### 文法解读文档

16061075 黄霁昀

注：由于文法之间存在推导，本文档将文法分组解读，文法限制在分析中。

一．

1. 文法

＜加法运算符＞ ::= +｜-

＜乘法运算符＞ ::= \*｜/

＜关系运算符＞ ::= <｜<=｜>｜>=｜!=｜==

＜字母＞ ::= ＿｜a｜．．．｜z｜A｜．．．｜Z

＜数字＞ ::= ０｜＜非零数字＞

＜非零数字＞ ::= １｜．．．｜９

＜字符＞ ::= '＜加法运算符＞'｜'＜乘法运算符＞'｜'＜字母＞'｜'＜数字＞'

＜字符串＞ ::= "｛十进制编码为32,33,35-126的ASCII字符｝"

1. 范例

‘+’ ‘\*’ ‘a’ ‘0’ //字符

“string!” //字符串

1. 分析

字母不仅包括26个英文字母大小写，还包括下划线。

字符只包括加法运算符、乘法运算符、字母、数字，不包括关系运算符、标点符号等其他字符。

字符声明需要单引号。

字符串不仅是字符的组合，还包括其他符号，但不包括换行符、回车符和双引号，即不识别\t,\n。

字符串声明需要双引号。

二．

1. 文法

＜无符号整数＞ ::= ＜非零数字＞｛＜数字＞｝｜０

＜整数＞ ::= ［＋｜－］＜无符号整数＞

1. 范例

17 100 //无符号整数

+17 -100 1 -0 //整数

1. 分析

整数不包括前导零，符号不强制，存在+0,-0。

三．

1. 文法

＜标识符＞ ::= ＜字母＞｛＜字母＞｜＜数字＞｝

1. 范例

a aa a1 a\_1

1. 分析

标识符必须由字母开头，字母后面可以加数字，标识符区分大小写。

四．

1. 文法

＜变量说明＞ ::= ＜变量定义＞;{＜变量定义＞;}

＜变量定义＞ ::= ＜类型标识符＞(＜标识符＞|＜标识符＞'['＜无符号整数＞']'){,(＜标识符＞|＜标识符＞'['＜无符号整数＞']') }

//＜无符号整数＞表示数组元素的个数，其值需大于0

＜类型标识符＞ ::= int | char

1. 范例

int a[1],b;

char c[1],d;

1. 分析

只支持int和char，同类型变量可以一起声明，同类型单一变量可以和数组一起声明，数组中中括号内只能为大于0无符号整数。

五．

1. 文法

＜常量说明＞ ::= const＜常量定义＞;{ const＜常量定义＞;}

＜常量定义＞ ::= int＜标识符＞＝＜整数＞{,＜标识符＞＝＜整数＞}

| char＜标识符＞＝＜字符＞{,＜标识符＞＝＜字符＞}

2.范例

const int a=1,b=2;

const char c = ‘a’;

1. 分析

常量中，同一类型的单一变量可以在一起声明，但不同类型的或是单一变量和数组不可以，区别于变量声明。不存在常量数组。

六．

1. 文法

＜表达式＞ ::= ［＋｜－］＜项＞{＜加法运算符＞＜项＞}

//[+|-]只作用于第一个<项>

＜项＞ ::= ＜因子＞{＜乘法运算符＞＜因子＞}

＜因子＞ ::= ＜标识符＞｜＜标识符＞'['＜表达式＞']'｜＜整数＞|＜字符＞｜＜有返回值函数调用语句＞|'('＜表达式＞')'

1. 范例

+aa //标识符做表达式

-a[1] //数组标识符做表达式

1 //整数做表达式

a // 字符做表达式

getSym() //有返回值函数调用语句做表达式

aa +b[1] \* c //复杂表达式

1. 分析

表达式由若干个项和加法运算符构成，不同的项之间只有一个加法运算符，第一个项之前的加法运算符不是必须。

项由若干个因子和乘法运算符构成，不同的因子之间只有一个乘法运算符，第一个因子前没有乘法运算符。

因子由标识符、标识符[表达式]、整数、字符、有返回值的函数调用语句构成。数组下标可以为表达式,即可以a[1+1],和数组定义不同。

五．

1. 文法

＜条件语句＞ ::= if '('＜条件＞')'＜语句＞［else＜语句＞］

＜条件＞ ::= ＜表达式＞＜关系运算符＞＜表达式＞｜＜表达式＞

//表达式为0条件为假，否则为真

1. 范例

a > b //条件

if(a > b){

...

}else{

...

}

1. 分析

条件可以为两个表达式之间的关系，如a>b，也可以为一个表达式，如1-1。

else部分可以不存在。

七．

1. 文法

＜赋值语句＞ ::= ＜标识符＞＝＜表达式＞|＜标识符＞'['＜表达式＞']'=＜表达式＞

1. 范例

a = 1

a = b[1]

a = 1+1

1. 分析

定义赋值语句，数组下标可以为表达式，和数组定义不同。

八．

1. 文法

＜循环语句＞ ::= do＜语句＞while '('＜条件＞')' |for'('＜标识符＞＝＜表达式＞; ＜条件＞;＜标识符＞＝＜标识符＞(+|-)＜步长＞')'＜语句＞

＜步长＞::= ＜无符号整数＞

1. 范例

//do while

do{

...

} while(a>1)

//for

for(i = 0;i<10;i = i+1){...}

1. 分析

循环语句支持do while和for。注意for循环不允许类似i++，i--格式出现。

九．

1. 文法

＜有返回值函数调用语句＞ ::= ＜标识符＞'('＜值参数表＞')'

＜无返回值函数调用语句＞ ::= ＜标识符＞'('＜值参数表＞')'

＜值参数表＞ ::= ＜表达式＞{,＜表达式＞}｜＜空＞

1. 范例

funcWithReturn(a,b)

funcWithNoReturn()

1. 分析

参数列表可以为空或者多个。

十．

1. 文法

＜读语句＞ ::= scanf '('＜标识符＞{,＜标识符＞}')'

＜写语句＞ ::= printf'('＜字符串＞,＜表达式＞')'|printf '('＜字符串＞')'|printf '('＜表达式＞')'

＜返回语句＞ ::= return['('＜表达式＞')']

1. 范例

scanf(a)

printf(b)

printf(“print”)

printf(“%c”,c)

return(c)

return

1. 分析

读语句没有scanf(“%c”,&c)这种形式，只有scanf(a,b,c)。

写语句支持printf(“%c”,c)这种形式，区别于读语句。

return可以有值也可以没有。

十一．

1. 文法

＜语句＞ ::= ＜条件语句＞｜＜循环语句＞｜＜有返回值函数调用语句＞;

|＜无返回值函数调用语句＞;｜＜赋值语句＞;｜＜读语句＞;｜＜写语句＞;｜＜空＞;｜＜返回语句＞;

＜语句列＞ ::=｛＜语句＞｝

1. 范例

语句部分见上上述具体语句，其中除条件语句和循环语句外，其他语句后要加分号。

语句列为语句加上大括号。

1. 分析

语句可以为空，且空语句后面也有分号，也就是说编译器支持处理多余分号。

十二．

1. 文法

＜声明头部＞ ::= int＜标识符＞|char＜标识符＞

＜有返回值函数定义＞ ::= ＜声明头部＞'('＜参数表＞')' '{'＜复合语句＞'}'

＜无返回值函数定义＞ ::= void＜标识符＞'('＜参数表＞')''{'＜复合语句＞'}'

＜参数表＞ ::= ＜类型标识符＞＜标识符＞{,＜类型标识符＞＜标识符＞}| ＜空＞

1. 范例

//有返回值函数定义，有参数

int funcWithReturn(int a, int b){

int c;

c = a + b;

return(c);

}

//无返回值函数定义，无参数

void funcWithNoReturn(){

printf(“no return”);

}

3.分析

函数可以有参也可以无参，可以有返回值也可以无返回值，返回值只有int和char类型。

十三．

1. 文法

＜程序＞ ::= ［＜常量说明＞］［＜变量说明＞］{＜有返回值函数定义＞|＜无返回值函数定义＞}＜主函数＞

＜复合语句＞ ::= ［＜常量说明＞］［＜变量说明＞］＜语句列＞

＜主函数＞ ::= void main'('')' '{'＜复合语句＞'}'

1. 范例

const int a=1,b=2;

const char c = ‘a’;

int d,bb[4];

char cc[3];

int plus(int a,int b){

int c;

c = a + b;

return(c);

}

void printABC(){

printf(“abc”);

}

void main(){

int d;

d = plus(a,b);

printABC();

}

1. 分析

程序为C0文法的标准格式。

程序各部分存在顺序。只能先声明常量，在声明变量，在定义函数，最后为主函数。顺序的颠倒将被视为错误。

程序只有主函数为必须部分，其他都为非必须，且其他部分虽然顺序不能颠倒但是可以有多个声明。

有返回值和无返回值函数声明没有顺序。