Compiladores – 2019-2

Considere a gramática seguinte para gerar o compilador da sua linguagem.

```
S → B
C → id = E
   | if ( E )M B; M else B
   | while N ( E )M do B
B → {L}
 |C
L → C;L
   |C
E → E + T
 | T

T → T * F
   | F

F → ( E )
   | id
   | n
M → ε
N → ε
```

Construir um compilador para a linguagem da gramática acima pelos métodos, tentando o sLR(1), laLR(1), ou LR(1);

1) Os símbolos gramaticais com mais de uma letra (if, else, while, do, id), bem como os delimitadores e sinais- =, (,), ;, {,},+, * - deverão ser identificados por 1 letra. Duas razões para isso: a) uma é que devem ser identificadores de enumerador, podendo ter o seu nome até mais de um caráter; b) outra é que serão índices da tabela de estados, e se puder ser resumido em 1 carater, simplificará o trabalho de quem vai fazer a tabela de estados.
2) Vamos aproveitar a ferramenta do flex ou lex, para a qual já está pronta a especificação para essa linguagem, incluída nas páginas seguintes.

3) O trabalho ficou dividido, na última sexta-feira, dia 01/11/2019, em 6 grupos.

Grupo 1: a tabela de estado, tentando o determinístico, por sLR(1), se não conseguir, por laLR(1), e se ainda não conseguir, por LR(1).
Erick Carvalho Veloso, Cesar Amorim M O, Matheus Brito Ribeiro, Nilson Junio Souza da Silva, Bruno Costa Criscuolo
Grupo 2: **biblioteca de Pilha**
Carolina Carvalhosa, Andre Luis Salgueiro Costa, Wilson Valente, Luiz Augusto;
Grupo 3: **Duas bibliotecas, a da Tabela de Símbolos e a da tabela de Quádruplas**
Wesley Henrique Costa Santos, Urias Abreu, Kleber Luiz Carlos da Silva,
Weverton Leite da Silva, Bruno Bernardo da Silva;
Grupo 4: Trata Vai Para. É o que empilha o token entregue do léxico. O Programa vai pedir trataVaiPara ou trataReducao.
Paulo Henrique, ..
Grupo 5: **Trata Redução** , gera Código, obedecendo a **Semântica – vai precisar entender bem as reduções, a pilha e a estrutura da unidade de pilha, e vai usar a tabela de símbolos e a de quádruplas**;
Gabriel Coutinho, Denilson Rodrigues, Luana Teles, Bruno Costa, João Pedro, Leonardo Santos;
Grupo 6: O loop, o programa em si (o programa com todas as chamadas e definição de dados)
Richard, Renan, Jhonatan Oliveira Alves, João Sampaio dos Santos, João Vitor Feijó Pereira, Wallace Vidal

Principais estruturas

estrutura Pilha definir o tipo e definir a variável na área global, nomear as duas variáveis que vão lidar mais com a pilha p e d .

p (será o vetor pilha) , e
d(será o vetor de desempilha (redução)
d[0] lado esquerdo da regra a ser empilhado após a redução;
d[1] a d[n] unidades obtidas do pop da direita para esquerda, 1º a unidade n e por último a unidade 1, n é o tamanho do lado direito da regra.

estrutura Tabela de Símbolos

estrutura Tabela de quádruplas.

Lexico para a linguagem

```
%option noyywrap
%{
#include <ctype.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
enum simbG{
S,C,B,L,E,T,F,M,N,v,i,a,f,p,e,w,d,o,c,m,t,n,s,q
};
struct {
char terminalOrigem[21];
int terminal;
} tabNomeTerm[12] = {
{"if",i},{"(",a},{")",f},{";",p},
{"else",e},{"while",w},{"do",d},{"{",o},{"}",c},
{"+",m},{"*",t},{"=",s}};
char cadeia [21];
char * aptCadeia = &cadeia[0];
%}
ALFA  [A-Za-z_]
PLUS            [\+]
TIMES           [\*]
DIGIT           [0-9]
NUMBER          ({DIGIT}+)
FARQ            "quit"
IF              "if"
ELSE            "else"
WHILE           "while"
DO              "do"
IDENT {ALFA}({ALFA}|{DIGIT})*
WS              [ \t]*
ATRIB "="
LP              "("
RP              ")"
LC              "{"
RC              "}"
PTV             ";"
RET             [\n]
%%

{WS}            {
                /* eat up white space */
                }
{RET}           {
                /* eat up CR LF (enter) */
                }
{PLUS}          {
                return m;
                }
{TIMES}         {
                return t;
                }
{FARQ}          {
                return (q);
```

```
  }
{IF}            {
                return i;
                }
{ELSE}          {
                return e;
                }
{WHILE}         {
                return w;
                }
{DO}            {
                return d;
                }
{ATRIB}         {
                return s;
                }
{LP}            {
                return a;
                }
{RP}            {
                return f;
                }
{LC}            {
                return o;
                }
{RC}            {
                return c;
                }
{PTV}           {
                return p;
                }
{NUMBER}        {
  strcpy(aptCadeia, yytext);
                return (n);
                }
{IDENT}         {
  strcpy(aptCadeia, yytext);
  return (v);
  }

%%
/*
esse main vai ser substituido pelo compilador
ele somente testa se os tokens vem sendo sendo gerados corretamente
*/
main () {
  int tok191;
  int k, l;
  while (1) {
    tok191 = yylex();
    if (tok191 == q){
      printf ("\nfim normal do programa \n");
      exit(0);
    }; // fim do fonte
    if (tok191 ==  v)
      printf("\nfoi lida a variavel: %s", aptCadeia);
```

```
    else
      if (tok191 ==  n)
printf("\nfoi lida a constante numerica : %s",aptCadeia);
      else { //vai ser tentada a busca por
reservadas/delimitadores/operadores
        for(k = 0; k < 12; k++)
   if (tok191 == tabNomeTerm [k].terminal){
     printf("\nfoi lido o terminal: %s",tabNomeTerm
[k].terminalOrigem);
     break; //encontrou um terminal e vai continuar o while
     }; // sucesso na busca por reservadas/delimitadores/operadores
 if (k == 12) // terminou for e busca fracassada? assim mesmo
continua busca no loop do while
   printf("\nlexema: % desconhecido na gramatica da linguagem",
cadeia);
 }; // else de tentativa de encontrar reservada/delimitador/operador
    //fim do if (tok191 ==  v) para continuar a ler o fonte
  };// while (1);
} // main
```

	original		símbolos resumidos definitivos Delimitadores e sinais viraram letras, permitindo ser enumeradores no código C		Correspondência entre símbolos originais e símbolos resumidos
0		0	S'→ S		if --- i
1	S → B	1	S → B		= --- s (store)
2	C → id = E	2	C → v s E		(--- a
3	C → if (E)M B; M else B	3	C → i a E f M B p M e B		) --- f
4	C → while N (E)M do B	4	C → w N a E f M d B		; --- p
5	B → {L}	5	B → o L c		else --- e
6	B → C	6	B → C		while --- w
7	L → C; L	7	L → C p L		do --- d
8	L → C	8	L → C		{ --- o (open)
9	E → E + T	9	E → E m T		} --- c (close)
10	E → T	10	E → T		+ --- m (mais)
11	T → T * F	11	T → T t F		* --- t (times)
12	T → F	12	T → F		id --- v
13	F → (E)	13	F → a E f		number --- n
14	F → id	14	F → v		fim str -- q(quit)
15	F → n	15	F → n		pula linha -- FARQ estou pensando em bypassar, como no branco
16	M → ε	16	M → ε		
17	N → ε	17	N → ε		

Estado 0	l. a.	Ações
S´→ .S		S 1
S → .B		B 2
B → .o L c		o 3
B → .C		C 4
C → .v s E		v 5
C → .i a E f M B p M e B		i 6
C → .w N a E f M d B		w 7

Estado 1	l. a.	Ações
S´→ S		R 0

Estado 2	l. a.	Ações
S → B.		R 1

Estado 3	l. a.	Ações
B → o .L c		L 8
L → .C p L		C 9
L → .C		C 9
C → .v s E		v 5
C → .i a E f M B p M e B		i 6
C → .w N a E f M d B		w 7

Estado 4	l. a.	Ações
B → C.		R 6

Estado 5	l. a.	Ações
C → v .s E		s 10

Estado 6	l. a.	Ações
C → i .a E f M B p M e B		a 11

Estado 7	l. a.	Ações
C → w .N a E f M d B		N 12
N → ε		R 17

Estado 8	l. a.	Ações
B → o L .c		c 13

Estado 9	l. a.	Ações
L → C .p L		p 14
L → C.		R 8

Estado 10	l. a.	Ações
C → v s .E		E 15
E → .E m T		E 15
E → .T		T 16
T → .T t F		T 16
T → .F		F 17
F → .a E f		a 18
F → .v		v 19
F → .n		n 20

Estado 11	l. a.	Ações
C → i a .E f M B p M e B		E 21
E → .E m T		E 21
E → .T		T 16
T → .T t F		T 16
T → .F		F 17
F → .a E f		a 18
F → .v		v 19
F → .n		n 20

Estado 12	l. a.	Ações
C → w N .a E f M d B		a 22

Estado 13	l. a.	Ações
B → o L c.		R 5

Estado 14	l. a.	Ações
L → C p .L		L 23
L → .C p L		C 9
L → .C		C 9
C → .v s E		v 5
C → .i a E f M B p M e B		i 6
C → .w N a E f M d B		w 7

Estado 15	l. a.	Ações
C → v s E.		R 2
E → E .m T		m 24

Estado 16	l. a.	Ações
E → T.		R 10
T → T .t F		t 25

Estado 17	l. a.	Ações
T → F.		R 12

Estado 18	l. a.	Ações
F → a .E f		E 26
E → .E m T		E 26
E → .T		T 16
T → .T t F		T 16
T → .F		F 17
F → .a E f		a 18
F → .v		v 19
F → .n		n 20

Estado 19	l. a.	Ações
F → v.		R 14

Estado 20	l. a.	Ações
F → n.		R 15

Estado 21	l. a.	Ações
C → i a E .f M B p M e B		E 27
E → E .m T		m 24

Estado 22	l. a.	Ações
C → w N a .E f M d B		E 28
E → .E m T		E 28
E → .T		T 16
T → .T t F		T 16
T → .F		F 17
F → .a E f		a 18
F → .v		v 19
F → .n		n 20