ACH2087 – Construção de Compiladores

Projeto de programação 1

Marcos Lordello Chaim

Escola de Artes, Ciências e Humanidades (EACH)
Universidade de São Paulo (USP)
Este material é baseado nos slides dos professores Fredrik Kjolstad e Alex Aiken da Universidade Stanford, EUA.

Projeto de programação I

Analisador léxico (lexer) da linguagem C-

- Nesta tarefa, você deve escrever regras flex que correspondem às expressões regulares que definem tokens válidos da linguagem C-.
- Estude a descrição de C- no e-disciplinas para definir os tokens a serem reconhecidos.
- O seu lexer deve retornar o lexeme, o nome correto do token e, quando for o caso, o atributo do lexeme identificado. Por exemplo, se o o lexeme for "$>$", deve-se retornar, a cadeia "$>$" (lexeme), **relop** (nome do token ou token), e **GT** (atributo do token). quando for o caso. Veja a tabela do próximo slide como exemplo.
- Você deve criar uma tabela de símbolos para armazenar os identificadores e constantes identificados pelo lexer.

Lexemes, tokens e atributos

Lexeme	Nome do token	Atributo
Qualquer *ws*	–	–
if	**if**	–
then	**then**	–
else	**else**	–
Qualquer *id*	**id**	Ponteiro TS
Qualquer *number*	**number**	Ponteiro TS
<	**relop**	LT
<=	**relop**	LE
=	**relop**	EQ
<>	**relop**	NE
>	**relop**	GT
>=	**relop**	GE

TS: Tabela de Símbolos

Resultados do lexer

- Seu lexer ou scanner deve ser robusto – deve funcionar para qualquer entrada possível. Por exemplo, você deve lidar com erros como um EOF (end of file – fim de arquivo) ocorrendo no meio de comentário. Este é apenas um dos erros que podem ocorrer; veja a descrição C- para identificar outros.
- Você deve incluir código para o término elegante do lexer se ocorrer um erro fatal. *Core dumps* ou exceções não reconhecidas são inaceitáveis.

Resultados do lexer

- Por exemplo, se você encontrar um token que é uma constante, por exemplo, constante (e.g., 34), seu lexer deve salvar o valor na tabela de símbolos; Da mesma forma, se você encontrar um token ID, você precisará salvar o lexeme do ID na tabela de símbolos. Note que nem todos os tokens exigem armazenar informações adicionais; Por exemplo, apenas retornar o tipo de token é suficiente para alguns tokens como as palavras-chaves.

Tratamento de erros

- Erros são comunicados para o parser, retornando um token de erro especial chamado **ERROR**.
- Quando um caractere inválido (um que não pode começar qualquer token) é encontrado, uma string contendo apenas esse caractere deve ser retornada como a string de erro. Retomar a análise léxica no caractere seguinte.
- Você deve salvar a linha de cada token identificado, especialmente o token **ERROR**, para poder reportar a linha onde ocorreu o erro.

Saída do lexer

- O seu lexer não deve imprimir nada. Ele deve passar para o parser as informações do token identificado ou do erro ocorrido. O parser deve imprimir as informações.
- O parser, por enquanto, é um programa (e.g., **main()**) que chama o lexer.
- O lexer não deve parar no primeiro erro.

Projeto de Programação I

Teste e entrega do seu lexer

- Use os exemplos contidos na descrição do C- e crie programas adicionais com erros.
- Cada grupo deverá criar um repositório no github com o código do seu lexer e colocar o professor como participante do projeto.
- A entrega do lexer é no dia 02 de outubro de 2024. Durante a aula os grupos deverão executar seus analisadores léxicos para programas escritos em C-.

Referências

1. *Compilers : principles, techniques, & tools*, Alfred V Aho; Monica S Lam; Ravi Sethi; Jeffrey D Ullman Publisher: Boston : Pearson/Addison Wesley, [2007]