编译原理

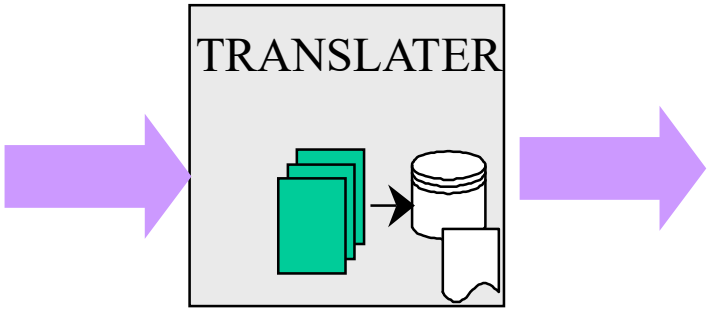
林志毅
广东工业大学计算机学院
lzy291@gdut.edu.cn

教学安排与要求——▸▸为什么要学习编译原理？

C语言程序

```
void main( )
{   int x,y,z;
    x=3;
    y=2;
    z=x+y;
}
```

?



汇编语言程序

```
……
mov ax,3
mov x,ax
mov ax,2
mov y,ax
mov ax,x
mov bx,y
add ax,bx
mov z,ax
……
```

300
302
304
306
308
……

教学安排与要求——▸▸为什么要学习编译原理？

■ 有助于深刻理解和正确使用程序设计语言，加深对高级语言程序执行过程的理解。

■ 有助于加深对整个计算机系统的理解。(见备注)

■ 设计开发编译程序的软件技术同样可以用于其他软件的设计开发。**程序格式化工具：使程序呈现清晰的结构**(Source insight, Editplus…) **一般的软件设计：文本编辑器、自动排版、模式识别、程序自动验证、程序自动调试、高级语言之间的转换工具、通信协议转换……**

■ **交叉编译技术：如嵌入式应用**

■ **硬件描述语言及其编译技术：如芯片设计**

■ **为计算机分析和理解自然语言提供参考**

■有利于进行**软件保护**（反编译、反汇编），编译原理在**反病毒技术中**同样得到了研究与应用。

教学安排与要求——《编译原理》课程在计算机科学中的重要地位

(1) 学习编程最初是学习一门高级语言，C或Java，掌握编写一些简单程序的方法；

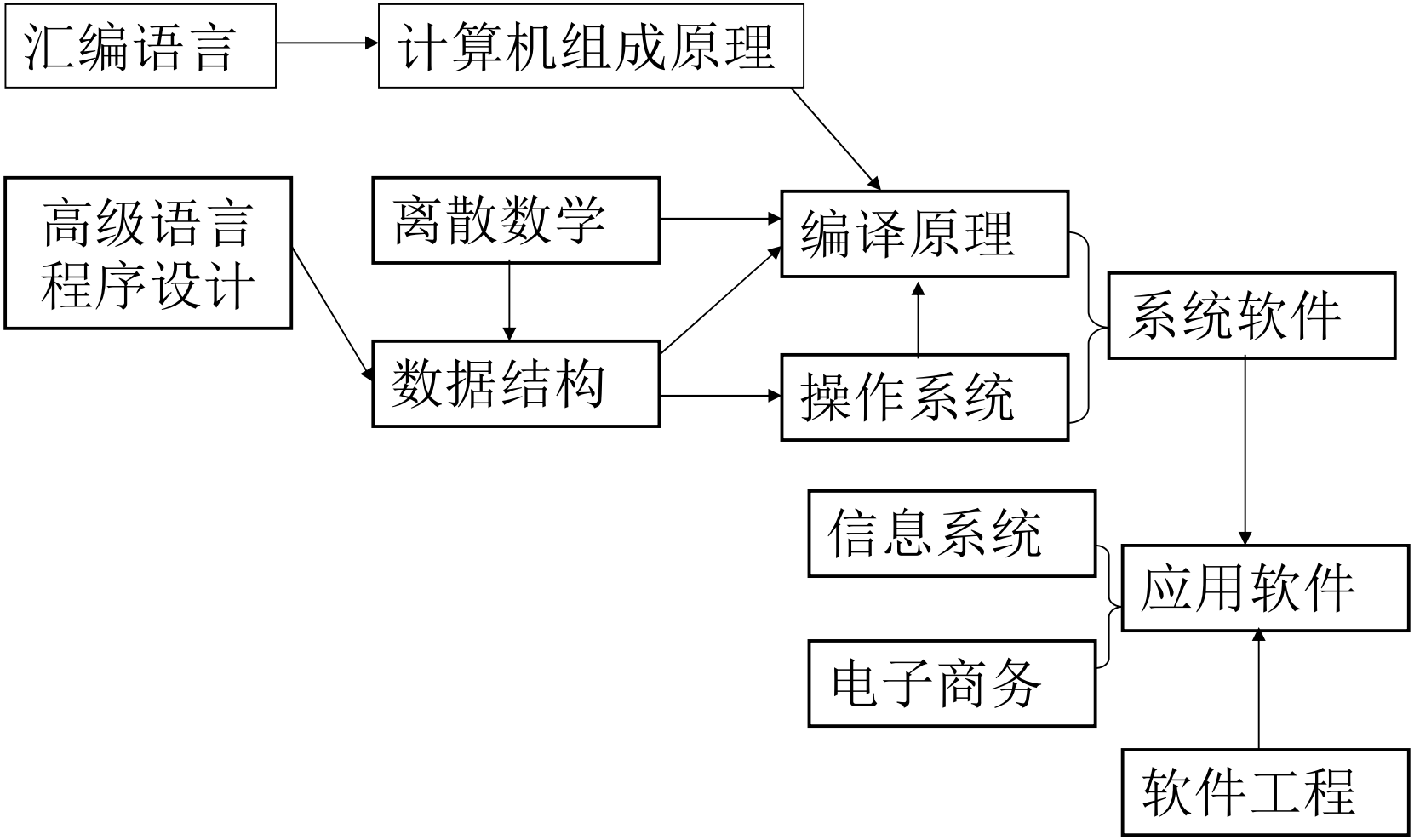
(2) 学习数据结构，建立“算法”的概念，对编程有更深入的理解。遇到问题的时候，能够寻找相应的数据结构模型，设计适当的算法来解决问题；

(3) 学习汇编语言，这门课程是我们真正深入了解计算机内部工作的第一门课程。通过学习了解汇编语言如何变为机器语言，如何对应于一条指令；

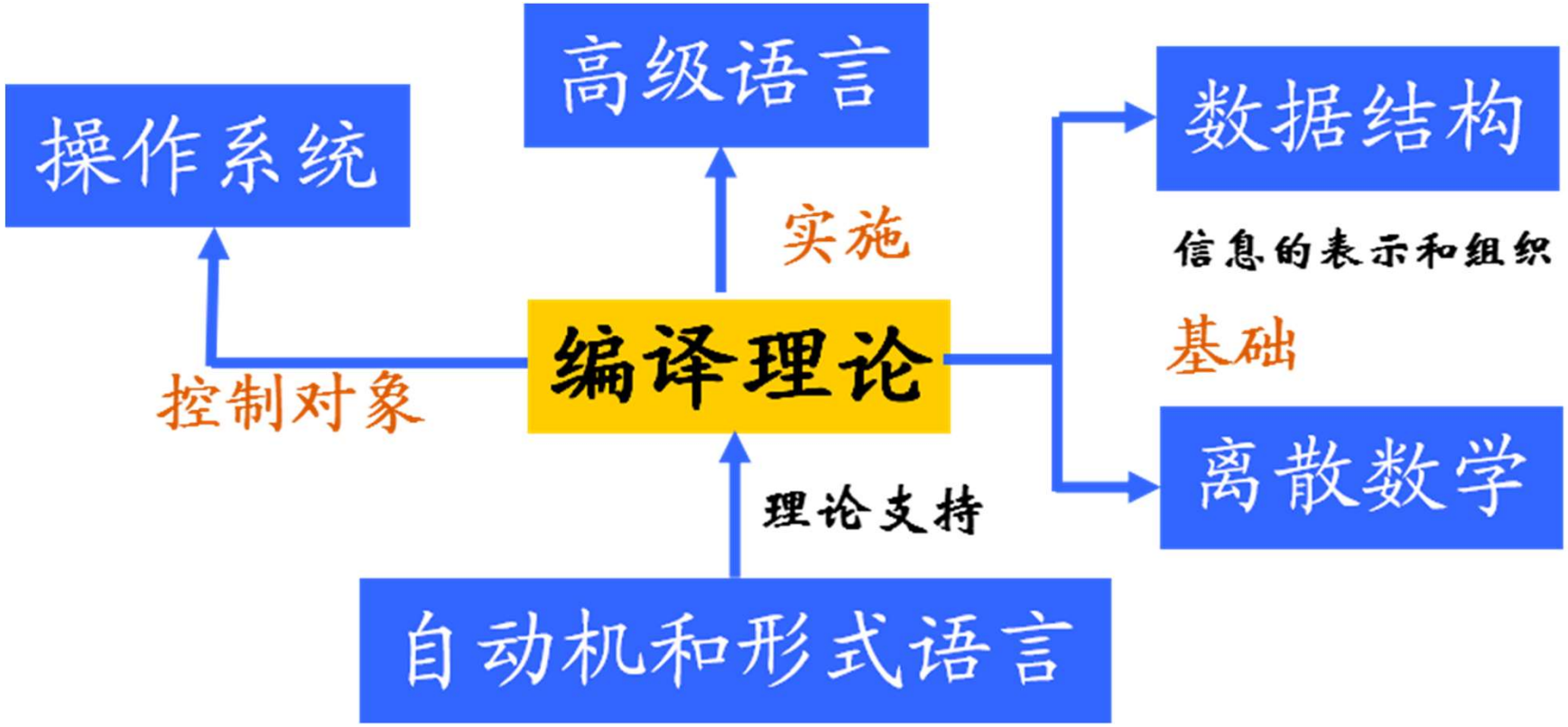
(4) 计算机组成原理课程的学习使我们了解到计算机的硬件组成，以及机器指令程序如何在计算机中运行的过程。

(5) 编译原理课程帮助我们了解高级语言程序转换成机器指令程序的过程。可以帮助我们更深刻地理解高级语言程序运行的内部机制。

教学安排与要求——《编译原理》课程在计算机科学中的重要地位



教学安排与要求——《编译原理》课程与其它课程的关系



教学安排与要求——▸▸ 先修课程

要求先学习以下课程

1.程序设计语言

2.算法与数据结构：栈分配、堆分配、静态分配等各种存储分配方式。线性表、二叉查找树、哈希表等多种数据结构。

3.离散数学：集合论与数理逻辑是进一步学习形式语言与自动机理论的数学基础。

最好学习过或同时学习以下课程

1.软件工程：掌握大型程序设计以及工程化的软件生产方法。

2.形式语言与自动机：相当于本课程中词法分析与语法分析的理论基础。

教学安排与要求——▸▸ 课程任务

■任务

■通过学习该课程，掌握编译的基本理论、常用的编译技术，了解编译过程及编译系统结构和机理。能运用所学技术解决实际问题，能独立编写一个小型编译系统。

■此外，通过学习编译原理可以更好地理解程序语言的内部机制,从而更好地理解和运用程序设计语言。能运用编译程序构造的原理和技术完成相关软件工具的设计和开发工作。

教学安排与要求——▸▸ 教材及主要参考资料

教材：编译原理（第3版），张素琴，吕映芝，清华大学出版社。

参考：

1. 编译原理实践及应用，黄贤英，清华大学出版社。
2. 编译原理，陈火旺，国防工业出版社。
3. 程序设计语言编译方法，肖军模，大连理工大学出版社。
4. 编译原理（龙书），alfred V.Aho等著，李建中等译，人民邮电出版社。
5. 《现代编译程序设计》，D.Grune等. 人民邮电出版社。
6. ……

教学安排与要求—▸▸ 课程内容

教材内容

- 第1章 引论
- 第2章 文法和语言
- 第3章 词法分析
- 第4章 自顶向下语法分析
- 第5章 自底向上语法分析
- 第6章 LR分析
- 第7章 语法制导的语义计算
- 第8章 静态语义分析和中间代码生成
- 第9章 运行时存储组织
- 第10章 代码优化和目标代码生成

学时：授课48 上机8 课程设计一周

教学安排与要求——▸▸ 课程内容

内容: 除以下说明外，按教材目录顺序

(1) 第一轮课堂和实验教学与附录A的PL/0编译程序源码分析紧密结合，要求读懂源程序，完成若干扩充编译器的实验，为期末课程设计打基础。

(2) 第6章为第二轮的内容。（若时间来不及，第6章有可能也有部分不讲

(3) 第§5.2-3, §9.5, §10.2, §10.4, §11.2-3和12章的内容不要求。

教学安排与要求——▸▸ 本课程的特点和学习方法

特点:

- 本课程的理论性很强。
- 涉及的算法较复杂，要深入地理解这些算法较困难。
- 整个编译程序的构造方法非常精妙，就像一部走时精确的时钟，很多齿轮、部件协调地运转，以驱动指针准确地旋转；编译程序也是如此，一边扫描源程序，一边经过各个部件的运算，准确地输出为目标语言。
- 编译原理课程各个部分之间的独立性很强，包括词法分析、语法分析、语义分析、存储的组织与分配、中间语言、语法制导翻译、代码生成与优化这几大部分。词法分析和语法分析是其中的重点，语义分析也是难点，需要掌握比较复杂的算法逻辑；其他部分相对来说知识性更强一些。各部分之间的方法也互相独立，在学习时，便于逐个击破。
- 考试考查的内容相对来说是很稳定的，绝大多数题目的解法都非常机械。

教学安排与要求——▸▸ 本课程的特点和学习方法

学习方法:

•尽可能地掌握编译原理的思想，要站得高一点，尽可能理解算法的思想，而不是背固定的算法。

•很多题目的解法比较固定，要熟练掌握相应的具体方法。

•多做习题，牢记重要算法。

• 一边学习，一边总结，关键是找差异：同一问题可以用多种方法来求解，不同方法适用于不同的文法，对文法的限制和要求，相应的表格的构造、使用等，各个方面的差异都要关注。

•亲自动手实现书上的一些算法,完成实验指导书上给出的一个简单的编译程序，或者编译程序的一部分，这样能更灵活地掌握编译程序构造的精髓。

第1章 概述 —▸▸

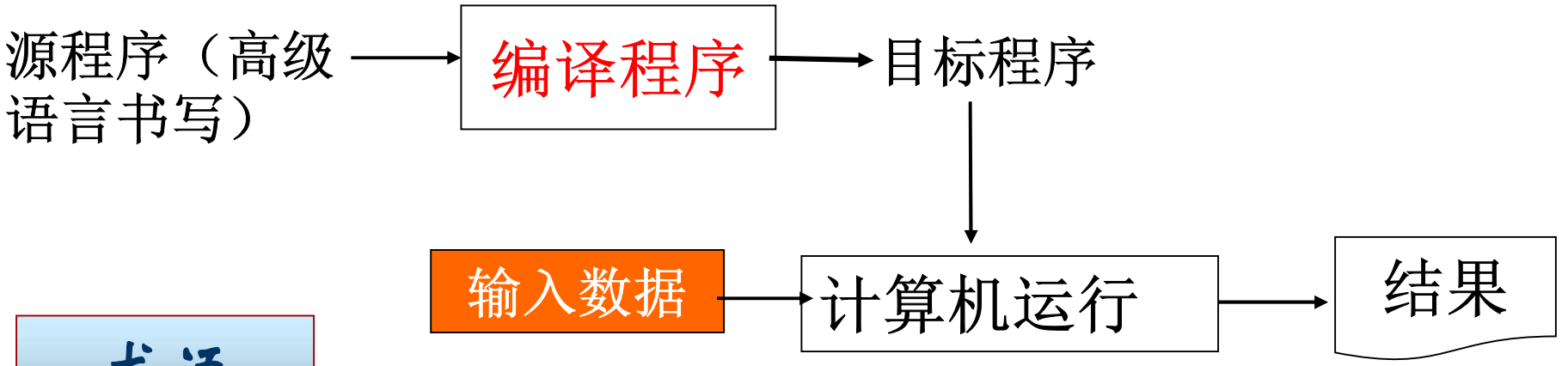
目录

编译程序(compiler)

将高级程序设计语言程序翻译成逻辑上等价的低级语言(汇编语言,机器语言)程序的翻译程序。

第1章 概述 —▸▸ 1.1 什么是编译程序

功能



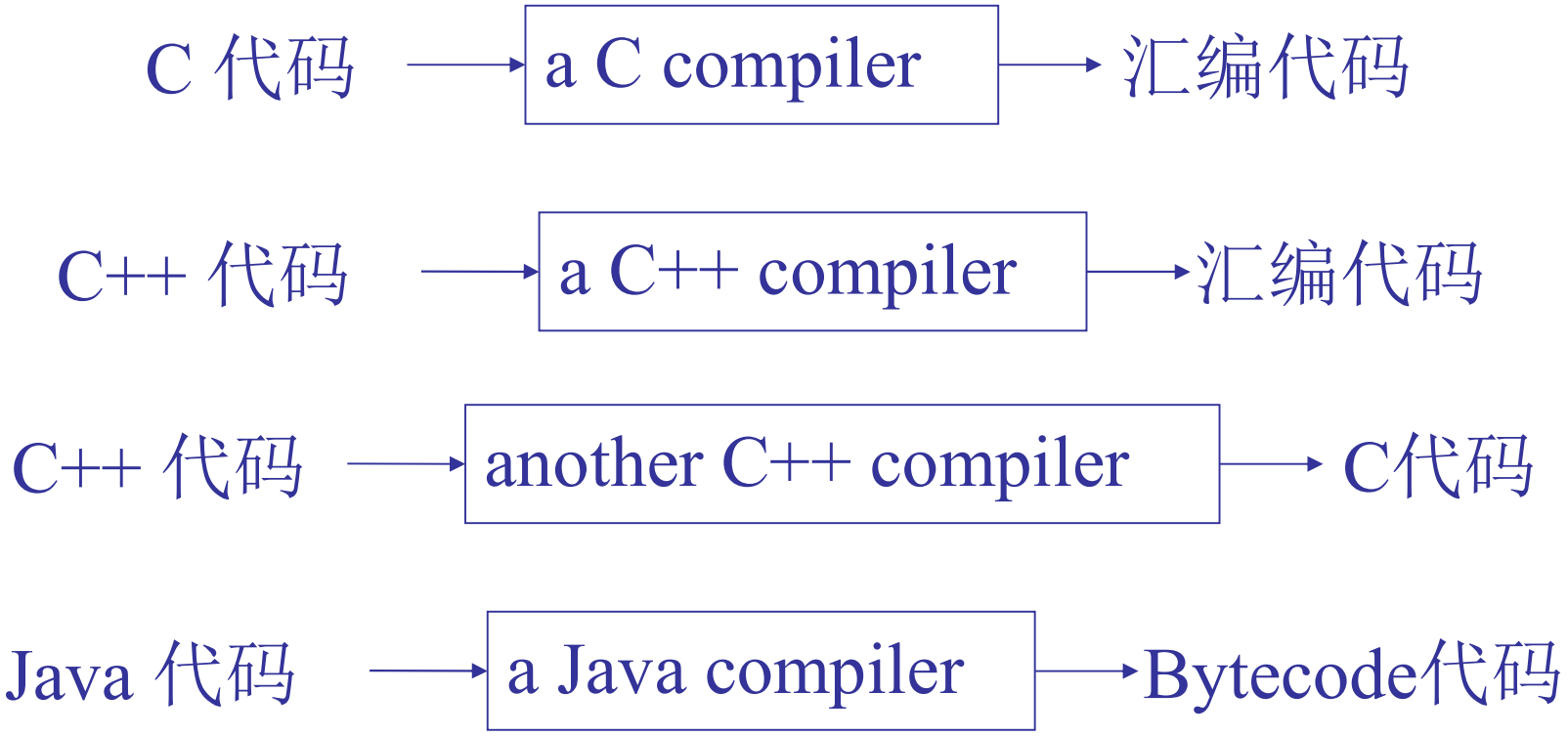
术语

编译程序的源语言 S (源程序)

编译程序的目标语言 O或T (目标程序)

编译程序的实现语言 I

✧ 编译程序通常是从较高级语言的程序翻译至较低级语言的程序，如



什么是编译程序

- 传统的编译程序
  - 源语言通常为高级语言（*High-Level Programming Languages*）

    Fortran, Algol, C, Pascal, Ada, C++, Java, Lisp, Prolog, Python…
  - 目标语言通常为机器级语言（*Machine-Level Languages*）或较低级的虚拟机语言

    汇编语言（*Assembly Languages*）

    机器语言（*Machine Languages*）

    Bytecode（*Java* 虚拟机语言）

单选题 2分

设置

编译程序是对( )

A 汇编程序的翻译

B 高级语言程序的解释执行

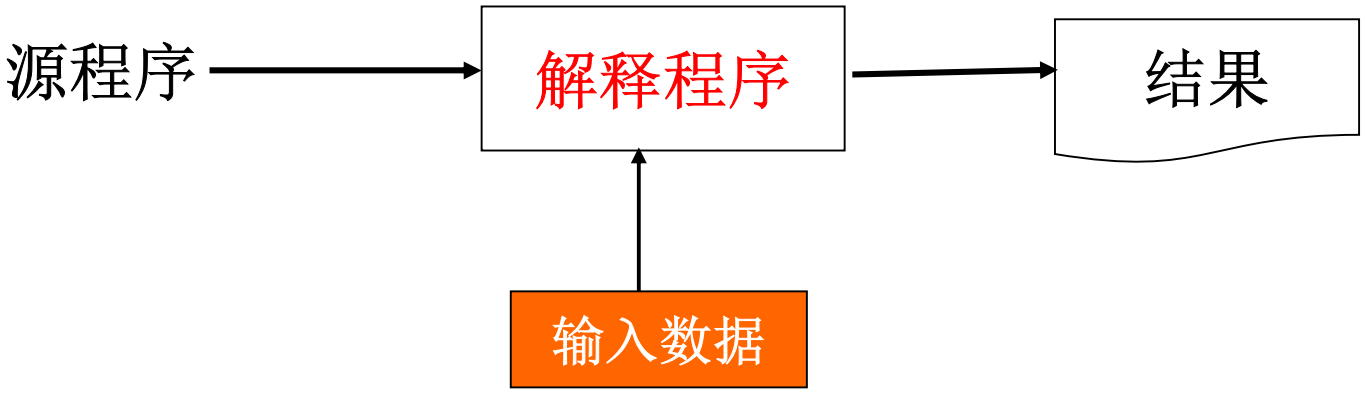
C 机器语言的执行

D 高级语言的翻译

提交

解释程序(Interpreter)

- 将高级程序设计语言写的源程序作为输入，边解释边执行源程序本身，而不产生目标程序的翻译程序。
- 功能：



第1章 概述 —▸▸ 1.1 什么是编译程序

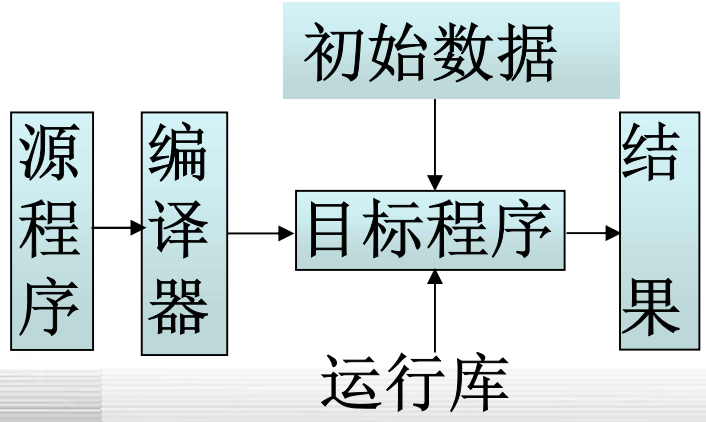
编译程序和解释程序比较

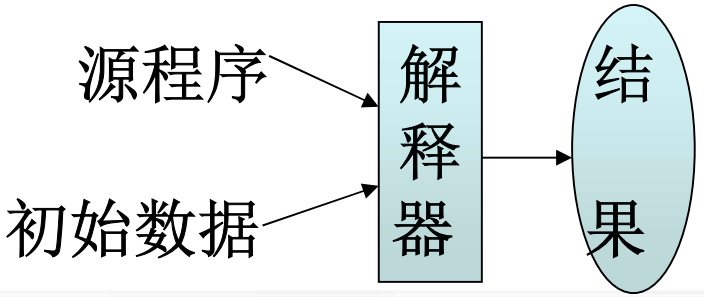
编　译（笔译）

产生目标程序，可多次执行，
优化较充分，执行效率高。
但编译器本身较大，
较复杂，不便人机对话
源语言：**Pascal**，C，
FORTRAN等

解　释（口译）

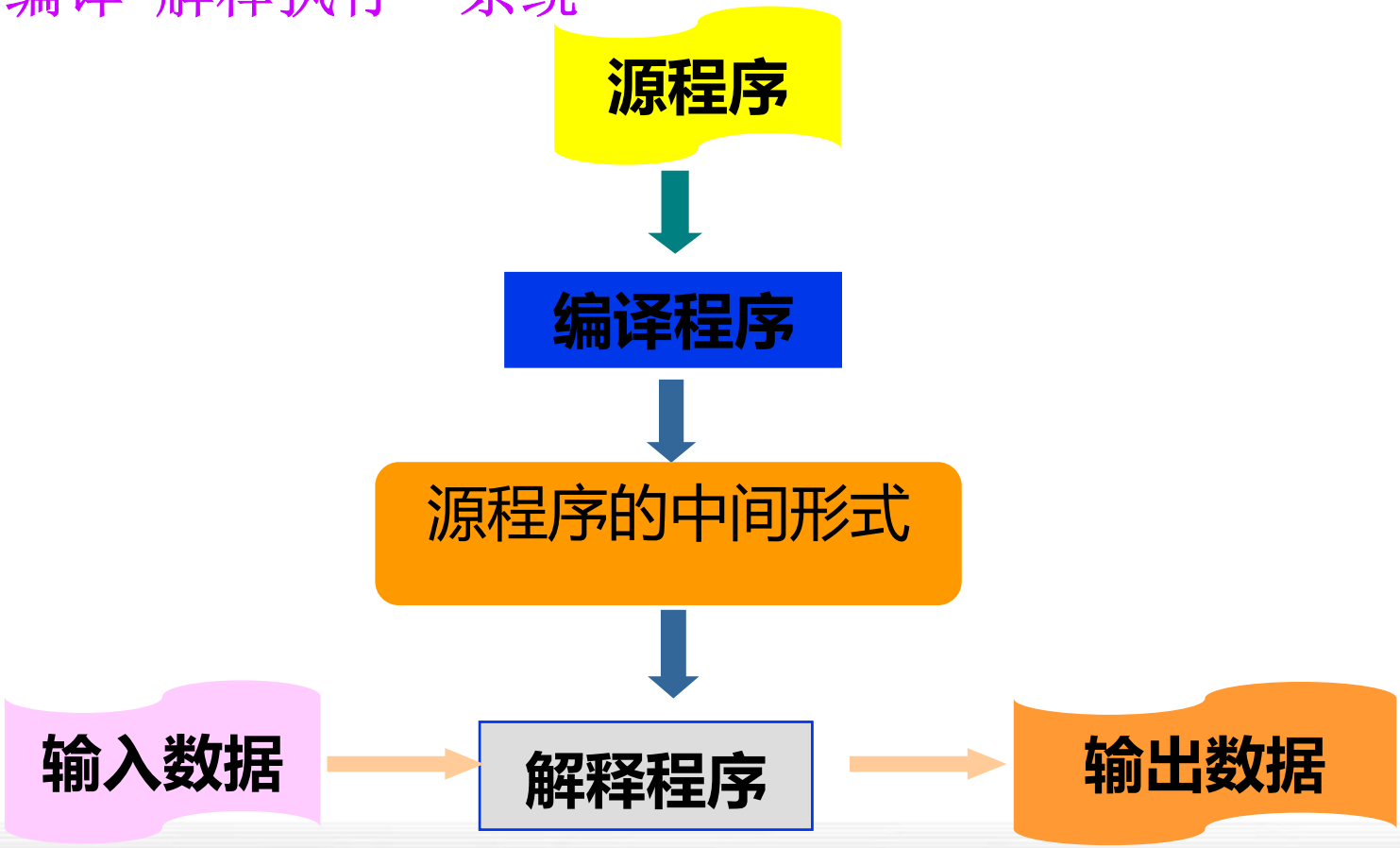
边解释边执行，不产生目标程序，
优化不充分，总体效率较低。
但解释器本身较小，
较简单，便于人机对话
源语言：**BASIC**，**LISP**等





第1章 概述 —▸▸ 1.1 什么是编译程序

- “编译-解释执行”系统



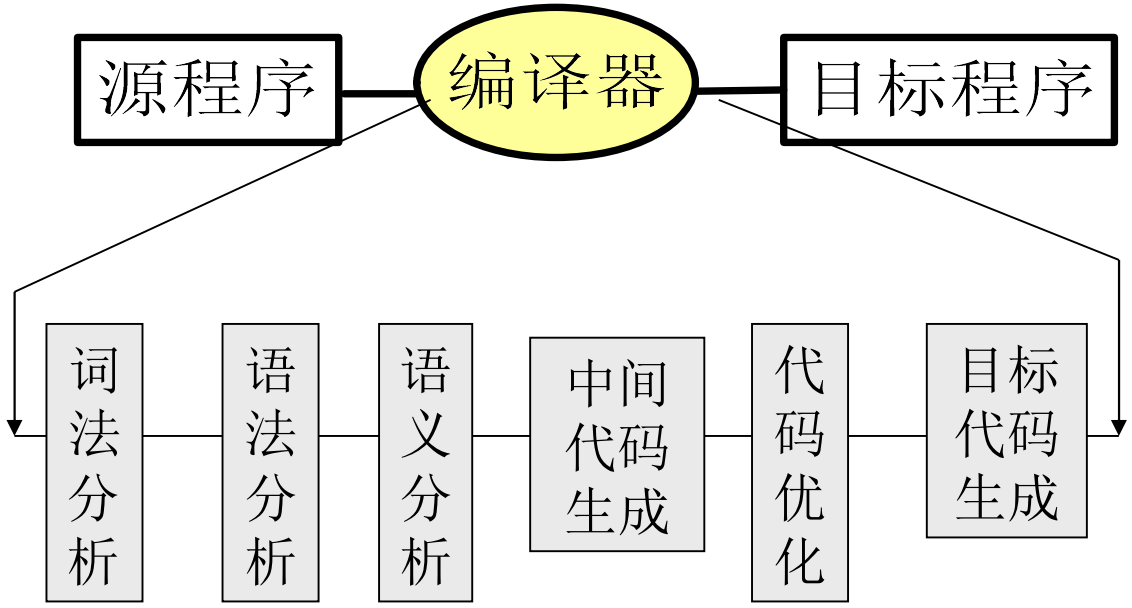
术语

- 编译程序(compiler)
- 编译程序的源语言(源程序)
  (source language / program)
- 编译程序的目标语言(目标程序)
  (object or target language / program)
- 编译程序的实现语言
  (implementation language)
- 翻译程序(language translator)

编译的逻辑过程

逻辑上分六个阶段：词法分析、语法分析、语义分析、中间代码生成、代码优化、目标代码生成。

每个阶段把源程序从一种表示形式转换成另一种表示形式。



单选题 2分

设置

编译原理六个阶段中第二个阶段是

A 词法分析

B 表格管理

C 语法分析

D 语义分析

提交

词法分析（扫描器 Scanner）

字符序列 ——▸ 单词序列

基本任务：从左到右扫描、分解源程序，识别出每个单词（基本字、标识符、常数、运算符、界限符），并给予种别（属性）和内部形式（值），构成单词的内部表示。

附加任务：a、滤掉空格 b.查填符号表 c.输出相应的错误信息等。

词法分析的工作主要依据语言的词法规则，描述词法规则的有效工具是正规式和有限自动机。（第2章、第3章）

词法分析（扫描器 Scanner）

position := initial + rate * 60;

单词类型	单词值
标识符1(id1)	position
算符(赋值)	:=
标识符2(id2)	initial
算符(加)	+
标识符3(id3)	rate
算符(乘)	*
整数(int1)	60
分号	;

词法分析（扫描器 Scanner）

单词	类型	内部形式
position	标识符	id_1
:=	界符	:=
initial	标识符	id_2
+	算符	+
rate	标识符	id_3
*	算符	*
60	常数	int_1
;	界符	;

术语

- 词法分析**(lexical analysis or scanning)**

  --The stream of characters making up a source program is read from left to right and grouped into tokens, which are sequences of characters that have a collective meaning.(从左到右读字符流的源程序、 识别（拼）单词。**)**

- 单词**---token(**具有独立意义的基本语法单位。 **)**
- 保留字**---reserved word(**具有特殊规定的意义，不允许用户将它们作为别用，是用户定义标识符的禁区。**)**
  标识符 **---identifier(user-defined name)**用来表示程序、过程、函数、类型、常量和变量等名称的符号。

语法分析（分析器 Analyzer）

- 在词法分析的基础上，根据语言的语法规则（文法规则），把单词符号串分解成各类语法单位，如“短语”、“句子”、“程序段”和“程序”。
- 通过语法分解，确定整个输入串是否构成一个语法上正确的程序. 例: 符号串 X+0.168*Y, 经语法分析就可识别出这个字符串属于算术表达式。 Y:= X+0.168*Y;
- 语法分析所依循的是语言的语法规则（表示语法规则的工具是上下文无关文法,用下推自动机实现），用产生式描述。（第2、4、5、6章）

- **功能**：层次分析。**依据**源程序的**语法规则**把源程序的单词序列组成语法短语(表示成语法树)。

position := initial + rate * 60 ;

语句分隔符

规则： <赋值语句>::=<标识符>“:=”<表达式

<表达式>::=<表达式>“+”<表达式>

<表达式>::=<表达式>“*”<表达式>

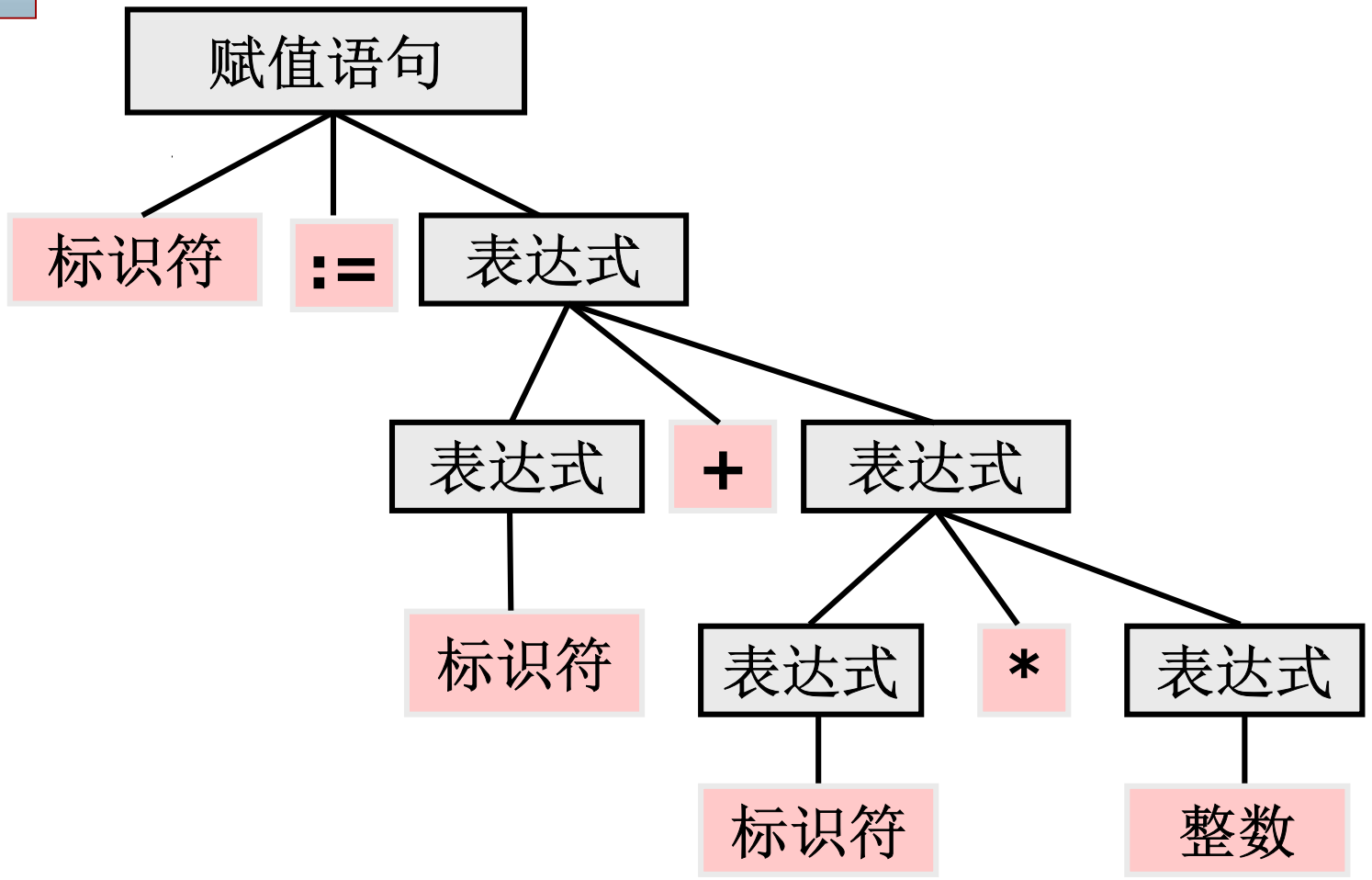
<表达式>::=“(”<表达式>“)”

<表达式>::=<标识符>

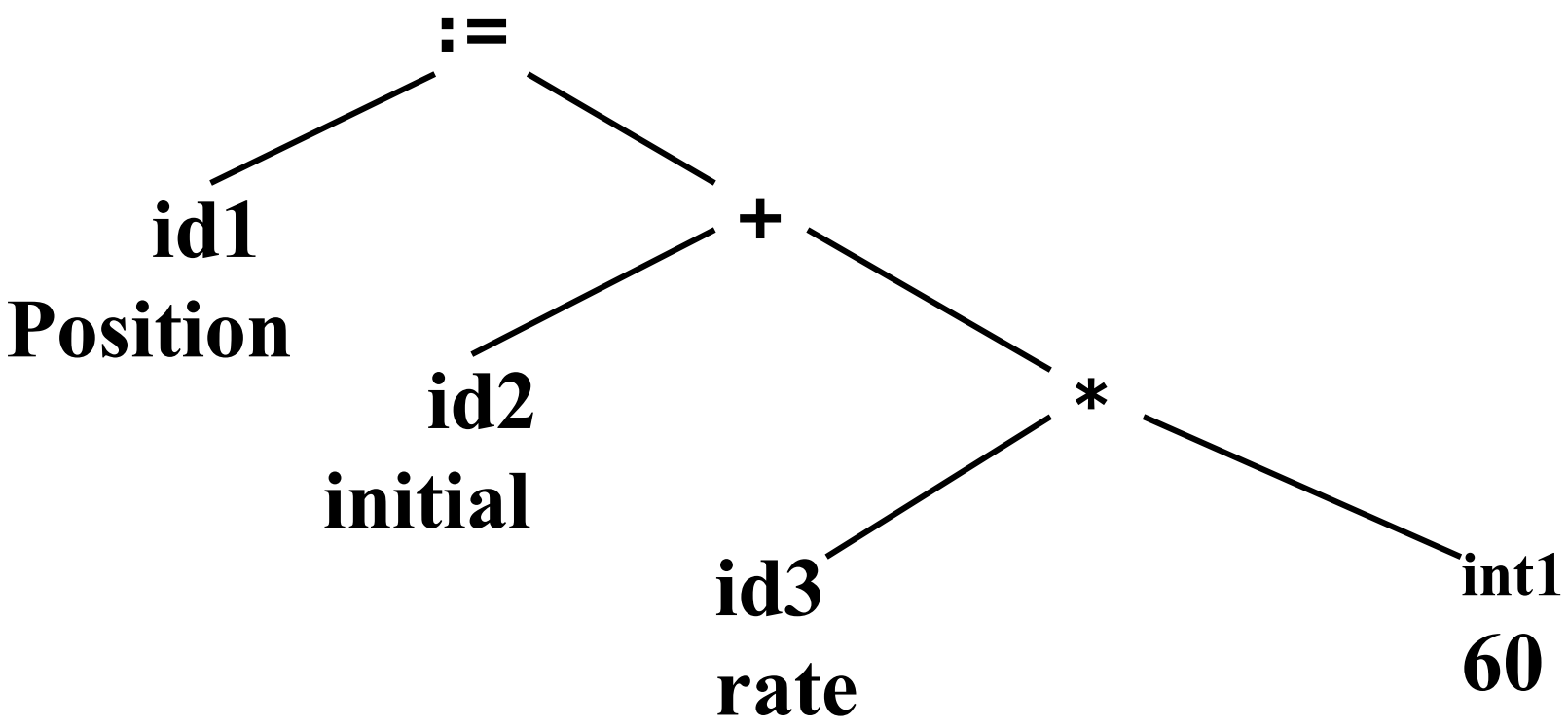
<表达式>::=<整数>

<表达式>::=<实数>

语法树



id1:=id2+id3*int1



术语

- **语法分析(syntax analysis or parsing)**

The purpose of syntax analysis is to determine the source program's phrase structure. This process is also called parsing. The source program is parsed to check whether it conforms to the source language's syntax, and to construct a suitable representation of its phrase structure.依据源程序的语法规则把源程序的单词序列组成语法短语(表示成语法树)。

- **语法树(推导树)(parse tree or derivation tree)** 表示句型推导（或归约）的树型结构。语法树有助于理解一个句子语法结构的层次

语义分析

语义审查(静态语义)

- 标识符是否定义
- 类型是否匹配
- 类型转换

例如下面的程序片段：

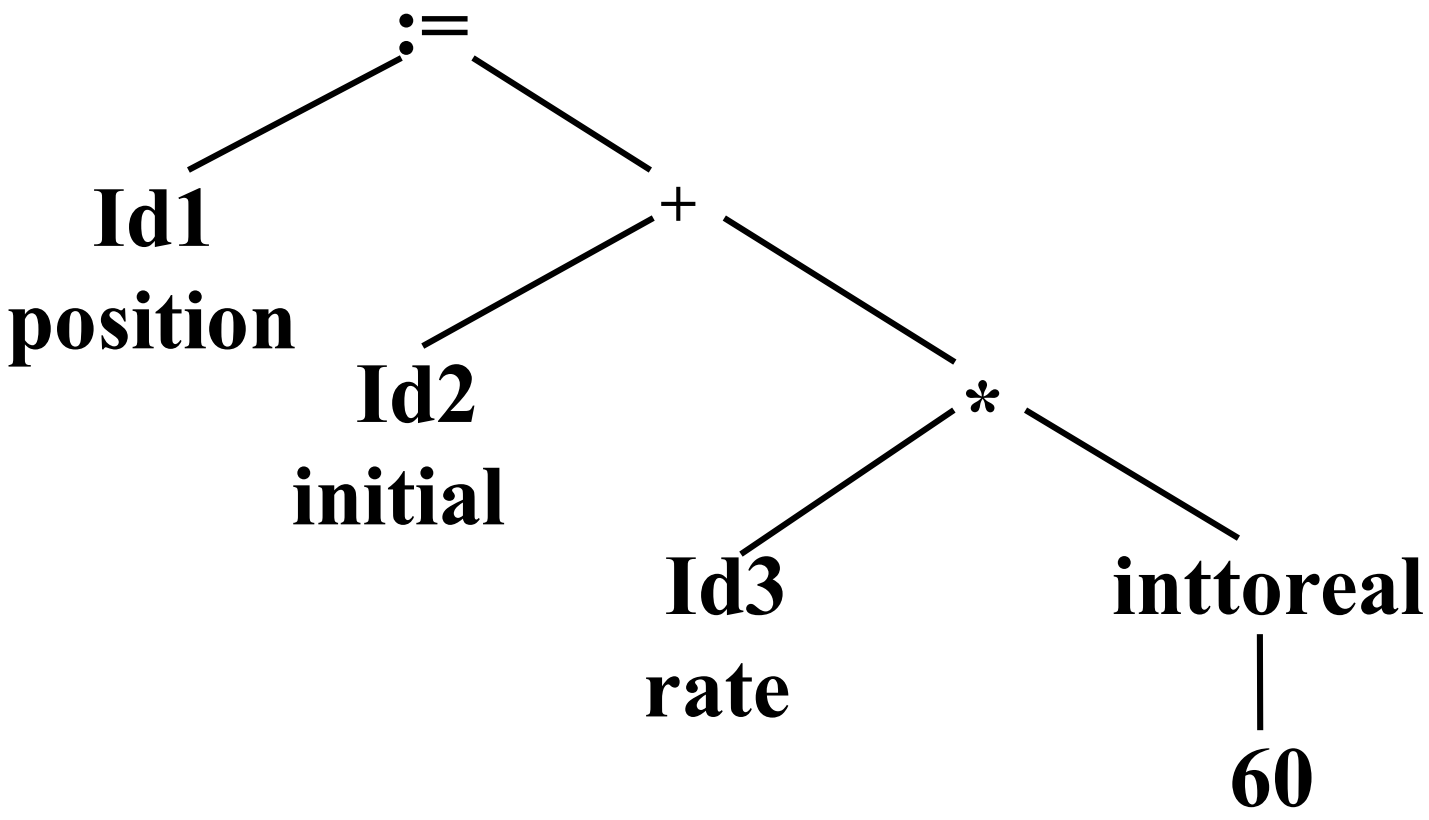
```
(1)  int arr[2], c;
(2)  c = arr1 * 10 ;
```

其中的赋值语句(2)是符合语法规则的，但是因为没有声明变量**arr1**，而存在语义错误。

又如：(语义分析的一个重要内容是类型检查,对表达式及语句中的各语法成分作类型检查和分析.)

```
int  arr [2], abc;
abc = arr * 10;
   …
program p();
    var  rate : real;
    var  initial : real;
    var  position : real ;
       …
    position := initial + rate * 60
```

语义分析



语义分析练习

请说出下面的程序段中，**error1**、**error2**和**warning**分别代表违背了什么语义规则？

```
program p(input,output);
   var rate : real;
   procedure initial;
   …
   position := initial +
    rate * 60

/* error1 */ /* error2 */
/* warning1*/
```

```
program p(input,output);
   var rate : real;
   var initial : real;
   var position : real ;
   …
   position := initial +
    rate * 60

/* warning1*/
```

error1：变量**position**没有声明

error2：**initial**是一个过程，没有返回值。

warning1：变量**initial**和**rate**没有初始化

术语

- 语义分析(semantic analysis)

The parsed program is further analyzed to determine whether it conforms to the source language's contextual constraints: scope rules, type rules

e.g. To relate each applied occurrence of an identifier in the source program to the corresponding declaration.

中间代码生成

语法分析和语义分析之后，有时将源程序变成一种内部表示形式，这种内部表示形式叫中间代码，该代码是一种简单的记号系统。例如四元式

四元式的形式为：（算符，运算对象1，运算对象2，结果）

id1:= id2 + id3 * 60

```
(1)  (inttoreal, 60,  -,   t1 )
(2)  (*,         id3, t1,  t2 )
(3)  (+,         id2, t2,  t3 )
(4)  (:=,        t3,  -,   id1 )
```

中间代码生成

中间代码的几种形式

- 逆波兰表示：运算符在其运算量之后的表达式。
- 三元式：（**OP**，**ARG1**，**ARG2**）
- 四元式：（**OP**，**ARG1**，**ARG2**，**RESULT**）
- 树：根结点**OP**，左子树**ARG1**，右子树**ARG2** 。

术语

■中间代码生成
(intermediate code generation)

This is where the intermediate representation of the source program is created. We want this representation to be easy to generate, and easy to translate into the target program. The representation can have a variety of forms, but a common one is called three-address code or 4- tuple code.

代码优化

主要任务：对中间代码进行变换，使目标代码更为高效。（节省时间和空间）

例：**id1:= id2 + id3 * 60**

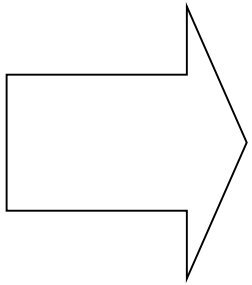
(1) (inttoreal, 60, -, t1)
(2) (*, id3, t1, t2)
(3) (+, id2, t2, t3)
(4) (:=, t3, -, id1)

变换 ⇒

(1) (*, id3, 60.0, t1)
(2) (+, id2, t1, id1)

代码优化

```
t1 = b* c
t2 = t1+ 0
t3 = b* c
t4 = t2 + t3
a = t4
```



```
t1 = b* c
t2 = t1 + t1
a = t2
```

术语

代码优化(code optimization)

- **Intermediate code optimization**

The optimizer accepts input in the intermediate representation and output a version still in the intermediate representation .

In this phase, the compiler attempts to produce the smallest, fastest and most efficient running result by applying various techniques.

- **Object code optimization**

目标代码生成

```
( *    id3    60.0   t1     )
( +    id2    t1     id1    )
```

```
mov   id3,  R2
mul   #60.0,  R2
mov  id2,  R1
add  R2,   R1
mov R1,  id1
```

表格管理

- 编译程序在其工作过程中使用最多的数据结构形式就是表格。各种各样的表格中记录着源程序的各种信息，以便需要时可以随时查询或者修改。
  - ⑩符号表：常量名、变量名、数组名、过程名、性质等
  - ⑩常数表：各种类型常数的值
  - ⑩入口名表：外部过程的名字、引用位置
  - ⑩过程引用表：过程的入口名和入口位置
  - ⑩.....
- 这些表中尤以符号表最为重要，它的生存期最长，使用也最频繁。

符号表管理

- 记录源程序中使用的名字
- 收集每个名字的各种属性信息
  - 类型、作用域、分配存储信息

名 字	种 类	类 型	层 次	偏移量
m	过 程		0	
a	变 量	real	1	d
b	变 量	real	1	d+4
c	变 量	real	1	d+8

- 在编译过程中，源程序中的标识符及其各种属性都要记录在符号表中，这些属性可以提供标识符的存储分配信息、类型信息、作用域信息等等。对于过程标识符，还要有参数信息，包括参数的个数和类型、实参和形参的结合方式等等。（第**9**章）

术语

符号表 (symbol table)

- Symbol table is a data structure which is employed to associate identifiers with their attributes .
- An identifier’s attribute consists of information relevant to contextual analysis, and is obtained from the identifier’s declaration.

出错处理

- 编译的每一个阶段都会发现源程序的错误，在发现错误之后，一般要对其有一定的处理措施，因而编译还可继续执行，不会一有错误就停止编译工作。
- 在词法分析中，能发现单词拼写错误。
- 在语法分析中，检查单词串是否符合语法的结构规则。
- 在语义分析中，编译程序进一步查出语法上虽正确但含有无意义的操作部分，如两个标识符相加，一个是数组名，一个是过程名，虽然语法上允许，但语义上不允许，各种错误，都应在相应的阶段进行处理。

出错处理过程：

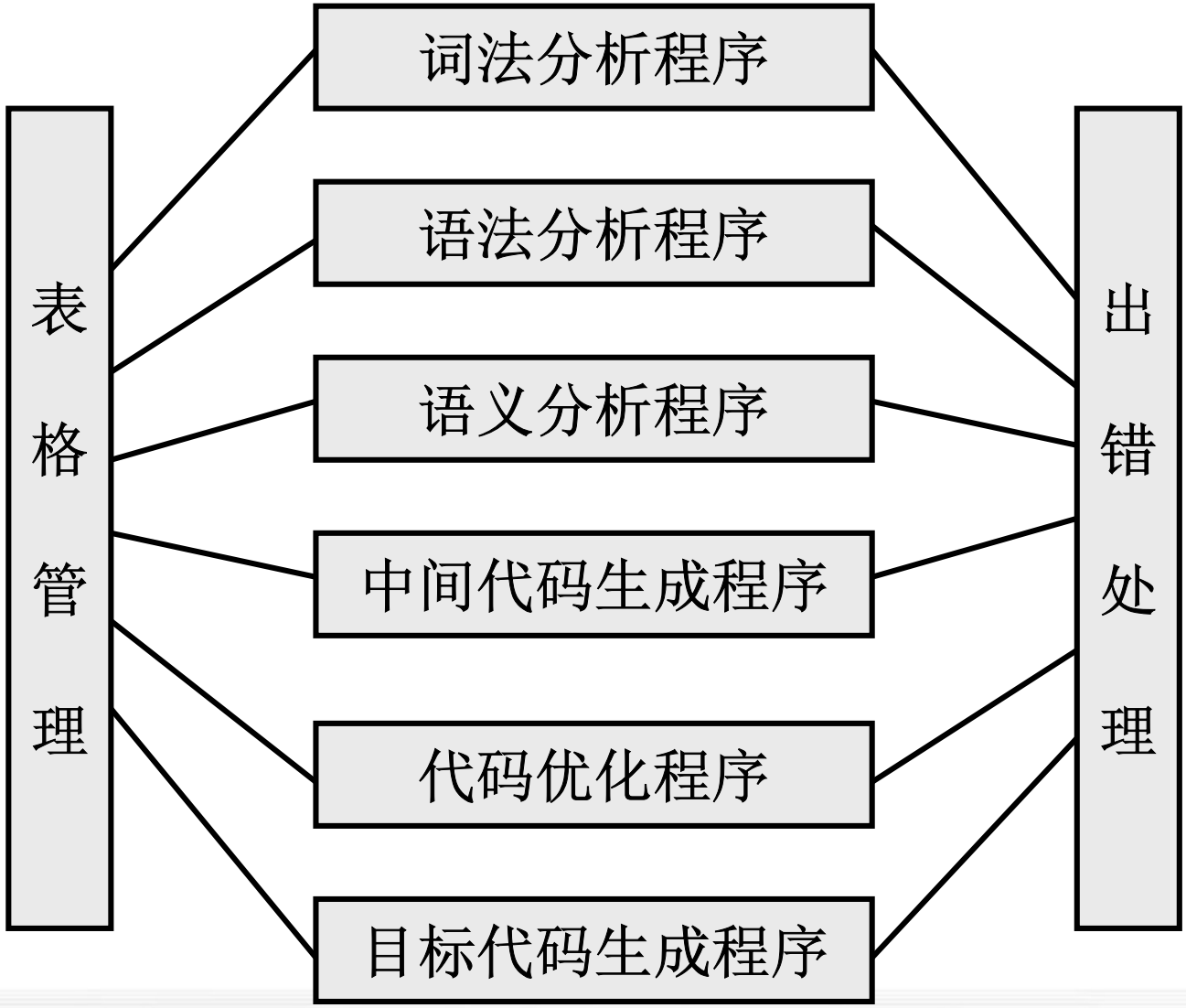
- 检查错误
- 报告出错信息
- 排错
- 恢复编译工作

术语

出错处理(error handling, error reporting, error recovery)

- The compiler should report the location of each error, together with some explanation. The major categories of compile-time error: syntax error, scope error, type error.
- After detecting and reporting an error, the compiler should attempt error recovery, means that the compiler should try to get itself into a state where analysis of the source program can continue as normally as possible.

- 词法分析程序
- 语法分析程序
- 语义分析程序
- 中间代码生成程序
- 代码优化程序
- 目标代码生成程序
- 符号表管理程序
- 出错处理程序


词法分析程序
语法分析程序
语义分析程序
中间代码生成程序
代码优化程序
目标代码生成程序
表格管理
出错处理

单选题 2分

设置

_____和代码优化部分不是每个编译程序都必需的。

A 语法分析

B 中间代码生成

C 词法分析

D 目标代码生成

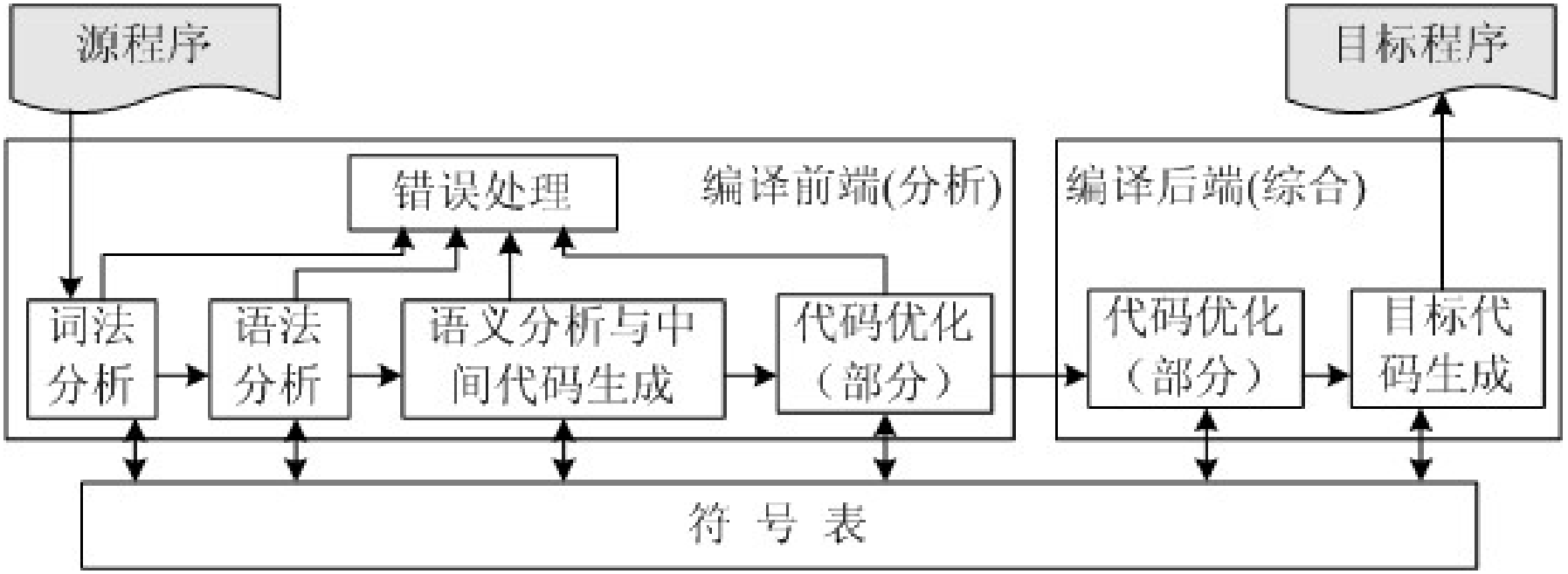
提交

- 编程过程的划分
  - **编译的前端（front end）：** 主要指与源语言有关，与目标语言无关的部分，通常包括词法分析、语法分析、语义分析和中间代码生成，与机器无关部分的代码优化(仅依赖于源程序）

以中间代码为界

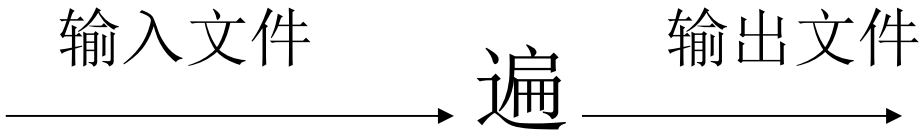
  - **编译的后端（back end）：** 指与目标机器有关的部分。如与机器有关的优化、目标代码生成。

- 编译阶段的组合


源程序
目标程序
错误处理
编译前端(分析)
编译后端(综合)
词法分析
语法分析
语义分析与中间代码生成
代码优化（部分）
代码优化（部分）
目标代码生成
符 号 表

分遍（趟，pass）问题

遍：对源程序或源程序的中间结果从头到尾扫描一次，并作有关的加工处理，生成新的中间结果或目标程序。



一个编译程序可由一遍、两遍或多遍完成。每一遍可完成不同的阶段或多个阶段的工作。

从时间和空间角度看：

- 多遍编译 — 少占内存，多耗时间
- 一遍编译 — 多占内存，少耗时间

分遍（趟，pass）问题

在编译过程中，扫描源（中间）程序的次数称为该编译程序的“遍”数。决定分遍次数的因素有：

（1）源语言

FORTRAN、**Pascal**等，只需单遍扫描；

ALGOL60要两遍扫描，

ALGOL68要三遍扫描。

若允许先使用、后说明，则通常需要多遍扫描。

N.Wirth的第一个**Pascal**编译器是在**CDC6000**上实现的一遍编译器（递归子程序法）。

“在一台内存相当大的高速计算机实现高效编译的关键是一遍加工方案”

（2）机器，尤其是内、外存的大小。

内存小，可能需要多遍。

（3）优化要求。充分优化需要多遍。

多遍的结构清晰，优化好，但重复工作多，输入输出多，编译器本身代码长，且编译速度慢，出错处理较困难。

本书的**PL/0**编译器就是一个单遍编译器。

第1章 概述 —▸▸ 1.5编译技术的应用及发展

⑩ 应用：大部分软件工具的开发，都要使用编译技术和方法

用到编译原理与技术的常见软件工具：

1、语言的结构化编辑器：

提供关键字及其匹配的关键字。

可减少语法错误，加快源程序调试。

2、语言程序的调试工具

提供判定程序的算法与功能是否正确。

3、程序格式化工具：使程序呈现清晰的结构

4、语言程序的测试工具：

第1章 概述 —▸▸ 1.5编译技术的应用及发展

测试工具
- 静态分析器 — 对源程序进行语法分析
- 动态测试器 — 测试用例、对程序进行动态测试。

- 5、高级语言之间的转换工具：
- 一种高级语言转化成另一种高级语言。

第1章 概述 —▸▸ 1.5编译技术的应用及发展

- 6、其它应用
  - 编译技术在反病毒方面的应用
  - 基于编译技术的可信赖计算方法
  - 基于编译技术的协议解析方法
  - 在云计算中运用
  - ……

第1章 概述 —▸▸ 1.5编译技术的应用及发展

⑩发展

- 基本的编译器设计近20多年来没有多大的改变，现在成为计算机科学课程中的中心一环。
- 与复杂的程序设计语言的发展结合在一起。面向对象技术兴起和应用对传统的编译技术提出新的要求。
- 新技术
  - ⑩并行编译技术
  - ⑩交叉编译技术
  - ⑩动态编译技术（运行时编译技术）

小结

- 内容
  - 1 什么是编译程序
  - 2 编译过程和编译程序的结构
  - 3 为什么要学习编译程序
- 重点
  - 对编译程序的功能和结构做一综述
- 难点
  - 了解编译程序各个成分在编译阶段的逻辑关系以及他们怎样作为一个整体完成编译任务的。

多选题 4分

设置

下列哪些说法是正确的?

A 编译程序的六个组成部分缺一不可。

B 高级语言程序到低级语言程序的转换是基于语义的等价变换。

C 含有优化部分的编译程序的执行效率高。

D 因为编译程序和解释程序具有不同的功能，所以它们的实现技术也完全不同。

E 无论一遍扫描的编译器还是多遍扫描的编译器都要对源程序扫描一遍。

F 编译程序和解释程序的根本区别在于解释程序对源程序并没有真正进行翻译。

提交

练　习

请指出下列错误信息是编译的哪个阶段报告的

(1) else没有匹配的if;

(2) 使用的函数没有定义;

(3) 在数中出现非数字字符。

答案：(1) 语法　(2) 语义　(3) 词法

课外

- 写出**pascal**语言中字符集、单词、数据类型、各种表达式、语句和程序的组成。
- 查阅如下一种资料：
  (1) 与某种语言(如**java**、**VB**等)的编译程序有关
  (2) 与编译程序的理论有关
  (3) 与某种高级语言的发展有关
  - 用某种熟悉的语言的编译程序来理解层次和遍