COMPILADORES

Prof. Marcus Ramos

LINGUAGEM MINI-PASCAL

Observações gerais:

- A documentação deverá ser entregue sempre em versão impressa; a entrega da mesma em versão digital é opcional;
- Os arquivos-fonte dos programas deverão ser entregues apenas em formato digital; eles não deverão ser entregues em formato impresso;
- O projeto é incremental: todo material (documentação e arquivos) elaborado para uma fase pode e deve ser revisto, corrigido e melhorado para as etapas seguintes;
- Não haverá controle de prazos, apenas verificação do resultado final. Em caso de dúvidas, consultar o professor durante o desenvolvimento do projeto.

Etapa 1 – ANÁLISE LÉXICA

- Criar, a partir da gramática fornecida, uma relação dos tokens da linguagem.
- Obter uma gramática léxica para este conjunto.
- Obter uma expressão regular para este conjunto.
- Implementar o analisador léxico conforme o modelo do livro.
- Testar e documentar o analisador léxico.

Etapa 2 - ANÁLISE SINTÁTICA

- Verificar se a gramática da linguagem é LL(1). Justificar a sua resposta.
- Obter uma gramática equivalente que seja LL(1).
- Demonstrar, através do cálculo dos conjuntos first e follow, que a nova gramática é LL(1).
- Obter, a partir da nova gramática uma gramática sintática para a linguagem.
- Implementar, através do método recursivo descendente, um analisador léxico para a linguagem;
- Implementar, através do método recursivo descendente, um analisador sintático para a linguagem;
- Integrar os analisadores léxico e sintático;
- Projetar e implementar uma interface com o usuário (linha de comando ou janela);
- Desenvolver os casos de teste;
- Documentar o trabalho (sintaxe da linguagem-fonte, estrutura léxica, estrutura sintática, exemplos de programas-fonte, transformações gramaticais efetuadas, técnicas de análise empregadas, estruturas de dados e algoritmos utilizados, descrição da interface com o usuário, mensagens de erro emitidas, exemplos de entradas e saídas, testes efetuados, manual de instalação e manual de operação).

Etapa 3: MONTAGEM DA AST

- Construir uma estrutura de dados que represente a estrutura sintática do programa-fonte (AST); para isso, deverão ser especificadas as classes abstratas e concretas que serão utilizadas para representar os nós da árvore. Depois, os métodos de análise

sintática deverão ser adaptados para construir a árvore durante o fluxo de processamento do programa-fonte.
- O programa deverá prever uma opção que permita ao usuário visualizar a árvore depois de montada. Utilizar o padrão de projeto VISITOR.

Etapa 4: ANÁLISE DE CONTEXTO

- Descrever, em detalhes e com exemplos, todas as dependências de contexto da linguagem (regras de escopo e regras de tipos). Casos omissos deverão ser definidos e documentos pela equipe de projeto.
- Implementar o analisador de contexto, incluindo a tabela de símbolos e os métodos que irão suportar as fases de identificação e verificação de tipos, observando as regras de escopo e as regras de tipo da linguagem. Utilizar o padrão de projeto VISITOR. Mensagens de erro deverão ser emitidas caso as dependências de contexto da linguagem sejam violadas pelo programa-fonte.

Etapa 5: GERAÇÃO DE CÓDIGO

- Implementar a geração de código para todos os comandos, funções e procedimentos da linguagem, usando os padrões de código apropriados. Implementar e utilizar um gerador de rótulos. Criar um arquivo TXT contendo as instruções da máquina-objeto. As variáveis deverão ser referenciadas pelos seus nomes. Utilizar como máquina-objeto a máquina TAM, definida no livro-texto. Considerar o modelo de máquina de pilha. Utilizar o padrão de projeto VISITOR.
- Opcional:
    - Gerar endereço para variáveis e parâmetros (deslocamento + registrador);
    - Gerar código para acessar vetores e matrizes;
    - Gerar link estático para chamada de procedimentos e funções;
    - Gerar código para declarações de variáveis;