**SAMPLE语言LL（1）文法定义**

1. **字符集定义**
   1. <字符集> → <字母>│<数字>│<单界符>
   2. <字母> → A│B│…│Z│a│b│…│z
   3. <数字> → 0│1│2│…│9
   4. <单界符> → +│-│\*│/│=│<│>│(│)│[│]│:│. │; │, │'
2. **单词集定义**

5．<单词集> → <保留字>│<双界符>│<标识符>│<常数>│<单界符>

6．<保留字> → **and**│**array**│**begin**│**bool**│**call**│**case**│**char**│**constant**│**dim**│**do**│**else**│**end**│**false**│**for**│**if**│**input**│**integer**│**not**│**of**│**or**│**output**│**procedure**│**program**│**read**│**real**│**repeat**│**set**│**stop**│**then**│**to**│**true**│**until**│**var**│**while**│**write**

7．<双界符> → <>│<=│>=│:= │/\*│\*/│..

8．<标识符> → <字母>│<标识符> <数字>│<标识符> <字母>

9．<常数> → <整数>│<布尔常数>│<字符常数>

10．<整数> → <数字>│<整数> <数字>

11．<布尔常数> → **true**│**false**

12．<字符常数> → ' 除 {'} 外的任意字符串 '

1. **数据类型定义**

13．<类型> → **integer**│**bool**│**char**

1. **表达式定义**

14．<表达式> → <算术表达式>│<布尔表达式>│<字符表达式>！！！这个太复杂

15．<算术表达式> → <算术表达式> + <项>│<算术表达式> - <项>│<项>

15． <算术表达式> → <项> arithmetic\_sentence\_1

arithmetic\_sentence\_1 → + <项>arithmetic\_se^\_1│- <项>arithmetic\_se^\_1| ε

16．<项> → <项> \* <因子>│<项> / <因子>│<因子>

16． <项> → <因子> xiang\_1

xiang\_1 → \* <因子>xiang\_1│/ <因子> xiang\_1| ε

17．<因子> → <算术量>│- <因子>

18．<算术量> → <整数>│<标识符>│（ <算术表达式> ）

19．<布尔表达式> → <布尔表达式> **or** <布尔项>│<布尔项>

19．<布尔表达式> → <布尔项> bool\_sentence\_1

bool\_sentence\_1 → **or** <布尔项> bool\_sentence\_1 | ε

20．<布尔项> → <布尔项> **and** <布因子>│<布因子>

20．<布尔项> → <布因子> bool\_xiang\_1

bool\_xiang\_1 → **and** <布因子> bool\_xiang\_1 | ε

21．<布因子> → <布尔量>│**not** <布因子>

22．<布尔量> → <布尔常量>│<标识符>│（ <布尔表达式> ）│

<标识符> <关系符> <标识符>│<算术表达式> <关系符> <算术表达式>

<布尔量> → <布尔常量>│( <布尔表达式> )│<算术表达式> <关系符> <算术表达式> | 标识符

23．<关系符> → <│<>│<=│>=│>│=

24．<字符表达式> → <字符常数>│<标识符>

1. **语句定义**

25．<语句> → <赋值句>│<**if**句>│<**while**句>│<**repeat**句>│<复合句>

26．<赋值句> → <标识符> := <算术表达式>

27．<**if**句>→ **if** <布尔表达式> **then** <语句>│**if** <布尔表达式> **then** <语句> **else** <语句>

27．<**if**句>→**if** <布尔表达式> **then** <语句> **if\_sentence\_1**（消除左公因式）

if\_sentence\_1 **->**( else<语句> | ε)

28．<**while**句> → **while** <布尔表达式> **do** <语句>

29．<**repeat**句> → **repeat** <语句> **until** <布尔表达式>

30．<复合句> → **begin** <语句表> **end**

31．<语句表> → <语句> ；<语句表>│<语句>

31．<语句表> → <语句> statement\_list\_1（消除左公因式）

**statement\_list\_1 ->**；<语句表>│ε

1. **程序定义**

32．<程序> → **program** <标识符> ；<变量说明> <复合语句> .

33．<变量说明> → **var** <变量定义>│ε

34．<变量定义> → <标识符表> ：<类型> ；<变量定义>│<标识符表> ：<类型> ；

34．<变量定义> → <标识符表> ：<类型> ；var\_declare\_4（消除左公因式）

var\_declare\_4 → <变量定义>│ε

35．<标识符表> → <标识符> ，<标识符表>│<标识符>

35．<标识符表> → <标识符> var\_declare\_3（消除左公因式）

var\_declare\_3 →，<标识符表>│ε

1. **SIMPLE语言单词编码**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 单 词 | 种别码 |  | 单 词 | 种别码 |  | 单 词 | 种别码 |
| **and** | 1 |  | **output** | 21 |  | \* | 41 |
| **array** | 2 |  | **procedure** | 22 |  | \*/ | 42 |
| **begin** | 3 |  | **program** | 23 |  | + | 43 |
| **bool** | 4 |  | **read** | 24 |  | , | 44 |
| **call** | 5 |  | **real** | 25 |  | - | 45 |
| **case** | 6 |  | **repeat** | 26 |  | . | 46 |
| **char** | 7 |  | **set** | 27 |  | .. | 47 |
| **constant** | 8 |  | **stop** | 28 |  | / | 48 |
| **dim** | 9 |  | **then** | 29 |  | /\* | 49 |
| **do** | 10 |  | **to** | 30 |  | : | 50 |
| **else** | 11 |  | **true** | 31 |  | := | 51 |
| **end** | 12 |  | **until** | 32 |  | ; | 52 |
| **false** | 13 |  | **var** | 33 |  | < | 53 |
| **for** | 14 |  | **while** | 34 |  | <= | 54 |
| **if** | 15 |  | **write** | 35 |  | <> | 55 |
| **input** | 16 |  | 标识符 | 36 |  | = | 56 |
| **integer** | 17 |  | 整数 | 37 |  | > | 57 |
| **not** | 18 |  | 字符常数 | 38 |  | >= | 58 |
| **of** | 19 |  | ( | 39 |  | [ | 59 |
| **or** | 20 |  | ) | 40 |  | ] | 60 |

1. 实验一：设计SAMPLE语言的词法分析器

检查要求：

* 1. 启动程序后，先输出作者姓名、班级、学号（可用汉语、英语或拼音）；
  2. 请求输入测试程序名，键入程序名后自动开始词法分析并输出结果；
  3. 输出结果为单词的二元式序列（样式见样板输出1和2）；
  4. 要求能发现下列词法错误和指出错误性质和位置：

非法字符，即不是SAMPLE字符集的符号；

字符常数缺右边的单引号（字符常数要求左、右边用单引号界定，不能跨行）；

注释部分缺右边的界符\*/（注释要求左右边分别用/\*和\*/界定，不能跨行）。

1. 实验二：设计SAMPLE语言的语法、语义分析器，输出四元式的中间结果。

检查要求：

* 1. 启动程序后，先输出作者姓名、班级、学号（可用汉语、英语或拼音）。
  2. 请求输入测试程序名，键入程序名后自动开始编译。
  3. 输出四元式中间代码（样式见样板输出3和4）。
  4. 能发现程序的语法错误并输出出错信息。

1. 测试样板程序与样板输出

测试程序3：程序名TEST4 测试程序4：程序名TEST5

program example4; program example5;

var A,B,C,D:integer; var A,B,C,D,W:integer;

begin begin

A:=1; B:=5; C:=3; D:=4; A:=5; B:=4; C:=3; D:=2; W:=1;

while A<C and B>D do if W>=1 then A:=B\*C+B/D

if A=1 then C:=C+1 else else repeat A:=A+1 until A<0

while A<=D do A:=A\*2 end.

end.

样板输出3：（要求在屏幕上显示） 样板输出4：（要求在屏幕上显示）

( 0) (program,example4,-,-) ( 0) (program,example5,-,-)

( 1) (:= , 1 , - , A) ( 1) (:= , 5 , - , A)

( 2) (:= , 5 , - , B) ( 2) (:= , 4 , - , B)

( 3) (:= , 3 , - , C) ( 3) (:= , 3 , - , C)

( 4) (:= , 4 , - , D) ( 4) (:= , 2 , - , D)

( 5) (j< , A , C , 7) ( 5) (:= , 1 , - , W)

( 6) (j , - , - , 20) ( 6) (j>=, W , 1 , 8)

( 7) (j> , B , D , 9) ( 7) (j , - , - , 13)

( 8) (j , - , - , 20) ( 8) (\* , B , C , T1)

( 9) (j= , A , 1 , 11) ( 9) (/ , B , D , T2)

1. (j , - , - , 14) (10) (+ , T1 , T2 , T3)
2. (+ , C , 1 , T1) (11) (:= , T3 , - , A)
3. (:= , T1 , - , C) (12) (j , - , - , 17)
4. (j , - , - , 5) (13) (- , A , 1 , T4)
5. (j<=, A , D , 16) (14) (:= , T4 , - , A)
6. (j , - , - , 5) (15) (j< , A , 0 , 17)
7. (\* , A , 2 , T2) (16) (j , - , - , 13)
8. (:= , T2 , - , A) (17) (sys , - , - , -)
9. (j , - , - , 14)
10. (j , - , - , 5)
11. (sys , - , - , -)