



第一讲 计算机基础

- 信息的表示与存储
- 算法

潘建瑜@MATH.ECNU

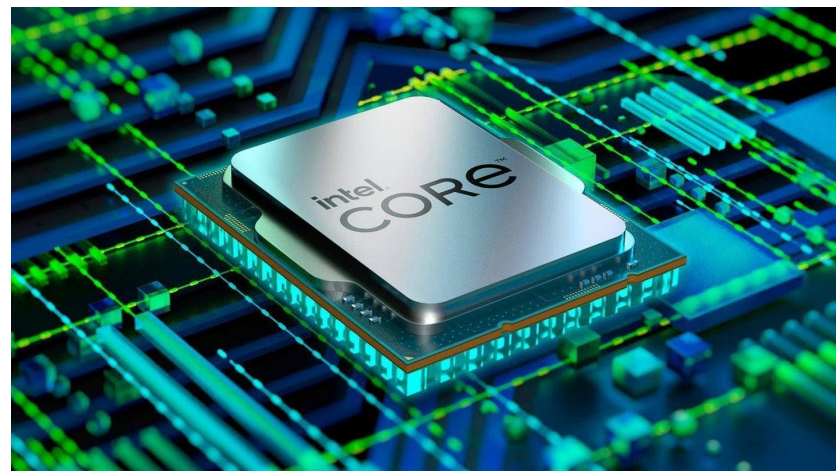


1

信息的表示与存储

2

算法基本概念



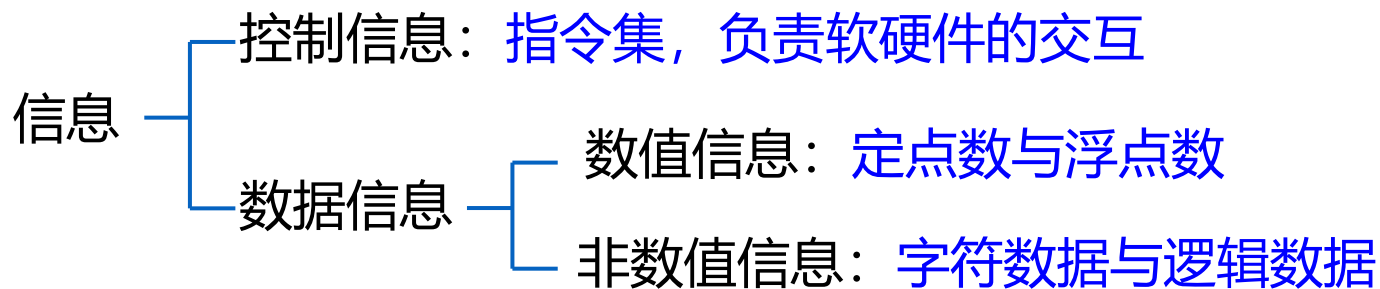
1

信息的表示与存储

- 计算机的数字系统
- 常见的进制数及它们之间的转换
- 原码，反码与补码
- 非数值信息的表示

计算机信息分类

计算机内部的信息分类



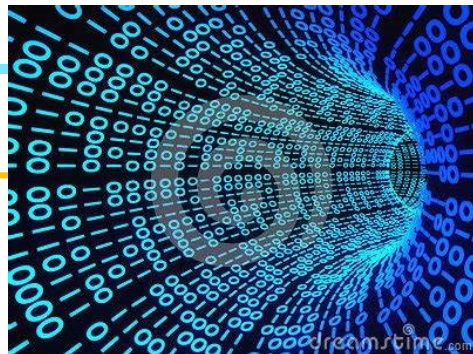
信息的存储单位

- ▶ 基本单位: 位 (bit) , 字节 (Byte=8bit) , 最小存储单元是字节
- ▶ 一个英文字符占一个字节, 一个汉字字符占两个字节

计算机数字系统

计算机数字系统

- 计算机采用的是 二进制 数字系统
- 基本符号：0、1
 - 优点：易于物理实现、运算简单、可靠性高、通用性强
 - 缺点：可读性差



常用的数制

- ▶ 二进制，八进制，十进制，十六进制

不同进制之间的转换

二进制、八进制、十六进制 → 十进制

各位数字与它的权相乘，然后相加

例：

$$(101.11)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = (5.75)_{10}$$

$$(506.2)_8 = 5 \times 8^2 + 0 \times 8^1 + 6 \times 8^0 + 2 \times 8^{-1} = (326.25)_{10}$$

$$(10.C)_{16} = 1 \times 16^1 + 0 \times 16^0 + 12 \times 16^{-1} = (16.75)_{10}$$

不同进制之间的转换

十进制 → 二进制

- 整数：辗转相除法
- 纯小数：与 2 相乘后取整数部分

2	34		余数
2	17	-----	0
2	8	-----	1
2	4	-----	0
2	2	-----	0
2	1	-----	0
	0	-----	1

低位



高位



$$34_{10} = 100010_2$$

† 十进制整数转化为其它进制的方法类似

不同进制之间的转换

十进制 \rightarrow 二进制

- 整数：辗转相除法
- 纯小数：与 2 相乘后取整数部分

$$0.3125 \times 2 = 0.625$$

$$0.625 \times 2 = 1.25$$

$$0.25 \times 2 = 0.5$$

$$0.5 \times 2 = 1.0$$



$$0.3125_{10} = 0.0101_2$$

► 每次相乘后去掉整数部分，不断乘下去，直到小数部分为 0 或达到指定的精度为止，然后取每次相乘后的整数部分即可

† 绝大部分浮点数无法用二进制精确表示，如 0.1, 0.2, 0.3, ...

不同进制之间的转换

八进制、十六进制 \leftrightarrow 二进制

- 每个八进制数对应于一个三位二进制数
- 每个十六进制数对应于一个四位二进制数

0 \leftrightarrow 000

1 \leftrightarrow 001

2 \leftrightarrow 010

3 \leftrightarrow 011

4 \leftrightarrow 100

5 \leftrightarrow 101

6 \leftrightarrow 110

7 \leftrightarrow 111

0 \leftrightarrow 0000

1 \leftrightarrow 0001

2 \leftrightarrow 0010

3 \leftrightarrow 0011

4 \leftrightarrow 0100

5 \leftrightarrow 0101

6 \leftrightarrow 0110

7 \leftrightarrow 0111

8 \leftrightarrow 1000

9 \leftrightarrow 1001

A \leftrightarrow 1010

B \leftrightarrow 1011

C \leftrightarrow 1100

D \leftrightarrow 1101

E \leftrightarrow 1110

F \leftrightarrow 1111

例:

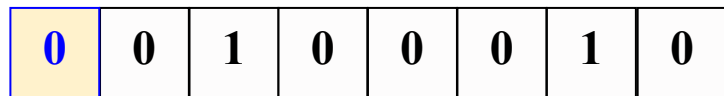
$11010.10_2 = 0001\ 1010\ .1000_2 = 1A.8_{16}$

原码

数在计算机内部的存储方式

- 原码、反码、补码
- 二进制数的原码：符号 + 大小
 - 符号位：用“0”表示正，用“1”表示负，放在最高位（最左边）

例：用 1 个字节表示正整数 34



符号位

34 ↔ 00100010

-34 ↔ 10100010

† 原码的优点：直观

† 缺点：(1) 四则运算要考虑符号位，规则复杂；(2) 零的表示不唯一

反码与补码

反码 与 补码

- 正数的反码和补码：与原码相同；
- 负数的反码：符号位不变，其它位取反（0变1，1变0）
- 负数的补码：反码的最末位加 1

		原码		反码		补码
34	↔	0 0100010	↔	0 0100010	↔	0 0100010
-34	↔	1 0100010	↔	1 1011101	↔	1 1011110

计算机以补码方式存放数据

† 注1：反码一般不直接使用，通常是作为求补码的中间码

† 注2：补码的补码就是原码

补码运算

- 符号位可以作为数值参加运算
- 减法可以转化为加法运算
- 运算结果仍为补码

例：用 8 位字长计算 $67 - 10$

$$67_{10} = 01000011_2 [\text{原码}] \leftrightarrow 01000011 [\text{补码}]$$

$$-10_{10} = 10001010_2 [\text{原码}] \leftrightarrow 11110101 [\text{反码}] \leftrightarrow 11110110 [\text{补码}]$$

$$01000011 + 11110110 = 1\ 00111001 \leftrightarrow 00111001 [\text{补码}] \leftrightarrow 00111001_2 [\text{原码}] = 57_{10}$$

符号位加入正常运算，超出字长部分自然丢失



思考：用 8 位字长计算 $85 + 44$ ，会出现什么问题？

非数值信息

非数值信息的表示

□ 西文字符：ASCII 码

完整的 ASCII 码表见课程主页

□ 中文字符：一个汉字（含标点符号）占两个字节

常见编码有 GB2312、GBK、GB18030、UTF-8

2

算法基本概念

- 什么是算法
- 算法的特征与评价
- 算法的描述方法
- 基本控制结构

算法

程序 = 算法 + 数据结构

—— 计算机科学家 Nikiklaus Wirth, 1976

+ 程序设计方法 + 语言工具和环境



lucid, systematic,
and penetrating
treatment of basic
and dynamic data
structures, sorting,
recursive algorithms,
language structures,
and compiling

NIKLAUS WIRTH

Algorithms +
Data
Structures =
Programs

Pascal 之父，图灵奖获得者

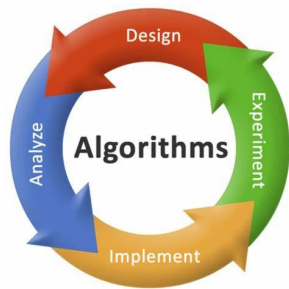

程序应该包括

- 对数据组织的描述：数据的类型和组织形式，即 **数据结构**
- 对操作流程的描述：即操作步骤，也就是 **算法**

DATA STRUCTURE
& ALGORITHM



算法：为解决某个问题而采取的方法和执行步骤。



- ▶ 学习程序设计的目并不是学习一种特定的语言，而是学习进行程序设计的一般方法
- ▶ 掌握了算法就是掌握了程序设计的灵魂，配合相关计算机语言，就能顺利编写出程序，解决相关的问题
- ▶ 但是，脱离了具体的语言去学习程序设计是困难的



算法特征与算法评价

算法的特征

- ❑ 输入：有零个或多个输入量
- ❑ 输出：通常有一个或以上输出量（计算结果）
- ❑ 明确性：算法的描述必须无歧义，保证算法的正确执行
- ❑ 有限性：有限个输入、有限个指令、有限个步骤、有限时间
- ❑ 有效性：又称可行性，能够通过有限次基本运算来实现

算法的评价

- ❑ 时间复杂度：所需的运算量 / 执行时间
- ❑ 空间复杂度：所占用的内存
- ❑ 实现复杂度：编程实现和后期升级维护等

Makes Good Algorithm

Correctness

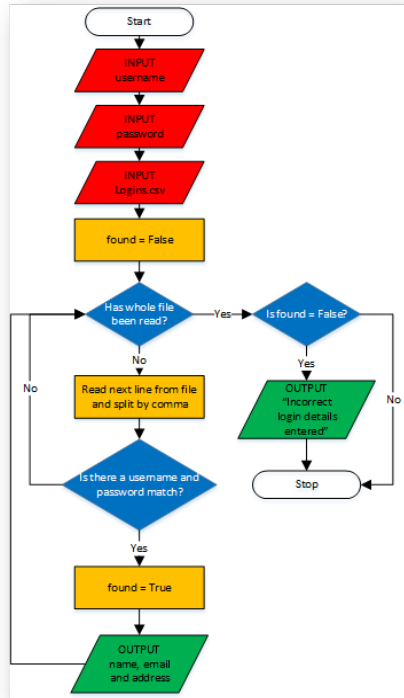


Efficiency



算法描述

算法的描述方法：自然语言、流程图、伪代码、... ..



流程图：简洁直观

算法 1.2 LU 分解

```
1: for  $k = 1$  to  $n - 1$  do
2:   for  $i = k + 1$  to  $n$  do
3:      $l_{ik} = a_{ik} / a_{kk}$     % 计算  $L$  的第  $k$  列
4:   end for
5:   for  $j = k$  to  $n$  do
6:      $u_{kj} = a_{kj}$     % 计算  $U$  的第  $k$  行
7:   end for
8:   for  $i = k + 1$  to  $n$  do
9:     for  $j = k + 1$  to  $n$  do
10:       $a_{ij} = a_{ij} - l_{ik}u_{kj}$     % 更新  $A(k + 1 : n, k + 1 : n)$ 
11:    end for
12:  end for
13: end for
```

伪代码：易于编程

算法基本结构

算法的三种基本控制结构：

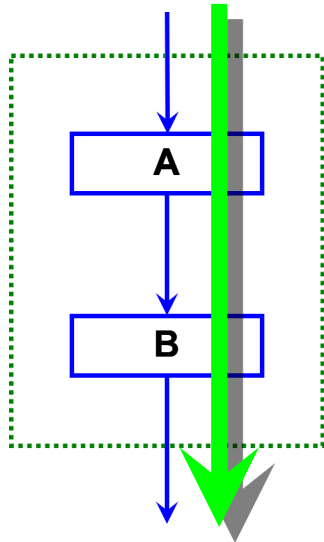
顺序结构、选择结构、循环结构

三种结构的基本要求

- ❑ 只有一个入口
- ❑ 只有一个出口
- ❑ 结构内每一部分都有机会被执行
- ❑ 结构内不能存在“死循环”

顺序结构

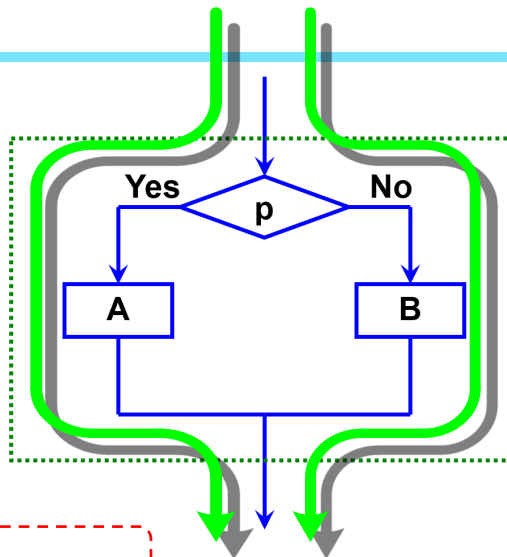
是最基本、也是最常用的程序设计结构，它按照程序语句行的自然顺序，一条一条地执行代码



选择结构

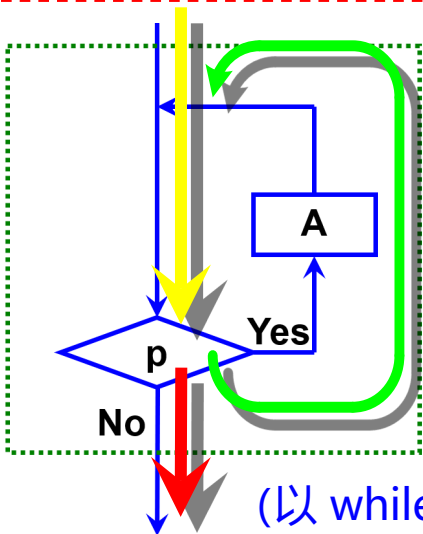
选择结构

也称分支结构或条件结构，根据条件，判断应该选择哪一条分支来执行，包括简单选择和多分支选择



循环结构

根据给定的条件，判断是否需要重复执行某一相同的程序段



(以 while 循环为例)

课后练习

1. 将下列二进制数转化为十进制数

101, 100111, 11010.011

2. 将下列十进制数转化为二进制数

101, 0.5625, 93.328125

思考

小数的补码如何计算？

课外阅读

IEEE 浮点运算标准 —— 计算机中浮点数是怎么表示和运算的