

让我们构建一个简单的解释器。第 12 部分。 (<https://ruslanspivak.com/lbasi-part12/>)

日期 2016 年 12 月 1 日, 星期四

“不怕慢；只怕站着不动。” - 中国谚语。

您好，欢迎回来！

今天我们将采取更多的婴儿步骤，并学习如何解析 Pascal 过程声明。



什么是**程序声明**？**过程声明**是一个语言结构，其限定与帕斯卡代码块的标识符（进程名）和它关联。

在我们深入研究之前，先谈谈 Pascal 过程及其声明：

- Pascal 过程没有 return 语句。当它们到达相应块的末尾时退出。
- Pascal 过程可以相互嵌套。
- 为简单起见，本文中的过程声明将没有任何形式参数。但是，别担心，我们将在本系列的后面介绍。

这是我们今天的测试程序：

计划第 12 部分:

VAR

a : 整数;

程序 P1 ;

VAR

a : 真实;

k : 整数;

程序 P2 ;

VAR

a , z : 整数;

开始 {P2}

z := 777 ;

结束; {P2}

开始 {P1}

结束; {P1}

开始 {Part12}

a := 10 ;

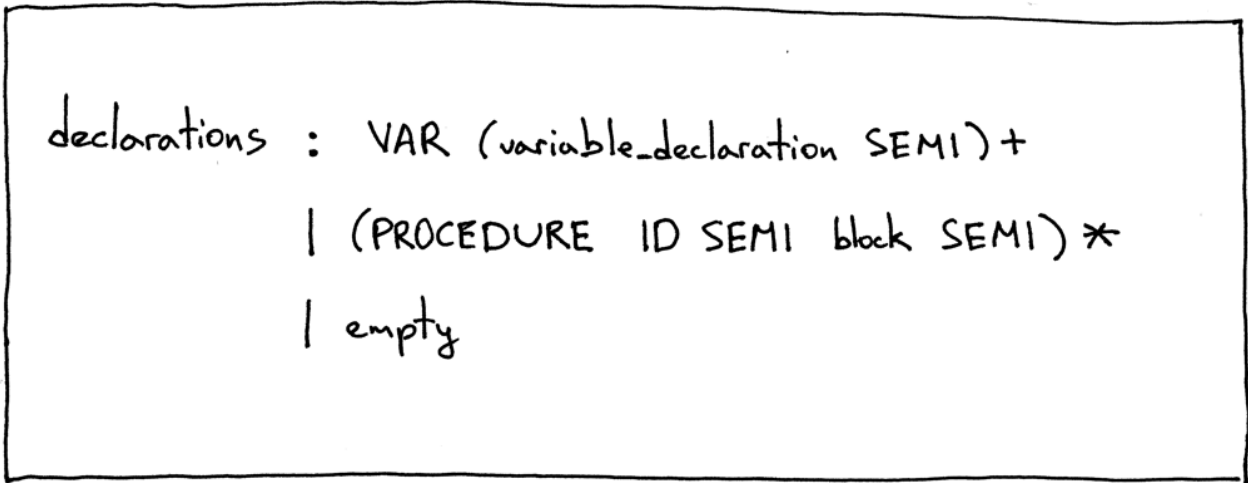
结束。 {Part12}

正如您在上面看到的，我们定义了两个过程（P1和P2）并且P2嵌套在P1 中。在上面的代码中，我使用带有过程名称的注释来清楚地指示每个过程的主体在哪里开始和结束。

我们今天的目标非常明确：学习如何解析这样的代码。

首先，我们需要对语法进行一些更改以添加过程声明。好吧，让我们这样做吧！

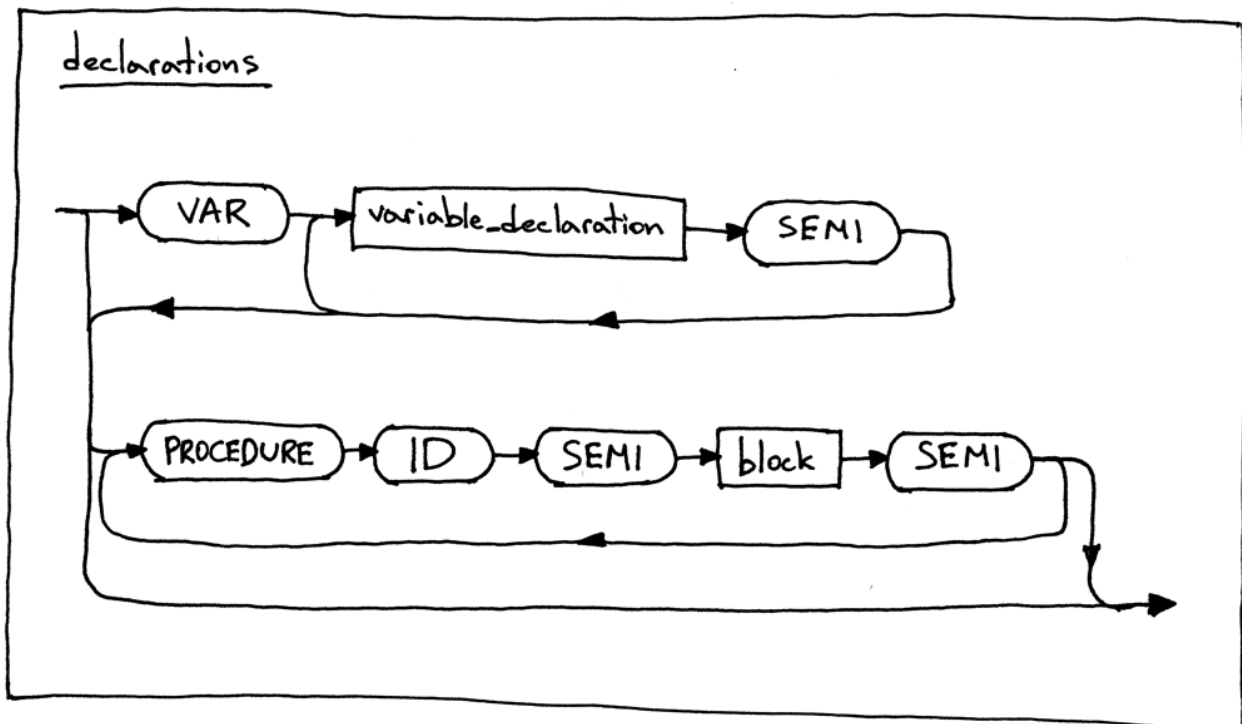
这是更新的声明语法规则：



declarations : VAR (variable_declaration SEMI)+
 | (PROCEDURE ID SEMI block SEMI)*
 | empty

过程声明子规则包含保留关键字**PROCEDURE**后跟一个标识符（过程名称），后跟一个分号，然后是一个块规则，以分号终止。哇！在这种情况下，我认为无论我在上一句话中放了多少字，图片实际上都值得！ :)

这是声明 规则的更新语法图：



从上面的语法和图表中您可以看到，您可以根据需要在同一级别上拥有任意数量的过程声明。例如，在下面的代码片段中，我们在同一级别定义了两个过程声明P1和P1A：

程序 测试：

VAR

a : 整数；

程序 P1 ；

开始 {P1}

结束； {P1}

程序 P1A ；

开始 {P1A}

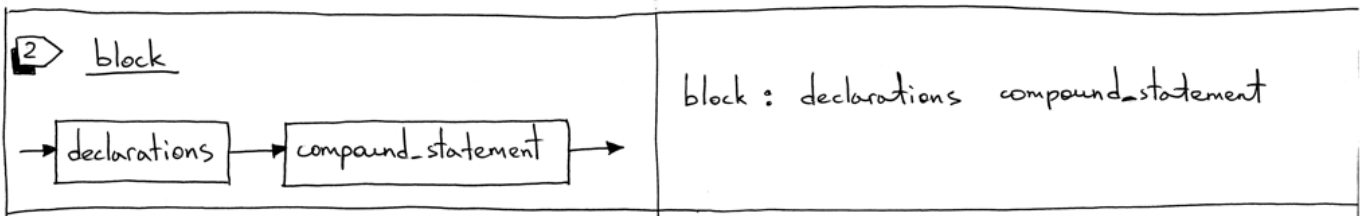
结束； {P1A}

开始 {测试}

a := 10 ；

结束。 {测试}

上面的图表和语法规则还表明过程声明可以嵌套，因为 过程声明子规则引用包含声明规则的块规则，而块规则又包含过程声明子规则。提醒一下，这里是Part10 中 (/lsbasi-part10/)块规则的语法图和语法： (/lsbasi-part10/)



好的，现在让我们关注需要更新以支持过程声明的解释器组件：

更新词法分析器

我们需要做的就是添加一个名为**PROCEDURE**的新令牌：

```
程序 = '程序'
```

并将 “PROCEDURE” 添加到保留关键字。这是保留关键字到标记的完整映射：

```
RESERVED_KEYWORDS = {
    'PROGRAM' : 令牌( 'PROGRAM' , 'PROGRAM' ),
    'VAR' : 令牌( 'VAR' , 'VAR' ),
    'DIV' : 令牌( 'INTEGER_DIV' , 'DIV' ),
    'INTEGER' : 令牌( 'INTEGER' , 'INTEGER' ),
    'REAL' : 令牌( 'REAL' , 'REAL' ),
    'BEGIN' : 令牌( '开始' , '开始' ),
    'END' : 令牌( 'END' , 'END' ),
    'PROCEDURE' : 令牌( 'PROCEDURE' , 'PROCEDURE' ),
}
```

更新解析器

以下是解析器更改的摘要：

1. 新的ProcedureDecl AST 节点
2. 更新解析器的声明方法以支持过程声明

让我们回顾一下这些变化。

1. 所述ProcedureDecl AST节点代表一个程序声明。类构造函数将过程名称和过程名称引用的代码块的AST节点作为参数。

```
类 ProcedureDecl ( AST ):
    def __init__( self , proc_name , block_node ):
        self . proc_name = proc_name
        自我 . block_node = block_node
```

2. 这是Parser 类的更新声明方法

```

def 声明( self ):
    """declarations : VAR (variable_declaration SEMI)+
                        / (PROCEDURE ID SEMI block SEMI)*
                        / 空的

    """
    声明 = []

    如果 自. current_token . 类型 == VAR :
        自我. 吃 (VAR )
        而 自我. current_token . 类型 == ID :
            var_decl = self . variable_declaration ()
            声明. 扩展 (var_decl )
            自我. 吃 (半)

    而 自我. current_token . 类型 == 程序:
        自我. 吃 (程序)
        proc_name = self . current_token . 重视
        自我. 吃 (ID )
        自我. 吃 (半)
        block_node = self . block ()
        proc_decl = ProcedureDecl ( proc_name , block_node )
        声明. 追加 (proc_decl )
        自我. 吃 (半)

    返回 声明

```

希望上面的代码是不言自明的。它遵循您在本文前面看到的过程声明的语法/语法图。

更新符号表构建器

因为我们还没有准备好处理嵌套过程范围，所以我们将简单地向SymbolTreeBuilder AST访问者类添加一个空的visit_ProcedureDecl方法。我们将在下一篇文章中填写。

```

def visit_ProcedureDecl ( self , node ):
    通过

```

更新解释器

我们还需要向Interpreter类添加一个空的visit_ProcedureDecl方法，这将导致我们的解释器默默地忽略我们所有的过程声明。

到现在为止还挺好。

现在我们已经进行了所有必要的更改，让我们看看抽象语法树在新的ProcedureDecl 节点中的样子。

这里又是我们的 Pascal 程序（你可以直接从[GitHub](https://github.com/rspivak/lbasi/blob/master/part12/python/part12.pas) (<https://github.com/rspivak/lbasi/blob/master/part12/python/part12.pas>) 下载）：

计划第 12 部分；

VAR

 a : 整数；

程序 P1 ；

VAR

 a : 真实；

 k : 整数；

程序 P2 ；

VAR

 a , z : 整数；

开始 {P2}

 z := 777 ；

结束； {P2}

开始 {P1}

结束； {P1}

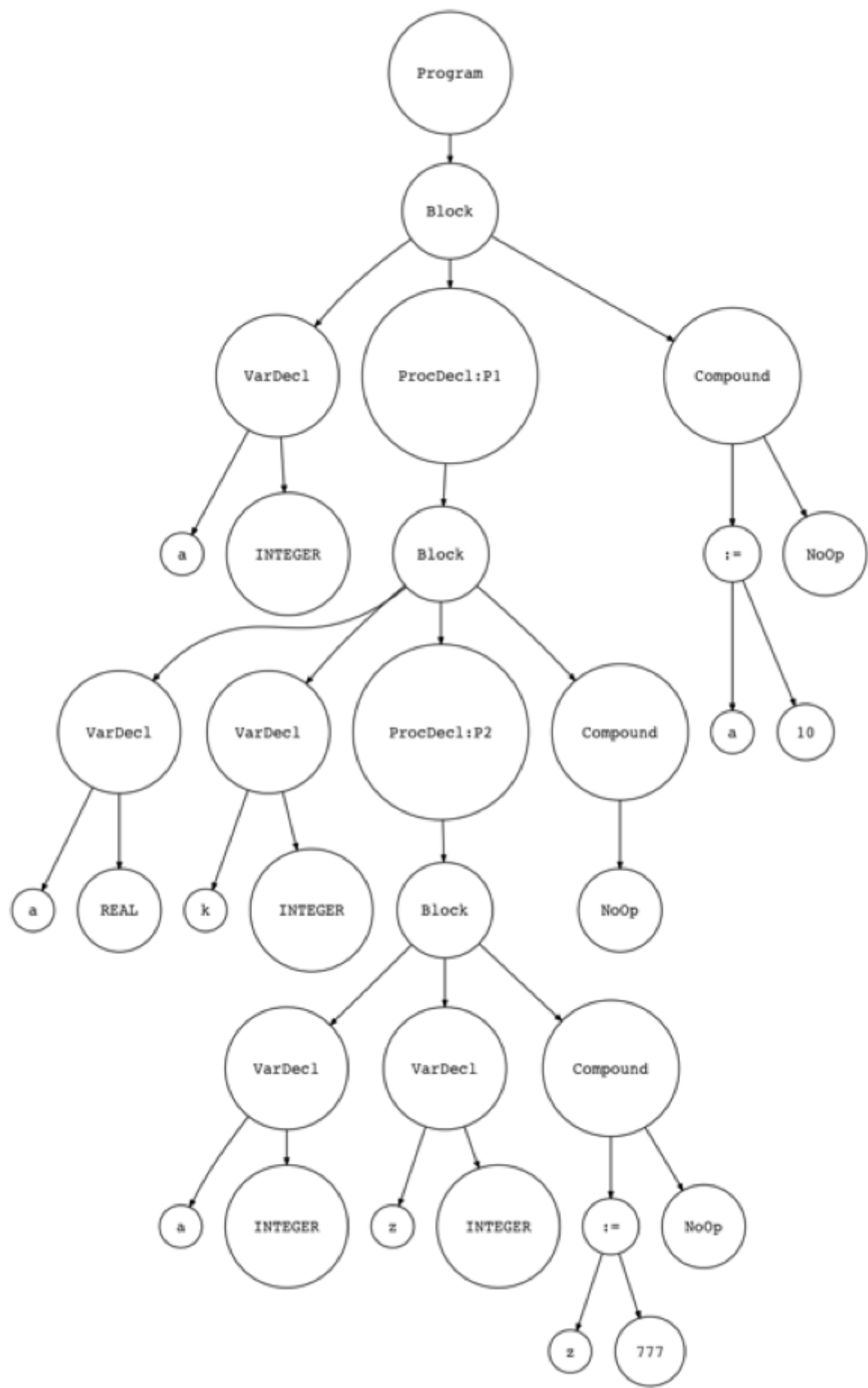
开始 {Part12}

 a := 10 ；

结束。 {Part12}

让我们生成一个AST并使用[genastdot.py](https://github.com/rspivak/lbasi/blob/master/part12/python/genastdot.py) (<https://github.com/rspivak/lbasi/blob/master/part12/python/genastdot.py>) 实用程序对其进行可视化：

```
$ python genastdot.py part12.pas > ast.dot && dot -Tpng -o ast.png ast.dot
```



在上图中，您可以看到两个ProcedureDecl节点： ProcDecl:P1和ProcDecl:P2，它们对应于过程P1和P2。任务完成。 :)

作为今天的最后一项，让我们快速检查一下，当 Pascal 程序中有过程声明时，我们更新的解释器是否像以前一样工作。如果您还没有下载解释器 (<https://github.com/rspivak/lbasi/blob/master/part12/python/spi.py>) 和测试程序 (<https://github.com/rspivak/lbasi/blob/master/part12/python/part12.pas>)，请下载并在命令行上运行它。您的输出应该类似于：

```
$ python spi.py part12.pas
定义： 整数
定义： 真实
查找： 整数
定义： <a:INTEGER>
查找： 一个

符号表内容：
符号： [ INTEGER, REAL, <a:INTEGER> ]

运行时 GLOBAL_MEMORY 内容：
一 = 10
```

好的，有了所有这些知识和经验，我们已经准备好处理我们需要理解的嵌套作用域的主题，以便能够分析嵌套过程并准备好处理过程和函数调用。这正是我们在下一篇文章中要做的：深入研究嵌套作用域。所以下次不要忘记带上你的游泳装备！请继续关注，我们很快就会见到你！

如果您想在收件箱中获取我的最新文章，请在下方输入您的电子邮件地址，然后单击“获取更新”！

输入您的名字 *

输入您最好的电子邮件 *

获取更新！

本系列所有文章：

- [让我们构建一个简单的解释器。第1部分。 \(/lbasi-part1/\)](#)
- [让我们构建一个简单的解释器。第2部分。 \(/lbasi-part2/\)](#)
- [让我们构建一个简单的解释器。第 3 部分。 \(/lbasi-part3/\)](#)

- [让我们构建一个简单的解释器。第 4 部分。 \(/lsbasi-part4/\)](#)
- [让我们构建一个简单的解释器。第 5 部分。 \(/lsbasi-part5/\)](#)
- [让我们构建一个简单的解释器。第 6 部分。 \(/lsbasi-part6/\)](#)
- [让我们构建一个简单的解释器。第 7 部分：抽象语法树 \(/lsbasi-part7/\)](#)
- [让我们构建一个简单的解释器。第 8 部分。 \(/lsbasi-part8/\)](#)
- [让我们构建一个简单的解释器。第 9 部分。 \(/lsbasi-part9/\)](#)
- [让我们构建一个简单的解释器。第 10 部分。 \(/lsbasi-part10/\)](#)
- [让我们构建一个简单的解释器。第 11 部分。 \(/lsbasi-part11/\)](#)
- [让我们构建一个简单的解释器。第 12 部分。 \(/lsbasi-part12/\)](#)
- [让我们构建一个简单的解释器。第 13 部分：语义分析 \(/lsbasi-part13/\)](#)
- [让我们构建一个简单的解释器。第 14 部分：嵌套作用域和源到源编译器 \(/lsbasi-part14/\)](#)
- [让我们构建一个简单的解释器。第 15 部分。 \(/lsbasi-part15/\)](#)
- [让我们构建一个简单的解释器。第 16 部分：识别过程调用 \(/lsbasi-part16/\)](#)
- [让我们构建一个简单的解释器。第 17 部分：调用堆栈和激活记录 \(/lsbasi-part17/\)](#)
- [让我们构建一个简单的解释器。第 18 部分：执行过程调用 \(/lsbasi-part18/\)](#)
- [让我们构建一个简单的解释器。第 19 部分：嵌套过程调用 \(/lsbasi-part19/\)](#)

注释

ALSO ON RUSLAN'S BLOG

Let's Build A Web Server. Part 2.

6 years ago • 61 comments

Remember, in Part 1 I asked you a question: "How do you run a Django application, ...

Let's Build A Simple Interpreter. Part 2.

6 years ago • 16 comments

In their amazing book "The 5 Elements of Effective Thinking" the authors ...

Let's Build A Simple Interpreter. Part 18: ...

2 years ago • 8 comments

Do the best you can until you know better. Then when you know better, do ...

Let's B Interpreter

4 years a

Only dea
flow. As
last artic

[29 Comments](#)[Ruslan's Blog](#)[Disqus' Privacy Policy](#)[Login](#)[Recommend 2](#)[Tweet](#)[Share](#)[Sort by Best](#)[LOG IN WITH](#)[OR SIGN UP WITH DISQUS](#)[Subscribe](#) [Add Disqus to your site](#) [Add DisqusAdd](#) [Do Not Sell My Data](#)

🏠 社会的

[github \(https://github.com/rspivak/\)](https://github.com/rspivak/)[推特 \(https://twitter.com/rspivak\)](https://twitter.com/rspivak)[链接 \(https://linkedin.com/in/ruslanspivak/\)](https://linkedin.com/in/ruslanspivak/)

🏠 热门帖子

[让我们构建一个 Web 服务器。第1部分。 \(https://ruslanspivak.com/lsbaws-part1/\)](https://ruslanspivak.com/lsbaws-part1/)[让我们构建一个简单的解释器。第1部分。 \(https://ruslanspivak.com/lsbasi-part1/\)](https://ruslanspivak.com/lsbasi-part1/)[让我们构建一个 Web 服务器。第2部分。 \(https://ruslanspivak.com/lsbaws-part2/\)](https://ruslanspivak.com/lsbaws-part2/)[让我们构建一个 Web 服务器。第 3 部分。 \(https://ruslanspivak.com/lsbaws-part3/\)](https://ruslanspivak.com/lsbaws-part3/)[让我们构建一个简单的解释器。第2部分。 \(https://ruslanspivak.com/lsbasi-part2/\)](https://ruslanspivak.com/lsbasi-part2/)

免责声明

这个网站上的一些链接有我的亚马逊推荐 ID，它为我提供了每次销售的小额佣金。感谢您的支持。

