14231006\_田宇\_文法解读

**1. ＜加法运算符＞ ::= +｜-**

作用：将加法运算符定义为“+”或“-”。其中，“+”表示相加，“-”表示相减。

限定条件：双目运算符。若左操作数缺省，则表示右操作数为有符号数，“+”表示正数，“-”表示负数。一般不允许右操作符缺省。

示例：（设a、b为无符号数）

①a+b：a加上b；

②a-b：a减去b；

③+a：正数，其数值为a；

④-b：负数，其绝对值为b。

1. **＜乘法运算符＞ ::= \*｜/**

作用：将乘法运算符定义为“\*”或“/”。其中，“\*”表示乘，“/”表示除。

限定条件：双目运算符。一般不允许操作数缺省。

示例：（设a、b为无符号数）

①a\*b：a乘b；

②a/b：a除b。

1. **＜关系运算符＞ ::= <｜<=｜>｜>=｜!=｜==**

作用：将关系运算符定义为“<”或“<=”或“>”或“>=”或“!=”或“==”。其中，“<”表示小于，“<=”表示小于或等于，“>”表示大于，“>=”表示大于或等于，“!=”表示不等于，“==”表示等于。

限定条件：双目运算符。一般不允许操作数缺省。由两个符号组合形成的关系运算符（如：<=，>=，!=，==）中间不允许插入其他任何字符，必须连在一起使用。

示例：（设a、b为无符号数）

①a<b：a小于b；

②a<=b：a小于或等于b；

③a>b：a大于b；

④a>=b：a大于或等于b；

⑤a!=b：a不等于b；

⑥a==b：a等于b。

1. **＜字母＞ ::= \_｜a｜．．．｜z｜A｜．．．｜Z**

作用：将字母定义为“\_”（下划线）或[a-z]或[A-Z]。其中，[a-z]为从“a”到“z”的小写字母，[A-Z]表示从“A”到“Z”的大写字母。

限定条件：单字母。不区分大小写。

示例：

①a：字母a；

②A：字母A；

③\_：（字母）下划线。

1. **＜数字＞ ::= ０｜＜非零数字＞**

作用：将数字定义为“0”或非零数字。

限定条件：单数字。

示例：

①0：数字0；

②1：数字1。

1. **＜非零数字＞ ::= １｜．．．｜９**

作用：将非零数字定义为[1-9]。其中，[1-9]表示从“1”到“9”的数字。

限定条件：单数字。

示例：9（非零数字9）。

1. **＜字符＞ ::= '＜加法运算符＞'**

**｜'＜乘法运算符＞'**

**｜'＜字母＞'**

**｜'＜数字＞'**

作用：将字符定义为包含加法运算符、乘法运算符、字母和数字的集合的元素。其中，加法运算符为“+”或“-”，乘法运算符“\*”或“/”，字母为“\_”（下划线）或[a-z]或[A-Z]，数字为[0-9]。

限定条件：单字符。不区分字母的大小写。

示例：

①’+’：字符’+’；

②’\*’：字符’\*’；

③’\_’：字符’\_’；

④’a’：字符’a’；

⑤’0’：字符’0’。

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

作用：将字符串定义为若干字符连接形成的有序组合。

限定条件：组成字符为十进制编码32、33、35-126的ASCII字符。字符串可为空。

示例：

①””：空字符串；

②”abc”：字符串abc。

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＞**

作用：将程序定义为说明块、若干定义块和主函数的组合。其中，说明块包括常量说明和变量说明，定义块为有返回值函数定义或无返回值函数定义。

限定条件：允许说明块缺省。定义块可缺省，也可有若干个。一般不允许主函数缺省。

示例：

const int max=100, min=0;

int in\_1, in\_2, out;

int add(int a, int b) { return(a+b); }

void print(int res) { if(res<=max) { if(res>min) { printf(res); printf(“, ”); } } return; }

void main() { scanf(in\_1, in\_2); out=add(int\_1, int\_2); print(out); return;}

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

作用：将常量说明定义为由不少于一条“const＜常量定义＞;”组成的语句块。

限定条件：关键字const表明为常量。

示例：

①const int a=1;：常量a为整数1；

②const int a=1; const char c=’d’;：常量a为整数1，常量c为字符’d’。

**11. ＜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＞}**

作用：将常量定义定义为类型（int或char）和不少于一条定义语句的顺序组合。其中，int表明为整型常量（整数），chat表明为字符型常量（字符）。

限定条件：若干条定义语句之间用逗号,分隔。每处常量定义只允许出现int、char中的一种类型。

示例：

①int a=3：a被定义为整数3；

②char a=’c’：a被定义为字符’c’。

1. **＜无符号整数＞ ::= ＜非零数字＞｛＜数字＞｝**

作用：将无符号整数定义为若干数字的有序组合。其中，数字包括0和非零数字。

限定条件：无符号整数的首数字不为0。

示例：

①1：无符号整数1；

②10：无符号整数10。

1. **＜整数＞ ::= ［＋｜－］＜无符号整数＞**

**｜０**

作用：将整数定义为有符号（“+”或“-”）整数或0。其中，“+”表明正整数，“-”表明负整数。

限定条件：若符号缺省且数值不为0，则表明正整数。

示例：

①0：整数0；

②1：正整数；

③+2：正整数2；

④-2：负整数-3。

**14. ＜标识符＞ ::= ＜字母＞｛＜字母＞｜＜数字＞｝**

作用：将标识符定义为若干字母和数字的有序组合。

限定条件：不允许为空。首字符必须为字母。不区分字母大小写。

示例：

①a：标识符a；

②\_007：标识符\_007；

③a1A2：标识符a1A2；

④\_abc56：标识符\_abc56。

1. **＜声明头部＞ ::= int ＜标识符＞**

**| char ＜标识符＞**

作用：将声明头部定义为类型（int或char）和标识符的顺序组合。

限定条件：每处声明头部只允许出现int、char中的一种类型。。

示例：

①int a：声明整型a；

②char a：声明字符型a。

**16. ＜变量说明＞ ::= ＜变量定义＞;{＜变量定义＞;}**

作用：将变量说明定义为若干条变量定义。

限定条件：变量定义不少于一条。若干条变量定义用分号;分隔。

示例：

①int a, b；：整型变量a，整型变量b；

②char String[10];：字符型变量数组String（数组维数为10）。

**17. ＜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＞‘]’ )}**

作用：将变量定义定义为类型标识符和若干标识符或标识符[无符号整数]的顺序组合。其中，标识符[无符号整数]表示数组定义，无符号整数表明数组维数。

限定条件：至少有一个标识符或标识符[无符号整数]。若干标识符或标识符[无符号整数]之间用逗号,分隔。

示例：

①int a, b：整型变量a，整型变量b；

②char String[10]：字符型变量数组String，数组维数为10。

1. **＜常量＞ ::= ＜整数＞**

**|＜字符＞**

作用：将常量定义为整数或字符。

限定条件：整数包括正整数、负整数和0。

示例：

①0：整数常量；

②’a’：字符常量；

③-1：整数常量。

1. **＜类型标识符＞ ::= int**

**| char**

作用：将类型标识符定义为int或char。

限定条件：类型标识符为固定字符串。

示例：

①int：整型标识符；

②char：字符型标识符。

1. **＜有返回值函数定义＞ ::= ＜声明头部＞‘(’＜参数＞‘)’ ‘{’＜复合语句＞‘}’**

作用：将有返回值函数定义定义为声明头部、(参数)和{复合语句}的有序组合。其中，声明头部中的类型即为函数返回值的类型。

限定条件：参数外部用左右小括号()括起来，复合语句外部用左右大括号{}括起来。

示例：

①int add(int a, int b) {int sum; sum=a+b; return(sum);}：add函数，计算两个整型形参值的和，返回值为整型；

②char sub(char a, int b) {char res; res=a-b; return(res);}：sub函数，计算两个整型形参的差，返回值为浮点型；

③char get() {return ‘d’;}：get函数，返回值为字符型。

1. **＜无返回值函数定义＞ ::= void＜标识符＞‘(’＜参数＞‘)’ ‘{’＜复合语句＞‘}’**

作用：将无返回值函数定义定义为void、标识符、(参数)和{复合语句}的有序组合。其中，void表明该函数无返回值。

限定条件：参数外部用左右小括号()括起来，复合语句外部用左右大括号{}括起来。

示例：

①void print() {printf(“Hello World!”); return;}：print函数，打印字符串“Hello World!”；

②void read(int tmp) { scanf(tmp); return;}：read函数，读入一个整数tmp。

**22. ＜复合语句＞ ::= ［＜常量说明＞］［＜变量说明＞］＜语句列＞**

作用：将复合语句定义为若干常量说明、若干变量说明和语句列的顺序组合。

限定条件：常量说明和变量声明允许缺省。

示例：

①const int a=1, b=2; int sum; sum=a+b;：已知常量a、b，计算其和并赋值给变量sum；

②int a; a=233;：将233赋给整型变量a；

③return(0);：返回值赋为0。

**23. ＜参数＞ ::= ＜参数表＞**

作用：将参数定义为参数表。

限定条件：无。

示例：

①int a：整型参数a；

②int b, char c：整型参数b，字符型参数c；

③：空，参数缺省。

1. **＜参数表＞ ::= ＜类型标识符＞＜标识符＞{,＜类型标识符＞＜标识符＞}**

**| ＜空＞**

作用：将参数表定义为若干个类型标识符和标识符的有序组合或空。

限定条件：参数表允许缺省。若干个若干个类型标识符和标识符的有序组合之间用逗号,分隔。

示例：

①int a：整型参数a；

②int b, char c：整型参数b，字符型参数c；

③：空，参数缺省。

1. **＜主函数＞ ::= void main ‘(’ ‘)’ ‘{’＜复合语句＞‘}’**

作用：将主函数定义为void、main、()和{复合语句}，其中，void表明无返回值，main表明主函数。

限定条件：无返回值。参数缺省。

示例：void main() {printf(“Hello World!”);}

1. **＜表达式＞ ::= ［＋｜－］＜项＞{＜加法运算符＞＜项＞}**

作用：将表达式定义为符号（“+”或“-”）、项、若干个加法运算符和项的顺序组合的有序组合。

限定条件：符号允许缺省。加法运算符和项的顺序组合允许缺省。

示例：（设a、b、c为变量或常量）

①a：表达式a；

②+a：表达式+a，同a；

③-a+b-c：表达式-a+b-c。

1. **＜项＞ ::= ＜因子＞{＜乘法运算符＞＜因子＞}**

作用：将项定义为因子、若干乘法运算符和因子的顺序组合的有序组合。

限定条件：乘法运算符和因子的顺序组合允许缺省。

示例：（设a、b、c为变量或常量）

①a：项a；

②a\*b\*c：项a\*b\*c。

1. **＜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＞‘)’**

作用：将因子定义为标识符或标识符[表达式]或整数或字符或又返回值函数调用语句或(表达式)。其中，标识符[表达式]表明数组元素。

限定条件：无。

示例：

①a：因子a；

②a[1+1]：因子a[2]；

④666：因子666；

⑤’a’：因子’a’，计算时其值为对应ASCII编码；

⑥(a+b)：因子(a+b)。

1. **＜语句＞ ::= ＜条件语句＞|＜情况语句＞｜＜循环语句＞**

**｜‘{’＜语句列＞‘}’｜＜空＞;**

**｜＜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＞;**

**｜＜返回语句＞;**

作用：将语句定义为条件语句或循环语句或{语句列}或有返回值函数调用语句或无返回值函数调用语句或赋值语句或读语句或写语句或空或返回语句。

限定条件：语句允许缺省。

示例：

①：条件语句；

②while(i>0) i=i-1;：循环语句；

③：赋值语句；

④scanf(a);：读语句；

⑤printf(“Hello World!”)：写语句；

⑥：空语句；

⑦return(0);：返回语句；

⑧scanf(a);if(a==0) a=1; else a=0;a=1+1;；语句列

⑨switch(a){case 1:a=a+1;case 2:a=a+2;default:a=0;}：情况语句

1. **＜赋值语句＞ ::= ＜标识符＞＝＜表达式＞**

**|＜标识符＞‘[’＜表达式＞‘]’=＜表达式＞**

作用：将赋值语句定义为标识符=表达式或标识符[表达式]=表达式。其中，标识符[表达式]表示数组元素。赋值运算符=为双目运算符，将右操作数的值赋给左操作数。

限定条件：左右操作数类型一致。一条赋值语句中赋值运算符=仅出现一次，即不允许连续赋值（连等）。

示例：

①a=1+1：a被赋值为2；

②a[1+1]=’z’：字符数组元素a[2]被赋值为’z’。

1. **＜条件语句＞ ::= if ‘(’＜条件＞‘)’＜语句＞［else＜语句＞］**

作用：将条件语句定义为if(条件)、语句、else(语句)的顺序组合。当条件为逻辑“真”（值不为0）时执行else前的语句，否则执行else后的语句。

限定条件：else(语句)允许缺省，等同于else后语句为空。

示例：

①if(a==0) a=1;：当a的值为0时将a赋值为1；

②if(a==0) a=1; else a=0;：当a的值为0时将a赋值为1，否则将a赋值为0。

1. **＜条件＞ ::= ＜表达式＞＜关系运算符＞＜表达式＞**

**｜＜表达式＞**

作用：将条件定义为表达式、关系运算符、表达式的顺序组合或单独的表达式。其中，顺序组合值或表达式值为0表明条件为逻辑“假”，否则为逻辑“真”。

限定条件：无。

示例：（设a、b为变量或常量）

①a==b：若a与b的值相等则“真”，否则“假”；

②a：若a的值不为0则“真”，否则“假”。

**33. ＜循环语句＞ ::= while ‘(’＜条件＞‘)’＜语句＞**

作用：将循环语句定义为while语句，即while语句定义为while(条件)和语句的有序组合。

限定条件：无

示例：（设a和i为变量）

①while(i>0) i=i-1;：当i的值大于0时执行循环体（i自减一）；

②while(a)a=0;：当a的值不为0时执行循环体（将a赋值为0）。

**34. ＜情况语句＞ ::= switch ‘(’＜表达式＞‘)’ ‘{’＜情况表＞［＜缺省＞］‘}’**

作用：将情况语句定义为switch语句。其中包含情况表（若干case语句）和缺省（default）语句，缺省语句是可以省略的。当switch后的表达式值满足情况表中的某种情况时执行其后的语句（执行到下一条case语句时跳出），否则执行default（若有）后的语句。

限定条件：switch后表达式的值为常量。

示例：（设a为常量）

①switch(a){case 1:a=a-1;case 2:a=a-2;default:a=0;}：判断a的值，当a的值为1时将a的值减1，当a的值为2时将a的值减2，否则将a赋值为0；

②switch(a){case 1:a=a-1;case ‘c’:a=a-2;}：判断a的值，当a的值为1时将a的值减1，当a的值为字符’c’所对应的ASCII码时将a的值减2，否则不进行任何操作。

**35. ＜情况表＞ ::= ＜情况子语句＞{＜情况子语句＞}**

作用：将情况表定义为若干条情况子语句。

限定条件：情况子语句不少于一条，每条情况子语句代表不同情况（case后常量值不同）。

示例：（设a为变量）

①case 1:a=a-1;case 2:a=a-2;：当switch后的表达式的值为1时将a的值减1，当switch后的表达式的值为2时将a的值减2；

②case ‘c’:a=a-2;：当switch后的表达式的值为字符’c’所对应的ASCII码时将a的值减2。

1. **＜情况子语句＞ ::= case＜常量＞：＜语句＞**

作用：将情况子语句定义为一条case语句。其中，case后为一常量，即若符合该常量所代表的情况，则执行其后的语句。

限定条件：常量为整数或字符（不同情况子语句中的常量值不同）。若为字符，则在比较时自动转换为其对应的ASCII码值。

示例：（设a为变量）

①case 1:a=a-1;：当switch后的表达式的值为1时将a的值减1；

②case ‘c’:a=a-2;：当switch后的表达式的值为字符’c’所对应的ASCII码时将a的值减2。

1. **＜缺省＞ ::= default : ＜语句＞**

作用：将缺省定义为一条default语句。当case语句中没有满足表达式值的情况时执行default后的语句。

限定条件：default语句在switch情况语句中至多有一条，可省略。

示例：

①default:a=a-1;：当switch后的表达式的值不等于任何一条case语句中的常量值时将a的值减1；

**38. ＜有返回值函数调用语句＞ ::= ＜标识符＞‘(’＜值参数表＞‘)’**

作用：将有返回值函数调用语句定义为标识符和(值参数表)的有序组合。

限定条件：标识符为已定义的有返回值函数名。值参数表允许缺省。

示例：（设func1和func2为已定义的有返回值函数）

①func1();：调用无参函数func1；

②func2(int a);：调用有参（a）函数func2。

**39. ＜无返回值函数调用语句＞ ::= ＜标识符＞‘(’＜值参数表＞‘)’**

作用：将无返回值函数调用语句定义为标识符和(值参数表)的有序组合。

限定条件：标识符为已定义的无返回值函数名。值参数表允许缺省。

示例：（设func1和func2为已定义的有返回值函数）

①func1();：调用无参函数func1；

②func2(a);：调用有参（参数a）函数func2。

1. **＜值参数表＞ ::= ＜表达式＞{,＜表达式＞}**

**｜＜空＞**

作用：将值参数表定义为若干表达式的有序组合或空。

限定条件：值参数表允许缺省。若干表达式之间用逗号,分隔。

示例：（设a、b为变量或常量）

①：空，值参数缺省；

②a：值参数a；

③a, a+b：值参数a，值参数a+b。

**41. ＜语句列＞ ::= ｛＜语句＞｝**

作用：将语句列定义为若干条语句的集合。

限定条件：语句列允许为空。

示例：

①：语句列为空；

②scanf(a);if(a==0) a=1; else a=0;a=1+1;：语句列（含读语句、条件语句和赋值语句）。

**42. ＜读语句＞ ::= scanf ‘(’＜标识符＞{,＜标识符＞}‘)’**

作用：将读语句定义为scanf和(不少于一个标识符)的顺序组合。

限定条件：若干个标识符之间用逗号,分隔。

示例：

①scanf(a)：将读入的值赋给a；

②scanf(a,b)：将读入的两个值按次序分别赋给a和b。

1. **＜写语句＞ ::= printf ‘(’＜字符串＞,＜表达式＞‘)’**

**| printf ‘(’＜字符串＞‘)’**

**| printf ‘(’＜表达式＞‘)’**

作用：将写语句定义为printf和(字符串, 表达式)或(字符串)或(表达式)的有序组合。其中，(字符串)原样输出，(表达式)输出其值。

限定条件：(字符串, 表达式)中确定了输出顺序。

示例：（设a为整型变量或常量）

①printf(“The result is: ”, a)：打印字符串“The result is: ”并在其末尾输出a值；

②printf(“Hello World!”)：打印字符串“Hello World!”；

③printf(a)；打印a值。

**44. ＜返回语句＞ ::= return[‘(’＜表达式＞‘)’]**

作用：将返回语句定义为return和(表达式)的有序组合。

限定条件：(表达式)允许缺省。

示例：

①return：无返回值，函数结束标志；

②return(a)：将a作为返回值；

**附加说明：**

（1）char类型的表达式，用字符的ASCII码对应的整数参加运算，在写语句中输出字符；

（2）标识符不区分大小写字母；

（3）写语句中的字符串原样输出；

（4）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；

（5）数组的下标从0开始。