Universidade Federal do Amazonas - UFAM
Instituto de Computação – IComp, Curso de Ciência da Computação
Compiladores -- Projeto I: Calculadora por Extenso (versão 1.0)

Objetivo

Neste projeto você deve escrever um interpretador de uma calculadora que aceita números por extenso e dá respostas por extenso. Embora simples, este projeto ilustra o uso de dois parsers distintos em um mesmo programa.

Descrição

A calculadora a ser implementada possui notação infixada, com suporte apenas às operações de adição e multiplicação ("mais" e "vezes"), além do uso de parênteses como forma de mudar a ordem de precedência das operações. A seguir temos um exemplo de interação com a calculadora:

```
Calculadora por extenso (digite 's' para sair)
Exemplos de expressoes: dois mais cinco, sete vezes (doze mais um)
? sete
  sete
? sete vezes sete
  quarenta e nove
? sete vezes sete mais cinco
  cinquenta e quatro
? sete vezes (sete mais cinco)
  oitenta e quatro
? um mais um mais um mais um mais um mais um vezes vinte e cinco
  trinta e um
? sete vezes vinte e cinco
  cento e setenta e cinco
? um mais um mais um mais um mais um
  cinco
? s
```

Este interpretador envolve dois parsers distintos. O primeiro reconhece expressões por extenso (ex: "`sete vezes sete`") e as traduz para o seu valor correspondente (ex: 49). O segundo traduz o número obtido para a sua versão por extenso (ex: "`quarenta e nove`"). No código base fornecido com este enunciado, o segundo parser opera sobre o inverso do número a ser traduzido (ex: 94 em vez de 49) para facilitar a escrita de uma gramática LL1.

Avaliação

O seu interpretador será avaliado pela gramática de atributos entregue, os códigos de apoio em Java e, fundamentalmente, pela execução de expressões de teste. Nos testes, nunca serão fornecidos números menores que 0 ou maiores que 1999. Nenhuma operação a ser realizada irá resultar em número acima de 9999. Strings de entrada terão, no máximo, 512 caracteres.

Observações

(a) Trabalho feito por, no máximo, dois alunos.
(b) Plágio não será tolerado, com anulação dos projetos envolvidos.
(c) Em anexo, há códigos mínimos para interface do usuário e os parsers a serem criados. Este código fonte também é fornecido com o enunciado do trabalho no site da disciplina.

ANEXOS

Os códigos abaixo representam um interpretador mínimo, dado como ponto de partida. O interpretador implementado com estes fontes bases permite operações simples de soma envolvendo números de 0 a 9. Os fontes fazem parte do enunciado disponível no site, que acompanha ainda os arquivos Parser.frame e Scanner.frame específicos para cada parser que deve ser implementado.

extcalc.atg

```
COMPILER ExtCalc

public int result;

IGNORE '\t' + '\r' + '\n'

PRODUCTIONS

ExtCalc                         (. int v1 = 0, v2 = 0; .)
= Unidade<out v1>               (. result = v1; .)
  [ "mais" Unidade<out v2>      (. result += v2; .)
  ]
.

Unidade<out int v>              (. v = 0; .)
=   "zero"                      (. v = 0; .)
  | "um"                        (. v = 1; .)
  | "dois"                      (. v = 2; .)
  | "tres"                      (. v = 3; .)
  | "quatro"                    (. v = 4; .)
  | "cinco"                     (. v = 5; .)
  | "seis"                      (. v = 6; .)
  | "sete"                      (. v = 7; .)
  | "oito"                      (. v = 8; .)
  | "nove"                      (. v = 9; .)
.

END ExtCalc.
```

n2ext.atg

```
COMPILER N2Ext

public String result;

IGNORE '\t' + '\r' + '\n'

PRODUCTIONS

N2Ext                           (. String u = ""; .)
= Unidade<out u>                (. result = u; .)
.

Unidade<out String v>
                                (. v = ""; .)
=   "0"                         (. v = "zero"; .)
  | "1"                         (. v = "um"; .)
  | "2"                         (. v = "dois"; .)
  | "3"                         (. v = "tres"; .)
  | "4"                         (. v = "quatro"; .)
```

```
  | "5"                          (. v = "cinco"; .)
  | "6"                          (. v = "seis"; .)
  | "7"                          (. v = "sete"; .)
  | "8"                          (. v = "oito"; .)
  | "9"                          (. v = "nove"; .)
.

END N2Ext.
```

Interpret.java

```
import java.io.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.lang.StringBuilder;

public class Interpret
{
   public static String reverse(String s) {
      return new StringBuilder(s).reverse().toString();
   }

   public static void main(String argv[])
   {
      String str;
      System.out.println("Calculadora por extenso (digite 's' para sair)");
      System.out.println("Exemplos de expressoes: dois mais cinco, sete"
                         + " vezes (doze mais um)");
      while (true) {
         try {
            System.out.print("? ");
            BufferedReader bufferRead = new
                           BufferedReader(new InputStreamReader(System.in));
            str = bufferRead.readLine();
            if (str.equalsIgnoreCase("s"))
                    System.exit(0);
            InputStream stream = new
                                 ByteArrayInputStream(str.getBytes("UTF-8"));
            ExtCalc.Parser ecp =
                              new ExtCalc.Parser(new ExtCalc.Scanner(stream));
            ecp.Parse();
            String result = reverse(String.valueOf(ecp.result));
            stream = new ByteArrayInputStream(result.getBytes("UTF-8"));
            Num2Ext.Parser n2e =
                              new Num2Ext.Parser(new Num2Ext.Scanner(stream));
            n2e.Parse();
            System.out.println("  " + n2e.result);

        } catch(IOException e) {
            e.printStackTrace();
        }
     }
  }
}
```