

# parser-stage 实验报告

秦若愚 2019011115

## 实验目标

- 完成一个手工实现的递归下降语法分析器

## 实验内容

下面依次介绍以下函数的实现：

```
p_relational p_logical_and p_assignment p_expression p_statement p_declaration  
p_block p_if p_return p_type
```

### p\_relational

与 relational 对应的语法等价的 EBNF 文法为：

```
relational: additive { '<' additive | '>' additive | '<=' additive | '>='  
additive }
```

参考 p\_equality 即可写出对应实现。

### p\_logical\_and

与 logical\_and 对应的语法等价的 EBNF 文法为：

```
logical_and: equality { '&&' equality }
```

参考 p\_logical\_or 即可写出对应实现。

### p\_assignment

当发现下一个 token 是 Assign 时，先使用 lookahead 读入 Assign，再调用 p\_expression() 函数解析 expression，最后构建 Assignment 结点并返回该结点。

### p\_expression

调用 p\_assignment() 函数解析 assignment，并返回结果。

### p\_statement

self.next 是 Semi 或者属于 p\_expression.first 的情况已给出。当 self.next 为 If 时，调用 p\_if() 函数解析；当 self.next 为 Return 时，调用 p\_return() 函数解析。

## p\_declaration

首先构建 `Declaration` 结点并设置 `var_t` 和 `ident`。当发现下一个 token 是 `Assign` 时，先使用 `lookhead` 读入 `Assign`，再调用 `p_expression()` 函数解析 `expression`，然后将 `Declaration` 结点的 `init_expr` 设置为解析 `expression` 的返回值。最后返回 `Declaration` 结点。

## p\_block

在 `p_block_item()` 中，若发现接下来是 `statement` 则调用 `p_statement()` 函数解析，若接下来是 `declaration` 则调用 `p_declaration()` 函数解析，否则报错。最后返回解析结果。其余部分已给出。

## p\_if

依次读入或解析 `If`，`LParen`，`cond`（调用 `p_expression()` 函数），`RParen`，`then`（调用 `p_expression()` 函数），然后构建 `If` 结点。若 `self.next` 是 `Else`，则继续读入并解析之后的 `otherwise`（调用 `p_expression()` 函数），将结果赋给 `If` 结点的 `otherwise`。最后返回 `If` 结点。

## p\_return

依次读入或解析 `Return`，`expression`（调用 `p_expression()` 函数），`Semi`，然后构建 `Return` 结点并返回该结点。

## p\_type

读入 `Int`，构建 `TInt` 结点并返回该结点。

## 思考题

1. 答：

```
additive : multiplicative P
P : Q multiplicative P
    | ε
Q : '+'
    | '-'
```

其中 `P` 和 `Q` 为新增加的非终结符。

2. 答：

出错例子如下：

```
int main() {
    int a = 1;
    a = a ++ a + 1;
    return a;
}
```

仅考虑解析 `a ++ a + 1` 的部分，文法如下所示。

```
additive : ident P
P : '+' ident P
    | ε
```

其中 `additive` 和 `P` 为非终结符，终结符 `ident` 为任意已声明变量。

首先匹配并读入 `a`，进入 `ParseP()`，然后先匹配 `+`，之后在匹配 `ident` 时遇到 `+`，失败。此时 `BeginSym = { ident }`，`EndSym = {  $\epsilon$  }`，因此补救符号集合  $S = \{ ident, \epsilon \}$ 。跳过不属于  $S$  的 `+` 后匹配 `a` 成功，从而可以继续解析。

3. 实验代码和课堂上讲授的方法不完全一致，不太能帮助理解课堂上的方法。