INF1301 Programação Modular
Período: 2018-2
Prof. Flavio Bevilacqua

3o. Trabalho

Data de divulgação: 03 de Outubro de 2018

Data de entrega: 05 de Novembro de 2018

1. Descrição do trabalho do semestre

Neste semestre desenvolveremos um analisador léxico genérico. A análise léxica pode ser realizada por um autômato finito. A figura a seguir ilustra um autômato reconhecedor léxico.

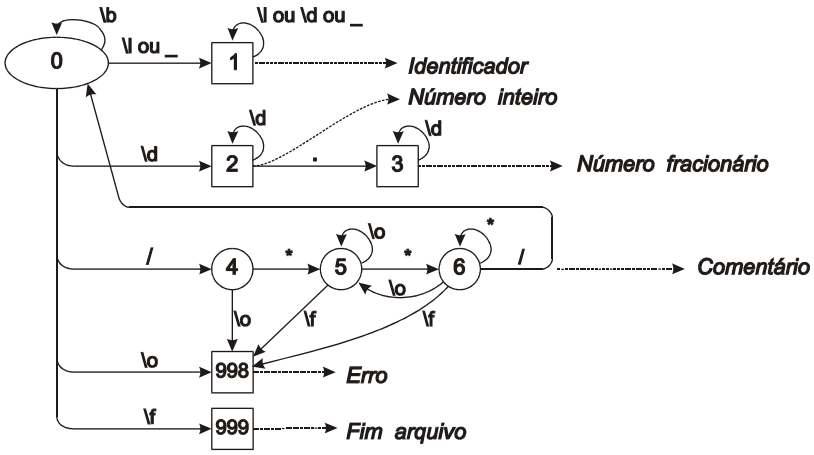

Figura 1. Autômato reconhecedor de alguns lexemas da linguagem C

O autômato é um ***grafo dirigido***. Os vértices são os estados, as arestas são as transições, o estado inicial (na figura o estado 0) é a origem. O autômato lê caracteres de um ***fluxo de entrada***. Este pode ser um *string* em memória ou um arquivo seqüencial.

As transições contêm um rótulo. Este rótulo corresponde ao caractere a ser lido do fluxo de entrada para que se prossiga pela correspondente transição. Por exemplo, no estado 4 prossegue-se para o estado 5 caso o fluxo de entrada contenha um caractere asterisco. A cada vez que se transita por uma aresta, o correspondente caractere do fluxo de entrada é consumido.

Estados podem ser de dois tipos: ***estado final*** e ***estado intermediário***. Os estados finais são representados como quadrados ou retângulos (exemplos: 1, 2 e 3) e os estados intermediários como círculos ou elipses (exemplos 0, 4 e 5). Embora o nome sugira o término do reconhecimento, é permitido continuar a analisar a partir de um estado final. Por exemplo, no estado final 2 pode-se continuar para o estado final 3 caso a entrada contenha um caractere ponto. Os estados finais identificam o lexema reconhecido.

Os rótulos das transições podem ser caracteres simples ou conjuntos de caracteres. Quando se trata de um caractere simples o fluxo de entrada deve conter exatamente aquele caractere para que se possa prosseguir pela correspondente transição. São definidos alguns caracteres especiais:

\f - “caractere” que corresponde ao fim do fluxo de entrada

\n - caractere que corresponde a nova linha

\r - caractere que corresponde a retrocesso de carro sem avanço de linha. Houve uma época em que se utilizava máquina de escrever ☺

\t - caractere que corresponde a uma tabulação

Quando o rótulo designa um conjunto, a transição ocorre se o fluxo de entrada contiver qualquer um dos caracteres desse conjunto. São definidos alguns conjuntos:

\l - o conjunto de todas as letras.

\d - o conjunto dos dígitos decimais.

\b - o conjunto dos caracteres de espaçamento: caractere espaço, tabulação, nova linha e retrocesso de carro, ou seja: { ' ', '\t', '\n', '\r' }.

\o - o conjunto de todos os caracteres que não ocorrem nas demais transições que emanam de um determinado estado. No máximo uma transição por estado pode ser rotulada com esse conjunto.

Cada vez que se chama o reconhecedor, ele iniciará a leitura no estado inicial e prosseguirá até chegar a um estado a partir do qual ele não mais poderá transitar com o caractere disponível no fluxo naquele momento. Se este estado for terminal, o autômato terá reconhecido o correspondente lexema. Se o estado não for terminal, precisa-se *retroceder sobre o caminho percorrido*, até que se chegue ou a um estado final ou ao estado inicial. Os caracteres do fluxo percorridos nesse trajeto inverso devem ser empilhados na pilha de releitura. Ao ler caracteres na próxima ativação do reconhecedor, primeiro serão desempilhados esses caracteres e, depois, quando a pilha de releitura estiver vazia, volta-se a ler do fluxo de entrada. Caso, ao retroceder, tenha-se chegado a um estado final, esse corresponderá ao lexema encontrado. Caso, ao retroceder, tenha-se chegado ao estado inicial, encontrou-se um erro de dados de entrada, mais especificamente, o caractere no topo da pilha de releitura não leva a um estado final. Neste caso elimina-se o caractere no topo da pilha, emite-se uma mensagem de erro e reinicia-se o reconhecimento.

2. Implementação do reconhecedor

O modelo físico do grafo reconhecedor léxico pode ser visto na figura 2. Este modelo é uma aproximação. Os hexágonos batizados de Lista correspondem ao uso do módulo Lista. Neste módulo os elementos da lista apontam para um valor. Esse ponteiro é o que emana do hexágono Lista. Evidentemente o modelo precisa ser melhorado, em particular no que toca a implementação de um grafo genérico.

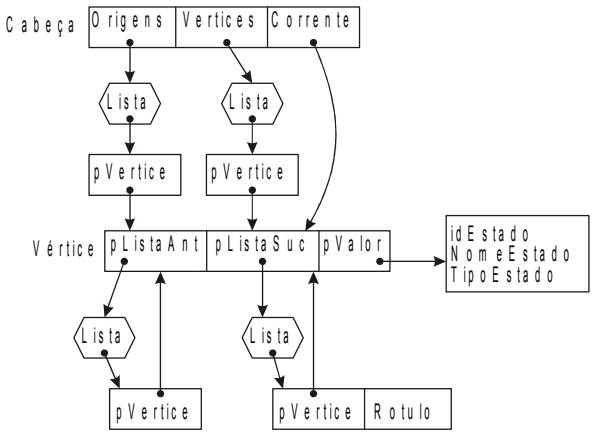

Figura 2. Modelo físico de um grafo dirigido

Os vértices, ou seja, estados, precisam identificar o estado (ex. o número do estado na figura 1), indicar o nome do estado (ex. o nome do que foi reconhecido no caso de um estado final) e indicar o tipo do estado (ex. estado final, estado intermediários). As arestas, ou seja, transições, precisam informar o Rótulo da transição. O rótulo pode ser um string que indica qual deve ser o caractere de entrada para que se possa transitar por aquela aresta (ex. na aresta de 0 para 4 `"/"`; na aresta de 1 para 1 `"\l ou \d ou _"`).

Devem ser criados dois módulos: o **GrafoGenerico** e o **ReconhecedorLexico**. Veja a figura 3. Os ponteiros que emanam do grafo genérico são um ponteiro **void** para o valor do vértice, i.e. o estado do reconhecedor, e outro ponteiro **void** para o rótulo da aresta. Este último deverá existir tanto para as arestas direcionadas para vértices antecessores como para vértices sucessores. No caso do reconhecedor, as arestas antecessores receberão um **NULL** e as sucessoras apontarão para um **Rotulo**. De ponto de vista generalidade é conveniente que os elementos das listas de origens e de vértices também possam apontar para algum valor. No caso do reconhecedor esse valor será **NULL**.

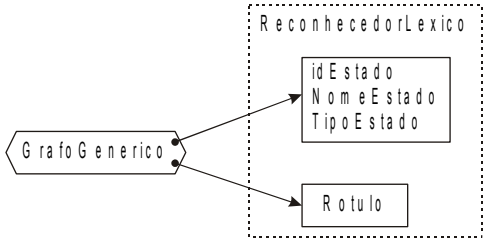

Figura 3. Modelo do reconhecedor léxico implementado com um grafo genérico

3. Descrição do terceiro trabalho

Neste trabalho deverá ser revisado o módulo **GRAFO** genérico do segundo trabalho. Ele será testado em detalhe no quarto trabalho. O grafo genérico deve ser utilizado para implementar o autômato reconhecedor. O resultado do trabalho será um programa capaz de ler os estados e arestas do reconhecedor e, a seguir, realizar o reconhecimento a partir de um arquivo **.txt**. Para simplificar o trabalho, sugere-se que o módulo **ReconhecedorLexico** seja uma adaptação do módulo de teste específico do grafo genérico.

O resultado do reconhecimento a ser exibido deve ser um texto em que cada linha informa a *linha* e a *coluna* do texto fonte em que inicia o lexema, o *string* do *lexema* reconhecido, o *id* e o *nome* do estado final.

Deve ser criada uma biblioteca estática: **MinhaBiblioteca.lib**. Esta biblioteca conterá o módulo **GrafoGenerico**, o módulo **Lista**, e quaisquer outros módulos que você considere necessários. Mas **NÃO** deve conter o módulo **ReconhecedorLexico**.

Procure seguir o seguinte roteiro para realizar o trabalho:

1. Se não o fez no primeiro trabalho, crie um diretório para elaborar o projeto do autômato. Baseie-se na estrutura de diretórios do exemplo **Instrum** contido no pacote do arcabouço de apoio ao teste. Ajuste os arquivos **.comp** e **.bat** para que tudo funcione a contento.
2. Evolua o módulo **GRAFO** do segundo trabalho de modo que se torne efetivamente genérico, conforme explicado acima.
3. Crie a biblioteca estática **MinhaBiblioteca.lib**.
4. Teste a biblioteca usando o módulo de teste específico do 2º. trabalho, adaptando-o se necessário.
5. Implemente o módulo **ReconhecedorLéxico**.
6. Redija alguns autômatos e teste cada um deles com alguns textos.

4. Entrega do Trabalho

O trabalho deve ser feito em grupos de dois ou três alunos. Os programas devem ser redigidos em "**C**". Não será aceita nenhuma outra linguagem de programação. Todos os programas devem estar em conformidade com os padrões dos apêndices de 1 a 10 do livro-texto da disciplina. Em particular, os módulos e funções devem estar devidamente especificados.

O trabalho deve ser enviado por e-mail em um único arquivo **`.zip`** (codificação do *attachment*: **`MIME`**). Veja os *Critérios de Correção de Trabalhos* contidos na página da disciplina para uma explicação de como entregar.

O arquivo **`.zip`** deverá conter:

- Os arquivos fonte dos diversos módulos que compõem os programas, inclusive os módulos de definição do arcabouço porventura utilizados.
- Os arquivos que descrevem os autômatos e os arquivos que correspondem ao texto analisado.
- Os arquivos *batch* (**`.bat`**) que coordenam a execução dos testes. (veja o exemplo **`testatudo.bat`** no arcabouço). Os arquivos **`.bat`** devem poder operar sem requerer modificações, independentemente de onde seja instalado o trabalho.
- A biblioteca **`MinhaBiblioteca.lib`**
- Os diversos programas executáveis (construtos): **`TRAB1-1.EXE`**, **`TRAB1-2.EXE`**, e assim por diante.
- Um arquivo **`LEIAME.TXT`** contendo a explicação de como utilizar o(s) programa(s).
- Tantos arquivos **`RELATORIO-nome.TXT`** quantos forem os membros do grupo. O tema **`nome`** deve identificar o membro do grupo ao qual se refere o relatório. Estes arquivos devem conter uma tabela de registro de trabalho organizada como a seguir:

Data Horas Trabalhadas, Tipo Tarefa, Descrição da Tarefa Realizada

Na descrição da tarefa redija uma explicação breve sobre o que o componente do grupo fez. Esta descrição deve estar de acordo com o Tipo Tarefa. Cada Tipo Tarefa identifica uma natureza de atividade que deverá ser discriminada explicitamente, mesmo que, durante uma mesma sessão de trabalho tenham sido realizadas diversas tarefas. Os tipos de tarefa são:

- ♦ estudar
- ♦ especificar os módulos
- ♦ especificar as funções
- ♦ revisar especificações
- ♦ projetar
- ♦ revisar projetos
- ♦ codificar módulo
- ♦ revisar código do módulo
- ♦ redigir script de teste
- ♦ revisar script de teste
- ♦ realizar os testes
- ♦ diagnosticar e corrigir os problemas encontrados

Observações:

- **Dica**: Preencha esta tabela de atividades ao longo do processo. NÃO DEIXE PARA ÚLTIMA HORA, POIS VOCÊ NÃO SE LEMBRARÁ DO QUE FEZ TAL DIA, TAL HORA. Com relatórios similares a esse você aprende a planejar o seu trabalho.
- **Importante**: O arquivo **`ZIP`**, DEVERÁ CONTER SOMENTE OS ARQUIVOS RELACIONADOS A ESTE TRABALHO. Gaste um pouco de tempo criando um *diretório de distribuição* e um **`.bat`** que copia do *diretório de desenvolvimento* para este diretório de distribuição somente os arquivos que interessam. Verifique se esta cópia está realmente completa!
- A mensagem de encaminhamento deve ter o assunto (*subject*) **`INF1301-Trab01-idGrupo`**. O tema **`idGrupo`** deve ser formado pelas iniciais dos nomes dos membros do grupo. O texto da mensagem deve conter somente a lista de alunos que compõem o grupo (formato: número de matrícula, nome e endereço do e-mail). **Perde-se 2 pontos caso não seja encaminhado**

desta forma. Mais detalhes podem ser encontrados no documento *Critérios de Correção dos Trabalhos* disponível na página da disciplina.

- O programa será testado utilizando o programa compilado fornecido. Deve rodar sem requerer bibliotecas ou programas complementares. O sistema operacional utilizado durante os testes será o **`Windows 10`**. Assegure-se que a versão do programa entregue é uma versão de produção, ou seja, sem dados e controles requeridos pelo *debugger*.

5. Critérios de correção básicos

Leia atentamente o documento *Critérios de Correção dos Trabalhos* disponível na página da disciplina. Muitas das causas para a perda substancial de pontos decorrem meramente da falta de cuidado ao entregar o trabalho.

Não deixem para a última hora.
Este trabalho dá trabalho!