Trabalho prático 1 - Análise léxica e sintática descendente recursiva.

Construção de Compiladores I

Prof. Rodrigo Ribeiro

1. Instruções importantes

Nesta seção são apresentadas diversas informações relevantes referentes a entrega do trabalho e orientações a serem seguidas durante a implementação do mesmo.

Leia atentamente antes de começá-lo.

1.1. Equipe de desenvolvimento

O trabalho será desenvolvido por grupos de até dois alunos. Não será permitido o trabalho ser realizado por três ou mais pessoas.

1.2. Escolha da linguagem de programação

O trabalho deverá ser desenvolvido na linguagem Haskell. No que diz respeito a implementação, todas as estruturas de dados referente a modelagem e desenvolvimento do trabalho deverão ser implementadas. Poderão ser usadas apenas bibliotecas de estruturas de dados elementares (listas, filas, pilhas, árvores, etc.), leitura de strings e arquivos ou ferramentas de uso geral, como o gerador de analisadores léxicos Alex.

1.3. Artefatos a serem entregues

Os artefatos a serem entregues são:

- Código fonte do programa;
- Relatório do trabalho em formato pdf. O repositório inicial do trabalho contém um template latex para relatório. **NÃO** serão aceitos relatórios escritos utilizando outras ferramentas.

Antes de enviar seu trabalho para avaliação, assegure-se que:

- Seu código compila e executa em ambiente Unix / Linux. Programas que não compilam receberão nota zero;
- Todos os fontes a serem enviados têm, em comentário no início do arquivo, nome e matrícula do(s) autor(es) do trabalho;
- Arquivo do relatório tenha a identificação do(s) autor(es) do trabalho;

1.4. Critérios de avaliação

A avaliação será feita mediante análise do código fonte, relatório e estrutura de commits do repositório. Os seguintes fatores serão observados na avaliação do código fonte: corretude do programa, estrutura do código e legibilidade. A corretude se refere à implementação correta de todas as funcionalidades especificadas, i.e., se o programa desenvolvido está funcionando corretamente e não apresenta erros. Os demais fatores avaliados no código fonte são referentes a organização e escrita do trabalho.

Nesse quesito, será observado:

- Estruturas de dados bem planejadas;
- Código modularizado em nível físico (separação em arquivos) e lógico (arquitetura bem definida);
- Legibilidade do código, i.e., nomes adequados de variáveis, funções e comentários úteis no código;

O relatório do código deve conter informações relevantes para compilar, executar e auxiliar no entendimento da estratégia adotada para resolução do problema e do código fonte.

Ressalta-se que no relatório não deve conter cópias do fonte – afinal o seu fonte é um dos artefatos entregue –, porém trechos de códigos podem ser incluídos caso isso facilite a explicação e elucidação das estratégias utilizadas.

O relatório deve apresentar as decisões de projetos tomadas: expressões regulares e autômatos para os tokens da linguagem, estruturas de dados usadas, estratégia de implementação do analisador léxico e detalhes de leitura da entrada, dentre outras informações.

Tenha atenção ao prazo de entrega, pois não serão permitidas alterações nos repositórios após a data limite.

2. Especificação técnica do trabalho

Ao longo do semestre será construído um compilador para a linguagem lang. Assim, neste primeiro trabalho pede-se que sejam implementados os analisadores léxico e sintático descendente recursivo desta linguagem.

O executável de sua ferramenta deve suportar as seguintes opções em linha de comando:

- `--lexer`: executa apenas o analisador léxico de lang, imprimindo todos os tokens encontrados na saída padrão. Por exemplo, para o programa

```
main() {
  print fat(10)[0];
}

fat(num :: Int) : Int {
  if (num < 1)
    return 1;
  else
    return num * fat(num-1)[0];
}

divmod(num :: Int, div :: Int) : Int, Int {
  q = num / div;
  r = num % div;
  return q, r;
}
```

a saída será:

```
ID:main
(
)
{
PRINT
ID:fat
(
INT:10
...
```

- `--recursive`: executa o analisador léxico e o analisador sintático descendente recursivo sob a lista de tokens produzido pelo analisador léxico. Como resultado, seu programa deve imprimir uma representação visual da árvore de sintaxe do código processado. Para desenvolvimento do analisador, você deverá utilizar a biblioteca de combinadores presente no repositório da disciplinas (módulo `Parser.Recursive.SimpleCombinators`). Uma maneira prática de utilizar uma representação visual para árvores é utilizar a função `drawTree :: Tree String -> String`, do módulo `Data.Tree`. O uso de representações geradas automaticamente pelo compilador GHC através de uma instância da classe `Show` não serão aceitas.

- `--help`: imprime uma mensagem de ajuda mostrando as diferentes opções suportadas por seu compilador.

3. Entrega do Trabalho

A data da entrega do trabalho será até o dia **11 de agosto de 2024**.