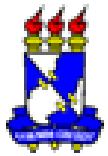
Framework para uma Família de LP

Prof. Alberto Costa Neto
alberto@ufs.br

Linguagens de Programação



Departamento de Computação
Universidade Federal de Sergipe

Linguagens

- Linguagem de Expressões 1 (LE1)
- Linguagem de Expressões 2 (LE2)
- Linguagem Funcional 1 (LF1)
- Linguagem Funcional 2 (LF2)
- Linguagem Funcional 3 (LF3)
- Linguagem Imperativa 1 (LI1)
- Linguagem Imperativa 2 (LI2)
- Linguagem Orientada a Objetos 1 (LO1)

LI1

Linguagem Imperativa 1

(Variáveis e Comandos)

LI1 - Características

- Estende LE2 com **variáveis e comandos**
- Um **programa passa a ser um comando**
- Traz vários comandos comuns em linguagens imperativas
  - **Atribuição, If-Then-Else, While, Read, Write**

LI1 – Características (cont.)

- Introduz a expressão condicional:
  if cond then e1 else e2
  - Caso a **cond** seja true, **e1** é avaliada e retornada. Caso contrário isto é feito com **e2**
- Eager evaluation (parâmetros)
- Utiliza escopo dinâmico

Ambientes (Contextos)

- O ambiente de execução inclui:
  - mapeamento de identificadores em valores (igual ao de LE2), mas permite modificar os valores associados ao identificadores
  - acesso aos valores de entrada e saída
  - lista de valores de saída do programa

- O ambiente de compilação estende o de LE2 (que mapeia Id em Tipo)
  - Inclui a obtenção do tipo dos dados de entrada

LI1 - Sintaxe

Programa ::= Comando

Comando ::= “skip” | Atribuicao | IO | Comando ";" Comando
| IfThenElse | While | ComandoDeclaracao

Atribuicao ::= Id ":=" Expressao

IO ::= "write" "(" Expressao ")" | "read" "(" Id ")"

Expressao ::= ...

IfThenElse ::= "if" Expressao "then" Comando "else" Comando

While ::= "while" Expressao "do" Comando

ComandoDeclaracao ::= "{" Declaracao ";" Comando "}"

Declaracao ::= DeclaracaoVariavel | DeclaracaoComposta

DeclaracaoVariavel ::= "var" Id "=" Expressao

DeclaracaoComposta ::= Declaracao "," Declaracao

LI1 - Exemplos

```
{ var a = 3; write(a);
   { var a = 2, var b = 5; write(a); write(b+a) };
   write(a)
}


{ var n = 0, var m = 0; read(n); read(m);
     if (m == n) then
        write("valores de entrada iguais")
     else
       write("valores de entrada diferentes")
}
```

LI1 – Exemplos (cont.)

```
{ var i = 0;
   while not (i == 3) do
       i := i + 1;
       write("Hello World")
}
```

LI2

Linguagem Imperativa 2

(Procedimentos parametrizados, recursivos e de primeira ordem)

LI2 - Características

- Estende LI1 com procedimentos **parametrizados**, **recursivos** e de **primeira ordem**

- Uma **declaração de procedimento** pode acontecer em qualquer bloco de comandos

- A **chamada de um procedimento** é um comando

LI2 – Características (cont.)

- A **declaração de um procedimento** inclui:
  - uma lista de identificadores (os parâmetros do procedimento) com os respectivos tipos; e
  - um corpo, constituído do comando a ser executado quando o procedimento é invocado
- A **chamada de um procedimento** envolve:
  - o identificador do procedimento invocado; e
  - uma lista de expressões, que deve obedecer a ordem e os tipos dos parâmetros

Ambientes (Contextos)

- O ambiente de execução estende o de LI1 com:
  - Mapeamento de Id em Procedimento
- O ambiente de compilação estende o de LI1 (que mapeia Id em Tipo e obtém o tipo dos dados de entrada)
  - Mapeamento do Id de um procedimento à sua lista de declarações de parâmetros

LI2 - Sintaxe

Programa ::= Comando

Comando ::= “skip” | Atribuicao | IO | Comando ";" Comando | IfThenElse | While | ComandoDeclaracao | ChamadaProcedimento

ComandoDeclaracao :: = "{" Declaracao ";" Comando "}"

Declaracao ::= DeclaracaoVariavel | DeclaracaoProcedimento | DeclaracaoComposta

DeclaracaoVariavel ::= ...

DeclaracaoComposta ::= ...

DeclaracaoProcedimento ::= "proc" Id "(" [ListaDeclaracaoParametro] ")" "{" Comando "}"

ListaDeclaracaoParametro ::= Tipo Id | Tipo Id "," ListaDeclaracaoParametro

ChamadaProcedimento ::= "call" Id "(" [ListaExpressao] ")"

LI2 - Exemplos

```
{ var a = 0, proc incA () {a := a + 1};
        call incA();
        call incA();
        write(a)
}
```

LI2 – Exemplos (cont.)

```
{ var b = 3,
  proc escreveRecursivo (int a) {
     if (not (a == 0)) then {
        var x = 0; x := a - 1;
        write("Ola");
        call escreveRecursivo(x)}
     else skip
  };
  call escreveRecursivo(b)
}
```