Atividade Avaliativa – RA3

Este trabalho pode ser realizado em grupos de até 4 alunos. **Grupos com mais de 4 alunos irão provocar a anulação do trabalho.** Você deve ler todo documento antes de começar e considerar o seguinte código de ética: *você poderá discutir todas as questões com seus colegas de classe, professores e amigos. Poderá também consultar os livros de referência da disciplina, livros na biblioteca virtual ou não, e a internet de forma geral e abrangente nos idiomas que desejar. Contudo, o trabalho é seu e deverá ser realizado por você. Cópias ensejarão a anulação do trabalho.*

OBJETIVO

Pesquisar e praticar. Pesquisar os conteúdos que irão complementar o material apresentado em sala ou nos livros sugeridos na ementa e praticar estes mesmos conceitos. Esta é uma oportunidade para aprimorar sua formação e se destacar. Uma avaliação com oportunidade de crescimento acadêmico e profissional.

DESCRIÇÃO DO TRABALHO

Seu objetivo será desenvolver um programa, usando Python, C, ou C++, capaz de abrir um arquivo de texto, contendo expressões e declarações simples, com uma expressão por linha e criar os analisadores léxico e sintático para a linguagem explicitada neste documento. Lembre-se a linguagem que será descrita a seguir **está incompleta e é baseada na linguagem** que usamos na avaliação da RA-1. Mas, nesta fase, você não irá precisar gerar qualquer código em Assembly.

O programa que você irá desenvolver deverá ler o código em txt e gerar um *string* de *tokens* contendo, no mínimo, o valor de cada *token*, sua classe e a posição deste *token* no texto.

CARACTERÍSTICAS ESPECIAIS DA LINGUAGEM

As expressões serão escritas em notação RPN, segundo o formato a seguir:

a) Adição:+, no formato (A B +);
b) Subtração: - no formato (A B -) ;
c) Multiplicação: * no formato (A B *);
d) Divisão Real: | no formato (A B |);
e) Divisão de inteiros: / no formato (A B /);
f) Resto da Divisão de Inteiros: % no formato (A B %);
g) Potenciação: ^ no formato (A B ^);

Neste caso, A e B representam números reais. **Use o ponto para indicar a vírgula decimal.** As sintaxes definidas nas alíneas a até g definem as operações binárias desta linguagem.

Todas as operações serão executadas sobre números reais de meia precisão (16 bits/ IEEE754) CASO VOCÊ USE UM MICROPROCESSADOR DE 16 BITS, DEVE USAR PRECISÃO SIMPLES (32 bits/ IEEE754). CASO VOCÊ USE UM MICROPROCESSADOR DE 32 BITS DEVE USAR PRECISÃO DUPLA (64 bits/ IEEE754). CASO VOCÊ USE UM MICROPROCESSADOR DE 64 BITS DEVE USAR PRECISÃO QUADRUDUPLA (128 bits/ IEEE754). As únicas exceções são a divisão de inteiros e a operação resto da divisão de inteiros que devem ser realizadas em números inteiros. Além disso, **O expoente da operação de potenciação será sempre um inteiro positivo**. Todas as expressões desta linguagem podem ser aninhadas para a criação de expressões compostas, sem nenhum limite definido de aninhamento. Isto implica na possibilidade de que o texto lido contenha linhas com expressões como:

a) (A (C D *) +)
b) ((A B %) (D E *) /)

Onde A, B, C, D e E representam números reais. Além das expressões já definidas existem três comandos especiais (N RES), (V MEM) e (MEM), de tal forma que:

a) (N RES): devolve o resultado da expressão que está N linhas antes, onde N é um número inteiro não negativo;
b) (V MEM): armazena um valor, V, em um espaço de memória chamado de MEM, capaz de armazenar um valor real;
c) (MEM): devolve o valor armazenado na memória. Se a memória não tiver sido usada anteriormente devolve o número real zero. Cada arquivo de textos é um escopo de aplicação.

Além disso, mantendo o uso de parênteses, na RA2 você criou a sintaxe para os artefatos *if...then...else* e *for* (tomada de decisão e laço de repetição) comuns na maior parte das linguagens de programação. E, certamente, deve ter entendido que a recursão é uma opção aos laços de repetição. Ainda que seja uma opção computacionalmente cara.

Os textos de teste deverão conter valores literais inteiros e reais.

Para teste do seu trabalho, você deverá fornecer, no mínimo 3 arquivos de texto contendo, no mínimo 3 linhas com expressões aritméticas e artefatos que você criou. Observe que agora, você pode criar programas. Para calcular o fatorial de um número, ou um determinado item da sequência de Fibonacci etc. Estes arquivos de teste deverão estar disponíveis no mesmo ambiente online, e no mesmo diretório do código fonte do seu programa. Ou no seu computador pessoal, desde que todo o código esteja postado em um repositório do GitHub, documentando e comentado em inglês. **Você precisará postar um link para este repositório no canvas para formalizar a entrega.**

Os arquivos de testes devem incluir situações de erro, relatórios de correção, ou a explicitação do erro para que o usuário possa corrigir o código.

Seu programa, o analisador sintático, não pode ter um menu, ou qualquer forma de seleção entre os seus arquivos de teste. O programa, o analisador léxico, deverá ser executado recebendo como argumento, na linha de comando, o nome do arquivo de teste.

Atenção: as primeiras linhas do seu código devem conter os nomes dos integrantes do grupo, em ordem alfabética, além do nome do grupo no ambiente virtual de aprendizagem (Canvas). A nota será lançada para o grupo.

TECNOLOGIAS ESSENCIAIS

Para fazer jus aos pontos referentes a este trabalho você deverá criar um analisador sintático baseado em um parser LL(1). A documentação do seu trabalho deverá incluir a gramática que você criou para a sua linguagem, os conjuntos First e Follow e a tabela de derivação. **Lembre-se, sua linguagem inclui as operações que definimos no começo do trabalho e os artefatos que você criou para tomada de decisão e repetição.** Além disso, o seu analisador sintático deve criar uma árvore sintática abstrata na qual as folhas serão os tokens criados pelo seu avaliador léxico. Esta árvore sintática pode ser criada em txt, diretamente no terminal, em imagem, em pdf, ou com qualquer outra tecnologia que o grupo escolher. Com uma única condição: deve ser possível visualizar a árvore na sua forma canônica: raiz, galhos, vértices e folhas estruturadas e visíveis.

ENTREGAS, AVALIAÇÃO e PROVA DE AUTORIA

Você deverá postar no Ambiente Virtual de Aprendizado da Instituição (Canvas) o link para o repositório do GitHub que contenha todo o seu código e toda a sua documentação.

Usando algum aplicativo gratuito, disponível online, será sorteado um dos alunos do grupo para representar o grupo na prova de autoria. Este aluno sorteado escolherá um número entre 1 e 10. O número escolhido corresponde a pergunta que o aluno deverá responder. Caso o aluno não responda o grupo perderá 35% da nota atribuída ao projeto. **AVALIAÇÃO, SERÁ REALIZADA DURANTE A PROVA DE AUTORIA**

O trabalho será avaliado quanto à:

1. Entrega da árvore sintática até **70% da nota total**;
    a) Cada operação não identificada resultará na perda de **10% destes 70%;**
    b) Falha nas definições dos laços de repetição, perda de **10% destes 70%;**
    c) Falha nas definições da tomada de decisão, perda de **10% destes 70%;**

d) A da gramática implicam na perda de **50% destes 70%;**
e) A falta dos conjuntos First e Follow implicam na perda de **25% destes 70%;**
f) A falta da tabela de derivação implica na perda de **25% destes 70%;**
g) Apenas números inteiros resulta na perda de **50% destes 70%**.

2. Organização e legibilidade do código **– 15%**;
3. Robustez diante de casos complexos ou com erros **– 15%**;
4. Trabalhos iguais provocam o zeramento de todos os trabalhos identificados como iguais. Cabe ao professor definir o quanto um trabalho é igual ao outro.