LINGUAGENS FORMAIS E COMPILADORES
AP1 – parte 2 – TRABALHO EM GRUPO
PROF. CLAYTON JONES ALVES DA SILVA

Condições gerais:

1. O trabalho (parte 2 da AP1) perfaz 50% da nota da primeira avaliação bimestral.
2. O trabalho deve ser realizado e submetido em grupo. Os grupos são os mesmos designados para a disciplina.
3. Data de entrega do trabalho: **13 de outubro de 2022**.

Dados do problema:

Definimos aqui uma linguagem de programação denominada C--, que é uma linguagem apropriada para um projeto de compilador por ser mais complexa que a linguagem TINY, pois inclui funções e matrizes.

Trata-se essencialmente de um subconjunto de C, mas sem algumas partes importantes, o que justifica seu nome.

Neste documento listamos as convenções léxicas da linguagem, incluindo uma descrição dos *tokens*.

CONVENÇÕES LÉXICAS DE C - -

Palavras-chave

As palavras-chave da linguagem são as seguintes:

{else if int float return void while main}

Todas as palavras-chave são reservadas e devem ser escritas com caixa baixa

Símbolos especiais

Os símbolos especiais são os seguintes:

{'+','–','', '/', '<', '<=', '>', '>=', '==', '!=','/*','*/', ';', '{', '}', '.', '(', ')'}*

Tokens

Utiliza os *tokens* SIMBOLO e RESERVADA, para descrever os símbolos especiais e palavras-chave. Utiliza, ainda, os *tokens* ID, NUMINT e NUMFLOAT, que são definidos pelas expressões regulares a seguir

ID = letra (letra|dígito)*

NUMINT = dígito dígito*

NUMFLOAT = NUMINT . NUMINT*

letra = al...IzIAl.. IZ

dígito = 0|1|...|9

Existe diferença entre caixa baixa e caixa alta.

Espaços em branco

Compostos por brancos, mudanças de linha e tabulações. O espaço em branco é ignorado, exceto como separador de IDs, NUMs e palavras-chave.

Comentários

São cercados pela notação usual de C /*...*/. Os comentários podem ser colocados em qualquer lugar que possa ser ocupado por um espaço em branco e só podem incluir uma linha. Comentários não podem ser aninhados.

Pedido:

1. Elaborar o Diagrama de Autômatos Finitos Não Determinístico (NFA) dos *tokens* da linguagem C--, utilizando as expressões regulares definidas.

2. Elaborar a função que implementa o NFA proposto. A função deve retornar o token ERRO, se as convenções léxicas da linguagem C-- não forem atendidas pelo código fonte. Em caso contrário, deve retornar o valor do token correspondente à cadeia de caracteres identificada.

3. Elaborar a função *getToken()* do analisador léxico para a linguagem C--, que recebe uma cadeia de caracteres válida como argumento, cria o registro do *token* com os campos adequados e retorna o valor do *token*.

4. Elaborar o analisador léxico que leia o código fonte e produza o *array* de registros de *token*.