# 宏的使用

宏名是合法的ID；

宏接受合法的ID作为参数；

宏的变量同样具有作用域，如果宏变量和一个宏名冲突，优先采用此变量；

如果宏引用了一个同样是宏的ID，则进行递归展开；注意，一个宏只能引用一个已经存在的宏而不能引用一个正在定义或尚未定义的宏；

宏的全局作用域：为宏分配一个id，在一个列表中，在此id之前的宏都是可用的。

宏的局部作用域：宏的参数构成一个局部作用域

引用宏时，先进行局部作用域替换；然后进行全局作用域替换；

总是引用逆向查找的第一个值，如果没有找到就保留原始值。可以使用$开头，表示总是引用原始值而不进行作用域替换，若需要使用$原值，使用\$,若需要使用\原值，使用\\.

注：c语言的宏具有非声明性质，因此不允许递归调用

宏可以使用正则表达式实现。

# 宏的文法

SPACES -> SPACE SPACES|e

S-> # SPACES ID\_DEFINE SPACES ID SPACES (e|HASARG SPACES) BODY (NEWLINE|e)

HASARG -> (SPACES MARGS ) | (SPACES)

MARG ->SPACES ID SPACES

MARGS -> MARG , MARGS | MARG

BODY -> (! NEWLINE)BODY|e (没有新的语法变量)

可以使用LL(1)文法。

## 简化表示

设

T=SPACES t=空格 e=空产生式 z=结束符 f=ID d=ID\_DEFINE n=NEW\_LINE H=HASARG M=MARGS N=MARG B=BODY x=非\n的字符

则语法表达为:

S -> #T d T f T (e | H T) B (e|n)

T -> t T | e

H -> (T M) | ( T )

N -> T f T

M -> N , M | N

B -> x|e

## 消除左递归

经检查，上述文法没有左递归，也没有二义性

## 消除左因子

引入

S\_A -> e | H T

S\_AA -> e | n

H\_A -> M | e

M\_A -> e | , M

则原语法变成

S -> #T d T f T S\_A B S\_AA

T -> t T | e

H -> (T H\_A )

N -> T f T

M -> N M\_A

B -> x|e

S\_A -> e | H T

S\_AA -> e | n

H\_A -> M | e

M\_A -> e | , M

## FIRST集和FOLLOW集

|  |  |  |
| --- | --- | --- |
| 变量 | FIRST集 | FOLLOW集 |
| S | # | z |
| T | t e | d f x ( n z ) t |
| H | ( | t x n z |
| N | t f | ) , |
| M | t f | ) |
| B | x e | n z |
| S\_A | e ( | z x n |
| S\_AA | n e | z |
| H\_A | t e f | ) |
| M\_A | e , | ) |

## 预测分析表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | t | # | d | f | n | ( | ) | , | x | z |
| S |  | S->#.. |  |  |  |  |  |  |  |  |
| T | T-> tT |  | T->e | T->e | T->e | T->e | T->e |  | T->e | T->e |
| H |  |  |  |  |  | H->(TH\_A) |  |  |  |  |
| M | M->NM\_A |  |  | M->NM\_A |  |  |  |  |  |  |
| N | N->TfT |  |  |  |  |  |  |  |  |  |
| B |  |  |  |  | B->e |  |  |  | B->x | B->e |
| S\_A |  |  |  |  | S\_A->e | S\_A->HT |  |  | S\_A->e | S\_A->e |
| S\_AA |  |  |  |  | S\_AA->n |  |  |  |  | S\_AA->e |
| H\_A | H\_A->M |  |  | H\_A->M |  |  | H\_A->e |  |  |  |
| M\_A |  |  |  |  |  |  | M\_A->e | M\_A->,M |  |  |

## 程序声明

**class** LL1Macro{

**public**:

**typedef** LL1Macro This;

**typedef** LexicalParser::WordStream WordStream;

**enum**{

*TYPE\_DEFINE*=LexicalParser::*TYPE\_ID*,

*TYPE\_SPACE*=LexicalParser::*TYPE\_SPACE*,

*TYPE\_ID*=LexicalParser::*TYPE\_ID*,

*TYPE\_NEWLINE*=LexicalParser::*TYPE\_NEWLINE*,

*TYPE\_END*=LexicalParser::*TYPE\_END*,

};

**enum**{

*CHAR\_EOF*=0

};

**public**:

**LL1Macro**()=**default**;

**LL1Macro**(**const** WordStream& s,WordStream::size\_type start);

**~LL1Macro**();

**void** **SPACES**();

**void** **HASARG**();

**void** **HASARG\_A**();

**void** **S**();

**void** **S\_A**();

**void** **S\_AA**();

**void** **MARG**();

**void** **MARGS**();

**void** **MARGS\_A**();

**void** **BODY**();

**void** **matchType**(**int** word);

**void** **matchSingle**(**char** ch);

AS\_MACRO **char** **getCurChar**()**const**;

AS\_MACRO **int** **getCurType**()**const**;

AS\_MACRO **void** **moveNext**();

AS\_MACRO **void** **movePrevious**();

**void** **error**();

**public**:

**const** WordStream &s;

**const** WordStream::size\_type len;

WordStream::size\_type index;

};

# 测试

文本：**# define** what (is, the ,fuck) is ()

其词法流为：

<start>

# SPACE ID SPACE ID SPACE ( ID , SPACE ID SPACE , ID ) SPACE ID SPACE ( ) NEWLINE END

<end>

匹配过程：

matching S

matching single : #

matching SPACES

mathcing type : SPACE

matching SPACES

mathcing type : ID

matching SPACES

mathcing type : SPACE

matching SPACES

mathcing type : ID

matching SPACES

mathcing type : SPACE

matching SPACES

matching S\_A

matching HASARG

matching single : (

matching SPACES

matching HASARG\_A

matching MARGS

matching MARG

matching SPACES

mathcing type : ID

matching SPACES

matching single : ,

matching MARGS

matching MARG

matching SPACES

mathcing type : SPACE

matching SPACES

mathcing type : ID

matching SPACES

mathcing type : SPACE

matching SPACES

matching single : ,

matching MARGS

matching MARG

matching SPACES

mathcing type : ID

matching SPACES

matching single : )

matching SPACES

mathcing type : SPACE

matching SPACES

matching body

mathcing type : NEWLINE

# 附录：LL（1）文法

步骤：

* 构造文法G
* 改造文法G：消除二义性、消除左递归、消除左因子
* 求每个候选式的FIRST集和FOLLOW集
* 构造分析表
* 构造语法图
* 简化语法图
* 为每个语法图设置一个子程序

## 消除左递归

确认没有循环推导（形如 A --+--> A，即A自身推出自身的形式）

消除直接左递归：A->Aa|b 可以化为 A->b A’ A’ -> a A’ | e(空产生式)

消除间接左递归（基本思想：将所有的间接左递归变为直接左递归，然后使用直接消除左递归）

所有的产生式记为A1,..,An

i=循环n次，j=每个比i小的表达式

如果Ai->Ajb , 直接将Aj替换掉

最后，消除所有直接左递归

## 消除左因子

如果存在可以提取的左因子，则提取，直到没有新的可提取因子

## 求FIRST集：

按照确定性递归—回溯

## 求FOLLOW集：

先写开始符号，记为EOF

所有具有FIRST集的邻接点

递归求邻接点为空的且较待求符号改变过的FOLLOW集

## 预测分析表的构成

对任意一个产生式A->a，如果输入符号t属于a的FIRST集，则填表；

如果a存在空产生式，将所有的A的FOLLOW中的输入符号填上A->a

如果a存在空产生式且结束符号在FOLLOW（A）中，则在结束符号上填入A->a