

Projeto 1.1 – Análise léxica – valor: 100 pontos

O prazo (*deadline*) de entrega é no sábado 03/07/2021, 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.
- O professor está à disposição (por *chat* e email) para esclarecer dúvidas sobre programação, entre outros conceitos relacionados ao trabalho.
- 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 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 a análise léxica:
 - Ao ser executado, pedir para o usuário digitar ou selecionar o nome do arquivo fonte (Ex: fonte1.txt, fonte2.txt);
 - Ler o código fonte a partir do arquivo selecionado;
 - Reconhecer classes de *tokens* no código fonte por meio da análise léxica;
 - Realizar tratamento de erro léxico, mostrando na tela qual o tipo de erro e apontando a posição do erro. Para ter uma maior cobertura de erros diferentes, é útil pesquisar os tipos de erros léxicos mais comuns em linguagens populares;
 - O software deve ser implementado em qualquer linguagem, com ou sem o apoio de ferramentas auxiliares como Lex, Yacc e Javacc.
4. 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 do funcionamento do software que realiza a análise léxica, 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;
 - Referências usadas.
5. 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 a análise léxica. 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 de linguagem”) de código fonte chamado fonte2.txt.
6. Encaminhar todo o projeto, incluindo um **arquivo POWERPOINT ou similar** a ser usado pela equipe no seminário, o software que realiza a análise léxica com documentação, bibliotecas necessárias para o software, os arquivos README.doc, fonte1.txt e fonte2.txt, em um único arquivo comprimido para o *chat* do professor **dentro do prazo previamente informado**.
7. As apresentações dos trabalhos serão realizadas nos dias 5, 7, 12 e 14/06/2021, podendo terminar antes do dia 14. **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.
8. **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 das expressões regulares ou autômatos implementados;
 - 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.
9. 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.

10. 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.
11. 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.
12. 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/lexical-analysis.html>.