《编译原理》第一次实验

词法和语法分析

李泽昆(161158028) 杜映潇(161130015)

一、完成情况

完成了必做内容和选做要求 1.1，实现了以下功能：

- 对于没有词法和语法错误的输入文件，能够按照要求构造并打印语法树
- 能够查出.cmm 文件中存在的词法错误和语法错误
- 能够识别八进制数和十进制数

二、实现方法

（1） 语法树的数据结构

首先考虑语法树是一颗多叉树，而且其子女个数预先是无法确定的，因此考虑使用数据结构课程上所学到的“子女-兄弟”表示法来表示语法树，具体思路为：在定义树的节点时，每个节点有两个指针，其中一个指向其第一个子女节点，另外的一个节点指向当前节点的下一个兄弟节点，如下图所示：

```
17 typedef struct GrammarTreeNode {
18     struct GrammarTreeNode *child; /* Pointer pointing to the first child of this node */
19     struct GrammarTreeNode *sibling; /* Pointer pointing to the sibling of this node */
20     Lexeme lexeme;
21 } Node;
```

使用这种定义方式的好处有二，首先我们能够通过沿着根节点的子女节点的兄弟节点一直遍历完整颗树，不论采用前序、中序还是后序遍历，实现起来非常的简单直接，其次使用这种方式具有良好的扩展性，因为兄弟节点理论上可以无限延伸，因此对于某个节点来说，我们可以插入任意数量的子节点。

另外，在我们看来，语法树需要能够存储各个词法单元，为此，我们另外设计了一个名为“Lexeme”的结构体，如下图所示：

```
11 typedef struct LexemeInNode {
12     char type[CAPACITY];
13     char value[CAPACITY];
14     int linenum;
15 } Lexeme;
```

在该结构体中我们定义了每一个词法单元的类型、具体取值和所在行号，具体在词法分析时便会对这些词法单元进行初始化

（2） 词法错误的识别

在完成词法错误识别的过程中，我们的处理思路是在匹配规则的最后添加通配符“.”来匹配所有不能被识别为有效词法单元的词素，并报错，但是在测试过程当中我们发现即使在前面的匹配规则中我们加入了对空白符号的匹配，最后一行的通配符仍然会匹配到换行符并造成信息的打印出现错误。后来当我们通过 Shell 的 IO 重定向功能将程序的输出打印到文件中，并使用 Vim 打开该输出文件后，发现文档中出现了“^M”这样的奇怪符号。通过之前课程的学习以及网上资料的查询，我们了解到在 Windows 系统和 Linux 系统中，文件中的换行存在差异，在 Windows 中为“\n\r”，而在 Linux 中为“\n”，于是我们更改了空白符的匹配规则，如下图所示：

```
52 SPACE [ \t\r]
```

之后程序运行正常。

（3） 语法错误的识别

对于正确的.cmm，程序可以通过 syntax.y 中所定义的语法规则和行为识别。而对于可能存在的语法错误，我们采取了在语法中添加含 error 的产生式的方法。将 SEMI,RC,RP,RB 等作为错误恢复的同步符号(具体请见 syntax.y 代码)。通过查阅 GNU 文档，我们在 syntax.y 中添加了下面的命令：

```
%define parse.error verbose
%define parse.lac none
```

从而程序能够自动地为查出的语法错误添加合理的提示信息。

在上述工作后，我们对程序的功能进行了测试，发现对于代码中缺少分号、右括号等的语法错误，在报错时会将错误出现的位置显示为缺少分号的下一行。我们通过 Google 查询了很多资料，也没有找到合适的解决方法。同时，我们还注意到，对于这样的语法错误，gcc 的行为与我们的程序是一致的，也是将报错的位置显示为缺少分号的下一行，不知道有没有方法可以解决这一问题。

（4） 优先级与结合性

为了避免二义性所带来的冲突，我们需要按照 C—的文法为运算符规定优先级与结合性。绝大多数运算符是容易处理的，只需要按照附录 A 中规定的优先级与结合性，使用%left 等声明直接“翻译”即可。只有 MINUS 终结符比较特殊，它既可以被理解为“减号”，也可以被理解为“负号”，且这两个运算符的优先级和结合性都是不同的。

通过查询 GNU 文档，我们通过为“负号”定义 UMINUS 算符的方法，来区别“减号”，如下图所示：

```
| MINUS Exp %prec UMINUS {
    $$ = initNode("Exp", " ", @$.first_line);
    addChild($$, $1);
    addChild($$, $2);
}
```

并对其优先级和结合性进行了规定：

```
%left PLUS MINUS
%left STAR DIV
%right NOT UMINUS
```

三、编译方法

执行 `make parser` 命令即可完成程序的编译。

我们没有修改框架代码中给定的 `Makefile` 文件。