoVariavel \rightarrow \epsilon \mid tipo : lista_ids$

$lista_ids \rightarrow ID; \mid ID, lista_ids$

$tipo \rightarrow CHAR \mid FLOAT \mid INT$

$comandoCondicional \rightarrow SE (cond) ENTAO bloco \mid SE (cond) ENTAO bloco SENAO bloco \mid SE (cond) ENTAO sequenciaComandos \mid SE (cond) ENTAO bloco SENAO sequenciaComandos$

$ComandoRepeticao \rightarrow ENQUANTO (cond) FACA bloco \mid REPITA bloco ATE(cond) \mid ENQUANTO (cond) FACA sequenciaComandos \mid REPITA sequenciaComandos ATE(cond)$

$ComandoAtribuicao \rightarrow ID = expressao \mid ID = CONSTCHAR;$

$cond \rightarrow expressao RELOP expressao$

expressao -> expressao + expressao | expressao - expressao | expressao * expressao
| expressao / expressao | expressao ^ expressao | ID | CONSTINT | CONSTFLOAT |
(expressao)

Identificação dos tokens usados na gramática

Identificação dos tokens usados na gramática	
Token	Atributo
id	Posição na tabela de símbolos
function	-
{	-
:	-
	-
int	-
char	-
float	-
,	-
;	-
se	-
entao	-
senao	-
(-
)	-
enquanto	-
faca	-
repita	-
ate	-
=	-
+	-
-	-
*	-
/	-
^	-
constInt	Valor da constante
consFloat	ponto fixo (PF) ou notação científica (NC)
consChar	Valor da constante
relop	Operador (EQ,NE,LT,LE,GT,GE)

Definição dos padrões (expressões regulares) de cada token (inclusive os tokens especiais)

Definição dos padrões (expressões regulares)	
Token	Expressão Regular
id	→ [a-zA-Z][a-zA-Z0-9]*
function	→ function
{	→ {
}	→ }
:	→ :
int	→ int
char	→ char
float	→ float
,	→ ,
;	→ ;
se	→ se
entao	→ entao
senao	→ senao
(→ (
)	→)
enquanto	→ enquanto
faca	→ faca
=	→ =
+	→ +
-	→ -
*	→ *
/	→ /
^	→ ^
constInt	→ [0-9][0-9]*
consFloat	→ [0-9][0-9]*\.[0-9][0-9]*(E[+-]?[0-9][0-9]*)?
consChar	→ '[.]'
relop	→ [== <> < > >= <=]

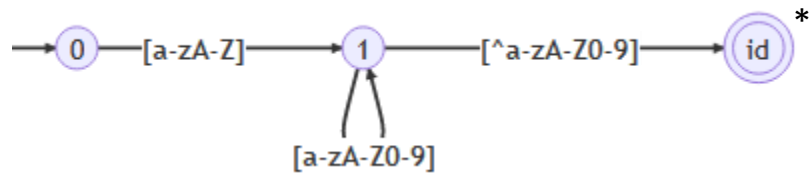
2ª Etapa do Projeto

Análise Léxica (especificação)

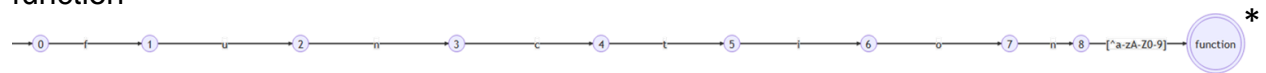
Elaboração do diagrama de transição

Gerar um diagrama de transição para cada token

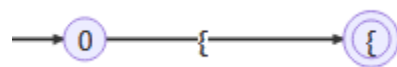
id



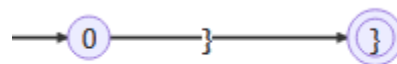
function



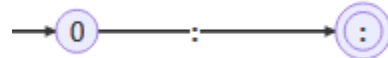
{



}



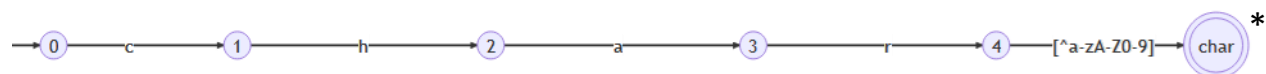
:



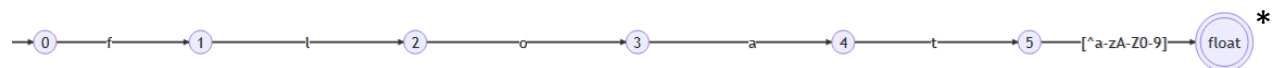
int



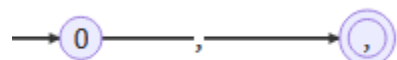
char



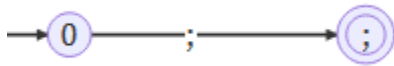
float



,



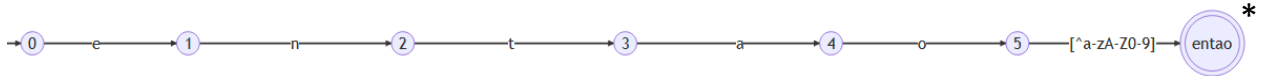
;



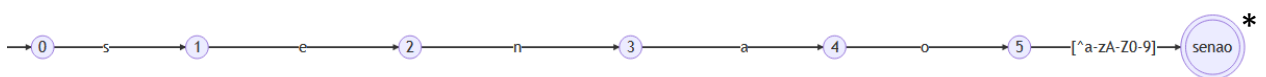
se



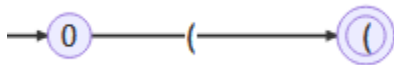
entao



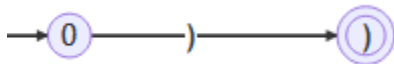
senao



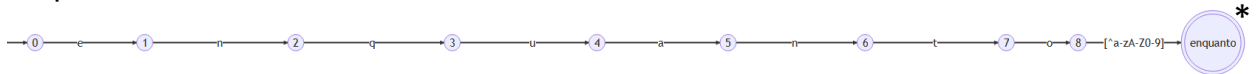
(



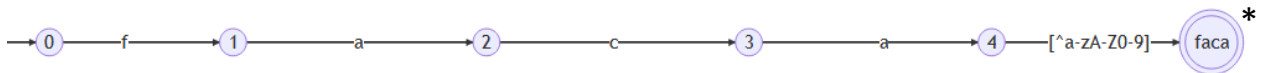
)



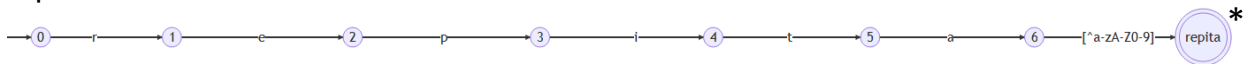
enquanto



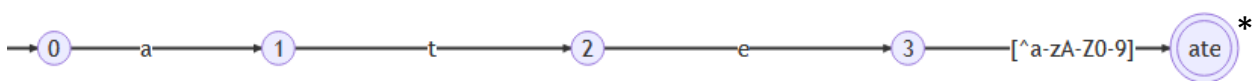
faca



repita



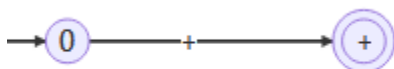
ate

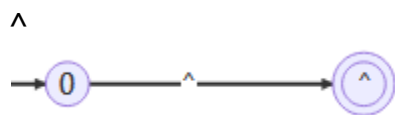
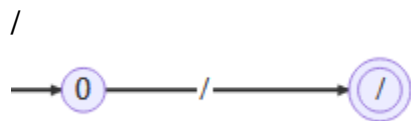
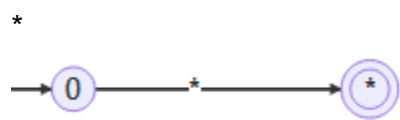


=

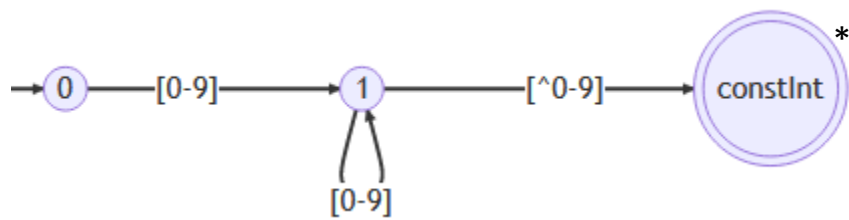


+

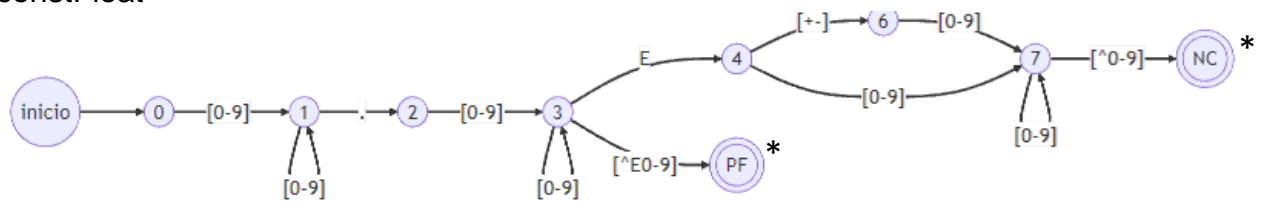




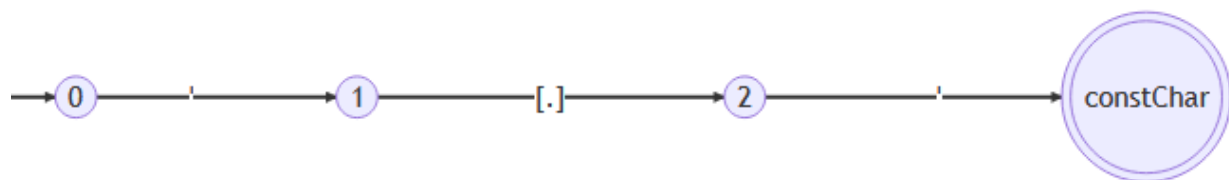
constInt



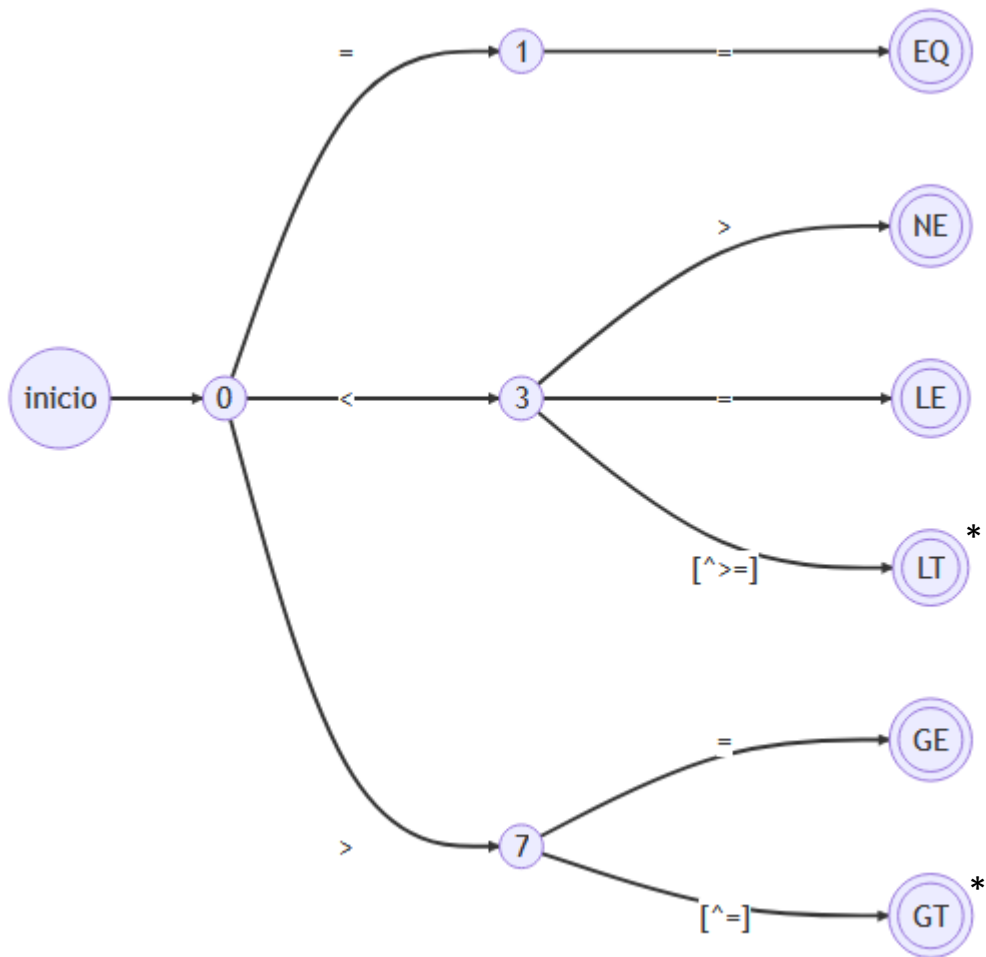
constFloat



constChar



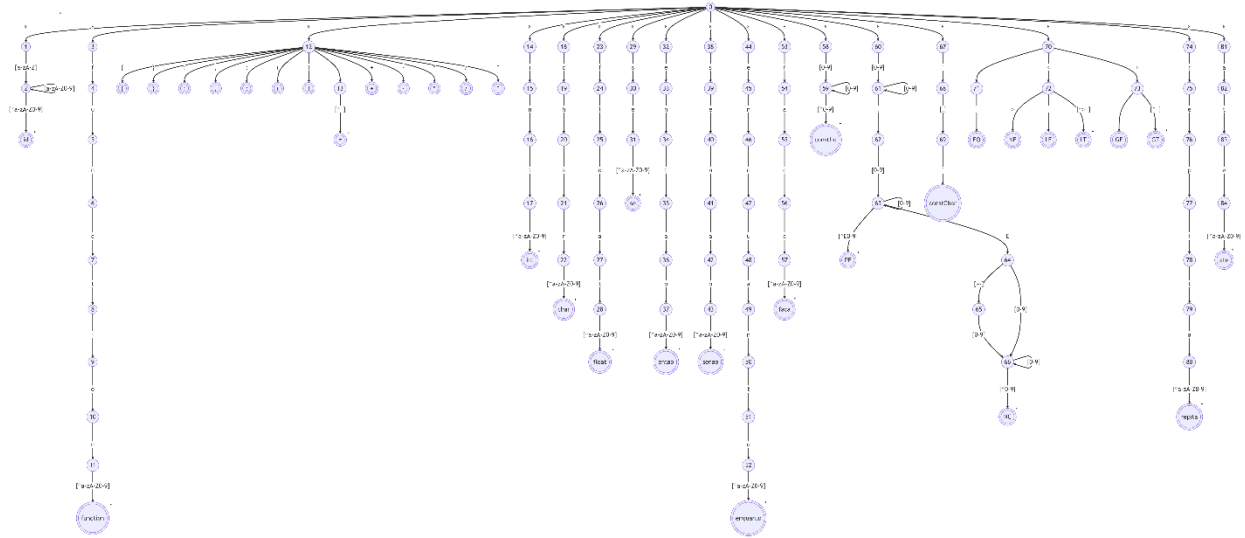
relop



Unificá-los em um diagrama não determinístico

Diagrama não determinístico

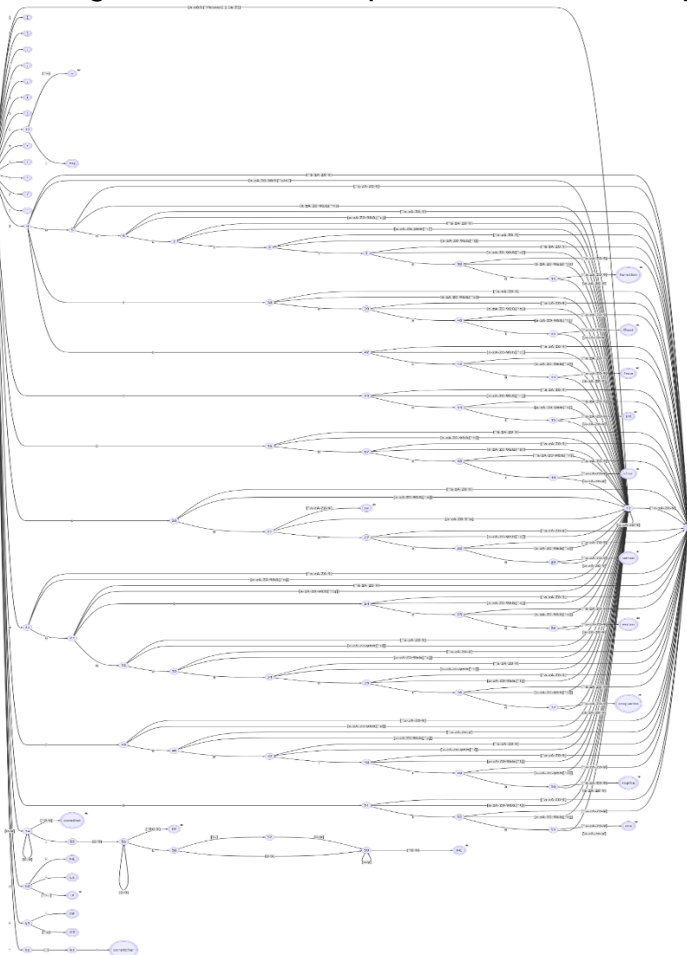
(A imagem com melhor qualidade está no arquivo “etapa 2 fase 2.png”)



Convertê-lo em um diagrama de transição determinístico

Diagrama determinístico

(A imagem com melhor qualidade está no arquivo “etapa 2 fase 3.png”)



Análise Léxica (implementação)

Vide arquivos "lexico.py"

3ª Etapa do Projeto

Análise sintática (especificação)

Fazer os ajustes necessários para que a GLC da linguagem seja LL(1)

Gramática livre de contexto (GLC) (LL(1))

$S = \langle S \rangle$

$T = \{ID, FUNCTION, \{ \{, \} \}, :, INT, CHAR, FLOAT, \{ \{, \} \}, \{ \{, \} \}, SE, ENTAO, SENAO, (,), ENQUANTO, FACA, REPITA, ATE, =, <, >, < >, ==, < =, > =, +, -, *, /, ^, CONSTINT, CONSTFLOAT, CONSTCHAR, RELOP \}$

$N = \{ \langle S \rangle, \langle bloco \rangle, \langle declaracaoVariavel \rangle, \langle sequenciaComandos \rangle, \langle ComandoAtribuicao \rangle, \langle ComandoRepeticao \rangle, \langle ComandoCondicional \rangle, \langle lista_ids \rangle, \langle tipo \rangle, \langle cond \rangle, \langle expressao \rangle, \langle ComandoCondicional' \rangle, \langle lista_ids' \rangle, \langle termo \rangle, \langle expressao' \rangle, \langle fator \rangle, \langle termo' \rangle \}$

P :

$S \rightarrow FUNCTION ID() bloco$

$sequenciaComandos \rightarrow ComandoAtribuicao \mid ComandoCondicional \mid ComandoRepeticao$

$bloco \rightarrow \{ declaracaoVariavel sequenciaComandos \}$

$declaracaoVariavel \rightarrow tipo: lista_ids; \mid \epsilon$

$lista_ids \rightarrow id lista_ids'$

$lista_ids' \rightarrow , lista_ids \mid \epsilon$

$tipo \rightarrow CHAR \mid FLOAT \mid INT$

$ComandoCondicional \rightarrow SE (cond) ENTAO ComandoBloco ComandoCondicional'$

$ComandoCondicional' \rightarrow SENAO ComandoBloco \mid \epsilon$

$ComandoRepeticao \rightarrow ENQUANTO (cond) FACA ComandoBloco \mid REPITA ComandoBloco ATE (cond)$

$ComandoBloco \rightarrow bloco \mid sequenciaComandos$

$ComandoAtribuicao \rightarrow ID = ComandoAtribuicao'$

$ComandoAtribuicao' \rightarrow expressao \mid CONSTCHAR;$

$cond \rightarrow expressao RELOP expressao$

$\text{expressao} \rightarrow \text{termo expressao}'$

$\text{expressao}' \rightarrow + \text{termo expressao}' \mid - \text{termo expressao}' \mid \varepsilon$

$\text{termo} \rightarrow \text{fator termo}'$

$\text{termo}' \rightarrow * \text{fator termo}' \mid / \text{fator termo}' \mid \varepsilon$

$\text{fator} \rightarrow \text{ID} \mid \text{CONSTINT} \mid \text{CONSTFLOAT} \mid (\text{expressao})$

Calcular FIRST e FOLLOW para os símbolos da gramática

Firsts:

$\text{FIRST}(\text{fator}) = \{ \text{ID}, \text{CONSTINT}, \text{CONSTFLOAT}, (\}$

$\text{FIRST}(\text{termo}) = \{ \text{ID}, \text{CONSTINT}, \text{CONSTFLOAT}, (\}$

$\text{FIRST}(\text{termo}') = \{ *, /, \varepsilon \}$

$\text{FIRST}(S) = \{ \text{FUNCTION} \}$

$\text{FIRST}(\text{sequenciaComandos}) = \{ \text{ID}, \text{SE}, \text{ENQUANTO}, \text{REPITA} \}$

$\text{FIRST}(\text{bloco}) = \{ \{ ' \}$

$\text{FIRST}(\text{declaracaoVariavel}) = \{ \text{CHAR}, \text{FLOAT}, \text{int}, \varepsilon \}$

$\text{FIRST}(\text{lista_ids}) = \{ \text{ID} \}$

$\text{FIRST}(\text{lista_ids}') = \{ , \}$

$\text{FIRST}(\text{tipo}) = \{ \text{CHAR}, \text{FLOAT}, \text{int} \}$

$\text{FIRST}(\text{ComandoCondicional}) = \{ \text{SE} \}$

$\text{FIRST}(\text{ComandoCondicional}') = \{ \text{SENAO}, \varepsilon \}$

$\text{FIRST}(\text{comandoRepeticao}) = \{ \text{ENQUANTO}, \text{REPITA} \}$

$\text{FIRST}(\text{comandoAtribuicao}) = \{ \text{ID} \}$

$\text{FIRST}(\text{ComandoAtribuicao}') = \{ \text{CONSCAR}, \text{ID}, \text{CONSTINT}, \text{CONSTFLOAT}, (\}$

$\text{FIRST}(\text{cond}) = \{ \text{ID}, \text{CONSTINT}, \text{CONSTFLOAT}, (\}$

$\text{FIRST}(\text{expressao}) = \{ \text{ID}, \text{CONSTINT}, \text{CONSTFLOAT}, (\}$

$\text{FIRST}(\text{expressao}') = \{ +, -, \varepsilon \}$

$\text{FIRST}(\text{ID}) = \{ \text{ID} \}$

$\text{FIRST}(\text{FUNCTION}) = \{ \text{FUNCTION} \}$

$\text{FIRST}(\text{'}) = \{ \text{'} \}$

$\text{FIRST}(\{ \}) = \{ \text{'} \}$

$\text{FIRST}(:) = \{ : \}$

$\text{FIRST}(\text{INT}) = \{ \text{INT} \}$

$\text{FIRST}(\text{CHAR}) = \{ \text{CHAR} \}$

$\text{FIRST}(\text{FLOAT}) = \{ \text{FLOAT} \}$

$\text{FIRST}(,) = \{ , \}$

$\text{FIRST} (;) = \{ ; \}$

$\text{FIRST}(\text{SE}) = \{ \text{SE} \}$

$\text{FIRST}(\text{ENTAO}) = \{ \text{ENTAO} \}$

$\text{FIRST}(\text{SENAO}) = \{ \text{SENAO} \}$

$\text{FIRST}(()) = \{ (\}$

$\text{FIRST}() = \{) \}$

$\text{FIRST}(\text{ENQUANTO}) = \{ \text{ENQUANTO} \}$

$\text{FIRST}(\text{REPITA}) = \{ \text{REPITA} \}$

$\text{FIRST}(\text{ATE}) = \{ \text{ATE} \}$

$\text{FIRST}(=) = \{ = \}$

$\text{FIRST}(<) = \{ < \}$

$\text{FIRST}(>) = \{ > \}$

$\text{FIRST}(<>) = \{ <> \}$

$\text{FIRST}(==) = \{ == \}$

$\text{FIRST}(<=) = \{ <= \}$

$\text{FIRST}(>=) = \{ >= \}$

$\text{FIRST}(+) = \{ + \}$

$\text{FIRST}(-) = \{ - \}$

$\text{FIRST}(\ast) = \{ \ast \}$

$\text{FIRST}(/) = \{ / \}$

$\text{FIRST}(\wedge) = \{ \wedge \}$

$\text{FIRST}(\text{CONSTINT}) = \{ \text{CONSTINT} \}$

$\text{FIRST}(\text{CONSTFLOAT}) = \{ \text{CONSTFLOAT} \}$

$\text{FIRST}(\text{CONSTCHAR}) = \{ \text{CONSTCHAR} \}$

$\text{FIRST}(\text{RELOP}) = \{ \text{RELOP} \}$

$\text{FIRST}(\text{ComandoBloco}) = \{ \{, \text{ID}, \text{SE}, \text{ENQUANTO}, \text{REPITA} \}$

Follow:

$\text{FOLLOW}(\$) = \{ \$ \}$

$\text{FOLLOW}(\text{sequenciaComandos}) = \{ '}' \}$

$\text{FOLLOW}(\text{bloco}) = \text{FIRST}(\text{ComandoCondicional}) - \{ \epsilon \} + \text{FOLLOW}(\text{ComandoBloco})$

$\text{FOLLOW}(\text{ComandoCondicional}) + \text{FOLLOW}(\text{ComandoRepeticao}) = \{ \$, \text{ATE}, '}' \}$

$\text{FOLLOW}(\text{declaracaoVariavel}) = \text{FIRST}(\text{ComandoAtribuicao}) +$

$\text{FIRST}(\text{ComandoCondicional}) + \text{FIRST}(\text{ComandoRepeticao}) = \{ \text{ID}, \text{SE}, \text{ENQUANTO}, \text{REPITA} \}$

$\text{FOLLOW}(\text{lista_ids}) = \text{FIRST}(\text{sequenciaComandos}) = \{ ; \}$

$\text{FOLLOW}(\text{lista_ids}') = \text{FOLLOW}(\text{lista_ids}) = \{ ; \}$

$\text{FOLLOW}(\text{tipo}) = \{ : \}$

$\text{FOLLOW}(\text{comandoCondicional}) = \text{FOLLOW}(\text{sequenciaComandos}) = \{ '}' \}$

$\text{FOLLOW}(\text{comandoRepeticao}) = \text{FOLLOW}(\text{sequenciaComandos}) = \{ '}' \}$

$\text{FOLLOW}(\text{comandoAtribuicao}) = \text{FOLLOW}(\text{sequenciaComandos}) = \{ '}' \}$

$\text{FOLLOW}(\text{cond}) = \{ '}' \}$

$\text{FOLLOW}(\text{expressao}) = \{ '}', \text{RELOP}, '}' \}$

$\text{FOLLOW}(\text{fator}) = \text{FIRST}(\text{termo}') = \{ * \}$

$\text{FOLLOW}(\text{termo}) = \text{FIRST}(\text{expressao}') = \{ +, - \}$

$\text{FOLLOW}(\text{termo}') = \text{FIRST}(\text{expressao}') - \{ \epsilon \} + \text{FOLLOW}(\text{termo}) = \{ +, - \}$

$\text{FOLLOW}(\text{comandoCondicional}') = \text{FOLLOW}(\text{comandoCondicional}) = \{ '}' \}$

$\text{FOLLOW}(\text{expressao}') = \{ + \}$

$\text{FOLLOW}(\text{ComandoAtribuicao}') = \text{FOLLOW}(\text{comandoAtribuicao}) =$

$\text{FOLLOW}(\text{sequenciaComandos}) = \{ '}' \}$

$\text{FIRST}(\text{ComandoBloco}) = \{ \{, \text{ID}, \text{SE}, \text{ENQUANTO}, \text{REPITA} \}$

Construção da Tabela de Produções

(A tabela original está disponível no arquivo “./compiladores-main/Codigo/producoes.xlsx”)

1	FUNCTION ID () bloco
2	ComandoAtribuicao
3	ComancoCondicional
4	ComandoRepeticao
5	{ declaracaoVariavel sequenciaComandos }
6	tipo : lista_ids ;
7	ϵ
8	ID lista_ids'
9	, lista_ids
10	ϵ
11	CHAR
12	FLOAT
13	INT
14	SE (cond) ENTAO ComandoBloco ComandoCondicional
15	SENAO ComandoBloco
16	ϵ
17	ENQUANTO (cond) FACA ComandoBloco
18	REPITA ComandoBloco ATE (cond)
19	ID = ComandoAtribuicao'
20	expressao
21	CONSTCHAR ;
22	expressao RELOP expressao
23	termo expressao'
24	+ termo expressao'
25	- termo expressao'
26	ϵ
27	fator termo'
28	* fator termo'
29	/ fator termo'
30	ϵ
31	ID
32	CONSTFLOAT
33	(expressao)
34	CONSTINT
35	bloco
36	sequenciaComandos
37	\$

Construção da tabela preditiva

(A tabela original está disponível no arquivo “./compiladores-main/Codigo/preditivo.xlsx”)

NT	{	}	ID	CHAR	FLOAT	INT	SE	SENAO	()	FACA	REPITA	ATE	CONST	CHAR	RELOP	+	-	*	/	CONSTINT	CONST	FLOAT	FUNCTION	ENQUANTO	:	;	,	\$
S	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	37
sequenciaComandos	-1	5	2	-1	-1	-1	3	-1	-1	-1	-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	4	-1	-1	-1	-1
ComandoAtribuicao	-1	-1	19	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
ComandoCondicional	-1	-1	-1	-1	-1	-1	14	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
ComandoRepeticao	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	18	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
bloco	5	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	18	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	37
declaracaoVariavel	-1	-1	5	6	6	6	14	-1	-1	-1	-1	18	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	17	-1	-1	-1	-1
fator	-1	-1	31	-1	-1	-1	-1	-1	33	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	34	32	-1	-1	-1	-1	-1	-1
tipo	-1	-1	-1	11	12	13	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	2	-1	-1	-1
lista_ids	-1	-1	8	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
lista_ids'	-1	-1	31	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	31	8	-1	-1
cond	-1	-1	31	-1	-1	-1	-1	-1	33	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	34	32	-1	-1	-1	-1	-1	-1	-1
ComandoCondicional'	-1	-1	-1	-1	-1	-1	-1	15	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
expressao	-1	-1	31	-1	-1	-1	-1	-1	33	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	34	32	-1	-1	-1	-1	-1	-1	-1
expressao'	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	24	25	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
termo	-1	-1	31	-1	-1	-1	-1	-1	33	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	34	32	-1	-1	-1	-1	-1	-1	-1
termo'	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	28	29	-1	-1	-1	-1	-1	-1	-1	-1	-1
ComandoAtribuicao'	-1	-1	31	-1	-1	-1	-1	-1	33	-1	-1	-1	-1	-1	-1	21	-1	-1	-1	-1	-1	34	32	-1	-1	-1	-1	-1	-1
ComandoBloco	35	-1	36	-1	-1	-1	36	-1	33	-1	-1	36	-1	-1	-1	-1	-1	-1	-1	-1	34	32	-1	-1	36	-1	-1	-1	-1

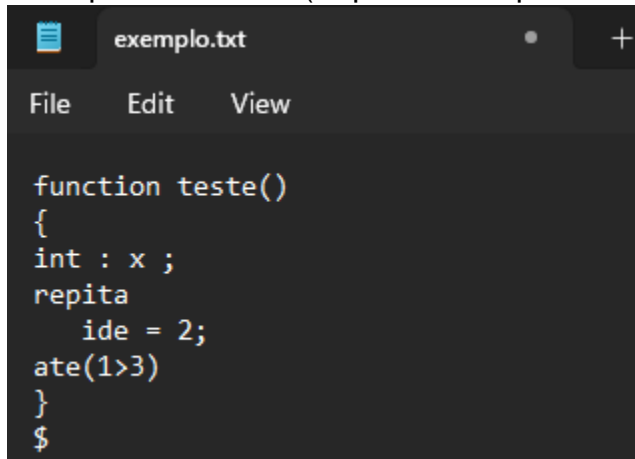
Análise Sintática (implementação)

Vide arquivo “./compiladores-main/Codigo/AnalizadorSintatico.py”

4ª Etapa do Projeto

Tradução dirigida por sintaxe

Exemplo de entrada (arquivo “./compiladores-main/Codigo/exemplo.txt”)



```
exemplo.txt
File Edit View

function teste()
{
  int : x ;
  repita
    ide = 2;
  ate(1>3)
}
$
```

Resultado

```
[tipo -> : -> lista_ids -> sequenciaComandos -> } -> None] tipo
tipo INT
aqui
8 6
13
[INT -> : -> lista_ids -> sequenciaComandos -> } -> None] INT
b' '
b': '
[lexical] Token TokenEnum.DPONTOS, TokenEnum.ATE, w: :
[: -> lista_ids -> sequenciaComandos -> } -> None] :
b' '
b'x'
b' '
[lexical] Token TokenEnum.ID, , w:
[lista_ids -> sequenciaComandos -> } -> None] lista_ids
lista_ids ID
```