编译原理lab1实验报告

181180050 孔孟荀

1571589383@qq.com

1.程序实现了哪些功能

词法分析、语法分析，包含实验要求里关于八进制、十六进制整数和科学计数法的浮点数以及注释的选做功能。

1. 文件结构

   - lexical.l:用flex实现词法分析的源码
   - syntax.y: 用bison实现的语法分析的源码
   - makefile
   - tree.c、tree.h：有关语法分析树的相关功能的定义与实现以及打印语法树的实现
   - main.c: 主函数

2. 词法分析
   主要功能是分析读入的文件，获得词法单元**token**并返回给语法分析器。
   在词法分析的过程中主要使用了正则表达式。书写出正确的正则表达式是这个过程的主要障碍。
   - 比如，当我想要为int型整数书写正则表达式时，我一开始书写的[0-2147483647]，我一开始以为他的效果是匹配0-2147483647中的任意一个数字，后来才发现并非如此，只能匹配一位数。
     这里我受到了附录“你可以假设每个整型数不超过32bits位”的误导，以为这个操作应该在正则表达式这一步完成，其实是放在flex代码的**规则部分**完成，也正是在**规则部分**中，返回了**token**给语法分析器。
   - 在flex代码的**定义部分**我为一些正则表达式起了别名以方便规则部分的代码书写。
   - 值得注意的是有一些词法单元和正则表达式的特殊符号冲突，需要用双引号特别处理

3. 语法分析

获得了词法分析的token以后，通过bison来完成语法分析的功能。在理论课已经学过正则表达式不足以满足语法分析的需要。bison采用自底向上的LALR(1)分析技术,我要做的主要是把产生式在规则部分书写出来。

- 首先要在定义部分定义出token，在这里涉及到运算符时，要按照附录的要求依照顺序定义，来实现**运算符的结合性和优先级**
- 为了解决悬空else的问题，按照教程说的

> 通过定义一个比ELSE优先级更低的LOWER_THAN_ELSE算符，降低了归约相对于移入ELSE的优先级'

4.语法分析树

完成上述基本功能后开始构建语法分析树，这部分内容在tree.c和tree.h中实现。

```
typedef struct _Node{
int childnum;
int lineno;
int type;
char *name;
union Val{
    int i;
    float f;
    char *s;
}val;
struct _Node* childlist[10];
}Node;
```

定义了这样的多叉树节点，每个节点存储了一些语法信息，用来在打印过程中方便输出。

在这里用了一个联合结构val，来解决节点携带的值的类型不同的问题。

除此之外，还实现了多叉树节点的分配和插入等函数：**malloc_node**和**add**

有了这两个函数，给语法树节点的相关操作提供了统一的接口，使得代码清晰明了。

回过头在bison代码中把词法单元的类型设置为token方便打印。

在bison代码的语义动作部分，通过这两个函数构造非终结符的节点并把它们链接，对于一些终结符，在flex代码的动作部分，也通过这两个函数构造语法树节点。

5. 打印语法分析树

打印过程通过一个先序遍历完成，对于缩进的要求也在先序遍历中通过函数print_space实现。

6. 选做部分

主要也是实现了一些正则表达式，对于注释/* */,用到了flex的库函数input()来保证一致性。

7. 错误打印

在bison的用户自定义代码部分重写了yyerror，并通过开启%define parse.error verbose来打印

一些错误信息。
8.main函数
main函数中通过全局变量error的值来判断是否发生了错误，假如没有发生再调用先序遍历打印语法分析树。

2.程序应该如何被编译

解压我提供的源码的压缩文件夹后，在该文件夹内执行make，会生成可执行程序parser，通过./parser xx.cmm即可对文本文件进行词法分析和语法分析。

3.总结

总体而言实现过程难度适中，就是在定义写词法单元语法单元的规则的时候有一些活比较重复机械，不会写脚本，导致手动复制粘贴了很久。定义部分代码写的有点凌乱。
在写syntax.y的位置相关代码的时候没注意到最前面要开启选项%locations，导致查了很久为什么yyltype类型没有被定义的bug。