INSTITUTO TECNOLÓGICO DE AERONÁUTICA

Compiladores

CES-41

Laboratório 4

COMP 20

Heládio Sampaio Lopes

Sebastião Beethoven Brandão Filho

Professor:

Fábio Carneiro Mokarzel

ITA-SJC-2019

Enunciado do Laboratório

Objetivo: Construção da tabela de símbolos e do analisador semântico para uma linguagem de programação, usando a ferramenta *Yacc*.

Tarefa:

- Implementar, usando a ferramenta *Yacc*, um construtor de tabela de símbolos e um analisador semântico para a mesma linguagem de programação do 3º Laboratório desta disciplina. O analisador deve usar a estrutura do analisador sintático construído também no referido laboratório.
- O programa resultante deve imprimir o conteúdo da tabela de símbolos dos programas analisados e emitir mensagens de erros caso sejam encontrados.
- Gerar o código C e o código executável desse analisador.

Execução: Usar o executável produzido para construir a tabela de símbolos e analisar semanticamente vários programas escritos na referida linguagem, suficientes para mostrar o bom funcionamento do analisador, incluindo também programas contendo erros semânticos.

Observação: Este laboratório poderá ser feito em dupla, sendo que a mesma dupla deverá fazer também os Laboratórios 5 e 6.

Relatório: Fazer um documento em Word ou pdf contendo a sigla e o nome da disciplina, os nomes dos alunos, o nome do professor, o número do laboratório, a data de realização, o assunto tratado e os resultados obtidos pela execução do programa.

Mensagem ao professor: Enviar o relatório acima pedido, os arquivos com os programas em *Yacc* e *Flex* e aqueles contendo os programas analisados, em um e-mail endereçado ao professor. Não é necessário entregar o código em C gerado pelo *Yacc* nem o executável gerado pelo *Gcc*.

Data de entrega: 04/06/2019.

Código implementado

As Figuras 1, 2, 3 e 4 apresentam as produções responsáveis por definir o funcionamento do analisador sintático.

6) Especificações semânticas de COMP-ITA 2019

- Qualquer identificador deve ser declarado antes de usado.
- Um identificador não pode estar declarado mais de uma vez dentro de uma função, ou como global, mas pode estar declarado ao mesmo tempo como global e numa função qualquer, ou em duas ou mais funções quaisquer.
- Identificadores podem ser do tipo nome de programa, nome de variável ou nome de função.
- Uma função não pode ter o mesmo nome que o de uma variável global.
- Variáveis escalares, expressões e elementos de variáveis indexadas podem ser do tipo inteiro, real, caractere ou lógico.
- A constante inteira usada no dimensionamento de uma variável indexada deve ser maior do que zero.
- Toda variável escalar e, ao menos, um elemento de cada variável indexada, devem ser inicializados e referenciados pelo menos uma vez no programa.
- Deve haver compatibilidade entre os tipos dos dois lados de um comando de atribuição, conforme a seguinte tabela:

Tipo do lado esquerdo	Tipo do lado direito
Inteiro	Inteiro ou Caractere
Real	Inteiro, Real ou Caractere
Caractere	Inteiro ou Caractere
Lógico	Lógico

- Variáveis escalares não podem ter subscritos.

Figura 1: Especifações semânticas relativas à Linguagem "COMP-ITA-2019".

- O número de subscritos de uma variável indexada deve ser igual ao seu número de dimensões declarado.
- Os elementos de uma variável indexada só poderão ser atribuídos ou receber atribuição um de cada vez.
- Os elementos de uma variável indexada só poderão ser lidos, ou escritos um de cada vez.
- Os tipos dos resultados das diversas classes de expressões só podem ser os seguintes:

Classe da expressão	Tipo
Aritmética	Inteiro, Real ou Caractere
Relacional	Lógico
Lógica	Lógico

- Os tipos dos operandos admitidos pelos operadores de expressões são os seguintes:

Operadores	Tipos admitidos dos operandos
|| && !	Lógico
< <= > >=	Inteiro, Real ou Caractere
= !=	Todos (se um for lógico o outro também deve ser)
+ - * / ~	Inteiro, Real ou Caractere
%	Inteiro ou Caractere

- As expressões nos cabeçalhos de comandos **if** e **while** e no encerramento de comandos **do** devem ser relacionais ou lógicas.

Figura 2: Especifações semânticas relativas à Linguagem "COMP-ITA-2019".

- A variável de inicialização do cabeçalho de um comando **for** deve ser escalar do tipo inteiro ou caractere.
- A variável da atualização do cabeçalho de um comando **for** deve ser a mesma da sua inicialização.
- A primeira e a terceira expressão do cabeçalho de um comando **for** deve ser do tipo inteiro ou caractere e a segunda expressão deve ser do tipo lógico.
- A expressão aritmética no subscrito de uma variável indexada deve ser do tipo inteiro ou caractere.
- O programa deve ter uma e uma só função de cabeçalho **main**.
- O identificador de um comando **call** e o de uma chamada de função numa expressão devem ser do tipo nome de função.
- O tipo de função do identificador de um comando **call** deve ser **void** e do identificador de chamada de função que aparece numa expressão não deve ser **void**.

Figura 3: Especifações semânticas relativas à Linguagem "COMP-ITA-2019".

- Um identificador de variável e de parâmetro deve ser do tipo nome de variável.
- O número de argumentos na chamada de uma função deve ser igual ao número de parâmetros da mesma.
- Deve haver compatibilidade entre um argumento de chamada de uma função e seu parâmetro correspondente, conforme a seguinte tabela:

Tipo do parâmetro	Tipo do argumento
Inteiro	Expressão inteira ou caractere
Real	Expressão inteira, real ou caractere
Caractere	Expressão inteira ou caractere
Lógico	Expressão de valor lógico

- Todo comando **return** dentro de uma função do tipo **void** não deve ser seguido de expressão e dentro de uma função que não seja do tipo **void** deve ser seguido por uma expressão.
- Deve haver compatibilidade entre o tipo de uma função que não seja **void** e o tipo da expressão de qualquer comando **return** em seu escopo, conforme a seguinte tabela:

Tipo da função	Tipo da expressão retornada
Inteiro	Inteiro ou Caractere
Real	Inteiro, Real ou Caractere
Caractere	Inteiro ou Caractere
Lógico	Lógico

- Funções não são usadas como parâmetros ou argumentos de chamada de outras funções.
- A linguagem não admite recursividade.

Figura 4: Especifações semânticas relativas à Linguagem "COMP-ITA-2019".

Os códigos anexados ao relatório (lab.y e lab.l) apresentam os programas implementandos em *Yacc* e *Flex* para a realização da análise semântica de algoritmos escritos na linguagem de programação "COMP-ITA-2019".

Além disso, anexou-se, também, a entrada (programa.dat) com um programa nessa linguagem (disponibilizado pelo professor) utilizado como referência para testes.

Validação do código implementado

Para a validação do que foi implementando foi inicialmente utilizado o programa associado à entrada já descrita.

O código original apresentava as seguintes incoerências semânticos: as variáveis globais 'c' e 'fim' não era inicializadas nem referenciadas durante o código do programa. Na função ExibirTabela() a variável 'j' não era declarada. Por fim, na função main dois argumentos eram passados durante a chamada da função Inserir(), que apresenta apenas um parâmetro.

Realizando um ajuste dos erros destacados no programa, tem-se a seguinte Tabela de Símbolo, ilustrada pela Figura 5.

```
   TABELA  DE  SIMBOLOS:

Classe 0:
  (ExibirTabela, IDFUNC,    VAZIO)
Classe 3:
  (   compara, IDVAR,   INTEIRO, 1, 1, Procura)
Classe 7:
  (         c, IDVAR, CARACTERE, 1, 1, Main)
  (      main, IDFUNC,   NAOVAR)
  (         c, IDVAR, CARACTERE, 0, 0, Global)
  (      ntab, IDVAR,   INTEIRO, 1, 1, Global)
  (   palavra, IDVAR, CARACTERE, 1, 1, Global, EH ARRAY, ndims = 1, dimensoes:  10)
Classe 11:
  (       med, IDVAR,   INTEIRO, 1, 1, Procura)
Classe 13:
  (     posic, IDVAR,   INTEIRO, 1, 1, Main)
  (         i, IDVAR,   INTEIRO, 1, 1, Main)
  (         i, IDVAR,   INTEIRO, 1, 1, ExibirTabela)
  (         i, IDVAR,   INTEIRO, 1, 1, Inserir)
  (     posic, IDVAR,   INTEIRO, 1, 1, Inserir)
  (     posic, IDVAR,   INTEIRO, 1, 1, Procura)
  (         i, IDVAR,   INTEIRO, 1, 1, Procura)
Classe 14:
  (         j, IDVAR,   INTEIRO, 1, 1, ExibirTabela)
  (         j, IDVAR,   INTEIRO, 1, 1, Inserir)
  (  fimteste, IDVAR,    LOGICO, 1, 1, Procura)
Classe 15:
  (    nocorr, IDVAR,   INTEIRO, 1, 1, Global, EH ARRAY, ndims = 1, dimensoes:  50)
Classe 17:
  (       fim, IDVAR,    LOGICO, 1, 1, Main)
  (       fim, IDVAR,    LOGICO, 1, 1, ExibirTabela)
  (       fim, IDVAR,    LOGICO, 1, 1, Inserir)
  (       fim, IDVAR,    LOGICO, 0, 0, Global)
  (     nomes, IDVAR, CARACTERE, 1, 1, Global, EH ARRAY, ndims = 2, dimensoes:  50  10)
Classe 18:
  (       inf, IDVAR,   INTEIRO, 1, 1, Procura)
Classe 19:
  (   Inserir, IDFUNC,    VAZIO, TEM PAR, npars = 1, parametros:  INTEIRO)
  (   Procura, IDFUNC,  INTEIRO)
Classe 22:
  (     achou, IDVAR,    LOGICO, 1, 1, Procura)
  (       sup, IDVAR,   INTEIRO, 1, 1, Procura)
  (AnaliseDeTexto, IDPROG)
```

Figura 5: Tabela de Símbolos para o programa utilizado como referência.

Em seguida, foram feitos alguns testes com erros semânticos. Todos os erros foram evidenciados pelo analisador implementado. As entradas (index.dat e funcao.dat) com os programas utilizados como base para os testes encontram-se anexadas ao relatório, e os resultados são ilustrados a seguir.

1. Erros de Indexação:

```
global :
        int x[10, 10];

functions :

        main
        {
        local :
                int y[10, 10];
        statements :
                x

***** Esperado: Subscrito(s) *****

 <- y

***** Esperado: Subscrito(s) *****

;
                x[1, 1, 1]

***** Incompatibilidade: Numero de subscritos incompativel com declaracao *****

 <- 12;
        }

}
```

Figura 6: Erros semânticos obtidos.

2. Erros de atribuição, erros associados a parâmetros e argumentos e erro de impossibilidade de recursividade:

```
functions :

        int f (int x, int y)
        {
        local :
                logic a;
        statements :
                a <- x + y;


***** Incompatibilidade: Lado direito de comando de atribuicao improprio *****

                return a;


***** Incompatibilidade: Retorno improprio *****

        }

        void g (int x, float y)
        {
        local :
                float a;
        statements :
                a <- 3.300000;
                a <- f (a

***** Incompatibilidade: Tipo nao aceito de parâmetro *****

)

***** Incompatibilidade: Numero de parametros incompativel com declaracao *****

;
                call g (3 , 3)

***** Incompatibilidade: Linguagem nao admite recursividade *****

;
                return a;


***** Incompatibilidade: Funcao void nao tem retorno *****

        }
```

Figura 7: Erros semânticos obtidos.