

**编译原理课程设计**

实验报告

北京航空航天大学

计算机学院

陈麒先

16061160

二○一八年十一月

**郑重声明**

**关于诚实守信公约：**

**本实验报告中参考了文献或互联网资料的部分均有引用标注，另有与老师或同学研究后的成果引用将在特别致谢中说明，其余部分均为本人通过查阅课件、课堂笔记和教材后独立思考的结果。另，实验报告撰写仓促，如有错误，在所难免，欢迎批评指正！特此声明。**

**陈麒先**

**原创性声明**

**本实验报告未注释部分内容由作者原创。抄袭行为在任何情况下都是不能容忍的(COPY is strictly prohibited under any circumstances)！转载或引用须征得作者本人同意，并注明出处！勿谓言之不预！**

**陈麒先**



# 文法解读文档

1.

＜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＞

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

＜变量说明＞ ::= ＜变量定义＞;{＜变量定义＞;}

＜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＞'}'

＜主函数＞ ::= void main'('')''{'＜复合语句＞'}'

【范例】

const int a = 1; // 常量说明

int b; // 变量说明

int add(int x,int y){ //有返回值函数定义

return x + y;

}

void print(){ //无返回值函数定义

printf("helloworld");

}

void main(){ //主函数

int c;

b = 1;

c = add(a,b);

print();

}

【分析】

- 根据程序的文法可知，程序各个组成成分的顺序已经被限定好了，不能随便更改顺序，如“int a,b; const x=10;”此种顺序是不允许的。

- 每一个程序按顺序由常量说明部分，变量说明部分，有无返回值函数定义部分，以及主函数组成。其中常量，变量，过程及函数说明说明，复合语句之间有严格的先后顺序，不能打乱。而相连的有无返回值的函数说明部分则可以随意改变顺序。变量说明，常量说明，过程及函数说明对于一个分程序来说均为可有可无的部分，有无返回值的函数定义可以有零到多个。只有最后的主函数是必须存在的。

2.

＜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＞}

＜标识符＞ ::= ＜字母＞｛＜字母＞｜＜数字＞｝

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

＜无符号整数＞ ::= ＜非零数字＞｛＜数字＞｝| 0

＜整数＞ ::= ［＋｜－］＜无符号整数＞

＜数字＞ ::= ０｜＜非零数字＞

＜非零数字＞ ::= １｜．．．｜９

＜字母＞ ::= ＿｜a｜．．．｜z｜A｜．．．｜Z

＜加法运算符＞ ::= +｜-

＜乘法运算符＞ ::= \*｜/

【范例】

const int \_aA = 0; //int常量

const char Ch = '+'; //char常量

【分析】

- 常量说明部分可以说明一至多个常量，每条定义必须有const开始，以分号结束，中间是常量定义。

- 常量定义可以是int也可以是char，对统一类型可定义多个常量，以逗号分隔，定义由等号连接，左边必须为标识符。

- 标识符必须以字母开头，有字母数字组成。

- 字母包含大小写和下划线，程序对大小写敏感。

- 整数的开头为可选的+-加上无符号整数。

- 无符号整数可以是0或者不允许前0的其他数字。

- 字符是能是+-\*/\_大小写字母和数字。

3.

＜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＞']' )}

＜类型标识符＞ ::= int | char

【范例】

int a; // 声明int变量

char b[100]; // 声明长100字符数组

【分析】

- 变量定义不支持在定义时直接赋值，如不允许‘int a = 1;’。

- 类型标识符包括2种，int和char。

- 变量定义包括变量和数组定义,数组定义中无符号整数值表示数组长度，值大于0。

- char类型的变量或常量，用字符的ASCII码对应的整数参加运算。

4.

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

＜声明头部＞ ::= int＜标识符＞ |char＜标识符＞

＜参数表＞ ::= ＜参数＞{,＜参数＞}| ＜空>

＜参数＞ ::= ＜类型标识符＞＜标识符＞

＜复合语句＞ ::= ［＜常量说明＞］［＜变量说明＞］＜语句列＞

【范例】

int add\_1(int x,int y){ //声明头部 参数表 复合语句

const int a = 1; // 常量说明

int result; // 变量说明

result = x + y + a; // 语句列

return result;

}

【分析】

- 有返回值函数定义有声明头部（参数表）{复合语句}组成，顺序不得更改，三者不可缺省，（）{}为必有字符。

- 声明头部由int,char+标识符组成，表示声明的函数返回值类型，二者只能选择其一进行声明。

- 参数表由零至多个参数，传入参数可以为空。各个参数之间由逗号分隔。

- 参数为一个类型标识符加一个标识符组成，不允许多个标识符的情况出现在参数中，如不允许‘int a,b’。

- 复合语句由常量、变量说明和语句列组成，顺序不可调换，常量、变量为可选项，必须有语句列。

- 有返回值的函数定义文法上允许无返回值，但是不符合语义需要报错。

5.

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

【范例】

void hello(){ //标识符 参数表 复合语句

printf("hello world!");

}

【分析】

- 无返回值即void 标识符为函数名，其余部分均与有返回值函数定义相同。

- 此处范例给出了参数表空的情况。

- return();是不合法的。

6.

＜主函数＞ ::= void main'('')''{'＜复合语句＞'}'

【范例】

void main(){

printf("hello world!");

}

【分析】

- 主函数中的各个成分是确定的void main(){ 中间为复合语句块 最后以}结尾。

- 无返回值的函数可以有return语句，有返回值可以没有return语句。

7.

＜表达式＞ ::= ［＋｜－］＜项＞{＜加法运算符＞＜项＞}

＜项＞ ::= ＜因子＞{＜乘法运算符＞＜因子＞}

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

【范例】

① a + b

② a[i] \* b + (c[j + k] + d \* e)

③ a \* 4 - 'a'/add(x,y)

④ - a[i \* 4 + base] + c/(d \* e[j])

【分析】

- 表达式有一至多个项组成，每个项之间有加法运算符链接，[+|-]只作用于第一个项，即第一项之前也可有加法算符。

- 项由一至多个因子组成，因子之间由乘法运算符连接。

- 因子可以是一个标识符，即常量或者变量；或是＜标识符＞'['＜表达式＞']'，即一个数组，数组下标为【】中的内容，也可以通过一个表达式进行计算；或是'('＜表达式＞')'由（）包含的部分整体为一个表达式的求值语句；或是一个带返回值的函数调用，通过函数获取数值；或是整数、字符。综上所述，因子应该得到一个数值带回到项中进行计算。

8.

＜语句＞ ::= ＜条件语句＞｜＜循环语句＞| '{'＜语句列＞'}'| ＜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＞;|＜情况语句＞｜＜返回语句＞;

【范例】

有关语句的范例将通过每个语句语法成分分别给出，敬请期待。

【分析】

- 语句的定义比较易于理解，即是各种不同的语句类型的集合。其中要注意有分号结尾的情况，除了条件、循环、情况语句和'{'＜语句列＞'}'之外，都有分号结尾。注意空语句直接只有一个分号。

9.

＜条件语句＞ ::= if '('＜条件＞')'＜语句＞

＜条件＞ ::= ＜表达式＞＜关系运算符＞＜表达式＞｜＜表达式＞

＜关系运算符＞ ::= <｜<=｜>｜>=｜!=｜==

【范例】

if(a == b){

printf("hello world");

}

【分析】

- 条件语句由关键字if引导，根据（）中的条件进行跳转，若条件为真则执行语句否则不执行。

- 条件分两种，一种是由有关系运算符连接的两个表达式形成的判断条件，另一种是单表达式形成的判断条件。由前文分析可知，表达式会返回数值，表达式为0条件为假，否则为真；而由关系运算符连接的两个数值运算结果将返回0、1，其中0为假，1为真。

- 关系运算符为数值比较运算，返回真值。

10.

＜循环语句＞ ::= while '('＜条件＞')'＜语句＞

【范例】

while(i < n){

printf(i);

i ++;

}

【分析】

- 关键字while ，条件为上文分析过的，如果条件值为真，则执行语句，直到不满足条件退出循环。

11.

＜语句列＞ ::= ｛＜语句＞｝

【范例】

a = 1 + 1;

b[9] = a;

c = b[9] \* 6;

【分析】

- 语句列即一组语句的序列集合，由零到多个语句构成。

12.

＜有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＞

【范例】

add(x,y)

【分析】

- 注意有返回值函数调用语句在语句定义中，其应有的分号在语句定义中体现。

- 注意各个成分不得缺省和调换顺序，标识符为调用的函数名。

- 值参数表为传入的实参列表，必须与函数定义的形参相对应，每个相应位置都应该出现数值，可通过表达式求得。

13.

＜无返回值函数调用语句＞ ::= ＜标识符＞'('＜值参数表＞')'

【范例】

print()

【分析】

- 比有返回值的函数调用简单，标识符为函数名，由（）包含参数列表，这里的范例给出值参数表为空的情况。

- 同样分号是定义在语句中的。

14.

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

【范例】

① a = 1

② b[0] = 1

③ c[i \* 4 + base] = 1

【分析】

- 赋值可以为变量或数组中的某一个元素赋值，有=连接。

- 所有形如＜标识符＞'['＜表达式＞']'均表示一个数组，表达式中计算数组索引值，数组的下标从0开始。

- 赋值语句等号右边不能是比较表达式。

15.

＜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＞')']

【范例】

① scanf(a)

② printf("hello world",a + b)

③ return （a + b）;

【分析】

- 读写语句关键词和（，）不可缺少，按顺序排列。标识符为变量名或数组名。

- 写语句分三种，可以写字符串也可写表达式，也可同时写。字符输出char，其余情况输出整数。不能printf(),字符串中不允许双引号（34号字符）。

- 返回语句由关键字return起始，后边的返回值为可选项。对于无返回值的函数定义直接return即可，若有返回值则必须在表达式外加上（）。

- 写语句中，字符串原样输出，单个char类型的变量或常量输出字符，其他表达式按整型输出，写语句自带换行。

- 读语句和变量类型有关，char读入字符，int读整数或字符的ascii码。

16.

＜情况语句＞ ::= switch '('＜表达式＞')' '{'＜情况表＞＜缺省＞ '}'

＜情况表＞ ::= ＜情况子语句＞{＜情况子语句＞}

＜情况子语句＞ ::= case＜常量＞：＜语句＞

＜缺省＞ ::= default : ＜语句＞|＜空＞

【范例】

switch(state){

case 0:{ // 情况表

a = 0;

b = 1;

}

case 1 : a = 1;

default: a = 10; // 缺省

}

【分析】

- 情况语句由关键字switch(){}引导，（）中的表达式为待判断的数量，由{}包含情况表case和缺省情况default。

- 情况表有一至多个情况子语句构成。

- 每个情况子语句由case引导，后面的常量为判断条件，若与表达式的值相等则执行冒号后面的语句。范例中给出了两种可能的语句情况。

- 缺省有default:引导，当前面没有一个case生效的时候执行default后面的语句，缺省情况可以为空。

- <空>是一个特殊的符号，他是一个终结符号ε。

- 情况语句中，switch后面的表达式和case后面的常量只允许出现int和char类型；每个情况子语句执行完毕后，不继续执行后面的情况子语句。

# 体会与感想

本次实验让我对本次编译技术课程设计的文法有了进一步的了解。本次实验共耗时在8小时左右，整体难度适中。但课程提供的指导过于简略，指导书相关要求与解释较为模糊，不能起到良好的引导作用。

希望能有更为清晰易懂的题目要求和老师、助教的指导来指导我们更高效的完成实验，也希望理论课能增加相关与实验联系更为紧密的章节进行讲解说明。

**陈麒先**

2018/11/10