Universidade Federal de Alagoas
Curso de Ciência da Computação

Compiladores - 1^o bimestre

1 Introdução

Compiladores caracteriza-se como uma disciplina cumulativa, não segmentável. O que é visto no primeiro bimestre é essencial no segundo bimestre.

Faltas prejudicam a continuidade e devem ser evitadas, caso inevitável, sugere-se que o aluno busque junto aos colegas o assunto abordado em aula, pois qualquer descontinuidade compromete o aprendizado. Lembrar que o limite de faltas é 25%, o que equivale a 15 aulas, ou 7 dias de aula.

- Avaliações bimestrais:
  - Prova: 60%
  - Trabalho: 40% (Individual ou duplas)
  - Exercícios individuais: até 1 ponto na média.
- Os trabalhos deverão ser apresentados ao professor por ambos os alunos do grupo quando for o caso.
- As datas a serem especificadas são datas limites, entregas antecipadas são preferidas.
- As apresentações deverão ser agendadas pessoalmente com o professor entre 9 e 19h, respeitados horários de aulas e outras atividades prioritárias do Instituto.
- O não comparecimento no horário agendado, com tolerância de 10 min, implicará em reagendamento.

2 Especificação mínima da linguagem (40%)

- A linguagem deve poder ser analisada em passo único.
- Como regra geral para definição, seguir a forma de análise dos trabalhos de CLP (ver documento anexo).
- Especificar a estrutura geral de um programa, indicando onde podem ser declaradas funções, variáveis, instruções, e qual o ponto inicial de execução.
- A especificação de tipos deve ser estática, com no mínimo o seguinte conjunto de tipos de dados:
  - inteiro;
  - ponto flutuante;
  - caractere;
  - booleano;
  - cadeia de caracteres;
  - arranjos unidimensionais.
- Especificar as constantes literais de cada tipo.

- Especificar as operações de cada tipo.
- Especificar coerções suportadas.
- Conjunto mínimo de operadores

  Especificar ordem de precedência e associatividade:

  - aritméticos: para tipos numéricos
    - aditivos, multiplicativos;
    - unário negativo;
  - relacionais:
    - para tipos numéricos, caracteres e cadeias de caracteres: todos;
    - para tipos booleanos: igualdade e desigualdade;
  - lógicos: para tipo booleano
    - negação, conjunção e disjunção;
    - concatenação: geram cadeias de caracteres
      - tipos caracteres e cadeias de caracteres;
      - tipos numéricos e booleanos: se concatenados a um caractere ou cadeia de caracteres, deve ser convertido para cadeia de caracteres.
- Instruções

  Especificar natureza, formas de controle e semântica:

  - estrutura condicional de uma e duas vias;
  - estrutura iterativa com controle lógico;
  - estrutura iterativa controlada por contador com incremento igual a um caso omitido;
  - entrada: deve permitir entrada de mais de uma variável em uma única instrução;
  - saída:
    - deve permitir mais de uma variável/constante literal em uma única instrução;
    - deve permitir minimamente formatação opcional de tamanho do campo e, para ponto flutuante, número de casas decimais, default 2 casas decimais; quando presente o formato deve preceder o elemento a ser impresso.
- Atribuição pode ser instrução ou operador (especificar).
- Funções:
  - Especificar modelos semânticos de passagem de parâmetros suportados e forma de implementação.
- Incluir os seguintes programas exemplos, que deverão ser testados usando o analisador léxico:
  - Alô mundo;
  - Série de Fibonacci: listar os elementos da série até um valor limite, separados por vírgula, implementada em uma função usando iteração com controle lógico; o limite deve ser lido no programa principal;
  - Shell sort: implementado em uma função usando pelo menos uma iteração controlada por contador, em um arranjo cujos valores devem ser lidos e listados no programa principal, também no programa principal listar os valores ordenados.

3 Especificação dos tokens (20%)

- Especificar a linguagem de programação em que os analisadores léxico e sintático serão implementados.
- Especificar a enumeração com as categorias dos tokens a ser obrigatoriamente usada nos analisadores léxico e sintático, usando a sintaxe da linguagem escolhida para a implementação dos analisadores; nomes simbólicos de até 10 caracteres.
- Especificar em dois grupos distintos:
  - tabela com nomes das expressões e as expressão regulares auxiliares, que não representam terminais da linguagem;
  - tabela com as categorias simbólicas dos tokens, especificadas na enumeração de categorias, e as expressões regulares dos lexemas correspondentes, que representam os terminais da linguagem.
- A especificação das expressões regulares devem seguir os padrões do Flex (arquivo anexo).

4 Analisador Léxico (40%)

- Deve ser implementado para fazer a análise on the fly, lendo e listando uma linha de cada vez, devolvendo o *token* identificado via um método/função cuja assinatura seja

  `Token nextToken();`

  para o analisador sintático, e não como um passo em separado que faça a análise léxica toda antes da análise sintática, também não deve ser lido o arquivo fonte inteiro em memória. **`Token`** deverá ser uma **`struct`**/**`class`** contendo os dados do *token*.
- Erros léxicos devem ser enviados para o analisador sintático tratar.
- Imprimir as linhas de código à medida que forem tratadas, numerando-as no formato "`%d4␣␣`" (4 dígitos, alinhados à direita com preenchimento de espaços em branco à esquerda, seguido de 2 espaços em branco).
- **Programa para testar o analisador léxico**:
  - O nome do programa analisado deve ser passado na linha de comando do programa de teste.
  - O programa deverá acessar nextToken() listando para cada *token*, nesta ordem:
    - posição: linha e coluna no programa exemplo;
    - categoria: o número e nome associados na enumeração;
    - valor léxico;
    - o formato de impressão deve seguir o modelo a seguir (em C):
      "`␣␣␣␣␣␣␣␣␣␣[%04d,␣%04d]␣(%04d,␣%20s)␣{%s}`"
      Obs.: indentação de 10 casas; caso o código numérico das categorias ultrapassar 4 dígitos, usar o número par superior mais próximo, garantindo o alinhamento dos campos.
- Como o analisador léxico imprime as linhas à medida que as lê, os tokens deverão estar abaixo da linha onde apareceram
- Devem ser apresentados os resultados dos testes para os três programas previamente solicitados.

SUGERE-SE FORTEMENTE:

- Que versões prévias sejam apresentadas ao professor para análise e discussão.
- Que seja usado o github para manter o trabalho, neste caso resultados poderão ser apresentados indicando o endereço relativo no repositório (o repositório será clonado pelo professor)