

Compiladores, 2019/2020

Trabalho prático, parte 2

– Análise Sintática –

Fernando Lobo

Prazos e afins

- **Pontuação máxima:** 50 pontos
- **Prazo de entrega:** 13/Abr/2020, 23:59
- **Entrega:** pela mooshak, concurso **Comp1920**, problema D.
<http://mooshak.deei.fct.ualg.pt/~mooshak/>
- **Login mooshak:** o vosso login no mooshak deve ser **Gxx** onde xx é o vosso número de grupo.
- **Nota importante:** O nome do ficheiro que submetem deve chamar-se forçosamente `Triangle.zip` e deverá manter a estrutura de pastas tal e qual como no zip que vos foi fornecido. Isto é, após descompactado deverá existir uma pasta de nome `Triangle` e debaixo dela deverá haver as pastas `SyntacticAnalyzer` e `AbstractSyntaxTrees`.

1 Introdução

Este trabalho consiste em modificar a análise sintática (parsing) da linguagem de programação `Triangle`, acrescentando à linguagem o `SingleCommand for` e o `SingleCommand case`. A sintaxe do `for` tem duas variantes, podendo ser,

```
for V from E1 to E2 do C
```

ou,

```
for V from E1 downto E2 do C
```

em que V é um *Vname*, $E1$ e $E2$ são *Expression*, e C é um *SingleCommand* da linguagem `Triangle`. O ficheiro `Parser.java` contido na pasta `Triangle/SyntacticAnalyzer` tem métodos para fazer o parsing destes símbolos não terminais da gramática da linguagem `Triangle`.

A sintaxe do `case` é,

```
case E of
  IL1: C1;
  IL2: C2;
  ...
else: C0
```

em que *E* é uma *Expression*, *IL1*, *IL2*, ..., são *IntegerLiteral*, e *C0*, *C1*, *C2*, ..., são *SingleCommand* da linguagem Triangle. O caso `else:` é obrigatório e é forçosamente o último a aparecer.

Na página da tutoria electrónica está um ficheiro ZIP que contém os ficheiros relevantes para a realização do trabalho. Também está um ZIP (`test-programs.zip`) com alguns inputs para teste.

- `tp2-code-input-from-stdin.zip`
- `test-programs.zip`

O código contido em `tp2-code-input-from-stdin.zip` está preparado para ler do standard input, adequado para a submissão ao Mooshak.

Executar TestParser

Supondo que descompactam os ZIPs dentro de uma pasta chamada `tp2`, devem ir para a pasta `tp2` e executar o seguinte comando:

```
cd tp2-code-input-from-stdin
java Triangle/TestParser < ../test-programs/prog1.tri
```

2 Recomendação

Para a realização deste trabalho é muito importante a leitura do capítulo 4 do livro *Programming Language Processors in Java*. O apêndice B do mesmo livro é igualmente importante, para esta e para as fases subsequentes do trabalho.

3 Mooshak

O input para o Mooshak é uma sequência de caracteres que supostamente deverá corresponder a um programa que obedeça às regras sintáticas da linguagem Triangle estendida para contemplar os comandos `for` e `case`. O output consiste numa única linha com a mensagem `Parsing successful.` ou `Parsing not successful.`, seguido de um caracter de mudança de linha. Seguem-se alguns exemplos de input e output respectivo.

Input 1

```
! This program is useless
! except for illustration.
let
  var n: Integer;
  var c: Char
in
  begin
    c := '&';
    n := n+1;
    for i from 2 to n do
      putint(i)
    end
  end
```

Output 1

Parsing successful.

Input 2

```
let
  var i: Integer;
  var n: Integer
in
  begin
    getint(var n);
    for i from n downto 1 do
      begin
        putint(i*i);
        put(' ');
        i := i+1
      end
    end
  end
```

Output 2

Parsing successful.

Input 3

```
let
  var month: Integer;
  var days: Integer;
  var leap: Boolean
in
  begin
    leap := false;
    getint(var month);
    case month of
      2: days := if leap then 29 else 28;
      4: days := 30;
      6: days := 30;
      9: days := 30;
      11: days := 30;
      else: days := 31;
    putint(days)
  end
```

Output 3

Parsing successful.

Input 4

```
let
  var month: Integer;
  var days: Integer;
  var leap: Boolean
in
  begin
    leap := false;
    getint(var month);
    case month of
      6: days := 30;
      else: days := 31;
      9: days := 30;
    putint(days)
  end
```

Output 4

Parsing not successful.