## 词法：

1. 定义

|  |  |
| --- | --- |
| letter | [\_a-zA-Z](注意：包括下划线) |
| digit | [0-9] |
| Hex | [a-fA-F0-9] |

1. Token

|  |  |
| --- | --- |
| Token | 识别 |
| NUMBER  （包含十进制表示的number  以及十六进制的number）  注意：本编译器仅支持int类型计算因此NUMBER的值是int类型 | [1-9]{digit}\*  (int yyval=十进制大小) |
| 0[0-7]\*  (int yyval=十进制大小) |
| 0[xX]{Hex}+  (int yyval=十进制大小) |
| CONST | “const” |
| INT | “int” |
| VOID | “void” |
| IF | “if” |
| ELSE | “else” |
| WHILE | “while” |
| BREAK | “break” |
| CONTINUE | “continue” |
| RETURN | “return” |
| IDENT | {letter}({letter}|{digit})\*  (yyval=字符串) |
| ‘+’ | “+” |
| ‘-’ | “-” |
| ‘\*’ | “\*” |
| ‘/’ | “/” |
| ‘%’ | “%” |
| LE\_OP | “<=” |
| GE\_OP | “>=” |
| ‘<’ | “<” |
| ‘>’ | “>” |
| EQ\_OP | “==” |
| NE\_OP | “!=” |
| ‘!’ | “!” |
| ‘=’ | “=” |
| AND\_OP | “&&” |
| OR\_OP | “||” |
| ‘;’ | “;” |
| ‘,’ | “,” |
| ‘[’ | “[“ |
| ‘]’ | “]” |
| ‘(’ | “(“ |
| ‘)’ | “)” |
| ‘{’ | “{“ |
| ‘}’ | “}” |

共34个token

1. 辅助：

词法需处理注释：

|  |  |
| --- | --- |
| “//”[^\n]\* | just consume |
| “/\*” | comment(); |

**void comment(void){**

**char c, prev = 0;**

**while ((c = input()) != 0){ /\* (EOF maps to 0) \*/**

**if (c == '/' && prev == '\*')**

**return;**

**prev = c;**

**}**

**error("unterminated comment");**

**}参见ASCII C的LEX网页**

词法需跳过空格和错误单词

|  |  |
| --- | --- |
| [ \t\n\v\f] | just consume（注意：包括空格） |
| . | error();错误处理 |

## 文法：

相较Sysy有一些修改：

1. 改为仅支持一维数组(方便后续的工作)
2. EBNF形式改为BNF形式
3. 注意：计算优先级和结合性已在文法中体现

|  |  |
| --- | --- |
| 类型 | 文法 |
| 编译单元 | CompUnit🡪 CompUnit Decl | Decl  | CompUnit FunDef | FunDef |
| 声明 | Decl🡪ConstDecl | VarDecl |
| 常量声明 | ConstDecl🡪 **CONST** BType ConstDef TempA ‘**;’**  TempA 🡪 TempA ‘**,’** ConstDef |
| 基本类型 | BType🡪**INT** |
| 常数定义 | ConstDef🡪**IDENT** ‘**=’** ConstInitVal  | **IDENT** ‘**[’** ConstExp ‘**]’ ‘=’** ConstInitVal |
| 常量初值 | ConstInitVal🡪ConstExp  |‘**{’ ‘}’**  |‘**{’** ConstExp TempB **‘}’**  TempB🡪 TempB ‘**,’** ConstExp |
| 变量声明 | VarDecl🡪Btype VarDef TempC ‘**;’**  TempC🡪TempC **‘,’** VarDef |
| 变量定义 | VarDef🡪 **IDENT**  | **IDENT** ‘**[’** ConstExp **‘]’**  | **IDENT** **‘=’** InitVal  | **IDENT** **‘[’** ConstExp ‘**]’ ‘=’** InitVal |
| 变量初值 | InitVal🡪Exp  | **‘{’ ‘}’**  | **‘{’**Exp TempD **‘}’**  TempD🡪TempD **‘,’** Exp |
| 函数定义 | FuncDef🡪FuncType **IDENT** **‘(’ ‘)’** Block  | FuncType **IDENT** **‘(’** FuncFParams **‘)’** Block |
| 函数类型 | FuncType🡪**VOID** | **INT** |
| 函数形参表 | FuncFParams🡪 FuncFParam TempG  TempE🡪TempE ‘**,’** FuncFParam |
| 函数形参 | FuncFParam🡪Btype **IDENT**  | Btype **IDENT** **‘[’ ‘]’** |
| 语句块 | Block🡪**‘{’** TempF **‘}’**  TempF🡪TempF BlockItem |
| 语句块项 | BlockItem🡪Decl | Stmt |
| 语句 | Stmt🡪LVal **‘=’** Exp **‘;’** | Exp **‘;’** | **‘;’** | Block  | **IF** **‘(’** Cond **‘)’** Stmt  | **IF ‘(’** Cond **‘)’** Stmt **ELSE** Stmt  | **WHILE ‘(’** Cond **‘)’** Stmt  | **BREAK ‘;’** | **CONTINUE ‘;’**  | **RETURN ‘;’** | **RETURN** Exp **‘;’** |
| 表达式 | EXP🡪AddExp |
| 条件表达式 | Cond🡪LOrExp |
| 左值表达式 | LVal🡪 **IDENT** | **IDENT** **‘[’** Exp **‘]’** |
| 基本表达式 | PrimaryExp🡪**‘(’** Exp **‘)’** | LVal| **NUMBER** |
| 一元表达式 | UnaryExp🡪PrimaryExp  | **IDENT** **‘(’ ‘)’**  | **IDENT** **‘(’** FuncRParams **‘)’**  | UnaryOp UnaryExp |
| 单目运算符 | UnaryOp🡪**‘+’**|**‘-’**|‘!’  (！仅在条件表达式中出现，需要在yacc中解决) |
| 函数实参表 | FuncRParams🡪Exp TempG  TempG🡪TempG **‘,’** Exp |
| 加减表达式 | AddExp🡪 MulExp  | AddExp **‘+’** MulExp  | AddExp **‘-’** MulExp |
| 乘除模表达式 | MulExp🡪UnaryExp  | MulExp **‘\*’** UnaryExp  | MulExp **‘/’** UnaryExp  | MulExp **‘%’** UnaryExp |
| 关系表达式 | RelExp🡪AddExp  | RelExp **‘<’** AddExp  | RelExp **‘>’** AddExp  | RelExp **LE\_OP** AddExp  | RelExp **GE\_OP** AddExp |
| 相等性表达式 | EqExp🡪RelExp  | EqExp **EQ\_OP** RelExp  | EqExp **NE\_OP** RelExp |
| 逻辑与表达式 | LAndExp🡪EqExp | LAndExp **AND\_OP** EqExp |
| 逻辑或表达式 | LOrExp🡪LAndExp | LOrExp **OR\_OP** LAndExp |
| 表达式 | ConstExp🡪AddExp  (使用的Ident必须是常量，需要在yacc中解决) |

ConstExp🡪AddExp (使用的Ident必须是常量，需要在yacc中解决)：

如果解决不了可以简化为若是ConstExp往后推的表达式只能是Number的计算式，不可以推出Lval，若推出则直接报错。