UNIOESTE – Universidade Estadual do Oeste do Paraná
Curso de Ciência da Computação
Disciplina de Compiladores
Prof. Newton Spolaôr

Projeto 1.2 – Análise léxica e sintática – valor: 100 pontos

O prazo (*deadline*) de entrega é no sábado 31/07/2021, 23:59 hs (horário de Brasília). A resolução deve ser enviada anexada em *chat* privado com o professor no Microsoft Teams. Todos os arquivos da resolução devem ser comprimidos (usando software Winrar ou similar) e enviados em um único anexo.

Observações:

- O projeto é em grupo.
- O projeto atual será incrementado ao longo da disciplina com mais analisadores inerentes a um compilador. Logo, para maior produtividade, a linguagem de programação a ser reconhecida pelo seu compilador agora deveria ser a mesma nos próximos projetos.
- Em caso de identificação de plágio, a nota 0 será atribuída para todos os envolvidos.
- Cada dia de atraso na entrega implica na perda de 50 pontos.
- A apresentação do projeto fora do dia destinado para o grupo, sem justificativa adequada, implica na perda de 25 pontos.
- O professor está à disposição (por *chat* ou email) para esclarecer dúvidas sobre programação, entre outros conceitos relacionados ao trabalho.
- Para cada resposta de questão escrita com apoio de uma fonte (exemplo: sites, artigos, livros...) que não sejam os slides da aula, inclua referência para essa fonte no final da resposta. No caso de falta dessa referência, há perda de 20 pontos.
- Parte da nota se destina a um software gerado por linguagem de programação e sua execução (vide critérios de avaliação após ler definições que seguem).

Definições:

1. Equipes de até dois acadêmicos (preferencialmente 2 acadêmicos) que irão desenvolver um software, elaborar sua documentação e apresentá-lo em uma aula, na forma de um seminário curto.
2. Criar uma gramática com no mínimo **26** regras sintáticas, bem como expressões regulares ou autômatos finitos para reconhecer no mínimo **12** classes de *tokens*, para uma linguagem de programação procedural simplificada, não orientada a objetos. Essa linguagem, denominada X, pode ser um subconjunto de C, Pascal, SQL, sh **ou outra linguagem já existente**. Obs.: **não é possível misturar linguagens para formar X**.
3. Implementar um software que simula um compilador capaz de realizar as análises léxica e sintática:
    - Ao ser executado, pedir para o usuário digitar ou selecionar o nome do arquivo fonte (Ex: fonte1.txt, fonte2.txt, fonte3.txt);
    - Ler o código fonte a partir do arquivo selecionado;
    - Reconhecer classes de *tokens* no código fonte por meio da análise léxica;
    - Reconhecer comandos (sequências de *tokens*) no código fonte por meio de análise sintática;

- Realizar tratamento de erro léxico e de erro sintático, mostrando na tela qual o tipo de erro (léxico ou sintático) e apontando a posição do erro. Para ter uma maior cobertura de erros diferentes, é útil pesquisar os tipos de erros léxicos e sintáticos mais comuns em linguagens populares;
- Se o código fonte não contém nenhum erro de compilação, para pelo menos 5 comandos desse código deve-se gerar a árvore sintática correspondente a cada comando, de modo similar a uma árvore de diretórios. As árvores resultantes devem ser escritas em um arquivo texto. Por exemplo, para o arquivo fonte1.txt, sem erros de compilação, deve-se gravar as árvores correspondentes no arquivo denominado arvores_fonte1.txt;
- O software deve ser implementado em qualquer linguagem, com ou sem o apoio de ferramentas auxiliares como Lex, Yacc e Javacc.

4. O projeto deve ser melhorado com as sugestões dadas durante os seminários anteriores e a resolução de *bugs* e problemas do projeto anterior.
5. Criar um arquivo README.doc contendo as seguintes informações:
   - Nome do software desenvolvido;
   - Nome dos integrantes da equipe;
   - Descrição da linguagem X;
   - Descrição de todas as expressões regulares ou autômatos usados para reconhecer *tokens*, bem como citação dos trechos do software desenvolvido associados especificamente a cada expressão regular/autômato;
   - Descrição da gramática, incluindo representação de todas as regras em notação EBNF ou notação equivalente usada nas aulas, bem como citação dos trechos do software desenvolvido associados especificamente a cada regra gramatical;
   - Descrição do funcionamento do software que realiza as análises léxica e sintática, incluindo informações para orientar o usuário a executar o software com autonomia;
   - Descrição da estratégia de tratamento de erros adotada e dos tipos de erros que podem ser tratados;
   - Descrição do processo de construção (*build*) do software desenvolvido, incluindo configurações, bibliotecas e ferramentas necessárias para compilar/interpretar esse software. Caso um *Integrated Development Environment* (IDE) tenha sido usado, ele deve também ser indicado. A versão de cada programa auxiliar citado neste tópico deve também ser informada;
   - Apresentação de uma das árvores sintáticas geradas pelo software;
   - Referências usadas.
6. A equipe deverá:
   - Documentar **manualmente e em Português** os principais métodos/funções do código fonte (ou as especificações Lex, Yacc e Javacc) do software que realiza as análises léxica e sintática. Essa documentação deve incluir uma breve descrição da finalidade de cada método e dos seus parâmetros;
   - Criar um exemplo correto (“sem erros de linguagem”) de código fonte chamado fonte1.txt;
   - Criar um arquivo incorreto (“com erros léxicos”) de código fonte chamado fonte2.txt.

- Criar um arquivo incorreto (“com erros léxicos e sintáticos”) de código fonte chamado fonte3.txt. É obrigatório constarem tanto erros léxicos quanto erros sintáticos.

7. Encaminhar todo o projeto, incluindo um **arquivo POWERPOINT ou similar** a ser usado pela equipe no seminário**,** o software que realiza as análises léxica e sintática com documentação, bibliotecas necessárias para o software, os arquivos README.doc, fonte1.txt, fonte2.txt e fonte3.txt, em um único arquivo comprimido para o *chat* do professor **dentro do prazo previamente informado**.
8. As apresentações dos trabalhos serão realizadas nos dias 2, 4 e 9/08/2021, podendo terminar antes do dia 9. **Quanto mais cedo a equipe entregar o trabalho completo, mais tarde fará a apresentação**. **Porém, uma equipe agendada para mais tarde pode apresentar mais cedo, se assim o desejar, desde que solicite ao professor com antecedência**. Todos os membros da equipe devem participar da apresentação.
9. **O software executado pela equipe no seminário deve ser o mesmo entregue dentro do prazo ao professor**. Cada apresentação deve durar até 5 minutos e deverá usar slides contendo:
   - Descrição breve da linguagem X e visão geral da gramática implementada;
   - Descrição de como cada membro da equipe contribuiu no desenvolvimento do trabalho;
   - No final da apresentação, deverá ser demonstrada a execução do software desenvolvido usando os arquivos de código fonte sem e com erro sintático (fonte1.txt e fonte3.txt), incluindo exibição de parte da árvore sintática para o código sem erro.
10. Após o encerramento do seminário, o professor pode fazer perguntas a cada membro da equipe sobre o que foi apresentado, incluindo conteúdo teórico e prático relacionado ao projeto. A equipe deverá ter disponível o código-fonte do software e demais materiais preparados como parte do projeto para esse momento.
11. A apresentação do seminário por meio de vídeo gravado é permitida, desde que a equipe de acadêmicos:
    a. Comunique ao professor com antecedência que adotará essa modalidade;
    b. Envie o vídeo da apresentação com antecedência, possibilitando ao professor baixar o vídeo antes das aulas e reproduzi-lo para a turma durante a aula síncrona;
    c. Esteja presente na aula para acompanhar o seminário, receber sugestões e responder eventuais perguntas por meio de microfone.
12. Critérios de avaliação:
    - Conhecimento demonstrado na apresentação: 30 pontos;
    - Software, conforme especificação do trabalho (programa, códigos-fonte, execução, documentação, README, respeito ao deadline): 40 pontos;
    - Comunicação utilizada na apresentação (clareza, objetividade, respeito ao limite de tempo): 20 pontos;
    - Qualidade dos slides: 10 pontos.
13. Referências recomendadas:
    - Bibliografia básica da disciplina, especialmente o livro voltado ao JavaCC;
    - *Websites* como o que segue: https://johnidm.gitbooks.io/compiladores-para-humanos/content/part1/syntax-analysis.html.