编译原理实验报告

实验名称：实验一 词法分析与语法分析

指导教师：许畅

实验时间：2023.3.11

任务号：14

报告人：

分配	姓名	学号
组长	张铭俊	201220065
组员	吴浩然	201220064

一、程序功能

必做内容

- 词法分析：找到词法错误（错误类型A）：即出现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
- 语法分析：找到语法错误（错误类型B）
- 在程序通过词法分析与语法分析后输出语法分析树

选做内容

- 识别八进制数和十六进制数。若输入文件中包含符合词法定义的八进制数（如0123）和十六进制数（如0x3F），得出相应的词法单元；若输入文件中包含不符合词法定义的八进制数（如09）和十六进制数（如0x1G），给出输入文件有词法错误（即错误类型A）的提示信息。
- 实现了浮点数的指数形式，形如1.05e-4，能够给出对应的词法单元，若输入的指数形式不合词法，也会给出错误信息。

二、程序运行

1. 程序划分

程序主要划分为了以下几个文件

- lexical.l：词法分析文件，用于给出对应词法的正则表达式和匹配时的动作，能够生成lex.yy.c
- syntax.y: 语法分析文件，用于给出语法的产生式以及产生式推导时的对应动作，能够生成syntax.tab.h及syntax.tab.c
- main.c: 主函数
- tree.c: 存放全局变量如firstNode(语法分析树头节点)，语法分析树节点结构体Node等等。

2. 程序实现

1. 语法分析树节点结构体

   该结构体设计思路为产生式右边部分为左边部分的子女，考虑到子女个数不能确定，并不满足二叉树建树，故采用森林建树法（兄弟-子女建树法），增添child与nextsibling指针，同时对于树节点增添名字保存语法单元或词法单元名称，用val、type_int、type_float构成的union来存放需要额外打印的信息，val存放ID和TYPE的额外打印字符串信息，type_int存放int值，

type_float存放float值，引入额外的判断是否为词法单元iflexcial（语法单元需要打印输出行数），整体结构体代码如下：

```
typedef struct Node{
    int lineno;
    char* name;
    union{
        char* val;  //for TYPE and ID
        int type_int;//for INT
        float type_float;//for FLOAT
    };
    int iflexcial;
    struct Node* child;
    struct Node* nextsibling;
}tNode;
```

2. 添加节点

在语法分析与词法分析中，每当匹配成功，则调用addNode(tNode* node,int num,...)来将该节点放到语法分析树中，...为可变参数，在num>0时为num个tnode*节点，存放其子女节点，注意，由于采用自底向上的语法分析树构造，所以此处的子女节点可以默认采用产生式右部的$1等等，在num=0时，意为匹配的为词法单元，此时右边参数为该词法单元所在的行数，该词法单元节点为叶子节点。

```
tNode* addNode(char* name,int num,...);
```

3. 先序遍历语法分析树并打印

遍历方式引入额外的参数level，用来指定该节点所在的语法分析树层数并打印前面的空格，对于当前的节点，通过其name进行分类，进而选择额外的打印信息，采用函数为dfs(tNode* head,int level);

```
void dfs(tNode* head,int level);
```

4. main函数书写

通过实验指导指导，在成功打开文件后，先通过yyrestart(f)进行词法分析，随后通过yyparse()进行语法分析，我们额外引入了一个全局变量iftrue用来判断是否在分析过程中出现错误，在main中初始化为1，在词法和语法部分如果发现错误则改为1，在完成了两步分析后，若iftrue仍为1，那么我们通过调用dfs(firstNode,0)来进行输出。

```
iftrue=1;
    yyrestart(f);
    yyparse();
    if(iftrue)
    {
        if(firstNode!=NULL)
            dfs(firstNode,0);
    }
```

5. 编译方式

编译的三条指令如下

```
bison -d syntax.y
flex lexical.l
gcc syntax.tab.c main.c tree.c -lfl -ly -o parser
```