Trabalho prático 3 - Análise semântica e geração de código.

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 e o gerador de analisadores sintáticos Happy.

1.3. Artefatos a serem entregues

Os artefatos a serem entregues são:

- Código fonte do programa;
- Relatório do trabalho em formato pdf.

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 terceiro trabalho pede-se que sejam implementados o analisador semântico e o geradores de código para Lang.

O software desenvolvido deverá fornecer as seguintes opções de linha de comando:

- `--lexer`: Deverá executar apenas o analisador léxico de Lang e apresentar na saída padrão todos os tokens encontrados. Esta funcionalidade já foi implementada por você no trabalho 1.
- `--recursive-tree`: Deverá realizar a análise sintática descendente recursiva produzindo, como resultado, uma representação visual da árvore de sintaxe utilizando o módulo `Data.Tree` de Haskell.
- `--lalr`: Deverá realizar a análise sintática utilizando o analisador LALR produzido pela ferramenta Happy e executar o interpretador sobre a árvore de sintaxe abstrata produzida pelo parser. Note que nesta opção você não deverá executar o analisador semântico para Lang.
- `--peg`: Deverá realizar a análise sintática utilizando o analisador PEG produzido por você usando a biblioteca desenvolvida durante a disciplina. Após a análise sintática, você deverá executar o código representado pela árvore de sintaxe produzida pelo parser usando o interpretador. Note que nesta opção você não deve executar o analisador semântico para Lang.
- `--typed-interp`: Deverá realisar a análise sintática utiizando o analisador sintático LALR e, em seguida, realizar a análise semântica do programa de entrada antes de sua intepretação.
- `--typed-python`: Deverá realisar a análise sintática utiizando o analisador sintático LALR e, em seguida, realizar a análise semântica do programa de entrada e produzir o código Python equivalente ao programa de entrada.
- `--typed-vm`: Deverá realisar a análise sintática utiizando o analisador sintático LALR e, em seguida, realizar a análise semântica do programa de entrada e produzir o código de máquina virtual correspondente ao programa de entrada.

Adicionalmente, como resultado deste trabalho, espera-se que seu interpretador seja capaz de executar diversos programas simples:

- Cálculo de fatorial (utilizando comandos de repetição e recursão).
- Cálculo do n-ésimo termo da sequência de Fibonnacci (utilizando comandos de repetição e recursão).
- Implementação de algoritmos de ordenação de arrays: bubble sort, selection sort, insertion sort e merge sort.
- Implementação de uma pilha, filas, listas encadeadas e de suas operações (criação, inserção, remoção, contagem de número de elementos).
- Implementação de uma árvore binária de busca e de suas operações (criação de uma árvore vazia, inserir, pesquisar, remover).

Todos esses exemplos devem ser implementados utilizando Lang e devem

possuir casos de teste que comprovem o correto funcionamento destas implementações.

3. Entrega do Trabalho

A data da entrega do trabalho será até o dia **19 de Março de 2025**.