实验2

digit=[0-9]
letter=[A-Za-z]
num100=(\+|-)?digit+
ID101=letter(letter|digit)*
specail200B= \+ | - | * | / | = | < | <= | <<

说明：

（1）通过命名中加了数字编码的，则表示该正则表达式需要生成DFA图。

（2）命名中的名字后的数值为对应单词的编码

（3）命名中的名字中数值后加上B(Begin)表示后面有多个单词，但对应的单词编码从这个数值开始编码。

（4）由于整数中的正号（+）,在系统已经被用作正比闭包运算符号，所以需要通过引入转义符号\来区分。运算符号*也是同样的做法。

实验2

1.输入一行(一个)或多行(多个)正则表达式：

letter=[A-Za-z]

digit=[0-9]

ID101=letter(letter|digit)*

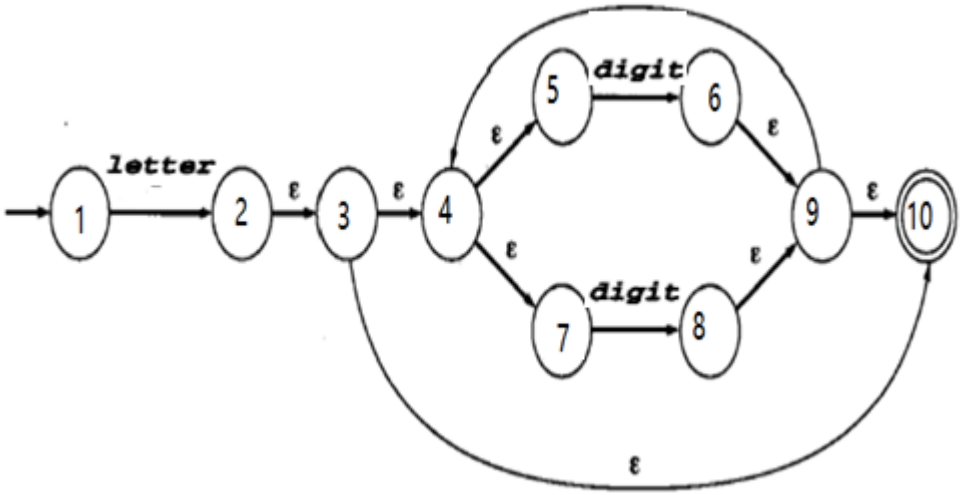
注意：

（1）通过命名中加了数字编码的，则表示该正则表达式需要生成DFA图 。

（2）命名中的名字后的数值为对应单词的编码

2.正则表达式→NFA

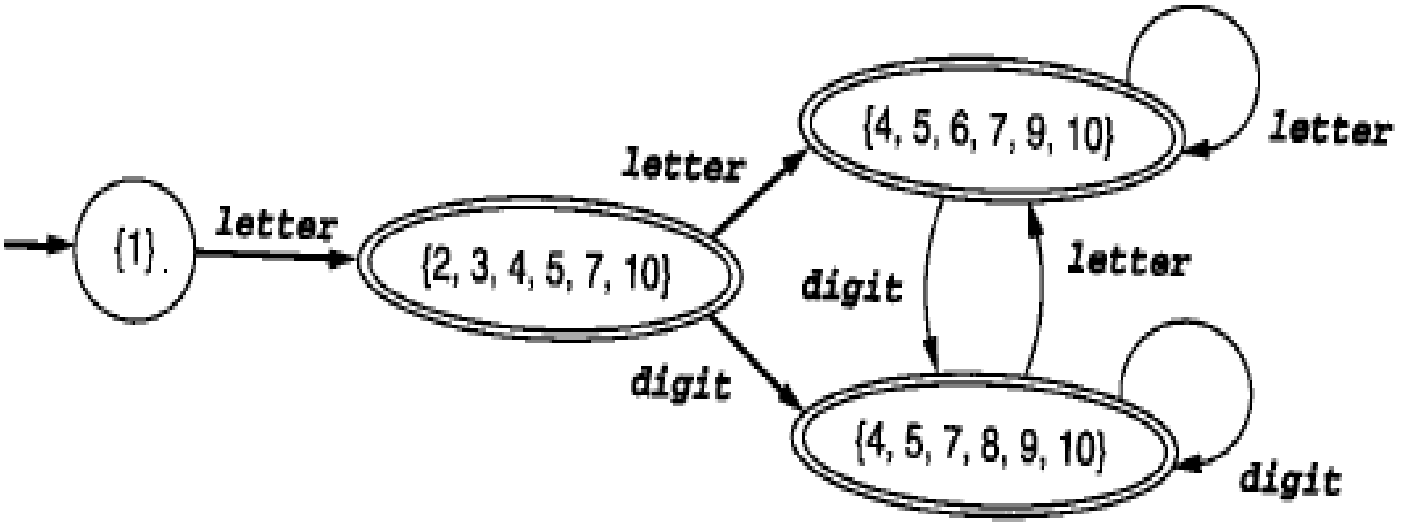
输出：如下图的NFA或该图对应的状态转换表



		letter	digit	#
→	1	2		
	2			3
	3			4, 10
	4			5, 7
	5	6		
	6			9
	7		8	
	8			9
	9			4, 10
←	10			

3.NFA→DFA

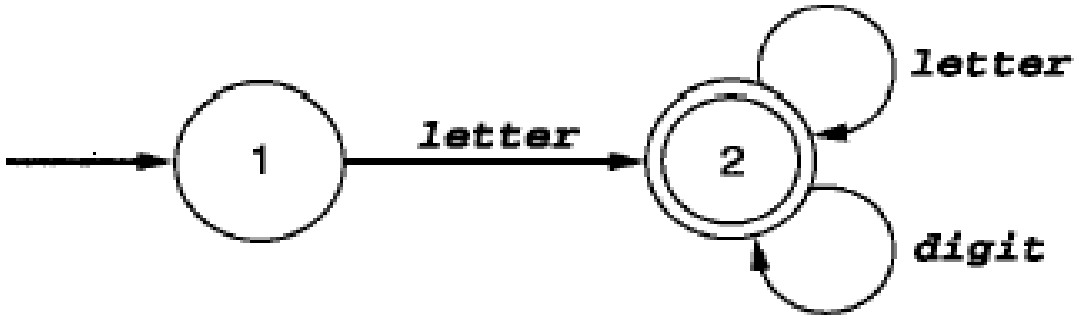
输出：如下图的DFA或对应的状态转换表(如下表)



状态集合 \ 字符	letter	digit
{ 1 }	{ 2,3,4,5,7,10 }	
{ 2,3,4,5,7,10 }	{ 6,9,4,7,10,5 }	{ 8,9,4,7,5,10 }
{ 6,9,4,7,10,5 }	{ 6,9,4,7,10,5 }	{ 8,9,4,7,5,10 }
{ 8,9,4,7,5,10 }	{ 6,9,4,7,10,5 }	{ 8,9,4,7,5,10 }

4.DFA最小化

输出：如下图的最小化DFA或对应的状态转换表



	letter	digit
1	2	
2	2	2

5.DFA→词法分析程序(选做内容)【注意：实验要求生成的分析程序是C/C++语言的表达方法】

输出：生成对应的词法分析程序(方法一)。

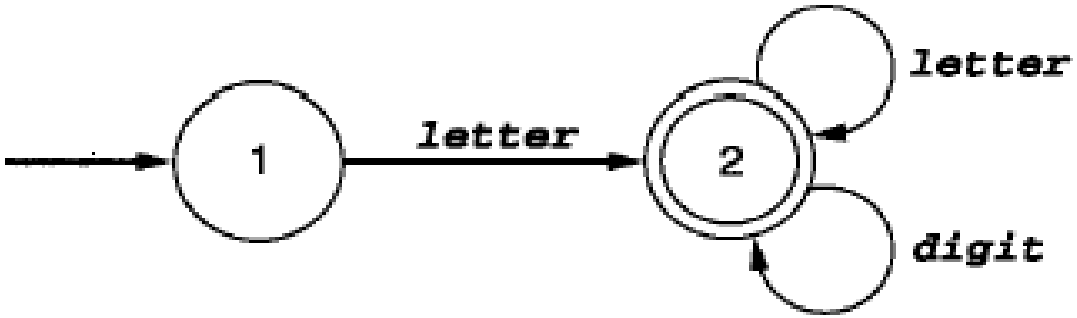
```
#include<iostream>
using namespace std;

int Judgeletter(char ch)
{
  if ( ch>='A' && ch<='Z' || ch>='a' && ch<='z')
     return 1;
  else
     return 0;
}
int Judgedigit(char ch)
{
  if ( ch>='0' && ch<='9' )
     return 1;
  else
     return 0;
}
int pos;
char ch;
char buffer[255];

char GetNext()
{   return buffer[pos++]; }

void GetToken()
{
   if (Judgeletter(ch)==1)
   { ch=GetNext();
     while ((Judgeletter(ch)==1)||(Judgedigit(ch)==1))
        ch=GetNext();
     cout<<"101";
   }
   else
     cout<<"Error";
}


void main()
{
  cin.getline(buffer,255);
  pos=0;
  ch=GetNext();
  GetToken();
}
```



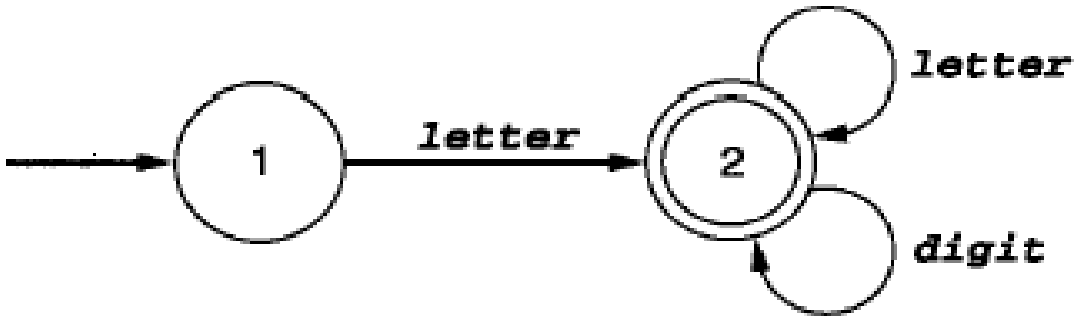
输出：生成对应的词法分析程序(方法二)

```
#include<iostream>
using namespace std;
int Judgeletter(char ch)
{
  if ( ch>='A' && ch<='Z' || ch>='a' && ch<='z')
     return 1;
  else
     return 0;
}
int Judgedigit(char ch)
{
  if ( ch>='0' && ch<='9' )
     return 1;
  else
     return 0;
}
int pos;
char ch;
char buffer[255];

char GetNext()
{   return buffer[pos++];  }

void GetToken()
{
   int state=1;
   bool done=false;
   while (!done)
   {
     switch(state)
     {
      case 1:
               if (Judgeletter(ch)==1)
               { state=2; ch=GetNext(); }
               else
               { cout<<"Error!";return; }
      case 2: if ((Judgeletter(ch)==1)||(Judgedigit(ch)==1))
                   ch=GetNext();
               else done=true;
     }
   }
   cout<<"101";
}
void main()
{
  cin.getline(buffer,255);
  pos=0;
  ch=GetNext();
  GetToken();

}
```



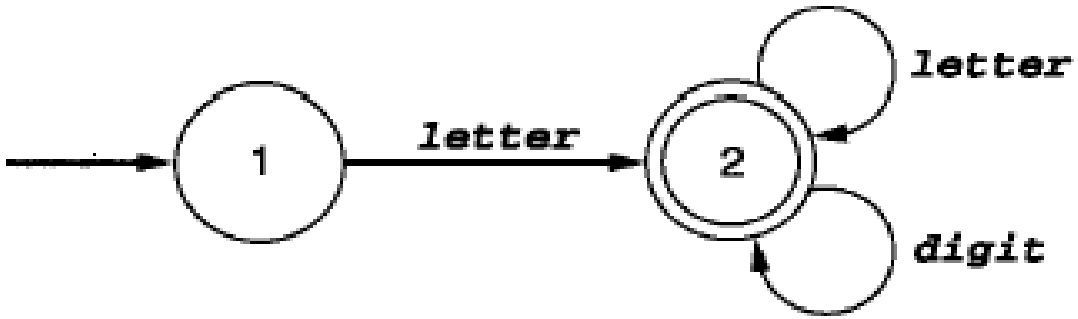
输出：生成对应的词法分析程序(方法二)

```
#include<iostream>
using namespace std;
int Judgeletter(char ch)
{
  if ( ch>='A' && ch<='Z' || ch>='a' && ch<='z')
     return 1;
  else
     return 0;
}
int Judgedigit(char ch)
{
  if ( ch>='0' && ch<='9' )
     return 1;
  else
     return 0;
}
int pos;
char ch;
char buffer[255];

char GetNext()
{   return buffer[pos++]; }

void GetToken()
{
   int state=1;
   while (1)
   {
     switch(state)
     {
       case 1:
       if (Judgeletter(ch)==1)
       { state=2; ch=GetNext(); }
      else
       { cout<<"Error!";return; }
      case 2:
           if ((Judgeletter(ch)==1)||(Judgedigit(ch)==1))
               ch=GetNext();
           else
           {  cout<<"101"; return; }
     }
   }
}

void main()
{
  cin.getline(buffer,255);
  pos=0;
  ch=GetNext();
  GetToken();
}
```



实验需要提交的内容及注意事项

- 第2次实验作业的提交，只能使用RAR文件或ZIP压缩文件。
- 压缩文件内含文件夹及文件如下：
- （1）源程序文件夹：内含整个实验2的所有源程序文件和编译方法的说明介绍文件
- （2）测试数据文件夹：内含所有的测试数据文件(应该能测试所支持的每个运算符)和测试结果的汇报文件
- （3）可执行程序文件夹：内含实验2的可执行程序以及使用说明书。