Analisador Sintático Preditivo

Professor: Luciano Ferreira Silva, Dr.

Reconhecimento de sentença

- O procedimento de reconhecimento de uma sentença com o analisador sintático preditivo é uma aplicação do autômato de pilha;
- A sentença a ser analisada é uma seqüência de tokens fornecida pelo analisador léxico, acrescida ao final do símbolo delimitador de sentença;
- O delimitador de sentença também é inserido como o primeiro elemento da pilha;
- Como a construção é descendente, a pilha recebe ainda o símbolo sentencial da gramática;

Reconhecimento de sentença

- **A operação do autômato analisa, a cada transição, o símbolo que está no topo da pilha e o primeiro símbolo da lista de tokens;**
- **As possibilidades de operação nesse analisador são:**

1. O símbolo no topo da pilha é *$* e o primeiro símbolo da lista de tokens também é *$*:
   - a sentença foi reconhecida
2. O símbolo no topo da pilha é um símbolo terminal *t* e o primeiro símbolo da lista de tokens é o mesmo símbolo terminal *t*:
   - os símbolos são eliminados do topo da pilha e do início da sentença;

Reconhecimento de sentença

3. O símbolo no topo da pilha é um símbolo não-terminal *A*, o primeiro símbolo da lista de tokens é um símbolo terminal *t* e há na tabela uma produção na linha A, coluna *t*:
   - o elemento no topo da pilha é retirado e os símbolos do lado direito da produção indicados na tabela são inseridos, em ordem reversa (da direita para a esquerda), na pilha;
   - não há alteração na lista de tokens;
4. Qualquer outra situação que não **1**, **2** e **3** representa uma condição de erro na análise, ou seja, a sentença não foi reconhecida;

Reconhecimento de sentença

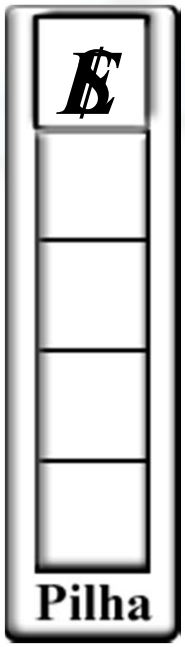
- Considere a gramática G``, V_t = {+, x, (,), v}, V_n = {*E, E`, M, M`, P*}, símbolo sentencial *E*

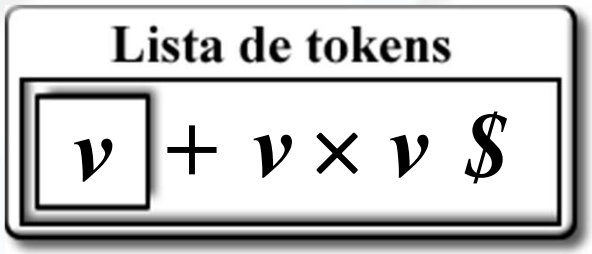
E → ME`	Produção 1: P1
E` → +ME`	Produção 2: P2
E` → ε	Produção 3: P3
M → PM`	Produção 4: P4
M` → xPM`	Produção 5: P5
M` → ε	Produção 6: P6
P → (E)	Produção 7: P7
P → v	Produção 8: P8

- Considere a sentença v + v x v para análise;

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	



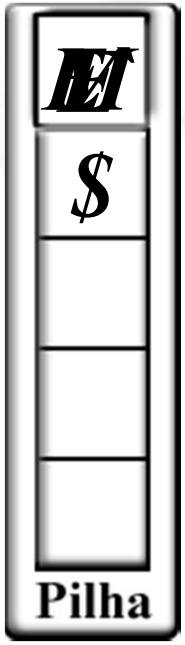


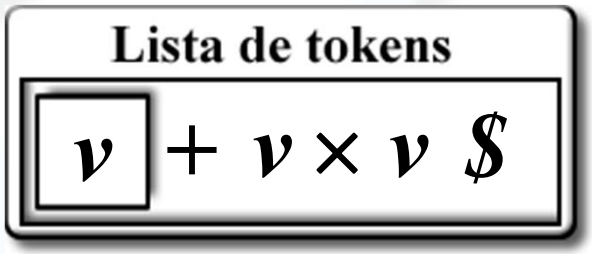
E

Tabela G``

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	





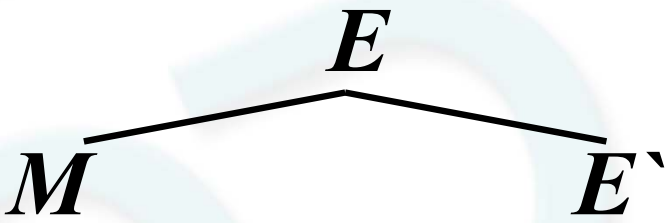
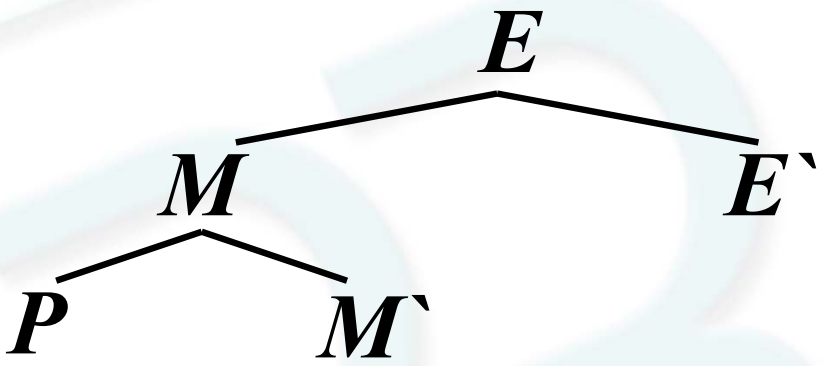
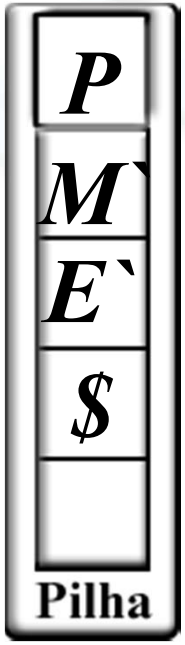


Tabela G``

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	





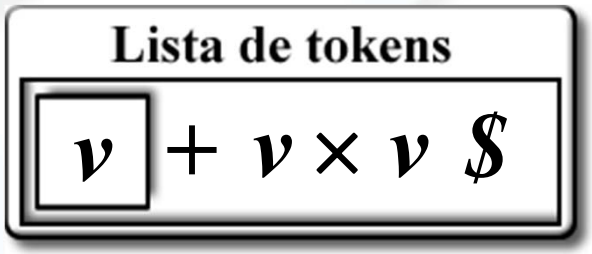
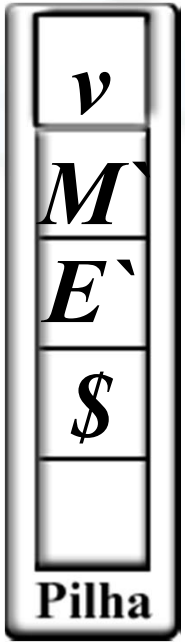
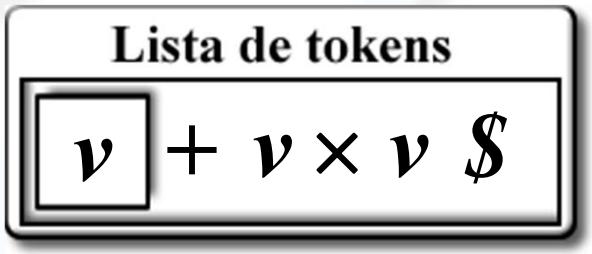


Tabela G``

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	





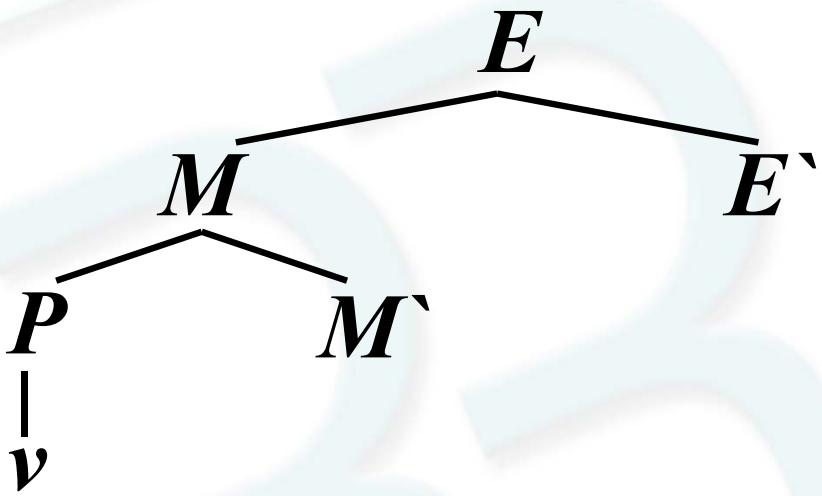
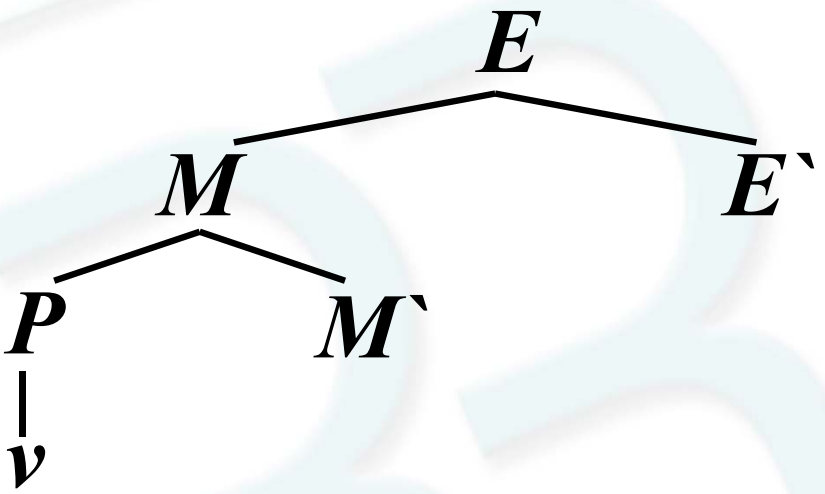


Tabela G``

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	



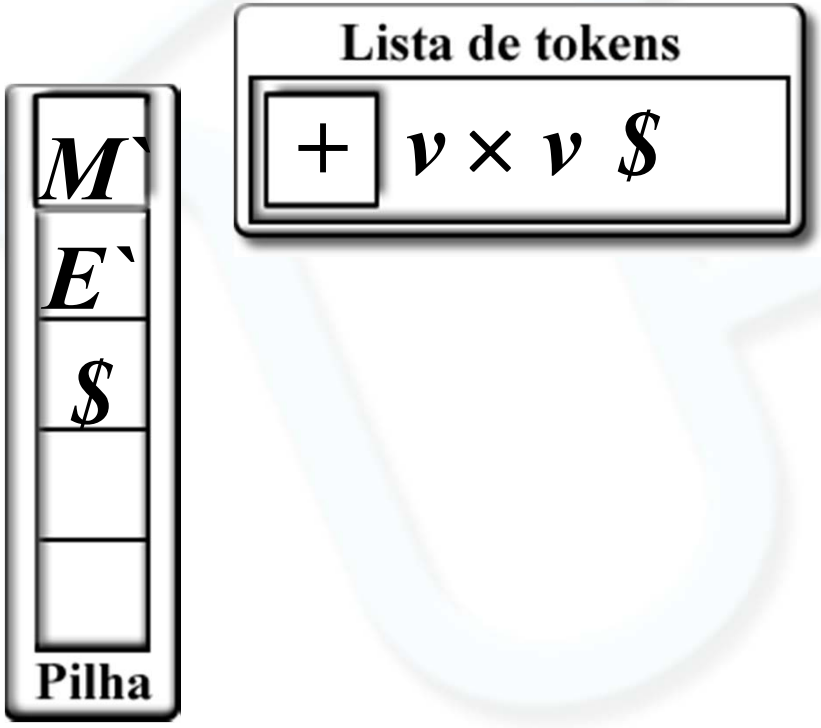
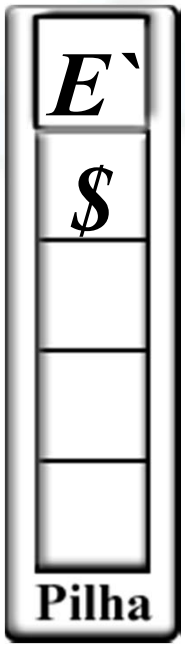


Tabela G``

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	



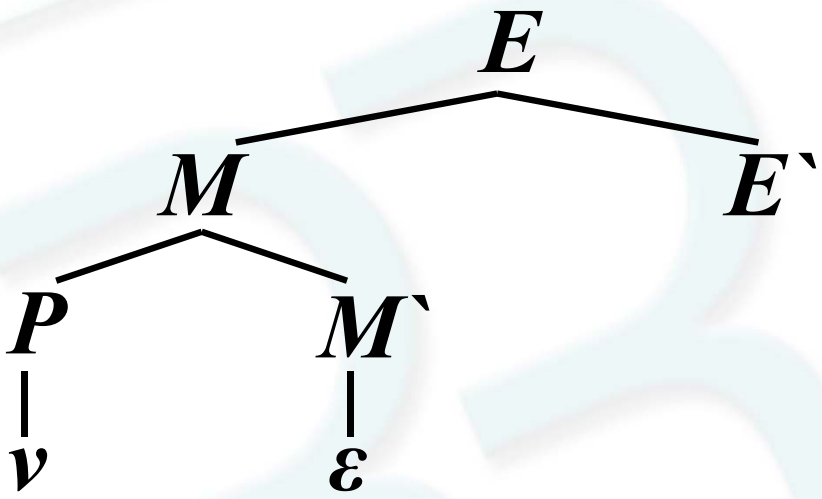
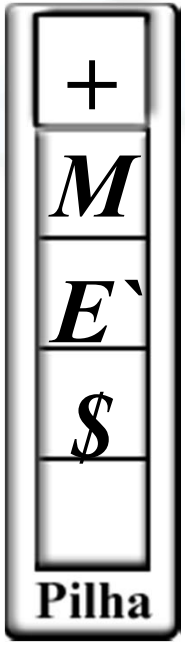


Tabela G``

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	



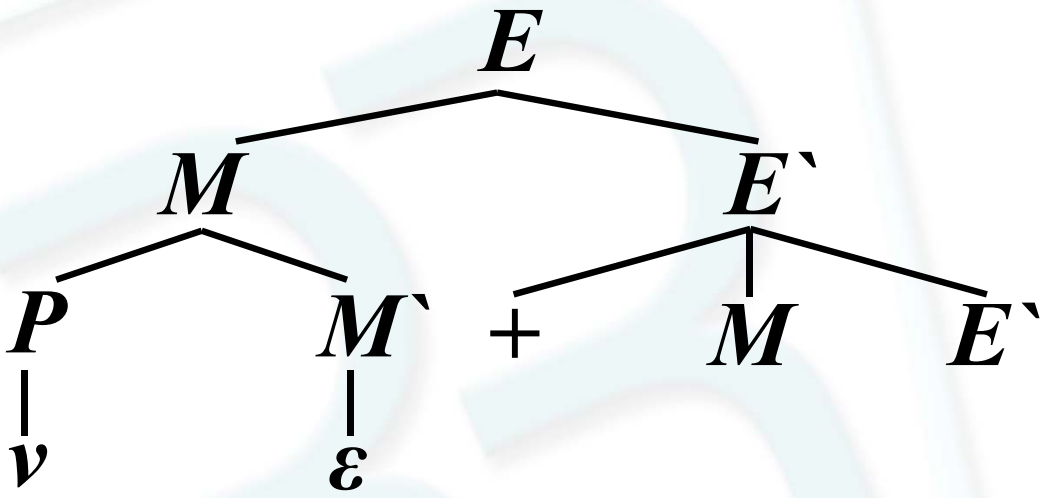
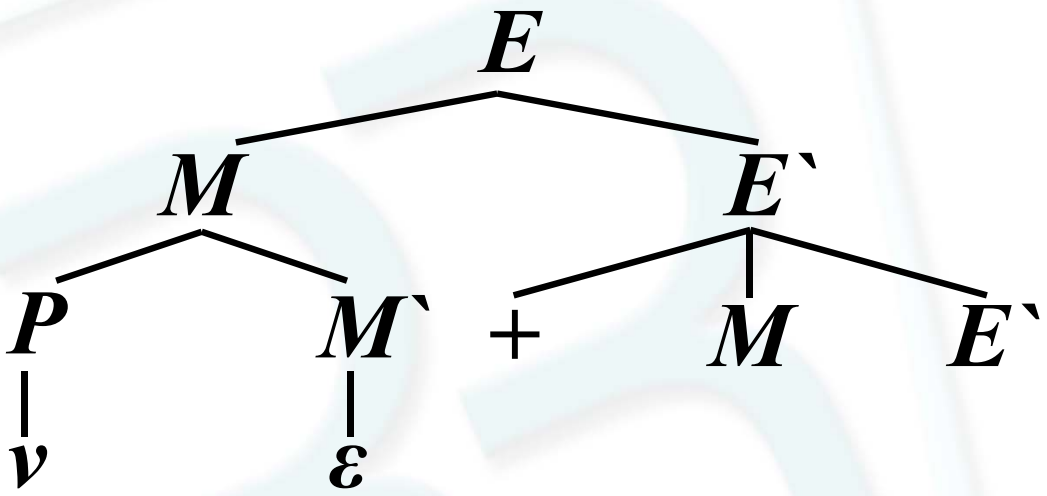
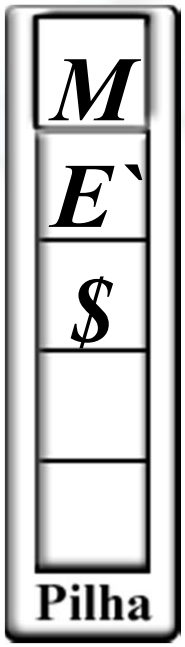


Tabela G``

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	





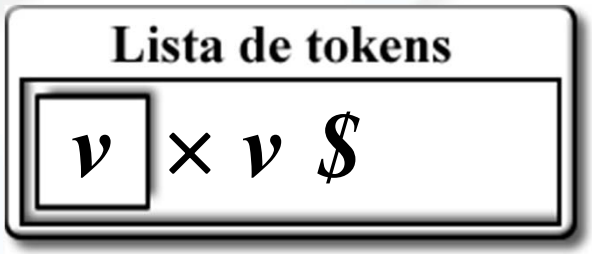
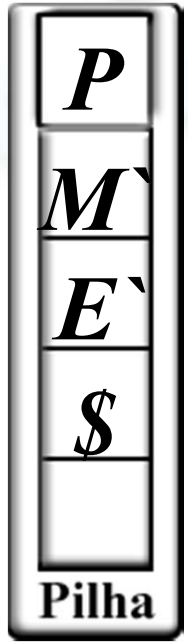
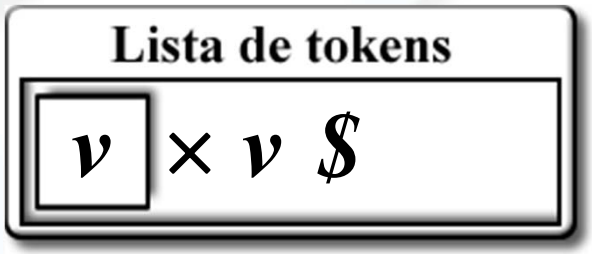


Tabela G``

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	





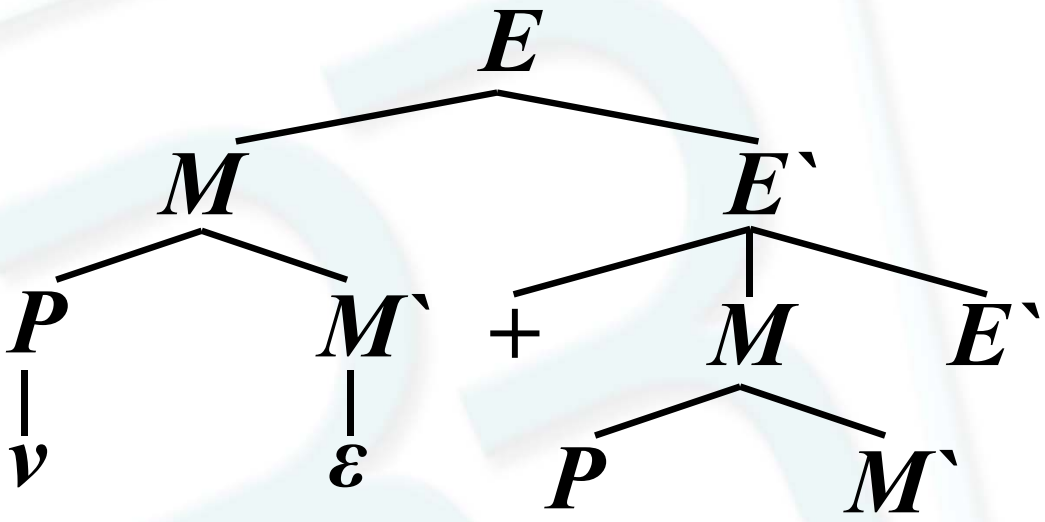
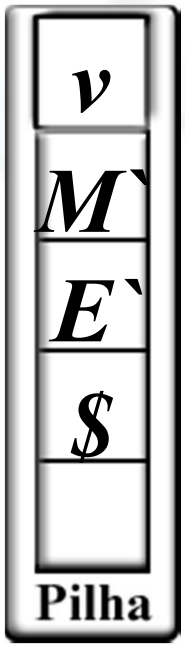
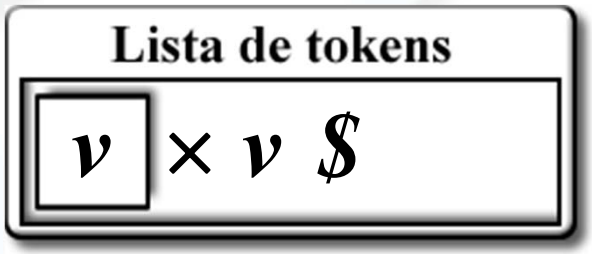


Tabela G``

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	





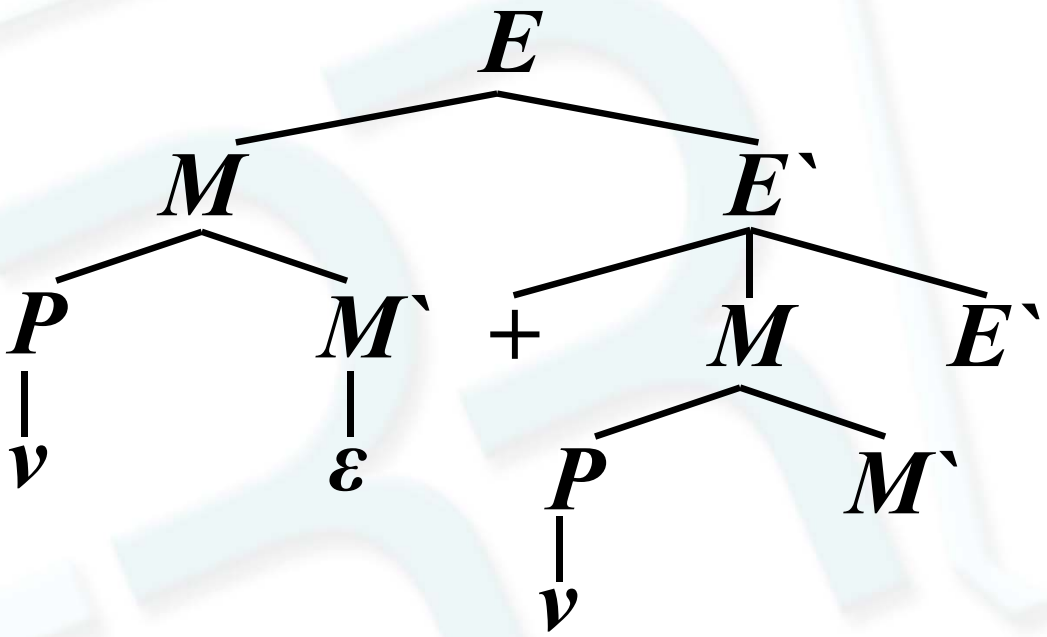
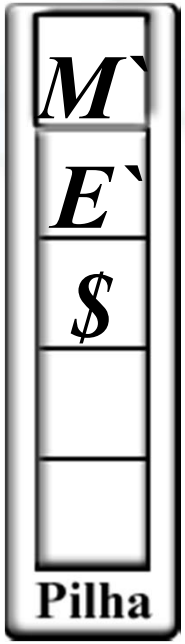


Tabela G``

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	



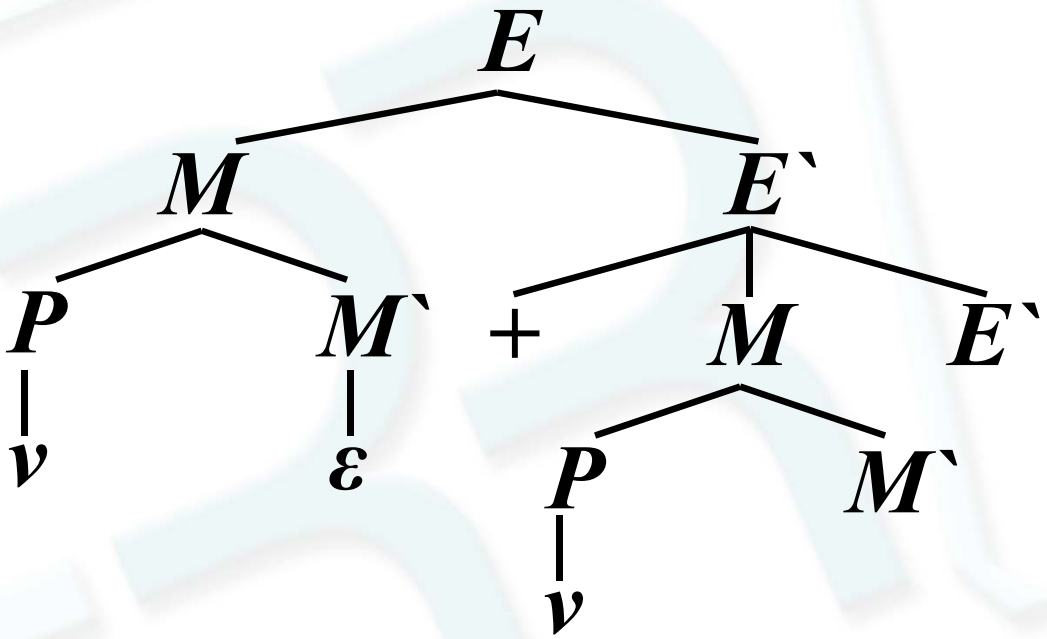
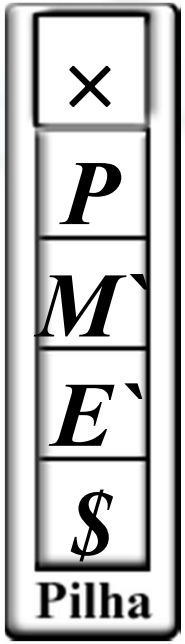


Tabela G``

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	



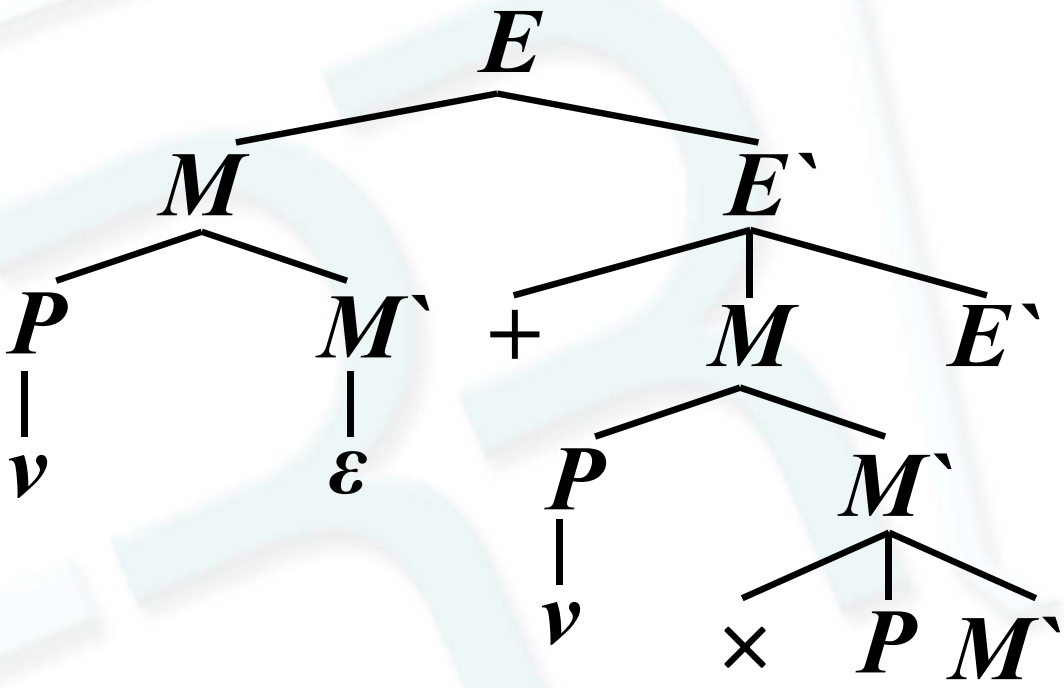
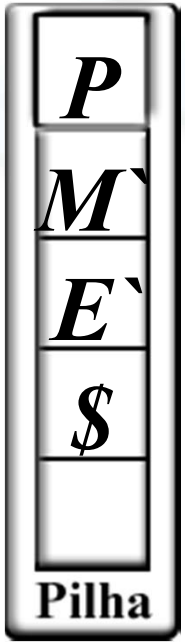


Tabela G``

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	



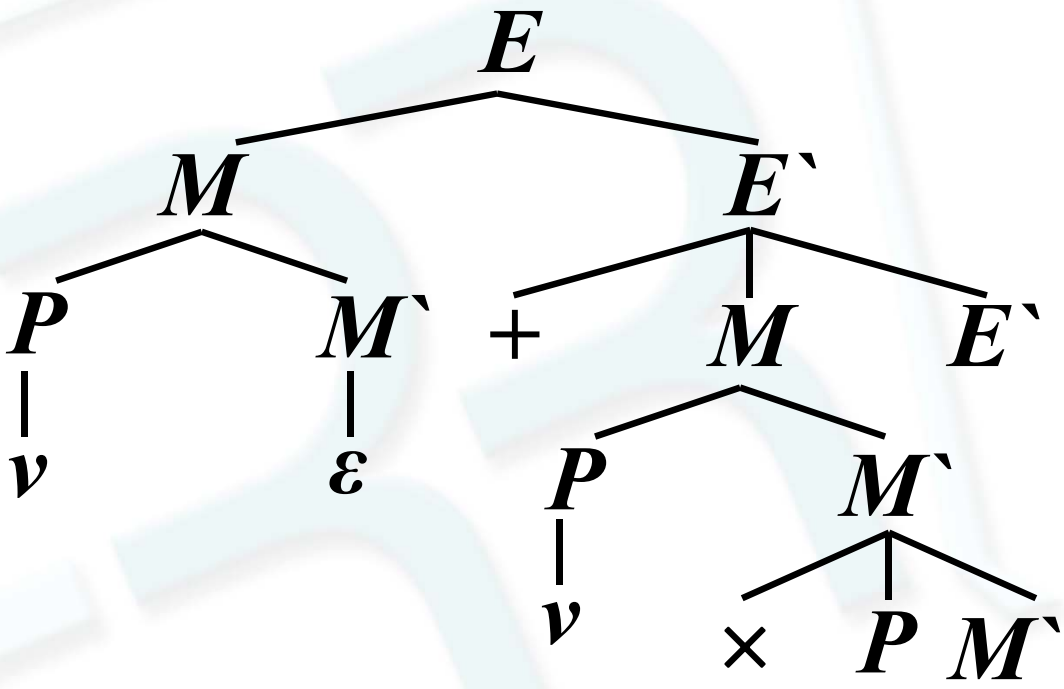
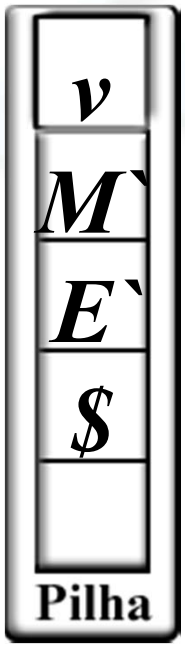


Tabela G``

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	



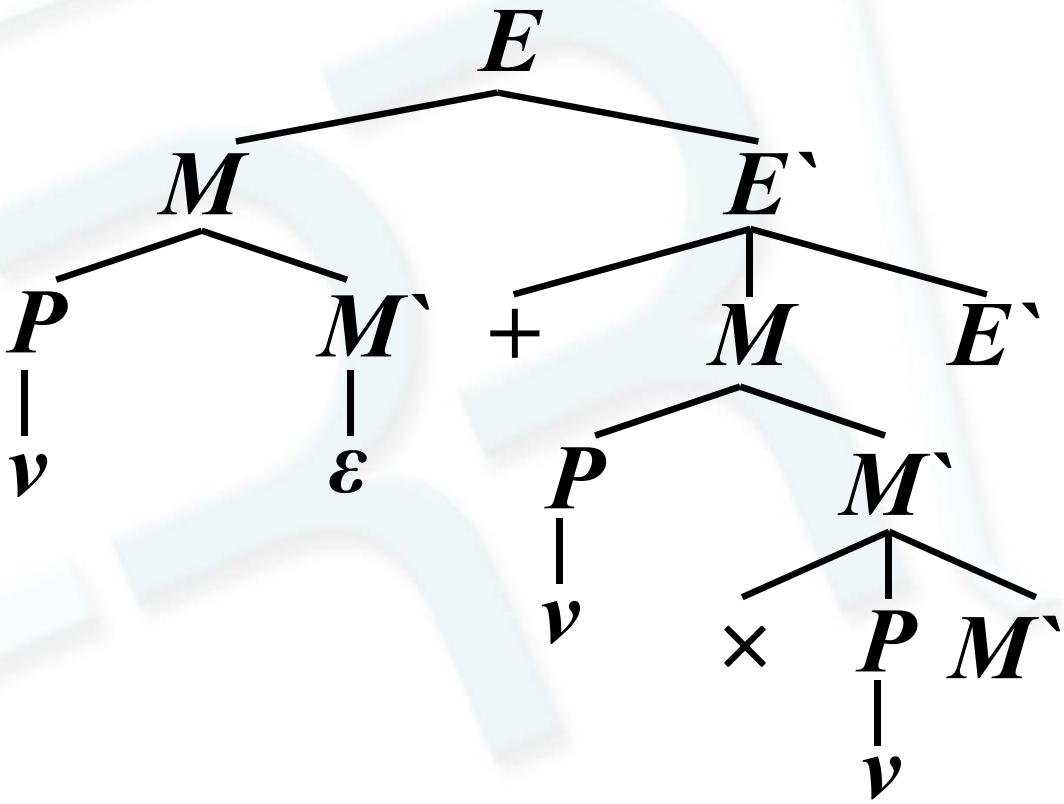
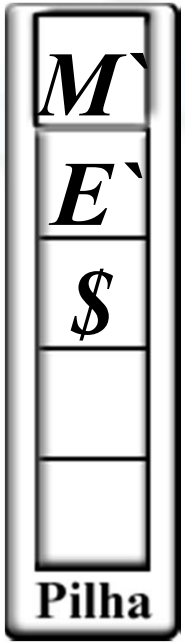


Tabela G``

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	



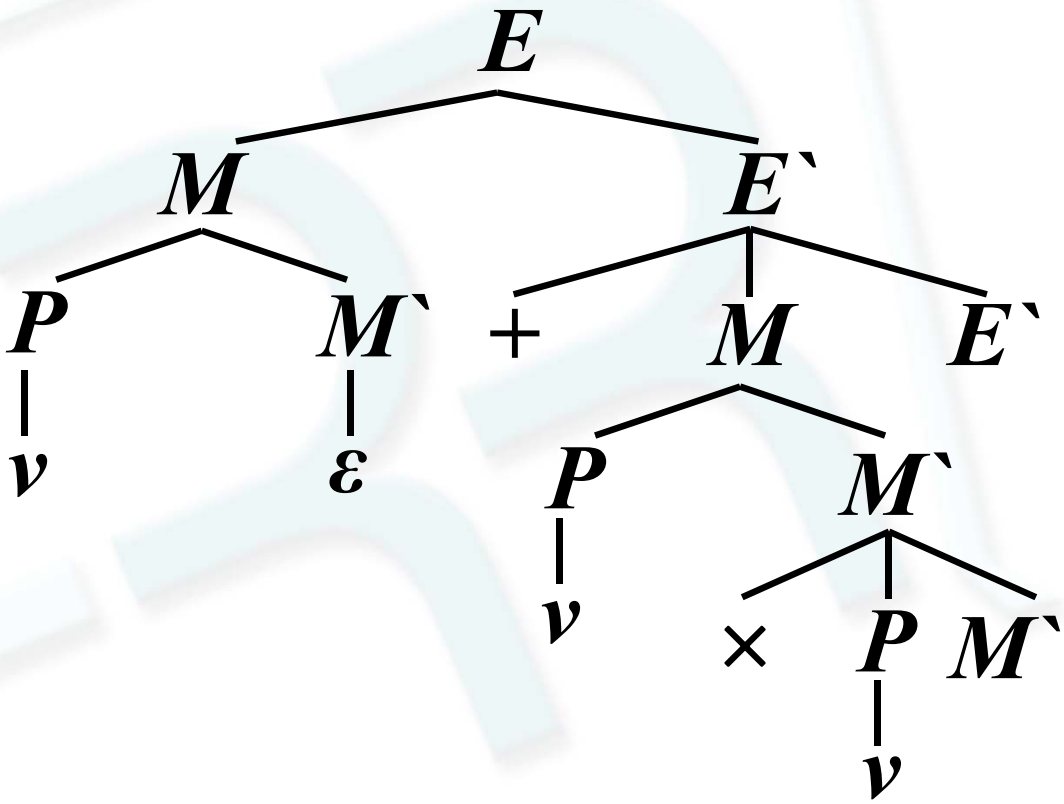
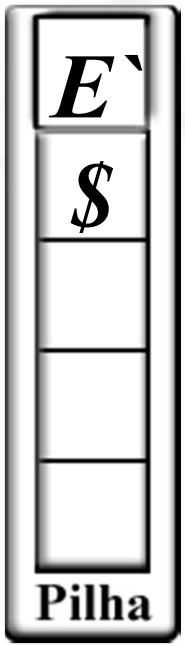


Tabela G``

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	



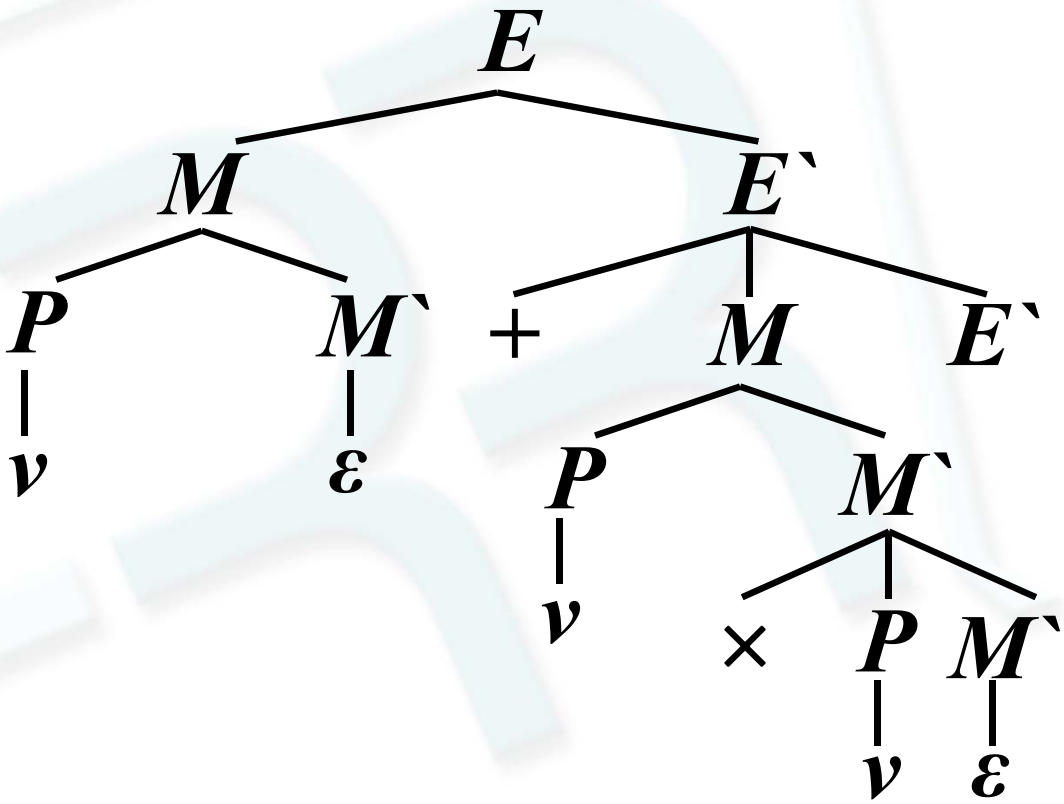
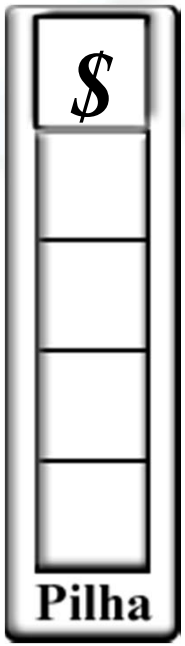
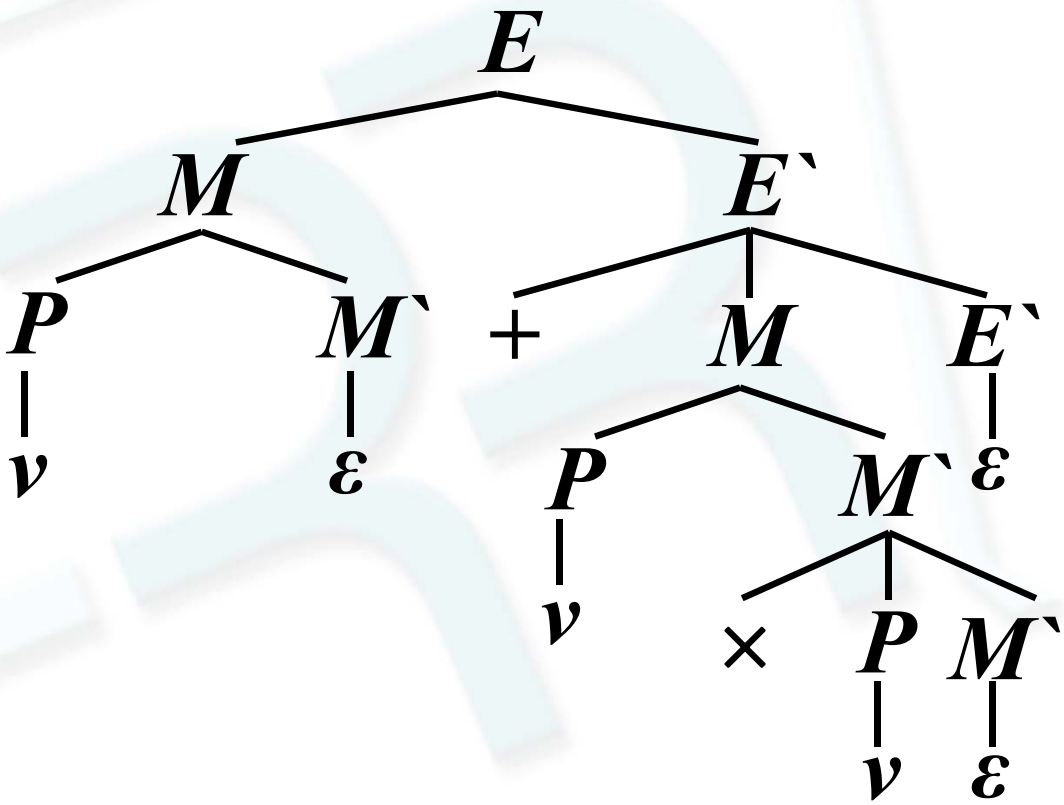


Tabela G``

Reconhecimento de sentença

	+	x	(	)	v	$
E			P1		P1	
E`	P2			P3		P3
M			P4		P4	
M`	P6	P5		P6		P6
P			P7		P8	





Fim do reconhecimento

Tabela G``

Implementação com C++


```
#include <cstdlib>
#include <iostream>
#include <string>
#include <stack>

using namespace std;

int AutomatoM (char *str)
{   /* concatenando o simbolo delimitador na sentenca */
    strcat(str,"$");

 /* Declarando a tabela sintatica para o automato M */
    int M[5][6]={{9, 9, 1, 9, 1, 9}, {2, 9, 9, 3, 9, 3}, {9,
  9, 4, 9, 4, 9}, {6, 5, 9, 6, 9, 6}, {9, 9, 7, 9, 8, 9},};
```

Implementação com C++


```
/*Declarando a pilha*/
   stack<char>pilha;

   /* Colocando o simbolo delimitador na pilha*/
   pilha.push('$');

   /* Colocando o simbolo sentencial na pilha*/
   pilha.push('E');

   /* Recebera a indexação referente ao matriz*/
   int c;
   int l;
   int i;
  /* Recebera as produções*/
   char Prod[4];
```

Implementação com C++

```
 /* Percorrendo toda a sentenca de avaliacao*/
for(i=0;str[i]!='\0';i++){
    switch(str[i]){
                    case '+':
                         c=0;
                         break;
                    case '*':
                         c=1;
                         break;
                    case '(':
                         c=2;
                         break;
                    case ')':
                         c=3;
                         break;
```

Implementação com C++

```
                    case 'v':
                         c=4;
                         break;
                     case '$':
                         c=5;
                         break;
                         default:
                  return 0;
    }
while(1)
              {
                 switch(pilha.top())
                 {
                 case 'E':
                         l=0;
                         break;
```

Implementação com C++

```
case 'F':
        l=1;
        break;
case 'M':
        l=2;
        break;
case 'N':
        l=3;
        break;
case 'P':
        l=4;
        break;
default:
        return 0;
}
```

Implementação com C++

```
/* Escolhendo a produção a ser aplicada pela tabela sintática */
            int Nprod = M[l][c];

/* Fazendo equivalência entre a produção e ordem inversa e o seu número da tabela */
            switch(Nprod)
              {
              case 1:
                      strcpy(Prod,"FM");
                      break;
              case 2:
                      strcpy(Prod,"FM+");
                      break;
              case 3:
                      strcpy(Prod,"&");
                      break;
```

Implementação com C++

```
case 4:
        strcpy(Prod,"NP");
        break;
case 5:
        strcpy(Prod,"NP*");
        break;
case 6:
        strcpy(Prod,"&");
        break;
case 7:
        strcpy(Prod,")E(");
        break;
case 8:
        strcpy(Prod,"v");
        break;
default:
        return 0;
}
```

Implementação com C++

```
 /*Aplicando a produção que leva para a string vazia*/
                if(Prod[0]=='&'){
                                pilha.pop();
                                }

/* Aplicando outros tipos de produções */
                else{pilha.pop();
                        for(int j=0;Prod[j]!='\0';j++)
                                {
                                pilha.push(Prod[j]);
                                }
                        }
```

Implementação com C++

```
/* Verificando se há igualdade no topo da pilha e o caractere
   em analise*/
        if(pilha.top()==str[i]){
                /*Reconhecimento da sentença*/
                        if(pilha.top()=='$'){
                                return 1;
                                        }
                /* Mudança de estado do autômato */
                                else{
                                   pilha.pop();
                                   break;
                                     }
                                }
            }
    }
};
```

Implementação com C++


```
int main()
{
    char sentenca[10];

   cout<<"|----> Digite a sentenca para reconhecimento; ";
    cin>>sentenca;
    cout<<endl;
    cout<<endl;
    int res = AutomatoM(sentenca);
    if(res==1)
 cout<<"|----> O automato reconheceu a sentenca; "<<endl;
    else
 cout<<"|----> O automato reconheceu NAO a sentenca; "<<endl;
  system("PAUSE");
  return EXIT_SUCCESS;
}
```

5° Trabalho

- Modifique o programa proposto nesta aula de forma a armazenar de maneira adequada a árvore sintática em uma estrutura de dados tipo árvore;
- Observação: Esta construção é descendente logo usa a estratégia de varredura pré-ordem, sua árvore deve propiciar esta estratégia;

5° Trabalho

1. **Pesquise e proponha uma gramática LL(1), sem ambigüidades, e sem recursões a esquerda;**
2. **Construa a sua tabela sintática;**
3. **Programe o analisador preditivo completo que reconhece esta gramática;**
    1. Ele deve armazenar a árvore sintática em uma estrutura de dados tipo árvore quando reconhecer uma sentença desta gramática;
    2. Ele deve emitir mensagens corretivas quando não reconhecer a sentença;

Reconhecimento de sentença

- Considere a gramática G``, V_t = {+, x, (,), v}, V_n = {*E, E`, M, M`, P*}, símbolo sentencial *E*

E → ME`	Produção 1: P1
E` → +ME`	Produção 2: P2
E` → ε	Produção 3: P3
M → PM`	Produção 4: P4
M` → xPM`	Produção 5: P5
M` → ε	Produção 6: P6
P → (E)	Produção 7: P7
P → v	Produção 8: P8