Projetos – Compiladores 2017-1

Franklin Ramalho
Professor DSC-UFCG

Motivação

- Projeto da disciplina permite:
  - Fixar e usar conceitos, técnicas e algoritmos estudados em sala de aula
  - Conhecer e usar linguagens e ferramentas criadas para auxiliar a construção de compiladores
  - Trabalhar em equipe
  - Conhecer novas linguagens de programação
  - Revisar e usar conceitos relativos às linguagens de programação
  - Conhecer linguagem de máquina
  - Construir um compilador!
  - Tornar-se um profissional mais capacitado e mais completo!

Etapas

- Construir compiladores
  - Análise Léxica (Completa)
  - Análise Sintática (Completa)
  - Análise Semântica (Restrita ao escopo)
  - Gerador de Código (Restrito ao escopo)

Linguagens

- Equipes:
  - 03 aluno(a)s
- Linguagem Fonte:
  - Java
- Linguagem Destino: *Assembly (versão simplificada estudada em sala de aula)*

Ferramentas

- Serão usadas duas linguagens/ferramentas:
  - JFLEX/CUP
  - Xtext
- Serão indicadas as preferências, mas caso não haja balanceamento, serão escolhidos de forma aleatória e balanceada (via sorteio)
  - Linguagem/Ferramenta

Análise

- A etapa de Análise Léxica e Sintática corresponderá à:
  - Analisador Léxico
  - Analisador Sintático
  - Entrega de analisador funcionando
  - Uso de linguagens/ferramentas (Jflex/CUP ou Xtext)
  - Eliminar eventuais problemas das gramáticas
  - Gramática completa da linguagem
- Será sugerida uma gramática BNF para a linguagem fonte

Em todas as etapas de Análise

- Detectar e informar todos os tipos de erros!
  - Apresentar mensagens precisas sobre o erro ocorrido (caso ocorram)
  - Indicar linhas de erro do código
- Construir e manipular a Tabela de Símbolos

Análise Semântica

- O sistema de tipos da linguagem precisa ser bem estudado
  - Essenciais para análise semântica e geração de código!
- Os contextos precisam ser conhecidos
  - Programas, variáveis, etc.
  - Abstrações de funções e procedimentos, comandos, etc.
  - Essenciais para análise semântica e geração de código!
- As restrições da linguagem fonte precisam ser conhecidas e respeitadas

Análise Semântica – Escopo comum

- Todos os projetos deverão:
  - Realizar checagem de tipos e contextos:
    - Tipos existentes
    - Abstrações (nome, quantidade e tipos de parâmetros de entrada e tipo de retorno)
    - Polimorfismos e sobrecarga
    - Declaração e uso de Variáveis
    - Comandos de atribuição
    - Expressões aritméticas

Síntese - Escopo comum

- Todos os projetos deverão:
  - Gerar código para:
    - Abstrações (nome, quantidade e tipos de parâmetros de entrada e tipo de retorno)
    - Declaração e uso de Variáveis
    - Comandos de atribuição
    - Expressões aritméticas

Análise Semântica e Síntese

- Porém, para as etapas de análise semântica e geração de código, as equipes terão um escopo ainda mais reduzido (definido através de sorteio em sala de aula) da seguinte forma:
  - Escopo A
    - Procedures/Funções
    - Polimorfismo e sobrecarga
    - Expressões relacionais , literais (inteiros, string, booleanos)
    - Comandos condicionais: *if-then-else*
  - Escopo B
    - Procedures/Funções
    - Polimorfismo e sobrecarga
    - Expressões booleanas, literais (inteiros, string, booleanos)
    - Comandos condicionais: *switch-case*

Análise Semântica e Síntese

- Escopo C
  - Procedures/Funções
  - Polimorfismo e sobrecarga
  - Expressões relacionais, literais (inteiros, string, booleanos)
  - Comandos iterativos: *for*
- Escopo D
  - Procedures/Funções
  - Polimorfismo e sobrecarga
  - Expressões booleanas, literais (inteiros, string, booleanos)
  - comandos iterativos: *while*

Análise Semântica e Síntese

- Escopo E
  - Procedures/Funções
  - Polimorfismo e sobrecarga
  - Expressões booleanas, literais (inteiros, string, booleanos)
  - Comandos condicionais: *if-then-else*
- Escopo F
  - Procedures/Funções
  - Polimorfismo e sobrecarga
  - Expressões relacionais, literais (inteiros, string, booleanos)
  - Comandos condicionais: *switch-case*
- Escopo G
  - Procedures/Funções
  - Polimorfismo e sobrecarga
  - Expressões booleanas, literais (inteiros, string, booleanos)
  - comandos iterativos: *for*

Análise Semântica

- Se alguma análise complementar precisar ser feita (para viabilizar as demais análises), ela deverá ser feita, mesmo que não esteja contemplada aqui na especificação.

Síntese

- A etapa de geração de código também terá escopo reduzido semelhante à etapa de análise semântica (mesmo escopo para as duas etapas).
  - Maiores detalhes serão dados após a entrega da etapa de análise semântica.

Síntese

- Se alguma geração complementar precisar ser feita (para viabilizar as demais gerações), ela deverá ser feita, mesmo que não esteja contemplada aqui na especificação.

Análise Semântica e Síntese

- As etapas de análise semântica e geração de código devem ser feitas através de **ações semânticas**, incorporadas na própria gramática da linguagem fonte!
  - JFlex + CUP
  - Xtext
- Se o compilador gerar código para exemplos com erros (léxicos, sintáticos ou semânticos), a etapa de geração de código será penalizada também.
  - Análise bem feita é essencial!

Prazos e entregas

- Análise léxica/Sintática: (20%): 26/07
- Análise Semântica (40%): 26/07
- Gerador de Código (40%): 15/08
- As etapas de análise são obrigatórias. Sem elas, a nota final do projeto será ZERO.
  - Ou seja, quem não entregar os analisadores, não poderá entregar o gerador de código!
  - Até porque a geração de código deverá ser feita com ações semânticas!

Projetos

- Equipe: (03)
- Nota: Individual
- Compõe uma das três notas da disciplina
- Projetos poderão ser cobrados em mini-provas!
- Serão solicitadas mudanças nos projetos no momento da defesa!

Projeto

- Mais que viabilizar uma boa nota, o projeto permite ao aluno exercer suas habilidades dentro de sua profissão
- O aprendizado com projetos é enorme!
- Desbravar linguagens e ferramentas faz parte do seu dia-a-dia
- Você estará praticando sua profissão!