编译系统实验报告

实验 1 – 词法分析与语法分析

程序实现的功能及如何实现：

程序主要实现的功能如下：

1) 能够读入程序文本文件，并识别出其中的词法错误与语法错误。
2) 对于程序中的词法错误（错误类型 A）：出现 C--词法中未定义的字符以及任何不符合 C--词法单元定义的字符。在遇到此类错误时，输出报错提示：
   Error type A at Line [行号]: [说明文字].
3) 对于程序中的语法错误（错误类型 B），在遇到此类错误时，输出报错提示：
   Error type B at Line [行号]: [说明文字].
4) 如果程序中没有任何词法或语法错误的输入文件，则构造出先序遍历语法树，打印每一个结点的信息。

具体的实现方法简要介绍如下：

1) 语法分析程序的入口点是 yyparse()。当程序调用 yyparse( )时,语法分析程序就试图分析输入流，如图所示，展示了读入程序文本的整个过程。

```
FILE *f = fopen(argv[i], "r");
if (!f)
{
    perror(argv[i]);
    return 1;
}
yyrestart(f);
yyparse();
fclose(f);
```

2) 语法分析树节点结构。
   设计结构体 ASTnode 来构建语法分析树。在 ASTnode 中，储存着当前节点所对应的行数，token 类型，节点类型（如果是 int，float 则保存具体数值）。词法分析过程中，在 lab1.l 文件中生成叶节点（终结符），语法分析过程中，在 lab1.y 文件中生成其他节点（非终结符），从而完成 AST 的构建。

```
// AST结构体
typedef struct ASTnode{
    int line;  // 行数
    char* name;  // token类型
    union{
        // 用联合体保存id或type（int/float）
        char* id_type;
        // int/float具体的数值（如果有）
        int int_value;
        float float_value;
    };
    struct ASTnode *fchild,*next;
// fchild：第一个孩子节点，next：兄弟节点
}* Ast,* tnode;
```

3) AST 的构建

首先通过 C 库宏 va_start()读取参数，然后根据当前父节点的子节点个数及存在情况进行如下不同的操作：若当前节点还有子节点，则通过 va_arg()检索下一个参数，作为第一个子节点，并通过函数 setChildTag() 修改相应状态；若当前节点是终结符或者 null，则将对应的值存入节点的 union 联合体中。由于代码篇幅较长，故在此处不作展示。

4) 先序遍历打印节点。

用递归的方法实现递归遍历，在遍历的过程中，输出当前节点的 type（由于代码篇幅较长此处不作展示）。

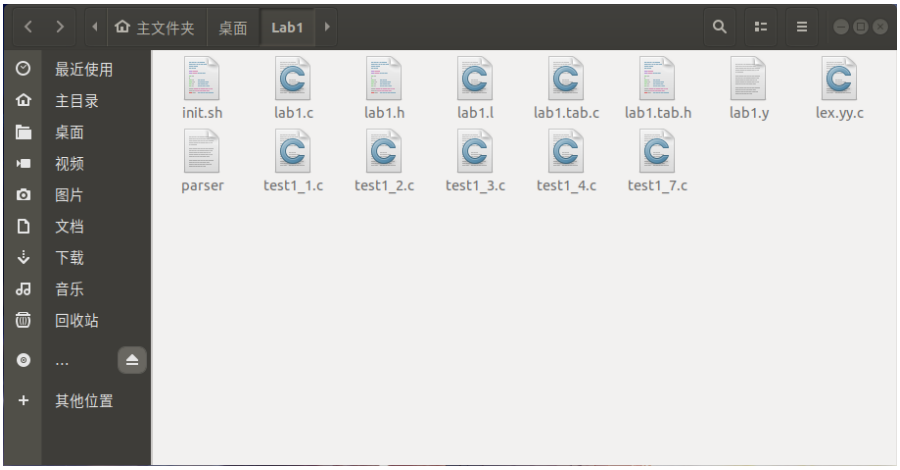
5) 用正则表达式完成不同类型数据的表示。具体如图所示：

```
/*二进制*/
INT_BIN 0[bB][01]+
/*八进制*/
INT_OCT 0[1-7][0-7]*
/*十进制*/
INT_DEC 0|[1-9][0-9]*
/*十六进制*/
INT_HEX 0[xX][a-fA-F0-9]+
/*INT类型*/
INT {INT_BIN}|{INT_OCT}|{INT_DEC}|{INT_HEX}
/*以科学计数法表示浮点数*/
FLOAT ((([0-9]+\.[0-9]*)|([0-9]*\.[0-9]+)|INT)[Ee][-+]?[0-9]+)|({INT}\.[0-9])

/*标识符*/
ID [a-z_A-Z][a-z_A-Z0-9]*
```

如何编译程序：

最终的代码文件如图所示：



其中：

lab1.c, lab1.h是包含AST结构体，函数定义以及部分错误处理的C文件。

lab1.tab.h, lab1.tab.c, lex.yy.c是中间过程生成的文件。

test1_x.c是测试文件，对应着指导书中的示例1.2~1.4，1.7。

parser 是gcc编译后得到的可执行文件。

执行以下几条命令以编译程序：

bison -d lab1.y

flex lab1.l

gcc lab1.tab.c lab1.c lex.yy.c -lfl -ly -o parser

在本实验中，我将以上命令写入脚本，在Linux下使用时，切入lab1文件夹，直接执行 init.sh 便可完成程序编译。

完成之后，输入 ./parser textfile 便可以用程序对进行词法分析与语法分析。

程序亮点：

1) 完成了部分附加功能：可以识别出以科学计数法形式表示的float的值。其正则表达式如下：

```
/*以科学计数法表示浮点数*/
FLOAT ((([0-9]+\.[0-9]*)|([0-9]*\.[0-9]+)|INT)[Ee][-+]?[0-9]+)|({INT}\.[0-9])
```

该部分的测试用例为test1_7.c（对应指导书的测试用例1.7），运行结果如下：

```
mincoolee@mincoolee:~/桌面/Lab1$ ./parser test1_7.c
Program(1)
  ExtDefList(1)
    ExtDef(1)
      Specifire(1)
        TYPE: int
      FunDec(1)
        ID: main
        LP(1)
        RP(1)
      Compst(1)
        LC(1)
        DefList(2)
          Def(2)
            Specifire(2)
              TYPE: float
            DecList(2)
              Dec(2)
                VarDec(2)
                  ID: i
                ASSIGNOP(2)
                Exp(2)
                  FLOAT: 0.000105
            SEMI(2)


        RC(3)
```