INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

ECM253 – Linguagens Formais, Autômatos e Compiladores

Atividade

Interpretador simples com análise sintática descendente recursiva

Marco Furlan

Setembro de 2024

Esta **atividade** consiste em **alterar** o **projeto** de **exemplo** de **analisador sintático descendente recursivo** em **Java desenvolvido** na **última atividade**.

O objetivo é construir um **interpretador** de **expressões** (sem o uso de variáveis, para simplificar) de modo que **cada expressão matemática válida** tenha seu **valor apresentado** na **tela**.

Utilizar a **seguinte gramática**:

```
goal = program;
program = expr,';', {expr,';'} ;
expr = term, eprime;
eprime = { ('+' | '-'), term };
term = factor, tprime;
tprime = { ( '*' | '/' ), factor };
factor = '(', expr, ')' | number | '-',factor;
```

1 Requisitos do projeto

Utilizar uma **pilha** no seu **interpretador**. Veja nos requisitos como utilizá-la.

(R1) Quando um **número for encontrado**, **empilhar**;

(R2) Quando uma **operação binária estiver sendo executada**, **desempilhar** o **segundo operando** e **depois** o **primeiro operando**, **realizar** a **operação** e o **resultado** deve ser novamente **empilhado**;

(R3) Para **inverter** o **sinal** de uma **expressão** (operador unário), primeiro **desempilhar** o **valor** da pilha, **inverter** seu **sinal** e depois **empilhar** novamente;

(R4) Quando uma **expressão** for **terminada**, **remover** seu **valor** da **pilha** e **exibir** seu **valor** na **tela**.

2 Exemplo de funcionamento

O arquivo de entrada:

```
4+3*-2;
7/2-1;
```

Ao ser interpretado produziu:

```
Value: -2.0
Value: 2.5
Análise léxica terminada com sucesso!
Análise sintática terminada com sucesso!
```

3 O que é para entregar

Enviar o projeto desenvolvido, compactado em arquivo tipo ZIP para a **tarefa indicada** no **Canvas** da disciplina.