Atividade 04 – Projeto Compiladores

Entrega: AVA (on-line)
Formato: PDF / DOCX / JAVA ou CPP ou PY
Data Entrega/Apresentação: 23/05/2025
Grupo: 4 pessoas

A. Definições

O projeto de Compiladores visa construir uma aplicação de Compilador passando pelas fases: Análise Léxica, Análise Sintática e Análise Semântica.

Cada fase deve ter pelo menos um arquivo OBJ que é a entrada para a fase seguinte.

As linguagens aceitas para a implementação são C/C++, JAVA e Python.

A entrega consiste de defesa do projeto em data previamente marcada em horário de aula e entrega de versão do código fonte compartilhada: fontes e documentação.

Não serão aceitos trabalhos entregues em atraso e/ou sem defesa.

B. Fases

1. Projeto do compilador: todas as expressões regulares e AFDs necessários do compilador.
2. Analisador Léxico: código que identifica os lexemas e gera os tokens equivalentes.
3. Analisador Sintático: código que identifica todas as gramáticas para reconhecimento das cadeias geradas na fase anterior.
4. Analisador Semântico: código que identifica a semântica definida da fase anterior.

C. Especificação

- Para uma linguagem PORTUGOL, codifique um analisador léxico que identifique os tokens da linguagem e crie a tabela de símbolos que será utilizada pelo analisador sintático. Para maior flexibilidade, o analisador léxico deve identificar token a token, cada vez que for requisitado pelo léxico, retornando de alguma forma o código associado ao token e a posição que ele ocupa na tabela, se for o caso.
- Para as palavras reservadas, considere um código único para cada palavra. Para facilitar a implementação do autômato, crie uma tabela de palavras reservadas, utilizando para as palavras reservadas a mesma especificação que

para identificadores e cada vez que for encontrado um identificador, teste se este é uma palavra reservada ou não. Se for o caso, utilize uma função auxiliar para isto, que retorna -1 se não for uma palavra-reservada ou o código da palavra se afirmativo.

- A função main deve ler um arquivo (*.POR) e simular o analisador sintático, chamando sucessivamente o léxico e exibindo outro arquivo na saída padrão (*.TEM) o token encontrado (código) e sua posição na tabela. Para possibilitar a correta implementação, considere um token extra, o de final de arquivo, como um código especial. Tal token será utilizado futuramente no sintático.
- O analisador semântico deve ler o arquivo sintático e todos os arquivos anteriores (se necessário) para ajudar no processo de análise semântica, verificando apenas se uma variável já havia sido definida ou não.

D. Linguagem PORTUGOL

- A linguagem possui apenas um tipo de dado, inteiro, que é equivalente ao tipo *int* da linguagem C. As constantes são tipos numéricas inteiras e literais (strings). As funções leia e imprima são comandos embutidos na linguagem, podendo ser consideradas como palavras reservadas para facilitar a implementação. No caso de imprima, são aceitas variáveis, constantes literais e inteiras. Para leia, apenas variáveis são aceitas.
- Não existem na linguagem variáveis reais ou string, no caso de composição de texto com valores numéricos, devemos utilizar comandos seguidos de imprima. Cada comando deve ser finalizado por ponto e vírgula. A declaração de variáveis é feita uma a uma, sendo opcional a declaração de múltiplas variáveis de acordo com a escolha da equipe.
- As composições de atribuição de variáveis, operações matemáticas e operadores lógicos devem seguir o padrão trabalhado regularmente nas linguagens.

E. Detalhes:

C.1-) As operações matemáticas:
Aritméticas: +, -, /, *, (,)
Lógicas: E, OU, NÃO, (,)
Relacionais: >, <, >=, <=, =, <>
Atribuição: <-

C.2-) Estruturas escreva / leia:
escreva(<valor_inteiro>)
escreva(<variável>)
escreva(<string>)
leia(<variável>)

C.3-) Tipos de dados: inteiro

Declaração:

<tipo>:<identificador>;

C.4-) Tipos de Atribuições

<identificador> <- <identificador>
<identificador> <- <número>
<identificador> <- <número> <operador_matemático> <número>
<identificador> <- <identificador> <operador_matemático> <número>
<identificador> <- <número> <operador_matemático> <identificador>

C.5-) Comandos da linguagem:

se <exp_lógica>
    então
        <lista_CMD>
    senão
        <lista_CMD>
fim_se

obs: senão é opcional

para <inic> até <valor> passo <incremento>
    <lista_cmd>
fim_para

<u>obs:</u> onde passo é opcional, default valendo 1

F. Tabela de Tokens:

Token	Descrição	Lexema Exemplo
ATE	Palavra reservada até	até
ATR	Atribuição <-	nome <- "Pupilo"
E	Operador lógico e	e
ENTAO	Palavra reservada então	então
ESCREVA	Palavra reservada escreva	escreva
FIMPARA	Palavra reservada fim_para	fim_para
FIMSE	Palavra reservada fim_se	fim_se
ID	Identificador para variáveis	nome, idade, sexo, endereço
LEIA	Palavra reservada leia	leia
LOGDIFF	Operador lógico <>	a <> b
LOGIGUAL	Operador lógico =	a = b
LOGMAIOR	Operador lógico >	a > b
LOGMAIORIGUAL	Operador lógico >=	a >= b
LOGMENOR	Operador lógico <	a < b
LOGMENORIGUAL	Operador lógico <=	a <= b
NAO	Operador lógico não	não

NUMINT	Número inteiro	10, 17, 53, 545
OPDIVI	Operador Matemático /	c <- a / b
OPMAIS	Operador Matemático +	c <- a + b
OPMENOS	Operador Matemático –	c <- a – b
OPMULTI	Operador Matemático *	c <- a * b
OU	Operador lógico ou	ou
PARA	Palavra reservada para	para
PARAB	Parênteses abrindo	c <- (a / b)
PARFE	Parênteses fechando	c <- (a / b)
PASSO	Palavra reservada passo	passo
SE	Palavra reservada se	se
SENAO	Palavra reservada senão	senão
STRING	Literal String	"Pupilo"
TIPO	Tipo de dado	inteiro

Grupo Anchieta
Prof. Clayton Valdo