Trabalho de Compiladores – Unidade I

Notas

Este trabalho valerá 10 pontos na nota da primeira unidade. Destes pontos, serão atribuídos 3 para o analisador léxico e 7 para o analisador sintático.

Tamanho dos grupos

Este trabalho é individual.

Especificações

Deve-se implementar o analisador léxico e analisador sintático LALR(1) ou LL(1) para uma versão reduzida inspirada na linguagem que cada aluno escolheu. As linguagens de cada aluno está na tabela abaixo.

Aluno	**Linguagem de desenvolvimento do compilador**
DANIEL SOUZA SILVA MAIA	
ERIK MONTGOMERY DE BRITTO	
GUILHERME EUGENIO BISPO NASCIMENTO DE OLIVEIRA	
JOAO PEDRO DALTRO MARQUES	
LUCAS RAPHAEL AMORIM GOMES ARAÚJO	
MICAEL SOUZA DE CASTRO	
VINICIUS NASCIMENTO DOS SANTOS SILVESTRE	
YURI LIMA DOS SANTOS SILVA	

Os detalhes da versão reduzida estão na seção seguinte. Para esta implementação, deve-se criar um analisador léxico e um analisador sintático na linguagem definida acima.

Seu programa deve usar como entrada um arquivo-texto na linguagem reduzida proposta. Seu código principal, deve permitir que seja fornecido este arquivo via linha de comando e caso o arquivo esteja sintaticamente correto, não deve fornecer saída alguma. Caso exista algum erro léxico ou sintático, deve-se no mínimo, exibir a linha e coluna onde o erro ocorreu. Use arquivos escritos na sua linguagem escolhida para testar, contendo apenas variáveis dos tipos permitidos, descritos mais abaixo.

O código principal do seu compilador deverá se chamar **Mini<Linguagem>,** onde <Linguagem> é **Haskell,** isto é, a entrada para seu compilador. Assuma que a linguagem para desenvolver

seu compilador seja Java. Sendo assim, um arquivo de teste qualquer deve ser compilado pela linha de comando da seguinte forma:

```
java MiniHaskell teste.hs
```

Onde `teste.hs` é um exemplo de nome de arquivo-texto de entrada a ser testado pelo seu compilador.

Linguagem reduzida

A linguagem escolhida é semelhante à original, mas não apresenta dados criados pelo usuário como objetos, nem Monad (do while, loop, etc.) nem importa ou implementa módulos. Comandos de escrita e leitura via terminal são permitidos. Não será usado infixl, infixr nem infix. Possui os comandos típicos de condicionais (if, case, guards). Serão aceitas variáveis do tipo Int, Float, Bool, Char, String e arrays e tuplas destes tipos. Mesmo que a linguagem original não seja, você deve forçar que ela seja sensível a maiúsculas e minúsculas.

Considere nas indentações em MiniHaskell que um tab representa 8 espaços em branco.

Na linguagem adotada, serão usadas as mesmas palavras reservadas da linguagem original, com exceção das palavras usadas para outros propósitos diferentes dos citados acima, como por exemplo, aquelas para criação de novos tipos.

As seguintes operações são permitidas:

- Operações aritméticas;
- Operações boolianas;
- Operações relacionais.

A tipagem da sua linguagem será fortemente estática mesmo que a linguagem original não seja. Os tipos serão unificados na declaração.

Os comentários são feitos da mesma maneira que a linguagem original.

Faz parte do trabalho buscar a gramática da linguagem original e utilizar somente a parte referente ao especificado neste documento. Usar partes da gramática além do especificado aqui terá desconto em sua nota.

Avaliação

Os seguintes itens serão considerados na avaliação:

- Documentação por comentários e legibilidade: cada arquivo escrito pelo(s) membro(s) dos grupos deve ser comentado de forma a esclarecer o que cada parte deste arquivo faz. Notem que não foi pedido arquivo algum além dos códigos fontes. Dessa forma, os próprios arquivos de código fonte deverão ter informações de como funciona cada parte do programa. Se necessário, pode ser criado um arquivo de documentação extra. Além disso, seu código deve ser feito de tal forma que um outro programador leia o que você implementou com

facilidade. Para isto escrevam códigos bem indentados e usem nomes para variáveis e outros campos bem claros e óbvios. Pesquisem como é feito um código legível e uma documentação de qualidade na Internet e se inspirem no que foi lido para documentar o código deste trabalho. Procurem também por padrões de codificação e se baseiem em algum para implementar seu trabalho. A nota para este item será proporcional à qualidade da informação disponível na forma de comentário e à legibilidade de seu trabalho. Regras gramaticais, coesão e coerência também serão avaliados em seus comentários.

- Funcionamento correto: Seu programa deve executar o que foi descrito acima corretamente. É de sua responsabilidade tratar possíveis erros. A nota para este item será plena se não houver erros durante os testes. No dia da entrega do trabalho, os alunos deverão apresentar o software em funcionamento para o professor. Durante esta apresentação, o professor pedirá para usar algumas entradas e seu programa deverá funcionar para estas entradas. As entradas não serão fornecidas antes desta avaliação. Teste seu programa para que as entradas possíveis sejam implementadas corretamente.

- Conformidade com as especificações: As funcionalidades descritas acima devem ser atendidas plenamente. A nota para este item será 100% se todas as funcionalidades descritas acima forem implementadas.

- Apresentação: O(s) membro(s) dos grupos deve(m) mostrar os arquivos para o professor no dia da apresentação. O grupo deve enviar uma cópia para o e-mail do professor. Não será aceita apresentação em dia diferente do combinado em sala. Haverá desconto na nota por atraso conforme seção “Descontos de ponto” abaixo. A apresentação consiste em arguição do código e compilação e execução durante a aula. Também serão testados alguns arquivos de teste durante a aula, como mencionado acima.

Abaixo está a tabela de pontuação para cada um destes pontos tanto para a análise léxica como para a sintática:

Item avaliado	**Peso**
Documentação, legibilidade e regras gramaticais	2
Funcionamento correto	5
Conformidade com o especificado	1
Arguição	2

Descontos de ponto

Exponenciais em relação ao dia de atraso na entrega (-2^d pontos, para d dias de atraso, d > 0).

Ponto extra

Arguição em inglês: se a conversa for toda na língua inglesa, isto é, o aluno explicar e responder às perguntas do professor feitas em inglês, o aluno terá um total de 2,0 pontos extras nesta unidade (0,6 no analisador léxico e 1,4 no sintático). O código fonte não precisa estar em inglês. Não serão descontados pontos devido a erros de pronúncia na comunicação em inglês.

Datas

18/09/24 Entrega do analisador léxico.

21/10/24 Entrega da primeira parte do compilador (analisadores léxicos e sintático) e gerador de árvore sintática.