



# 编译原理与技术 课程设计



李文生

2024年2月28日 星期三



北京邮电大学计算机学院(国家示范性软件学院)

SCHOOL OF COMPUTER SCIENCE(NATIONAL PILOT SOFTWARE ENGINEERING SCHOOL)BUPT

# 课程简介

## ■ 课程基本信息

课程名称	中文：编译原理与技术课程设计		课程编号	3132102100
	英文： <b>Course Design of Compiler Principle and Technology</b>			
学分/学时	1.5/36	必修（） / 选修（√）	开课学期	6
课程类别	实践教学课程	适用专业	计算机科学与技术	
先修课程	编译原理与技术			

# 课程简介

## ■ 课程目标

▣ 对标毕业要求：12大项，32个指标点

(1) 工程知识	(2) 问题分析	③ 设计/开发解决方案
(4) 研究	⑤ 使用现代工具	⑥ 工程与社会
(7) 环境和可持续发展	(8) 职业规范	⑨ 个人与团队
⑩ 沟通	(11) 项目管理	(12) 终身学习

▣ 本课程的目标：5个目标，对应5个指标点

3.2    5.3    6.2    9.2    10.1

# 课程目标

课程目标	对应指标点	描述
1	3.2 设计/开发解决方案	针对课程设计的任务要求，对编译程序的设计与实现进行分解和细化，培养学生的系统分析和设计能力，培养学生设计开发功能模块及计算机系统软件的能力。
2	5.3 使用现代工具	培养学生能够运用计算机开发环境和工具，对设计的编译程序原型系统进行功能仿真、测试，提高系统分析问题和解决问题的能力。
3	6.2 工程与社会	培养学生合理分析和评价编译程序设计与实现相关的工程实践和复杂工程问题解决方案的可行性，及解决方案可能带来的安全、法律等方面的影响的能力，理解应承担的责任的能力。
4	9.2 个人与团队	培养学生根据所承担的角色，组织、协调和带领团队开展工作的能力，在团队中完成自己承担的任务的能力。
5	10.1 沟通	培养学生根据课程设计的任务要求，撰写设计文档和课程设计报告的能力，通过对课程设计成果进行陈述、展示和答辩，培养学生针对计算机及相关信息领域复杂工程问题与业界同行及社会公众进行有效沟通和交流的能力。

# 课程目标达成途径

## ■ 教学活动

- 课堂讲解和交流、自主设计开发、成果验收

## ■ 考核环节：

- 程序设计验收答辩(70%)、设计报告(30%)

## ■ 程序设计

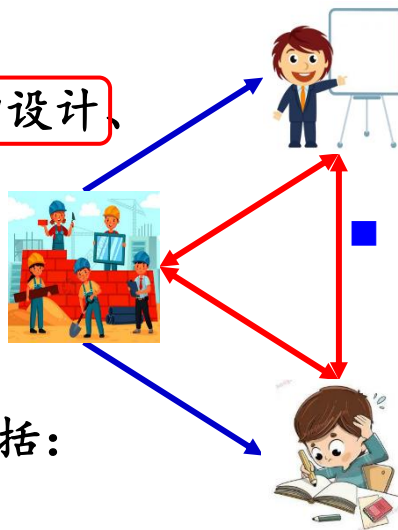
### ① 系统分析与总体设计，包括：

- 设计目标、需求分析、**程序架构设计**，以及程序总体功能设计；
- 符号表设计：内容、结构及相关函数。

### ② 开发环境及工具的选择；

### ③ 程序功能模块详细设计与实现，包括： 词法分析模块、语法分析模块、 类型检查模块、以及代码生成模块； 符号表管理、错误处理等

### ④ 程序功能测试，包括单元测试与集成测试。



## ■ 验收答辩

### ① 陈述：

- 总体设计：程序架构、总体功能；
- 符号表设计：内容、结构及相关函数；

### ② 程序功能演示，包括：

- 词法分析、语法分析、类型检查、以及代码生成等功能；
- 错误检测和处理。

### ③ 各成员陈述自己负责的内容、答辩。

## ■ 设计报告

### ① 课程设计目标任务

### ② 需求分析及总体设计

### ③ 功能模块的详细设计

### ④ 程序源代码结构及功能说明；

### ⑤ 程序测试报告；

### ⑥ 课程设计收获/体会及建议。

# 评分参考标准

## ■ 程序设计验收答辩：

- ① 按时参加验收答辩并提交资料，程序具有基本的分析功能：计60分；
- ② 设计思路清晰正确，程序运行正常，符号表设计合理：加5-10分；
- ③ 程序运行稳定，视分析功能是否完善，错误提示信息是否准确详细，是否具有错误恢复功能：加5-10分；
- ④ 程序运行稳定，且可以正确输出目标代码：加5-10分；
- ⑤ 程序测试用例设计合理、测试结果分析完整：加5-10分；

## ■ 设计报告：

- ① 完成课程设计内容，按时提交课程设计报告，报告能够基本清晰地描述课程设计目标任务、开发环境、系统分析与设计、团队分工情况、系统测试情况，个人总结：计60分；
- ② 报告能够清晰准确地描述系统分析与设计，需求分析明确、设计正确、描述规范、语句通顺：加10-20分；
- ③ 测试报告内容完整、描述规范：加5-10分；
- ④ 课程设计报告与程序设计内容一致，成员分工合理，课程设计收获/体会总结真实，建议合理：加5-10分。

# 课程设计任务和目的

## ■ 任务

- 按照源语言（比如Pascal语言、C语言等）的语法和语义，设计并实现相应的编译程序，给出各阶段的设计文档和实现成果。

## ■ 目的

### □ 知识和技能

- 原理：词法分析、语法分析、语义分析、（中间）代码生成

- 技术：基于自动机的分析技术、语法制导翻译技术

### □ 综合能力，解决复杂工程问题的能力

- 分析问题、问题分解

- 解决方案、方案实施

- 表达、沟通交流

# 编译器的种类

## ■ 本地编译器

- 生成的目标代码，在与编译器本身所在的计算机和操作系统（平台）相同的环境下运行。

## ■ 交叉编译器

- 生成的目标代码，在其它平台上运行。

## ■ “源码到源码编译器”

- 输入是一种高级语言，输出是另一种高级语言。



# 发展历史

(1924年12月3日-2007年3月17日)，1977年图灵奖得主。  
美国计算机科学家，全世界第一个高级语言的发明小组组长。  
提出了BNF（用来定义形式语言语法的记号法）  
被誉为“Fortran 语言之父”。

## ■ 1950s

□ IBM的**John Backus**带领一个研究小组对 Fortran 语言及其编译器进行开发。

□ **Noam Chomsky** 开始了对自然语言结构的研究。

(1928年12月7日—) 美国哲学家。  
MIT 语言学的荣誉退休教授。

《句法结构》20世纪理论语言学研究上最伟大的贡献。

➤ 使得编译器的结构异常简单，甚至还带有了一些自动化。

➤ 根据语言文法的难易程度以及识别它们所需要的算法来对语言分类。

□ 上下文无关文法被证明是程序设计语言中最有用的，  
程序设计语言结构的标准方式。

□ 有限自动机和正则表达式同上下文无关文法紧密相关，与Chomsky的3型文法相对应。表示程序设计语言的单词的符号方式。

## ■ 1960s~1970s

□ 分析问题（parsing problem，用于2型文法识别的有效算法）的研究，并相当完善的得到解决。它已是编译原理中的一个标准部分。

# 发展历史

## ■ 编译器自动构造

- 最初称为：编译器的编译器（Compiler-Compiler）
- 确切称为：分析程序生成器（Parser Generator）  
因为它们仅仅能够自动处理编译的一部分。
- 著名：YACC（Yet Another Compiler-Compiler）  
1975年，Steve Johnson为Unix系统编写。
- 类似的，有限自动机的研究也发展了一种自动工具，  
称为扫描程序生成器（Scanner Generator）。  
佼佼者：LEX（与YACC同时，由Mike Lesk为Unix系统开发）。
- 1970s后期~1980s早期，大量项目关注于编译器其它部分的生成自动化，  
这其中就包括代码生成。
  - 并未取得多少成功，原因：操作太复杂，了解不够。

# 发展历史

## ■ 1990s

- 作为GNU项目或其它开源项目的一部分，开发出许多免费编译器和编译器开发工具。

## ■ 常用的 C/C++ 编译器

- MSVC (cl.exe): Microsoft Visual Studio
- GCC/G++
- Clang: based on LLVM, for iOS / OS X
- BCC/Borland C++: Borland
- ICC: Intel

## ■ 即时编译器 JIT

## ■ OpenMP (Open Multi-Processing)

- OpenMP Architecture Review Board牵头提出
- 用于共享内存并行系统的多处理器程序设计的一套指导性编译处理方案
- OpenMP支持的语言包括 C、C++、Fortran。
- 支持 OpenMP的编译器包括：  
Sun Compiler, GNU Compiler 和 Intel Compiler 等，比如 VS、gcc、clang。
- 可移植性好：Unix/Linux和Windows
- 提供了对并行算法的高层的抽象描述
- `#include <omp.h>`  
`#pragma omp parallel`  
`gcc file.c -fopenmp`

# 国产编译器

## ■ 华为方舟

- 2019.8.31 正式上线
- 2019.11 获“2019东北亚优秀开源项目”奖
- <https://www.openarkcompiler.cn/>
- 专门为软件厂商研发的统一编程平台，包含编译器、工具链、运行时等关键部件。
- 支持多种编程语言、多种芯片平台的联合编译与运行。
- 能够有效解决安卓程序“边解释边执行”的低效率问题。
- 基于 GCC 开发的交叉编译器套件，包括了：
  - C、C++、Fortran的前端
  - 这些语言的库 (如 libstdc++、libgcc 等)。
- HCC运行在X86 linux架构服务器上，生成的二进制运行在Aarch64架构服务器上。

# 国产编译器

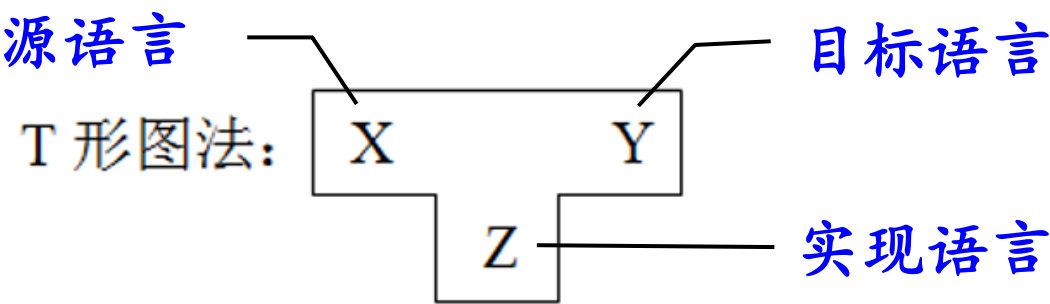
## ■ UCC

- 遵从 ANSI C89 标准的编译器，大约 15,000 行C代码。
- 支持x86平台上的Linux和Windows系统，能正确编译自身并成功运行。
- <https://download.csdn.net/tagalbum/705638>
- <https://www.oschina.net/p/ucc-compiler/related?sort=time&lang=21>

## ■ YC

- <http://www.ycbro.com/>
- <https://blog.csdn.net/xiaobingyang/article/details/104614780>
- <https://zhuanlan.zhihu.com/p/112244597>

# 编译程序的表示方法

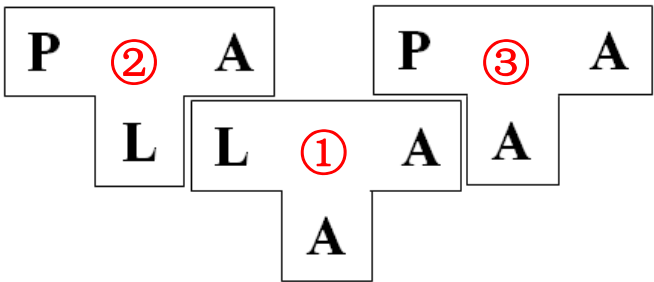


记号法:  $C_Z^{XY}$

若在A机型上已有语言L的编译程序，用记号  $C_A^{LA}$  表示。  
若L是自编译语言，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  $C_A^{PA}$ 。



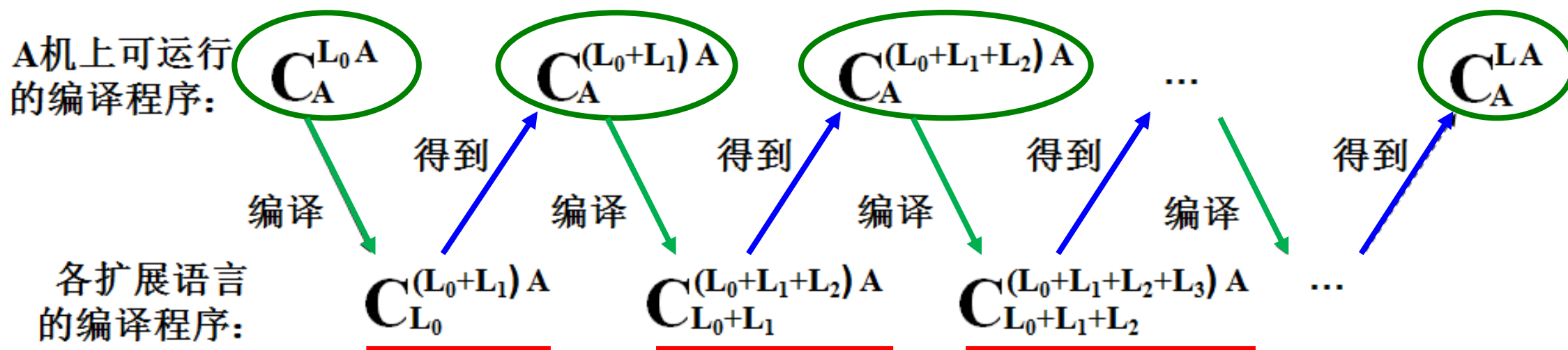
(a) 记号描述



(b) T形图表示

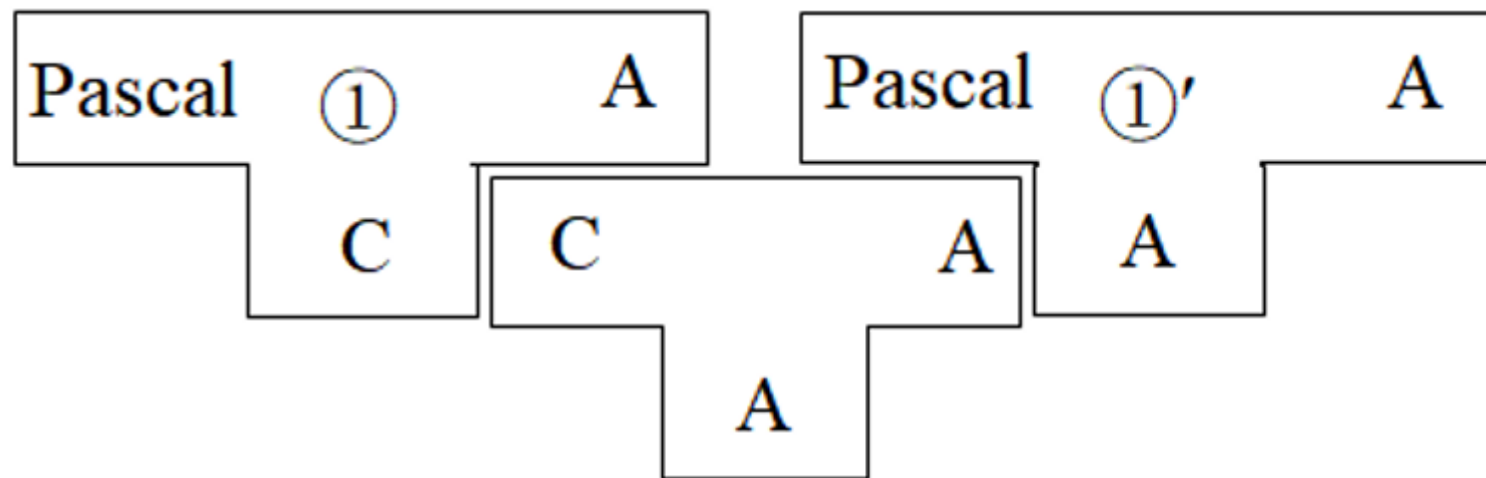
# 编译程序的实现方法—自展法

- 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  $C_A^{LA}$
- 语言子集： $L_0$ ,  $L_0+L_1$ ,  $L_0+L_1+L_2$ ,  $L_0+L_1+L_2+L_3$ , ...,  $L$



# 编译程序的实现方法（T型图表示）

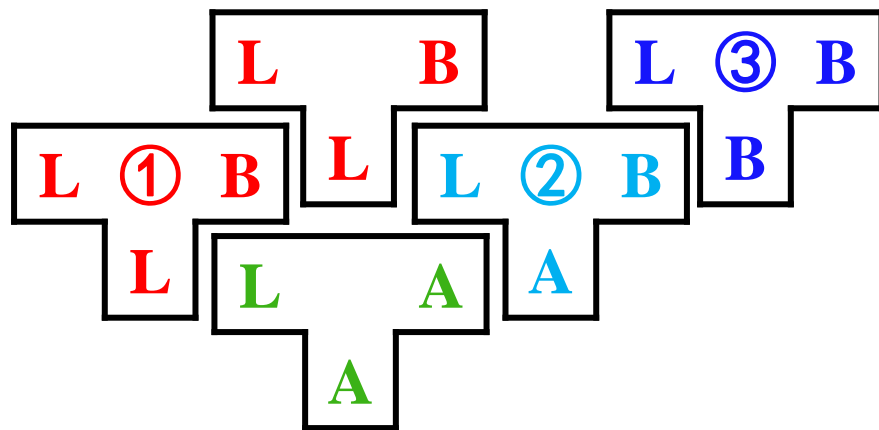
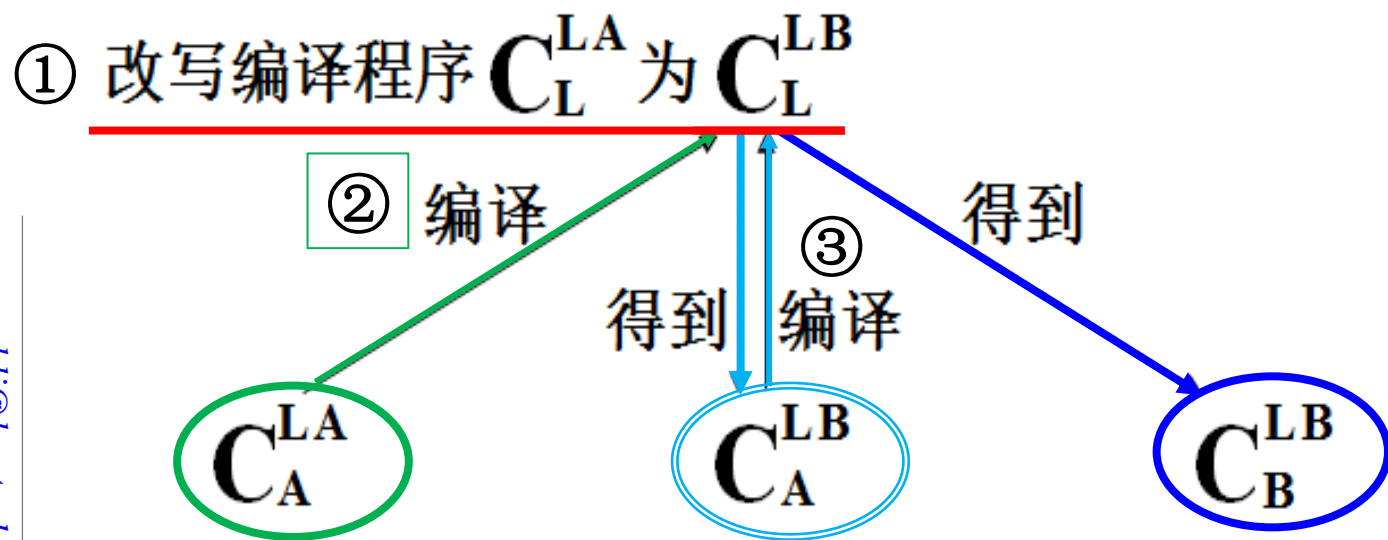
- 在A种机上已经有 C 语言可用
- 如何使一种新的语言（如Pascal语言）也可以在A机上使用？





# 编译程序的实现方法--移植法

- 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  $C_L^{LA}$
  - 可在A机上运行的语言L的编译程序  $C_A^{LA}$
- 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  $C_B^{LB}$



# 课程内容

## 1. 课程设计任务与要求

1.1 MiniPascal语言语法图

1.2 MiniPascal语言文法产生式

1.3 文法说明

1.4 关于词法的约定

1.5 课程设计建议

1.6 测试

1.7 开发方法

1.8 设计报告要求

## 2. 教学安排

# 1. 课程设计任务说明

- 题目：MiniPascal语言编译程序的设计与实现

- 目标：

按照所给MiniPascal语言的语法，参考Pascal语言的语义，设计并实现MiniPascal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

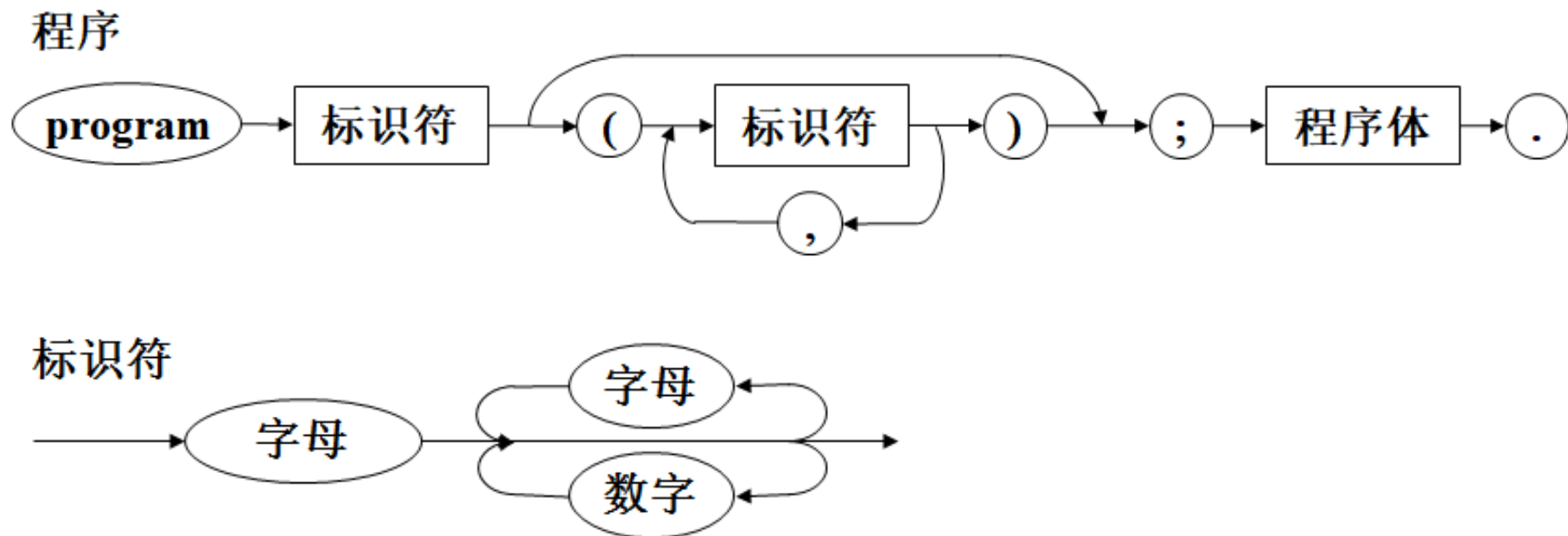
- 总体设计报告（软件功能描述、功能模块划分、软件结构图、符号表结构设计、模块间接口定义等）

- 详细设计报告（模块功能、输入/输出、处理逻辑等）

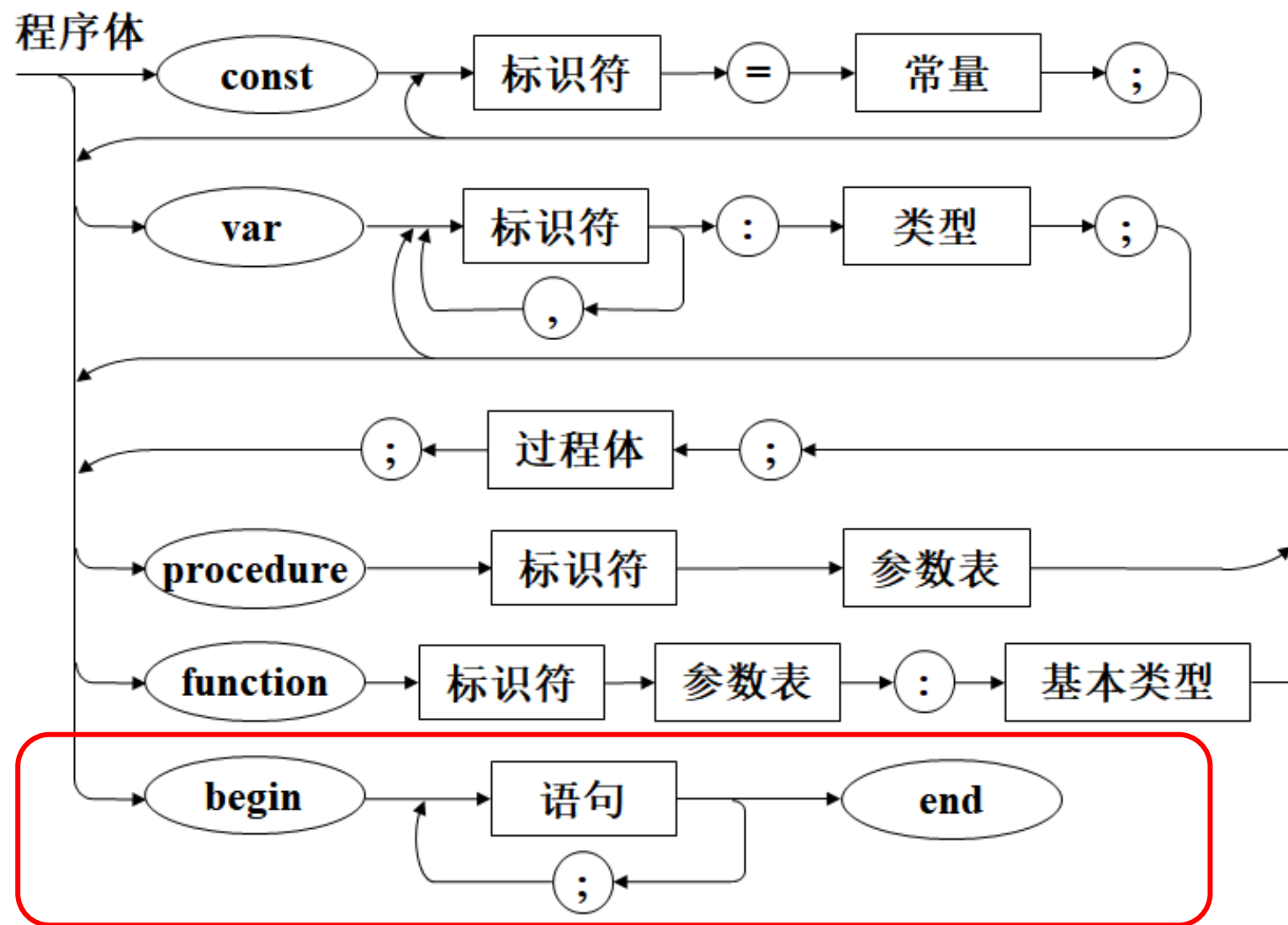
- 编码实现（源程序、可执行程序）

- 测试报告（测试计划、测试用例、测试结果及分析等）

# 1.1 MiniPascal语言语法图

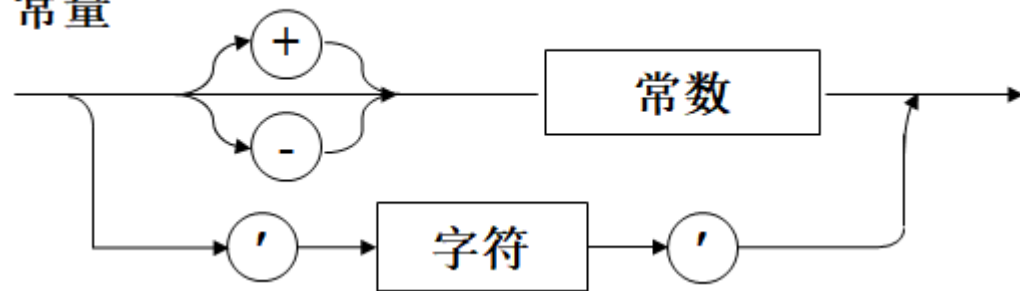


# MiniPascal语言语法图

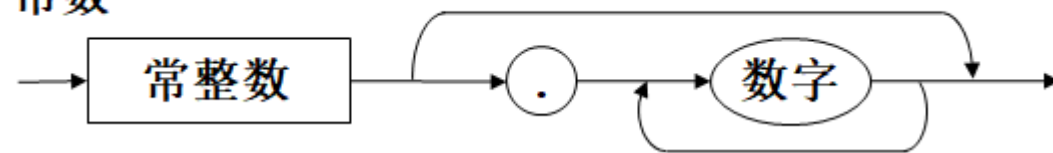


# MiniPascal语言语法图

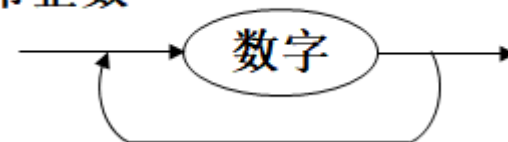
常量



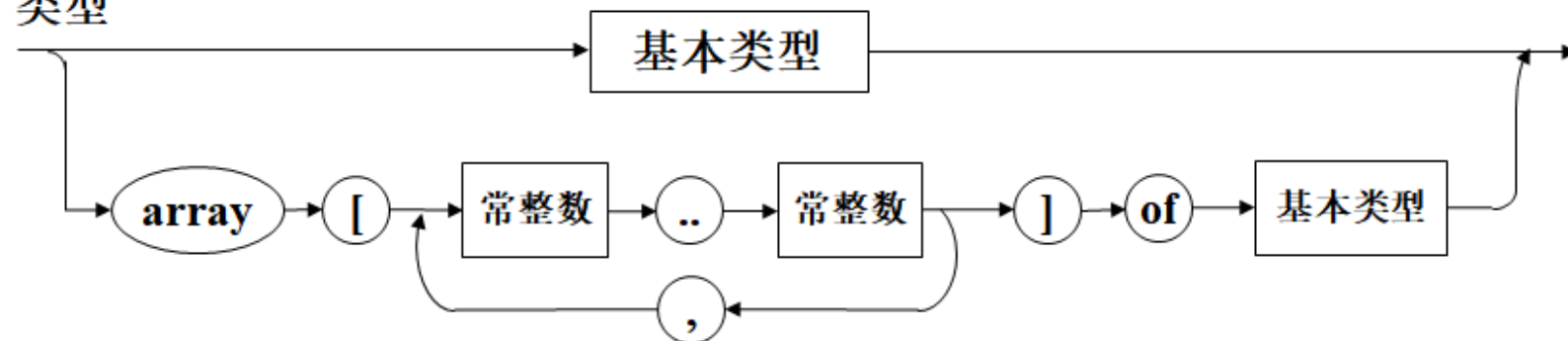
常数



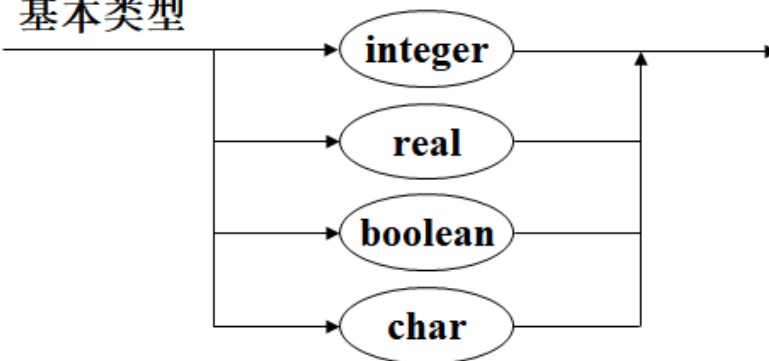
常整数



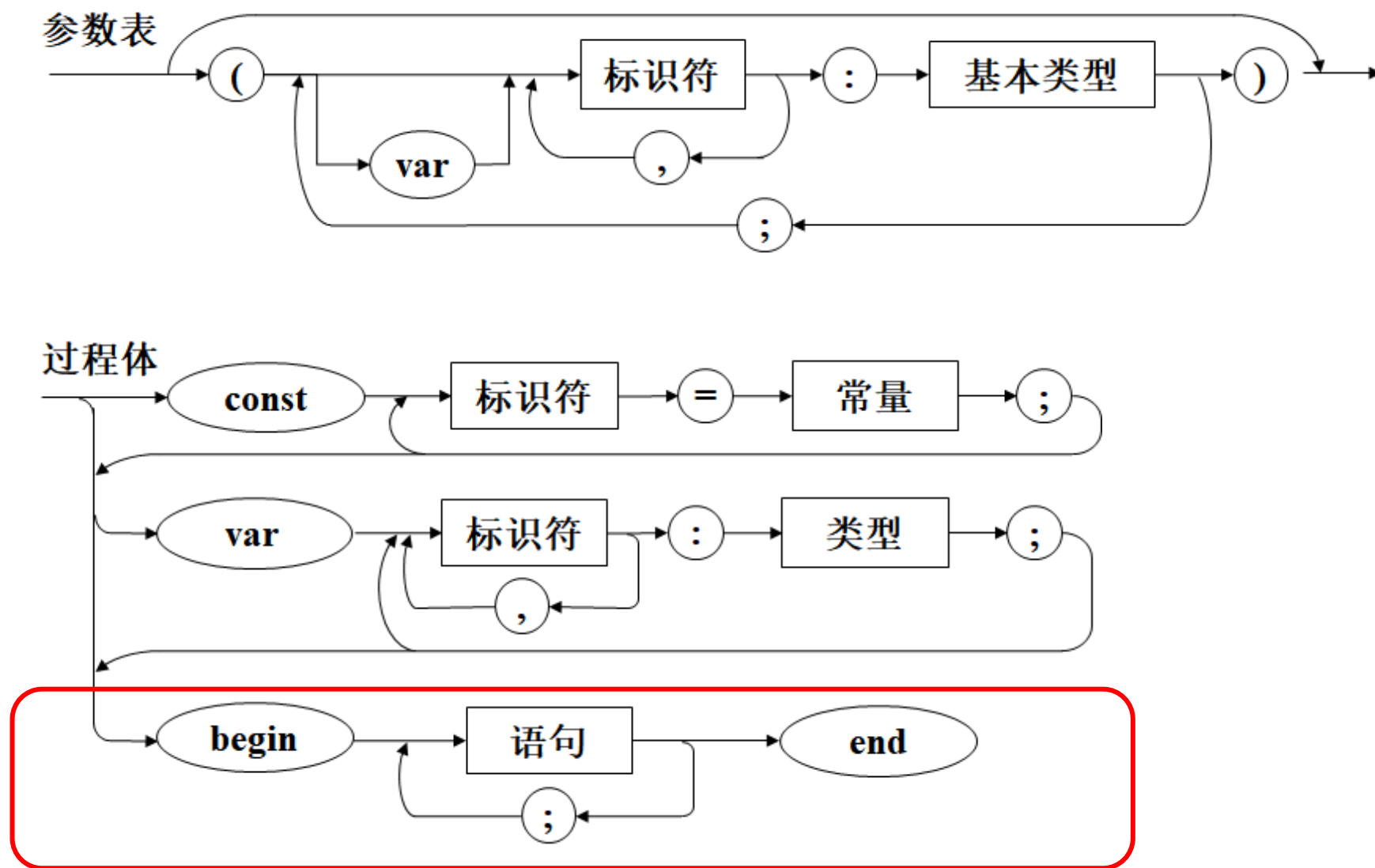
类型



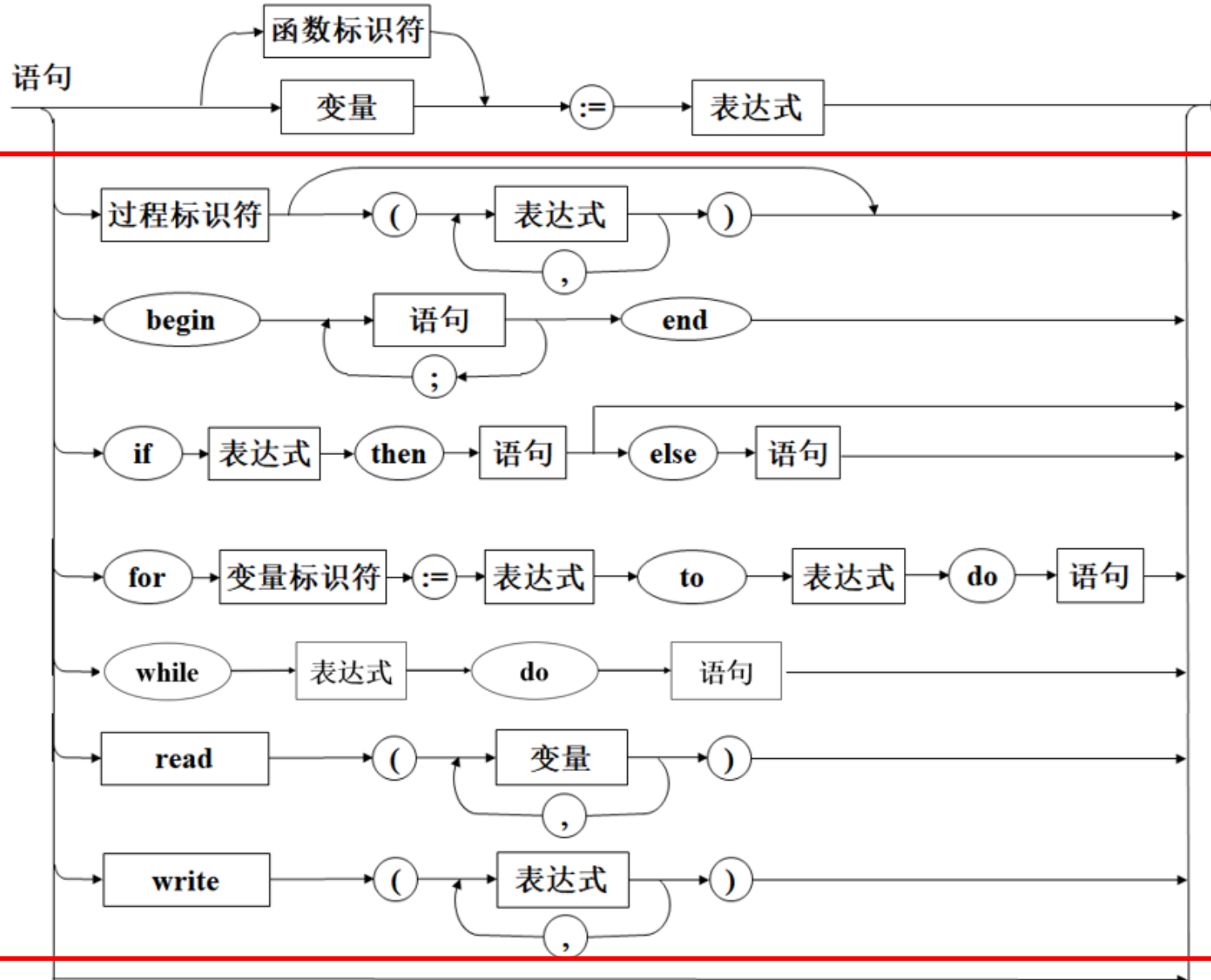
基本类型



# MiniPascal语言语法图

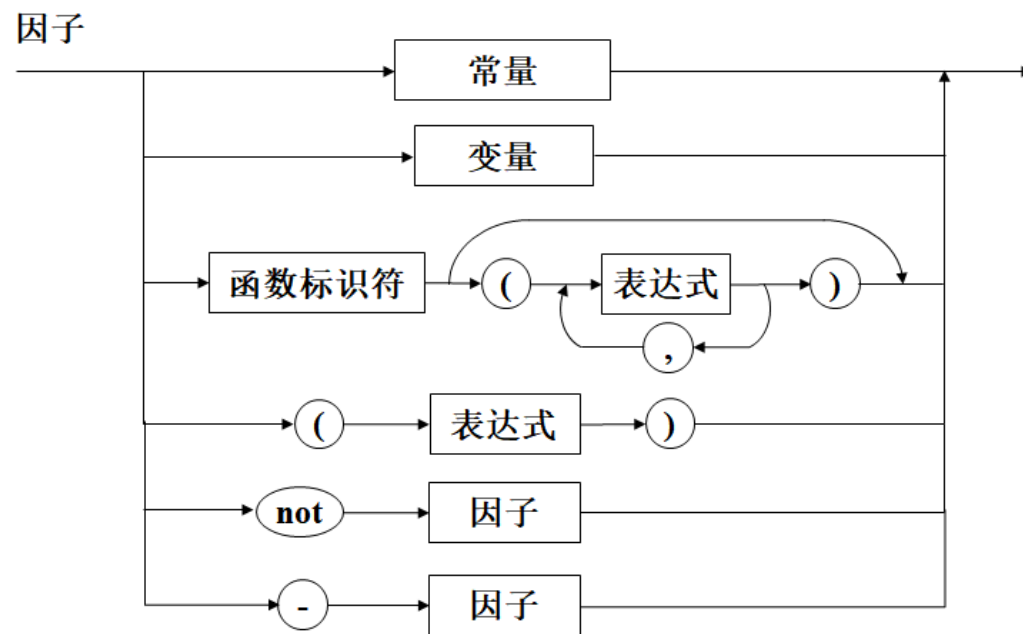
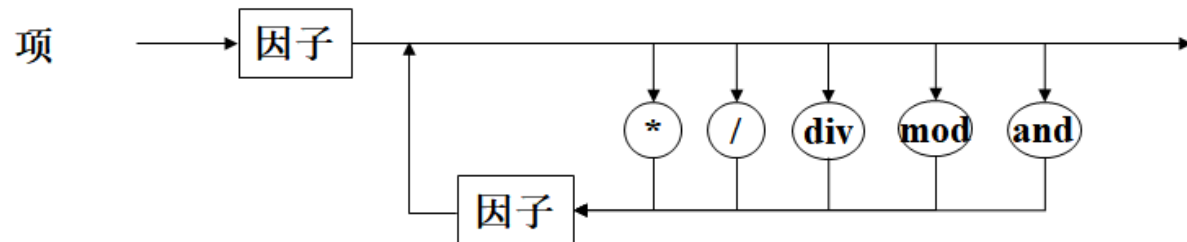
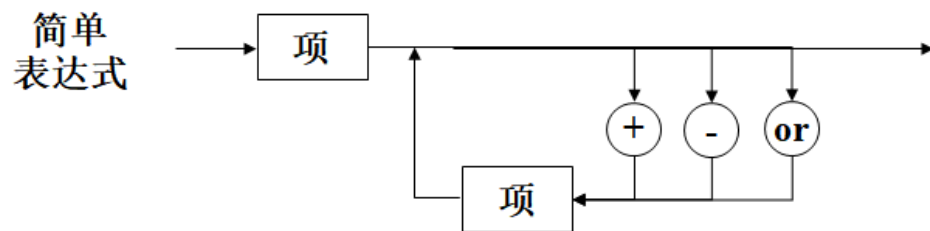
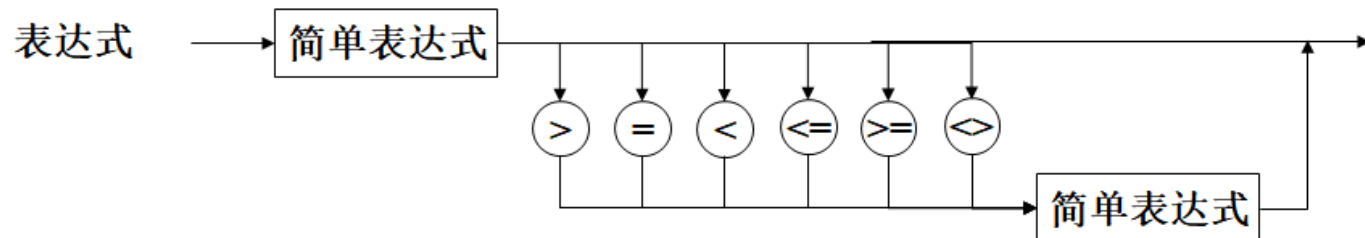
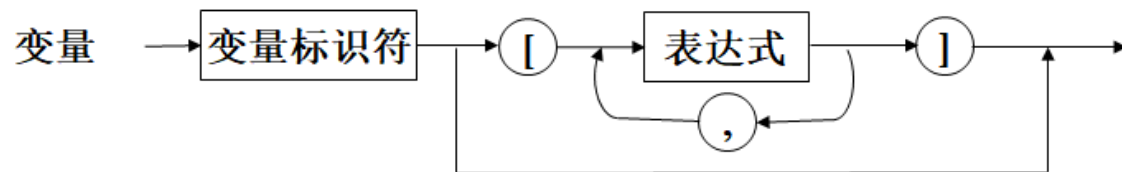


# MiniPascal语言语法图





# MiniPascal语言语法图



## 1.2 MiniPascal语言文法产生式

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

```
program sort(input, output)
program gcd(arg[2])
program example
```

# MiniPascal语言的文法产生式

- $const\_declarations \rightarrow \epsilon \mid \text{const } const\_declaration ;$
- $const\_declaration \rightarrow id = const\_value \mid const\_declaration ; id = const\_value$
- $const\_value \rightarrow + num \mid - num \mid num \mid 'letter'$

```
const a=1; b=-2;  
      ch='c';
```

- $var\_declarations \rightarrow \epsilon \mid \text{var } var\_declaration ;$
- $var\_declaration \rightarrow idlist : type \mid var\_declaration ; idlist : type$
- $type \rightarrow basic\_type \mid \text{array } [ period ] \text{ of } basic\_type$
- $basic\_type \rightarrow integer \mid real \mid boolean \mid char$
- $period \rightarrow digits .. digits \mid period , digits .. Digits$

```
var a, b: integer;  x, y: real;  
    ch: char;  
    mx: array [3..9] of integer;
```

# MiniPascal语言的文法产生式

- $subprogram\_declarations \rightarrow \epsilon \mid subprogram\_declarations \ subprogram \ ;$
- $subprogram \rightarrow subprogram\_head \ ; \ subprogram\_body$
- $subprogram\_head \rightarrow \text{procedure id formal\_parameter}$   
 $\mid \text{function id formal\_parameter} : basic\_type$
- $formal\_parameter \rightarrow \epsilon \mid ( \ parameter\_list \ )$
- $parameter\_list \rightarrow parameter \ / \ parameter\_list \ ; \ parameter$
- $parameter \rightarrow var\_parameter \mid value\_parameter$
- $var\_parameter \rightarrow var \ value\_parameter$
- $value\_parameter \rightarrow idlist : basic\_type$
- $subprogram\_body \rightarrow const\_declarations$   
 $\quad \quad \quad var\_declarations$   
 $\quad \quad \quad compound\_statement$
- $compound\_statement \rightarrow \text{begin statement\_list end}$
- $statement\_list \rightarrow statement \ / \ statement\_list \ ; \ statement$

```
function f(a, b: integer): integer;  
procedure p(x, y: real; var b, c: integer);
```

# MiniPascal语言的文法产生式

- $statement \rightarrow \varepsilon$ 
  - | *variable assignop expression*
  - | *func\_id assignop expression*
  - | *procedure\_call*
  - | *compound\_statement*
  - | *if expression then statement else\_part*
  - | *for id assignop expression to expression do statement*
  - | *while expression do statement*
  - | *read ( variable\_list )*
  - | *write ( expression\_list )*
- $variable\_list \rightarrow variable \quad / \quad variable\_list , variable$
- $variable \rightarrow id \ id\_varpart$
- $id\_varpart \rightarrow \varepsilon \quad | \quad [ expression\_list ]$

```
function gcd(a, b: integer): integer;  
begin  
    if b=0 then gcd:=a  
    else gcd:=gcd(b, a mod b)  
end;
```

# MiniPascal语言的文法产生式

- $procedure\_call \rightarrow id \mid id ( expression\_list )$
- $else\_part \rightarrow \varepsilon \mid else\ statement$
- $expression\_list \rightarrow expression \mid expression\_list , expression$
- $expression \rightarrow simple\_expression \mid simple\_expression\ relop\ simple\_expression$
- $simple\_expression \rightarrow term \mid simple\_expression\ addop\ term$
- $term \rightarrow factor \mid term\ mulop\ factor$
- $factor \rightarrow num \mid variable$ 
  - $\mid ( expression )$
  - $\mid id ( expression\_list )$
  - $\mid not\ factor$
  - $\mid uminus\ factor$

# 1.3 文法说明

- 该文法是一个LALR(1)文法。
  - 采用LR分析方法;
  - 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MiniPascal程序。包括:
  - 程序头: `program .....`
  - 全局常量的声明语句 (可选): `const .....`
  - 全局变量的声明语句 (可选): `var .....`
  - 过程和/或函数的定义 (可选): `procedure ..... / function .....`
  - 主程序体/复合语句: `begin ..... end`
- 过程和函数定义不允许嵌套。
- 复合语句允许嵌套。
- 过程和函数可以递归调用。
- 参数传递方式: 传值调用和引用调用 (传地址)

# 程序示例

```
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.
```



# 文法说明

- 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  $a$ 。

思考：

如何区分名字  $b$  是常量名、变量名、还是函数名？

## 1.4 关于词法的约定

- 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，其它单词间的空格符是可选的。
- 源程序中的注释
  - 形式：{.....}
  - 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
  - 以字母开头的字母数字串，不区分大小写。
  - 定义为：  
     $\text{letter} \rightarrow [\text{a-zA-Z}]$   
     $\text{digit} \rightarrow [0-9]$   
     $\text{id} \rightarrow \text{letter} ( \text{letter} | \text{digit} ) ^*$
  - 对其最大长度可以规定一个限制，比如 8 个字符。

# 关于词法的约定

## ■ 常数

- 整型常数、实型常数。

- 定义为：

$\text{digits} \rightarrow \text{digits digit} \mid \text{digit}$

$\text{optional\_fraction} \rightarrow . \text{digits} \mid \epsilon$

$\text{num} \rightarrow \text{digits optional\_fraction}$

- 关键字作为保留字，如 **program, var, procedure, begin, if, else, ...**

- 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 **:=**

# 1.5 课程设计建议

## ■ 详细的需求分析

要求：每个需求都必须是可以验证的。

### □ 词法分析

- 单词种类、单词构成、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？）

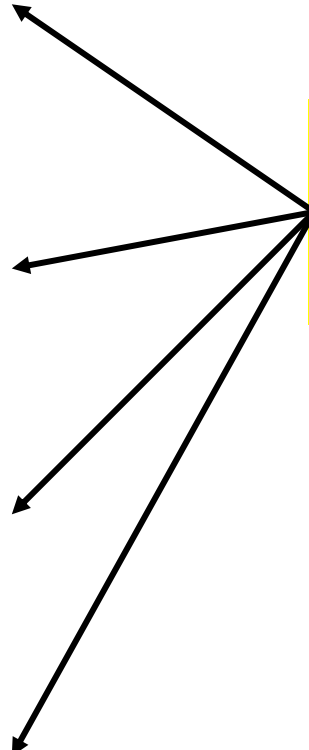
### □ 语义分析

- 类型、类型表示、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

## ■ 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

## ■ 符号表结构及其管理程序的设计

### □ 首先设计符号表的结构（逻辑结构和物理结构）

### □ 符号表内容

- 名字、类型（常量名及值？变量值存在？变量值？）
- 数组（维数、每维的上下界）
- 函数（参数个数、参数类型、传递方式）

### □ 管理程序

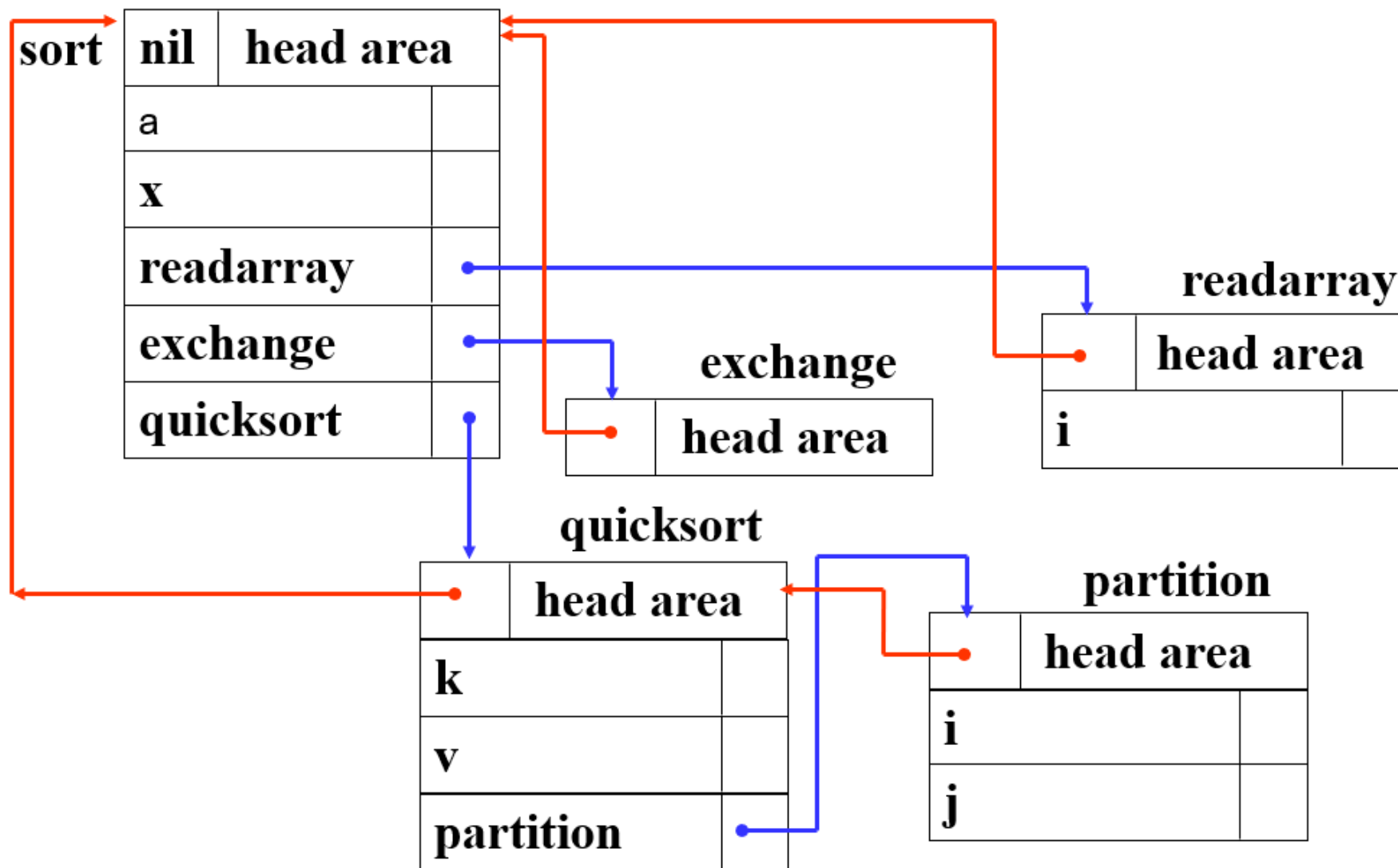
- 查找操作：按给定的名字查表。
- 插入操作：在表中建立新的一行。
- 定位操作：创建符号子表，确定一个新作用域的起点。
- 重定位操作：从符号表中“删除”局部于当前过程的所有名字，退出当前作用域。

# 参考：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;
```

```
peocedure 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 }
```

# 参考：符号表的逻辑结构





## ■ 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。
- 实现方法
  - 手工编码
  - Lex自动生成

# 课程设计建议

## ■ 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。（程序、手工？）

构造预测分析表。（程序、手工？）

编码实现预测分析程序。

➤ LR分析方法

构造LALR(1)分析表。（程序、手工？）

编码实现LR分析程序。

### □ 用YACC工具实现（推荐）。

# 课程设计建议

## ■ 语义动作和翻译程序的详细设计

- 按照语言的语义规则设计语义子程序。
- 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。
- 注意：
  - 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MiniPascal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

## ■ 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MiniPascal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。
- 分析错误类型、错误原因、错误处理
- 打印错误位置、错误信息。
- 如有可能，应对错误进行恢复，并继续进行编译。

## 1.6 测试

- 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
  - 设计含有各类错误的测试用例源程序。
  - 用所实现的编译器对测试用例程序进行编译。
  - 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

```
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.  
end.
```

## ■ 用MiniPascal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。

# 1.7 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。

# 1.8 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



## 2. 课程设计教学安排

### ■ 分组

- 6±1人一组，自由组合。
- 每组推荐一名项目组长
  - 主导总体设计、系统测试
  - 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

### ■ 手工编码，分工建议

- 词法分析+编写测试用例：1人
- 语法分析+语义分析：3人
- 代码生成：2人

### ■ 使用生成工具，分工建议

- 词法分析+语法分析：1人
- 编写测试用例：1人
- 语义分析：3人
- 代码生成：1人

### ■ 要求

- 每个人的任务明确，相对独立。
- 每个成员须承担部分功能模块设计开发工作。完成相应模块的
  - 详细设计，准备设计文档，编码
  - 单元测试，细化完善测试用例，测试报告。
- 组长统筹安排软件的集成测试。

# 教学安排

## ■ 集中授课

- 第1周：课程说明
- 第4周：交流
- 第7周：中期检查
- 第12周：验收

## ■ 第2周

### □ 课程设计任务

- 组建课程设计团队；
- 理解课程设计目标和任务；
- 查阅资料，掌握源语言的语法和语义规则

### □ 自学要求

- 分析比较不同的实现方法；
- 需求分析方法及描述规范。

## ■ 第3周

### □ 课程设计任务

- 初步完成需求分析报告；
- 设计测试用例，制定测试计划；
- 开发方法比较和选择。

### □ 自学要求

- 确定并熟悉开发环境和工具；
- 测试用例设计方法；
- 总体设计方法及描述规范。

## ■ 第4周

### □ 课程设计任务

- 完善需求分析报告；
- 完善测试计划；
- 完成总体设计报告；
- 完成团队成员分工。

### □ 自学要求

- 详细设计方法及描述规范；
- 熟悉开发环境及工具。

# 教学安排

## ■ 第5~7周

### □ 课程设计任务

- 进一步完善总体设计;
- 完成模块的详细设计。

### □ 自学要求

- 熟悉开发环境及工具

## ■ 第8-9周

### □ 课程设计任务

- 完成模块编码;
- 设计测试用例;
- 完成单元测试

### □ 第8~9周自学要求

- 单元测试技术
- 测试用例设计方法
- 程序调试方法

## ■ 第10-11周

### □ 课程设计任务

- 完成系统测试、撰写测试报告
- 撰写软件使用说明
- 完成课程设计报告
- 准备验收资料

### □ 自学要求

- 集成测试方法
- 系统调试方法

## ■ 第12周

### □ 验收

# 验收安排

## ■ 验收时间、地点及要求

- 时间：第12周
- 地点：（另行通知）
- 要求：
  - 项目小组成员全部参加，否则，不予验收。
  - 验收前提交相关资料。
  - 没有按时参加验收的同学，按“缺考”计。

## ■ 验收前需提交以下资料（电子版）：

1. 课程设计表格（分组表）（附件1，填写完整）
2. 课程设计报告（详细报告，见附件2）
3. 程序及其使用说明，包括：
  - （1）源程序
  - （2）可运行程序
  - （3）测试用例程序
  - （4）程序使用说明

## ■ 验收时：

- 答辩 PPT
- 填写完整并打印好的附件3《验收登记表》

## ■ 如发现抄袭，一律按0分计。

# 验收过程

- 讲解符号表的内容和结构、在符号表上的操作（查找、插入、定位、重定位）
- 词法分析：翻译表、注释的处理、与语法分析器的接口等
- 语法分析：实现方法，能检测哪些语法错误？输出？
- 语义分析：实现方法，支持哪些类型？实现了哪些类型检查？  
表达式、数组？参数？函数/过程调用？语句？
- 代码生成：输入？采用的技术？算法思想？
- 错误处理与恢复策略
- 运行程序，演示测试用例的编译过程和结果  
要求：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。
- 查看设计文档

# 成绩评定

## ■ 小组基准成绩评定

### □ 软件设计验收答辩情况：70%

➤ 平时表现 20%

➤ 验收答辩 80%

✓ 完成的功能

✓ 讲解、运行过程、答辩

### □ 设计报告：30%

➤ 正确性、完整性

➤ 与程序的一致性

## ■ 组与组比较

### □ 质

### □ 量

➤ 扩充语言结构，如  
增加类型（如：record）、  
语句（如：while）等。

➤ 缩减语言结构，比如去掉参数的引用调用。

## ■ 组内成员成绩

□ 视成员完成工作情况，上下浮动。

□ 各组提供成员的贡献率/排名

## ■ Flex

□ [flex开源项目](https://github.com/westes/flex)      <https://github.com/westes/flex>

□ [flex使用手册](https://www.cs.princeton.edu/~appel/modern/c/software/flex/flex_toc.html)

[https://www.cs.princeton.edu/~appel/modern/c/software/flex/flex\\_toc.html](https://www.cs.princeton.edu/~appel/modern/c/software/flex/flex_toc.html)

## ■ Bison

□ [bison官方手册](https://www.gnu.org/software/bison/manual/)      <https://www.gnu.org/software/bison/manual/>

□ [bison的源码](https://ftp.gnu.org/gnu/bison/)      <https://ftp.gnu.org/gnu/bison/>

# 编译工具

## ■ Flexc++

- [flexcpp](https://gitlab.com/fbb-git/flexcpp) 开源项目 `https://gitlab.com/fbb-git/flexcpp`
- [flexc++](https://fbb-git.gitlab.io/flexcpp/manual/flexc++.html) 用户手册 `https://fbb-git.gitlab.io/flexcpp/manual/flexc++.html`

## ■ Bisonc++

- [bisoncpp](https://gitlab.com/fbb-git/bisoncpp) 开源项目 `https://gitlab.com/fbb-git/bisoncpp`
- [bisonc++](https://fbb-git.gitlab.io/bisoncpp/manual/bisonc++.html) 用户手册 `https://fbb-git.gitlab.io/bisoncpp/manual/bisonc++.html`



## ■ Antlr4

- [antlr官网](https://wwwantlr.org/) <https://wwwantlr.org/>
- [antlr开源项目](https://github.com/antlr/antlr4) <https://github.com/antlr/antlr4>

## ■ LLVM

- [llvm官网](https://llvm.org/) <https://llvm.org/>
- [llvm手册](https://llvm.org/docs/) <https://llvm.org/docs/>
- [LLVM开源项目](https://github.com/llvm/llvm-project) <https://github.com/llvm/llvm-project>

- 全国大学生计算机系统能力大赛
- <https://compiler.educg.net>



主动参与

认真参与

团结协作

携手前行

加油 ❤️

