CONSTRUÇÃO DE COMPILADORES I

Checkpoint 01: Analisador Léxico

Esta atividade pode ser feita em grupo (até 2 integrantes), e consiste em fazer alterações no código do analisador léxico feito em sala de aula, disponível no seguinte repositório do gitlab https://gitlab.com/maelso/mini_compiler, na branch master:

Obs. O aluno pode escolher implementar a solução em Java, Python ou em C/C++. Porém, o analisador léxico deve possuir sua estrutura semelhante à que foi mostrada em sala de aula. Sendo proibido o uso de ferramentas automatizadoras de reconhecimento.

Tendo como base o código implementado em sala e enviado para a branch acima mencionada, faça as seguintes alterações:

1. (1 ponto) Um identificador pode conter, também, o caractere *underscore* (_), ou seja:
   a) (a-z | A-Z | _)(a-z | A-Z | _ | 0-9)*
2. (1 ponto) Adicione suporte ao analisador léxico para reconhecer os seguintes operadores matemáticos:
   a) soma(+)
   b) subtração (-)
   c) multiplicação (*)
   d) divisão(/).
3. (1 ponto) Adicione suporte ao analisador léxico para reconhecer o operador de atribuição (=).
4. (1 pontos) Adicione suporte ao analisador léxico para reconhecer os seguintes operadores relacionais:
   a) maior que (>)
   b) maior ou igual que (>=)
   c) menor que (<)
   d) menor ou igual que (<=)
   e) diferente (!=)
5. (1 ponto) Adicione suporte ao analisador léxico para reconhecer:
   a) parêntese esquerdo (‘(’)
   b) parêntese direito (‘)’)
6. (1 ponto) Adicione suporte no analisador léxico para reconhecer constantes numéricas com ponto decimal, ou seja, devem ser aceitos números do tipo 123 e 123.456 e .456, conforme a seguinte expressão regular:
   a) ((0-9)*.)?(0-9)+
   (*obs.: números do tipo 1. ou 12. ou 156. devem ser rejeitados*)
7. (1.5 pontos) Adicione suporte no analisador léxico para reconhecer as seguintes palavras reservadas:
   a) int
   b) float
   c) print
   d) if
   e) else
   Obs.: Use uma tabela de palavras reservadas e, antes de criar um novo token, verifique se é uma palavra reservada, caso positivo, retorne da tabela.
8. (1 ponto) Adicione suporte a comentários no código-fonte. Comentários não geram token, devem apenas ser pulados como se fossem espaço em branco. Comentários devem começar

com # e continuam até o final da linha. Após encontrar #, o analisador léxico deve ignorar todos os caracteres até encontrar um caractere de final de linha (\n ou \r).

9. (1.5 ponto) Ao encontrar um erro léxico retorne uma mensagem de erro com a linha e a coluna do código fonte que foi encontrado o símbolo gerador do erro.

Entrega

Os trabalhos devem ser entregues até o dia 22/03/2022, impreterivelmente até as 23 horas e 59 minutos. Exclusivamente para esta atividade, o envio deverá ser feito por e-mail para maelso.bruno@ci.ufpb.br, com o título Compiladores – Checkpoint 01.

Emails não enviados, salvos como rascunhos, caixa de saída, etc, serão considerados como atrasos.

Deve ser enviado o código fonte compactado, junto com instruções para execução (possíveis dependências para instalação).

Cada dia de atraso na entrega do trabalho acarretará em um desconto de 10% na nota atual do trabalho. Por exemplo, considerando-se um trabalho avaliado inicialmente com nota 10:

1. Se o trabalho for entregue no prazo, o aluno receberá nota 10;
2. Se o trabalho for entregue com 1 dia de atraso, os(as) integrantes receberá(ão) nota 9 (= 10−10%);
3. Se o trabalho for entregue com 2 dias de atraso, os(as) integrantes receberão nota 8,1 (= (10 − 10%) − 10%);
4. Se o trabalho for entregue com 3 dias de atraso, os(as) integrantes receberão nota 7,3 (= ((10− 10%) − 10%) − 10%);
5. Após 3 dias de atraso, o trabalho não será avaliado, e a nota será zero.