Trabalho de Implementação de Analisador Léxico – Parte 1

Nesta página estão detalhados os procedimentos que devem ser seguidos para o desenvolvimento do Trabalho 1 da disciplina de Compiladores I. Este trabalho será desenvolvido ao longo deste semestre letivo e constituirá como parte da nota final da referida disciplina. É fortemente recomendado que os estudantes acessem com freqüência esta página para: esclarecer possíveis dúvidas, estar ciente do cronograma, estar a par de possíveis atualizações/alterações no trabalho.

Objetivos: Projetar e Implementar Analisador Léxico (Parte I) para um sub-conjunto da linguagem Pascal. Para tanto, utilize a linguagem Java e tome como base as aulas teóricas e práticas do curso de Compiladores I até o momento.

O que deve ser feito?

1. Construir a classe AnaLex e métodos para análise léxica. Pelo menos um método (AnaLex) deve existir.
    1. O método deve "ler" o conteúdo do programa passado como parâmetro e retornar os tokens encontrados.

    2. Controlar a numeração das linhas assim como deve-se ignorar comentários. Para cada token reconhecido, deve ser possível exibir: <Numero da linha do token, Token, Atributo (quando possuir atributo)>
    3. Observe que o método AnaLex retorna um token sempre que for chamado. Esse método não deve escrever o token na tela, mas sim produzir um token para um outro método (método main, por exemplo) que será o responsável pela saída

4. As palavras-chave são consideradas palavras reservadas, i.e. não podem ser utilizadas como identitificadores. A lista de palavras-reservadas é dada por: {AND, ARRAY, BEGIN, CASE, CONST, DIV, DO, DOWNTO, ELSE, END, FOR, FUNCTION, GOTO, IF, LABEL, MOD, NOT, OF, OR, POINTER, PROCEDURE, PROGRAM, RECORD, REPEAT, SET, THEN, TO, TYPE, UNTIL, VAR, WHILE, WITH}

5. Criar e gerenciar uma tabela de palavras reservadas (tabela hash ou vetor de strings) para as palavras reservadas da linguagem. Toda vez que um identificador for encontrado, deverá ser feita uma busca na tabela de palavras reservadas. Uma busca com sucesso nessa tabela indica que uma palavra-chave foi reconhecida e então seu token será retornado. Caso contrário, um identificador foi reconhecido e o mesmo deverá ser inserido na tabela de símbolos, caso ainda não tenha sido inserido.

6. Implementar tabela de símbolos como uma tabela hash ou tabela de dispersão. Quando um identificador é reconhecido, deve-se inicialmente fazer uma busca na tabela de palavras reservadas para verificar se é uma palavra reservada. Caso a busca não tenha sucesso, deve-se fazer uma nova busca na tabela de símbolos, e retornar o token associado, caso o identificador seja encontrado. Em caso negativo, o identificador deve ser inserido na tabela de símbolos e então retornar o token associado.

7. Pascal não faz distinção entre identificadores com letras maiúsculas ou minúsculas. ARRAY é equivalente à array. Sugere-se que todas as letras sejam convertidas para uma das duas formas antes de qualquer comparação.

8. Seguir metodologia para convenção de código em programas Java descrita aqui.

9. Definições regulares para tokens sem atributos:

assign_op -> :=	dotdot -> ..	dot -> .	colon -> :
semicolon -> ;	comma -> ,	lb -> [	rb ->]
lp -> (	rp ->)	equal -> =	le -> <=
ge -> >=	ne -> <>	gt -> >	lt -> <
divide -> /	minus -> -	times -> *	plus -> +
simb_pointer -> ^	ender -> @		

10. Definições regulares para tokens com atributos:

letter -> [_A-Za-z]	digits -> digit+	num-> digits optional_fraction optional_expoent	character -> {qualquer cara do alfabeto}
digit -> [0-9]	optional_fraction -> (.digits)?	char1-> {qualquer carac. do alfabeto menos '\n'}	Comment -> (* character* *) | { char1*}
identifier -> letter (letter | digit)*	optional_expoent -> ((E|e)(+|-)? digits)?	string -> 'char1'	

Obs: Note que o reconhecimento de palavras reservadas será no reconhecimento de indentificadores (definição identifier).

11. Delimitadores:

Delim -> ' ' | \t | \n	ws -> delim+

2. Implementar um analisador sintático preditivo recursivo de acordo com a gramática do mini-pascal especificada aqui.
3. Deve ser construído um analisador sintático descendente recursivo sem recuperação de erros. Quando um erro for detectado, o analisador deve emitir uma mensagem de erro, mostrando o tipo de erro detectado e a linha que o erro ocorreu e terminar a execução do programa.

Sugestão de metodologia para desenvolvimento do trabalho

- Trabalho cooperativo e participativo em dupla (ou individual). Na medida do possível, utilize a metodologia XP para realização desse trabalho.
- Faça o projeto do analisador léxico. Organize as classes, atributos e métodos. Lembre-se que, mais tarde, acrescentaremos um analisador sintático nesse projeto!
- Implemente o reconhecimento de um token, faça testes, passe para o próximo. Fazendo isso, erros simples detectados nessa implementação não se repetirão nas próximas.
- Não apague os programas que está criando para testar seu analisador léxico. O conjunto de programas testes deve incrementar de tamanho a medida que novas características são implementadas.
- Utilize o compilador-exemplo que temos adotado em sala de aula para iniciar o entendimento e/ou a implementação do analisador léxico.
- Faça o projeto do analisador sintático. Organize as classes, atributos e métodos. Lembre-se que, mais tarde (próximo semestre), acrescentaremos um analisador semântico nesse projeto!
- Na implementação do analisador sintático, elabore programas-exemplo que explorem as características sintáticas da linguagem.
- Discussão com outras pessoas (professores e alunos do 9 sem.) a fim de entender e tirar dúvidas de implementação.

Qual o Cronograma ?

- 11/03: Inicio do trabalho. Observe que ao longo de todo o semestre teremos aulas dedicadas, exclusivamente, ao desenvolvimento do Trabalho I.
- 10/04: 1o. Checkpoint (Parte I): O horário da aula será dedicado para

avaliação do andamento da Parte I (Analisador Léxico) do Trabalho I. O Professor fará entrevistas com cada grupo procurando saber o que foi implementado, o que não foi implementado, quais são os casos de testes. Será pedido para o grupo explicar partes do código fonte a critério do professor. É esperado que a dupla tenha implementado a classe Analex e os recursos do Analisador Léxico (>70%).

- 29/05: 2o. (e último) Checkpoint (Parte II): O horário da aula será dedicado para avaliação do andamento da Parte II (Analisador Sintático) do Trabalho I. O Professor fará entrevistas com cada grupo procurando saber o que foi implementado, o que não foi implementado, quais são os casos de testes. Será pedido para o grupo explicar partes do código fonte a critério do professor. É esperado que a dupla tenha implementado a classe Parser e os recursos do Analisador Sintático (>80%).
- 12/06 : Entrega e Apresentação do Trabalho:

Como será a avaliação do trabalho na disciplina de Compiladores I?

- NAPC - Avaliações Parciais dos Checkpoints (presença, participação, objetivos cumpridos, etc.) - Cada checkpoint terá Peso 1.5.
- NImp - Implementação (clareza do código, comentários, programas teste, documentação geral do código) - Peso 4
- NRel - Relatório (organização, qualidade da escrita, clareza) - Peso 3
- NB - Bonus (participação na disciplina, utilização/demonstração de ferramentas, qualidade do relatório, adoção de convenção de código, destaque em relação aos demais trabalhos) - Peso 1
- Soma de todas as notas anteriores com seus respectivos pesos. Lembre-se que a média final da disciplina é dada por MF=(P1+P2+T1)/3

Dicas/Sugestões

- Procure entender o código Java que utilizamos em aulas anteriores. Isso ajudará a definir o analisador léxico.
- Inicie o trabalho o quanto antes. O tempo voa!
- Não copie código de ninguém! Nunca, jamais. Se encontrar código semelhante ao que você deseja implementar, estude-o, entenda-o, mas faça a sua prória implementação.

Perguntas frequentes:

P: Se faltar no dia do checkpoint, fica com 0?

R: Há duas possibilidades: 1) Se souber, previamente, que irá faltar em dia de checkpoint, a dupla deve avisar ao professor que irá agendar um horário anterior ao dia do checkpoint. 2) caso a falta seja devido a um imprevisto, a dupla deve procurar o professor e deve justificar a ausência e, após isso, o professor marcará um horário para a entrevista. Faltas sem justificativa, receberão 0 como nota do checkpoint.

P: Há possibilidade de adiar a entrega do trabalho por alguns dias?

R: Não. As datas de entrega estã no limite máximo aceitável. Trabalhos entregues fora do prazo correm o sério risco de não serem avaliados e, no melhor dos casos, serão avaliados mas os atrasos serão descontados no valor da nota, de acordo com critérios definidos pelo professor.

P: O que fazer em caso de dúvidas no entendimento do trabalho?

R: Procure o professor o quanto antes: pessoalmente ou enviando mensagem eletrônica com *subject*: "Dúvidas: Trabalho1 Compiladores I". Nesse último caso, procure discorrer sucintamente a sua dúvida. Além disso, aproveite bem os encontros de *checkpoint* e os horários de atendimento da disciplina para retirar suas

dúvidas e justificar suas escolhas. Isso contará na sua participação e dedicação do trabalho.

P: O que apresentarei no dia 12/06 ?

R: No dia 12/06 os grupos apresentarão ao professor o que foi implementado, o que não foi implementado e farão uma demonstração prática da implementação. O professor poderá pedir para o grupo explicar partes do código fonte e executar a implementação para um conjunto de entradas fornecida por ele.

P: Posso implementar meu analisador léxico/sintático utilizando uma ferramenta para geração automático de analisadores léxico/sintático em Java como Jlex ou JFlex?

R: Não. O grupo é fortemente motivado a utilizar geradores automáticos de analisadores léxico/sintático durante o curso e/ou o desenvolvimento do trabalho. No entanto, a implementação deve ser feita baseando-se unicamente nos conceitos vistos em sala de aula e nos recursos da linguagem Java.

P: Se obter nota máxima em todos os itens da avaliação ficarei com 11?

R: Não. A nota máxima é 10! Além disso, esses pontos não são cumulativos. Isto é, não serão usados em Arq. de Computadores II.