UNIVERSIDADE FEDERAL DA FRONTEIRA SUL

Curso de Ciência da Computação

Disciplina: CONSTRUÇÃO DE COMPILADORES

Período: 2013.2

**Analisador léxico do Ptgrila**

ALEX REIMANN CUNHA LIMA

ANDREY VINICIUS FAGUNDES

GUSTAVO KNOB

MICHEL COSTA

RAMON PERONDI

Chapecó – SC, Novembro de 2013

***Resumo***

[Análise léxica](http://pt.wikipedia.org/wiki/Análise_léxica) é o processo de verificar uma entrada com linhas de caracteres e produzir uma sequência de símbolos chamado "símbolos léxicos" (lexical tokens) ou apenas "símbolos" ([tokens](http://pt.wikipedia.org/wiki/tokens)), que podem ser manipulados mais facilmente por um leitor de saída ([parser](http://pt.wikipedia.org/wiki/parser)). Em outras palavras, a análise léxica é a forma de conferir um determinado [alfabeto](http://pt.wikipedia.org/wiki/alfabeto). Quando analisamos lexicamente uma palavra, podemos ver se existe ou não algum [caractere](http://pt.wikipedia.org/wiki/Caractere) que não faz parte do alfabeto de uma linguagem.

O componente do compilador responsável pela execução desse processo é conhecido como [Analisador léxico](http://pt.wikipedia.org/wiki/Analisador_léxico) (scanner), o qual iremos implementar no presente trabalho.

***Introdução***

O objetivo deste trabalho é conhecer um pouco mais a fundo o analisador léxico de um compilador: sua estrutura e seu funcionamento. Mas para isso, foi preciso entender mais a fundo sobre linguagem de programação: suas gramaticas, regras e sintaxes. Para tanto, foi desenvolvido uma linguagem hipotética: a linguagem Ptgrila, que foi construída unicamente para aplicar os conceitos vistos em sala de aula afim de compreender o funcionamento das partes de um compilador, bem como noções de gramatica e linguagens formais.

***Tokens da linguagem***

A linguagem Ptgrila é composta basicamente de palavras reservadas: palavras que devem aparecer literalmente na linguagem, sem variações (e.g. SI, SINON, PUIS, etc). Constantes: literais TRUE e FALSE, símbolos especiais: sequências de um ou mais símbolos que não podem aparecer em identificadores nem palavras reservadas. São utilizados para composição de expressões aritméticas ou lógicas, comando de atribuição, etc. São exemplos de símbolos especiais: “<” (menor), “:” (dois pontos), “:” (atribuição), etc. Identificadores/variáveis: palavras que seguem algumas regras de escrita, porém podem assumir diversos valores. São definidos através da expressão regular: “[\$]+[a-b-c]{3}”, ou seja, palavras que começam obrigatoriamente com o caractere ‘$’ onde o restante pode ser composto pelas letras a,b ou c e devem ter tamanho igual a 3, exemplo: $aaa, $abc, $cbc, etc. O objetivo de usar esta regra para a criação das variáveis foi apenas para facilitar na construção dos autômatos e, consequentemente, na criação do analisador léxico.

As tabelas abaixo mostram todos os tokens desta linguagem:

|  |  |
| --- | --- |
| **Palavras reservadas** | |
| se | si |
| então | puis |
| senao | sinon |
| enquanto | pendant |
| faça | faire |
| para | paire |
| inteiro | ensemble |
| flutuante | flottant |
| caractere | lettre |
| palavra | mat |
| incluir | coprendre |
| principal | principale |
| retornar | retour |
|  |  |
| **Constantes** | |
| verdadeiro | true |
| falso | nalse |
|  |  |
| **Variáveis / Identificadores** | |
| [\$]+[a-b-c]{3} | |
|  |  |
| **símbolos especiais** | |
| Multipli-caçao | \* |
| divisão | / |
| adiçao | + |
| subtraçao | - |
| igualdade | = |
| Desigual-dade | != |
| mod | % |
| menor que | < |
| maior que | > |
| menor ou igual | <= |
| maior ou igual | >= |
| comentário | // |
| atribuição | : |
| abre parêntese | ( |
| fecha parêntese | ) |
| operador e | ^ |
| operador ou | v |

E do alfabeto definido por L = {S,I,P,U,N,O,E,D,A,T,F,R,P,M,B,L,C,$,\*,/,+,-,=,!,%,<,>,:,(,),^,V}.

***Tabelas de transições***

Uma tabela de transição (tabela de estados) é uma tabela que mostra para qual estado a máquina de estados finitos (autômato finito) irá se mover, com base no estado corrente e do próximo evento ocorrido. Uma tabela de estados é, essencialmente, uma [tabela verdade](http://pt.wikipedia.org/wiki/Tabela_verdade) que mapeia o estado atual para um próximo estado. Ela é uma das muitas maneiras de especificar o comportamento de um autômato finito (AF) que verifica os tokens da linguagem. Um AF é o tipo mais simples de reconhecedor de linguagem e pode ser usado como reconhecedor de padrões de processamento de textos e/ou analisador léxico em linguagens de programação (como será usado para este trabalho).

Abaixo, as tabelas de transições da linguagem Ptgrila.

**Palavras reservadas**

|  |  |
| --- | --- |
| si | |
| S | s1 |
| 2 | i |
|  |  |
| puis | |
| S | p3 |
| 3 | u4 |
| 4 | i5 |
| 5 | s |
|  |  |
| sinon | |
| S | s6 |
| 6 | i7 |
| 7 | n8 |
| 8 | o9 |
| 9 | n |
|  |  |
| pendant | |
| S | p10 |
| 10 | e11 |
| 11 | n12 |
| 12 | d13 |
| 13 | a14 |
| 14 | n15 |
| 15 | t |
|  |  |
| faire | |
| S | f16 |
| 16 | a17 |
| 17 | i18 |
| 18 | r19 |
| 19 | e |
|  |  |
| paire | |
| S | p20 |
| 20 | a21 |
| 21 | i22 |
| 22 | r23 |
| 23 | e |
|  |  |
| ensemble | |
| S | e24 |
| 24 | n25 |
| 25 | s26 |
| 26 | e27 |
| 27 | m28 |
| 28 | b29 |
| 29 | l30 |
| 30 | e |
|  |  |
| flottant | |
| S | f31 |
| 31 | l32 |
| 32 | o33 |
| 33 | t34 |
| 34 | t35 |
| 35 | a36 |
| 36 | n37 |
| 37 | t |
|  |  |
| lettre | |
| S | l38 |
| 38 | e39 |
| 39 | t40 |
| 40 | t41 |
| 41 | r42 |
| 42 | E |
|  |  |
| Mat | |
| S | m43 |
| 43 | a44 |
| 44 | T |
|  |  |
| coprendre | |
| S | c45 |
| 45 | o46 |
| 46 | p47 |
| 47 | r48 |
| 48 | e49 |
| 49 | n50 |
| 50 | d51 |
| 51 | r52 |
| 52 | e |
| principale | |
| S | p53 |
| 53 | r54 |
| 54 | i55 |
| 55 | n56 |
| 56 | c57 |
| 57 | i58 |
| 58 | p59 |
| 59 | a60 |
| 60 | l61 |
| 61 | e |
|  |  |
| retour | |
|  | r62 |
| 62 | e63 |
| 63 | t64 |
| 64 | o65 |
| 65 | u66 |
| 66 | r |

**Constantes**

|  |  |
| --- | --- |
| true | |
| S | t67 |
| 67 | r68 |
| 68 | u69 |
| 69 | e |
|  |  |
|  |  |
| nalse | |
| S | n70 |
| 70 | a71 |
| 71 | l72 |
| 72 | s73 |
| 73 | e |

**Variáveis / identificadores**

|  |  |
| --- | --- |
| **Variáveis / identificadores** | |
| S | $74 |
| 74 | a75,b75,c75 |
| 75 | a76,b76,c76 |
| 76 | a,b,c |

**Símbolos especiais**

|  |  |
| --- | --- |
| Multiplicação | |
| S | \* |
|  |  |
| Divisão | |
| S | / |
|  |  |
| Adição | |
| S | + |
|  |  |
| Subtração | |
| S | - |
|  |  |
| Igual | |
| S | = |

|  |  |
| --- | --- |
| Comentário | |
| S | /104 |
| 104 | / |
|  |  |
| Diferente | |
| S | !92 |
| 92 | = |
|  |  |
| Mod | |
| S | % |
|  |  |
| Menor que | |
| S | < |
|  |  |
| Maior que | |
| S | > |
|  |  |
| Menor ou igual | |
| S | <92 |
| 97 | = |
|  |  |
| Maior ou igual | |
| S | >92 |
| 98 | = |
|  |  |
| Atribuição | |
| S | : |
|  |  |
| Abre parêntese | |
| S | ( |
|  |  |
| Fecha parêntese | |
| S | ) |
|  |  |
| E (^) | |
| S | ^ |
|  |  |
| Ou (v) | |
| S | v |

***Nomenclatura de estados finais***

A seguinte tabela traz os estados finais da linguagem, bem como seus respectivos nomes. Ao todo, são 34 estados: um para cada token da linguagem e o estado de erro (0). Para identificar quais são os estados finais foram usados números negativos, apenas para simplificar a implementação do algoritmo escrito em linguagem C.

|  |  |
| --- | --- |
| Estado final | Nome estado |
| 0 | Estado de erro |
| -2 | si |
| -6 | puis |
| -10 | sinon |
| -16 | pendant |
| -20 | faire |
| -24 | paire |
| -31 | ensemble |
| -38 | flottant |
| -43 | lettre |
| -45 | mat |
| -53 | coprendre |
| -62 | principale |
| -67 | retour |
| -70 | true |
| -74 | nalse |
| -77 | Variáveis / identificadores |
| -78 | \* |
| -79 | / |
| -80 | + |
| -90 | - |
| -91 | = |
| -93 | != |
| -94 | % |
| -95 | < |
| -96 | > |
| -99 | : |
| -100 | ( |
| -101 | ) |
| -102 | ^ |
| -103 | v |
| -105 | // |
| -106 | <= |
| -107 | >= |

**Autômato finito determinístico do Ptgrila**

**T**odos os AFs implementados foram mapeados para uma única tabela para que assim pudessem ser determinizados e simplificados, facilitando na construção do analisador léxico. A gramatica de linguagem Ptgrila está definida na tabela de transições abaixo:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | S | I | P | U | N | O | E | D | A | T | F | R | M | B | L | C | $ | \* | / | + | - | = | ! | % | < | > | : | ( | ) | ^ | V |
| S | 160 | 0 | 3102053 | 0 | 70 | 0 | 24 | 0 | 0 | 67 | 163170 | 62 | 43 | 0 | 38 | 45 | 74 | -78 | -79104 | -80 | -90 | -91 | 92 | -94 | -9597 | -9698 | -99 | -100 | -101 | -102 | -103 |
| 4 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | -6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -7 | 0 | 0 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | -10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | -20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 21 | 0 | 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 23 | 0 | 0 | 0 | 0 | 0 | 0 | -24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 24 | 0 | 0 | 0 | 0 | 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 25 | 26 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 26 | 0 | 0 | 0 | 0 | 0 | 0 | 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 30 | 0 | 0 | 0 | 0 | 0 | 0 | -31 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 32 | 0 | 0 | 0 | 0 | 0 | 33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 34 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 34 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 35 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 35 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 36 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 36 | 0 | 0 | 0 | 0 | 37 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 37 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -38 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 38 | 0 | 0 | 0 | 0 | 0 | 0 | 39 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 39 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 40 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 40 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 41 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 41 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 42 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 42 | 0 | 0 | 0 | 0 | 0 | 0 | -43 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 43 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 44 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 44 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -45 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 45 | 0 | 0 | 0 | 0 | 0 | 46 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 46 | 0 | 0 | 47 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 47 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 48 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 48 | 0 | 0 | 0 | 0 | 0 | 0 | 49 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 49 | 0 | 0 | 0 | 0 | 50 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 50 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 51 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 51 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 52 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 52 | 0 | 0 | 0 | 0 | 0 | 0 | -53 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 53 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 54 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 54 | 0 | 55 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 55 | 0 | 0 | 0 | 0 | 56 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 56 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 57 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 57 | 0 | 58 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 58 | 0 | 0 | 59 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 59 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 60 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 60 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 61 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 61 | 0 | 0 | 0 | 0 | 0 | 0 | -62 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 62 | 0 | 0 | 0 | 0 | 0 | 0 | 63 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 63 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 64 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 64 | 0 | 0 | 0 | 0 | 0 | 65 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 65 | 0 | 0 | 0 | 66 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 66 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -67 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 67 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 68 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 68 | 0 | 0 | 0 | 69 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 69 | 0 | 0 | 0 | 0 | 0 | 0 | -70 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 70 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 71 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 71 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 72 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 72 | 73 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 73 | 0 | 0 | 0 | 0 | 0 | 0 | -74 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 74 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 75 | 0 | 0 | 0 | 0 | 75 | 0 | 75 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 75 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 76 | 0 | 0 | 0 | 0 | 76 | 0 | 76 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 76 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -77 | 0 | 0 | 0 | 0 | -77 | 0 | -77 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 92 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -93 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -31 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -38 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -43 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -45 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -53 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -62 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -67 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -70 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -74 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -77 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -78 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -80 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -91 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -93 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -94 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -99 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -100 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -101 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -102 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -103 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -105 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -106 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -107 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 163170 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 17 | 0 | 0 | 0 | 0 | 0 | 32 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 160 | -2 | -7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3102053 | 0 | 0 | 0 | 4 | 0 | 0 | 11 | 0 | 21 | 0 | 0 | 54 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -9597 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -106 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -9698 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -107 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -79104 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -105 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

***Código fonte***

**#include <stdlib.h>**

**#include <string.h>**

**#include <stdio.h>**

**#include <stdbool.h>**

**#include <ctype.h>**

**#define FILE\_NAME "data/auto.txt"**

**static** **int** num\_states;

**static** **int** num\_symbols;

**static** **int** **\***table;

**static** **int** **\***states;

**static** **char** **\***symbols;

**static** **bool** **\***final\_states;

**static** **int** sym2col[256];

**#define TAB(state, symbol) table[(state)\*num\_symbols+symbol]**

**void** **readfile**(**FILE** **\***file, **int** phase)

{

**int** row;

**char** **\***lineptr;

    fseek(file, 0, SEEK\_SET);

    row **=** 0;

**for** (;;) {

**size\_t** n;

**char** **\***endptr;

**int** cols;

**int** nread;

        lineptr **=** NULL;

        nread **=** getline(**&**lineptr, **&**n, file);

**if** (nread **<** 0) {

            free(lineptr);

**break**;

        }

        endptr **=** lineptr;

        cols **=** 0;

**if** (row **==** 0) {

**int** i;

**for** (i **=** 0; lineptr[i] **!=** '\0'; i**++**) {

**unsigned** **char** c **=** lineptr[i];

**if** (c **>=** 'A' **&&** c **<=** 'Z') {

                    c **+=** ('a' **-** 'A');

                }

**if** (c **>** ' ') {

**if** (phase **==** 1) {

                        symbols[cols] **=** c;

                        sym2col[(**int**)c] **=** cols **+** 1;

                    }

                    cols**++**;

                }

            }

        } **else** {

**for** (;;) {

**const** **char** **\***oldnptr **=** endptr;

**int** number **=** strtol(endptr, **&**endptr, 0);

**if** (number **==** 0 **&&** oldnptr **==** endptr) {

**break**;

                }

**if** (phase **==** 1 **&&** cols **==** 0) {

                    states[row **-** 1] **=** number;

                    final\_states[row **-** 1] **=** (number **<** 0);

                } **else** **if** (phase **==** 2 **&&** cols **>** 0) {

**int** i;

**int** state\_code;

**if** (number **==** 0) {

                        state\_code **=** **-**1;

                    } **else** {

                        state\_code **=** **-**1;

**for** (i **=** 0; i **<** num\_states; i**++**) {

**if** (states[i] **==** number) {

                                state\_code **=** i;

**break**;

                            }

                        }

                    }

                    TAB(row **-** 1, cols **-** 1) **=** state\_code;

                }

                cols**++**;

            }

            cols**--**;

        }

*//printf("%d agora %d\n", row, cols);*

**if** (num\_symbols **<** cols) {

            num\_symbols **=** cols;

        } **else** **if** (cols **<** num\_symbols) {

            fprintf(stderr, "ERROR! wrong number of symbols on row %d\n", row **+** 1);

        }

        row**++**;

        free(lineptr);

    }

    row**--**;

    num\_states **=** row;

*//printf("%d\n", num\_symbols);*

*//printf("%d\n", num\_states);*

}

**static** **void** **showtab**(**void**)

{

**int** i, j;

**for** (i **=** 0; i **<** num\_symbols; i**++**) {

        printf("%c", symbols[i]);

    }

    printf("\n");

**for** (i **=** 0; i **<** num\_states; i**++**) {

        printf("%c %d\n", final\_states[i] **?** '\*' **:** ' ', states[i]);

    }

    printf("\n");

**for** (i **=** 0; i **<** num\_states; i**++**) {

**for** (j **=** 0; j **<** num\_symbols; j**++**) {

            printf("%d ", TAB(i, j));

        }

        printf("\n");

    }

}

**static** **const** **char** **\*doparse**(**const** **char** **\***tape)

{

**int** i;

**int** state;

**for** (state **=** 0, i **=** 0; tape[i] **!=** '\0' **&&** **!**isspace(tape[i]); i**++**) {

**unsigned** **char** sym **=** tape[i];

**int** col **=** sym2col[(**int**)sym];

*//printf("state %d(%d)ffoi %c %d\n", state, states[state], sym, col);*

**if** (col **==** 0) {

*//printf("symbol not found\n");*

            printf("%d ", 0);

**return** NULL;

        }

        col**--**;

        state **=** TAB(state, col);

**if** (state **<** 0) {

*//printf("unexpected symbol (%c)\n", sym);*

            printf("%d ", 0);

**return** NULL;

        }

    }

    printf("%d ", states[state]);

*//printf("State = %d\n", states[state]);*

**return** (tape[i] **==** '\0' **||** tape[i] **==** '\n' **||** tape[i] **==** '\r') **?** NULL **:** **&**tape[i **+** 1];

}

**static** **void** **lexparse**(**const** **char** **\***tape)

{

**do** {

        tape **=** doparse(tape);

    } **while**(tape **!=** NULL);

}

**int** **main**(**void**)

{

**FILE** **\***file;

**int** nread;

**size\_t** n;

**char** **\***lineptr;

    file **=** fopen(FILE\_NAME, "rt");

**if** (file **==** NULL) {

        fprintf(stderr, "Unable to open %s\n", FILE\_NAME);

**return** 1;

    }

    num\_states **=** 0;

    num\_symbols **=** 0;

    memset(sym2col, 0, **sizeof**(sym2col));

    readfile(file, 0);

    table **=** (**int** **\***)malloc(**sizeof**(**int**) **\*** num\_states **\*** num\_symbols);

    symbols **=** (**char** **\***)malloc(**sizeof**(**char**) **\*** num\_symbols);

    states **=** (**int** **\***)malloc(**sizeof**(**int**) **\*** num\_states);

    final\_states **=** (**bool** **\***)malloc(**sizeof**(**bool**) **\*** num\_states);

    readfile(file, 1);

    readfile(file, 2);

*//showtab();*

    lineptr **=** NULL;

    nread **=** getline(**&**lineptr, **&**n, stdin);

**if** (lineptr **!=** NULL) {

*//lexparse("SI PUIS SINON PENDANT FAIRE PAIRE ENSEMBLE FLOTTANT LETTRE MAT COPRENDRE PRINCIPALE RETOUR TRUE NALSE $AAA $CBA \* / + - = != % < > <= >= : ( ) ^ V //");*

*//lexparse("si i puis sinon pendant faire paire ensemble flottant lettre mat coprendre principale retour true nalse $aaa $cba \* / + - = != % < > <= >= : ( ) ^ v //");*

        lexparse(lineptr);

        free(lineptr);

        printf("\n");

    }

*//lexparse("SI PUIS");*

    fclose(file);

    free(states);

    free(symbols);

    free(table);

**return** 0;

}

***Conclusão***

Um analisador precisa se basear em uma linguagem, e esta linguagem pode ter um ou mais scanners diferentes, além disso, cada analisador pode verificar apenas uma única linguagem. Foi preciso então construir primeiramente a linguagem de programação, para assim o analisador léxico fosse implementado. A linguagem criada possui tokens, autômatos e tabelas de transições próprias. Esta análise é a primeira etapa do processo de compilação de um código fonte e seu objetivo é dividi-lo em símbolos, preparando-o para a próxima fase de compilação: a [Análise Sintática](http://pt.wikipedia.org/wiki/Análise_Sintática). Ela é responsável pela: extração e classificação dos tokens, eliminação de delimitadores e comentários e recuperação de erros.