programstruct -> program\_head **;** program\_body**.**

program\_head -> **program** **id(** idlist **)** | **program** **id**

program\_body -> const\_declarations

var\_declarations

subprogram\_declarations

compound\_statement

idlist -> idlist **, id** | **id**

const\_declarations -> **const** const\_declaration;

const\_declaration -> const\_declaration **; id =** const\_value | **id =** const\_value

const\_value -> **+**num | **-**num | num | **‘letter’**

num -> **digseq** | **realnumber**

var\_declarations -> **var** var\_declaration **;**

var\_declaration -> var\_declaration **;** idlist **:** type | idlist **:** type

type -> basic\_type | **array [** period **]** **of** basic\_type

basic\_type -> **integer** | **real** | **boolean** | **char**

period -> period **, digseq .. digseq** | **digseq .. digseq**

subprogram\_declarations -> subprogram\_declarations subprogram **;**

subprogram -> subprogram\_head **;** subprogram\_body

subprogram\_head -> **procedure id** formal\_parameter | **function id** formal\_parameter : basic\_type

formal\_parameter -> **(** parameter\_list **)**

parameter\_list -> parameter\_list **;** parameter | parameter

parameter -> var\_parameter | value\_parameter

var\_parameter -> **var** value\_parameter

value\_parameter -> idlist **:** basic\_type

subprogram\_body -> const\_declarations

var\_declarations

compound\_statement

compound\_statement -> **begin** statement\_list **end**

statement\_list -> statement\_list **;** statement | statement

statement -> variable **assignop** expression

|procedure\_call

|compound\_statement

|**if** expression **then** statement else\_part

|**for id assignop** expression **to** expression **do** statement

|**read (** variable\_list **)**

|**write (** expression\_list **)**

|**while** expression **do** statement

variable\_list -> variable\_list **,** variable | variable

variable -> **id** id\_varpart

id\_varpart -> **[** expression\_list **]**

procedure\_call -> **id** | **id** **(** expression\_list **)**

else\_part -> **else** statement

expression\_list -> expression\_list **,** expression | expression

expression -> simple\_expression relop simple\_expression | simple\_expression

simple\_expression -> simple\_expression addop term | term

term -> term mulop factor | factor

factor -> num

| variable

| **id (** expression\_list **)**

| **(** expression **)**

| **not** factor

| **-** factor

relop -> **>** | **=** | **<** | **<=** | **>=** | **<>**

addop -> **+** | **-** | **or**

mulop -> **\*** | **/** | **mod** | **and** | **div**

**关键字：**

**program**

**const**

**var**

**array**

**of**

**integer**

**real**

**boolean**

**char**

**preocedure**

**function**

**begin**

**end**

**if**

**then**

**for**

**to**

**do**

**read**

**write**

**while**

**else**

**叶子节点：**

**id：关键字（宏定义数字）**

**attr：关键字名字**

**标识符：**

**IDENTIFIRE**

**叶子节点：**

**id：IDENTIFIRE（宏定义数字）**

**attr：id名字**

**常数：**

**letter**

**digseq**

**realnumber**

**叶子结点：**

**id：letter，digseq，realnumber（宏定义数字）**

**attr：a-z ，123 ，1.123**

**运算符：**

**+**

**-**

**\***

**/(除法)**

**mod**

**assignop(:=)**

**not**

**and**

**div(或)**

**=**

**>**

**<**

**<=**

**>=**

**<>**

**叶子结点：**

**id：运算符（宏定义数字）**

**attr：对应符号（string类型）**

**分界符：**

**(**

**)**

**[**

**]**

**,**

**;**

**‘**

**:**

**..**

**叶子结点：**

**id：分界符符（宏定义数字）**

**attr：对应符号（string类型）**