TRABALHO FINAL DE COMPILADORES

ALUNOS: ESTER TOJA E FRANCIS LUIS

Arquivos de códigos-fonte:

- *erros.py=* Arquivo que contém as mensagens de erro.
- *gramatica.py=* Arquivo contém a gramática.
- *lexico.py=* Arquivo que contém o analisador léxico, que tem como função validar caracter por caracter e traduzir em uma sequência de tokens.
- *main.py=* Arquivo que contém a ordem de execução dos analisador do compilador.
- *teste.txt=* Arquivo no qual onde deve ser colocada as sentenças para realizar as validações.

Fases de Desenvolvimentos e Requisitos:

Etapa 1 - (Finalizado):

- Devem reconhecer os tipos: char – int – float;
- O char pode ser um caractere ou uma cadeia;
- Identificadores: de acordo com as regras da Linguagem C (iniciam por letras ou _, depois do segundo caractere pode ser número, letra ou _ e o único caractere especial é o_);
- Podes ser declarados individualmente ou por uma lista (sendo lista separados por vírgulas);
- A finalização de cada declaração será por ponto-e-vírgula (;);

Etapa 2 - (Finalizado):

- Comandos de seleção: if e switch-case;
- Sintaxe: if(condição){<comandos>} else {<comandos>};
- **Comando if-else;**
  - O comando else é opcional (assim, como na linguagem C);
  - Os comandos serão apenas matemáticos, ou seja, operações matemáticas simples: soma, subtração, multiplicação e divisão (operadores matemáticos), que serão atribuídos, por meio dos operadores de atribuição a uma variável;
- Sintaxe: switch(variável){case 1:{<comandos> break;} case 2: case 3: {<comandos> break;} default: {<comandos>}};
- **Comando switch-case;**
  - Mesmas informações do comando if na questão dos comandos;
  - Os cases podem ser unitários ou até a quantidade de três;
  - O comando default é opcional (assim, como na linguagem;

Etapa 3 (Finalizado):

- Comandos de seleção: while e for;
- Sintaxe: while(condição){<comandos>};
- **Comando while;**
  - Os comandos serão apenas matemáticos, ou seja, operações matemáticas simples: soma, subtração, multiplicação e divisão (operadores matemáticos), que serão atribuídos, por meio dos operadores de atribuição a uma variável;
- Sintaxe: for(int variável; condição; incremento/decremento){<comandos>};
- **Comando for;**

- Os comandos serão apenas matemáticos, ou seja, operações matemáticas simples: soma, subtração, multiplicação e divisão (operadores matemáticos), que serão atribuídos, por meio dos operadores de atribuição a uma variável;
- A inicialização será por meio de uma atribuição.

Etapa 4 (Finalizado):

Operadores Matemáticos

Operador	Exemplo	Comentário
+	x + y	Soma x e y
-	x – y	Subtrai y de x
*	x * y	Multiplica x e y
/	x / y	Divide x por y

Operadores de Atribuição

Operador	Exemplo	Comentário
=	x = y	Atribui o valor de y a x
+=	x +=y	Equivale a x = x + y
-=	x -= y	Equivale a x = x – y
*=	x *= y	Equivale a x = x * y
/=	x/=y	Equivale a x = x / y

Operadores Relacionais

Operador	Exemplo	Comentário
==	x == y	O conteúdo de x é igual ao de y
!=	x != y	O conteúdo de x é diferente de y
<=	x <= y	O conteúdo de x é menor ou igual ao de y
>=	x >= y	O conteúdo de x é maior ou igual ao de y
<	x < y	O conteúdo de x é menor que o de y
>	x > y	O conteúdo de x é maior que o de y

- Apresentação e explicações do processo de construção do Trabalho, com horário marcado e tempo de apresentação de 15 minutos.

Gramática:

<programa>::= <declaracao_variavel> | <declaracao> | <comentario>

<declaracao>::=<for_declaracao> | <while_delcaracao> | <if_declaracao> | <switch_declaracao>

<comentario>::= //<programa> | /*<programa>*/

<variavel_Declaracao>::= <tipo_primitivo><sequencia_variaveis>;

<tipo_primitivo>::= FLOAT | INT | CHAR

<sequencia_variaveis>::= <variveis_caracteristicas>,<sequencia_variaveis> | <variaveis_caracteristicas>

<variaveis_caracteristicas>::= NAME [NUMBER] | NAME = <expressao> | NAME = NUMBER.NUMBER | NAME = NUMBER | NAME | <espressao>

<if_declaracao>::= IF (<expressao>){<bloco>} ELSE {<bloco>} | IF (<expressao>){<bloco>}

<while_declaracao>::= WHILE(<expressao>){<bloco>}

<for_declaracao>::= FOR(<tipo_primitivo> NAME = <expressao>; <expressao>; <expressao>) {<bloco>}

<switch_declaracao>::= SWITCH (NAME {<corpo_case> <corpo_case> <corpo_case> <default>}|SWITCH (NAME {<corpo_case> <corpo_case> <default>}| SWITCH (NAME {<corpo_case> <default>}

<corpo_case>::= CASE <comando_case>

<comando_case>::= <expressao>:{<bloco> BREAK;}

<default>::= DEFAULT: {<bloco>} | ε

<credemento>::= NAME-- | NAME++ | NAME = <expressao> | <expressao>

<expressao>::= <expressao_logico> | <expressao_operadores> | <expressao_atribuicao> | <literal> | <variavel>

<expressao_logico>::= <expressao> > <expressao> | <expressao> < <expressao> | <expressao> >= <expressao> | <expressao> <= <expressao> | <expressao> = <expressao> | <expressao> != <expressao> | <expressao> && <expressao> | <expressao> || <expressao>

<expressao_operadores>::= <expressao> + <expressao>|<expressao> - <expressao>|<expressao> * <expressao>|<expressao>/<expressao>

<expressao_atribuicao>::= <expressao> += <expressao>|<expressao> -= <expressao>|<expressao> *= <expressao>|<expressao> /= <expressao>

<variavel>::= <variavel>

<literal>::= NORMALSTRING | NUMBER

<variavel>::= NAME

<bloco>::= <lista_declaracao>

<lista_declaracao> ::= <variavel_Declaracao> <lista_declaracao>| <sequencia_variaveis>;<lista_declaracao> | ε

Sentenças e Derivações:

Declaração de Variável:

Sentença: int a = 10;

Gramática:

```
<programa>
<declaracao_variavel>
<tipo_primitivo><sequencia_variaveis>;
INT <variaveis_caracteristicas>;
INT NAME = NUMBER;
```

Derivação pelo código:

```
program
variable_Declaration
type  sequence_variable SEMICOLON
INT   variable_characteristics SEMICOLON
INT   NAME ASSIGN NUMBER      SEMICOLON
```

Declaração de Variável Array :

Sentença: float a[3];

Gramática:

```
<programa>
<declaracao_variavel>
<tipo_primitivo><sequencia_variaveis>;
FLOAT <variaveis_caracteristicas>;
FLOAT NAME [NUMBER];
```

Derivação pelo código:

```
program
variable_Declaration
type  sequence_variable SEMICOLON
FLOAT variable_characteristics SEMICOLON
FLOAT NAME LCOLC NUMBER RCOLC  SEMICOLON
```

Declaração de Variável :

Sentença: char a[3], b=3, c ;

Gramática:

```
<programa>
<declaracao_variavel>
<tipo_primitivo><sequencia_variaveis>;
CHAR <variveis_caracteristicas>,<sequencia_variaveis>;
CHAR NAME [NUMBER], <variveis_caracteristicas>,<sequencia_variaveis>;
CHAR NAME [NUMBER], <variveis_caracteristicas>,<sequencia_variaveis>;
CHAR NAME [NUMBER], NAME = NUMBER, <variveis_caracteristicas>;
CHAR NAME [NUMBER], NAME = NUMBER, NAME ;
```

Derivação pelo código:

```
program
variable_Declaration
type sequence_variable SEMICOLON
CHAR variable_characteristics COMMA sequence_variable SEMICOLON
CHAR NAME LCOLC NUMBER RCOLC COMMA variable_characteristics COMMA
sequence_variable SEMICOLON
```

CHAR NAME LCOLC NUMBER RCOLC COMMA NAME ASSIGN NUMBER COMMA variable_characteristics SEMICOLON
CHAR NAME LCOLC NUMBER RCOLC COMMA NAME ASSIGN NUMBER COMMA NAME SEMICOLON

Condicional IF ;

Sentença:

```
if (cont > -5) {
    x = a + b;
    x = a * c;
    a = 5 / divisor;
}
```

Gramática*:*

<programa>
<declaracao>
<if_declaracao>
IF(<expressao>){<bloco>}
IF(<expressao_logico>){ <lista_declaracao>}
IF(<expressao> > <expressao>){<sequencia_variaveis>;<lista_declaracao>}
IF(<variavel> > <variavel>){ NAME = <expressao>; <sequencia_variaveis>; <lista_declaracao>}
IF(NAME>NAME){NAME = <expressao_operadores>; NAME = <expressao>; <sequencia_variaveis>; <lista_declaracao>}
IF(NAME>NAME){NAME = <expressao>+<expressao>; NAME = <expressao_operadores>; NAME = <expressao>;}
IF(NAME>NAME){NAME = <variavel>+<variavel>; NAME = <expressao><expressao>; NAME= <expressao_operadores>;}*
IF(NAME>NAME){NAME = NAME+NAME; NAME = <variavel><variavel>; NAME= <expressao>/<expressao>;}*
*IF(NAME>NAME){NAME = NAME+NAME; NAME = NAME*NAME; NAME= <variavel>/ <variavel>;}*
*IF (NAME>NAME){NAME = NAME+NAME; NAME = NAME*NAME; NAME= NAME/NAME;}*

Derivação pelo código:

program

statement

if_statement

IF LPAREN expression RPAREN LBRACE block RBRACE

IF LPAREN expression BIGGER expression RPAREN LBRACE list_Declarations RBRACE

IF LPAREN variable BIGGER variable RPAREN LBRACE sequence_variable SEMICOLON list_Declarations RBRACE

IF LPAREN NAME BIGGER NAME RPAREN LBRACE variable_characteristics SEMICOLON sequence_variable SEMICOLON list_Declarations RBRACE

IF LPAREN NAME BIGGER NAME RPAREN LBRACE NAME ASSIGN expression SEMICOLON variable_characteristics SEMICOLON list_Declarations sequence_variable SEMICOLON RBRACE

IF LPAREN NAME BIGGER NAME RPAREN LBRACE NAME ASSIGN expression PLUS expression SEMICOLON NAME ASSIGN expression SEMICOLON variable_characteristics SEMICOLON RBRACE

IF LPAREN NAME BIGGER NAME RPAREN LBRACE NAME ASSIGN variablePLUS variable SEMICOLON NAME ASSIGN expression TIMES expression SEMICOLON NAME ASSIGN expression SEMICOLON RBRACE

IF LPAREN NAME BIGGER NAME RPAREN LBRACE NAME ASSIGN NAME PLUS NAME SEMICOLON NAME ASSIGN variable TIMES variable SEMICOLON NAME ASSIGN expression DIVIDE expression SEMICOLON RBRACE

IF LPAREN NAME BIGGER NAME RPAREN LBRACE NAME ASSIGN NAME PLUS NAME SEMICOLON NAME ASSIGN NAME TIMES NAME SEMICOLON NAME ASSIGN variable DIVIDE variable SEMICOLON RBRACE

IF LPAREN NAME BIGGER NAME RPAREN LBRACE NAME ASSIGN NAME PLUS NAME SEMICOLON NAME ASSIGN NAME TIMES NAME SEMICOLON NAME ASSIGN NAME DIVIDE NAME SEMICOLON RBRACE

Condicional IF ELSE :

Sentença: if (cont < 5.345) {
x = a + b;
x *= c;
a = 43.3 / divisor;
} else {
y = valor;
x = a + 324.234;
y = 32423.23 * 5;
}

Gramática*:*

<programa>

<declaracao>

<if_declaracao>

IF (<expressao>){<bloco>} ELSE {<bloco>}
IF(<expressao_logico>){ <lista_declaracao>}ELSE {<lista_declaracao>}

IF(<expressao><<expressao>){<sequencia_variaveis>;<lista_declaracao>}
ELSE {<sequencia_variaveis>;<lista_declaracao>}

IF(<variavel><<literal>){<variaveis_caracteristicas>;
<sequencia_variaveis> ; <lista_declaracao>} ELSE {
<variaveis_caracteristicas> ; <sequencia_variaveis> ;
<lista_declaracao>}

IF(NAME < NUMBER){NAME = <expressao> ; <variaveis_caracteristicas> ;
<sequencia_variaveis> ; <lista_declaracao>}ELSE {NAME = <expressao>;
<variaveis_caracteristicas> ;
<sequencia_variaveis>;<lista_declaracao> }

IF(NAME < NUMBER){NAME = <expressao> ; <variaveis_caracteristicas>;
<sequencia_variaveis> ; <lista_declaracao>}ELSE {<expressao> ;

<variaveis_caracteristicas> ; <sequencia_variaveis> ; <lista_declaracao>}

IF(NAME < NUMBER){NAME = <expressao_operadores> ; <expressao> ; <variaveis_caracteristicas> ; }ELSE { <expressao_logico> ; <expressao> ; <variaveis_caracteristicas> ; }

IF(NAME < NUMBER){NAME = <expressao> + <expressao> ; <expressao_atribuicao> ; <expressao> ; }ELSE { <expressao>=<expressao>; <expressao_logico> ; <expressao> ; }

*IF(NAME < NUMBER){ NAME = <variavel> + <variavel>; <expressao> *= <expressao> ; <expressao_logico>; }ELSE { NAME = NAME ; <expressao> = <expressao>; <expressao_logico>; }*

*IF(NAME < NUMBER){ NAME = NAME + NAME ; <variavel> *= <variavel> ; <expressao> = <expressao> ; } ELSE { NAME = NAME ; <expressao> = <expressao_operadores> ; <expressao> = <expressao> ; }*

*IF(NAME < NUMBER){ NAME = NAME + NAME ; NAME *= NAME ; <variavel> = <expressao_operadores> ; }ELSE { NAME = NAME ; NAME = <expressao> + <expressao> ; <variavel> = <expressao_operadores> ; }*

*IF(NAME < NUMBER){ NAME = NAME + NAME ; NAME *= NAME ; NAME = <expressao>/<expressao> ; }ELSE { NAME = NAME; NAME = <variavel> + <literal>; NAME = <expressao> * <expressao>;}*

*IF(NAME < NUMBER){ NAME = NAME + NAME; NAME *= NAME; NAME = <literal>/<variavel>; }ELSE { NAME = NAME; NAME = NAME + NUMBER; NAME = <literal> * <literal>;}*

*IF(NAME < NUMBER){NAME = NAME + NAME;NAME *= NAME;NAME = NUMBER/NAME;}ELSE {NAME = NAME; NAME = NAME + NUMBER; NAME = NUMBER * NUMBER;}*

Derivação pelo código:

program

statement

if_statement

IF LPAREN expression RPAREN LBRACE block RBRACE ELSE LBRACE block RBRACE

IF LPAREN variable BIGGER literal RPAREN LBRACE list_Declarations RBRACE ELSE LBRACE list_Declarations RBRACE

IF LPAREN NAME BIGGER NUMBER RPAREN LBRACE sequence_variable SEMICOLON sequence_variable SEMICOLON list_Declarations RBRACE ELSE LBRACE sequence_variable SEMICOLON sequence_variable SEMICOLON list_Declarations RBRACE

IF LPAREN NAME BIGGER NUMBER RPAREN LBRACE variable_characteristics SEMICOLON sequence_variable SEMICOLON sequence_variable SEMICOLON list_Declarations RBRACE ELSE LBRACE variable_characteristics

SEMICOLON sequence_variable SEMICOLON sequence_variable SEMICOLON list_Declarations RBRACE

IF LPAREN NAME BIGGER NUMBER RPAREN LBRACE NAME = <expressao> SEMICOLON variable_characteristics SEMICOLON sequence_variable SEMICOLON RBRACE ELSE LBRACE <espressao> SEMICOLON variable_characteristics SEMICOLON sequence_variable SEMICOLON RBRACE

IF LPAREN NAME BIGGER NUMBER RPAREN LBRACE NAME = <expressao> SEMICOLON <espressao> SEMICOLON variable_characteristics SEMICOLON RBRACE ELSE LBRACE <espressao> SEMICOLON NAME = <expressao> SEMICOLON variable_characteristics SEMICOLON RBRACE

IF LPAREN NAME BIGGER NUMBER RPAREN LBRACE NAME = <expressao_operadores> SEMICOLON <expressao_atribuicao> SEMICOLON NAME = <expressao> SEMICOLON RBRACE ELSE LBRACE <expressao_logico> SEMICOLON NAME = <expressao_operadores> SEMICOLON NAME = <expressao> SEMICOLON RBRACE

*IF LPAREN NAME BIGGER NUMBER RPAREN LBRACE NAME = <expressao> + <expressao> SEMICOLON <expressao> *= <expressao> SEMICOLON NAME = <expressao_operadores> SEMICOLON RBRACE ELSE LBRACE <expressao> = <expressao> SEMICOLON NAME = <expressao> + <expressao> SEMICOLON NAME = <expressao_operadores> SEMICOLON RBRACE*

*IF LPAREN NAME BIGGER NUMBER RPAREN LBRACE NAME = <variavel> + <variavel> SEMICOLON <variavel> *= <variavel> SEMICOLON NAME = <expressao> / <expressao> SEMICOLON RBRACE ELSE LBRACE <variavel> = <variavel> SEMICOLON NAME = <variavel> + <expressao_operadores> SEMICOLON NAME = <expressao> * <expressao> SEMICOLON RBRACE*

*IF LPAREN NAME BIGGER NUMBER RPAREN LBRACE NAME = NAME + NAME SEMICOLON NAME *= NAME SEMICOLON NAME = <literal> / <variavel> SEMICOLON RBRACE ELSE LBRACE NAME = NAME SEMICOLON NAME = <expressao> + <expressao> SEMICOLON NAME = <literal> * <literal> SEMICOLON RBRACE*

*IF LPAREN NAME BIGGER NUMBER RPAREN LBRACE NAME = NAME + NAME SEMICOLON NAME *= NAME SEMICOLON NAME = NUMBER / NAME SEMICOLON RBRACE ELSE LBRACE NAME = NAME SEMICOLON NAME = <variavel> + <literal> SEMICOLON NAME = <literal> * <literal> SEMICOLON RBRACE*

*IF LPAREN NAME BIGGER NUMBER RPAREN LBRACE NAME = NAME + NAME SEMICOLON NAME *= NAME SEMICOLON NAME = NUMBER / NAME SEMICOLON RBRACE ELSE LBRACE NAME = NAME SEMICOLON NAME = NAME + NUMBER SEMICOLON NAME = NUMBER * NUMBER SEMICOLON RBRACE*

Condicional Switch :

Sentença:

```
switch (a) {
        case 1: {
                b = c + d;
        break; }
```

}

Gramática*:*

<programa>
<declaracao>
<shitch_declaracao>
SWITCH (NAME {<corpo_case> <default>}
SWITCH (NAME {CASE <comando_case> ε}
SWITCH (NAME {CASE <expressao>: {<bloco> BREAK;}}
SWITCH (NAME {CASE <literal>: {<lista_declaracao> BREAK;}}
SWITCH (NAME {CASE NUMBER : {<variavel_Declaracao>
<lista_declaracao> BREAK;}}
SWITCH (NAME {CASE NUMBER : { NAME = <expressao> BREAK;}}
SWITCH (NAME {CASE NUMBER : { NAME = <expressao_operadores> BREAK;}}
SWITCH (NAME {CASE NUMBER : { NAME = <expressao> + <expressao>
BREAK;}}
SWITCH (NAME {CASE NUMBER : { NAME = <variavel> + <variavel>
BREAK;}}
SWITCH (NAME {CASE NUMBER : { NAME = NAME + NAME BREAK;}}

Derivação pelo código:

program

variable_Declaration

switch_statement

SWITCH LPAREN NAME RPAREN LBRACE body_case sintax_default RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE sintax_case empty RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE literal COLON LBRACE
list_Declarations BREAK SEMICOLON RBRACE RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE NUMBER COLON LBRACE
sequence_variable SEMICOLON list_Declarations BREAK SEMICOLON RBRACE
RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE NUMBER COLON LBRACE
sequence_variable SEMICOLON empyt BREAK SEMICOLON RBRACE RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE NUMBER COLON LBRACE
variable_characteristics SEMICOLON empyt BREAK SEMICOLON RBRACE
RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE NUMBER COLON LBRACE NAME ASSIGN
expression SEMICOLON empyt BREAK SEMICOLON RBRACE RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE NUMBER COLON LBRACE NAME ASSIGN
expression PLUS expression SEMICOLON empyt BREAK SEMICOLON RBRACE
RBRACE

Condicional Switch :

Sentença:

```
switch (a) {
        case 1: {
                b = c + d;
```

```
                    break; }
                    case 2: {
                            c = d - b;
                    break; }
                    case 3: {
                             d = c * d;
                    break; }
                    default: {
                    c = a++
                    }
            }
```

Gramática*:*

<programa>
<declaracao>
<shitch_declaracao>
SWITCH (NAME {<corpo_case> <default>}
SWITCH (NAME {CASE <comando_case> ε}
SWITCH (NAME {CASE <expressao>: {<bloco> BREAK;}}
SWITCH (NAME {CASE <literal>: {<lista_declaracao> BREAK;}}
SWITCH (NAME {CASE NUMBER : {<variavel_Declaracao>
<lista_declaracao> BREAK;}}
SWITCH (NAME {CASE NUMBER : { NAME = <expressao> BREAK;}}
SWITCH (NAME {CASE NUMBER : { NAME = <expressao_operadores> BREAK;}}
SWITCH (NAME {CASE NUMBER : { NAME = <expressao> + <expressao>
BREAK;}}
SWITCH (NAME {CASE NUMBER : { NAME = <variavel> + <variavel>
BREAK;}}
SWITCH (NAME {CASE NUMBER : { NAME = NAME + NAME BREAK;}}

Derivação pelo código:

program
variable_Declaration
switch_statement
SWITCH LPAREN NAME RPAREN LBRACE body_case body_case body_case
sintax_default RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE sintax_case CASE sintax_case
CASE sintax_case sintax_default RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE expression COLON LBRACE block
break_statement RBRACE CASE expression COLON LBRACE block
break_statement RBRACE CASE expression COLON LBRACE block
break_statement RBRACE DEFAULT COLON LBRACE block RBRACE RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE literal COLON LBRACE
list_Declarations
BREAK SEMICOLON RBRACE CASE literal COLON LBRACE list_Declarations
BREAK SEMICOLON RBRACE CASE literal COLON LBRACE list_Declarations
BREAK SEMICOLON RBRACE DEFAULT COLON LBRACE list_Declarations RBRACE
RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE NUMBER COLON LBRACE
sequence_variable SEMICOLON list_Declarations BREAK SEMICOLON RBRACE
CASE NUMBER COLON LBRACE sequence_variable SEMICOLON
list_Declarations BREAK SEMICOLON RBRACE CASE NUMBER COLON LBRACE

sequence_variable SEMICOLON list_Declarations BREAK SEMICOLON RBRACE DEFAULT COLON LBRACE sequence_variable SEMICOLON list_Declarations RBRACE RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE NUMBER COLON LBRACE sequence_variable SEMICOLON empty BREAK SEMICOLON RBRACE CASE NUMBER COLON LBRACE sequence_variable SEMICOLON empty BREAK SEMICOLON RBRACE CASE NUMBER COLON LBRACE sequence_variable SEMICOLON empty BREAK SEMICOLON RBRACE DEFAULT COLON LBRACE sequence_variable SEMICOLON empty RBRACE RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE NUMBER COLON LBRACE variable_characteristics SEMICOLON empty BREAK SEMICOLON RBRACE CASE NUMBER COLON LBRACE variable_characteristics SEMICOLON empty BREAK SEMICOLON RBRACE CASE NUMBER COLON LBRACE variable_characteristics SEMICOLON empty BREAK SEMICOLON RBRACE DEFAULT COLON LBRACE variable_characteristicsSEMICOLON empty RBRACE RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE NUMBER COLON LBRACE NAME ASSIGN expression SEMICOLON BREAK SEMICOLON RBRACE CASE NUMBER COLON LBRACE NAME ASSIGN expression SEMICOLON BREAK SEMICOLON RBRACE CASE NUMBER COLON LBRACE NAME ASSIGN expression SEMICOLON BREAK SEMICOLON RBRACE DEFAULT COLON LBRACE NAME ASSIGN expression SEMICOLON RBRACE RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE NUMBER COLON LBRACE NAME ASSIGN expression PLUS expression SEMICOLON BREAK SEMICOLON RBRACE CASE NUMBER COLON LBRACE NAME ASSIGN expression MINUS expression SEMICOLON BREAK SEMICOLON RBRACE CASE NUMBER COLON LBRACE NAME ASSIGN expression TIMES expression SEMICOLON BREAK SEMICOLON RBRACE DEFAULT COLON LBRACE NAME ASSIGN variable SUMEQUALS expression SEMICOLON RBRACE RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE NUMBER COLON LBRACE NAME ASSIGN variable PLUS variable SEMICOLON BREAK SEMICOLON RBRACE CASE NUMBER COLON LBRACE NAME ASSIGN variable MINUS variable SEMICOLON BREAK SEMICOLON RBRACE CASE NUMBER COLON LBRACE NAME ASSIGN variable TIMES variable SEMICOLON BREAK SEMICOLON RBRACE DEFAULT COLON LBRACE NAME ASSIGN variable SUMEQUALS variable SEMICOLON RBRACE RBRACE

SWITCH LPAREN NAME RPAREN LBRACE CASE NUMBER COLON LBRACE NAME ASSIGN NAME PLUS NAME SEMICOLON BREAK SEMICOLON RBRACE CASE NUMBER COLON LBRACE NAME ASSIGN NAME MINUS NAME SEMICOLON BREAK SEMICOLON RBRACE CASE NUMBER COLON LBRACE NAME ASSIGN NAME TIMES NAME SEMICOLON BREAK SEMICOLON RBRACE DEFAULT COLON LBRACE NAME ASSIGN NAME SUMEQUALS NAME SEMICOLON RBRACE RBRACE

Repetição WHILE ;

Sentença:

```
while (a <= b) {
   x = y * 5;
}
```

Gramática*:*

<programa>
<declaracao>

<while_delcaracao>
WHILE(<expressao>){<bloco>}
WHILE(<expressao_logico>){<lista_declaracao>}
WHILE(<expressao> <= <expressao>){<sequencia_variaveis>; ε }
WHILE(<variavel> <= <variavel>){<variaveis_caracteristicas>; ε }
WHILE(NAME <= NAME){ NAME = <expressao>; ε }
WHILE(NAME <= NAME){ NAME = <expressao_operadores>; ε }
*WHILE(NAME <= NAME){ NAME =<expressao> * <expressao>; ε }*
*WHILE(NAME <= NAME){ NAME =<variavel> * <literal>; ε }*
*WHILE(NAME <= NAME){ NAME = NAME * NUMBER; ε }*

Derivação pelo código:

program

statement

while_statement

WHILE LPAREN expression RPAREN LBRACE block RBRACE

WHILE LPAREN expression BIGGEREQUALS expression RPAREN LBRACE list_Declarations RBRACE

WHILE LPAREN expression BIGGEREQUALS expression RPAREN LBRACE sequence_variable SEMICOLON list_Declarations RBRACE

WHILE LPAREN expression BIGGEREQUALS expression RPAREN LBRACE sequence_variable SEMICOLON empty RBRACE

WHILE LPAREN expression BIGGEREQUALS expression RPAREN LBRACE variable_characteristics SEMICOLON empty RBRACE

WHILE LPAREN expression BIGGEREQUALS expression RPAREN LBRACE NAME ASSIGN expression SEMICOLON empty RBRACE

WHILE LPAREN expression BIGGEREQUALS expression RPAREN LBRACE NAME ASSIGN expression TIMES expression SEMICOLON empty RBRACE

WHILE LPAREN expression BIGGEREQUALS expression RPAREN LBRACE NAME ASSIGN variable TIMES literal SEMICOLON empty RBRACE

WHILE LPAREN expression BIGGEREQUALS expression RPAREN LBRACE NAME ASSIGN NAME TIMES NUMBER SEMICOLON empty RBRACE

Repetição FOR :

Sentença:
```
for (int i = a; i != b; i = c/ 255.666) {
        y = a * 5;
      }
```

Gramática:

<programa>
<declaracao>
<for_declaracao>
FOR(<tipo_primitivo> NAME = <expressao>; <expressao>; <expressao>) {<bloco>}
FOR(INT NAME = <variavel>; <expressao_logico>; <expressao_operadores>) {<lista_declaracao>}

FOR(INT NAME = NAME; <expressao>!=<expressao> ; <expressao>/<expressao>) {<sequencia_variaveis>;<lista_declaracao>}
FOR(INT NAME = NAME; <variavel>!=<variavel> ; <variavel>/<literal>) {<sequencia_variaveis>;ε}
FOR(INT NAME = NAME; NAME!=NAME; NAME/NUMBER) {<variaveis_caracteristicas> ; ε}
FOR(INT NAME = NAME; NAME!=NAME; NAME/NUMBER) {NAME = <expressao>;ε}
*FOR(INT NAME = NAME; NAME!=NAME; NAME/NUMBER) {NAME = <expressao> * <expressao>;ε}*
*FOR(INT NAME = NAME; NAME!=NAME; NAME/NUMBER) {NAME = NAME * NUMBER; ε}*

Derivação pelo código:

program

statement

for_statement

FOR LPAREN type NAME ASSIGN expression SEMICOLON expression SEMICOLON down_up RPAREN LBRACE block RBRACE

FOR LPAREN INT NAME ASSIGN expression EQUALS expression SEMICOLON expression DIFF expression SEMICOLON NAME PLUS PLUS RPAREN LBRACE sequence_variable SEMICOLON list_Declarations RBRACE

FOR LPAREN INT NAME ASSIGN variable EQUALS variable SEMICOLON variable DIFF variable SEMICOLON NAME PLUS PLUS RPAREN LBRACE sequence_variable SEMICOLON empty RBRACE

FOR LPAREN INT NAME ASSIGN NAME EQUALS NAME SEMICOLON NAME DIFF NAME SEMICOLON NAME PLUS PLUS RPAREN LBRACE variable_characteristics SEMICOLON empty RBRACE

FOR LPAREN INT NAME ASSIGN NAME EQUALS NAME SEMICOLON NAME DIFF NAME SEMICOLON NAME PLUS PLUS RPAREN LBRACE NAME ASSIGN expression SEMICOLON empty RBRACE

FOR LPAREN INT NAME ASSIGN NAME EQUALS NAME SEMICOLON NAME DIFF NAME SEMICOLON NAME PLUS PLUS RPAREN LBRACE NAME ASSIGN expression TIMES expression SEMICOLON empty RBRACE

FOR LPAREN INT NAME ASSIGN NAME EQUALS NAME SEMICOLON NAME DIFF NAME SEMICOLON NAME PLUS PLUS RPAREN LBRACE NAME ASSIGN variable TIMES literal SEMICOLON empty RBRACE

FOR LPAREN INT NAME ASSIGN NAME EQUALS NAME SEMICOLON NAME DIFF NAME SEMICOLON NAME PLUS PLUS RPAREN LBRACE NAME ASSIGN NAME TIMES NUMBER SEMICOLON empty RBRACE

Palavra	**Expressão Regular Correspondente**
Palavras Reservadas	IF, ELSE, SWITCH, CASE, DEFAULT, BREAK, INT, FLOAT, CHAR, FOR, WHILE, FOR , WHILE
Identificadores (NAME)	(a..z), (a..z)(0..9), _(a..z), _(a..z)(0..9)
Numeros (NUMBER)	(0..9)
Operadores	+ - / * == != > >= < <=
Delimitadores	; , ( ) { }