# 软件设计思想与方法

## Ch1 程序设计基础知识



# 什么是计算思维?

- **计算思维——运用计算科学的概念和方法去求解问题、设计系统和行为建模。**

2006 年卡耐基 . 梅隆大学**周以真**教授首次提出“计算思维”，她认为到 21 世纪计算思维能力是人人都应具备的基本能力。

- **计算思维是人的思维!**
  - 实验思维：实验　观察　发现、推断与总结 —— 观察与归纳
  - 理论思维：假设 / 预设　定义 / 性质 / 定理　证明 —— 推理和演绎
  - 计算思维：建模、构造与计算 —— 设计与构造

  计算思维关注有关可行性、可构造性和可评价性的部分；

  当前理论分析与实验在面临大规模数据的情况下，不可避免地要用计算手段来辅助进行。

- **计算思维四步骤**：

  问题分解 -> 模式识别 -> 概括与抽象 -> 算法（解题步骤）

# 如何学习计算思维?

（1） “知识/术语”随着“思维”的学习而展开，“思维”随着“知识”的贯通而形成，“能力”随着“思维”的理解而提高。

（2）从问题分析着手，强化如何进行抽象，如何将现实问题抽象为一个数学问题或者一个形式化问题，提高问题表述及问题求解的严谨性。

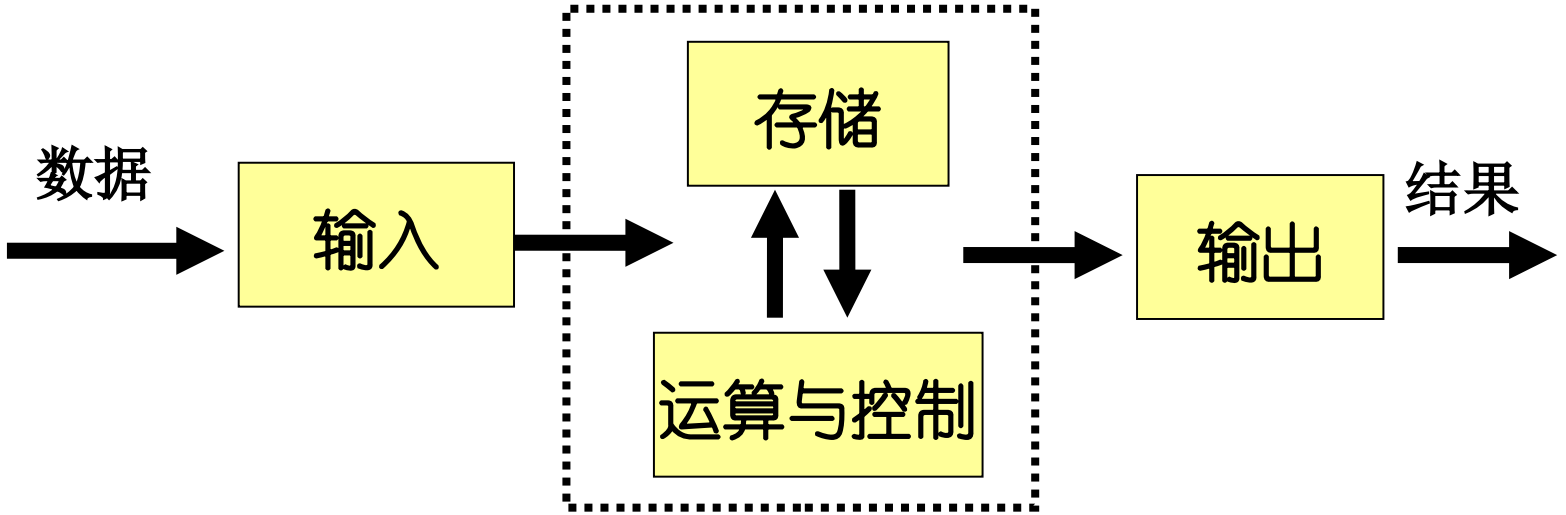
（3）通过图示化方法来展现复杂的思维可以一目了然；通过规模较小的问题求解示例来理解复杂问题的求解方法；通过从社会/自然等人们身边的问题理解到计算科学家是如何进行问题求解。

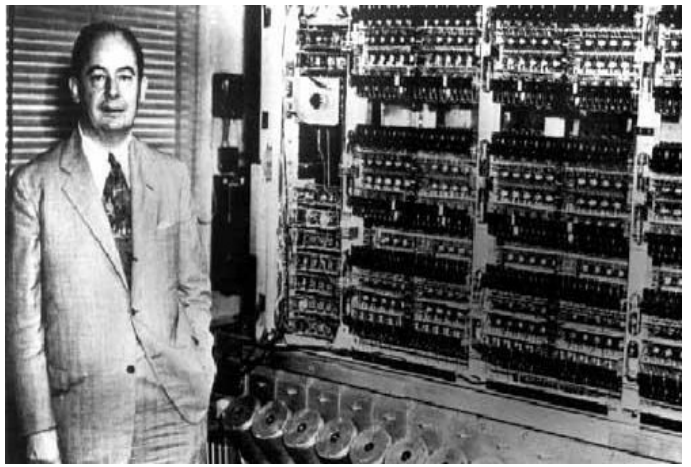
（4）追求“问题”及问题的讨论，通过逐步地提出问题，使自己从一个较浅的理解层次逐步过渡到较深入的理解层次，通过不同视角和递阶的讨论，使自己理解和确定前行的方向。

（5）宽度与深度相结合，从宽度学习开始，深度学习结束，既能够使自己理解相关的思维与知识，还能够有助于建立起较为科学的研究习惯与研究方法。

（6）思维蕴含在案例中，案例蕴含着思维。

# 冯．诺依曼计算机结构



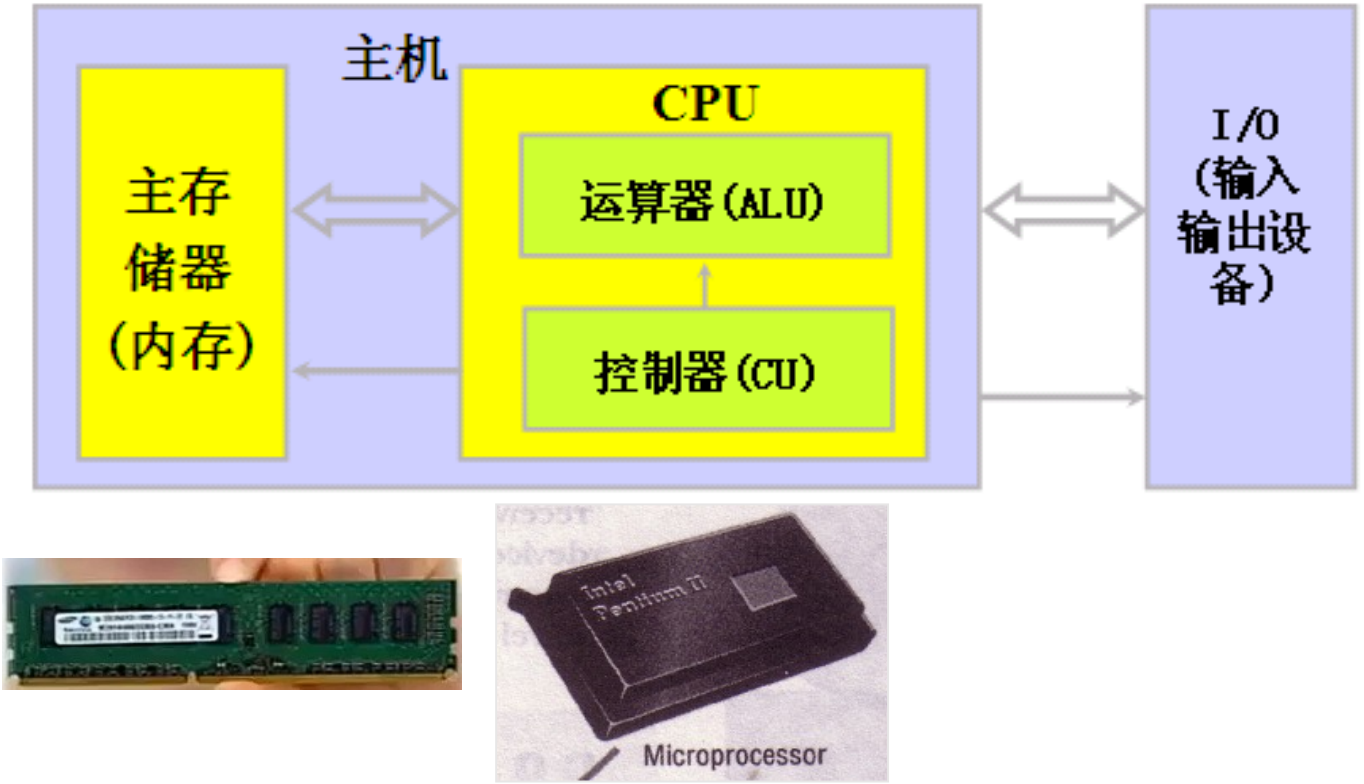
①数据或指令以二进制形式存储在存储器中；

②在控制器的指挥下，从存储器取出指令，根据指令运算；

③运算结果存入存储器（存储器的内容可以被修改）。

④完成一个任务的指令序列为程序，控制器使程序自动顺序执行；

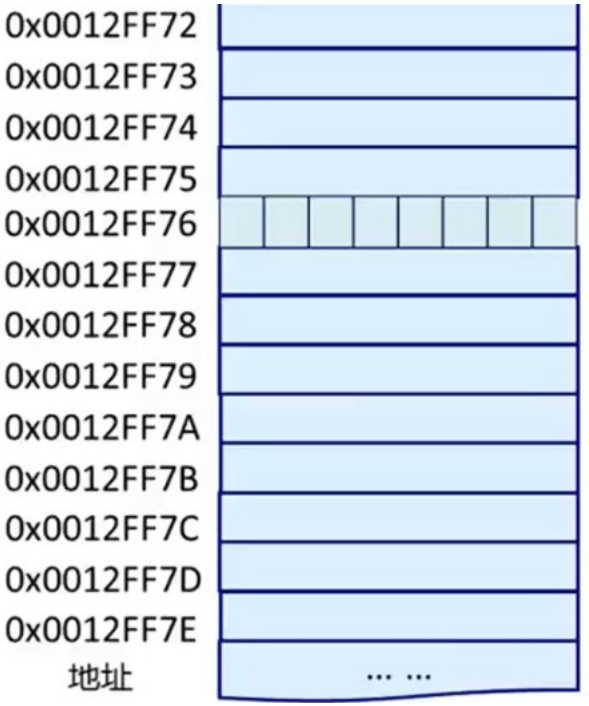
# 冯．诺依曼计算机结构












■ 存储空间单位

$2^{10}$=1024

1B (Byte) = 8 b (bit)

1KB = 1024Byte

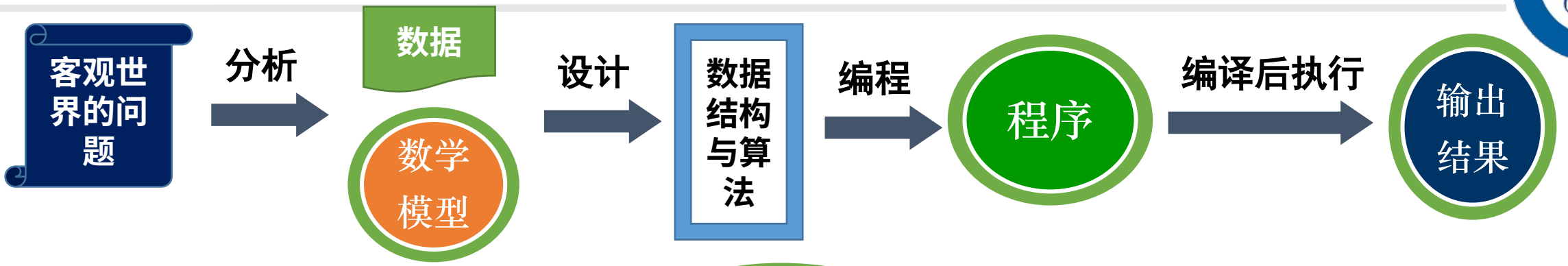
1MB = 1024KB

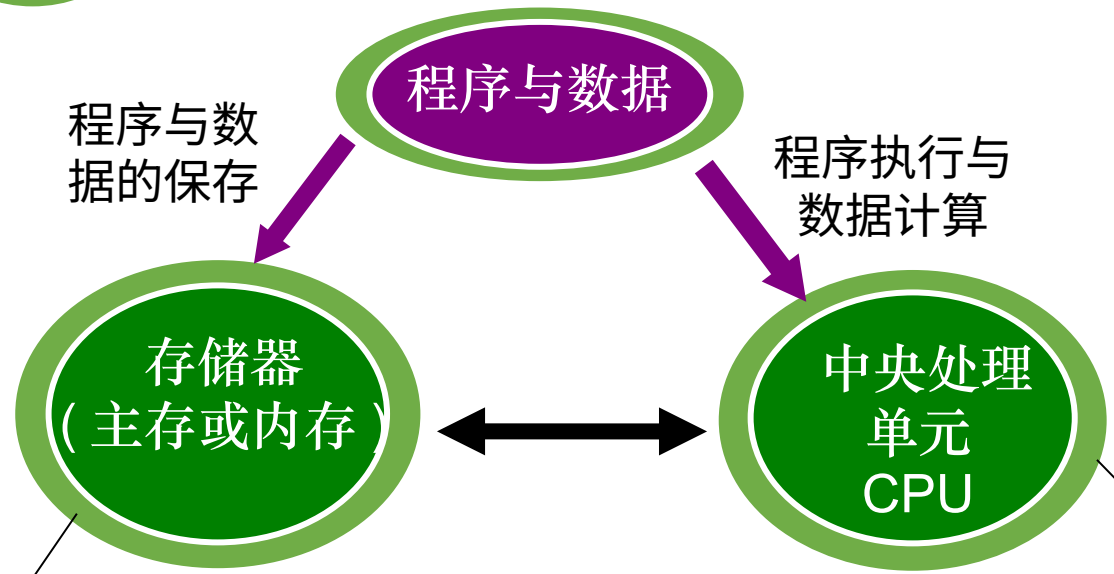
1GB= 1024MB

1TB = 1024GB

1PB = 1024TB

# 计算机的问题求解过程

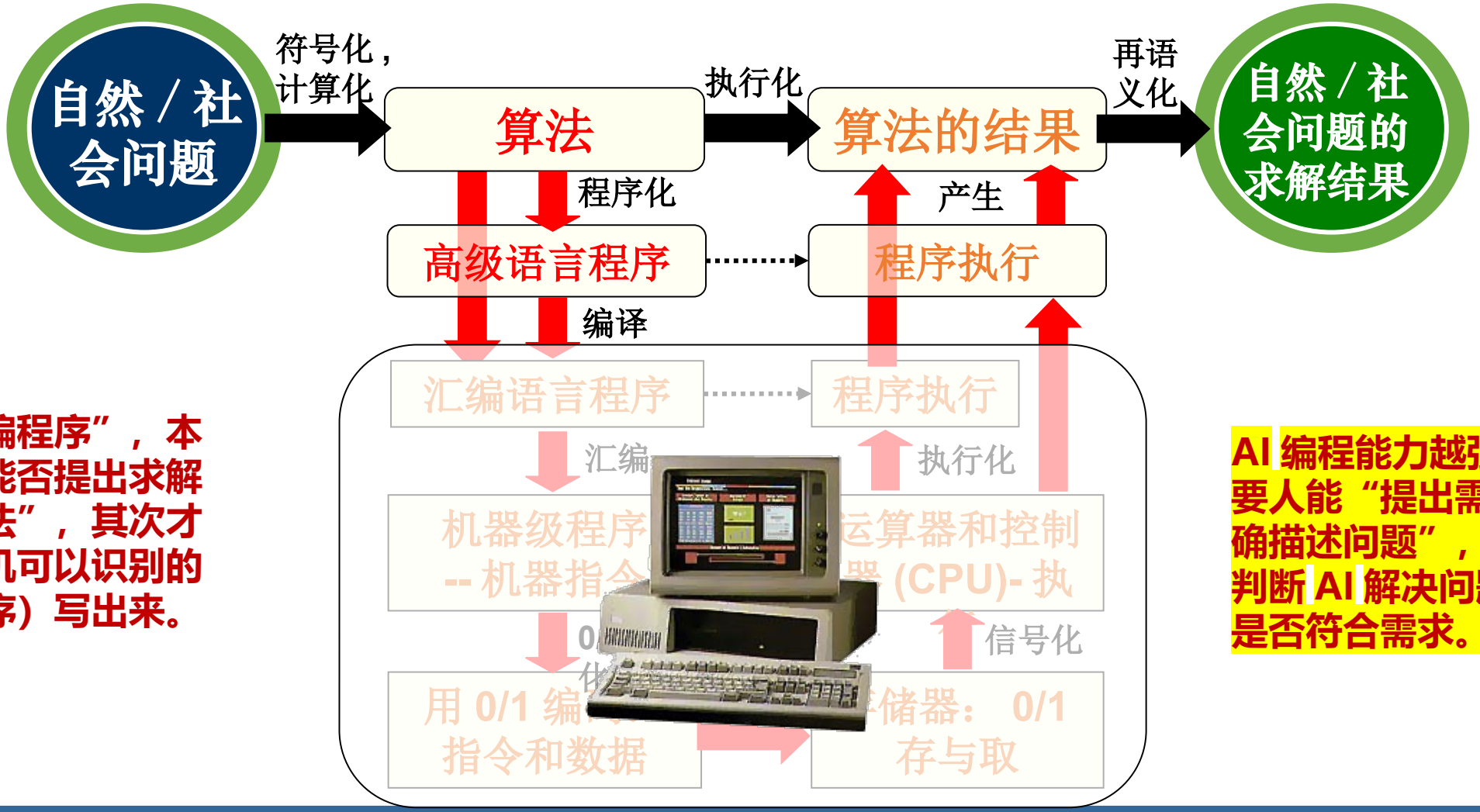



以【存储单元】为单位进行存储
一个存储单元可以是 8 位、 16 位、 32 位或 64 位
按【地址（编号）】访问存储单元

以【指令】为单位进行读取并执行。一条指令可以保存在 1 个或多个存储单元中。基本的运算指令就是两个数的加、减、乘、除等运算。

# 计算机的问题求解过程
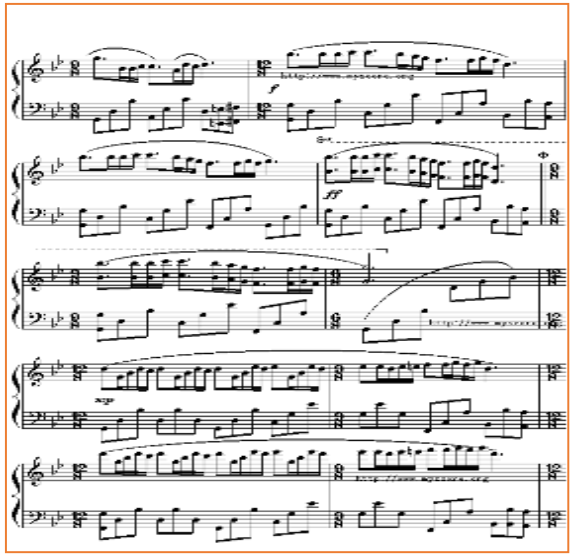



“是否会编程序”，本质上是“能否提出求解问题的算法”，其次才是用计算机可以识别的形式（程序）写出来。

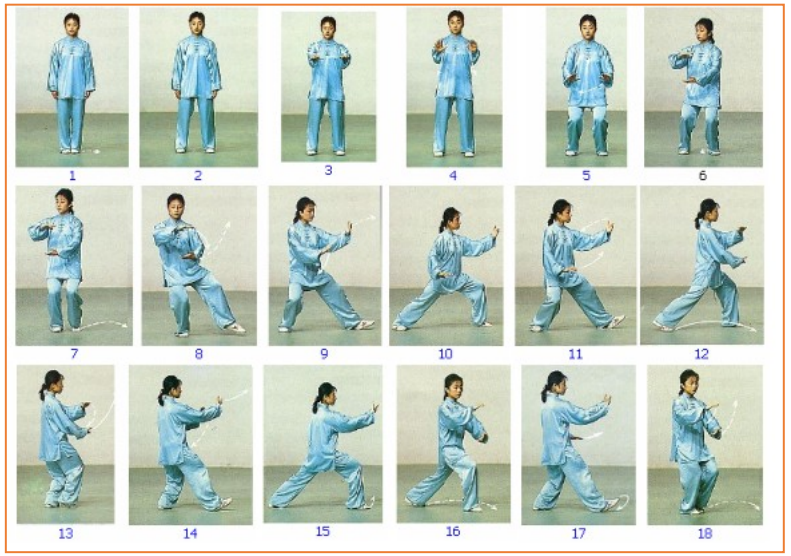
AI 编程能力越强，越需要人能“提出需求并准确描述问题”，还需能判断 AI 解决问题的思路是否符合需求。

# 算法的概念

“算法” (Algorithm) 一词源于阿拉伯数学家阿科瓦里茨米，其在公元 825 年写了著名的《波斯教科书》，书中概括了进行四则算术运算的计算规则。

**算法**是一个有穷规则的集合，它用规则规定了解决某一特定类型问题的运算序列，或者规定了任务执行或问题求解的一系列步骤。






如音乐乐谱、太极拳谱等都可看作广义的算法

# 算法的概念

例： 设计算法求两个自然数的最大公约数。

【中学计算方法】将两个自然数分别进行质因数分解，然后找出所有公因子并将这些公因子相乘。例如， $48=2\times2\times2\times2\times3$ ， $36=2\times2\times3\times3$ ，公因子有 2 、 2 、 3 ，因此， 48 和 36 的最大公约数为 $2\times2\times3=12$ 。

【算法思路】设两个自然数是 $m$ 和 $n$ ，步骤如下：

步骤 1： 找出 $m$ 的所有质因子
步骤 2： 找出 $n$ 的所有质因子
步骤 3： 从第 1 步和第 2 步得到的质因子中找出所有公因子
步骤 4： 将找到的所有公因子相乘，结果即为 $m$ 和 $n$ 的最大公约数

有什么问题吗？

- 如何找出所有质因子？如何找出所有公因子？
- 质因数分解尚未找到多项式时间算法

# 算法的概念

## 欧几里得算法——辗转相除求两个正整数的最大公约数

**欧几里德算法**

**输入**：正整数 *M* 和正整数 *N*

**输出**： *M* 和 *N* 的最大公约数 ( 设 M>N)

**步骤**：

Step 1. *M* 除以 *N*，记余数为 *R*

Step 2. 如果 *R* 不是 *0* ，将 *N* 的值赋给 *M*, *R* 的值赋给 *N*，返回 Step 1; 否则，最大公约数是 *N*，输出 *N*，算法结束。






| | M | N | R | 最大公约数 |
|---|---|---|---|---|
| 具体问题 | 32 | 24 | | ? |
| 算法计算过程 | | | | |
| 1 | 32 | 24 | 8 | |
| 2 | 24 | 8 | 0 | 8 |
| | | | | |
| 具体问题 | 31 | 11 | | ? |
| 算法计算过程 | | | | |
| 1 | 31 | 11 | 9 | |
| 2 | 11 | 9 | 2 | |
| 3 | 9 | 2 | 1 | |
| 4 | 2 | 1 | 0 | 1 |

# 进位计数制

## 进位计数制与十进制

◆进位计数制：用数码和带有权值的数位来表示有大小关系的数值性信息的表示方法。

◆十进制

- 有 0,1,2,3,4,5,6,7,8,9 共十个数码
- 数码的位置规定了数码的等级“权 / 数位”： $10^i$
- 逢十进一、借一当十。高数位的 1 相当于低数位的 10
- “十”——基值、十进制

$$(2\ \ 4\ \ 5.\ \ 2\ \ 5)_{10}$$

$10^2\ 10^1\ 10^0\ 10^{-1}\ 10^{-2}$

$$(245.25)_{10} = 2\times10^2 + 4\times10^1 + 5\times10^0 + 2\times10^{-1} + 5\times10^{-2}$$

# 进位计数制

## 二进制

- 有 0,1 共两个数码
- 数码的位置规定了数码的等级“权 / 数位”： $2^i$
- 逢二进一、借一当二。高数位的 1 相当于低数位的 2
- “二” ---- 基值，二进制

$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0\ .\ 2^{-1}\ 2^{-2}$ ———— 数位的权值

7 6 5 4 3 2 1 0 .-1 -2 ———— 数位

例如： $(1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ .\ 0$ ———— 二进制数

$$1)_2 = 1\times2^7+1\times2^6+1\times2^5+1\times2^4+0\times2^3+1\times2^2+0\times2^1$$
$$+1\times2^0+0\times2^{-1}+1\times2^{-2} = (245.25)_{10}$$

# 进位计数制

## r 进制

- 有 0,1,…, 共 r 个数码
- 数码的位置规定了数码的等级“权 / 数位”： $r^i$
- 逢 r 进一、借一当 r。高数位的 1 相当于低数位的 r
- “r”---- 基值，r 进制

$r^{n-1}\ r^{n-2}\ \ldots\ r^2\ r^1\ r^0\ .\ r^{-1}\ r^{-2}\ \ldots\ r^{-m}$ ——— 数位的权值

$n\text{-}1\ \ n\text{-}2\ \ldots\ 2\ \ 1\ \ 0\ .\text{-}1\ \ \text{-}2\ \ldots\ \text{-}m$ ——— 数位

$(d_{n-1}d_{n-2}\ldots\ldots d_2 d_1 d_0 . d_{-1} d_{-2}\ldots\ldots d_{-m})_r$ ——— r 进制数

$$= d_{n-1}r^{n-1} + d_{n-2}\ r^{n-2}+\cdots+d_2\ r^2 + d_1\ r^1 + d_0\ r^0 + d_{-1}\ r^{-1} + d_{-2}\ r^{-2} + \cdots + d_{-m}\ r^{-m}$$

- 十六进制： 0,1,2,3,4,5,6,7,8,9,A(10), B(11), C(12), D(13), E(14), F(15)

# 进位计数制

**一个数值用不同进制表示，其数码串是不同的**

$(7\ 5\ 3\ .\ 3\ 7)_8$ = 7 5 3 . 3 7 O

$=7\times8^2+5\times8^1+3\times8^0+3\times8^{-1}+7\times8^{-2}$

$=(491.484375)_{10}$

$(7\ 5\ 3\ .\ 3\ 7)_{16}$ = 7 5 3 . 3 7 H = 0x 7 5 3 . 3 7

$=7\times16^2+5\times16^1+3\times16^0+3\times16^{-1}+7\times16^{-2}$

$=(1875.2148)_{10}$

$(7\ 5\ 3\ .\ 3\ 7)_{12}$

$=7\times12^2+5\times12^1+3\times12^0+3\times12^{-1}+7\times12^{-2}$

$=(1071.2986)_{10}$

245 的十进制表示记为：

245

245 的二进制表示记为：

11110101

245 的八进制表示记为：

365

245 的十六进制表示记为：

F5

同一个数串，不同进制，表示不同的数

# 总结：数的进制及转换

| 数制 | 十进制数 | 二进制数 | 八进制数 | 十六进制数 |
|---|---|---|---|---|
| 数码 | 0~9 | 0~1 | 0~7 | 0~9,A~F,a~f |
| 基 | 10 | 2 | 8 | 16 |
| 权 | $10^0$,$10^1$,$10^2$,… | $2^0$, $2^1$, $2^2$,… | $8^0$, $8^1$, $8^2$,… | $16^0$,$16^1$,$16^2$,… |
| 表示 | | | | |
| 特点 | 逢十进一 | 逢二进一 | 逢八进一 | 逢十六进一 |

- 八进制： $4275=4\times8^3+2\times8^2+7\times8^1+5\times8^0$

$+5\times10^0$

$+14\times16^0$

## ■进制之间的转换

- 二进制、八进制、十六进制转换成十进制
  - 方法：按权相加

例 $(111011)_2 = 1\times2^5 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = (59)_{10}$

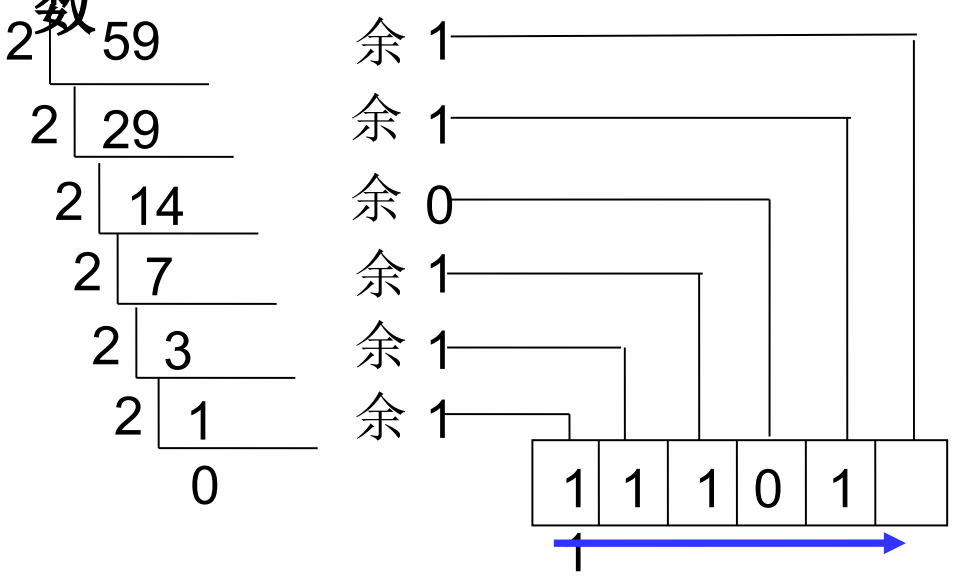
例 $(136)_8 = 1\times8^2 + 3\times8^1 + 6\times8^0 = (94)_{10}$

例 $(1F2A)_{16} = 1\times16^3 + 15\times16^2 + 2\times16^1 + 10\times16^0 = (7978)_{10}$

# 十进制转换成二进制、八进制、十六进制

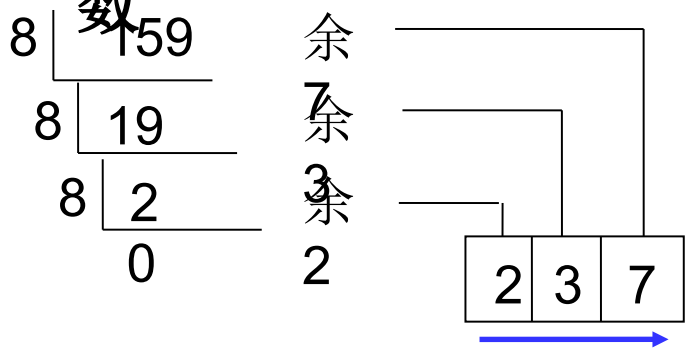
- 整数：连续除以基，从低到高记录余数，直至商为 0
- 小数：连续乘以基，从高到低记录整数，直至小数为 0

## 十进制数 59 转换成二进制数



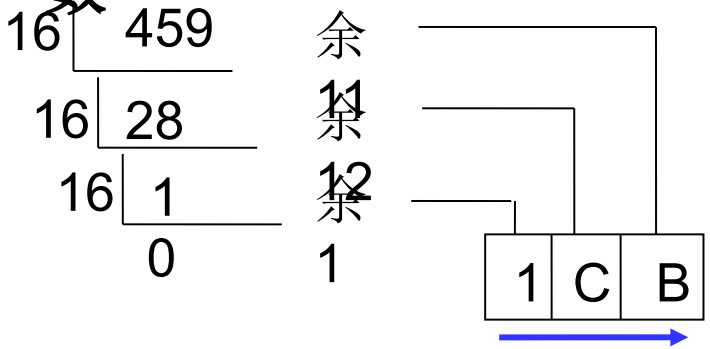

$(59)_{10}=(111011)_2$

## 159 转换成八进制数




$(159)_{10}=(237)_8$

## 459 转换成十六进制数




$(459)_{10}=(1CB)_{16}$

# 例如：将$(237.625)_{10}$转化成二进制




则$(237.625)_{10} = (11101101.101)_2$

- **二进制、八进制与十六进制之间的互换**
  - **二进制 → 八进制、十六进制（收缩法）**

**3 位二进制 = 1 位八进制；**

**4 位二进制 = 1 位十六进制；**

例如：$(10001011)_2 = (010,001,011)_2 = (213)_8$

又如：$(10001011)_2 = (1000,1011)_2 = (8B)_{16}$

- **八进制、十六进制 → 二进制（扩展法）**

**1 位八进制 = 3 位二进制；**

**1 位十六进制 = 4 位二进制；**

例如：$(162)_8 = (001,110,010)_2 = (1110010)_2$

又如：$(3E)_{16} = (0011,1110)_2 = (111110)_2$

# 信息的表示

## 机器字长

【机器字长】是指机器内部进行数据处理、信息传输等的基本单元所包含的二进制位数，通常是 8 位、 16 位、 32 位、 64 位等。

**例：（78）$_{10}$ 转换成二进制数。不同字长的表示，结果是不同的**

**（1）（78）$_{10}$ = （1001110）$_2$**

**（2）（78）$_{10}$ = （01001110）$_2$**

**（3）（78）$_{10}$ = （00000000 01001110）$_2$**

**（4）（78）$_{10}$ = （00000000 00000000 00000000 01001110）$_2$**

未考虑机器字长

8 位机器字长

16 位机器字长

32 位机器字长

机器字长

数的表示范围

溢出

比特（bit）与字节（Byte）

# 信息的表示

## 有符号数和无符号数

16 位机器字长

**无符号数**

无符号机器数的表示范围：$0 \le X < 2^{16}$，即 0 至 65 535

（00000000 00000000）$_2$——（11111111 11111111）

**有符号数**

数的符号也用 0 和 1 表达，0 表“+”号，1 表“-”号

（0000000 00000000）$_2$　　（0 1111111 11111111）$_2$

（1 0000000 00000000）$_2$　　（1 1111111 11111111）$_2$

有符号数的大小?

有符号数的表示范围?

# 信息的表示

## 原码、反码和补码

**正数**

原码、反码和补码是一样的

**负数**

原码、反码和补码表示的大小是不同的

原码、反码和补码的表示范围也是不同的

| 真实数值 (带符号的 n 位二进制数) | 十进制数 | 机器数(n+1 位二进制数,其中第 n+1 位表符号,0 表示正号,1 表示负号) | | |
|---|---|---|---|---|
| | | 原码 | 反码 | 补码 |
| + 11...11 | $+(2^n-1)$ | 0 11...11 | 0 11...11 | 0 11...11 |
| + 10...00 | $+2^{n-1}$ | 0 10...00 | 0 10...00 | 0 10...00 |
| + 00...00 | +0 | 0 00...00 | 0 00...00 | 0 00...00 |
| - 00...00 | -0 | 1 00...00 | 1 11...11 | 0 00...00 |
| - 10...00 | $-2^{n-1}$ | 1 10...00 | 1 01...11 | 1 10...00 |
| - 11...11 | $-(2^n-1)$ | 1 11...11 | 1 00...00 | 1 00...01 |
| -100...00 | $-2^n$ | - | - | 1 00...00 |
| | | 正数的原码、反码同补码形式是一样的。最高位为 0 表示正数 | | |
| | | 负数的最高位为 1,表示负数。其余同真实数值的二进制数。 | 负数的最高位为 1,表示负数。其余在真实数值的二进制数基础上逐位取反。 | 负数的最高位为 1,表示负数。其余在反码基础上最低位加 1 后形成的。它的负数不包括 0，但包括 $-2^n$ |
| | | 机器数由于受到表示数值的位数的限制，只能表示一定范围内的数。超出此范围则为“溢出” | | |

# （例：设用一字节表示数）

| | 原码 | 反码 | 补码 |
|---|---|---|---|
| +7 | 00000111 | 00000111 | 00000111 |
| -7 | 10000111 | 11111000 | 11111001 |
| +0 | 00000000 | 00000000 | 00000000 |
| -0 | 10000000 | 11111111 | 00000000 |
| 数的范围 | 01111111~11111111 (-127~+127) | 01111111~10000000 (-127~+127) | 01111111~10000000 (-128~+127) |

- 数的范围：

  与计算机的字长有关，字长越长，运算精度越高。

  例如：字长为 32 位，则所能表示的无符号整数范围是：

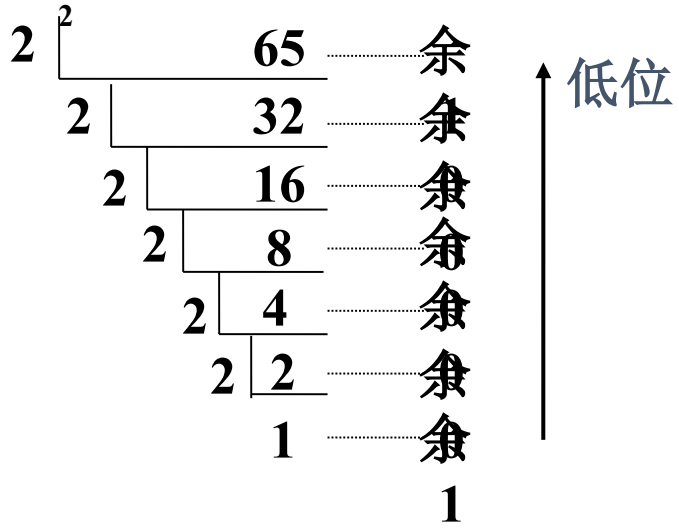
$$0 \leq X \leq 2^{32} - 1$$

# 8 位二进制数的不同含义

| 8 位二进制数据 | 十六进制数 | 无符号十进制数 | 原码 | 反码 | 补码 |
|---|---|---|---|---|---|
| 0000 0000 | 00H | 0 | +0 | +0 | +0 |
| 0000 0001 | 01H | 1 | +1 | +1 | +1 |
| 0000 0010 | 02H | 2 | +2 | +2 | +2 |
| …… | …… | …… | …… | …… | …… |
| 0111 1100 | 7CH | 124 | +124 | +124 | +124 |
| 0111 1101 | 7DH | 125 | +125 | +125 | +125 |
| 0111 1110 | 7EH | 126 | +126 | +126 | +126 |
| 0111 1111 | 7FH | 127 | +127 | +127 | +127 |
| 1000 0000 | 80H | 128 | −0 | −127 | −128 |
| 1000 0001 | 81H | 129 | −1 | −126 | −127 |
| 1000 0010 | 82H | 130 | −2 | −125 | −126 |
| …… | …… | …… | …… | …… | …… |
| 1111 1100 | FCH | 252 | −124 | −3 | −4 |
| 1111 1101 | FDH | 253 | −125 | −2 | −3 |
| 1111 1110 | FEH | 254 | −126 | −1 | −2 |
| 1111 1111 | FFH | 255 | −127 | −0 | −1 |

# 例：设机器字长为 16 ，求十进制数 +65 、 -65 的原码、反码和补码

1. 先求（65）$_{10}$ =（ ？ ）




2. 将得到的二进制码用 0 补足数值位的位数——这里为 15，得到：

$(65)_{10}$ = (000 0000 0100 0001)$_2$

3. 对正数，符号位（最高位）补 0，且其原码＝反码＝补码，则：

（＋65）$_{10}$ 的原码＝反码＝补码 ＝ 0000 0000 0100 0001

4. 对负数，符号位（最高位）补 1，则：（－65）$_{10}$ 原码 ＝ 1000 0000 0100 0001

（－65）$_{10}$ 的反码 ＝ 1111 1111 1011 1110

（－65）$_{10}$ 的补码 ＝ 1111 1111 1011 1111

# 原码、反码、补码的比较

- 原码表示直观
  - 符号位和数值位不等同，加减运算的规则比较复杂
  - N 位的原码表示真值范围为 $-(2^{N-1}-1)$ ~ $+(2^{N-1}-1)$
- 反码表示法将符号位和数值位等同看待
  - 即符号位和数值位一起参加运算，运算规则较简单
  - N 位的反码表示真值范围为 $-(2^{N-1}-1)$ ~ $+(2^{N-1}-1)$
- 补码表示法是计算机中最普遍采用的数据表示方法。
  - 数据符号位都可以参加运算，可使减法运算转换为加法
  - 0 的表示形式是唯一的
  - N 位的补码表示真值范围为 $-(2^{N-1})$ ~ $+(2^{N-1}-1)$

# 非数值性信息：编码与组合

**编码**

编码是以若干位数码或符号的不同组合来表示非数值性信息的方法，它是人为地将若干位数码或符号的每一种组合指定一种唯一的含义。

**ASCII码**

## 英文字母与符号的二进制编码方法

**规律性：英文字母与符号是按照二进制次序进行编码，连续排列**

**机器字长8位及其倍数**

| $b_3b_2b_1b_0$ \ $b_6b_5b_4$ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 0000 | NUL | DLE | SP | 0 | @ | P | ` | p |
| 0001 | SOH | DC1 | ! | 1 | A | Q | a | q |
| 0010 | STX | DC2 | " | 2 | B | R | b | r |
| 0011 | ETX | DC3 | # | 3 | C | S | c | s |
| 0100 | EOT | DC4 | $ | 4 | D | T | d | t |
| 0101 | ENQ | NAK | % | 5 | E | U | e | u |
| 0110 | ACK | SYN | & | 6 | F | V | f | v |
| 0111 | BEL | ETB | ' | 7 | G | W | g | w |
| 1000 | BS | CAN | ( | 8 | H | X | h | x |
| 1001 | HT | EM | ) | 9 | I | Y | i | y |
| 1010 | LF | SUB | * | : | J | Z | j | z |
| 1011 | VT | ESC | + | ; | K | [ | k | { |
| 1100 | FF | FS | , | < | L | ] | l | \| |
| 1101 | CR | GS | - | = | M | \ | m | } |
| 1110 | SO | RS | . | > | N | ^ | n | ~ |
| 1111 | SI | US | / | ? | O | - | o | DEL |

# 程序和程序设计语言

- 程序：完成一定功能、可被计算机解释、执行的命令组合（代码）。
- 程序采用某种特定的编程语言来实现，包括：
  - 汇编语言：与具体硬件对应， PC 为 X86 指令集等
  - 高级语言：与硬件基本无关，采用类似自然语言的编程语言，分为结构化语言和面向对象语言等。如 C 为结构化语言； C++ 、 java 等为面向对象语言；
- C 程序设计语言
  - 函数式语言，一个程序包含一个或多个函数，但只有 1 个 main 函数，程序从 main 函数开始执行，并在 main 函数中结束。
  - 有字符，数（整数、实数），指针，数组，结构等数据类型。
  - 包含顺序，选择，分支等几种流程控制功能。

# C 程序语言的发展

## 产生背景与过程

时间： 1972~1973

地点：美国贝尔实验室

目的： UNIX 操作系统

设计人： Ken.Thompson 和 Dennis.M.Ritchie

## C 标准

K&R 合著《 The C Programming Language 》

ANSI C： 1987 年

C90： 1990 年， ISO 接受 C89 作为国际标准

C99： 1999 年发布， ISO 标准

C11： 2011 年发布

C17： 2018 年发布

# 例：第一个C程序

```
/*example1.1   The  first  C  Program*/
#include <stdio.h>
int main(void)
{
        printf("Hello,World!");
        return 0;
}
```

注释

编译预处理

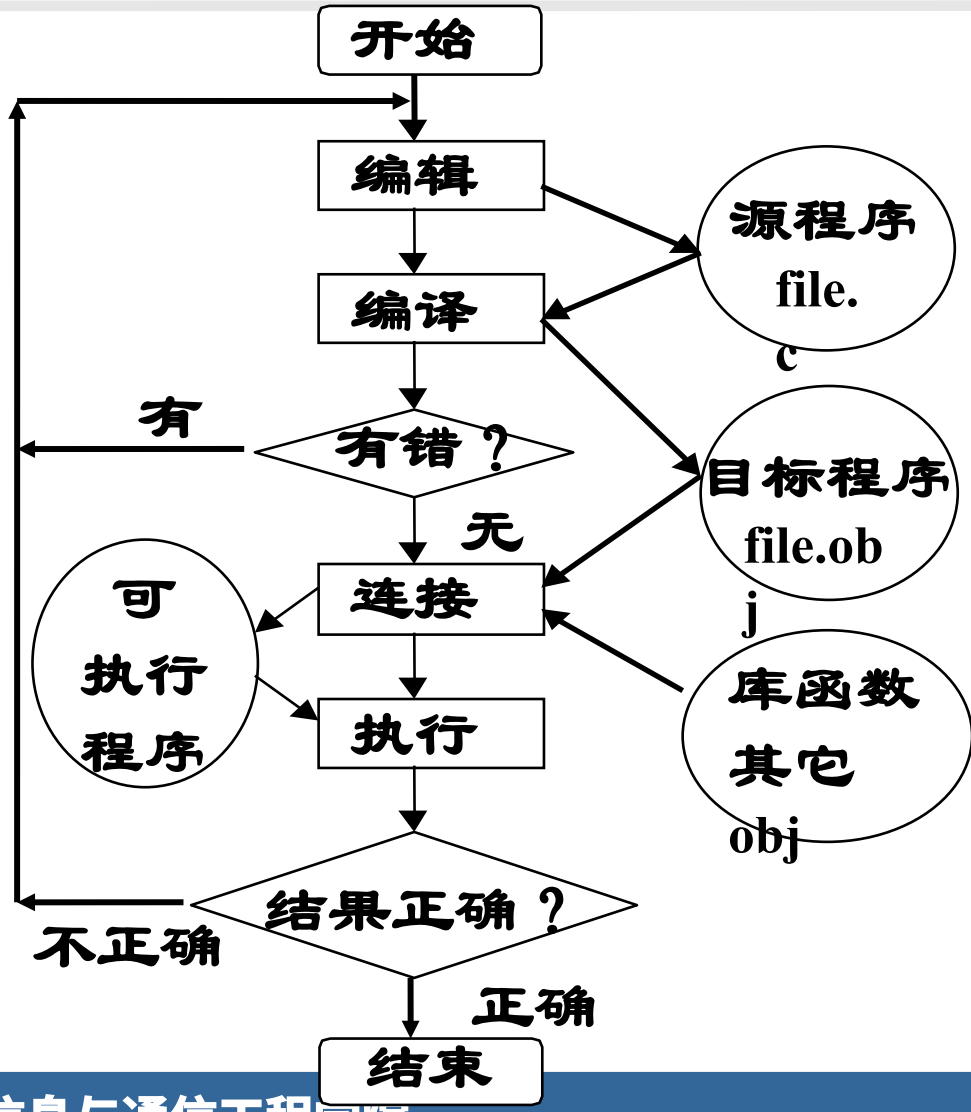
函数

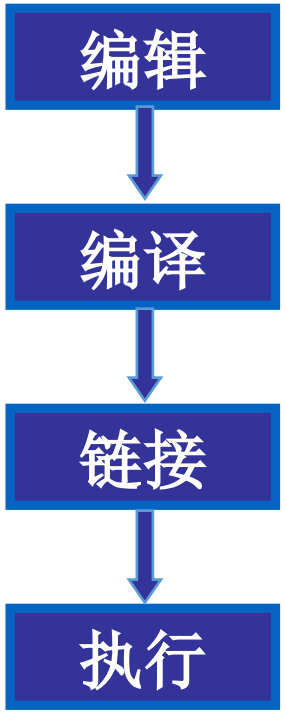
语句

输出：

Hello,World!

# C 程序的上机步骤




程序代码的录入，生成源程序 file.c

语法分析查错，翻译生成目标程序 file.obj

与其它目标程序或库链接装配，生成可执行程序 file.exe