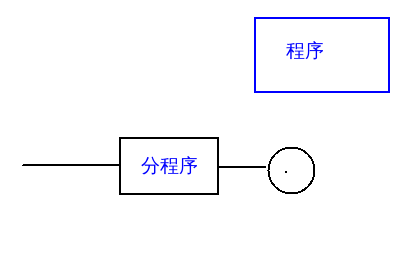
文法解读

11091222 余锋伟 110641班

1、<程序> ::= <分程序>.

语法图为：

说明程序主体部分是一个分程序，加上一个句号为结尾。

例：var a:integer begin end.

2、分程序

<分程序> ::= [<常量说明部分>][<变量说明部分>]{[<过程说明部分>]|[<函数说明部分>]}<复合语句>

<常量说明部分> ::= const<常量定义>{,<常量定义>};

<常量定义> ::= <标识符>＝<常量>

<常量> ::= [+|-]<无符号整数>|<字符>

<字符> ::= '<字母>'|'<数字>'

<无符号整数> ::= <数字>{<数字>}

<标识符> ::= <字母>{<字母>|<数字>}

<变量说明部分> ::= var<变量说明>;{<变量说明>;}

<变量说明> ::= <标识符>{,<标识符>}:<类型>

<类型> ::= <基本类型>|array'['<无符号整数>']'of<基本类型>

<基本类型> ::= integer|char

<过程说明部分> ::= <过程首部><分程序>{;<过程首部><分程序>};

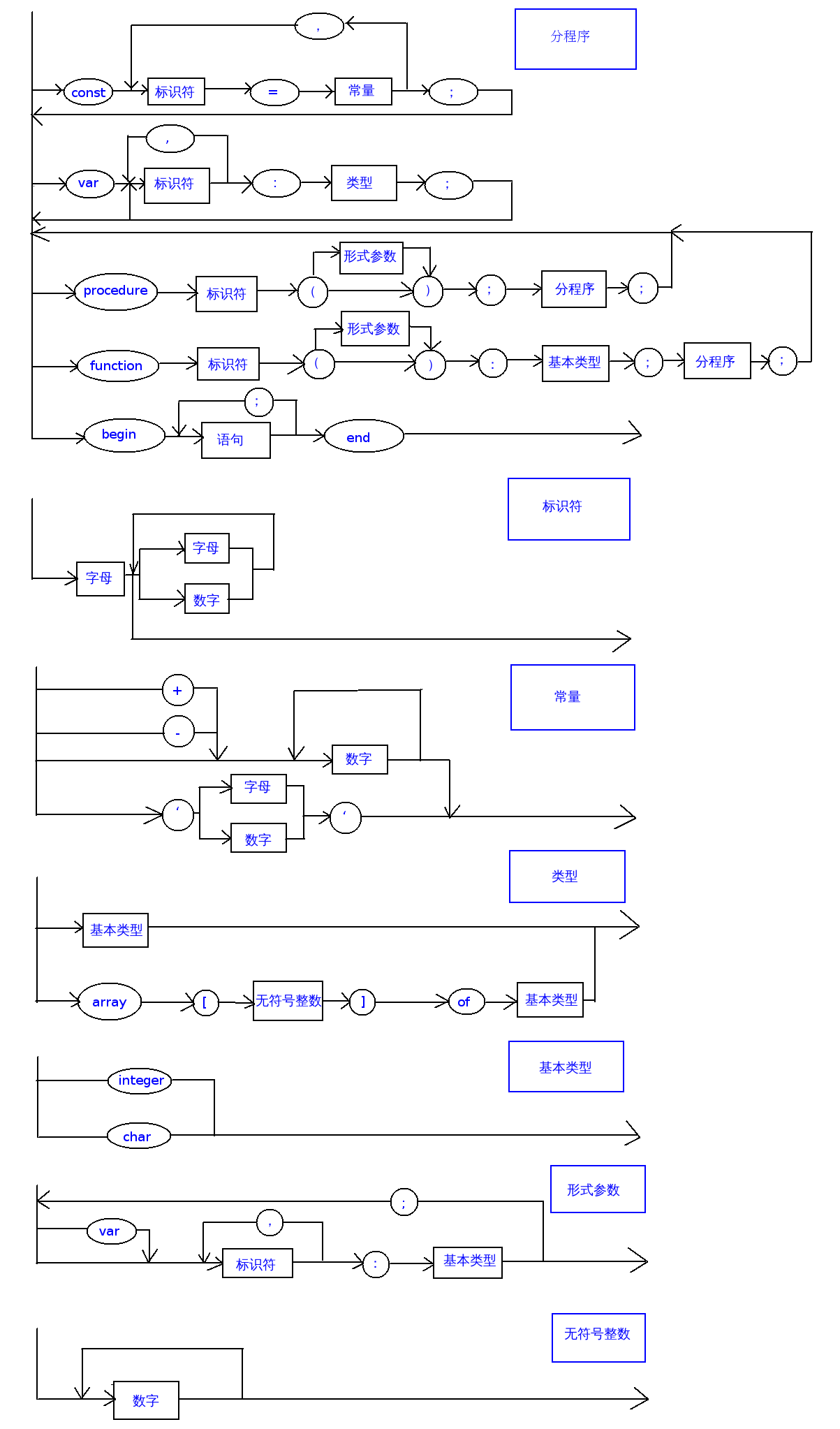
<函数说明部分> ::= <函数首部><分程序>{;<函数首部><分程序>};

<过程首部> ::= procedure<标识符>'('[<形式参数表>]')';

<函数首部> ::= function<标识符>'('[<形式参数表>]')':<基本类型>;

<形式参数表> ::= [var]<标识符>{,<标识符>}:<基本类型>{;<形式参数表>}

<复合语句> ::= begin<语句>{;<语句>}end



语法图如上。

根据分程序的定义，程序可以有常量声明和变量声明，但语法规定了，如果有变量声明必须先有常量声明，然后如果有变量声明再跟变量，比如说var i:integer;const a=10; 这样的顺序是不允许的。但函数和过程声明则可以用任意顺序并且可以任意声明任意多个。最后程序主体部分是一个begin end。

文法规定了程序返回值只能为整型或者字符型。常量可以为正负整数，字符，但不能有浮点型。

常量声明和变量和声明不允许分多个const和var。

另外数组不允许作为形式参数传入过程或者函数。参数允许带var，表示为引用调用。

示例：

const n = 10;

var sum:integer;

function getp(x:integer):integer;

略

procedure heihei(x:char);

略

begin

略

end.

3、语句

<语句> ::= <赋值语句>|<条件语句>|<repeat循环语句>|<过程调用语句>|<复合语句>|<读语句>|<写语句>|<for循环语句>|<空>

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

<函数标识符> ::= <标识符>

<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>']'|<无符号整数>|'('<表达式>')'|<函数调用语句>

<函数调用语句> ::= <标识符>'('[<实在参数表>]')'

<实在参数表> ::= <实在参数>{,<实在参数>}

<实在参数> ::= <表达式>

<加法运算符> ::= +|-

<乘法运算符> ::= \*|/

<条件> ::= <表达式><关系运算符><表达式>

<关系运算符> ::= <|<=|>|>=|=|<>

<条件语句> ::= if<条件>then<语句>|if<条件>then<语句>else<语句> ###???

<repeat循环语句> ::= repeat<语句>until<条件>

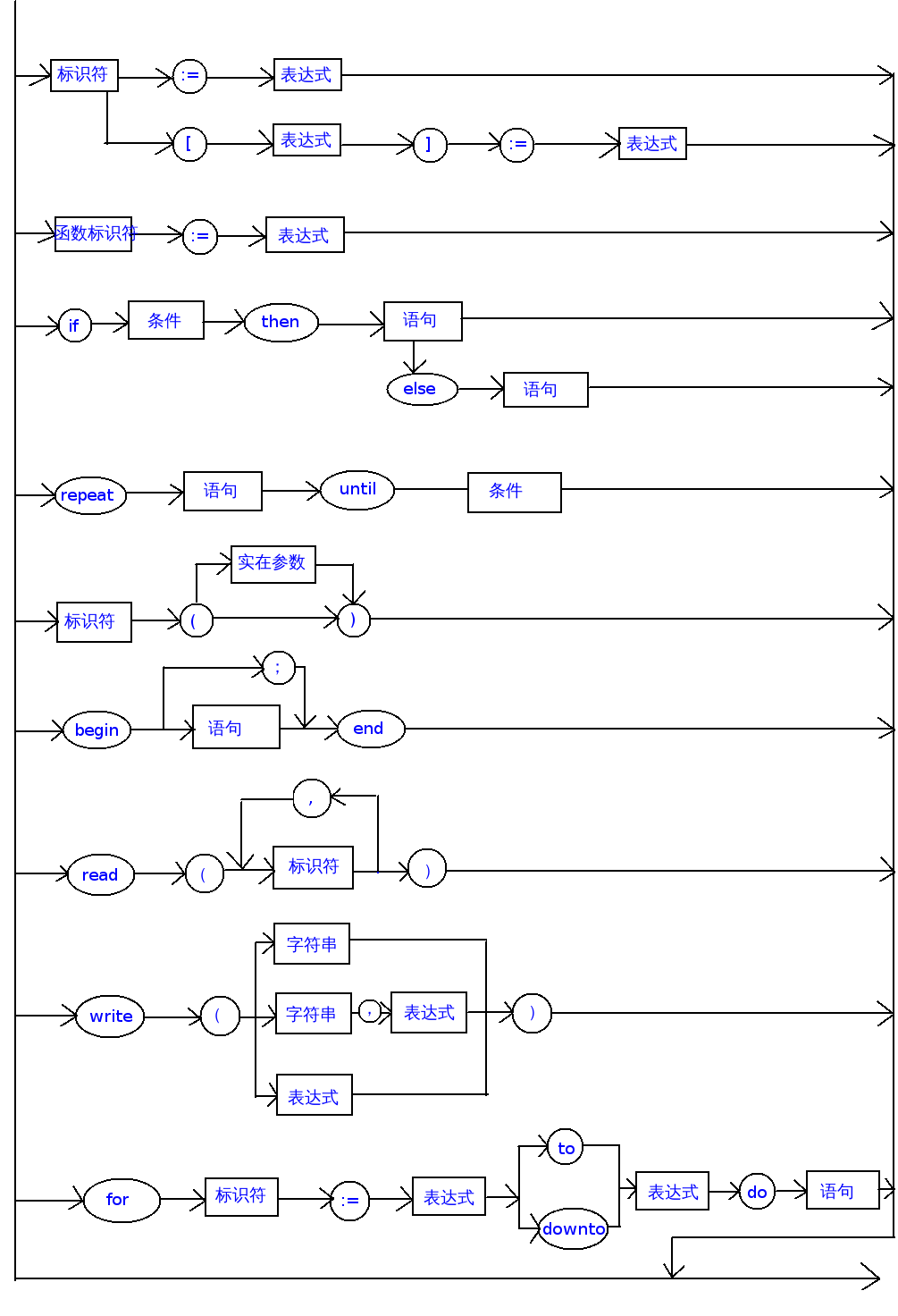
<for循环语句> ::= for<标识符>:=<表达式>(to|downto)<表达式>do<语句>

<过程调用语句> ::= <标识符>'('[<实在参数表>]')'

<读语句> ::= read'('<标识符>{,<标识符>}')'

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

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



语法图如上：

该文法支持几种基本语句和控制流：赋值、if条件语句、repeat until循环语句、过程调用read、write，for支持downto和to且默认为1步长。

赋值：赋值语句左段允许是变量，数组，和函数标识符。

if条件语句：支持两种条件跳转，if then和if then else。从文法可以看出，else采取就近匹配原则。

repeat until循环语句：这是一种常见的控制流方式，不断运行repeat和until中的语句知道until后的条件满足即退出循环。

过程调用：使用过程名(实在参数)进行调用。

read：读入语句允许读入变量，并且允许多个变量以逗号隔开读入。

write：输出语句略显奇怪，只允许三种形式的输出：字符串+表达式，字符串，表达式。值得注意的是这里不允许一次输出多个表达式和多个字符串。

for：for循环默认步长为1并且无法修改，支持to和downto。

示例：

var a,b:integer;r array[10]of integer;

begin

read(b);

a := b+1;

r[2] := a;

if r[2]>0 then

a = a +1

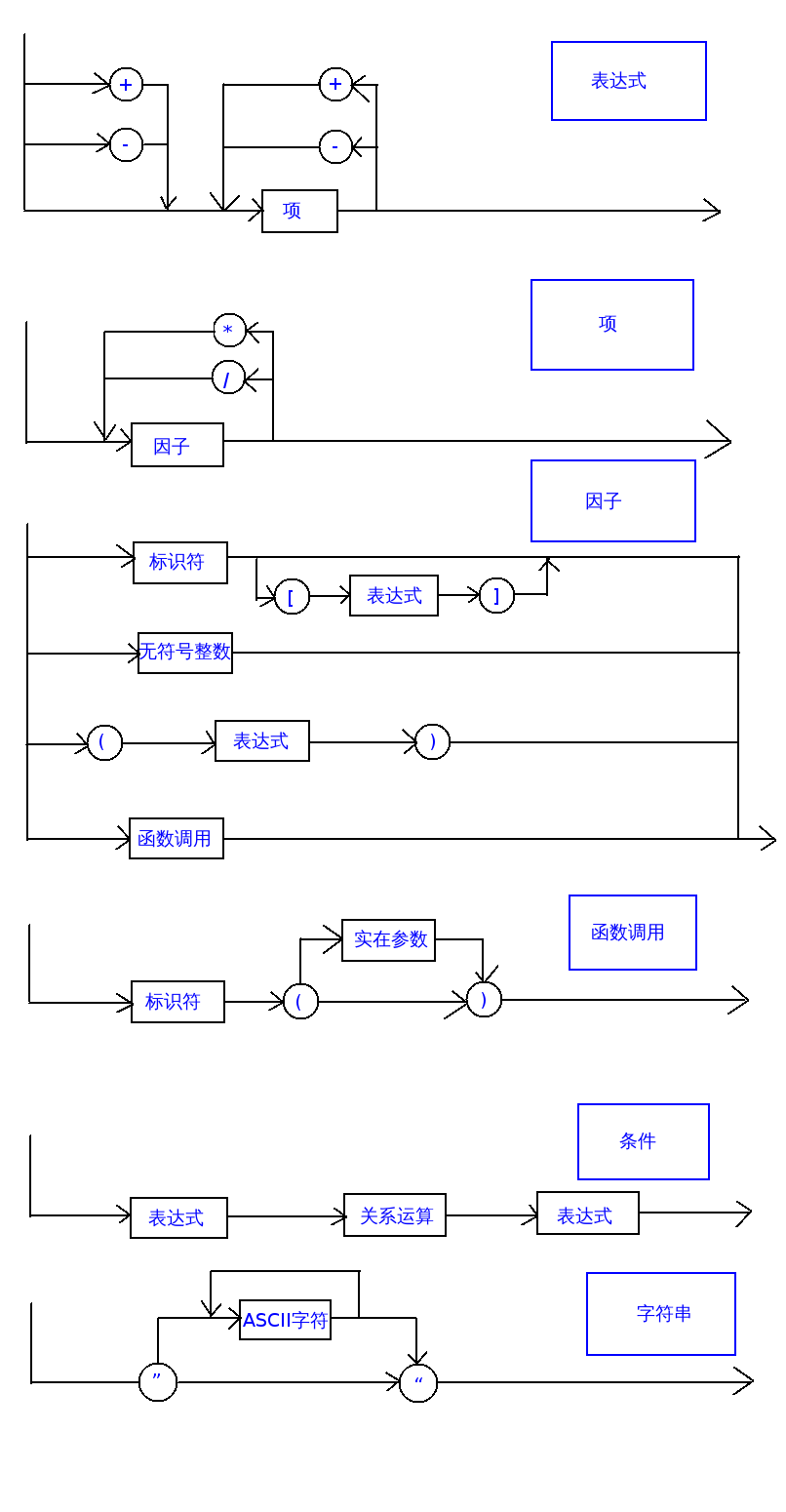
else

a = a - 1;

write(“the value a is ”,a);

end.

其他文法细则



上图为其他文法，分别为：表达式，因子，项，函数调用，条件，字符串。

表达式：表达式可以以正号或者负号开头，并不断读入正号负号，调用项。

项：项由因子\*/因子...组成，可以没有\*/号。

因子：因子可以由变量、常量、数组的项、无符号整数、函数调用，还可以嵌入(表达式)。不断递归调用。

函数调用：由函数名(实在参数)组成，注意，函数调用本身也是一个表达式。

条件：支持几种最基本的比较<|>|<=|>=|<>|=，这意味着条件不能含有逻辑运算符比如and or not。

字符串：字符串只允许十进制编码为32,33,35-126的ASCII字符，分别为：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 32 |  | 52 | 4 | 71 | G | 90 | Z | 109 | m |
| 33 | ! | 53 | 5 | 72 | H | 91 | [ | 110 | n |
| 35 | # | 54 | 6 | 73 | I | 92 | \ | 111 | o |
| 36 | $ | 55 | 7 | 74 | J | 93 | ] | 112 | p |
| 37 | % | 56 | 8 | 75 | K | 94 | ^ | 113 | q |
| 38 | & | 57 | 9 | 76 | L | 95 | \_ | 114 | r |
| 39 | ' | 58 | : | 77 | M | 96 | ` | 115 | s |
| 40 | ( | 59 | ; | 78 | N | 97 | a | 116 | t |
| 41 | ) | 60 | < | 79 | O | 98 | b | 117 | u |
| 42 | \* | 61 | = | 80 | P | 99 | c | 118 | v |
| 43 | + | 62 | > | 81 | Q | 100 | d | 119 | w |
| 44 | , | 63 | ? | 82 | R | 101 | e | 120 | x |
| 45 | - | 64 | @ | 83 | S | 102 | f | 121 | y |
| 46 | . | 65 | A | 84 | T | 103 | g | 122 | z |
| 47 | / | 66 | B | 85 | U | 104 | h | 123 | { |
| 48 | 0 | 67 | C | 86 | V | 105 | i | 124 | | |
| 49 | 1 | 68 | D | 87 | W | 106 | j | 125 | } |
| 50 | 2 | 69 | E | 88 | X | 107 | k | 126 | ~ |
| 51 | 3 | 70 | F | 89 | Y | 108 | l |  |  |