2012 年编译原理复习提纲

考试题型：（1 班的题型和要求，谨供参考)

一、词法分析算法题（1 题，20 分）

（题目会给出一个单词）

二、正则表达式分析算法题（1 题，20 分）

（要求给出一个正则表达式，会转换成 NFA，子集构造 DFA，DFA 最小化）

三、语法分析算法题（1 题，20 分）

（递归下降法）

四、SLR 分析算法题（1 题，20 分）

LR（0） DFA 图，会解决冲突，会利用 Follow、First 判断是不是 LR（0）

五、语义分析题（1 题，10 分）

（给出 c 语言，会将其转换成四元组、后缀表达、三元组）

六、课程总结题（1 题，10 分）

（写感受、体会）

编译原理复习

实验一 （词法分析）

1、基本正则表达式

L(**a**)={*a*}：基本表达

L(ε)={ ε }：空串

L(Φ)={}：空集

L(**r**|**s**)=L{(**r**)∪L(**s**)：从各选择对象中选择

L(**rs**)=L(**r**)L(**s**)：连接

L(**r***)=L(**r**)*：重复，r 的任意有穷连接，可以是空串

优先级：重复 > 连接 > 选择

2、扩展正则表达式

L(**r+**)=L(**r**)+：重复，r 的任意有穷连接，但不可以是空串

L(**.**)={一个任意字符}

[ ]：表示字符范围，里面连续写出的字符是选择的意思，当中可以用 ‘ – ’ 表示字母表中连续的字符

‘~’：前置符号，非运算，不再给定集合中的任意字符

‘ ？’：后置符号，可选子表达式

3、程序设计语言记号的正则表达式

（1）数字：

nat = [0-9]+

signedNat = (+|-) ? *nat*

number = *signedNat* (“ . ” *nat*) ? (E *signedNat*)

（2）保留字：

reserved = if | while | do | ……

（3）标识符：

letter = [a – z A – Z]

digit = [0 – 9]

identifier = *letter* (*letter* | *digit*) *

（4）分隔符/运算符：

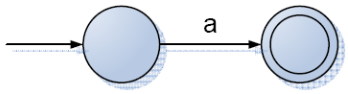
sign = + | - | * | / | ; | ’ | “ | ……

（5）注释：

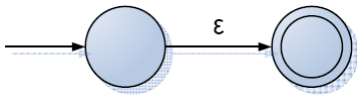
实验二（正则表达式 → NFA → DFA → 最小化）

1、正则表达式 → NFA （P47 例 2.13）

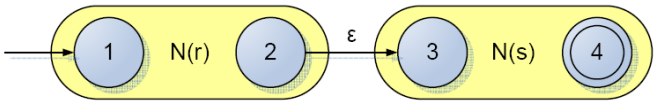
L(a)={a}



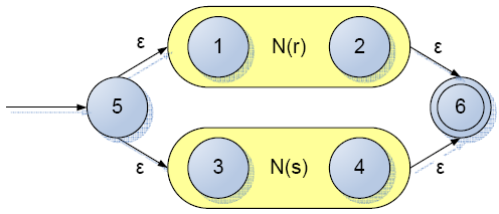
L(ε)={ ε }



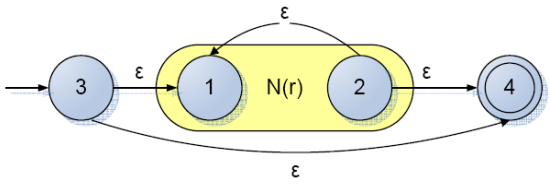
连接 r s：



选择 r | s：



重复 r*



2、NFA → DFA （P50 例 2.17）

首先写出所有的 ε -闭包 _s，再删去给别的包含的 ε -闭包。
然后，子集构造法：从 NFA 的 M 构造出与其等价的 DFA 的 _M
（1）_M 的初始状态为 M 的初始状态 s 的 ε -闭包 _s 。
（2）假设 S 为 _M 上的一个状态，a 为字母表中的一个字符，记 Sa’为所有 S 中的状态通过 a 可以转换到的状态集合：
Sa’ = { t | 存在某个 s∈S，在 a 上有从 s 到 t 的转换 }
（3）在 _M 中增加一个新的状态 _Sa’和一个新的转换 _S → _Sa’ 。
（4）重复（2）和（3）直到不再产生新的状态或转换为止。
（5）将 _M 中至少包含了一个 M 的接受状态的状态标志为接受状态。

3、DFA 的最小化 （P52 例 2.18）

（1）删除 DFA 中永远不能到达的状态；
（2）将所有的状态分成为两个小组：一个由所有的非接受状态组成，另一个由所有的接受状态组成；
（3）假如 S 是某一个组中所有状态的集合，a 是一个字母表中的字符，如果 S 中的状态经过 a 被转移到了若干不同的组，则将 S 分为若干个组；（a 为可分成最少组的优先选）
（4）重复进行（3）中的操作，直到不再产生新的组为止；
（5）将原来 DFA 中同一个组中的状态合并，得到新的 DFA 。

4、运算符优先算法（只对应（， ），*， 连接，| ）

5、DFA → 代码的表驱动算法

根据 DFA 的状态转换函数，给出状态转换表，通过一个通用的表驱动算法来实现 DFA。

```
state = 1;      //从状态 1 开始
ch = getNext ( );      //取出输入串的下一个字符
while ( Accept ( state ) = = false && error ( state ) = = false )
{//当状态不是接受状态或者就不是出错状态的话，就继续循环
newState = T ( state, ch );      //T ( ) 方法是得到由 state 通过 ch 到达的状态
if ( Advance ( state, ch ) = = true )      //Advance ( ) 方法是预测下一个状态是否//满足循环条件
      ch = getNext ( );
state = newState;
}
if ( Accept ( state ) = = true ) accept ( );
```

1. 实验三 tiny 语法分析程序
   1. 递归下降方法

      P109

递归下降分析：例子（factor）

$$factor \rightarrow (exp) \mid \textbf{number}$$

```
void factor(){
  switch(token){
    case '(' :
        match('('); exp(); match(')');
        break;
    case number :
        match(number); break;
    default:
        error();
    } // switch
}// factor
```

递归下降分析：例子（match）

```
void match(expectedToken)
{
  if (token == expectedToken) 当前记号匹配
    getToken(); 取下一个记号，存入token
  else
    error();出错
}// match
```

递归下降分析：例子（exp）

$$exp \rightarrow term \{ addop\ term \}$$

```
void exp()
{
  term();
  while ((token == '+')||(token == '-'))
  {
    match (token) ;
    term();
  }//while
}//exp
```

2. 消除左递归

A->Aa|b

非递归:

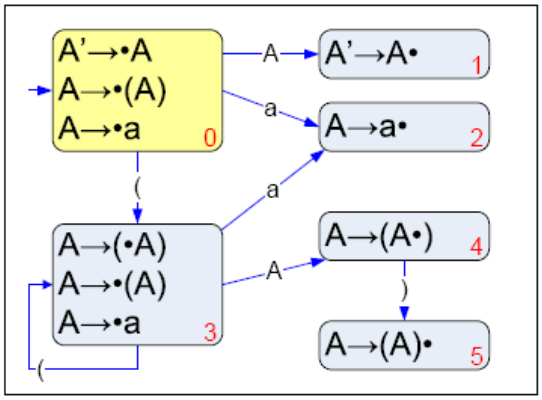
A->bA'

A'->aA'|E

2. 实验四 SLR(1)

用DFA构造LR(0)分析表：例子3

$$(1)A' \rightarrow A \qquad (2)A \rightarrow (A) \qquad (3)A \rightarrow a$$



状态	Action & Goto				
	(	a	)	$	A
0	s3	s2			1
1				acc	
2	r3	r3	r3	r3	
3	s3	s2			4
4				s5	
5	r2	r2	r2	r2	

非$LR(0)$文法的冲突：例子1

如果一个文法不是$LR(0)$文法，则可能存在以下两种冲突：

- 移进-归约冲突：某个项目集中同时存在两个项目：

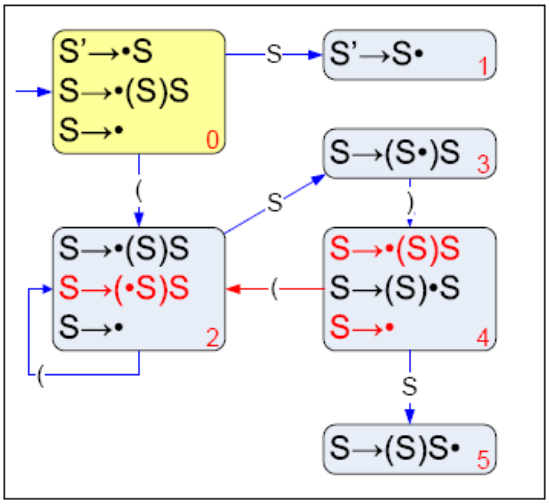
$$\begin{cases} A \rightarrow \alpha \cdot a\beta \text{（移进）} \\ B \rightarrow \gamma \cdot \text{（归约）} \end{cases}$$

- 归约-归约冲突：某个项目集中同时存在两个项目：

$$\begin{cases} A \rightarrow \beta \cdot \\ B \rightarrow \gamma \cdot \end{cases} \text{（两个不同的归约规则）}$$

有冲突的$LR(0)$分析表：移进-归约冲突

$(1)S' \rightarrow S \quad (2)S \rightarrow (S)S \quad (3)S \rightarrow \varepsilon$

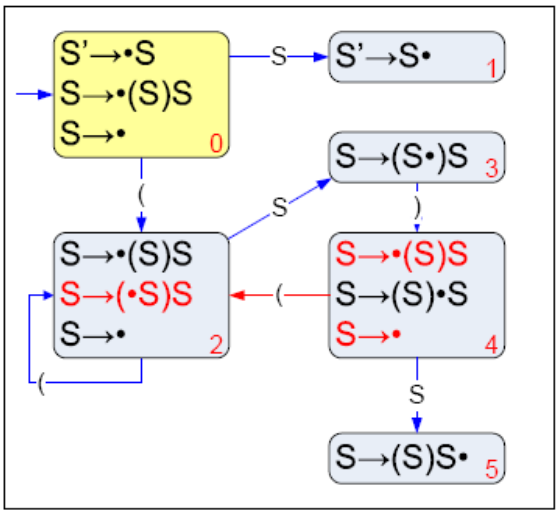


状态	Action & Goto			
	(	)	$	A
0	r3, s2	r3	r3	1
1			acc	
2	r3, s2	r3	r3	3
3		s4		
4	r3, s2	r3	r3	5
5	r2	r2	r2	

注意：$follow(S) = \{\ ),\ \$\ \}$

SLR(1)分析：例子1

$(1)S' \to S \qquad (2)S \to (S)S \qquad (3)S \to \varepsilon$



$follow(S) = \{\$,)\}$

状态	Action & Goto			
	(	)	$	A
0	s2	r3	r3	1
1			acc	
2	s2	r3	r3	3
3		s4		
4	s2	r3	r3	5
5	X	r2	r2	

First 集合，follow 集合计算

first集合的计算：例子1

$$
\begin{array}{lcl}
S \to AB & first(S) = \{a, b, \varepsilon\} & first(AB) = \{a, b, \varepsilon\} \\
S \to bC & first(A) = \{b, \varepsilon\} & first(bC) = \{b\} \\
A \to \varepsilon & first(B) = \{a, \varepsilon\} & first(\varepsilon) = \{\varepsilon\} \\
A \to b & first(C) = \{a, b, c\} & first(b) = \{b\} \\
B \to \varepsilon & first(D) = \{a, c\} & first(aD) = \{a\} \\
B \to aD & & first(AD) = \{a, b, c\} \\
C \to AD & & first(b) = \{b\} \\
C \to b & & first(aS) = \{a\} \\
D \to aS & & first(c) = \{c\} \\
D \to c & &
\end{array}
$$

first 集合的计算：例子2

$$
\begin{array}{lcl}
S \to aH & first(S) = \{a\} & first(aH) = \{a\} \\
H \to aMd & first(H) = \{a, d\} & first(aMd) = \{a\} \\
H \to d & first(M) = \{a, e, \varepsilon\} & first(d) = \{d\} \\
M \to Ab & first(A) = \{a, e\} & first(Ab) = \{a, e\} \\
M \to \varepsilon & & first(\varepsilon) = \{\varepsilon\} \\
A \to aM & & first(aM) = \{a\} \\
A \to e & & first(e) = \{e\}
\end{array}
$$

follow 集合的计算：例子1

$$
\begin{array}{lcl}
S \to AB & follow(S) = \{\$\} \cup follow(D) \\
S \to bC & follow(A) = (first(B) \setminus \{\varepsilon\}) \cup follow(B) \\
A \to \varepsilon & \quad \cup follow(S) \cup first(D) \\
A \to b & follow(B) = follow(S) \\
B \to \varepsilon & follow(C) = follow(S) \\
B \to aD & follow(D) = follow(B) \cup follow(C) \\
C \to AD & \\
C \to b & \\
D \to aS & \\
D \to c &
\end{array}
$$

follow集合的计算：例子1

$$
\begin{array}{lcl}
S & \rightarrow & AB \\
S & \rightarrow & bC \\
A & \rightarrow & \varepsilon \\
A & \rightarrow & b \\
B & \rightarrow & \varepsilon \\
B & \rightarrow & aD \\
C & \rightarrow & AD \\
C & \rightarrow & b \\
D & \rightarrow & aS \\
D & \rightarrow & c
\end{array}
\qquad
\begin{array}{lcl}
follow(S) & = & \{\$\} \\
follow(A) & = & \{\$, a, c\} \\
follow(B) & = & \{\$\} \\
follow(C) & = & \{\$\} \\
follow(D) & = & \{\$\}
\end{array}
$$

follow集合的计算：例子2

$$
\begin{array}{lcl}
S & \rightarrow & aH \\
H & \rightarrow & aMd \\
H & \rightarrow & d \\
M & \rightarrow & Ab \\
M & \rightarrow & \varepsilon \\
A & \rightarrow & aM \\
A & \rightarrow & e
\end{array}
\qquad
\begin{array}{lcl}
follow(S) & = & \{\$\} \\
follow(H) & = & \{\$\} \\
follow(M) & = & \{b, d\} \\
follow(A) & = & \{b\}
\end{array}
$$

3. 四元组

```
while  ( A ∧B∨ C )  do
    begin
        k := k+1;
        if  ( m>n )  then    x := x + y;
        else      x := x-y;
```

```
        y := y*2;
    End
```

翻译成四元组：

```
100    ( jnz , A   ,     , 102 )
101    ( j       ,       ,     , 104 )
102    ( jnz , B   ,     , 106 )
103    ( j       ,       ,     , 104 )
104    ( jnz , C   ,     , 106 )
105    ( j       ,       ,     , 118     )
106    ( +     , k    , 1 ,  T1 )
107    ( :=    , T1,     ,    k    )
108    ( j>    , m , n ,  110 )
109    ( j      ,      ,     ,  113 )
110    ( +     , x   , y ,    T2 )
111    ( :=    , T2,     ,    x    )
112    ( j      ,      ,     ,  115 )
113    ( -      , x   , y ,  T3   )
114    ( :=    , T3,     ,    x    )
115    ( *     ,  y , 2 , T4 )
116    ( := , T4,     ,    y   )
117    ( j    ,       ,  ,  100 )
118
```

编译原理题型及分数比例

一、词法分析题（30 分）

实验一　第二章　重点是完成实验一词法分析所涉及到的方法和过程

二、语法及语义分析（30 分）

实验二及实验三　　第三四章　实验二三所涉及到方法及过程

三、 LR 分析题（15 分）

第五章 以课本例子为主

四、 代码生成题（15 分）

第六、八章 （PPT）能将条件语句、各种循环语句转换为三元组、四元组表示

五、 学习体会题（10 分）

（红色内容为考试重点）

第一章

程序语言的分类？

程序翻译的方式有哪几种？
编译方式和解释方式有何不同？
编译程序包含有多少个阶段，各阶段功能和任务分别是什么？

第二章

词法分析工作特点？
高级程序语言的单词分多少种？
什么是 NFA,什么是 DFA，有什么区别？ r
什么是正则表达式？正则表达式有什么作用？
如何构造正则表达式，看课本例子 2.1-2.5 r
如何将一个正则表达式转换为 NFA，看课本例子 2.12-2.13 r
如何将一个 NFA 转换为 DFA，看课本例子 2.14-2.17 r
如何将一个 DFA 进行最小化，看例子 2.18-2.19 r
将 DFA 转换为相应的分析程序 P41-P45 r
重点：词法分析构造步骤
正则表达式→NFA→DFA→DFA 最小化方法→词法分析程序

第三章

什么是文法？什么是语言？
文法的分类是怎样的？它们之间有何关系？
什么是递归规则，递归文法？
文法的推导和规约指的是什么？
推导方法有多少种？
什么是语法树？如何画语法树（看讲稿 PPT 例子，会画 if、while 语法分析树） r
文法二义性指的是什么？如何判断文法具有二义性？ r
文法二义性的消除方法有多少种？

第四章

什么是自顶向下语法分析方法？
自顶向下分析有多少种具体的分析方法？
如何求 First 集合？如何求 Follow 集合？ r
如何进行左递归的消除？（看课本例子） r
如何进行左分因子的消除？（看课本例子） r
如何利用递归子程序方法进行语法分析？（看课本例子） r
如何利用 LL（1）分析方法进行语法分析？（看课本例子） r
什么是 LL(1)文法？ r
重点：语法及语义分析程序的构造步骤：
——构造文法规则
——递归下降语法分析方法
——左递归下降分析程序基础进行语义分析并生成中间代码
左递归的消除
FIRST 集合和 FOLLOW 集合
LL（1）分析方法。

第五章　（LR 方法课本例题 5.6—5.17）　　r

什么是自底向上语法分析方法？

自底向上语法分析有多少种具体的分析方法？

如何构造 LR(0)DFA 、 LR(1)DFA 、 LALR(1)DFA?　　r

如何构造 LR(0)分析表、LR(1)分析表、SLR（1）分析表、LALR（1）分析表？　　r

什么是 LR(0)文法、LR（1）文法、SLR（1）文法、LALR（1）文法？

如何利用 LR(0)、LR(1)、SLR(1)、LALR(1)进行语言语法分析，看课本 5.6-5.17　　r

第六章

语法制导翻译的方法有多少种？

高级程序设计语言的语句是如何分类的？其各自的翻译任务又是怎样的？

常见的中间代码有哪些，其展示形式是怎样的？

如何将一个算术表达式转换为逆波兰表示，四元组表示，三元组表示？

如何将一段代码翻译为中间代码（四元组，三元组，后缀表示）？（看 PPT）　　r

04 年编译原理

（1）　LL(1)不考

（2）　SLR

（3）　自顶向下：左递归

左公共因子

(4)(Dot)<lex and yacc>(精通正则表达式)

一、词法分析算法题（1 题，20 分）给一个单词

二、正则表达式分析算法题（1 题，20 分）

写正则表达式，画 DFA 转换 NFA，给一个正则表达式，转换成 NFA，（子集构造）DFA

三、语法分析算法题（ 1 题，20 分）　递归下降法

四、SLR 分析算法题（1 题，20 分）　LR（0）DFA 图——>求 Follow 、First

五、语义分析题（1 题　，10 分）　（C 语言）四元组、后缀表示、三元组

六、课程总结题（感受、体会）　（1 题，10 分）

实验二免代码，只画图

实验三主要是数据结构，语法分析树，还有 do while 语句

实验四要写 First Follow 的算法，其他要求手工从头做到尾

第六章给出一段代码写四元组（看课件例子）

1-4 题就是考四个实验 80 分

第 5 题考第六章的内容`10 分

第 6 题写感想 10 分

2007 年编译原理试题：

1、词法分析：对输入的型如：232、232.23、232E-2 的数写一个分析程序；
2、正则表达式到 NFA 然后到 DFA 的实验中，怎么表达正则表达式的运算符的优先级，写出思路跟算法；
3、TINY 扩充语言实验中，求余运算是怎么实现的？写出思路跟算法；
4、给出一个语法，画出其 LR（1）的 DFA；
5、写出 repeat~until 的语法规则和语义动作；
6、学习编译后感想

2008 年编译考试题目：
1、 词法分析：/* */的分析程序或其 DFA 图；
2、 给出一个正则表达式，手工画其 NFA 和 DFA；
3、 写出在实验三，语法分析，所使用的数据结构；
4、 在实验四中，写出实现一个非终结符号的 first 集合求解的算法；
5、 给出一段代码，写出其四元组表达；
6、 学习编译后感想

1、 词法分析：对输入的型如：232、232.23、232E-2 的数写一个分析程序；

正则表达式：

X->AB

A->232

B->.C|E+C|E-C|空

C->2|23

递归分析程序：

```
void X()
{
    if(token=='2')
    {A();}
    else error();
    if(token=='.'|token=='+'|token=='-')
   { B();}
    else error();
}
```

```
void A()
{
  match(2);
  match(3);
  match(2);
}
void B()
{
  if(token=='.')
  {
      C();
   }
  if(token=='E')
  {
     getToken();//取下一字符
     if(token=='+'|token=='-')
     {
        C();
     }
   }
}
void C()
{
      match(2);
      getToken();
      if(token=='3')
```

```
    {
      match(3);
     }
}
void main()
{
   getToken();
    X();
}
```

2、 正则表达式到 NFA 然后到 DFA 的实验中，怎么表达正则表达式的运算符的优先级，写出思路跟算法。有多少种优先级就引入多少个非终结符号

例如+-*/%（）数字 0~9

E->E+T|E-T|T //T 高于+-优先级

T->T*F|T/F|T%F|F //F 的优先级高于*/%

F->(E)|n //n 表示数字 0~9

写成程序：

消除左递归：

方法 1：

E->TE'

E'->+TE'|-TE'|空

T->FT'

T'->*FT'|/FT'|%FT'|空

F->(E)|n

方法 2：

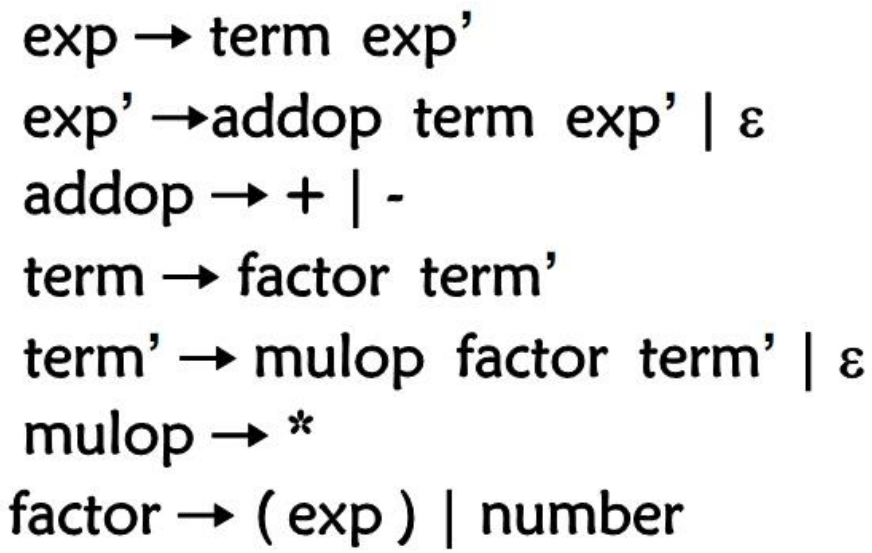
E->T+T|T-T|T

T->F*F|T/F|T%F|F

F->(E)|n

然后每一个非终结符号一个函数，要 5 个函数+main 函数

程序参考黄煜廉老师课件第 4 章



3、TINY 扩充语言实验中，求余运算是怎么实现的？写出思路跟算法

由于求余运算%和*、/的运算优先级相同。只需在对*和/操作时增加对%的操作就可以了。

4、　　给出一个语法，画出其 LR（1）的 DFA；

课本第 5 章有例子：page167，例 5.14，5.16，　5.17

3、请将下面 C 语言代码段翻译成四元组的形式

```
if(A<B&&C>D||E!=F)
    if(X>Y&&S<T) G=0;else G=1;
else H=1;
```

1、4 元组

1(J<,A,B,3)
2(J,,,5)
3(J>,C,D,7)
4(J,,,5)
5(J!=,E,F,7)
6(J,,,15)
7(J>,X,Y,9)
8(J,,,13)

9(J<,S,T,11)
10(J,,,13)
11(=,0,,G)
12(J,,,16)
13(=,1,,G)
14(J,,,16)
15(=,1,,H)
16

1、实验一是对 C/C++语言进行词法分析,请写出 C 语言注解/* */的词法分析代码段.(代码段可以用 C/C++/JAVA 进行描述,也可以只画出/* */的 DFA 图)

词法分析：/* */

```
SWITCH '/':
if(getToken=='*')
{
cout<<"/*";
l:
while(getToken(!='*')
cout<<Token;
cout<<"*";
if(getToken=='/')
{
cout<<"/注释"<<endl;
}
else goto l;
}
else .....
```