Lab1 词法分析器

(2021 学年春季学期)

课程名称：编译原理 任课教师：沈成飞

年级+班级	19级计科（超算）	专业（方向）	计算机科学与技术（超级计算方向）
学号	19335091	姓名	康文生
Email	kangwsh@mail2.sysu.edu.cn	完成日期	2022年3月20日

目录

I 实验要求

> 1. 实验报告
> 2. C语言的LEX源程序：clang.lex
> 3. C语言词法分析程序C源程序：lex.yy.c
> 4. C语言词法分析程序的可执行文件：clang.out/clang.exe
> 5. C语言源程序文件：demo.c(实验输入)
> 6. 词法分析及结果文件：tokens.txt(实验输出)

II 实验过程

- token构造

  词法分析过程中，需要从左向右逐行扫描源程序的字符，识别出各个单词，确定单词的类型。

  将识别出的单词转换成统一的机内表示——token形式

  token<种别码，属性值>

	单词类型	种别	种别码
1	关键字	if，else，return，break…	一词一码
2	标识符	变量名，数组名，函数名…	多词一码
3	常量	整型、浮点型、字符型…	一型一码
4	运算符	算数运算符（+，-，*，\）、关系运算符（>,<,==,!=）、逻辑运算符（&&，||，！）	一型一码
5	界限符	；() = {} ,…	一词一码

其中对于多词一码和一型一码我们需要标注其属性值

- 根据以上token构造原则定义以下语法符号的类别码和部分正则表达式：

  其中正则表达式满足flex工具所定义语法。

单词符号类型	类别码	正则表达式
int	INT	
float	FLOAT	
char	CHAR	
break	BREAK	
continue	CONTINUE	
if	IF	
else	ELSE	
return	RETURN	
do	DO	
while	WHILE	
for	FOR	
{idn}(标识符)	IDN	[A-Za-z][A-Za-z0-9]*
{int}(int字面常量)	CONST_INT	[0-9]+
{float}(浮点字面常量)	CONST_FLOAT	([0-9]*.[0-9]+)|([0-9]+.)
">"|"<"|">="|"<="|"=="|"!="	REL_OP	">"|"<"|">="|"<="|"=="|"!="
"+"|"-"|"+="|"-="|"++"|"--"|"*"|"/"	AR_OP	"+"|"-"|"+="|"-="|"++"|"--"|"*"|"/"
"&&"|"||"|"!"	LOGIC_OP	"&&"|"||"|"!"
";"	SEMI	
","	COMMA	
"="	ASSIGN	
"("	LP	
")"	RP	
"["	SLP	
"]"	SRP	
"{"	BLP	
"}"	BRP	
换行符		[\n]
空格等隐藏符		[\r\t]

单词符号类型	类别码	正则表达式
单行注释		\/\/[^\n]*
多行注释		\/*(\s|.)*?*\/

以上便是所有支持的单词，以及其正规式

- lex源码：

  首先lex源程序主要由三部分组成，相邻两部分用`%%`号隔开

  - 第一部分

    首先在第一部分我们可以开启一些编译lex源程序时的可选项，如下：

    ```
    /*开启行号*/
    %option yylineno
    /*禁用默认主函数*/
    %option noyywrap
    /*不要添加默认规则*/
    %option nodefault
    ```

    1. 第一个规则会使在后续程序中，由变量yylineno记录输入的行号
    2. 第二个规则会禁用默认的主函数，我们可以使用自己定义的main函数
    3. 第三个会禁止添加第二部分默认的正则表达式

    当然第一部分还允许我们添加一些额外的定义，这些定义会被原封不动地添加到生成的源程序的开头部分，如下所示：

    ```
    #include <stdio.h>
    #include <stdlib.h>
    #include <string.h>
    #include "stdarg.h"
    int yycolumn = 1;
    enum yytokentype{
    INT = 258, FLOAT , CHAR, BREAK, CONTINUE,
    IF , ELSE , RETURN, DO , WHILE , FOR,

    CONST_INT,CONST_FLOAT,CONST_CHAR,

    IDN,

    LP , RP , SLP , SRP , BLP , BRP , SEMI , COMMA , ASSIGN,

    REL_OP , AR_OP , LOGIC_OP
    };

    typedef union{
    int type_int;
    float type_float;
    char type_char;
    char type_id[32];
    ```

```
}YYLVAL;
#define ECHO fwrite(yytext,yyleng,1,yyout)
YYLVAL yylval;
```

- 其中我们引用了一些必要的头文件用于字符及输入输出处理，另外定义了种别码，并让其值从258开始取值（如参考书籍所说，是为了避免一些混淆）。
- 最后我们定义了多词一码或者一型一码时要使用的属性值yylval

- 第二部分

这部分应该是最关键的一部分，主要是定义了各个单词的正则表达以及遇到这些单词的行为。

一般而言，行为都是返回其类别码，但是有时候我们需要先保存其属性值，再返回类别码，如下：

```
{idn} {strcpy(yylval.type_id,yytext); return IDN;}
{int} {yylval.type_int=atoi(yytext);return CONST_INT;}
{float} {yylval.type_float=atof(yytext); return CONST_FLOAT;}
```

另外我们在遇到换行时，需要将列号重新更新为1，遇到注释和隐藏符时，不用做任何动作。

最后我们添加我们自己的默认规则，即报错信息：

```
.   {printf("Error type A: Mysterious character\"%s\" at line
%d,column %d\n",yytext,yylineno,yycolumn);}
```

其中yytext保存了当前输入单词的值

另外在定义正则表达式时，遇到二义性时，处理原则如下：

- 词法分析器匹配输入时匹配尽可能多的字符串
- 如果两个模式都可以匹配的话，匹配在程序中更早出现的模式

- 第三部分

第三部分主要是我们写的main函数(展示部分)

```
int  main(int argc,char *argv[]){
    if(argc > 1){
        if(!(yyin = fopen(argv[1],"r"))){
            perror(argv[1]);
            return (1);
        }
    }
    char out_file[100] = "tokens.txt";
    FILE *f_o;
    if((f_o = fopen(out_file,"a+")) == NULL){
        fprintf(stderr,"Can't open %s\n",out_file);
        return (1);
    }

    int tok;
    while(tok = yylex()){
```

```
        switch(tok){
            case INT:{
                fprintf(f_o,"< INT%d, - >\n",tok);
                break;
            }
    }
```

- 我们将输入文件句柄赋值给yyin
- 然后我们每次调用yylex（）就能返回新的匹配到的单词的类别码，属性值存储在yylval中
- 我们根据类别码tok，通过switch case语句将不同的token输出到输出文件

III 操作方法及结果

已经编写了makefile文件，使用方法如下：

```
make clean
make all
make run
```

结果见tokens.txt文件

github链接如下：

sysu-courses/compilers/lab1 at main · snailkk/sysu-courses (github.com)