

Construção de Compiladores I

Atividade: Mini-Linguagem

Bernardo Marotta e Melina Lopes

16 de Maio de 2018

Resumo

A proposta dessa atividade é especificar uma mini linguagem de programação simples que aborde alguns dos conceitos apresentados em sala de aula.

1 Sumário

A linguagem

Aspectos léxicos

Aspectos sintáticos

Aspectos semânticos

Biblioteca padrão

Exemplos

2 A linguagem

A linguagem **Lhama** (Language of Heuristic Algorithms and Multithread Architecture) é uma mini-linguagem construída na disciplina de Construção de Compiladores I. É uma linguagem imperativa simples e de tipagem estática. Na linguagem, existem tipos primitivos com inteiro e ponto flutuante e tipos compostos, como arrays e estruturas. Existem também operadores aritméticos, relacionais, lógicos, expressões etc.

3 Aspectos léxicos

Comentários de linha: Começam com o caracter `#` e comentam a linha inteira

Comentários de bloco:

Abre com: `~[`

Fecha com: `]~`

Caracteres brancos: Não interferem no interpretador léxico (servem de separador)

Literais inteiros: Sequência de dígitos decimais

Literais float: Sequência de dígitos decimais reais

Literais booleanos: true e false

Literais string: Delimitado por aspas duplas , dentro da string, pode ser usado qualquer caracter especial, usando \$ para começar e terminar a sequência a ser escapada.

Identificadores: Sequência de letras e/ou dígitos. Pode conter alguns caracteres especiais como _, -, &, etc.

4 Aspectos sintáticos

A fim de mostrar os aspectos sintáticos, será apresentada uma GLC que define a sintaxe das construções da linguagem.

Declaração de funções:

Program \rightarrow Functions

Functions \rightarrow Function

Functions \rightarrow Function Functions

Function \rightarrow Type (Types): Exp end

Type \rightarrow bool id

Type \rightarrow int id

Type \rightarrow string id

Type \rightarrow float id

Types \rightarrow Type TypeRest

TypeRest \rightarrow , Type TypeRest

TypeRest $\rightarrow \lambda$

Tipos literais:

Exp \rightarrow id

Exp \rightarrow id = Exp

Exp \rightarrow Lit

Exp \rightarrow Type[Lit]

Lit \rightarrow lint

Lit \rightarrow lstring

Lit \rightarrow lbool

Lit \rightarrow lfloat

Lit $\rightarrow \lambda$

Lits \rightarrow Lit LitRest

LitRest \rightarrow , Lit LitRest

LitRest $\rightarrow \lambda$

Operações aritméticas:

Exp \rightarrow Exp + Exp

Exp \rightarrow Exp - Exp

Exp \rightarrow Exp * Exp

Exp \rightarrow Exp % Exp

Exp \rightarrow Exp / Exp

Operações relacionais:

$\text{Exp} \rightarrow \text{Exp} = \text{Exp}$

$\text{Exp} \rightarrow \text{Exp} > \text{Exp}$

$\text{Exp} \rightarrow \text{Exp} < \text{Exp}$

$\text{Exp} \rightarrow \text{Exp} \leq \text{Exp}$

$\text{Exp} \rightarrow \text{Exp} \geq \text{Exp}$

Operações lógicas:

$\text{Exp} \rightarrow \text{Exp} \text{ and } \text{Exp}$

$\text{Exp} \rightarrow \text{Exp} \text{ or } \text{Exp}$

Chamada:

$\text{Exp} \rightarrow \text{id} (\text{Exp})$

Expressões:

$\text{Exp} \rightarrow \text{if } \text{Exp} : \text{Exp} \text{ else } \text{Exp} \text{ end}$

$\text{Exp} \rightarrow \text{while } \text{Exp} : \text{end}$

$\text{Exp} \rightarrow \text{for } \text{Exp} \text{ in } \text{Exp} : \text{end}$

$\text{Exp} \rightarrow \text{for } \text{Exp} , \text{Exp} : \text{end}$

$\text{Exp} \rightarrow \text{Exps}$

$\text{Exps} \rightarrow \text{Exp } \text{ExpRest}$

$\text{Exps} \rightarrow , \text{Exp } \text{ExpRest}$

$\text{Exps} \rightarrow + \text{Exp } \text{ExpRest}$

$\text{Exps} \rightarrow \lambda$

Precedência:

$*, /, \%$

$+, -$

$=, >, <, \geq, \leq$

and

or

else, then, :

5 Aspectos semânticos

Atribuição:

A atribuição é feita através do operador $=$.

Ex: $a = 5$

Operadores:

O uso de operadores é feito de forma simples, com as expressões dos dois lados do operador.

Ex: $a < 5, a > 5, a > 5 \text{ and } b < 10$

Declaração:

A declaração de variáveis ou funções segue o padrão:

Ex: `a = 5,`

`a[] = [1,2,3]`

`a[1] = [0]`

`bool function(int)`

Condicionais:

Sendo `c1` uma sequência de comandos e `c2` outra:

`if (BOOL) : c1 end`

`else c2 end`

Repetição:

`while (BOOL):`

`c1`

`end`

`for (ITEM) in (LISTA):`

`c2`

`end`

`for (i) , (i2):`

`c3`

`end`

6 Biblioteca padrão

A id das funções da biblioteca padrão começam com `lh`.

Exemplo: `lh.getLine(Exp)` Guarda a entrada do teclado na variável `id` (entre parênteses).

Outros exemplos:

`lh.print(Exp)` Imprime na tela o conteúdo

`lh.pow(Exp, Exp)` Calcula potência

`lh.fibonacci(Exp)` Calcula fibonacci

`lh.factorial(Exp)` Calcula fatorial

7 Exemplos

Função Fibonacci em Lhama Language:

```
function fibonacci(int n):  
    a = 0  
    b = 1  
    for i = 1, n:  
        a = b  
        b = a + b  
    end  
    return a  
end
```

```
function insertionSort(int array[], int n):  
    int i, key, j;  
    for 1, n:  
        key = arr[i]  
        j = i-1  
        while (j >= 0 and arr[j] > key)  
            arr[j+1] = arr[j]  
            j = j-1  
        end  
        arr[j+1] = key  
    end  
end
```

```
int main():  
    int a  
    lh.getLine(a)  
    fibonacci(a)  
    lh.print("The fibonacci of a is", a)  
end
```