

编译原理词法分析器实验报告

学院：计算机学院 班级：2017211314 学号：2017213508 学生：蒋雪枫

1.实验内容：

设计并实现 C 语言词法分析器程序

2.实验要求：

- 1) 识别出 C 语言编写的源程序的单词符号，输出记号流
- 2) 可以识别并且跳过代码中的注释
- 3) 可以统计源程序中的语句行数、各类单词个数、以及字符总数，并且最后输出统计结果
- 4) 检查源程序中的语法错误并且报告错误的位置
- 5) 对源程序中出现的错误进行适当恢复，使得词法分析可以继续进行，一次扫描便可以报告源程序中出现的错误

3.实现方式：

C/C++语言

4. 问题分析与设计思路浅析：

- 1) 本次实验是针对 C 语言，所以我们至少应该存储 C 语言保留的关键字，我们用 C++ 语言的 string 数组来存储，便于我们后期编写代码。
- 2) 识别注释并且跳过它。对于这个任务，首先它的入口就是 '/' 这个字符，分为 "//" 情况和 "/*" 两种情况分别处理，如果是 "//" 情况，我们直接往后读，读到换行或者 EOF (虽然基本不用考虑 EOF)，并且把这行信息记录为一个 annotation，即注释。如果是 "/*" 情况，我们就继续往后面读，直到读到了 "*" 这个特征标记或者是 EOF 就代表这个可跨行的注释检测完毕，记作一次 annotation。
- 3) 统计语句行数可以用一个全局变量 line 来记录，单词个数同理，不过在这里是没有算上字符串 "****" 这样的语句内的和注释等，统计结果在程序最后 return 0 前 print 出来就好。
- 4) 要报告错误的位置，我们就需要一个“指针”xy 来表示第几行第几列，当然他们也是我们本身设计词法分析器中必须用到的工具，全局变量是必须的。由于 C 语言可能出现的错误实在过多，学生在这里只识别了课堂上讲过的一些比较经典的由于词法问题产生的错误，比如标识符命名错误。
- 5) 对于错误的适当恢复，解决方案就是跳过会引起错误的字符，往后读取。其余的错误适当恢复学生实在想不出来了...
- 6) 总之，本次程序是 One-Pass，即扫描一遍字符流，并且利用 C++ 的文件输入输出流进行 I/O 处理。而主体实现的思路当然是自动机理论与状态转移图，以及 C 语言标准，性能有限，并且很难让词法分析器性能真的足够强大，达到 MinGW 编译器的标准，比如没有几位同学可以想到去解析 i+++++1，当然 C 语言也有 <flex> 库文件来专门处理词法分析。本次尽力完成本次作品，只能尽可能让这个词法分析器实现作业的基本要求，但要实现一个可以真正拿出来用的还需要不断调优。

5. 重要函数与变量说明:

1. 全局变量

```
//C语言保留字
string keyword[32]={ "auto", "break", "case", "char", "const",
                    "continue", "default", "do", "double", "else", "extern",
                    "enum", "float", "for", "goto", "if", "int", "long", "return",
                    "register", (const char [7]) "switch", "unsigned",
                    "struct", "switch", "sizeof", "typedef", "union",
                    "volatile", "void", "while" };

//行列指针
int column=0,row=1;
//character字符个数, word单词个数
int character=0,word=0;
//从example.txt中读入测试程序
ifstream infile("example.txt",ios::in);
//每一次处理的字符
char c;
```

2. 读取输入文件字符流的下一个字符

```
void get() //全局方法, 直接读取下一个字符
{
    character++;
    c=infile.get();
    column++;
    return;
}
```

3. 遇到空格、tab、换行的处理

```
void choice()
{
    while(c=='\n' || c==' ' || c=='\t') //不计空白符
    {
        //遇到回车, 则跳到下一行, 并且get到下一个字符, col++
        if (c=='\n')
        {
            row++;
            column=0;
        }
        c=infile.get();
        column++;
    }
    return;
}
```

4. 识别是不是 C 语言保留字

```
//检查是否是C语言关键字
int find_keyword(string word)
{
    for(int i=0;i<32;i++)
        if(keyword[i].compare(word)==0)//string类型的compare方法, 如果相等该
            return 1;

    return 0;
}
```

5. 主体架构: (switch-case 语句和 while 循环构成的自动机)

```
int parser()
{
    string once=""; //单次记号流

    if(infile.fail())
        cout<<"请先创建example.txt并向其中输入示例程序"<<endl;
    else
    {
        ofstream outfile("out.txt");
        outfile<<setw(10)<<"Expression"<<setw(30)<<"Mark"<<setw(30)<<"Type";

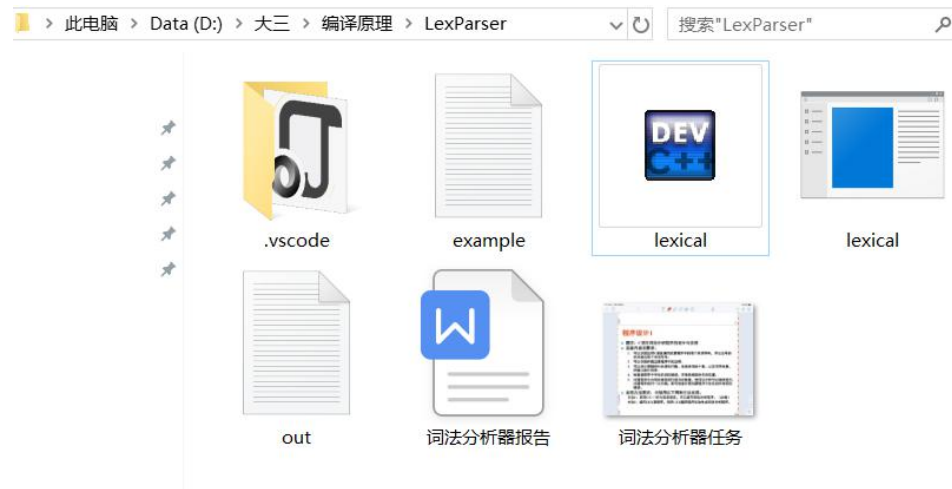
        c=infile.get();
        column++;
        // Automa
        while(c!=EOF)
        {
            switch(c){
                //identifier
                case 'A'...'Z':
                case 'a'...'z':
                case '_':
                    word++;
                    while(isalpha(c)||isdigit(c)||c=='_')
                    {
                        once+=c; //继续往后识别, once连缀上c, 直到没有遇见字母、数字
                        get(); //往后读取一个字符
                    }
                    //遇到构成标识符的不合法字符, 提示错误
                    ....

                    outfile<<"语句行数: "<<row<<endl;
                    outfile<<"单词个数: "<<word<<endl;
                    outfile<<"字符个数: "<<character<<endl;
                    infile.close();
                    outfile.close();
                    return 1;
                }

            infile.close();
            return 0;
        }
    }
}
```

6. 项目文件说明与运行效果

1. 文件说明



Example.txt: 输入文件，为一个设置了错误的 C 语言实例

Lexical.c: 词法分析器

Lexical.exe: 可执行文件

out.txt: 输出文件

词法分析器任务.jpg: 作业任务截图

剩余为实验报告与 vscode 配置文件

2. 运行效果

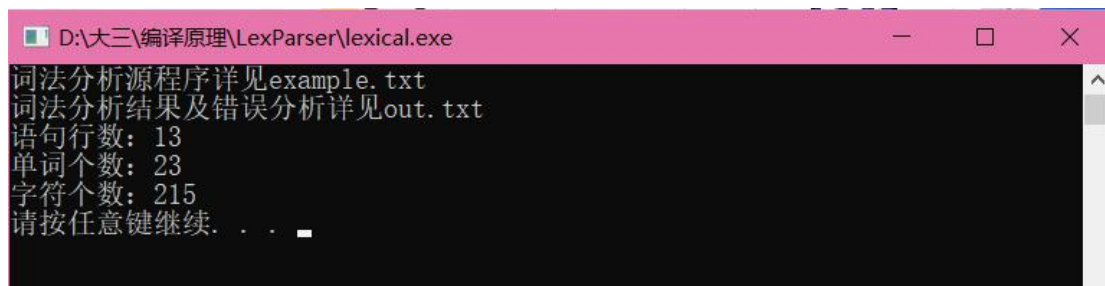
```
example - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
#include<stdio.h>

int main()
{
    int num,summation,8errorName,fla&g;    //发生错误

    printf("Please input a num:"); /*输出*/
    scanf("%d",&num);
    summation=num+9.4EE+10; //常数错误
    printf("The result is %d",summation);

    system("pause");
    return 0;
}
```

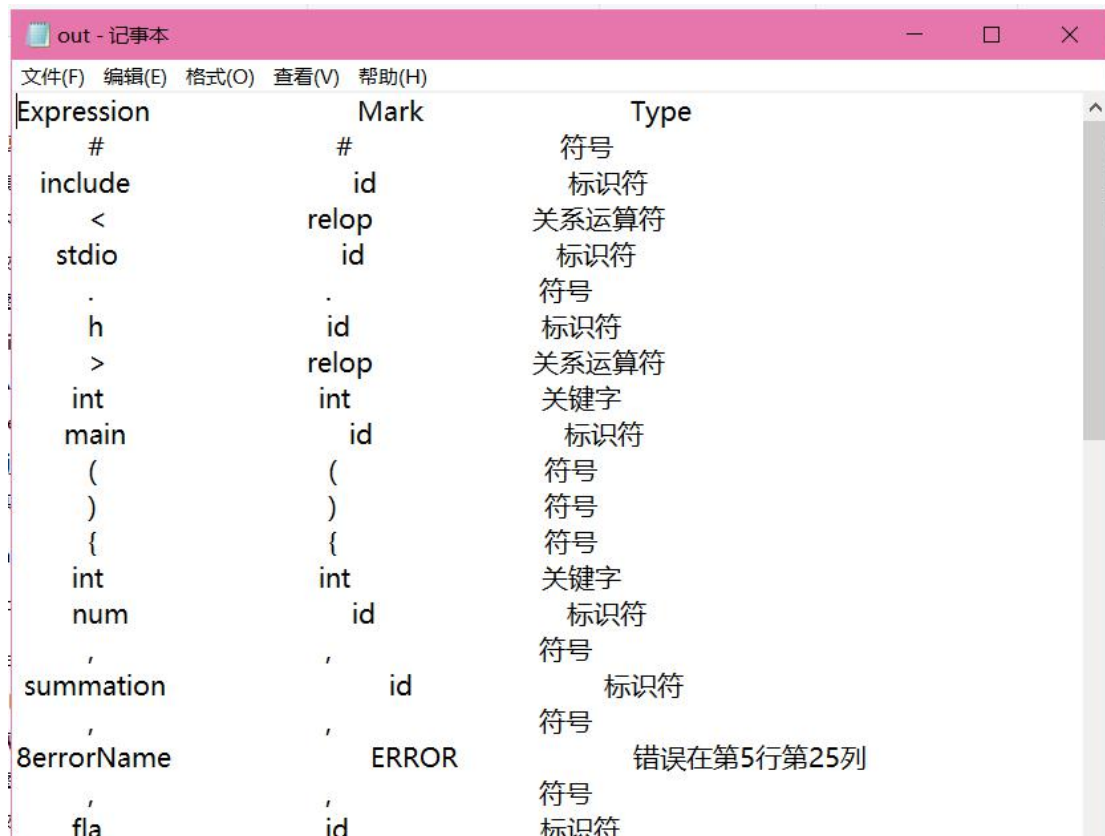
C 语言实例



D:\大三\编译原理\LexParser\lexical.exe

词法分析源程序详见example.txt
词法分析结果及错误分析详见out.txt
语句行数: 13
单词个数: 23
字符个数: 215
请按任意键继续. . .

执行.exe 文件



Expression	Mark	Type
#	#	符号
include	id	标识符
<	relop	关系运算符
stdio	id	标识符
.	.	符号
h	id	标识符
>	relop	关系运算符
int	int	关键字
main	id	标识符
((符号
))	符号
{	{	符号
int	int	关键字
num	id	标识符
,	,	符号
summation	id	标识符
,	,	符号
ErrorName	ERROR	错误在第5行第25列
,	,	符号
fla	id	标识符

输出文件

7. 实验小结:

通过本次实验巩固了对于《编译原理与技术》词法分析本章基本知识, 在实践中加强了自己对 C/C++ 语言的应用, 提高了自己借助外界学习的资料实现自己程序功能的能力。其他的也没啥说的了, 不过 C++ 的文件读写的确用得比较少, 我们这次使用 `infile.get()` 实现逐个读取字符的功能。

8. 源代码附录:

见 lexical.c 文件