编译原理实验一报告

周羽萱 211220074 1813156367@qq.com

一. 实现功能

所有的必做+第 1 个选做功能，具体如下：

(1) 使用 flex 完成词法分析：使用正则表达式识别 int,float,id，8 进制数，16 进制数等词法单元。

(2) 识别错误的词法单元：通过正则表达式识别词法错误。遇到错误时仍返回对应的词法单元（如遇到错误形式的 float,仍 return float）,防止语法分析二次报错。

(3) 使用 bison 完成语法分析，包括对负号的处理等。

(4) 完成语法树的构建。

(5) 完成语法错误的识别。

(6) 本来完成了所有的选做功能，但是因为 oj 要求对 2,3 组的选做报错，所以修改了。

亮点：

可以报出错误类型，输出提示信息。例如：

遇到注释：

```
Error type B at Line 1: Can't recognize the annotation
```

遇到错误的 float:

```
Error type A at Line 2: Wrong type of float near: .1424.
```

遇到指数形式的 float:

```
Error type A at Line 2: Can't recognize the exponential form of float near: 1.05e-4.
```

遇到错误的 8 进制数：

```
Error type A at Line 2: Wrong type of oct near: 09.
```

……

二. 程序编译

1. flex lexical.l
2. bison -t syntax.y
3. gcc -w main.c syntax.tab.c common.c -lfl -ly -o parser

三. 代码实现

因为 lexical.l 和 syntax.y 两个文件中都需用到添加节点的操作，因此我把构建节点，生成树，打印树的内容放在 common.h 和 common.h 中，方便引用。

1. 添加节点

   每个节点的数据结构如下：

```
typedef struct
{
    char name[35];  //每个节点的名字
    data_type type; //每个节点的类型
    int line;       //每个节点的行号
    int token_flag; //是否是终结符
    union
    {
        int val_int;//如果是int,就保存它的值
        float val_float;//如果是float,就保存它的值
        char val_id[35];//如果是id,就保存它的值
    } val;                //节点的值
    struct Node *child;   //每个节点的孩子节点
    struct Node *brother; //每个节点的兄弟节点
} Node;
```

```
Node *add_node(char *name, char *val, data_type type, int token_flag, int line, int num, ...);
```

   为了使函数能够接受可变数量的参数,这里使用 valist， num 表示添加子节点的数量。

2. 打印语法树

```
void print_AST(Node *root, int depth)
```

其余部分均是按照指导手册完成的，没什么特别之处，不再赘述。

四． 总结

本次实验主要在语法分析部分出错，尤其是处理错误部分，总是不符合预期的输出，因此 de 了很久的 bug。本次实验我学会了 flex 和 bison 两个工具的使用，还是很有收获的。感谢助教哥哥批改！