词法分析实验报告

计科1505　　金波　　15281124

词法分析是编译过程的第一步工作，将字符流转换为单词序列，输出到中间文件中，这个中间文件将会作为语法分析程序的输入，进行下一步工作。

1.程序功能

识别输入文件中的关键字、分隔符、标识符、数字、运算符、注释等。大小写不敏感，字母为a~z, A~Z，数字为0~9。

- **什么是token**

  编译器从左向右扫描源文件，将其字符流分割成一个个的词，这里的词就是‘**token**’，即源文件中不可以再进一步分割的一串字符。

- **token类别**：

Token Key	Token Value
Space	空格
Separator	分隔符(;，{，}，(，))
single-ch operator	单字符运算符(=，+，-，*，/，%，;，|，:，!)
two-ch operator	双字符运算符(:=，!=，==)
num	数字
less equal	<=
NE	<>
less than	<
greater equal	>=
greater than	>
reserved word	保留字
identifier	标识符
string	字符串

comment	注释（单行、多行）
error	一些错误情况，例如数字开头的字符串
other	其他符号（换行、制表等）

- **输入输出**
  - 程序提供两种输入方式：
    1. 控制台输入

       用于测试，输入以'#'结束的字符串。
    2. 文件输入

       读取项目目录下的test.txt文件
  - 运行结果输出也有两种方式
    1. 控制台

       用于测试，输出_tokenArray结构体数组
    2. 文件输出

       输出到result.xml文件中。

2.数据结构

主要的数据结构如下：

name	**feature**
reservedWord()	字符串数组，保存保留字
str()	字符数组，保存从文件或控制台读取的字符
token()	在每一遍扫描中暂存当前token，每一次扫描完成后重置
_token	键值对结构体，保存每个token的类型和内容
_tokenArray	_token结构体的结构体数组
cursorOfStr	记录当前扫描位置
cursorOfStr	记录当前结构体数组存放位置

```
char* reservedWord (9) = {"begin", "end", "if", "then", "else","for", "do", "while", "and or not"};
char str (10240);
char token (128);
int cursorOfStr;
int cursorOfToken;

typedef struct _token {
   char str(128);  //存放token内容
   char key(128);  //存放类型
}_tokenStruct;

struct _token _tokenArray(10240);
```

3.程序结构

- **开发环境**

  系统：MacOs

  IDE：Xcode Version 9.3

  编译：Apple LLVM 9.0

- **实现思路**

  分词方法有直接扫描和正则表达式匹配扫描法，本程序使用直接扫描法实现。

  直接扫描字符进行筛选，通过每轮扫描，判断当前token属于哪种类型，扫描出当前完整的token，将其类型与对应的内容存放到结构体数组中。

  > 例如：
  >
  > 在扫描一个小数时，一直向后扫描直到遇到第一个非数字且不是‘.’的字符，然后将数字类型以及这个数字串存放到结构体数组中。

- **主要函数**

  1. main()函数

     作为入口函数，调用了三个函数，分别是str_config()、getToken()以及out()。

  2. str_config()函数

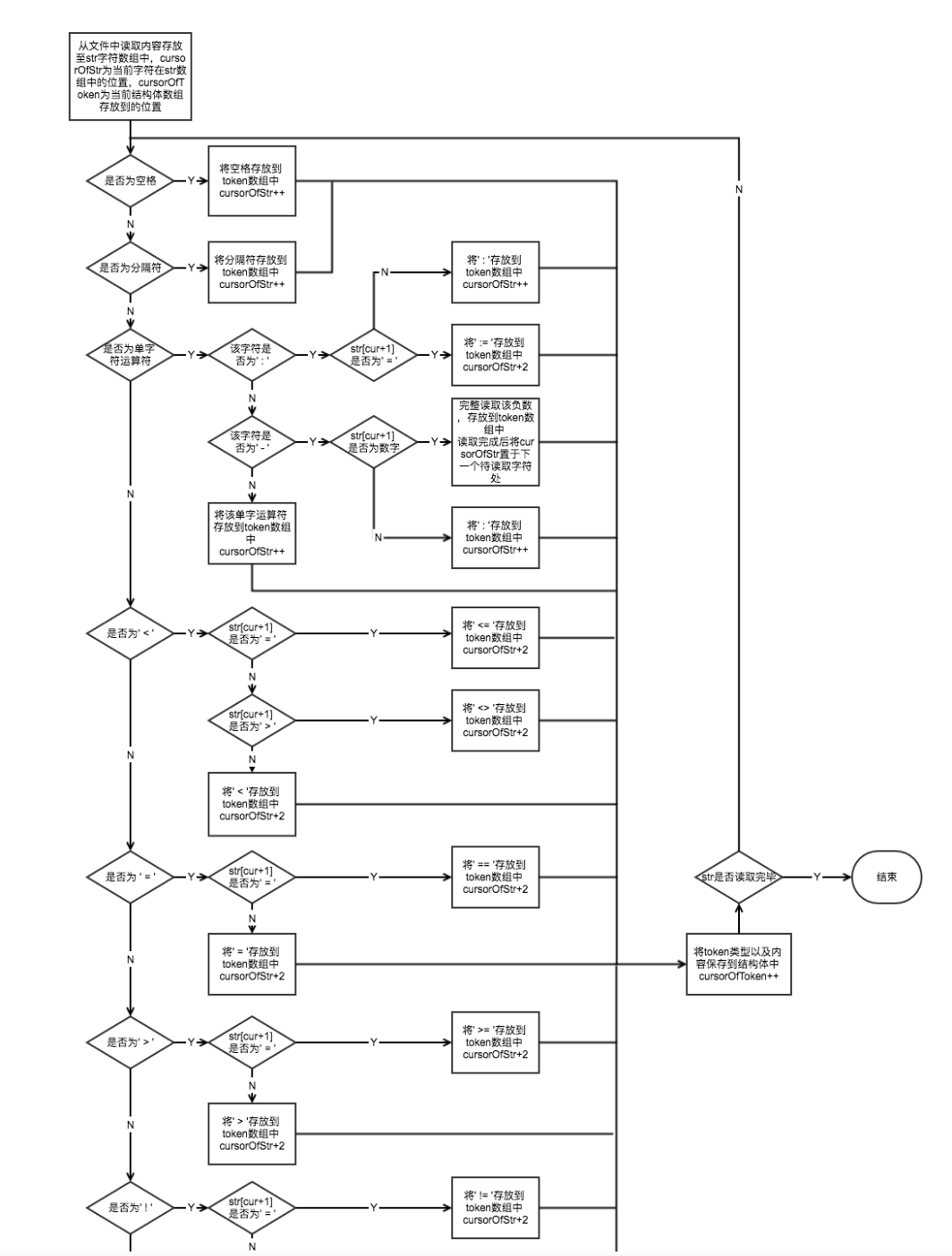
     这个函数的工作是将待分析的内容保存到str数组中，提供了从文件读取或控制台输入两种方式。

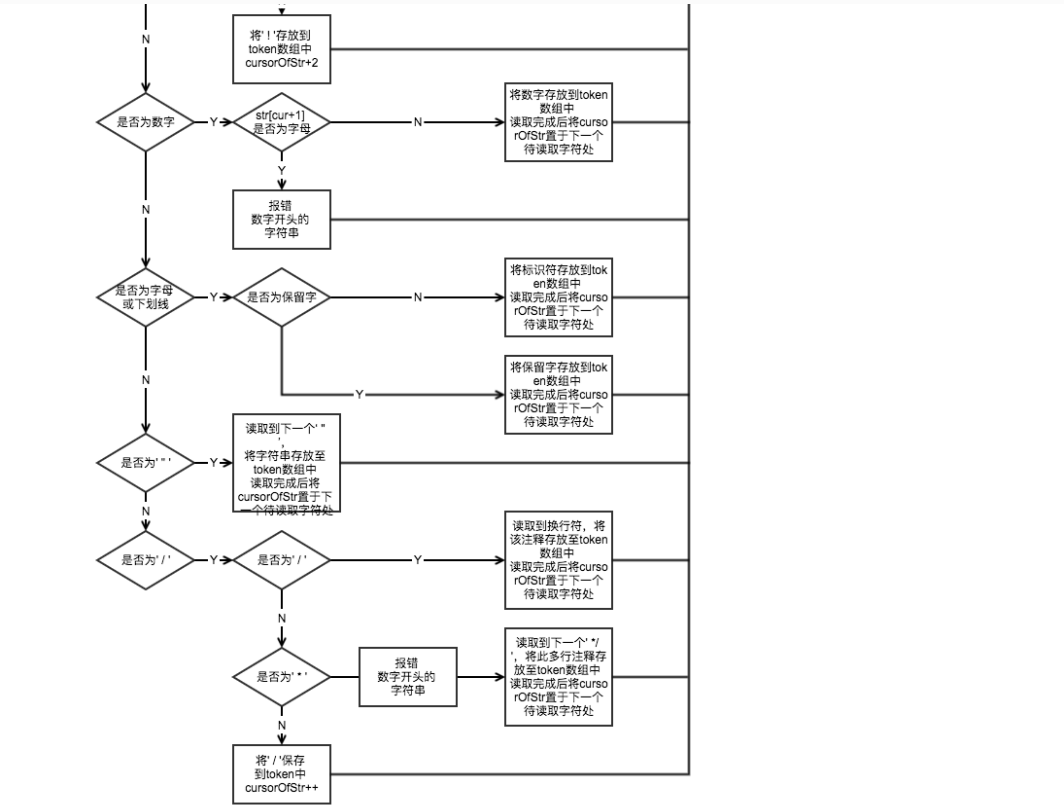
     > c语言读取文件：
     >
     > - 使用fopen()函数创建或打开一个新的文件
     > - 使用fgetc()函数读取字符
     > - 读取完成后，使用fclose()函数关闭文件

```
    FILE *fp;
cursorOfStr = 0;
cursorOfToken = 0;
fp = fopen("/Users/panda_0129/Documents/Learn/编译原
理/lexicalAnalysis/lexicalAnalysis/test1.txt", "r+");
while ((ch=fgetc(fp)) != EOF) {
  str(i) = ch;
  i++;
}
fclose(fp);
```

3. getToken()函数

这是实现扫描、获取token类型的函数，通过if条件判断实现。具体过程如下图所示：





4. out()函数

   顾名思义，out函数将分词完成得到的token结构体数组输出到文件或控制台。

   输出文件为xml格式，具体格式如下：

```
<?xml version="1.0" ?>
<root>
   <Token>
   <
   </Token>
</root>
```

> c语言写入文件：
>
> - 使用fopen()函数创建或打开一个新文件，注意指定打开方式，允许写入
> - 使用fputs()函数写入字符串
> - 写入完成后，使用fclose()函数关闭文件

5. 其他函数

```
bool isASingle(char);      //判断是否为单字符运算符
int isReservedWord(char*); //判断是否为保留字
int isInteger(char);       //判断是否为数字
int isChar(char);          //判断是否为字母（a~z，A~Z）
int isSeparator(char);     //判断是否为分隔符
int isSpace(char);         //判断是否为空格
char* toLower(char*);      //将大写字母转换为小写，实现大小写不敏感
```

4.测试结果

使用test.txt文件测试，测试文件内容和输入文件内容分别如下：

test.txt：

```
//测试用例

 <= >= == != < > +
-3.1415926

BEgin
x:=9;
if x>0
then
  x:=2*x+1/3
end
```

result.xml：

见目录下**result.xml**文件

5.实验总结

实验最主要的部分就是扫描提取出token，在编写getToken()函数时，我遇到了一些bug。通过调试发现主要的问题都出现在了对于str数组下标取值的处理上，这也提醒了我自己在写代码时应该更加思路清晰，尽量减少不必要的错误。

用直接扫描法实现词法分析并不是一件很难的事情，但是使用直接扫描法的弊端在于很难对程序进行拓展，针对复杂一些的语言进行分析时，代码将会变得十分繁琐，不如基于正则匹配的词法分析。

这次实验加深了我对词法分析原理的理解，让我对课上学习到的一些原理有了更深刻的了解。同时也让我回忆了一些C语言的知识。