

## 1. Descrição do trabalho:

Como parte da nota da segunda avaliação de Compiladores para o Curso de Ciência da Computação, os alunos deverão desenvolver um tradutor de linguagens. Esse tradutor, baseado na técnica de tradução dirigida pela sintaxe deverá conter as fases descritas no item 3 desse documento e ser desenvolvido em Linguagem C, nas plataformas de sistemas operacionais Windows ou GNU/Linux.

## 2. Objetivo

O objetivo principal dessa atividade é implementar alguns mecanismos de compilação e incentivar o estudo da teoria que abrange as fases de um compilador de linguagens de programação.

## 3. As Fases do Compilador:

Como pode ser ilustrado na figura 01, o trabalho consistirá de duas partes distintas:

- Parte 01 – como fase inicial do compilador, o aluno deverá implementar o analisador léxico juntamente como analisador sintático, usando como base, o autômato descrito pela figura 02. O programa deverá ler um arquivo em Linguagem “Portugol” (.por) e realizar as operações de análise. Em pontos estratégicos, o programa deve executar uma das regras semânticas descritas no item 5 desse documento.
- Parte 02 – após a realização da análise léxica, sintática e semântica, o programa deve chamar o tradutor, a fim de converter as entradas da Linguagem “Portugol” definida, para a linguagem C. Uma vez gerado um arquivo em Linguagem C, o compilador dessa linguagem poderá ser usado para teste (ou não – opcional), como forma de validar a tradução resultante.

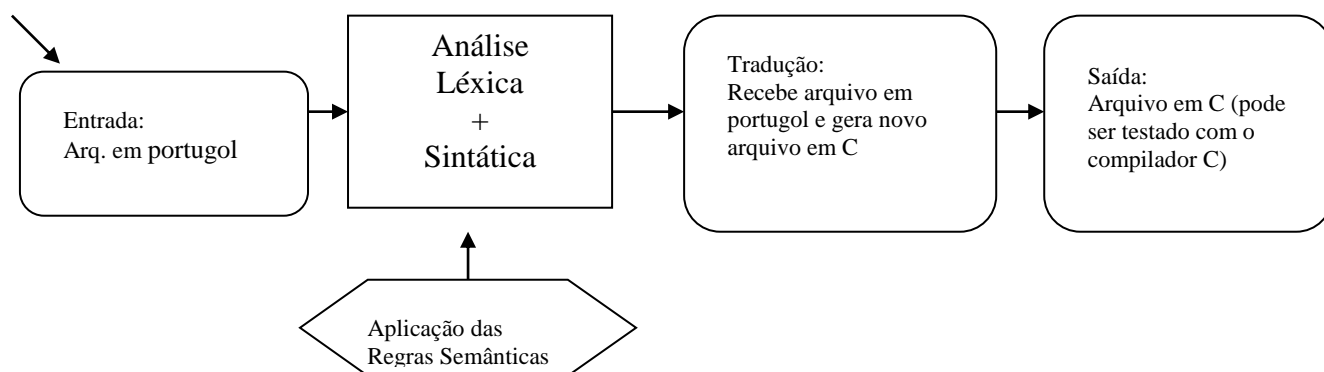


Figura 01 – Fases do Compilador

## 4. Informações sobre Grupos de Trabalho e Data Limite de Entrega:

Esse trabalho terá o valor total de 40% de N2. Os grupos poderão ter até 5 pessoas no máximo, com exceção no caso de faltarem integrantes para cumprir esse requisito.

Obs.: o programa deverá ser VISUAL, com menus de manipulação de arquivos, menu EDITAR com funções de cópia e cola, tela de editor, tela de mensagens de erros após o processo de compilação.

A data final de entrega será dia **08/12/2017**, impreterivelmente.

## 5. Tratamento de erros e considerações de implementação:

O analisador léxico deverá tratar erros léxicos, parando a compilação sempre que encontrar um erro. A seguinte tabela de erros é sugerida para o analisador léxico:

Estado 0: "Esperado Id=programa"
Estado 1: " Esperado Id=var"
Estados 2,3: " Esperado Id=;"
Estado 4: " Esperado Id=leia"
Estados 5, 11, 16, 22: " Esperado uma variável"
Estado 6: "Esperado Id=leia ou escreva"
Estado 7: "Esperado Id=( ou uma variável"
Estados 8,10: "Esperado Id=at ou leia"
Estados 9, 31: -- sem mensagem - visto que qualquer texto é aceito
Estados 12, 23: "Esperado Id= ="
Estados 13, 18, 24: "Esperado um numero inteiro ou uma variável"
Estados 14, 15: "Esperado Id= se"
Estado 17: "Esperado Operadores (<,>,<=,>=,==,!=) "
Estados 19, 20: "Esperado Id= entao"
Estado 21: "Esperado Id=at"
Estados 25, 27: "Esperado Operacoes (+,-,*) ou ;"
Estado 28: "Esperado Id=senao"
Estado 29: "Esperado Id=escreva"
Estado 30: "Esperado Id=("
Estado 32: "Esperado Id=fim"

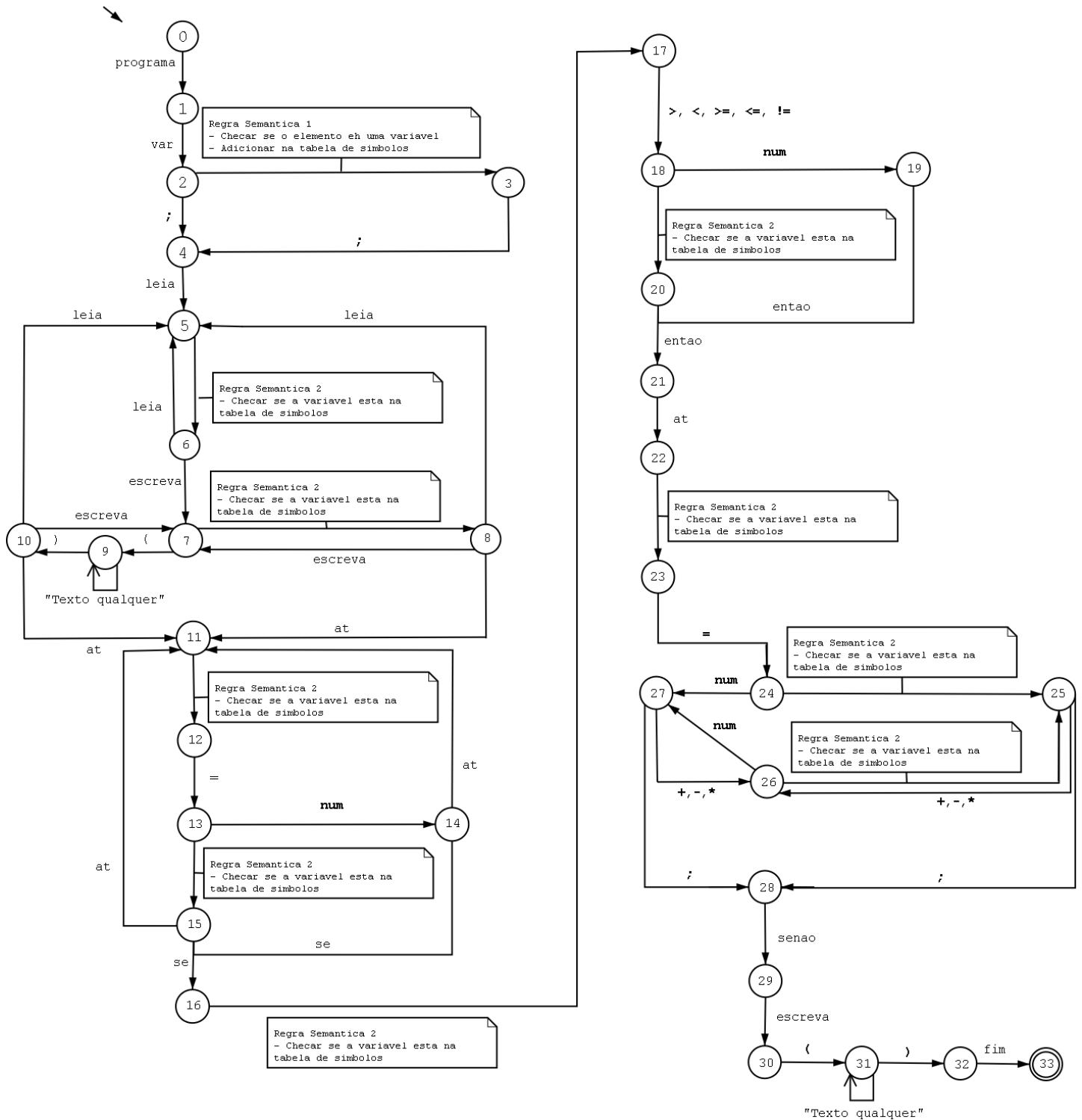


Figura 02 – autômato para analisador léxico-sintático

O erro do analisador sintático resume-se somente na questão de qual estado o autômato atinge após ler o arquivo: se for estado final, não emite-se um erro, senão:

```
"Erro sintatico: Programa invalido.\n"
```

Nos momentos indicados pelo autômato, o analisador semântico deverá ser acionado. São definidas as seguintes regras semânticas:

- Regra semântica 1 - é uma variavel que termina em \$ e começa por uma letra [a..z] ou [A..Z]?. Se sim, guarde-a na tabela de simbolos. Se não, emite o erro “variável inválida”
- Regra semântica 2 - palavra esta na tabela de símbolos? Se sim, continue a análise. Se não, emite-se o erro “Variável não declarada”.
- Regra semântica opcional (regra zero):
  - A) Checar sempre se o tamanho da palavra que indica um nome de variável é maior que 255 caracteres.
    - Erro relacionado: “Semantic Error %d: Variavel %d com nome muito grande. (MAX=%d)\n”
  - B) Checar se a tabela de símbolos está cheia.
    - Erro relacionado: “Semantic Error %d: Tabela de Simbolos cheia. Numero de Entradas=%d variavel\n”

Declaração de constantes utilizadas (sugestão):

```
define REGRA0 999 //Define regra semantica 0 - valor 999 - Tabela de Simbolos cheia
#define REGRA00 998 //Define regra semantica 0 - valor 998 - Tamanho de uma variavel
(evita buffer overflow)
#define REGRA1 100 //Define regra semantica 1 - valor 100 - Se a palavra eh uma variavel
valida na linguagem
#define REGRA2 200 //Define regra semantica 2 - valor 200 - Se a variavel foi declarada -
pertence aa tabela de simbolos
#define VARIAVEISOK 0 //Analisador semantico nao acusou erros.
#define TAMANHOTABELA 1000 //Quantidade de variaveis suportadas pela linguagem
#define TAMANHOVARIABEL 255 //Tamanho de um identificador de uma variavel
#define IDENCONTRADO 0 //Variavel encontrada na tabela de simbolos
#define IDNAOENCONTRADO 1 //Variavel encontrada na tabela de simbolos
#define OPERANDOOK 0 // Verifica se a palavra de entrada eh um operando: < | > | = | != |
<= | >=
#define OPERACAOOK 0 // Verifica se a palavra de entrada eh uma operacao: + | - | *
#define NUMOK 0 // Verifica se a palavra de entrada eh um numero: 1|2|3|4|5|6|7|8|9|0
#define ESTADOFINAL 33 // Verifica se a palavra de entrada eh uma operacao: + | - | *
#define LETRA 2 //Define que uma var sempre começa com uma letra

#define OFF 0 // Modo debug on/off
#define ON 1
```

## 6. Exemplos de Traduções:

Programa  var cont1\$ cont2\$ ;  leia cont1\$ leia cont2\$ escreva ( teste ) leia cont2\$ escreva ( teste2 ) leia cont2\$ escreva cont2\$ at cont1\$ = cont1\$ at cont2\$ = cont1\$ se cont1\$ >= cont2\$ entao at cont1\$ = cont2\$ + 10 * 8909 - cont2\$ ; senao escreva ( erro ) fim	#include <stdio.h>  int cont1,cont2;  int main () { scanf("%d",cont1); scanf("%d",cont2); printf("teste \n"); scanf("%d",cont2); printf("teste2 \n"); scanf("%d",cont2); printf("%d\n",cont2); cont1=cont1; cont2=cont1; if (cont1>=cont2) { cont1=cont2+10*8909-cont2; } else printf("erro \n"); exit(0); }
---	--

Programa  var cont1\$ cont2\$ ;  leia cont1\$ escreva ( primeiro programa em linguagem portugol ) leia cont2\$ escreva cont2\$ at cont1\$ = cont1\$ at cont2\$ = 1200 se cont1\$ < 2000 entao at cont1\$ = cont2\$ + 10 * 8909 - cont2\$ ; senao escreva ( erro ) fim	#include <stdio.h>  int cont1,cont2;  int main () { scanf("%d",cont1); printf("primeiro programa em linguagem portugol \n"); scanf("%d",cont2); printf("%d\n",cont2); cont1=cont1; cont2=1200; if (cont1<2000) { cont1=cont2+10*8909-cont2; } else printf("erro \n"); exit(0); }
---	---

Obs.: apesar de os espaços em branco serem desconsiderados pelos analisadores léxicos e sintáticos, utilize-os para separarem as palavras. Fica bem mais fácil de se implementar.

Ex.: cont1; é diferente de cont1 ;