**PL/0文法解读作业**

周美廷-76066002

首先PL/0文法中，{} 代表多于等于0， 而 [] 代表可有可无。PL/0程序是由分程序（block）组成的。

**<程序> ::= <分程序>.**

**Program** = **block** "." .

然后我们就从分程序说起。分程序也就是我们平常看到的程序代码。大体来看，分程序具有这些结构:

1. 常量说明部分（可有可无）
2. 变量说明部分（可有可无）
3. 过程说明**亦或者**函数说明部分（可有可无）
4. 复合语句（必须）

**<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>**

**block** = [ "const" **ident** "=" **number** {"," **ident** "=" **number**} ";"]

[ "var" **ident** {"," **ident**} ";"]

{ "procedure" **ident** ";" **block** ";" } **statement** .

一个程序中会常常用到的有：

1. 字母

**<字母> ::= a|b|c|d...|x|y|z |A|B...|Z**

1. 数字

**<数字> ::= 0|1|2|3...|8|9**

1. **数据类型**

PL/0中有这几种数据类型：

1. 常量（const）
2. 变量（var）
3. 过程（procedure）
4. 函数（function）

每个数据类型定义都有标识符，标识符必须是小于等于11个字符组成，必须以字符开头，可以有大写或小写字母也可以有数字。

**<标识符> ::= <字母>{<字母>|<数字>}**

数据类型的基本类型一般是整数，实数，字符/字符串

**<基本类型> ::= integer | real | char**

* 1. **常量说明部分**

条件/规定：

1. 它们在每一个程序中只能被定义一次
2. 常量是不可变的，也就是说，定义后不能再赋值
3. 可以同时定义多于一个常量，以逗号分
4. 前面写const跟着有标识符，等号，和常量值，定义后面必须有分号

**<常量说明部分> ::= const<常量定义>{,<常量定义>};**

**<常量定义> ::= <标识符>＝ <常量>**

1. 常量值可以是整数或实数，前面可以有符号（+/-）也可以没有，常量也可以是字符。

**<常量> ::= [+| -] (<无符号整数>| <无符号实数>)|<字符>**

**<无符号整数> ::= <数字>{<数字>}**

**<无符号实数> ::= <无符号整数>.<无符号整数>**

字符以 ’ 开头，跟着可以是字母或者数字，然后后面要有 ’

**<字符> ::= '<字母>' | '<数字>'**

字符串以 ” 开头，跟着可以十进制编码中为32，33，35-126中的任何ASCII字符组成。

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

例子：

const FOO = 1, BAR = 2; {\*多常量定义\*}

const

Name = “Tao Yue”; {\*字符串类常量定义\*}

FirstLetter = 'a'; {\*字符类常量定义\*}

Year = 1997; {\*数字类常量定义\*}

pi = 3.1415926535897932; {\*浮点数类常量定义\*}

UsingNCSAMosaic = TRUE; {\*布尔类常量定义\*}

* 1. **变量说明部分**

1. 变量说明以var开头，跟着是标识符，冒号和变量的类型

**<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>**

1. 变量类型是基本类型中的任何一个也可以是基本类型的数组（array），语法为array'['<无符号整数>']' of <基本类型>（<无符号整数>表示数组元素个数，其值需大于0）

**<类型> ::= <基本类型>|array'['<无符号整数>']' of <基本类型> //此处<无符号整数>表示数组元素个数，其值需大于0**

例子：

var ch : char;

num: integer;

line: array[1..81] of char;

sfile: string;

* 1. **过程说明和函数说明部分**

过程和函数也可以了解为程序中的子程序。他们可以容纳一个完整程序的大部分内容。

**<过程说明部分> ::= <过程首部><分程序>;**

**<函数说明部分> ::= <函数首部><分程序>;**

过程和函数说明，根据情况可以有形式参数也可能没有

**<形式参数表> ::= <形式参数>{; <形式参数>}**

形式参数的语法为以下表示

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

1. 过程的语法为：

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

例子：

procedure getsym; {\*无参数过程说明\*}

procedure gen( x: fct; y,z : integer ); {\*有参数过程说明\*}

1. 函数的语法为：

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

例子：

function write ( id : string ): integer; {\*有参数函数说明\*}

1. **表达式**
   1. **语句**

语句可分为几种语句，分别是赋值语句，条件语句，repeat循环语句，for循环语句，过程调用语句，复合语句，读语句，写语句。

**<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>**

* + 1. **赋值语句**

正如它的名字，赋值语句的作用是用来给变量赋值的（表达式）。

**<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>']'|<无符号整数>| <无符号实数>|'('<表达式>')' | <函数调用语句>**

赋值过程一般会用以下其中一个表达式来进行赋值：

1. 简单赋值（:=）

**<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>**

例子：

t := 0; {\*t=0\*}

1. 运算赋值（加减乘除）
2. 加法运算（+=）

**<加法运算符> ::= +**

1. 减法运算（-=）

**<减法运算符> ::= -**

1. 乘法运算（\*=）

**<乘法运算符> ::= \***

1. 除法运算（/=）

**<除法运算符> ::= /**

例子：

{\*t=3\*}

t+=1; {\*t=4\*}

t-=1; {\*t=2\*}

t\*=2; {\*t=6\*}

t/=3; {\*t=1\*}

* + 1. **有条件语句**

**<条件> ::= <表达式><关系运算符><表达式>**

* + - 1. **条件语句**

条件语句允许我们执行某个（些）代码如果符合我们给的条件。若不符合，那就执行其他代码段。条件语句中一般会用关系运算符分别为：

1. <> （不等于）
2. = （等于）
3. >= （大于等于）
4. > （大于）
5. <= （小于等于）
6. < （小于）

**<关系运算符> ::= <|<=|>|>= |=|<>**

条件语句的语法为：

**<条件语句> ::= if<条件>then<语句> | if<条件>then<语句>else<语句>**

例子：

{\*if then例子\*}

if cx > cxmax

then begin

writeln('program too long');

end

{\*if then else例子\*}

if sym = ident

then begin

getsym;

end

else error(4)

* + - 1. **Repeat循环语句**

Repeat循环语句是用来执行某段代码直到符合某个条件。语法为：

**<repeat循环语句> ::= repeat <语句>until<条件>**

例子：

repeat

WriteLn (’I =’,i);

I := I+2;

until I>100;

* + - 1. **For循环语句**

For循环语句是用来循环执行某些代码而执行次数是已定的。语法为：

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

1. To

如果初始值大于最终值，那么语句不会被执行。

例子：

For Day := Monday to Friday do Work;

1. Downto

如果初始值小于最终值，那么语句不会被执行。

例子：

For I := 100 downto 1 do

WriteLn (’Counting down : ’,i);

* + 1. **过程/函数调用语句**

过程/函数调用语句是用来调用某个过程/函数的语句。调用是根据情况可能有参数或没有。语法为：

**<过程调用语句> ::= <标识符>'('[<实在参数表>]')'**

**<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>**

例子：

WriteLn(’Pascal is an easy language !’); {\*有参数例子\*}

Doit(); {\*无参数例子\*}

Usage;

* + 1. **复合语句**

符合语句平常会在程序中见到。当程序员要为一个条件，一个函数或一个过程写很多语句时，会先以begin开头后以end关闭。语法为：

**<复合语句> ::= begin<语句>{; <语句>}end**

例子：

begin

read(fin,ch);

write(ch);

end;

* + 1. **读语句**

读语句是用来从某个设备（键盘，文件，scanner等）中读取它的内容。语法为：

**<读语句> ::= read'('<标识符>{,<标识符>}')'**

例子：

read(fin,ch);

* + 1. **写语句**

写语句是用来把某些内容写到某个设备（显示器，文件等）。语法为：

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

例子：

write('Enter the radius of the circle');