**[c语言](http://lib.csdn.net/base/c" \o "C语言知识库" \t "_blank)**的文法产生式：

program ->external\_declaration | program external\_declaration

程序 -><外部声明> | <程序><外部声明>

external\_declaration->function\_definition | declaration

外部声明-> <函数定义>|<声明>

function\_definition->type\_specifier declarator compound\_statement

函数定义-> <函数类型><说明符><复合语句>

type\_specifier->VOID | CHAR | INT | FLOAT

函数类型-> void|char|int|float

declaratory->pointer direct\_declarator | direct\_declarator

说明符-> <指针 ><直接说明符>|<直接说明符>

Pointer-> '\*' | '\*' pointer

指针-> <\*>|<\*><指针>

direct\_declarator-> IDENTIFIER | direct\_declarator’[‘ ‘]’ |direct\_declarator ’[’ constant\_expression ’]’ | IDENTIFIER '(' parameter\_list ')' | IDENTIFIER '(' ')'  |direct\_declarator  ‘,’  identifier\_list

直接说明符-> <标识符>|<直接说明符[]>|<直接说明符[常量表达式]>|<标识符（参数序列）>|<标识符（）>|<直接说明符，标识符序列>

identifier\_list-> IDENTIFIER | identifier\_list ',' IDENTIFIER

标识符序列-> <标识符>|<标识符序列><，><标识符>

constant\_expression->conditional\_expression

常量表达式-> <条件表达式>

parameter\_list->parameter\_declaration | parameter\_list ',' parameter\_declaration

参数序列-> <参数声明>|<参数声明><,><参数声明>  
parameter\_declaration-> declaration\_specifiers  IDENTIFIER

参数声明-> <声明说明符><声明符>

compound\_statement-> '{' '}' | '{' statement\_list '}' | '{' declaration\_list statement\_list '}'

复合语句-> <{}>|<{<陈述序列>}>|<{<声明序列><陈述序列>}>

declaration\_list-> declaration | declaration\_list declaration

声明序列-> <声明符>|<声明序列><声明符>

Declaration->init\_declarator | init\_declarator\_list ',' init\_declarator

声明符-> <初始化说明符>|<初始化说明符><,><初始化说明符>

init\_declarator->declarator | declarator '=' initializer

初始化说明符-> <说明符>|<说明符><=><初始化程序>

Initializer-> assignment\_expression | '{' initializer\_list '}' | '{' initializer\_list ',' '}'

初始化程序-> <赋值表达式>|<{<初始化序列>}>|<{<初始化序列><,>}>

initializer\_list-> initializer | initializer\_list ',' initializer

初始化序列-> <初始化程序>|<初始化序列><,><初始化>

statement\_list-> statement | statement\_list statement

陈述序列-> <陈述语句>|<陈述序列><陈述语句>

Statement-> compound\_statement | expression\_statement | selection\_statement | iteration\_statement | jump\_statement

陈述语句-> <复合语句>|<表达式语句>|<选择语句>|<循环语句>|<转移语句>

expression\_statement->  ';'  | expression ';'

表达式语句-> <;>|<表达式语句><;>

selection\_statement->IF '(' expression ')' statement  | IF '(' expression ')' statement ELSE statement

选择语句-> <IF(表达式)><语句>|<IF(表达式)><ELSE><语句>

iteration\_statement->  WHILE '(' expression ')' statement | FOR '(' expression\_statement expression\_statement ')' statement | FOR '(' expression\_statement expression\_statement expression ')' statement

循环语句-> <WHILE(表达式)><语句>|<FOR(表达式;表达式;表达式)><语句>

jump\_statement-> CONTINUE ';'  | BREAK ';' | RETURN ';'  | RETURN expression ';'

转移语句-> <continue><;>|<break><;>|<return><;>|<return><表达式><;>

expression-> assignment\_expression | expression ',' assignment\_expression

表达式-> <赋值表达式>|<表达式><,><赋值表达式>

assignment\_expression->conditional\_expression | unary\_expression assignment\_operator assignment\_expression

赋值表达式-> <条件表达式>|<一元表达式><赋值运算符><赋值表达式>

conditional\_expression-> logical\_or\_expression | logical\_or\_expression '?' expression ':' conditional\_expression

条件表达式-> <逻辑或表达式>|<逻辑或表达式><？><表达式><:><条件表达式>

logical\_or\_expression-> logical\_and\_expression | logical\_or\_expression OR\_OP logical\_and\_expression

逻辑或表达式-> <逻辑和表达式>|<逻辑或表达式><逻辑和表达式>

logical\_and\_expression-> inclusive\_or\_expression | logical\_and\_expression AND\_OP inclusive\_or\_expression

逻辑和表达式-> <或表达式>|<和表达式><AND><或表达式>

inclusive\_or\_expression->exclusive\_or\_expression | inclusive\_or\_expression '|' exclusive\_or\_expression

包含或表达式-> <异或表达式><包含或表达式>|<异或表达式>

exclusive\_or\_expression->and\_expression | exclusive\_or\_expression '^' and\_expression

异或表达式-> <和表达式>|<异或表示><和表达式>

and\_expression->equality\_expression | and\_expression '&' equality\_expression

和表达式-> <等值式>|<和等值式><等值式>

equality\_expression-> relational\_expression | equality\_expression EQ\_OP relational\_expression | equality\_expression NE\_OP relational\_expression

等值式-> <关系表达式>

relational\_expression->shift\_expression | relational\_expression '<' shift\_expression | relational\_expression '>' shift\_expression | relational\_expression LE\_OP shift\_expression | relational\_expression GE\_OP shift\_expression

关系表达式-> <转移表达式>|<关系表达><'<'><转移表示式> |<关系表达><'>'><转移表示式>

shift\_expression->additive\_expression | shift\_expression LEFT\_OP additive\_expression | shift\_expression RIGHT\_OP additive\_expression

位移表达式-><相加表达式>|<位移表达式>LEFT\_OP<相加表达式>|<位移表达式>RIGHT\_OP<相加表达式>

additive\_expression->multiplicative\_expression | additive\_expression '+' multiplicative\_expression | additive\_expression '-' multiplicative\_expression

相加表达式-><乘法表达式>|<相加表达式>'+'<乘法表达式>|<相加表达式>'-'<乘法表达式>

multiplicative\_expression->cast\_expression | multiplicative\_expression '\*' cast\_expression | multiplicative\_expression '/' cast\_expression | multiplicative\_expression '%' cast\_expression

 乘法表达式-><计算表达式>|<乘法表达式>'\*'<计算表达式>|<乘法表达式>'/'<计算表达式>|<乘法表达式>'%'<计算表达式>

cast\_expression-> unary\_expression | '(' type\_name ')' cast\_expression

强制转换表达式-><一元表达式>|(<类型名>)<强制转换表达式>

unary\_expression-> postfix\_expression | INC\_OP unary\_expression | DEC\_OP unary\_expression | unary\_operator cast\_expression | SIZEOF unary\_expression | SIZEOF '(' type\_name ')'

一元表达式-><后缀表达式>| <INC运算一元表达式> |< DEC运算一元表达式> |< 一元运算符将表达式> | < 结构体变量的一元表达式 >| < 结构体变量的><‘(‘>< 类型名称>< ‘)’>

postfix\_expression->primary\_expression | postfix\_expression '[' expression ']' | postfix\_expression '(' ')'  | postfix\_expression '(' argument\_expression\_list ')' | postfix\_expression '.' IDENTIFIER | postfix\_expression PTR\_OP IDENTIFIER | postfix\_expression INC\_OP | postfix\_expression DEC\_OP

 后缀表达式-><基本表达式> | <后缀表达式><‘[‘><表达式><‘]‘> |< 后缀表达式><‘(‘ ><‘)'> | <后缀表达式>< ‘(‘><参数表达式列表><‘)‘ > | <后缀表达式><‘.’>< 标识符 >| <后缀表达式指针运算标识符> |< 后缀表达式INC运算> |< 后缀表达式DEC运算>

primary\_expression-> IDENTIFIER | CONSTANT | STRING\_LITERAL | '(' expression ')'基本表达式-><标识符> |<常量 >| <字符串常量> | < ‘(’><表达式><‘)’>

argument\_expression\_list-> assignment\_expression | argument\_expression\_list ',' assignment\_expression

参数表达式列表-><赋值表达式>| <参数表达式列表><’,’><赋值表达式>

unary\_operator->'&' | '\*' | '+' | '-' | '~' | '!'

 一元运算符->&|\*|+|-|~|!

assignment\_operator-> '=' | MUL\_ASSIGN | DIV\_ASSIGN | MOD\_ASSIGN | ADD\_ASSIGN  | SUB\_ASSIGN | LEFT\_ASSIGN | RIGHT\_ASSIGN  | AND\_ASSIGN | XOR\_ASSIGN | OR\_ASSIGN

 赋值运算符-> <=>|<\*=>|</=>|<%=>|<+=>|<-=>|<<<=>|<>>=>|<&=>|<^=>|<|=>

storage\_class\_specifier-> TYPEDEF | EXTERN | STATIC | AUTO | REGISTER

存储类说明符-><定义类型> |< 外部变量> |< 静态 >|< 自动> | <寄存器>

struct\_or\_union\_specifier-> struct\_or\_union IDENTIFIER '{' struct\_declaration\_list '}' | struct\_or\_union '{' struct\_declaration\_list '}' | struct\_or\_union IDENTIFIER

 结构或联合说明符-><结构或联合标识符><‘{‘><结构体声明列表><‘}‘> |< 结构或联合><‘{'><结构体声明列表><‘}‘ > |< 结构或联合标识符>

struct\_or\_union->STRUCT | UNION

结构或联合-><结构体> |< 联合>

struct\_declaration\_list->struct\_declaration | struct\_declaration\_list struct\_declaration

结构体声明列表-><结构体声明> |< 结构体列表声明结构体声明>

struct\_declaration-> specifier\_qualifier\_list struct\_declarator\_list ';'

结构体声明-><说明符限定符列表 结构说明符><‘;‘>

specifier\_qualifier\_list-> type\_specifier specifier\_qualifier\_list | type\_specifier | type\_qualifier specifier\_qualifier\_list | type\_qualifier

说明符限定符列表-><类型说明符 说明符限定符列表> |< 类型说明符> |< 类型限定符说明符限定符列表 >| <类型限定符>

struct\_declarator\_list-> struct\_declarator | struct\_declarator\_list ',' struct\_declarator

结构说明符列表-><结构体声明> |< 结构说明符列表><‘,‘><结构体声明>

struct\_declarator->declarator | ':' constant\_expression | declarator ':' constant\_expression

结构体声明-><：><声明 >| < ‘:‘><常量表达式> |< 声明><‘:‘><常量表达式>

enum\_specifier-> ENUM '{' enumerator\_list '}' | ENUM IDENTIFIER '{' enumerator\_list '}' | ENUM IDENTIFIER

枚举声明-><枚举><‘{‘><枚举器列表><‘}‘> | <枚举标识符><‘{‘><枚举器列表><‘}‘> | <枚举标识符>

enumerator\_list-> enumerator | enumerator\_list ',' enumerator

枚举符列表-><枚举符> | <枚举符列表><‘,‘>

Enumerator-> IDENTIFIER | IDENTIFIER '=' constant\_expression

枚举符枚举器-><标识符>|<标识符><‘=‘><常量表达式>

type\_qualifier-> CONST | VOLATILE

类型限定符-><常量 >|< 易失的>

type\_qualifier\_list-> type\_qualifier | type\_qualifier\_list type\_qualifier

类型限定符列表-><类型限定符> | <类型限定符列表>< 类型限定符>

parameter\_type\_list-> parameter\_list | parameter\_list ',' ELLIPSIS

参数类型列表-><参数列表> | <参数列表><‘,‘><省略符号>

parameter\_list-> parameter\_declaration | parameter\_list ',' parameter\_declaration

参数列表-><：><声明参数> |< 参数列表><‘,‘><声明参数>

type\_name-> specifier\_qualifier\_list | specifier\_qualifier\_list abstract\_declarator

类型名-><说明修饰列表>|<说明修饰列表 抽象说明符>

abstract\_declarator-> pointer | direct\_abstract\_declarator | pointer direct\_abstract\_declarator

抽象说明符-><指针> | <直接抽象说明符> | <指针直接抽象说明符>

direct\_abstract\_declarator-> '(' abstract\_declarator ')' | '[' ']' | '[' constant\_expression ']' | direct\_abstract\_declarator '[' ']' | direct\_abstract\_declarator '[' constant\_expression ']'  | '(' ')' | '(' parameter\_type\_list ')' | direct\_abstract\_declarator '(' ')' | direct\_abstract\_declarator '(' parameter\_type\_list ')'

直接抽象说明符-> <‘(‘> <抽象说明符 ><‘)‘ > |< ‘[‘ ><‘]‘ >| < ‘[‘>< 常量表达式><‘]‘>| <直接抽象说明符><‘[‘>< ‘]‘> | <直接抽象说明符><‘[‘>< 常量表达式>< ‘]‘> | < ‘(‘>< ‘)‘> | < ‘(‘><参数类型列表><‘)‘> |< 直接抽象说明符><‘(‘>< ‘)‘> | <直接抽象说明符><‘(‘><参数类型列表><‘)‘>

labeled\_statement-> IDENTIFIER ':' statement | CASE constant\_expression ':' statement | DEFAULT ':' statement

有标号语句-><标识符><‘:><‘语句> | <CASE常量表达式><‘:‘><语句 > | < DEFAULT><‘:‘><语句>