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

2o. Trabalho

Data de divulgação: 05 de Setembro de 2018

Data de entrega: 03 de Outubro 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.

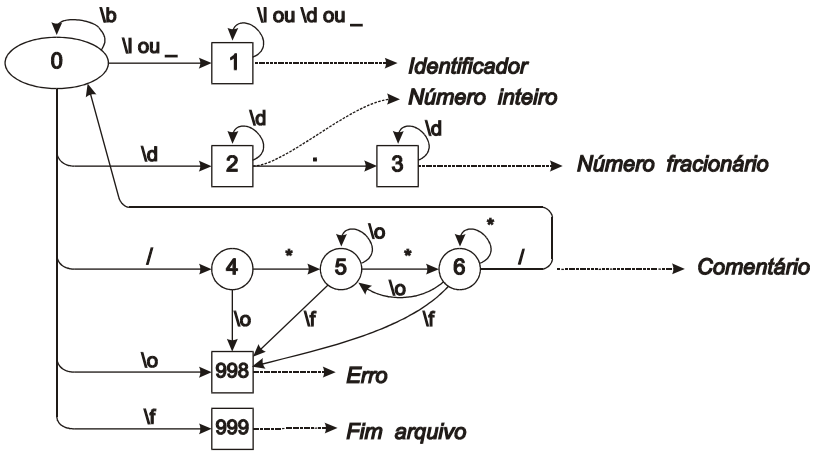

Figura 1. Autômato reconhecedor de alguns lexemas da linguagem C

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. 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.

O modelo básico de um grafo dirigido implementado usando listas de arestas pode ser visto na figura 2. Este modelo evidentemente precisa ser adaptado para as necessidades específicas do autômato.

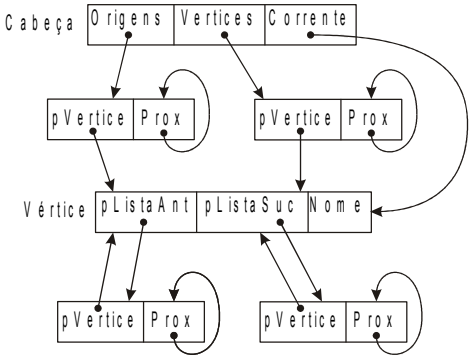

Figura 2. Modelo físico de um grafo dirigido

Os vértices, ou seja, estados, precisam identificar o nome e precisam informar o tipo do estado, se é final ou intermediário. As arestas, ou seja, transições, precisam informar o tipo de comparação a usar, pesquisa em um conjunto, ou comparação do caractere. E precisam informar também, o conjunto no qual se procurará o caractere ou o caractere específico a comparar.

2. Descrição do segundo trabalho

Neste trabalho deverá ser criado o módulo **GRAFO** genérico. Os vértices deste grafo apontam para valores do tipo **void**. Os elementos das listas de antecessores e sucessores também apontam para valores do tipo **void**. Junto com o trabalho é fornecido um módulo **LISTA** que implementa uma lista genérica. Este módulo deve ser utilizado neste trabalho. Em princípio este módulo não necessitará de qualquer adaptação. Serve também para ilustrar como implementar uma estrutura genérica. Outro

exemplo é o módulo **TABSIMB** que faz parte do arcabouço de teste. Se for desejado, este módulo pode ser utilizado para procurar vértices pelo nome.

Procure seguir o seguinte roteiro para realizar o trabalho:

1. Crie um diretório para elaborar o projeto do autômato. Siga a estrutura de diretórios do exemplo **Instrum** contido no pacote do arcabouço de apoio ao teste.
2. Expanda o módulo **LISTA** e distribua os seus elementos sobre a estrutura de diretório do autômato segundo o padrão de estrutura de diretórios usado.
3. Faça todos os ajustes de modo que o módulo **LISTA** compile e teste corretamente tal como fornecido.
4. Projete o módulo **GRAFO**. Reformule o modelo da figura 2 e redija ***todas*** as assertivas estruturais. O modelo e as assertivas deverão constar do arquivo **zip** de entrega do trabalho e deverão estar em conformidade com o grafo utilizado para teste do módulo.
5. Implemente e teste o módulo **VERTICE**. Para efeito de teste, os vértices devem referenciar *strings*.
6. Implemente e teste o módulo **GRAFO** compondo-o com os módulos **VERTICE** e **LISTA**. Para efeito de teste as listas de sucessores devem referenciar rótulos do tipo *string*, tal como está no módulo de teste fornecido junto com o módulo **LISTA**. As listas de antecessores não devem referenciar rótulos.

3. 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.

Recomenda-se fortemente a leitura do exemplo e do texto explanatório do arcabouço. Além de mostrar como implementar um teste automatizado dirigido por *script*, ilustra também as características de um programa desenvolvido conforme os padrões do livro.

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.
- Os arquivos de *script* de teste desenvolvidos pelo grupo.
- Os arquivos *batch* (**.bat**) que coordenam a execução dos testes. (veja o exemplo **testatudo.bat** no arcabouço).
- Os diversos programas executáveis (construtos): **TRAB1-1.EXE**, **TRAB1-2.EXE**, **TRAB1-3.EXE**, e assim por diante.
- O modelo físico do grafo e as correspondentes assertivas estruturais.
- 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. Caso o arquivo enviado contenha outros arquivos que os acima enumerados (por exemplo: toda pasta do arcabouço, arquivos **`.bak`**, arquivos de trabalho criados pelo ambiente de desenvolvimento usado, etc.) o grupo **perderá 2 pontos**. 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*.

4. 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!