实验二 标识符的识别

一、 实验目的

1. 根据给定源语言的构词规则，从任意字符串中识别出所有的合法标识符
2. 输入：字符串
3. 输出：单词符号流

二、 实验过程

1. 考虑到后续符号表的建立以及和语法、语义分析的对接，这里需要对所有的字符进行分类，按照《编译原理》课程中的介绍，所有的字符可以分成五类：
   a. 关键词：if else while main int
   b. 运算符：+ - * / > >= < <= = == !=
   c. 分隔符：{ } ( ) , ;
   d. 标识符：letter(letter | digit | _)*
   e. 常数：(| -)(digit)(digit)* （仅考虑整正负常数）

   所以这里按照分类给出其对应的编号：

表 1 字符分类表

类别	符号
keyword	if else while main int
op	+ - * / < <= > >= = == !=
id	letter(letter | digit | _)*
sep	{ } ( ) , ;
num	(| -)(digit)(digit)*

2. 具体思路如下：

a. 首先跳过所有的空格和换行符

b. 识别开头的字符是否是字母，如果是则只有两种情况：标识符或者关键字，我们先将后续连续的字母、数字和下划线读入组成keyword，再进行判断：如果是关键字则返回关键字，否则返回标识符

c. 识别开头的字符如果是数字，那么一定是常数，识别连续的数字并返回常数

d. 然后剩下负数和一系列运算符，进行逐个判断并返回即可

3. 实现细节注意

a. 需要和实验一中的注释删除函数协作，首先利用注释删除函数得到不含注释的源程序字符串，再调用编写的分词函数进行标识符的识别

b. 由于负数和减法运算符的开头一致，所以需要额外的判断，在流

程示意图中已给出

c. 涉及到两个字符的运算符也需要进行连续判断，在源代码中有所体现

d. 实现时并没有将识别的分词进行保存，后续可以进行改进，建立符号表将识别的串进行保存

e. 利用实验一写好的文件读写模块，从文件中读入源程序进行标识符的识别

三、 实验结果

我们使用文件读写的方式将源程序输入 input.txt，在 result.txt 中得到删除注释的源程序代码，在 cmd 命令行中得到标识符识别的结果。

输入：

图 2 输入源程序 input.tx

输出：

图 3 输出文件 result.txt

标识符识别结果：

图 4 标识符识别结果

四、 实验总结

1. 编译原理实验不是独立分开的实验，每个实验都是紧密联系的，我们应该充分利用前面实验得到的函数、结果，并考虑到后续实验的要求，设计并实现本次实验的内容，为下一次实验打好基础
2. 善于使用流程图来完善程序设计的思路，使得写代码时效率更高
3. 编写程序时需要有项目工程意识，有意识地将各个功能分模块实现

五、 代码展示

```
/*
函数功能：读入除去注释后的源程序进行词法分析
输入：源程序字符串
输出：一行一个二元组
*/
void getToken(char * str)
{
    string keywords;
    for(int i=0;str[i]!='\0';i++)
    {
        if(str[i]!='\n'&&str[i]!=' ')
        {
            //打头的是字母——两种可能：标识符或者关键字
            if(isalpha(str[i]))
            {
                keywords+=str[i++];
                while(isdigit(str[i])||isalpha(str[i])||str[i]=='_')
                {
                    keywords+=str[i++];
                }
                if(keywords=="if")
                {
                    cout << "keyword:if" << endl;
                }
                else if(keywords=="while")
                {
                    cout << "keyword:while" << endl;
                }
                else if(keywords=="else")
                {
                    cout << "keyword:else"  << endl;
                }
```