

编译原理实验报告



实验三、词法分析实验报告

姓 名： 刘宇诺

班 级： 计算机 20-1

学 号： 20201210207

任课教师： 韩连金

2022 年 9 月 30 日

目录

一、实验目的	3
二、实验内容	3
三、实验环境	7
四、数据准备	7
五、词法分析器程序设计描述	8
六、词法分析器程序实现展示	9
七、实验结果及分析	11
八、实验心得体会	15

一、实验目的

理解词法分析在整个编译过程中的作用，掌握词法分析的基本原理及运行过程，并实现一个简单的词法分析器

二、实验内容

根据给定的编程语言文法（见下表），编写一个简单的词法分析器，要求该词法分析器读入指定的待分析源代码后，以“(单词/符号编码，单词/符号)”的格式，依次输出该源代码中所有的单词与符号。

语言文法：单词与符号分为五类——保留字、标识符、常数、运算符、界符，分类及编码的具体详情见下列各表。

保留字：

单词/符号	编码	正规式
begin	beginsym	begin
end	endsym	end
input	inputsym	input
output	outputsym	output
if	ifsym	if
then	thensym	then
else	elsesym	else
elif	elifsym	elif

单词/符号	编码	正规式
do	dosym	do
while	whilesym	while
break	breaksym	break
var	varsym	var
and	andsym	and
or	orsym	or
not	notsym	not
true	truesym	true
false	falsesym	false

标识符：

单词/符号	编码	正规式
<标识符>	ident	(字母)(字母 数字)*

常数：

单词/符号	编码	正规式
<常数>	digit	(数字)(数字)*

运算符：

单词/符号	编码	正规式
+	plus	+
-	minus	-
*	times	*
/	divide	/

单词/符号	编码	正规式
%	mod	%
=	becomes	=
==	eql	==
!=	neq	!=
<	lss	<
<=	leq	<=
>	gtr	>
>=	geq	>=

界符：

单词/符号	编码	正规式
(lparen	(
)	rparen)
[lbracket	[
]	rbracket]
{	lcurbkt	{
}	rcurbkt	}
,	comma	,
.	period	.

输入样例：

begin

var a = 6

if $a \% 2 \neq 0$

then $a = a + 1$

else $a = a - 1$

end

输出样例:

(beginsym , begin)

(varsym , var)

(ident , a)

(becomes , =)

(digit , 6)

(ifsym , if)

(ident , a)

(mod , %)

(digit , 2)

(neq , \neq)

(digit , 0)

(thensym , then)

(ident , a)

(becomes , =)

(ident , a)

(plus , +)

(digit , 1)

(elsesym , else)

(ident , a)

(becomes , =)

(ident , a)

(minus , -)

(digit , 1)

(endsym , end)

注：本次实验会进行大量的字符串处理及查表操作，建议各位同学使用 C++、Java 或 python 语言完成作业，不建议使用 C 语言。

三、实验环境

Windows 或 Linux 系统, gcc 7.3.0, g++ 7.3.0, Java JDK 1.8, Python 3.6(以上编译器\解释器均可使用更高版本)。

四、数据准备

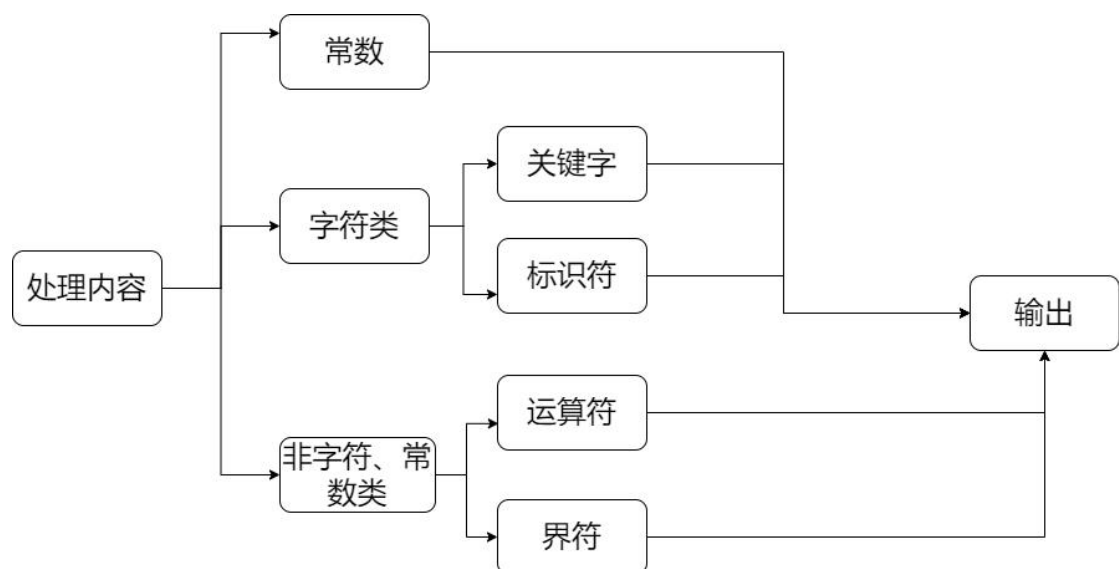
data 文件夹下的三个 txt 文件，为本次实验中需要进行词法分析的源程序代码数据。

五、词法分析器程序设计描述

从文件中读取数据后，逐行进行处理。

可以先分为三大类。常数：相对好处理放到第一处理；字符类：包含关键字和标识符，需要进行细化判断处理，标识符可以用 c++ 的 map 存储，进行比对；算符和界符一起判断。

对于 !=, ==, <=, >= 这样的算符先进行处理，然后一个字符的算符和界符放到一个 map 中，进行匹配输出。



六、词法分析器程序实现展示

对于关键字和只有一个字符的运算符和界符，放到 map 容器里，提取后用 map 的 find 函数进行查找，找到后用 find 函数返回的迭代器输出结果。

```
0 map<string, string> keywords = {"begin", "beginsym"}, {"end", "endsym"}, {"input", "inputsym"}, {"output", "outputsym"},
1 {"if", "ifsym"}, {"then", "thensym"}, {"else", "elsesym"}, {"elif", "elifsym"}, {"do", "dosym"}, {"while", "whilesym"},
2 {"break", "breaksym"}, {"var", "varsym"}, {"and", "andsym"}, {"or", "orsym"}, {"not", "notsym"}, {"true", "truesym"},
3 {"false", "falsesym"};
4
5 map<char, string> other = {'%', "mod"}, {'+', "plus"}, {'-', "minus"}, {'*', "times"}, {'/', "divide"}, {'(', "lparen"},
6 {')', "rparen"}, {'[', "lbracket"}, {'}', "rbracket"}, {'{', "lcurbkt"}, {'}', "rcurbkt"}, {';', "comma"}, {'.', "period"};
7
```

对于数字和字母的判断，c++有 isdigit() 和 isalpha() 这两个函数，可以直接用，不用再编写函数。

常数类型情况少，好判断，故先进行常数的提取输出判断：

```
int idx = 0; // 防止在逐行读取时 while无限循环

cout << "请输入要进行词法分析的文件名称:" << endl;
cin >> filename;
ifstream fin(filename.c_str());
while (getline(fin, strline) && idx < 200)
{
    int n = strline.size();
    for (int i = 0; i < n; i++)
    {
        if (strline[i] == ' ' && i < n) continue; // 如果是空格就跳过
        else if (isdigit(strline[i])) // 如果是常数
        {
            string str;
            while (isdigit(strline[i]))
                str += strline[i++];

            cout << "( digit, " << str << " )" << endl;
            i--;
        }
    }
}
```

进行字符类型的单词提取，第一个一定是字母。

字符型单词提取模块：

```
else if (isalpha(strline[i])) // 如果是关键字 或 标志符
{
    string str;
    str += strline[i ++ ];
    while(isdigit(strline[i]) || isalpha(strline[i]))
        str += strline[i ++];
    i -- ;
    print_letter(str); // 在函数里判断是关键字还是标志服
}
```

字符提取后进行判断是关键字还是标识符。

匹配输出的函数：

```
void print_letter(string str)
{
    map<string, string>::iterator iter, none = keywords.end();
    iter = keywords.find(str);
    if (iter != none)
    {
        cout << "( " << iter->second << ", " << iter->first << " )" << endl;
    }
    else cout << "( ident, " << str << " )" << endl;
}
```

然后是进行运算符和界符模块的判断。先对有两个字符的运算符进行判断。

```
else // 算符 和 界符判断
{
    switch(strline[i])
    {
        case '<':
            i ++ ;
            if (strline[i] == '=')
                cout << "( leq, <= )" << endl;
            else
            {
                i -- ;
                cout << "( lss, < )" << endl;
            }
            break;

        case '=':
            i ++ ;
            if (strline[i] == '=')
                cout << "( eql, == )" << endl;
            else
            {
                i -- ;
                cout << "( becomes, = )" << endl;
            }
            break;
    }
}
```

```
case '>':
    i ++ ;
    if (strline[i] == '=')
        cout << "( geq, >= )" << endl;
    else
    {
        i -- ;
        cout << "( gtr, > )" << endl;
    }
    break;

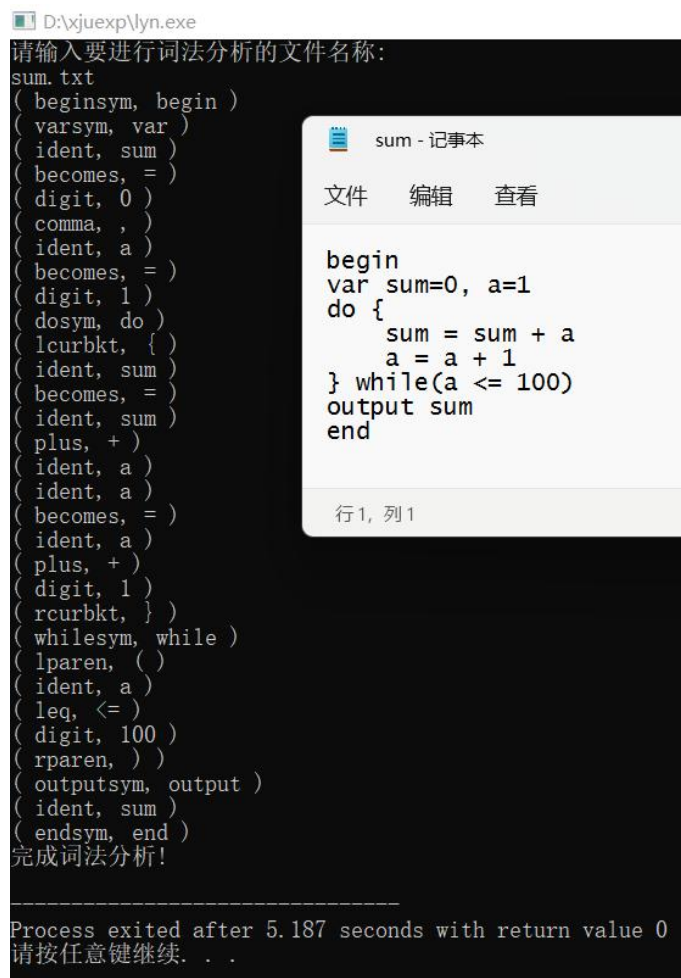
case '!':
    i ++ ;
    if (strline[i] == '=')
        cout << "( neq, != )" << endl;
    else
    {
        i -- ;
        cout << "error" << endl;
    }
    break;
```

最后是对只有一个字符的运算符符合界符进行匹配输出。

```
default:
    char ch = strline[i];
    map<char, string>::iterator iter = other.find(ch);
    cout << "(" << iter->second << ", " << iter->first << ")" << endl;
    break;
```

七、实验结果及分析

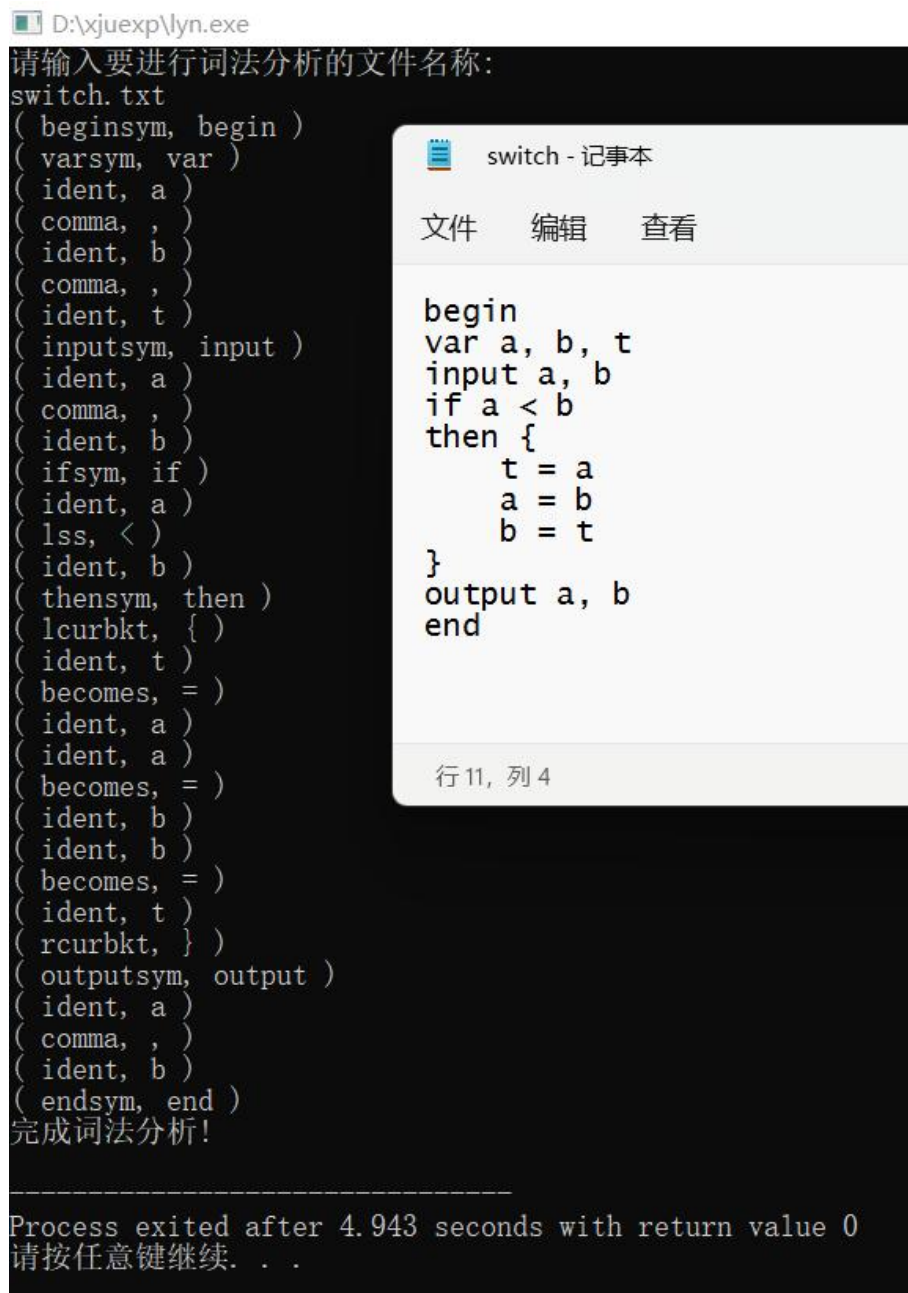
sum.txt 文件进行词法分析的结果。



```
D:\xjuexp\lyn.exe
请输入要进行词法分析的文件名称:
sum.txt
( beginsym, begin )
( varsym, var )
( ident, sum )
( becomes, = )
( digit, 0 )
( comma, , )
( ident, a )
( becomes, = )
( digit, 1 )
( dosym, do )
( lcurbkt, { )
( ident, sum )
( becomes, = )
( ident, sum )
( plus, + )
( ident, a )
( ident, a )
( becomes, = )
( ident, a )
( plus, + )
( digit, 1 )
( rcurbkt, } )
( whilesym, while )
( lparen, ( )
( ident, a )
( leq, <= )
( digit, 100 )
( rparen, ) )
( outputsym, output )
( ident, sum )
( endsym, end )
完成词法分析!

-----
Process exited after 5.187 seconds with return value 0
请按任意键继续. . .
```

switch.txt 文件的词法分析结果



```
D:\xjuexp\lyn.exe
请输入要进行词法分析的文件名称:
switch.txt
( beginsym, begin )
( varsym, var )
( ident, a )
( comma, , )
( ident, b )
( comma, , )
( ident, t )
( inputsym, input )
( ident, a )
( comma, , )
( ident, b )
( ifsym, if )
( ident, a )
( lss, < )
( ident, b )
( thensym, then )
( lcurbkt, { )
( ident, t )
( becomes, = )
( ident, a )
( ident, a )
( becomes, = )
( ident, b )
( ident, b )
( becomes, = )
( ident, t )
( rcurbkt, } )
( outputsym, output )
( ident, a )
( comma, , )
( ident, b )
( endsym, end )
完成词法分析!

-----
Process exited after 4.943 seconds with return value 0
请按任意键继续. . .
```

switch - 记事本

文件 编辑 查看

```
begin
var a, b, t
input a, b
if a < b
then {
    t = a
    a = b
    b = t
}
output a, b
end
```

行 11, 列 4

fill_blank.txt 文件的词法分析结果

D:\xjuexp\lyn.exe

请输入要进行词法分析的文件名称:

fill_blank.txt

```
(beginsym, begin)
(varsym, var)
(ident, array)
(lbracket, [)
(digit, 10)
(rbracket, ])
(comma, ,)
(ident, n)
(becomes, =)
(digit, 1)
(comma, ,)
(ident, i)
(becomes, =)
(digit, 0)
(dosym, do)
(lcurbkt, {)
(ifsym, if)
(ident, n)
(mod, %)
(digit, 2)
(eql, ==)
(digit, 0)
(thensym, then)
(ident, array)
(lbracket, [)
(ident, i)
(rbracket, ])
(becomes, =)
(ident, n)
(elsesym, else)
(ident, array)
(lbracket, [)
(ident, i)
(rbracket, ])
(becomes, =)
(digit, 2)
(times, *)
(ident, n)
(ident, n)
(becomes, =)
(ident, n)
(plus, +)
(digit, 3)
(ident, i)
(becomes, =)
(ident, i)
(plus, +)
```

fill_blank - 记事本

文件 编辑 查看

```
begin
var array[10], n=1, i=0
do {
    if n % 2 == 0
    then array[i] = n
    else array[i] = 2*n
    n = n + 3
    i = i + 1
} while(i < 10)
i = 0
do {
    output array[i]
    i = i + 1
} while(i < 10)
end
```

行1, 列1


```
D:\xjuexp\lyn.exe
( ident, i )
( becomes, = )
( ident, i )
( plus, + )
( digit, 1 )
( rcurbkt, } )
( whilesym, while )
( lparen, ( )
( ident, i )
( lss, < )
( digit, 10 )
( rparen, ) )
( ident, i )
( becomes, = )
( digit, 0 )
( dosym, do )
( lcurbkt, { )
( outputsym, output )
( ident, array )
( lbracket, [ )
( ident, i )
( rbracket, ] )
( ident, i )
( becomes, = )
( ident, i )
( plus, + )
( digit, 1 )
( rcurbkt, } )
( whilesym, while )
( lparen, ( )
( ident, i )
( lss, < )
( digit, 10 )
( rparen, ) )
( endsym, end )
完成词法分析!

-----
Process exited after 5.909 seconds with return value 0
请按任意键继续. . .
```

```
fill_blank - 记事本
文件 编辑 查看

begin
var array[10], n=1, i=0
do {
    if n % 2 == 0
    then array[i] = n
    else array[i] = 2*n
    n = n + 3
    i = i + 1
} while(i < 10)
i = 0
do {
    output array[i]
    i = i + 1
} while(i < 10)
end

行1, 列1
```

由图可以词法分析结果是正确的，三个文件的词法分析都成功的分离出来常数、关键词、标识符、运算符符合界符。也表明编码正确。

八、实验心得体会

通过该次实验进一步深入了解了词法分析，通过实验编码，把课本中的理论用于了实践。运用 c++编程语言编写了一个词法分析器，通过对三个文件的测试和对比，验证了编码是正确的。在这个过程中不仅将自己对词法分析的理解运用到了实际而且锻炼了自己的编程能力，进一步提高了自己的综合能力。在进行单个字符是字母还是数字的判断时，用到了 c++的 `isdigit()` 和 `isalpha()` 两个函数，大大减少了编程代码量。在进行关键字和单个字符的运算符、界符的判断时，用到了 c++的容器 `map`，存储好规则后，通过 `map` 容器中封装好的 `find` 函数判断是哪一个结果，然后再通过返回的迭代器输出结果，容器的运用大大提高编程速度。通过这次实验，对理论的认识和代码实操都有提升。

源代码

```
#include <cstring>
#include <iostream>
#include <algorithm>
#include <string>
#include <map>
#include <fstream>

using namespace std;

map<string, string> keywords = {"begin", "beginsym"}, {"end", "endsym"}, {"input",
"inputsym"}, {"output", "outputsym"},
{"if", "ifsym"}, {"then", "thensym"}, {"else", "elsesym"}, {"elif", "elifsym"}, {"do", "dosym"},
```

```

{"while", "whilesym"},
{"break", "breaksym"}, {"var", "varsym"}, {"and", "andsym"}, {"or", "orsym"}, {"not",
"notsym"}, {"true", "truesym"},
{"false", "falsesym"}}};

```

```

map<char, string> other = {{ '%', "mod"}, { '+', "plus"}, { '-', "minus"}, { '*', "times"}, { '/', "divide"},
{ '(', "lparen"},
{ ')', "rparen"}, { '[', "lbracket"}, { ']', "rbracket"}, { '{', "lcurbkt"}, { '}', "rcurbkt"}, { ',', "comma"},
{ '.', "period"} };

```

```

void print_letter(string str);

```

```

int main()

```

```

{

```

```

    // 文件读取

```

```

    string filename;

```

```

    string strline;

```

```

    int idx = 0; // 防止在逐行读取时 while 无线循环

```

```

    cout << "请输入要进行词法分析的文件名称:" << endl;

```

```

    cin >> filename;

```

```

    ifstream fin(filename.c_str());

```

```

    while (getline(fin, strline) && idx < 200)

```

```

    {

```

```

        int n = strline.size();

```

```

        for (int i = 0; i < n; i++)

```

```

        {

```

```

            if (strline[i] == ' ' && i < n) continue; // 如果是空格就跳过

```

```

            else if (isdigit(strline[i])) // 如果是常数

```

```

            {

```

```

                string str;

```

```

                while (isdigit(strline[i]))

```

```

                    str += strline[i++];

```

```

                cout << "( digit, " << str << " )" << endl;

```

```

                i--;

```

```

            }

```

```

            else if (isalpha(strline[i])) // 如果是关键字 或 标志符

```



```

{
    string str;
    str += strline[i ++ ];
    while(isdigit(strline[i]) || isalpha(strline[i]))
        str += strline[i ++];
    i -- ;

    print_letter(str); // 在函数里判断是关键字还是标志服
}

```

else // 算符 和 界符判断

```

{
    switch(strline[i])
    {
        case '<':
            i ++ ;
            if (strline[i] == '=')
                cout << "( leq, <= )" << endl;
            else
            {
                i -- ;
                cout << "( lss, < )" << endl;
            }
            break;

        case '=':
            i ++ ;
            if (strline[i] == '=')
                cout << "( eql, == )" << endl;
            else
            {
                i --;
                cout << "( becomes, = )" << endl;
            }
            break;

        case '>':
            i ++ ;
            if (strline[i] == '=')
                cout << "( geq, >= )" << endl;
            else
            {
                i -- ;
                cout << "( gtr, > )" << endl;
            }
            break;
    }
}

```

```

        }
        break;

    case '!':
        i ++ ;
        if (strline[i] == '=')
            cout << "( neq, != )" << endl;
        else
        {
            i -- ;
            cout << "error" << endl;
        }
        break;

    default:
        char ch = strline[i];
        map<char, string>::iterator iter = other.find(ch);
        cout << "( " << iter->second << ", " << iter->first << " )" << endl;
        break;
    }
}

}

idx ++ ;
}

// 关闭文件

fin.close();

cout << "完成词法分析!" << endl;

return 0;
}

void print_letter(string str)
{
    map<string, string>::iterator iter, none = keywords.end();
    iter = keywords.find(str);
    if (iter != none)
    {
        cout << "( " << iter->second << ", " << iter->first << " )" << endl;
    }
    else cout << "( ident, " << str << " )" << endl;
}

```