【SEU&SE】编译原理 - 词法分析器实验报告

原创

Mister_Yu 于 2020-03-02 15:27:27 发布 2094 收藏 13

文章标签： 编译器 c++

本文链接：https://blog.csdn.net/Mister_Yu/article/details/104611570

【SEU&SE】编译原理 - 词法分析器实验报告

README

本篇文章仅供参考，严禁直接抄袭！

一、 实验目的

1. 巩固有限自动机理论与正规文法、正规式三者之间的关系和相关的知识点；
2. 了解词法分析器具体的实现方法及所用算法的原理；
3. 编写相关的词法分析程序实现对C++语言程序文件的部分词法分析；

二、 实验环境

1. 开发环境：

- OS：macOS Sierra 10.12.5
- 语言：C++
- 编译器：Xcode 8.3.3
- 作图：ProccessOn

2. 运行环境

- Windows、macOS或者Linux；

三、 实验内容

1. 主要内容

编写代码实现一个词法分析程序，可以对C++语言中的 .cpp 或者 .h 文件中的部分单词，如关键字、操作运算符、界符、标识符、字符串、整数、浮点数、注释等进行识别并提取出相关的单词，形成token序列，并对一些可能出现的错误进行相应的处理。

2. 主要功能

- **读写文件**：该词法分析器可以读取指定的文件，进行词法分析过程，并将得到的词法分析的token流在控制台中打印出来并写到词法分析程序根目录下的输出文件中；
- **程序预处理**：删除无用的空白字符、回车字符以及其他非实质性字符，删除两种形式的注释；
- **词法分析**：该词法分析器逐个扫描读进的文件字符流，然后进行相应的词法分析。通过扫描，找出相应的单词以及其相应的种别码，形成token序列，为输出到语法分析器中做准备；
- **错误报告**：报告词法检查中遇到的错误以及错误出现的行数，错误包括浮点数小数点错误、数字中混杂字母、字符位数过多、字符丢失单引号、字符串丢失双引号、注释格式不正确、标识符命名错误等等；

3. 种别码

单词符号	种别码
main	1
if	2
else	3
while	4
do	5
for	6
int	7
double	8
float	9
char	10
long	11
short	12

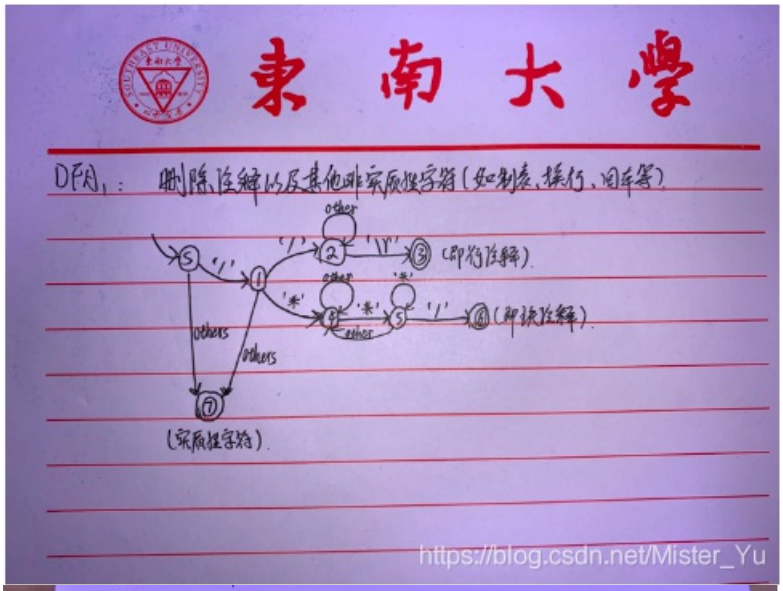
单词符号	种别码
enum	13
static	14
bool	15
void	16
switch	17
case	18
break	19
continue	20
signed	21
unsigned	22
return	23
default	24
const	25
union	26
struct	27
auto	28
include	29
define	30
class	31
virtual	32
friend	33
public	34
private	35
protected	36
this	37
false	38
true	39
try	40
catch	41
throw	42
goto	43
using	44

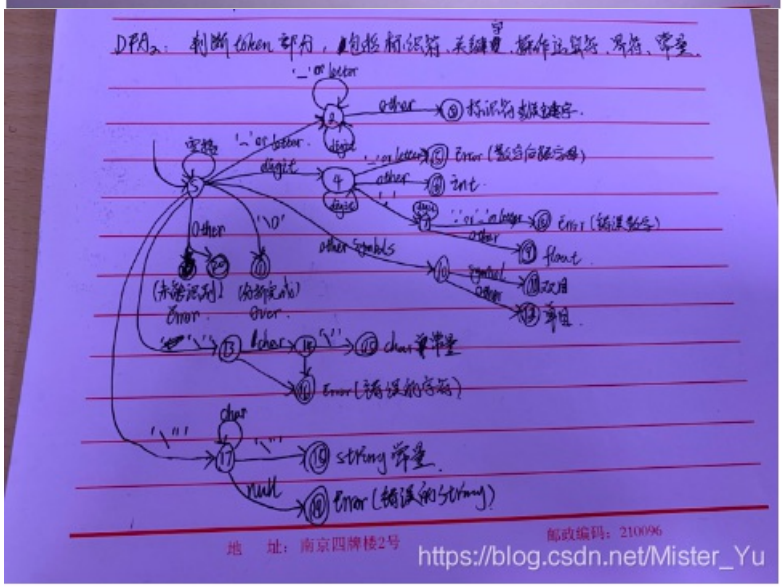
单词符号	种别码
template	45
new	46
namespace	47
operator	48
register	49
inline	50
+	51
-	52
++	53
–	54
*	55
/	56
%	57
<	58
<=	59
>	60
>=	61
=	62
==	63
!=	64
<<	65
>>	66
&&	67
||	68
!	69
^	70
,	71
;	72
.	73
{	74
}	75
[	76

单词符号	种别码
]	77
(	78
)	79
#	80
\	81
?	82
标识符表	83
字符常量表	84
字符串常量表	85
整数常量表	86
浮点数常量表	87

其中，1～50为基本关键字、51～70为操作运算符、71～82为界符。

4. DFA





5. 设计思路

1. 打开源文件，读取文件内容，直至遇上‘$’文件结束符，然后读取结束。
2. 对读取的文件进行预处理，从头到尾进行扫描，去除 // 和 /* */ 的内容，以及一些无用的、影响程序执行的符号如换行符、回车符、制表符等。
3. 接下来对源文件从头到尾进行扫描。从头开始扫描，这个时候分析程序首先去除开头的所有空格，直到扫描出的字符不是空格。然后判断当前字符是不是字母，若是则进行标识符和保留字的识别；若这个字符是数字，则进行数字的判断；若是界符则直接收集；若是操作运算符则进行操作运算符的判断；若是单引号则进行字符判断；若是双引号则进行字符串判断。若是将所有可能都走了一遍还是没有知道它是谁，则认定为错误符号，输出该错误符号，然后结束。每次成功识别了一个单词后，单词都会存在token[ ]中。然后确定这个单词的种别码，最后进行下一个单词的识别。
4. 主控程序主要负责对每次识别的种别码syn进行判断，对于不同的种别做出不同的反应。如对于识别符则将其插入标识符表中；对于保留字则输出该保留字的种别码等等。直至遇到 syn = 0 ，程序结束。

6. 主要数据结构

名称	类型	描述
reserveWords	`const static string*`	存放C++部分关键字
operators	`const static string*`	存放C++部分操作运算符
delimiters	`const static string*`	存放C++部分界符
identitierTable	`vector<string>`	存放标识符
charTable	`vector<string>`	存放字符常量
stringTable	`vector<int>`	存放字符串常量
intTable	`vector<int>`	存放整数常量
floatTable	`vector<float>`	存放浮点数常量

7. 主要算法

RE to NFA to DFA to DFA’

四、 实验结果

程序1:

```
include <iostream>

/*
 让n+1并返回
 */
int inc(int n)
{
    return ++n;
}

int main()
{
    //声明三个变量
    int x = inc(0);
    string y = "test!";
    float z = 6.66;

    return 0;
}
```

输出1:
<include , 29> ----- 关键字
<< , 58> ----- 操作运算符
<iostream , 83> ----- 标识符
<> , 60> ----- 操作运算符
<int , 7> ----- 关键字
<inc , 83> ----- 标识符
<(, 78> ----- 界符
<int , 7> ----- 关键字
<n , 83> ----- 标识符
<) , 79> ----- 界符
<{ , 74> ----- 界符
<return , 23> ----- 关键字
<++ , 53> ----- 操作运算符
<n , 83> ----- 标识符
<; , 72> ----- 界符
<} , 75> ----- 界符
<int , 7> ----- 关键字

五、 遇到的问题及解决方法

问题：程序预处理之后进行词法分析的时候，一直提示“错误的数字”等错误。
解决方法：经过排查之后发现是程序在预处理的时候将所有空格都删除了，导致所有词素都连在了一起，所以一直报错。修改预处理部分使得空格不会被删除之后发现程序即可正常运行了。

六、 实验感想

在本次实验中，我使用C++实现了C++语言的子集的词法分析器。通过本次实验，我对词法分析的流程有了更为清晰的认识，也对课本上所学的知识有了更为深刻的理解，同时也懂得了许多在课堂上没有学到的关于编译的新知识。但总体来看本次完成的词法分析器还是比较低级的，本想可以做到提示错误并作出相应的错误处理，可是写到一半的时候才发现预处理的时候已经吧所有换行符删除了，所以这部分功能无法实现，以后若有机会的话我会将本次实验结果加以改善！