

# 编译原理与技课 程设计术

Email: [wenshli@bupt.edu.cn](mailto:wenshli@bupt.edu.cn)  
TEL: 62282929

# 课程设计的目的

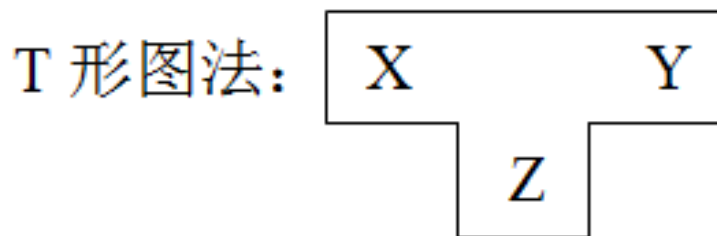
---

- 进一步掌握编译原理
  - 词法分析、语法分析、语义分析
  - （中间）代码生成
- 通过课程设计，掌握编译技术
  - 有限自动机技术
  - 语法制导翻译技术
- 提高解决复杂工程问题的能力
  - 分析问题、分解
  - 解决方案、方案实施

# 内容

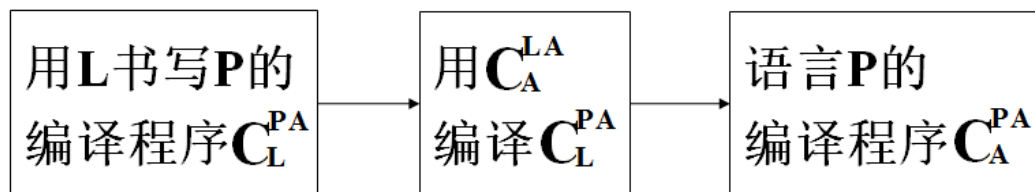
1. 编译程序的表示及实现
2. 课程设计任务与要求
  - 2.1 Pascal-S语言语法图
  - 2.2 Pascal-S语言文法产生式
  - 2.3 文法说明
  - 2.4 关于词法的约定
  - 2.5 课程设计建议
  - 2.6 测试
  - 2.7 开发方法
  - 2.8 设计报告要求
3. 教学安排

# 1. 编译程序的表示及实现

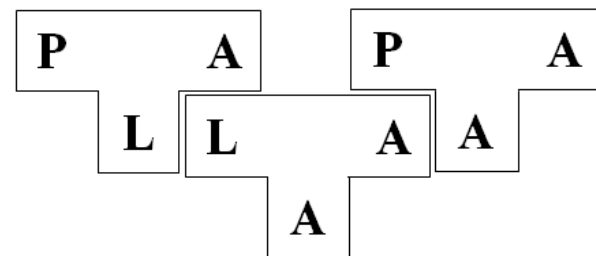


记号法:  $C_Z^{XY}$

- 若在A机型上已有语言L的编译程序，用记号  $C_A^{LA}$  表示，由于L是自编译语言，所以可以用语言L来书写语言P的编译程序  $C_L^{PA}$ ，再用  $C_A^{LA}$  编译  $C_L^{PA}$ ，可以得到能够在A机上运行的语言P的编译程序  $C_A^{PA}$ 。



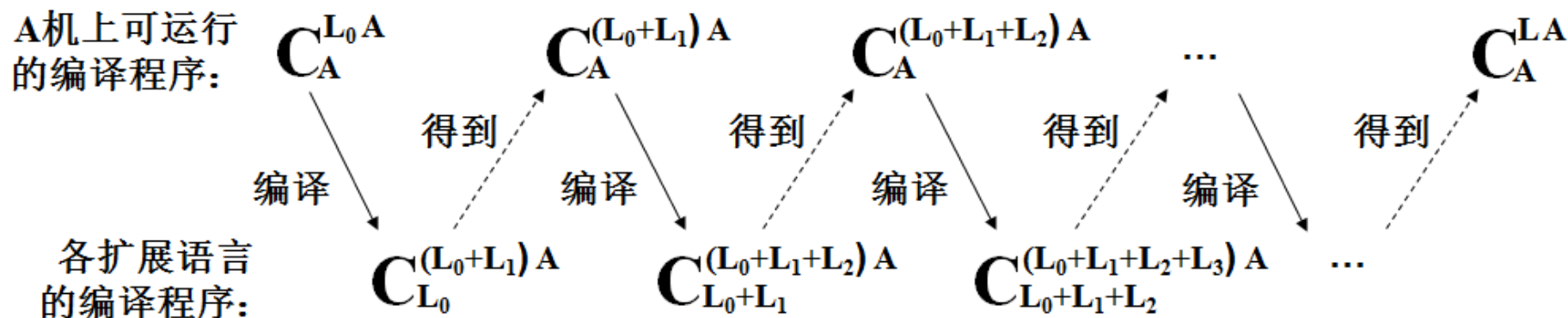
(a) 记号描述



(b) T形图表示

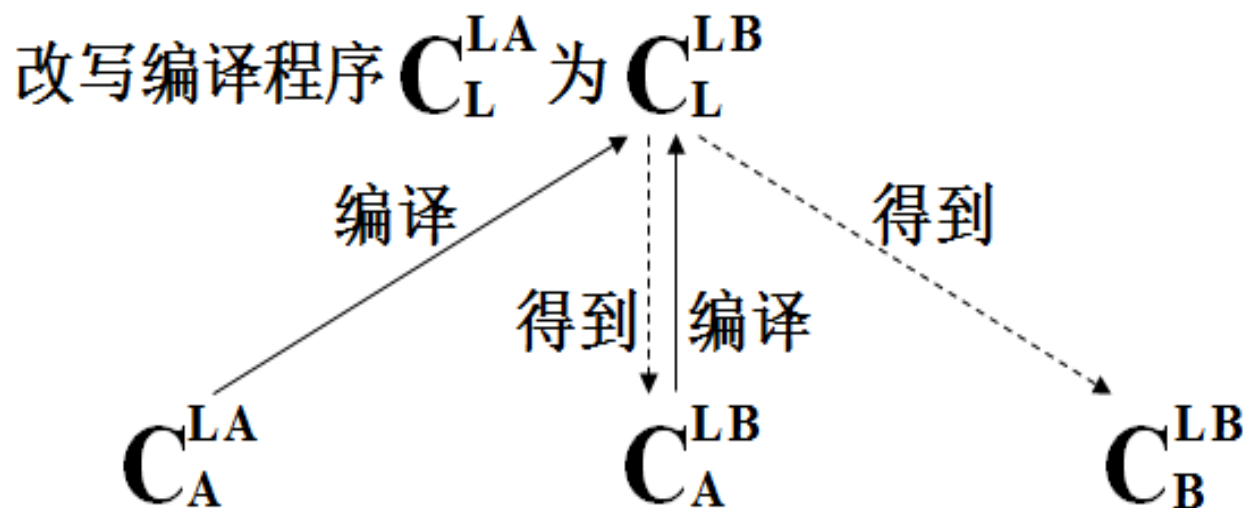
# 实现方法—自展法

- 有自编译语言L，通过自展的方法构造其在机器A上的编译程序  $C_A^{LA}$ ：



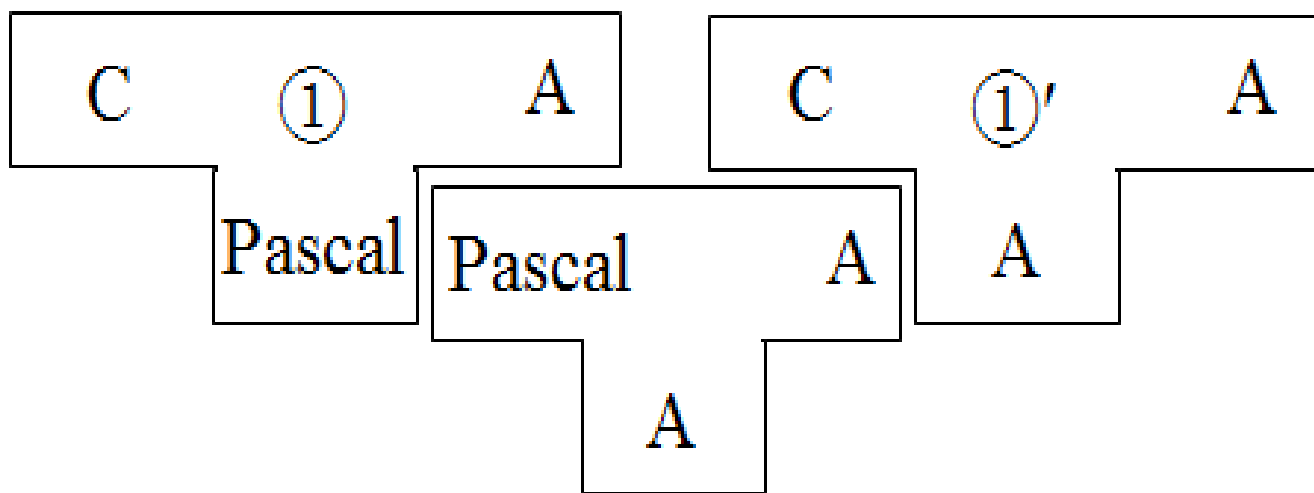
# 实现方法—移植法

- 若宿主机A上有用自编译高级语言L书写的语言L的编译程序  $C_L^{LA}$ ，以及可在A机上运行的语言L的编译程序  $C_A^{LA}$ ，那么，就可以将语言L的编译程序从A机移植到B机，即得到在B机上运行的语言L的编译程序  $C_B^{LB}$ 。



# 实现方法一—移植法

- 在A种机上已经有Pascal语言可用，如何使一种新的语言（如C语言）也可以在A机上使用？



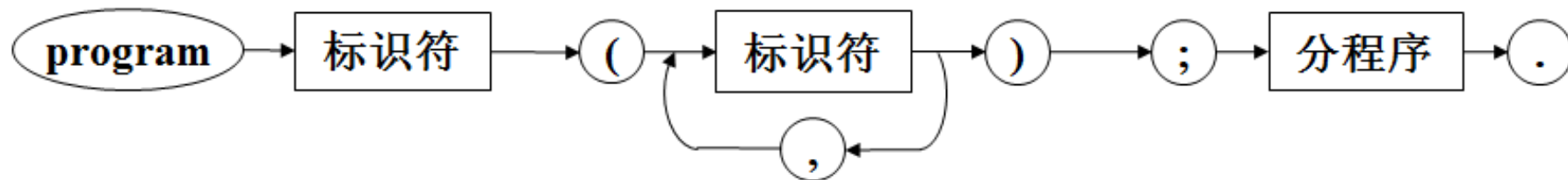
## 2. 课程设计任务说明

- 题目：Pascal-S语言编译程序的设计与实现
- 目标：  
按照所给Pascal-S语言的语法，参考Pascal语言的语义，设计并实现Pascal-S语言的编译程序。
- 要求给出各阶段的设计成果，如：
  - 需求分析报告
  - 总体设计报告（软件功能描述、功能模块划分、软件结构图、符号表结构设计、模块间接口定义等）
  - 详细设计报告（模块功能、输入/输出、处理逻辑等）
  - 编码实现（源程序、可执行程序）
  - 测试报告（测试计划、测试用例、测试结果及分析等）

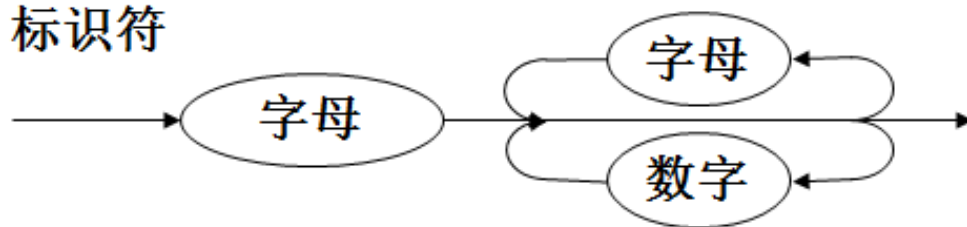


## 2.1 Pascal-S语言语法图 (1/7)

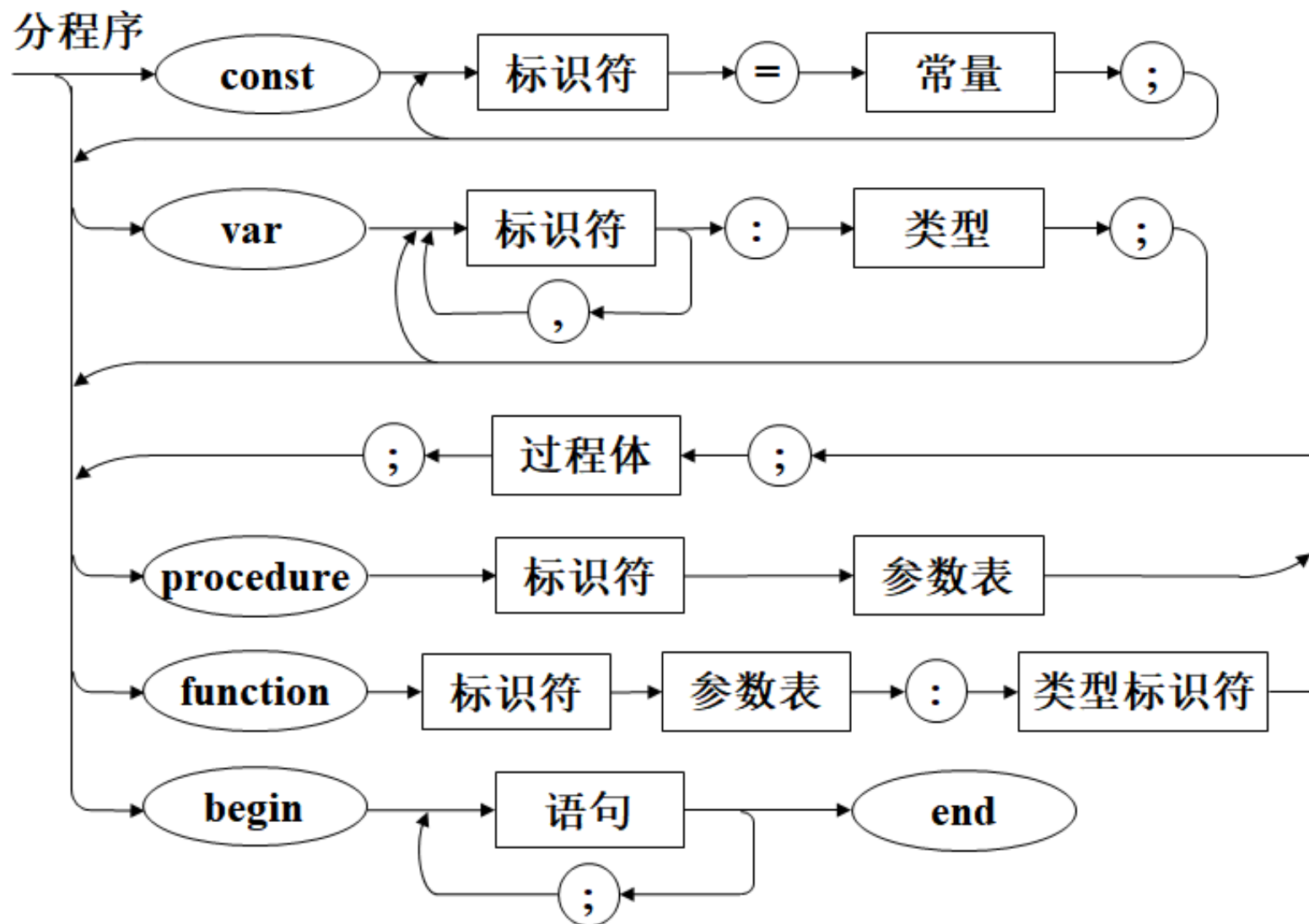
程序



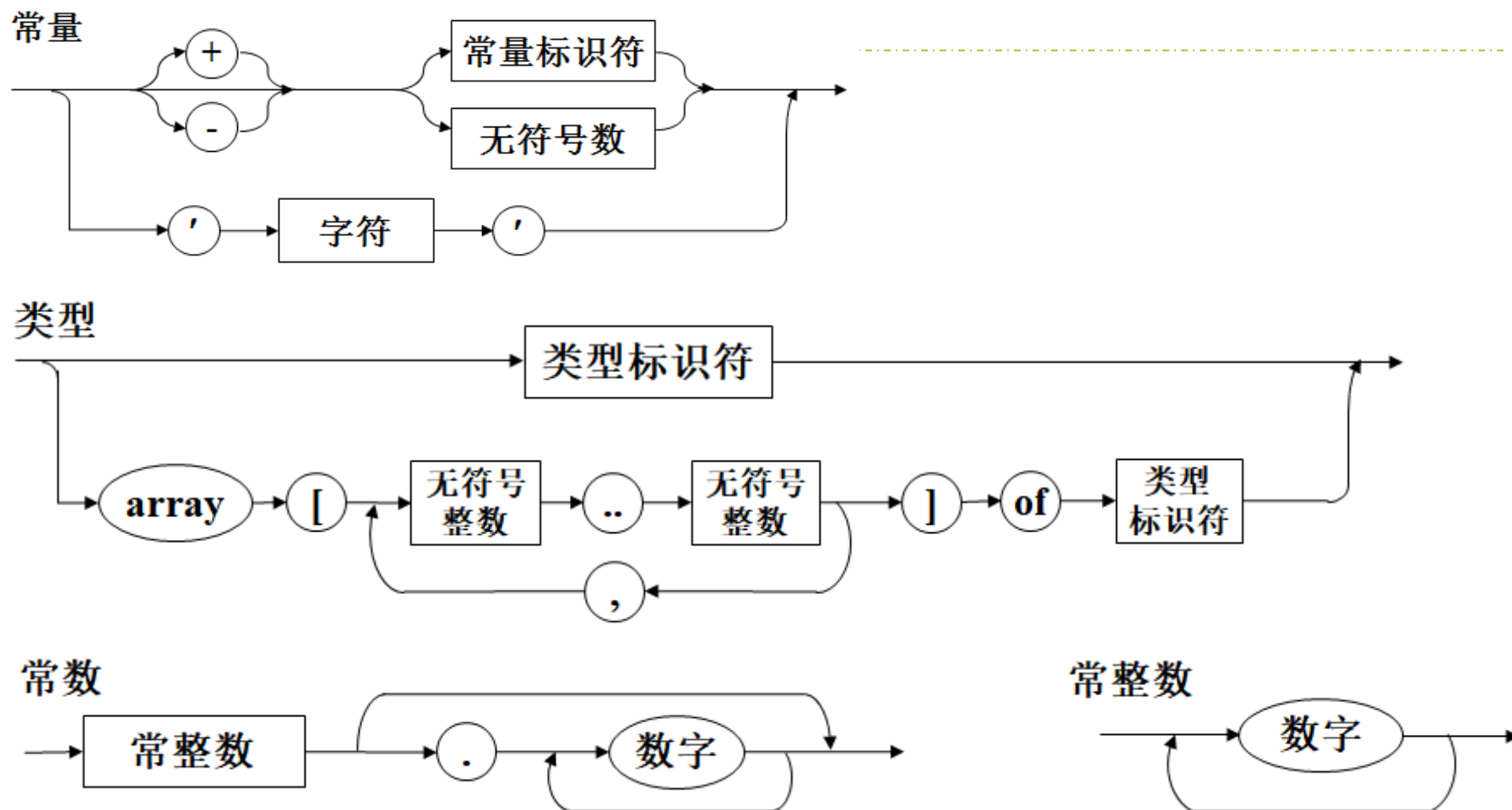
标识符



# Pascal-S语言语法图 (2/7)

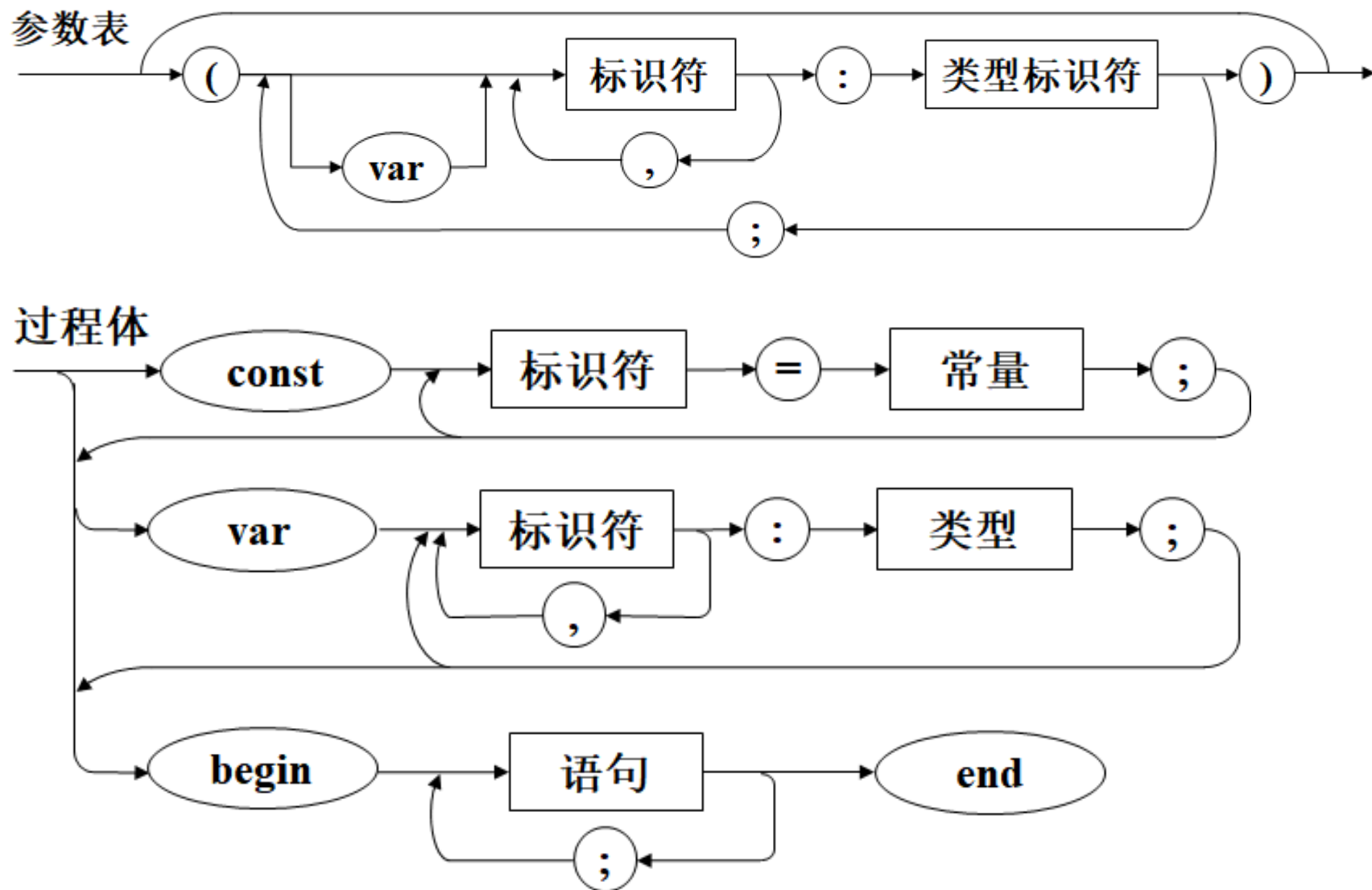


# Pascal-S语言语法图 (3/7)

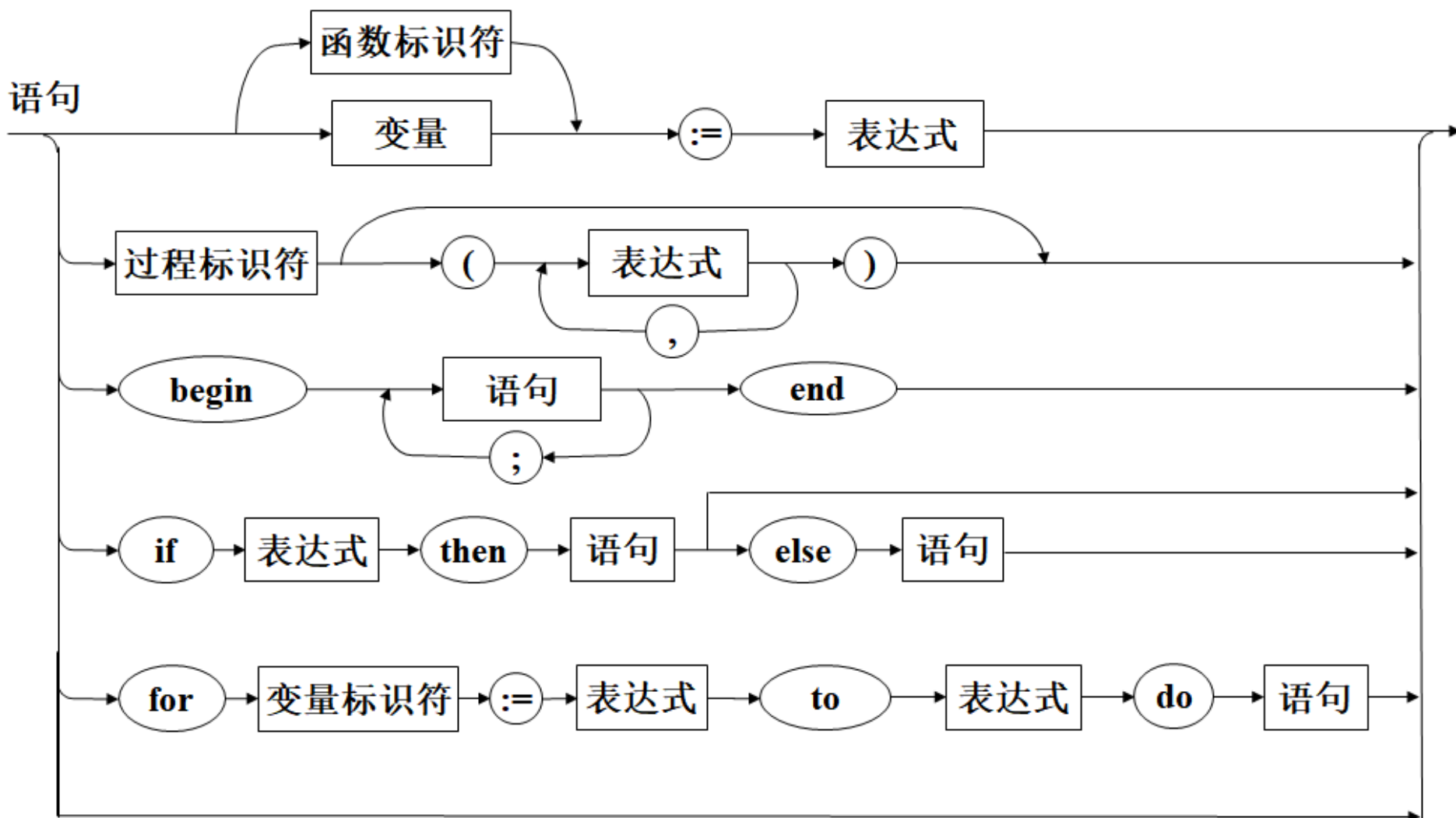


类型标识符: integer, real, boolean, char

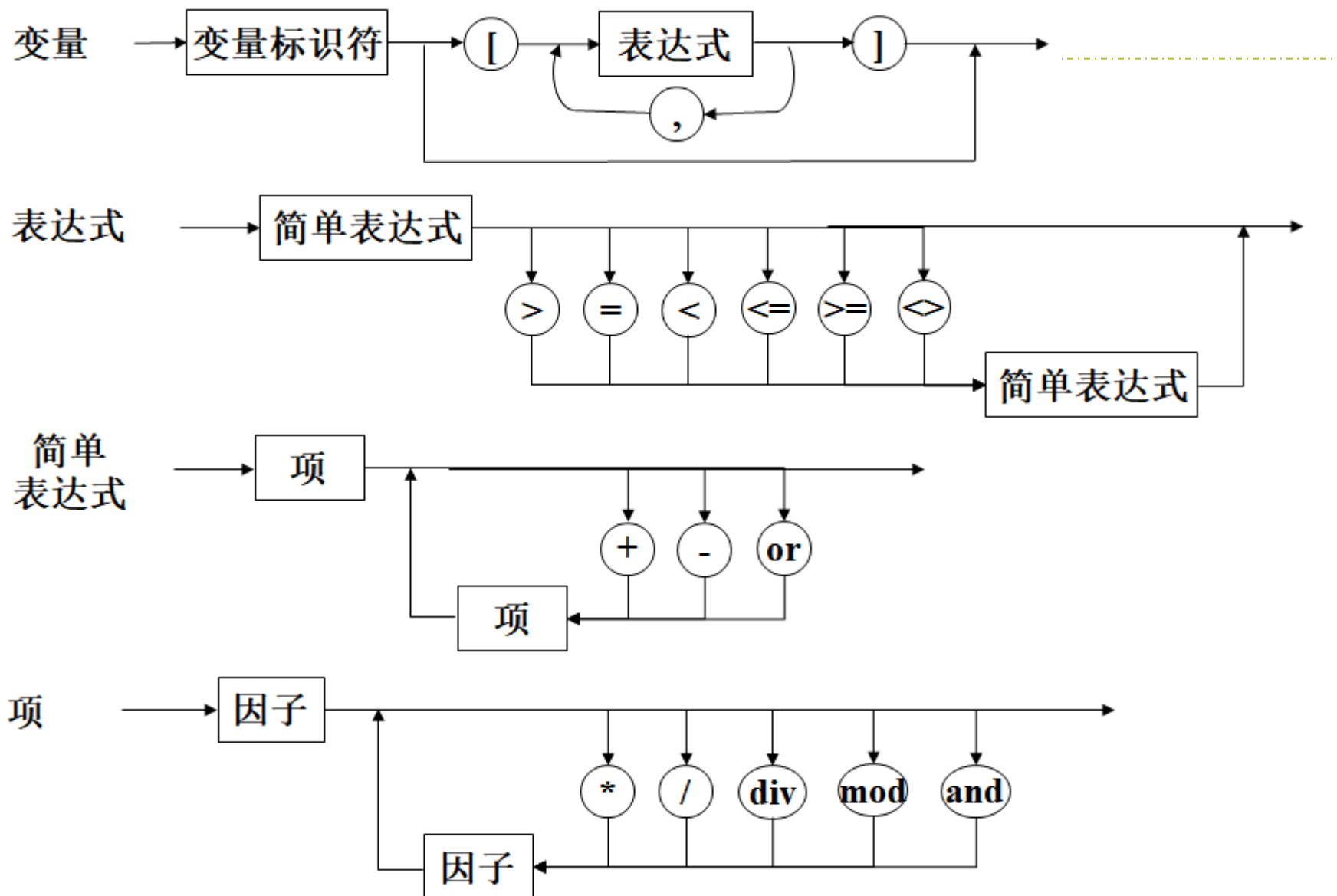
# Pascal-S语言语法图 (4/7)



# Pascal-S语言语法图 (5/7)

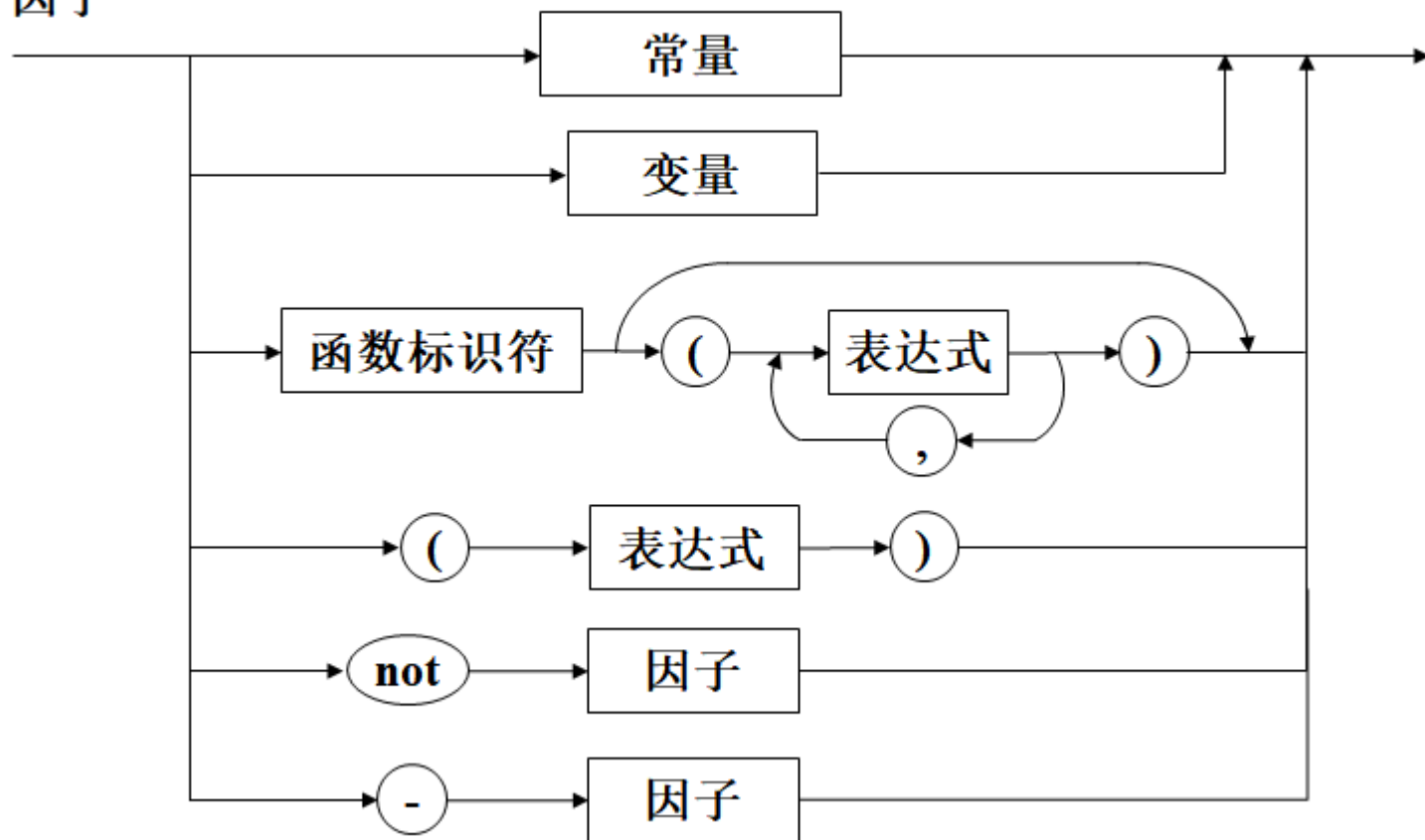


# Pascal-S语言语法图 (6/7)



# Pascal-S语言语法图 (7/7)

因子



## 2.2 Pascal-S语言文法产生式(1/7)

- $programstruct \rightarrow program\_head ; program\_body .$
- $program\_head \rightarrow \mathbf{program\ id} ( idlist )$
- $program\_body \rightarrow const\_declarations$   
 $var\_declarations$   
 $subprogram\_declarations$   
 $compound\_statement$
- $idlist \rightarrow idlist , \mathbf{id} \mid \mathbf{id}$



# Pascal-S语言的文法产生式(2/7)

- $const\_declarations \rightarrow \mathbf{const} \ const\_declaration \ ;$   
 $\quad \quad \quad | \ \varepsilon$
- $const\_declaration \rightarrow const\_declaration \ ; \ \mathbf{id} = const\_value$   
 $\quad \quad \quad | \ \mathbf{id} = const\_value$
- $const\_value \rightarrow + \mathbf{id} \mid - \mathbf{id} \mid \mathbf{id}$   
 $\quad \quad \quad | + \mathbf{num} \mid - \mathbf{num} \mid \mathbf{num}$   
 $\quad \quad \quad | ' \mathbf{letter} '$

# Pascal-S语言的文法产生式(3/7)

- $var\_declarations \rightarrow \mathbf{var} \ var\_declaration ;$   
 $\quad \quad \quad | \ \varepsilon$
- $var\_declaration \rightarrow var\_declaration ; idlist : type$   
 $\quad \quad \quad | \ idlist : type$
- $type \rightarrow simple\_type$   
 $\quad \quad \quad / \ \mathbf{array} [ period ] \ \mathbf{of} \ simple\_type$
- $simple\_type \rightarrow \mathbf{integer} \mid \mathbf{real} \mid \mathbf{boolean} \mid \mathbf{char}$
- $period \rightarrow period , \ \mathbf{digits} .. \mathbf{digits}$   
 $\quad \quad \quad | \ \mathbf{digits} .. \mathbf{digits}$

# Pascal-S语言的文法产生式(4/7)

- $subprogram\_declarations \rightarrow subprogram\_declarations$   
 $subprogram ;$   
 $| \epsilon$
- $subprogram \rightarrow subprogram\_head ; subprogram\_body$
- $subprogram\_head \rightarrow \mathbf{procedure\ id\ formal\_parameter}$   
 $| \mathbf{function\ id\ formal\_parameter : simple\_type}$
- $formal\_parameter \rightarrow ( parameter\_list )$   
 $| \epsilon$
- $parameter\_list \rightarrow parameter\_list ; parameter$   
 $| parameter$

# Pascal-S语言的文法产生式(5/7)

- $parameter \rightarrow var\_parameter \mid value\_parameter$
- $var\_parameter \rightarrow \mathbf{var} \ value\_parameter$
- $value\_parameter \rightarrow idlist : simple\_type$
- $subprogram\_body \rightarrow const\_declarations$   
 $var\_declarations$   
 $compound\_statement$
- $compound\_statement \rightarrow \mathbf{begin} \ statement\_list \ \mathbf{end}$
- $statement\_list \rightarrow statement\_list ; statement$   
 $\mid statement$

# Pascal-S语言的文法产生式(6/7)

- $statement \rightarrow variable \textbf{ assignop } expression$ 
  - |  $procedure\_call$
  - |  $compound\_statement$
  - |  $\textbf{ if } expression \textbf{ then } statement \textbf{ else\_part }$
  - |  $\textbf{ for id assignop } expression \textbf{ to } expression \textbf{ do } statement$
  - |  $\epsilon$
- $variable \rightarrow \textbf{ id } id\_varpart$
- $id\_varpart \rightarrow [ expression\_list ] \mid \epsilon$
- $procedure\_call \rightarrow \textbf{ id } \mid \textbf{ id } ( expression\_list )$
- $else\_part \rightarrow \textbf{ else } statement \mid \epsilon$

# Pascal-S语言的文法产生式(7/7)

- $expression\_list \rightarrow expression\_list , expression \mid expression$
- $expression \rightarrow simple\_expression \mathbf{relop} simple\_expression$   
 $\mid simple\_expression$
- $simple\_expression \rightarrow simple\_expression \mathbf{addop} term \mid term$
- $term \rightarrow term \mathbf{mulop} factor \mid factor$
- $factor \rightarrow \mathbf{num}$   
 $\mid variable$   
 $\mid \mathbf{id} ( expression\_list )$   
 $\mid ( expression )$   
 $\mid \mathbf{not} factor$   
 $\mid \mathbf{uminus} factor$

## 2.3 文法说明(1/2)

- 该文法是一个LALR(1)文法。
  - 可以采用LR分析方法；
  - 也可以对文法消除左递归后，采用递归下降分析法。
- 该文法产生的每一个句子都是一个Pascal-S程序。
  - 每个程序可以包含若干个全程常量/变量的声明语句、过程和函数的定义、以及作为主程序体的复合语句。
- 过程和函数中不允许嵌套其他的函数/过程定义。
- 复合语句允许嵌套。
- 过程和函数都可以递归调用。
- 支持传值调用和引用调用（传地址）两种参数传递方式。

# 文法说明(2/2)

- 注意：产生式  $factor \rightarrow id$   
id既可以是函数名，也可以是常量名、简单变量名。  
对无参数函数的调用和引用简单变量的值，在语法上没有区别。  
例如，对于赋值语句  $a:=b$ 
  - 如果 b 是常量名，则把 b 的值赋予 a；
  - 如果 b 是简单变量，则把 b 的右值赋予 a；
  - 如果 b 是函数，则把函数 b 的返回值赋予 a。
- 对全程变量可以进行静态存储分配，过程和函数的局部变量分配在活动记录里（栈式存储分配）。
- 读/写过程可以直接引用实现环境里的相应过程。



## 2.4 关于词法的约定 (1/3)

- 源程序中的关键字（除开头的 **program** 和末尾的 **end** 之外）前、后必须有空格符或换行符，其它词汇间的空格符是可选的。
- 源程序中的注释：用一对花括号括起来，可以出现在任何单词之后。
  - 编译程序应该可以处理注释。
- 标识符的记号 **id**，匹配以字母开头的字母数字串。  
定义为：  
**letter**  $\rightarrow$  [a-zA-Z]  
**digit**  $\rightarrow$  [0-9]  
**id**  $\rightarrow$  **letter** ( **letter** | **digit** ) \*
  - 对其最大长度可以规定一个限制。

## 关于词法的约定 (2/3)

- “常数”的记号 **num**，匹配整型常数或实型常数。  
定义为：

**digits** → **digits digit | digit**

**optional\_fraction** → **. digits |  $\epsilon$**

**num** → **digits optional\_fraction**

- 关键字作为保留字。

# 关于词法的约定 (3/3)

- 关系运算符 **relop** 代表  
=、<>、<、<=、>、>=
- **addop** 代表运算符 +、- 和 or
- **mulop** 代表运算符 \*、/、div、mod 和 and
- **assignop** 代表赋值号 :=

## 2.5 课程设计建议(1/7)

### ■ 详细的需求分析

#### ● 词法分析

- 单词种类、单词模式、右线性文法
- 注释、分隔符
- 错误

#### ● 语法分析

- 语法结构、语法错误类型?
- 文法, 分析方法选择
- 改写文法 (扩展、简化?)

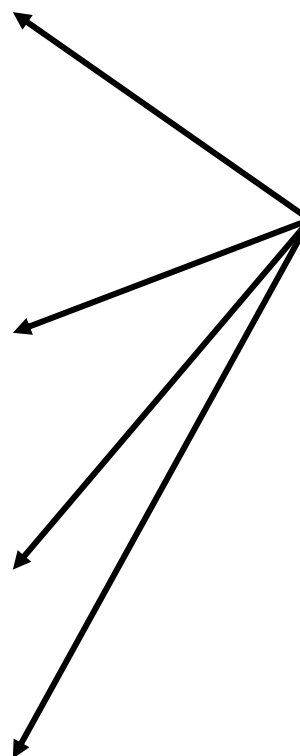
#### ● 语义分析

- 类型、类型检查、作用域、符号表内容

#### ● 代码生成

- 目标语言、源/目标语言的映射关系

测试?  
用例?



# 课程设计建议(2/7)

## ■ 软件总体设计

- 软件功能（功能模块划分）
- 软件结构（功能模块之间的关系）
  - 递归调用函数
  - LR分析技术
    - YACC
    - 手工编码
- 模块之间的接口
  - 数据结构
  - 文件
- 符号表设计（内容、逻辑结构）
- 错误处理（恢复策略）

语法制导定义  
/  
翻译方案

# 课程设计建议(3/7)

## ■ 符号表结构及其管理程序的设计

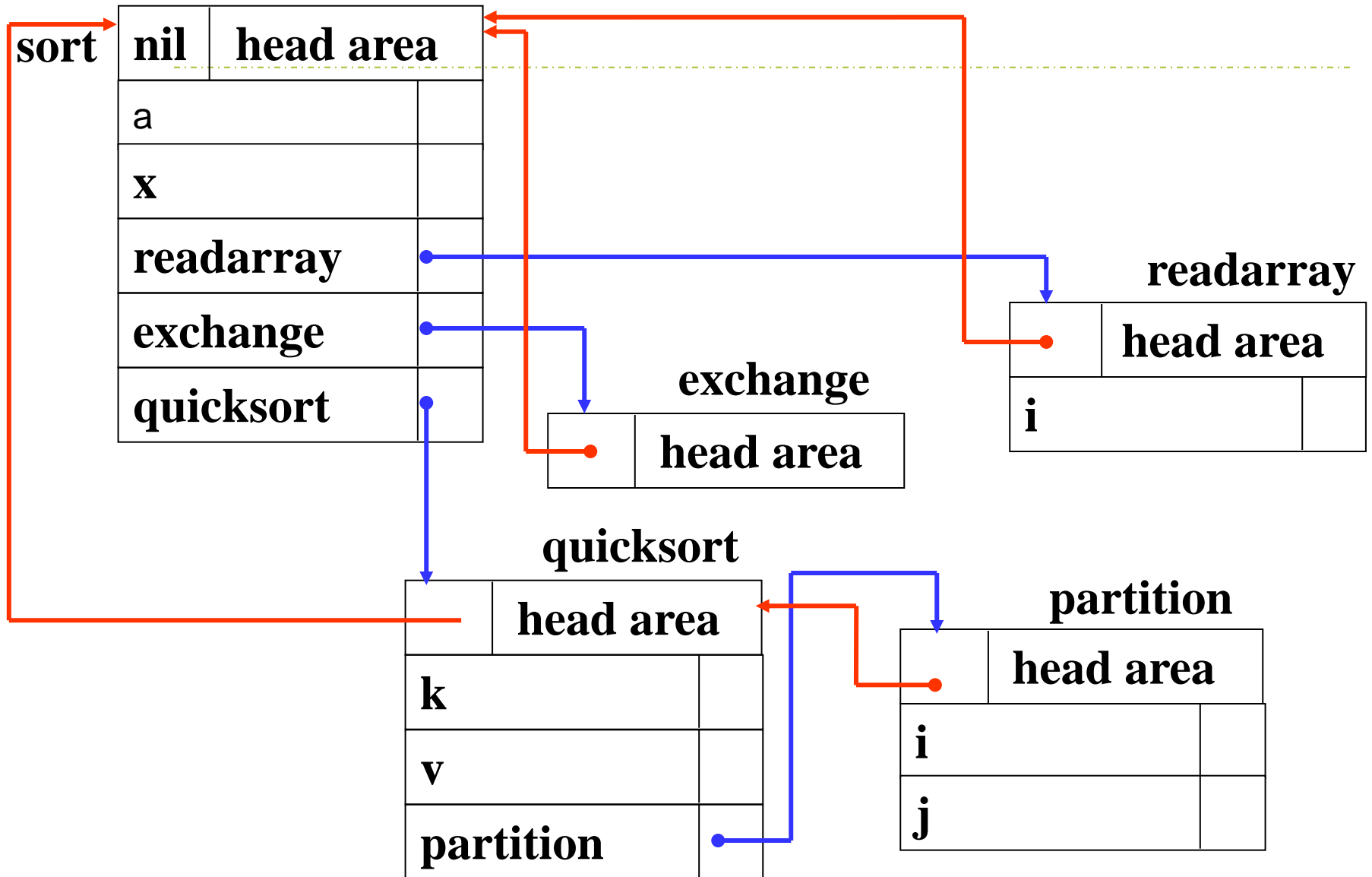
- 首先设计符号表的结构，允许在编译的各个阶段填入或查找关于名字的相关信息。
  - 名字、类型（常量名？）
  - 数组（维数、上下界）
  - 函数（参数个数、参数类型、传递方式）
  - 变量值存在？变量值？
- 管理程序
  - 查找操作：按给定的名字查表。
  - 插入操作：在表中建立新的一行。
  - 定位操作：创建符号子表，确定一个新作用域的起点。
  - 重定位操作：从符号表中“删除”局部于当前过程的所有名字，退出当前作用域。

# 符号表的逻辑结构

```
program sort (input,output);  
  var a : array[0..10] of integer;  
      x : integer;  
  
  procedure readarray;  
    var i : integer;  
    begin  
      for i:=1 to 9 do read(a[i])  
    end;  
  
  procedure exchange (i,j:integer)  
    begin  
      x:=a[i]; a[i]:=a[j]; a[j]:=x  
    end;
```

```
  procedure quicksort (m,n:integer);  
    var k,v : integer;  
  function partition (y,z :integer):integer;  
    var i,j : integer;  
    begin  
      ...a...;    ...v...;  
      exchange(i,j);    .....  
    end;  
    begin  
      .....  
      k=partition(m,n);  
      quicksort(m,k-1);  
      quicksort(k+1,n);    .....  
    end;{quicksort}  
  begin readarray; quicksort(1,9)  
  end. {sort }
```

# 符号表的逻辑结构





# 课程设计建议（4/7）

## ■ 词法分析器的详细设计

- 首先确定单词符号的种类，给出每类单词的文法。
- 确定有哪些关键字；
- 考虑并确定每类单词的内部编码及其属性值（给出翻译表）。
- 考虑每个单词在源程序中出现的位置信息（行/列），用于向用户提供错误定位信息。
- 把词法分析器作为语法分析器调用的函数，词法分析器返回词汇的类别编码和属性值。

# 课程设计建议 (5/7)

## ■ 语法分析器的详细设计

- 选择适当的分析方法
  - 自顶向下的预测分析方法（消除左递归）
  - 自底向上的LR分析方法
- 手工地设计语法分析器。
  - 预测分析方法  
消除左递归，给出消除左递归之后的文法。  
构造预测分析表。  
编码实现预测分析程序。
  - LR分析方法  
构造LALR(1)分析表。  
编码实现LR分析程序。
- 用YACC工具实现。

# 课程设计建议（6/7）

## ■ 语义动作和翻译程序的详细设计

- 按照语言的语义动作设计语义子程序。为了便于语法制导翻译，需要对给定的文法进行改造，要注意数据类型的相容性，必要时要进行数据类型转换(如inttoreal)。
- 注意：
  - 本语言不支持过程嵌套定义
  - 需要进行类型检查
  - 选择目标语言，汇编、**C（建议C）**
  - 若选C语言作为目标语言，设计Pascal-S语言与C语言之间的对应关系，语义动作，写出翻译方案。
  - 若选汇编语言作为目标语言，用三地址代码或四元式组作为中间表示，在写程序之前，先设计Pascal语言与中间语言、中间语言与汇编语言之间的对应关系，语义动作，写出翻译模式。

# 课程设计建议 (7/7)

## ■ 代码生成程序的详细设计

- 用C语言作为目标语言，可以直接从源程序生成目标程序，省去中间代码生成部分。  
但是，由于Pascal-S程序和C程序的结构不完全一致，故不能逐句翻译输出。
- 采用汇编语言作为目标语言，以Intel机器作为目标机器，利用教材中介绍的目标代码生成算法。

## ■ 错误处理与恢复

- 在词法分析、语法分析、语义分析、代码生成等各个过程中都要考虑错误的处理与恢复。打印错误信息，如有可能，应对错误进行恢复，并继续进行编译。

## 2.6 测试（1/3）

- 根据需求分析的结果，制定测试计划，设计测试用例。
- 根据测试用例对所实现编译程序进行测试，记录测试结果，并进行分析。
- 错误检测和处理
  - 设计含有错误的测试用例源程序。
  - 用所实现的编译器对测试程序进行编译。
  - 展示错误检查、恢复结果。
  - 白盒测试
  - 黑盒测试
- 编译下面的源程序example。
  - 记录生成的目标程序。
  - 对目标程序进行编译运行，记录运行结果。

## 测试 (2/3)

```
program example(input,output);  
  var x,y:integer;  
  function gcd(a,b:integer):integer;  
    begin  
      if b=0 then gcd:=a  
      else gcd:=gcd(b, a mod b)  
    end;  
  begin  
    read(x, y);  
    write(gcd(x, y))  
  end.
```

# 测试（3/3）

- 用Pascal-S语言编写快速排序程序。
  - 用所实现的编译器对快速排序程序进行编译，输出/保存所生成的目标程序。
  - 对目标程序进行编译、运行，输出/保存运行结果。
  - 要求：可以随机输入待排序的数据。

## 2.7 开发方法

- 遵循软件工程的思想，分阶段开发，分阶段测试。
  - 先进行需求分析和总体设计
  - 需求分析：重点分析编译程序的功能需求。
  - 总体设计：主要包括分析方法的选择、软件功能结构及模块划分、模块间接口的定义(包括全局数据结构的定义)。
  - 然后根据软件功能划分，进行组员分工。
  - 保证每个局部结果的正确，保证最终结果的正确。
- 开发过程中采用“滚雪球”的方法
  - 使工作顺利进行
  - 企图一步到位，往往欲速则不达。



## 2.8 设计报告要求

1. 课程设计题目
2. 课程设计目标和要求
3. 需求分析，包括：数据流图、功能及数据说明等
4. 开发环境
5. 总体设计说明，包括：
  - 1) 数据结构设计
  - 2) 总体结构设计：包括
    - 功能模块的划分
    - 模块功能
    - 模块之间的关系
    - 模块之间的接口
  - 3) 用户接口设计
6. 各部分的详细设计说明，包括：
  - 接口描述
  - 功能描述
  - 所用数据结构说明
  - 算法描述
7. 程序清单
  - 注意编程风格，如：
    - 使用有意义的变量名、程序的缩排、程序的内部注释
8. 测试报告，包括：
  - 1) 测试环境
  - 2) 测试计划
  - 3) 针对每个功能的测试情况，包括：测试用例、预期的结果、测试结果及其分析

在设计测试计划时，不但要考虑正确的测试用例，还要考虑含有错误的测试用例。
9. 实验总结
  - 1) 实验中遇到或存在的主要问题
  - 2) 改进建议
  - 3) 体会/收获

# 3. 课程设计安排（1/5）

## ■ 分组

- 5至6人一组，自由组合。
- 小组成员分工合作、每个人的任务明确。
- 每组选定一名项目组长
  - 负责管理项目的进度、质量
  - 负责组员的任务分配、沟通、协调
  - 完成自己承担的任务
- 要求每个组员必须承担一定的功能模块设计开发工作，要求每个人所完成的任务相对独立；

## ■ 集中上课安排

- 第一周：介绍课程设计任务和要求，授课计划和安排、完成分组等。
- 第二周：各组汇报需求分析结果、制定的测试计划、编制的测试用例等。
- 第三周：各组汇报软件总体设计结果
  - 系统总体结构设计、功能设计、及功能模块划分、全局数据结构的定义、模块间接口的定义；
  - 重点：符号表的结构、模块划分及模块接口。

# 课程设计安排 (2/5)

## ■ 分工建议

- 词法分析+测试计划：1人
- 语法分析
  - 手工编码：1-2人
- 语义分析：
  - 手工编码：1-2人
  - YACC：语法+语义：1-2人
- 代码生成：1-2人

■ 整理文档是每个组员任务的一部分。

■ 每个组员完成自己所编写模块的单元测试。

■ 组长统筹安排软件的集成测试。

# 课程设计安排（3/5）

## ■ 验收时间、地点、及要求

- 时间：

- 在第8周之前完成的各小组，随时可以约时间申请验收。
- 第8、9周验收。最晚，第10周完成验收。

- 地点：教3-623

- 要求：

- 通过邮件预约验收，预约的同时提交相关资料。
- 项目小组成员全部参加，否则，不予验收。
- 没有参加验收的同学，没有成绩。

# 课程设计安排（4/5）

## ■ 申请验收时，需要提交以下资料（电子版）：

- 1、课程设计表格（分组表）
- 2、课程设计报告
- 3、程序，包括：
  - 1) 源程序
  - 2) 可运行的程序
  - 3) 测试用例程序
- 4、程序使用说明（可以在课程设计报告中体现，也可以是单独的文件）
  - 如发现抄袭，一律按0分计。

## ■ 验收时，携带打印好的《验收登记表》

# 课程设计安排 (5/5)

## ■ 验收内容及过程：

- 查看设计文档；
- 讲解所设计符号表的结构、及在符号表上的操作（查找、插入、定位、重定位）
- 源程序中注释的处理
- 能检测哪些语法错误？
- 支持的类型有哪些？实现了哪些类型检查？  
表达式、语句、参数？
- 所采用的中间代码生成技术、及目标代码生成算法
- 错误处理与恢复策略
- 演示测试用例的编译过程和结果
- 每个人讲解自己完成的工作内容，必要时需要讲解所编写的功能代码。

## 4. 成绩评定

### ■ 小组基准成绩评定：

- 软件完成情况50：（30+）
  - 完成的功能
  - 运行过程、结果可视化展示
- 平时成绩30：考勤（10）、课堂交流（10）、验收（10）
- 文档20：正确性、与程序的一致性。

### ■ 小组成员成绩

- 各组提供成员的贡献率/排名
- 视小组成员完成工作情况，上下浮动。

### ■ 组与组之间比较，根据各组工作的难易程度适当浮动成绩。

- 有能力的同学可以把子集扩大，如增加类型、语句，进行代码优化等。
- 也可以对所给语言子集进一步缩减

---

希望大家通过本次教学实践：  
理解、掌握编译原理和技术  
提高编程能力

谢 谢！