

第一章 C++语言概述

主讲：刘晓光
张海威 张莹
殷爱茹 李雨森
宋春瑶 沈玮
卢少平

南开大学

计算机学院&网络空间安全学院





课程安排

周课时安排

- 第一学期讲授3课时，实验4课时
- 第二学期讲授2课时，实验4课时

成绩

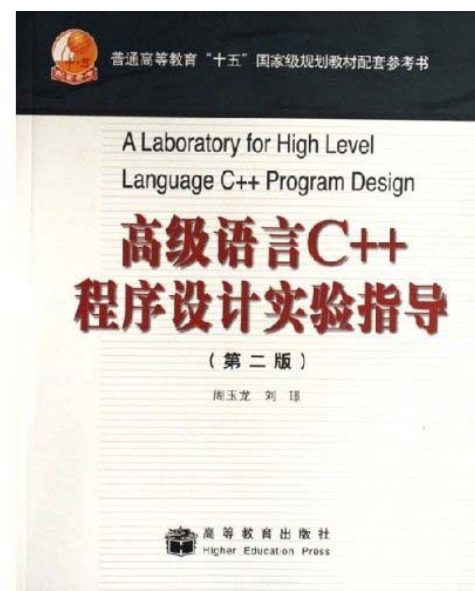
- 期末笔试成绩（50%）
- 上机考试成绩（30%）
- 平时成绩（包括出勤、作业、测验等，共20%）



教学用书

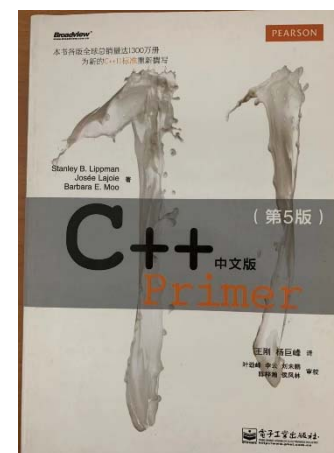
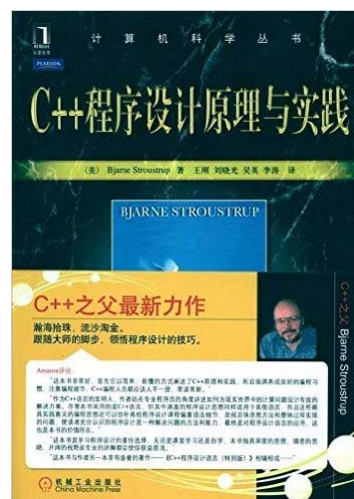
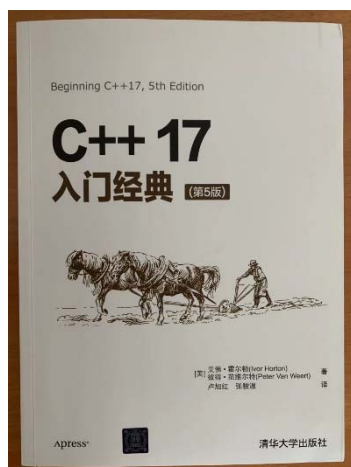
教材

- 《C++程序设计》，刘璟编著，高等教育出版社
- 《高级语言C++程序设计实验指导》，周玉龙、刘璟编著，高等教育出版社



参考资料

- 机械工业出版社，《C++17入门经典》，
- 机械工业出版社，《C++ 程序设计原理与实践》，
Bjarne Stroustrup 编著，。
- 电子工业出版社，《C++ Primer 中文版》，
S.B.Lippman,J.Lajoie编著，。





计算机与程序设计语言



数制转换与数据存储



C++语言的基本概念



C++语言的词汇



C++程序的结构



计算机与程序设计语言 ☒
数制转换与数据存储 ☐
C++语言的基本概念 ☐
C++语言的词汇 ☐

☐ 计算机的组成
☐ 计算机程序设计语言
☐ 程序设计方法学
☐ C++语言简史



计算机与程序设计语言



数制转换与数据存储



C++语言的基本概念



C++语言的词汇



C++程序的结构



计算机与程序设计语言 ☐
数制转换与数据存储 ☐
C++语言的基本概念 ☐
C++语言的词汇 ☐

☒ 计算机的组成
☐ 计算机程序设计语言
☐ 程序设计方法学
☐ C++语言简介



计算机 (Computer)

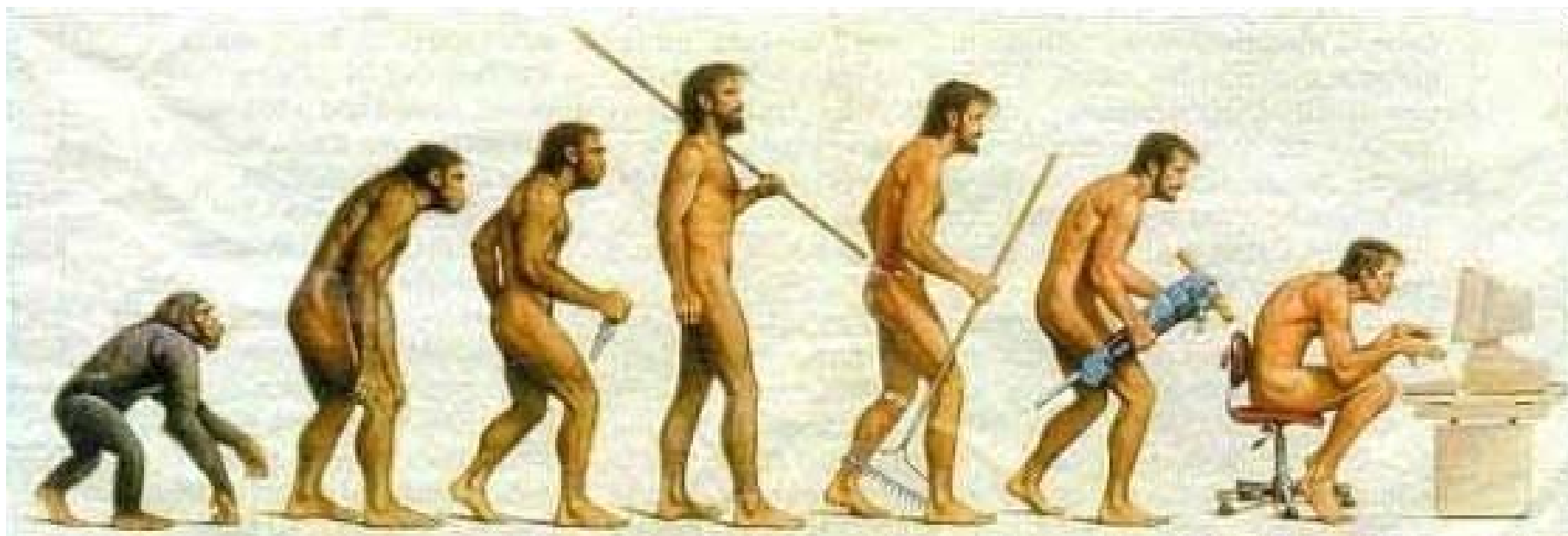


计算机与程序设计语言 ☐
数制转换与数据存储 ☐
C++语言的基本概念 ☐
C++语言的词汇 ☐

计算机的组成 ☒
计算机程序设计语言 ☐
程序设计方法学 ☐
C++语言简介 ☐



计算机 (Computer) Vs Tools



计算机发展

- 算盘：东汉时期
- 1623，德国人 **Schickard** 和 **Kepler** 发明第一台机械计算机

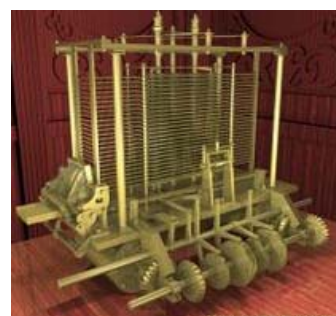


- 1674，**Leibniz** 发明的机械计算机可完成乘法和除法



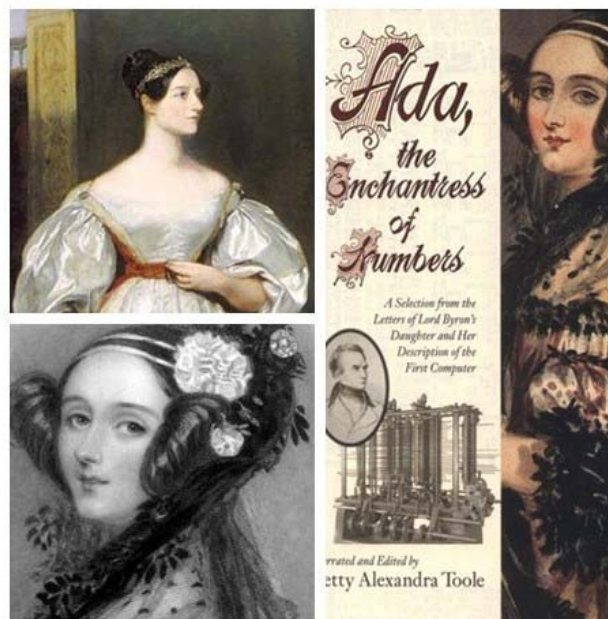
计算机发展

- 1823, 英国人 **Babbage** 发明差分机



- 世界上
第一个程序员

Augusta Ada King





现代计算机

• 希尔伯特纲领

20世纪初，逐步形成了关于数学基础研究的逻辑主义、直觉主义和形式主义三大流派。其中，形式主义流派的代表人物是数学家希尔伯特（D. Hilbert）。他在数学基础的研究中提出了一个设想，其大意是：将每一门数学的分支形式化，构成形式系统或形式理论，并在以此为对象的元理论

元数学中，证明每一个形式系统的相容性，从而导出全部数学的相容性，希尔伯特的这一设想，就是所谓的“**希尔伯特纲领**”。希尔伯特纲领的目标，其实质就是要寻找通用的形式逻辑系统，该系统应当是完备的，即在该系统中，可以机械地判定任何给定命题的真伪



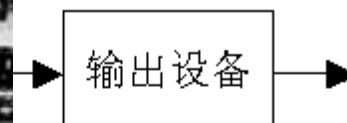
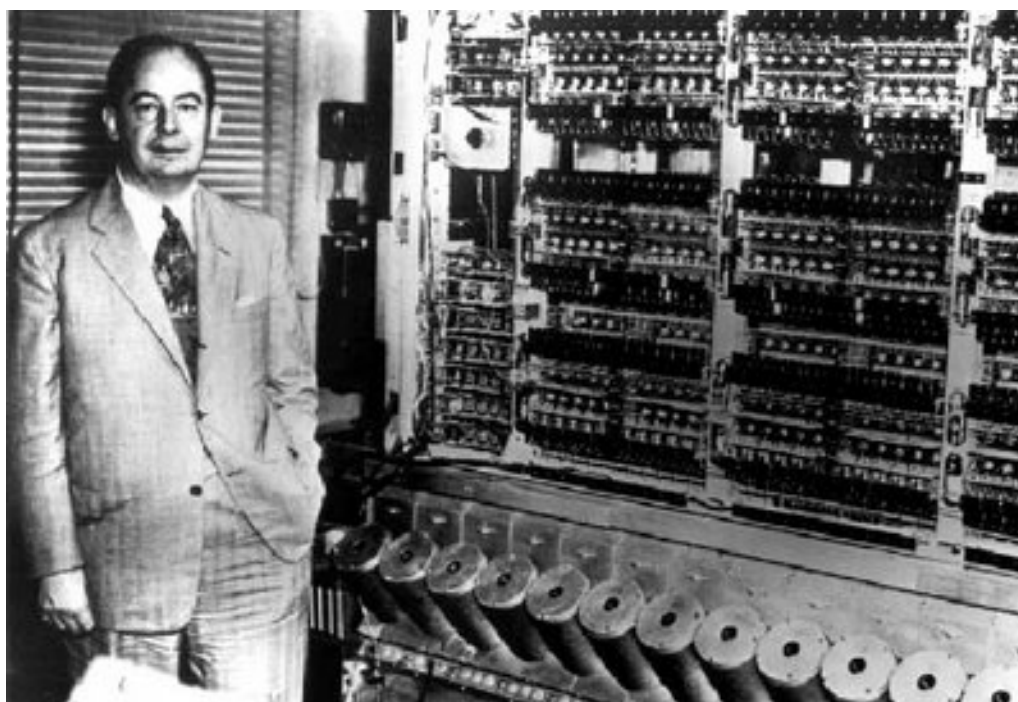
图灵 图灵机 ENIGMA



— the



计算机体系结构——冯·诺依曼结构





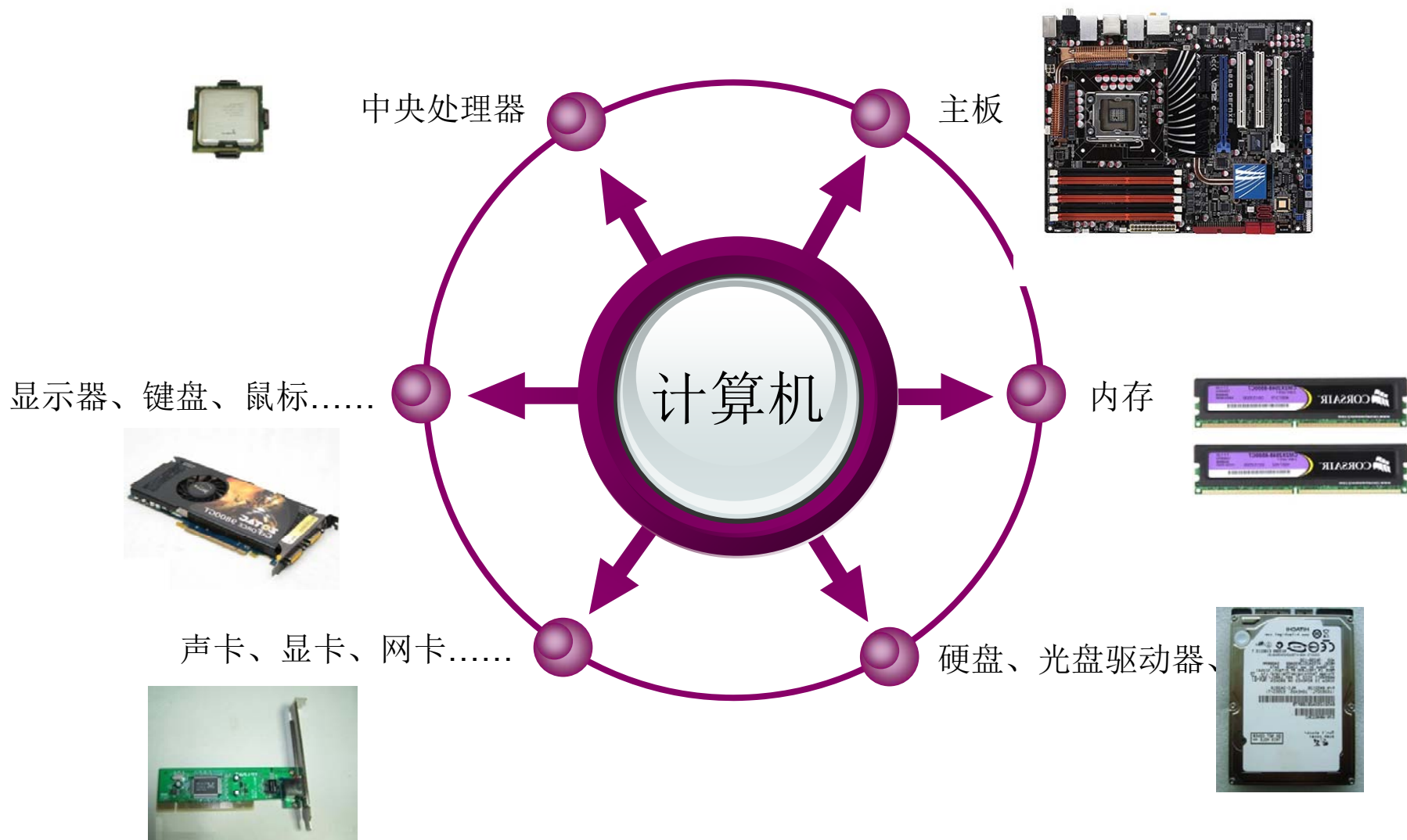
计算机的组成

- 存储器（RAM — Random Access Memory）：存储程序指令和数据，包括随机存取存储器和只读存储器（Read Only Memory）
- 中央处理器（CPU — Central Processing Unit）：又可细分为控制器（CU）和运算器（ALU），即，CPU = CU + ALU。
- 输入输出设备（I/O — Input / Output）：也称外部设备，负责对数据和程序进行输入与输出。



计算机与程序设计语言 ☐
数制转换与数据存储 ☐
C++语言的基本概念 ☐
C++语言的词汇 ☐

☒ 计算机的组成
☐ 计算机程序设计语言
☐ 程序设计方法学
☐ C++语言简介





计算机的组成

三级存取器

- CPU二级缓存
- 内存
- 外存
 - 硬盘
 - 光盘
 - 软盘
 - 磁带
 -





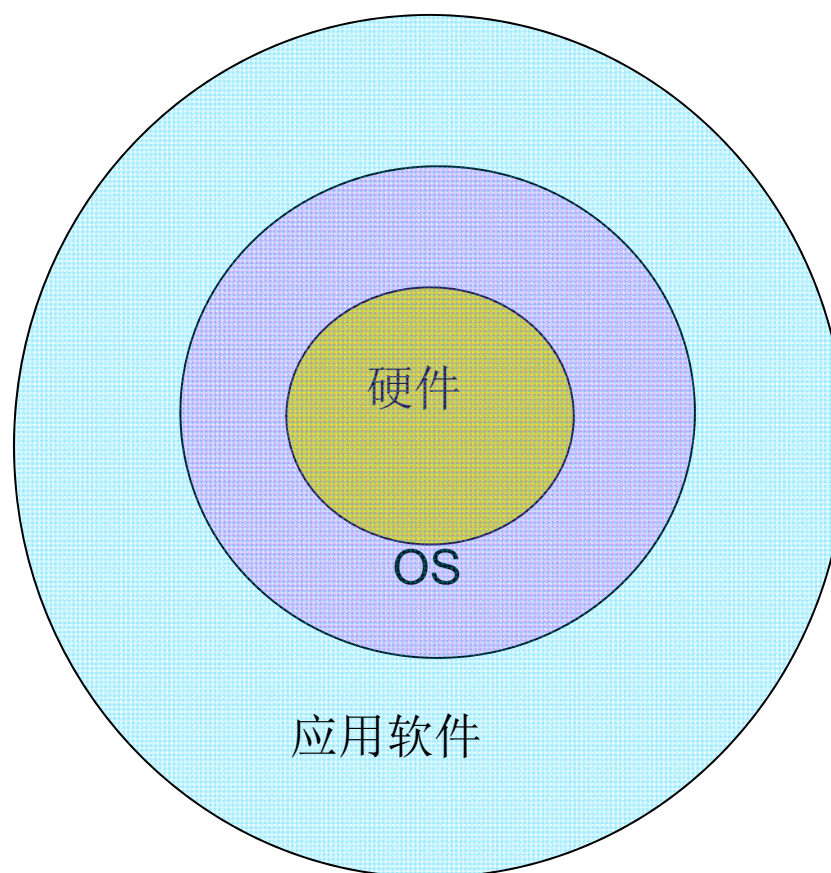
计算机的组成

计算机软件

- 系统软件
 - 操作系统（Operating System）
 - Windows
 - Unix
 -
- 应用软件
 - Office系列
 - Media Player
 - Internet Explorer
 - 魔兽争霸



计算机的组成





程序设计

程序的含义

- 要计算机完成某一任务所规定的一系列动作或步骤

程序在计算机系统中的地位

- 软件的根基

程序设计的基础——计算机语言

- 计算机指令系统：机器语言
- 低级编程语言
- 高级程序设计语言





机器指令

计算机设计者把计算机可以完成的动作编辑成一个机器指令表，并为每种动作赋予一个二进制代码，通常由指令码（操作码）和内存地址（操作数）来构成。通过机器指令来编写的程序称为机器语言程序。





机器指令

机器语言示例

- 使用“机器语言”编出的做一次加法“TOTAL = PRICE + TAX”的程序为：

```
156C          // 取6C内容送寄存器5
166D          // 取6D内容送寄存器6
5056          // 把二值相加，结果送寄存器0
306E          // 把寄存器0中的结果送地址6E
C000          // 停机
```





机器指令

机器指令的执行

- 在内存中执行
- 举例：计算： $5+15=?$
 - 将0010H存储单元的数据（5）取出，存放在ALU。
 - 将0011H存储单元的数据（15）取出，与 ALU 的数据相加，运算结果存放在ALU。
 - 将 ALU 中的数据（20）存放到0012H存储单元。
 - 停止执行
- 计算机依赖**机器指令**运行，机器指令以及各种被处理的数据都以**二进制形式**存储。





低级编程语言

汇编语言

- 引入“助记符”
 - ADD、SUB、MOV、.....
- 汇编程序系统(Assembler)
 - 汇编语言源程序 \longrightarrow 机器语言

低级语言的缺点

- 依赖于机器，可移植性差
- 代码冗长，不易于编写大规模的程序
- 可读性差
- 可维护性差





高级程序设计语言

高级语言

- 易于理解、记忆和使用
- 更加接近人的思维方式和自然语言
- 应用广泛的高级语言：
 - FORTRAN、ALGOL、COBOL、BASIC、PASCAL、C、LISP、PROLOG, C++, C#, Java等





高级程序设计语言

高级语言程序的运行

- 编译程序系统
 - 编译
 - 将高级语言源程序转换为汇编语言源程序（目标程序）
 - 连接
 - 将目标程序转换为机器指令程序（可执行程序）
 - 执行
 - 执行可执行程序，得到所需的结果





程序设计方法学的发展

程序设计技术的初级阶段

- 计算机诞生，von Neumann 模式形成，低级语言编程是主要开发形式。
- 第一代高级语言（以 FORTRAN 和ALGOL60 为代表）诞生，从低级语言编程转向高级语言编程。
 - 高级语言的出现使得程序设计的难度降低，导致了计算机应用在五六十年代的发展进入新的阶段。
 - 60年代，以大规模程序频频出错(例如1962年，因软件出错导致美国金星探测器水手II号卫星发射失败)为特征的“软件危机”发生，引起关于“Goto语句”的辩论。





程序设计方法学的发展

结构程序设计（Structured Programming）阶段

- 以Pascal 语言和C语言为代表
 - 强调数据类型、程序结构
 - 注重可靠性、可维护性
- 主要特点
 - 采用自顶向下、逐步求精的设计方法
 - 程序运行的动态结构与程序书写的静态结构相对一致
 - 严格区分数据类型
- 缺点
 - 程序的可重用性差





面向对象程序设计阶段

80年代，面向对象程序设计逐渐从理论转向实践，程序设计理论步入成熟期。

- A.Kay 研制了Smalltalk 语言
- B.Stroustrup 开发了C++ 语言

OOP方法在90年代盛行

- OOP方法从思想上与SP方法相比是抓住了软件开发的本质和规律，计算机所要解决的问题越来越重要，越来越复杂。
- OOP技术之所以能适应今天软件产业的需要，是因为它比较好地解决了软件模块化、信息隐蔽和抽象的目标。





面向对象程序设计阶段

常用的面向对象语言

- C++
- Visual C++
 - 基于Windows窗体规范的C++的具体实现版本
- Smalltalk
- Simula67
- LISP家族语言
- Java
- C#





面向对象程序设计阶段

面向对象程序设计方法的特点

- 将数据以及对这些数据进行操作的方法放在一起，形成一个相互依存、不可分离的整体——对象。
- 通过对事物的抽象找出同一类对象的共同属性(静态特征)和行为(动态特征)，从而形成类。类是面向对象程序设计方法中的程序主体，类中的大多数数据只能用本类的方法进行处理，以保障程序模块的独立性以及数据的安全性。类通过一个简单的公共对外接口与外界发生联系，对象与对象之间通过消息进行通讯。
- 面向对象程序设计的三大特征是：封装性、继承性、多态性。



面向对象程序设计阶段

C++对面向对象程序设计方法的支持

- 支持数据封装
 - C++语言中的类(class)是支持数据封装的工具。通过类(class)类型对所要处理的问题进行抽象描述，从而将逻辑上相关的数据与函数进行封装。
- 支持继承性
 - C++语言允许单继承和多继承。类之间可形成多层次的派生以及继承关系。
- 支持多态性
 - 允许对函数和运算符进行重载。通过在基类及其派生类间对虚函数进行使用体现出另一种多态性。



程序设计的范型

命令型程序设计

- 过程型程序设计，指令序列

OOP程序设计

- 组合成类或对象

函数性程序设计

- “黑盒子”方式

逻辑性程序设计

- 申述型程序设计





C++语言的优势

为什么选择C++语言

- 面向对象程序设计正在逐渐成为主流设计技术
- OOP 技术并不取代SP 和一般的程序设计的技能技巧
- 由于各大公司的竞相开发，C++语言在各种不同机型上都有优秀的编译系统和相关的环境与工具
- C++语言最可能取代C 而成为主流的软件开发语言之一
- C++语言已成为计算机专业主要的教学语言

课后阅读： **Science:改改python代码，速度提高6万倍**





C++语言简史

C++语言的作者是美国AT&T公司Bell实验室的Bjarne Stroustrup

- 带类的C
- C++语言的诞生
 - 1985年, C++1.0版
- C++语言的发展
 - 1989年, C++2.0版
 - 1993年, C++3.0版
 - 1998年, C++标准诞生
 - 2011年, C++11标准
 - 2014年, C++14标准
 - 2017年, C++17标准
 - 2020年, ? 标准





C++语言的特点

C++语言是支持面向对象程序设计的最主要的代表语言之一。

- 封装和信息隐藏
- 抽象数据类型
- 继承和派生
- 函数与运算符的重载
- 模板

C++语言是程序员和软件开发者在实践中的创造，时时处处体现了面向实用，面向软件开发者的思想

C++语言是C语言的超集



计算机与程序设计语言 ☐
数制转换与数据存储 ☒
C++语言的基本概念 ☐
C++语言的词汇 ☐

☐ 数的进制
☐ 数制转换
☐ 数据存储



计算机与程序设计语言



数制转换与数据存储



C++语言的基本概念



C++语言的词汇



C++程序的结构





开放问题

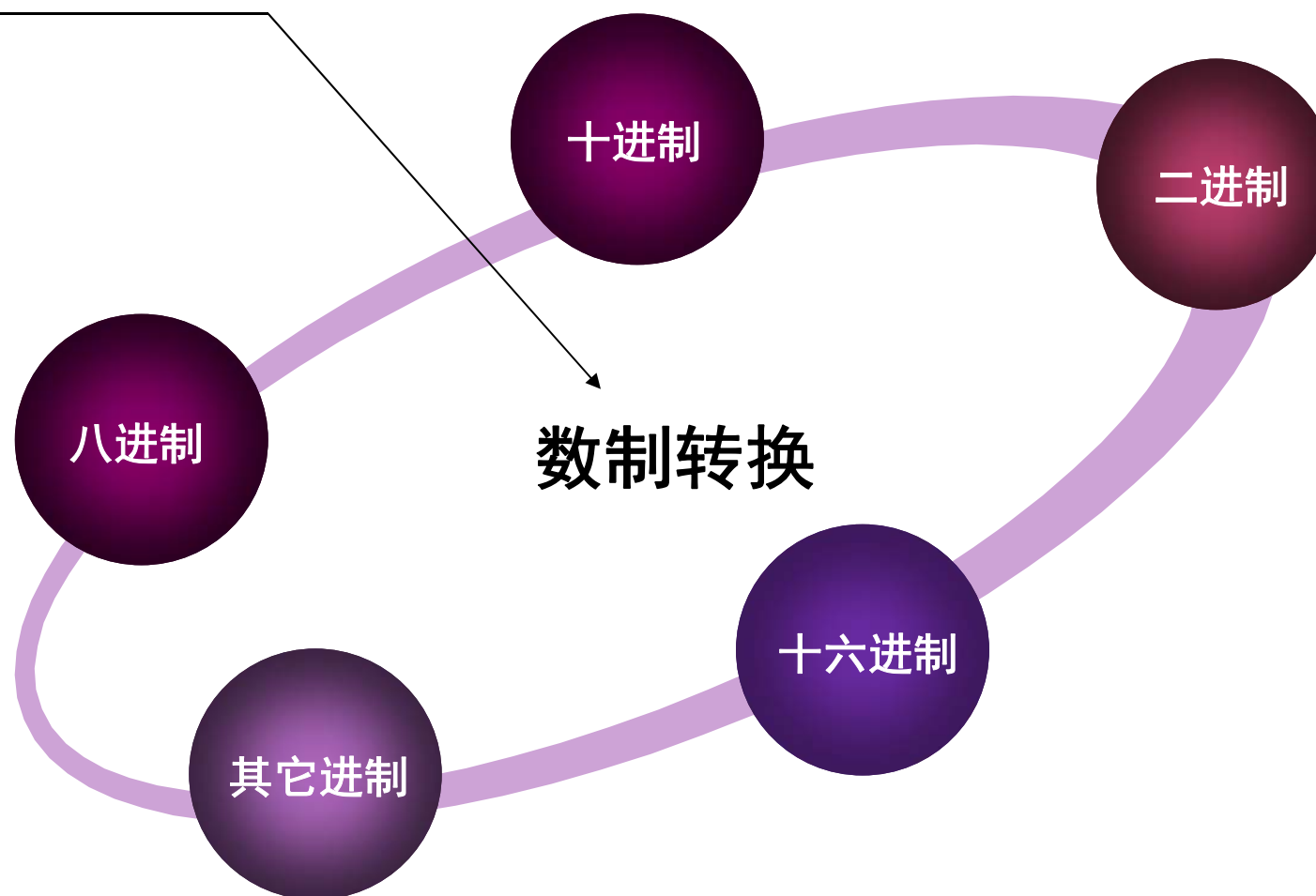
进制是计数的规则

除了十进制以外我们日常生活中都有哪些进制？

算盘是什么进制？



数的进制

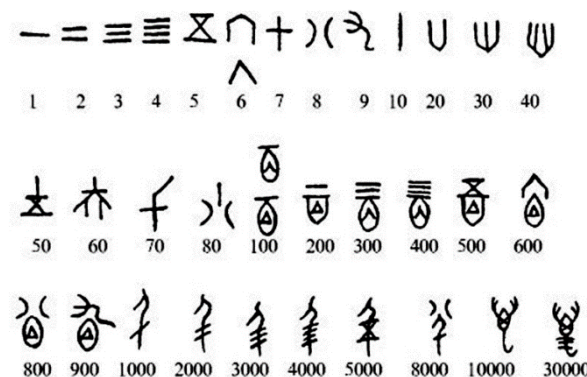
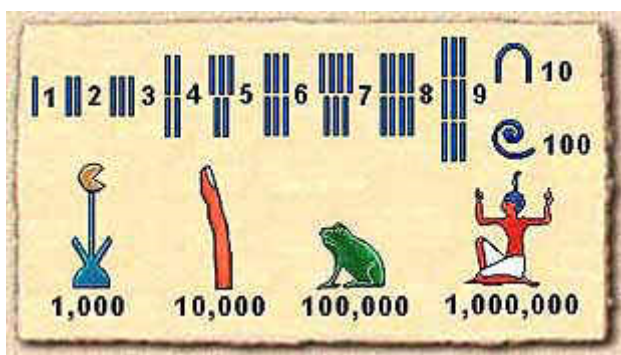


十进制

- 亚里士多德：十进制，因为绝大多数人生来就有10根手指
- 对美洲印第安部落研究证实了这一点

将近三分之一的人在使用十进制，另有大约三分之一的人在使用五进制或者五进制和十进制混合使用，剩下的不到三分之一的人在使用二进制，而那些使用三进制的人则不到百分之一。二十进制（以20为基数），出现在大约百分之十的部落中作者：不知了链接：

<https://www.jianshu.com/p/f87481c6ff11>来源：简书著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。



- 5000年前埃及

- 3000年前甲骨文

二十进制

- 来自手+脚
- 玛雅文化
- 法语vingt是20的意思， quatre vingt是四个20，意思是80

0	1	2	3	4
	•	••	•••	••••
5	6	7	8	9
—	•	••	•••	••••
10	11	12	13	14
==	•	••	•••	••••
15	16	17	18	19
===	•	••	•••	••••

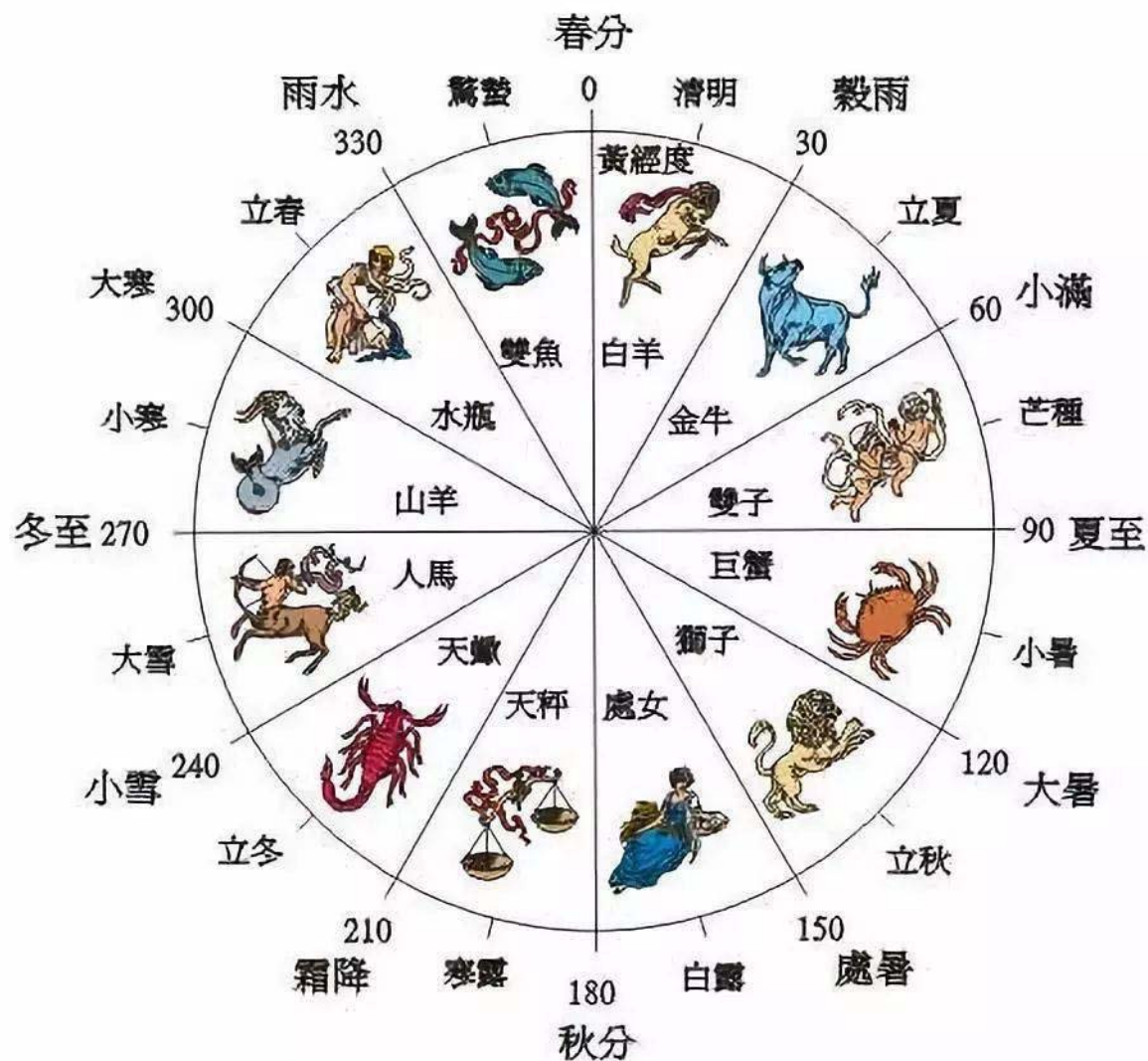
•	× 8000=	8000
•	× 400=	400
•	× 20=	20
•	× 1=	1
		8421



雅玛数字有两套表示方式：横点和头像。竖式位算法，20进制。

十二进制

• 时间



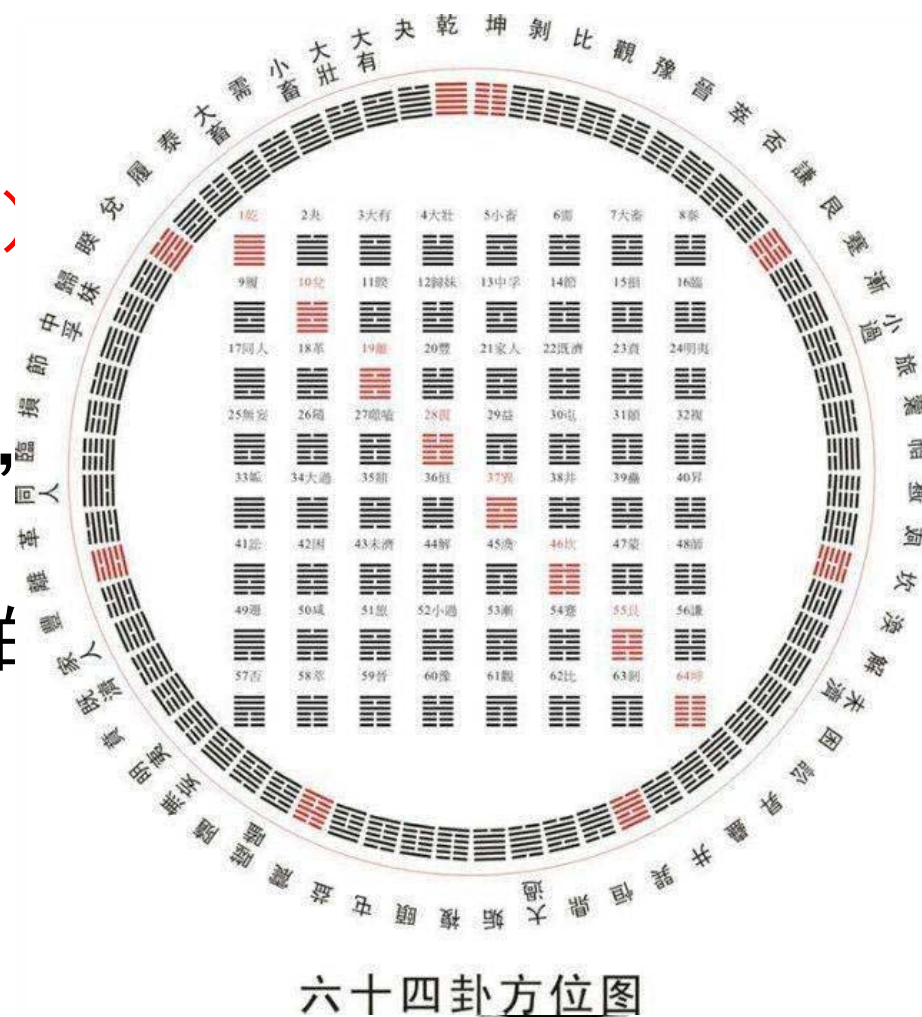
二进制数

每个二进制数位 (bit)

逢2进1

每个二进制数位的权,
 $n=0,1,2,3, \dots$)

据说数学家莱布尼兹
 明了二进制





二进制数

二进制数的表示方法

- 1011 (2)

转换为十进制数的方法

- 按“权”展开

$$- 1011(2) = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 0 + 2 + 1 = 11(10)$$



八进制数

每个八进制数位只能是数字：0、1、2、...、7

逢8进1

每位的权（weight）从右往左依次为： 8^n （ $n = 0, 1, 2, 3 \dots$ ）。

表示方法

- 1011(8)

$$\begin{aligned} - 1011(8) &= 1 \times 8^3 + 0 \times 8^2 + 1 \times 8^1 + 1 \times 8^0 = 512 + 0 + 8 + 1 \\ &= 521(10) \end{aligned}$$



十六进制数

每个十六进制数位只能够出现：0、1、2、...、9、A、B、C、D、E、F。

逢16进1

每位的权（weight）从右往左依次为： 16^n （ $n = 0, 1, 2, 3 \dots$ ）。

表示方法

- 1011(16)

$$\begin{aligned} - 1011(16) &= 1 \times 16^3 + 0 \times 16^2 + 1 \times 16^1 + 1 \times 16^0 = 4096 + 0 \\ &+ 16 + 1 = 4113(10) \end{aligned}$$

十六进制数

每个十六进制数位只能够出现：0、1、2、...、9、A、B、C、D、E、F。

逢16进1

每位的权（weight）从右往左依次为： 16^n （ $n = 0, 1, 2, 3 \dots$ ）。

表示方法

- 1011(16)

$$\begin{aligned} - 1011(16) &= 1 \times 16^3 + 0 \times 16^2 + 1 \times 16^1 + 1 \times 16^0 = 4096 + 0 \\ &+ 16 + 1 = 4113(10) \end{aligned}$$



数制转换

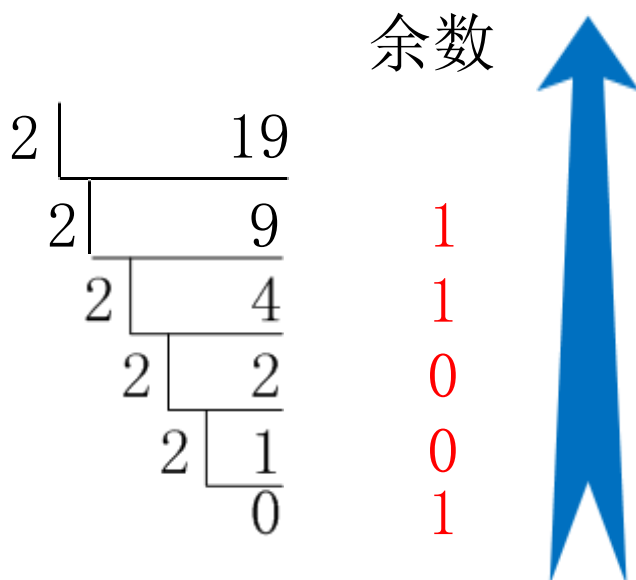


数制转换

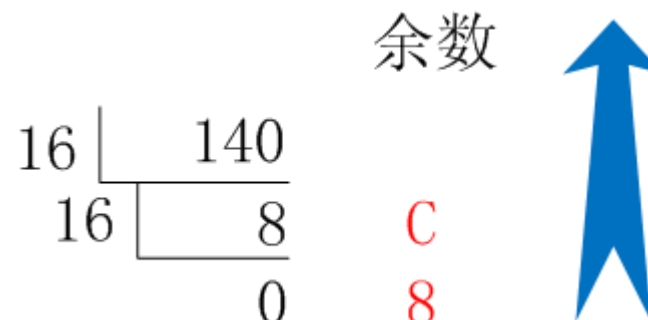
十进制转换为R进制

- 除R取余，倒排余数

– 19 (10) = ~~10~~ 1 0 1 1



• 140 (10) = 8C (16)



数制转换

二进制转换为十进制

- 按“权”展开，累加求和

二进制转换为八进制

- 从小数点位置，整数向左，小数向右，每三位二进制数为一组，不足三位补0
 - $1001_2 \Rightarrow (001)(001)$
- 将每组三位二进制数转换为相应的八进制数并按顺序排列在一起
 - $101010111_2 \Rightarrow (101)(010)(111) \Rightarrow 527_8$



数制转换

二进制与八进制数对照表

二进制数	八进制数
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7



数制转换

二进制转换为十六进制

- 从小数点位置，整数向左，小数向右，每四位二进制数为一组，不足四位补0
 - $10010(2) \Rightarrow (0001)(0010)$
- 将每组四位二进制数转换为相应的十六进制数并按顺序排列在一起
 - $100111110010(2) \Rightarrow (1001)(1111)(0010) \Rightarrow 9F2(16)$

数制转换

二进制与十六进制数对照表

二进制数	十六进制数	二进制数	十六进制数
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F





数制转换

八进制转换二进制

- 一位化三位，按序连一起
 - $527(8) \Rightarrow (101)(010)(111) \Rightarrow 101010111(2)$

十六进制转换二进制

- 一位化四位，按序连一起
 - $9F2(16) \Rightarrow (1001)(1111)(0010) \Rightarrow 100111110010(2)$





数制转换

八进制转换十六进制

- 先转换成十进制，再转换为十六进制

十六进制转换八进制

- 先转换成十进制，再转换为八进制





数制转换

十进制小数转换为二进制数

- 整数部分换算
 - 除2取余，余数倒排
- 小数部分换算
 - 乘2取整，小数部分继续乘2取整
 - 直到小数部分为0 或者达到指定的精度为止
- 十进制小数转换为二进制多数情况下得到近似数





数制转换

十进制小数转换为二进制数

- $0.6875 (10) = 0.1011 (2)$
 - $0.6875 * 2 = 1.3750$ 取个位数1
 - $0.375 * 2 = 0.75$ 取个位数0
 - $0.75 * 2 = 1.5$ 取个位数1
 - $0.5 * 2 = 1.0$ 取个位数1
- $0.33 (10) = 0.0101..... (2)$
 - $0.33 * 2 = 0.66$ 取个位数0
 - $0.66 * 2 = 1.32$ 取个位数1
 - $0.32 * 2 = 0.64$ 取个位数0
 - $0.64 * 2 = 1.28$ 取个位数1，取小数部分0.28继续计算





数制转换

十进制负数转换为二进制数

- 与计算机中整数的存储格式有关
- 原码
 - 一个整数，按照绝对值大小转换成的二进制数
- 反码
 - 将二进制数按位取反，所得的新二进制数称为原二进制数的反码
- 补码
 - 反码加1称为补码
- 符号位





数制转换

十进制负数转换为二进制数

- 将十进制负数-22转换为二进制数
 - 将22转换为二进制数
 - 10110
 - 补足4个字节的二进制数位（根据系统中整数存储的位数）
 - 00000000 00000000 00000000 00010110
 - 按位取反
 - 11111111 11111111 11111111 11101001
 - 加1
 - 11111111 11111111 11111111 11101001+1
 - =11111111 11111111 11111111 11101010





数制的选择

计算机内部采用二进制

- 技术实现简单，计算机是由逻辑电路组成，逻辑电路通常只有两个状态，开关的接通与断开，这两种状态正好可以用“1”和“0”表示。
- 简化运算规则：两个二进制数和、积运算组合各有三种，运算规则简单，有利于简化计算机内部结构，提高运算速度。
- 适合逻辑运算：逻辑代数是逻辑运算的理论依据，二进制只有两个数码，正好与逻辑代数中的“真”和“假”相吻合。
- 易于进行转换，二进制与十进制数易于转换。
- 八进制和十六进制是计算机的辅助数进制，用于缩短二进制数的长度



使用二进制数





存储器单元

字节 (Byte)

- 8个二进制数位 (bit) 为一个字节
- 1个字节可以表示的范围
 - 0 (00000000) 至255 (11111111)
- 字节是计算机中指令和数据的基本存储单元
 - 不同类型的指令使用不同的字节数
 - 不同类型的数据用不同的字节数存储

十六进制数表示地址

--	--	--	--	--	--	--	--



存储器单元

存储器单元

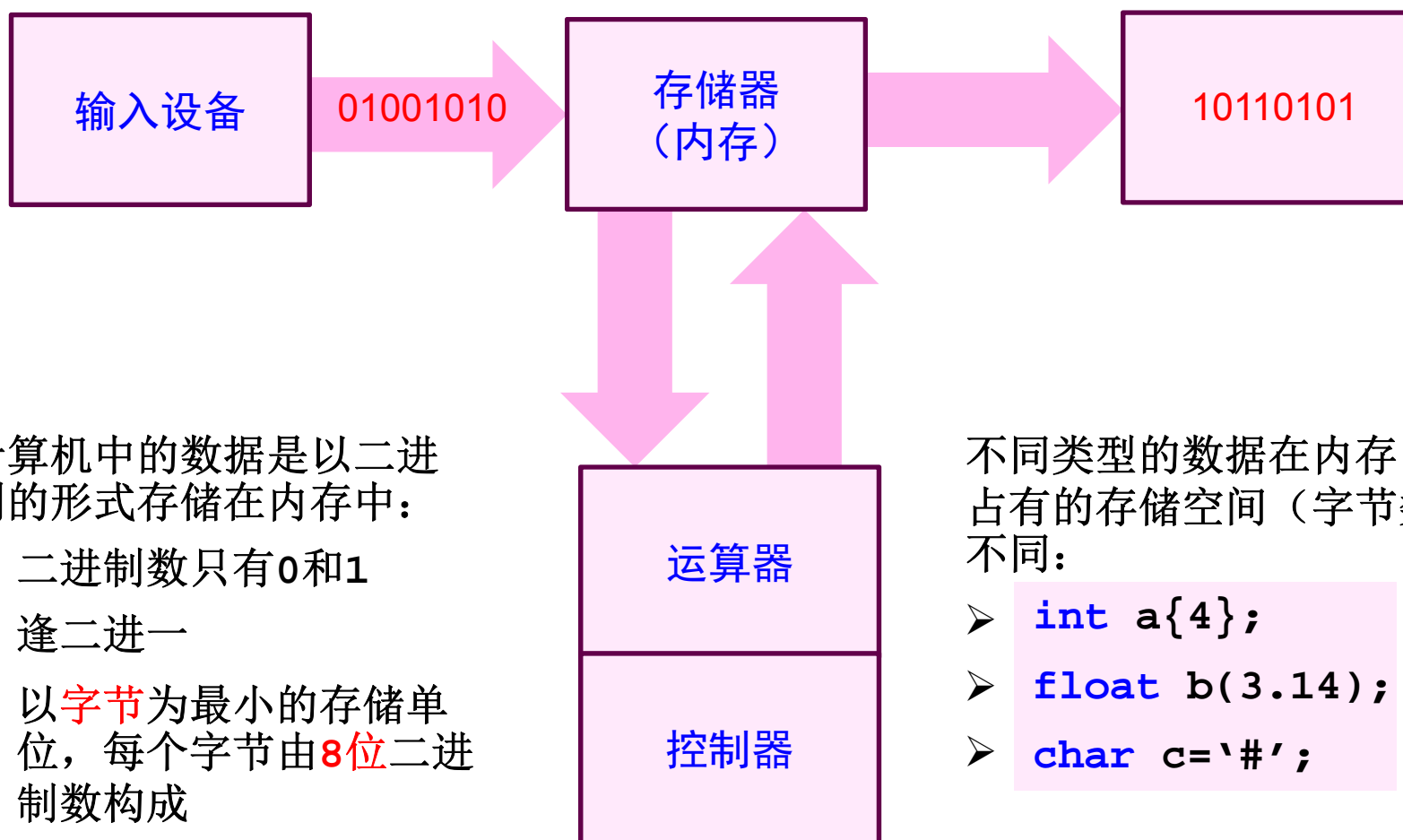
- 按字节安排

存储地址

- 顺序号，为每个存储器单元指定一个序号

1008							
1009							
100A							
100B							

数据存储与表示



计算机与程序设计语言 ☐
数制转换与数据存储 ☐
C++语言的基本概念 ☒
C++语言的词汇 ☐

☐ 源文件与头文件
☐ 注释与空白
☐ 预处理指令
☐ 函数

☐ 语句
☐ 数据的输入与输出
☐ 命名空间
☐ 标识符与关键字

☐ 类和对象
☐ 模板



计算机与程序设计语言



数制转换与数据存储



C++语言的基本概念



C++语言的词汇



C++程序的结构





源文件与头文件

源文件

- 扩展名为.cpp
- 包含函数和全部可执行代码

头文件

- 扩展名为.h
- 标准库的头文件不带扩展名
- 包含许多内容
 - 函数原型
 - 类定义
 - 模板定义

```
#define _CRT_SECURE_NO_WARNINGS

#include<iostream>
#include<cstring>
#include<iomanip>
using namespace std;
enum Gender {M, F}; //枚举类型
class Student
{
    int id;
    char* name;
    Gender gender;
public:
    Student(int i, char* n, Gender g)
    {
        id = i;
        name = new char[strlen(n) + 1];
        strcpy(name, n);
        gender = g;
    }
    void print()
    {
        cout.setf(ios::left);
        cout << setw(4) << id << setw(8) << name;
        if (gender == M)
            cout << setw(4) << "男";
        else
            cout << setw(4) << "女";
    }
};

void main()
{
    Student stu[3] = { Student(1, "Zhang", M), Student(2, "Li", F), Student(3, "Wang", M) };
    cout.setf(ios::left);
    cout << setw(4) << "ID" << setw(8) << "Name" << setw(4) << "Gender" << endl;
    for (Student si : stu) {
        si.print();
        cout << endl;
    }
}
```



注释和空白

注释

- 编译器忽略
- 一行或多行
 - //
 - /*和*/

空白

- 空格、制表符、换行符和换页符的任意序列
- 编译器一般会忽略

```
#define _CRT_SECURE_NO_WARNINGS

#include<iostream>
#include<cstring>
#include<iomanip>
using namespace std;
enum Gender {M, F} //枚举类型
class Student
{
    int id;
    char* name;
    Gender gender;
public:
    Student(int i, char* n, Gender g)
    {
        id = i;
        name = new char[strlen(n) + 1];
        strcpy(name, n);
        gender = g;
    }
    void print()
    {
        cout.setf(ios::left);
        cout << setw(4) << id << setw(8) << name;
        if (gender == M)
            cout << setw(4) << "男";
        else
            cout << setw(4) << "女";
    }
};

void main()
{
    Student stu[3] = { Student(1, "Zhang", M), Student(2, "Li", F), Student(3, "Wang", M) };
    cout.setf(ios::left);
    cout << setw(4) << "ID" << setw(8) << "Name" << setw(4) << "Gender" << endl;
    for (Student si : stu) {
        si.print();
        cout << endl;
    }
}
```



预处理指令

预处理指令

- 以某种方式修改源代码，并编译为可执行的形式
 - #include
 - #define
 - ...

```
#define _CRT_SECURE_NO_WARNINGS

#include<iostream>
#include<cstring>
#include<iomanip>
using namespace std;
enum Gender {M,F}; //枚举类型
class Student
{
    int id;
    char* name;
    Gender gender;
public:
    Student(int i, char* n, Gender g)
    {
        id = i;
        name = new char[strlen(n) + 1];
        strcpy(name, n);
        gender = g;
    }
    void print()
    {
        cout.setf(ios::left);
        cout << setw(4) << id << setw(8) << name;
        if (gender == M)
            cout << setw(4) << "男";
        else
            cout << setw(4) << "女";
    }
};

void main()
{
    Student stu[3] = { Student(1, "Zhang", M), Student(2, "Li", F), Student(3, "Wang", M) };
    cout.setf(ios::left);
    cout << setw(4) << "ID" << setw(8) << "Name" << setw(4) << "Gender" << endl;
    for (Student si : stu) {
        si.print();
        cout << endl;
    }
}
```





函数

函数是命名的代码块，
执行定义好的操作
每个C++程序至少包含一个函数

- 主函数：程序的入口

函数的优点：

- 程序被分解为不同的单元，易于开发和测试
- 一个函数可以在程序的几个不同的地方重用，避免了代码的重复
- 一个函数可以在不同程序中重用
- 函数是程序的子结构

```
#define _CRT_SECURE_NO_WARNINGS
```

```
#include<iostream>
#include<cstring>
#include<iomanip>
using namespace std;
enum Gender {M, F}; //枚举类型
class Student
```

```
{
    int id;
    char* name;
    Gender gender;
public:
```

```
    Student(int i, char* n, Gender g)
    {
        id = i;
        name = new char[strlen(n) + 1];
        strcpy(name, n);
        gender = g;
    }
```

```
    void print()
    {
        cout.setf(ios::left);
        cout << setw(4) << id << setw(8) << name;
        if (gender == M)
            cout << setw(4) << "男";
        else
            cout << setw(4) << "女";
    }
```

```
};

void main()
{
    Student stu[3] = { Student(1, "Zhang", M), Student(2, "Li", F), Student(3, "Wang", M) };
    cout.setf(ios::left);
    cout << setw(4) << "ID" << setw(8) << "Name" << setw(4) << "Gender" << endl;
    for (Student si : stu) {
        si.print();
        cout << endl;
    }
}
```




语句 (Statement)

语句是C++程序的基本单元
分号表示语句的结束

C++语句的分类

- 标签语句 (Labeled statement)
- 表达式语句 (Expression statement)
- 复合语句或语句块 (Compound statement or block)
- 选择语句 (Selection statement)
- 循环语句 (Iteration statement)
- 转向语句 (Jump statement)
- 说明语句 (Declaration statement)

```
#define _CRT_SECURE_NO_WARNINGS

#include<iostream>
#include<cstring>
#include<iomanip>
using namespace std;
enum Gender {M, F}; //枚举类型
class Student
{
    int id;
    char* name;
    Gender gender;

public:
    Student(int i, char* n, Gender g)
    {
        id = i;
        name = new char[strlen(n) + 1];
        strcpy(name, n);
        gender = g;
    }
    void print()
    {
        cout.setf(ios::left);
        cout << setw(4) << id << setw(8) << name;
        if (gender == M)
            cout << setw(4) << "男";
        else
            cout << setw(4) << "女";
    }
};

void main()
{
    Student stu[3] = { Student(1, "Zhang", M), Student(2, "Li", F), Student(3, "Wang", M) };
    cout.setf(ios::left);
    cout << setw(4) << "ID" << setw(8) << "Name" << setw(4) << "Gender" << endl;
    for (Student si : stu) {
        si.print();
        cout << endl;
    }
}
```



数据的输入输出

使用输入输出流执行

- 输入输出流类对象

- cin

- cout

插入运算符<<

提取运算符>>

格式控制

```
#define _CRT_SECURE_NO_WARNINGS
```

```
#include<iostream>
#include<cstring>
#include<iomanip>
using namespace std;
enum Gender {M, F}; //枚举类型
class Student
{
    int id;
    char* name;
    Gender gender;
public:
    Student(int i, char* n, Gender g)
    {
        id = i;
        name = new char[strlen(n) + 1];
        strcpy(name, n);
        gender = g;
    }
    void print()
    {
        cout.setf(ios::left);
        cout << setw(4) << id << setw(8) << name;
        if (gender == M)
            cout << setw(4) << "男";
        else
            cout << setw(4) << "女";
    }
};

void main()
{
    Student stu[3] = { Student(1, "Zhang", M), Student(2, "Li", F), Student(3, "Wang", M) };
    cout.setf(ios::left);
    cout << setw(4) << "ID" << setw(8) << "Name" << setw(4) << "Gender" << endl;
    for (Student si : stu) {
        si.print();
        cout << endl;
    }
}
```

计算机与程序设计语言 ☐
数制转换与数据存储 ☐
C++语言的基本概念 ☒
C++语言的词汇 ☐

☐ 源文件与头文件
☐ 注释与空白
☐ 预处理指令
☐ 函数

☐ 语句
☐ 数据的输入与输出
☒ 命名空间
☐ 标识符与关键字

☐ 类和对象
☐ 模板



命名空间

```
#include <cstring>
#include <iomanip>
#include <vector>

enum Gender {M, F};
class Student
{
    int id;
    char* name;
    Gender gender;
public:
    Student(int i, char* n, Gender g)
    {
        id = i;
        name = new char[strlen(n) + 1];
        strcpy(name, n);
        gender = g;
    }
    void print()
    {
        std::cout.setf(std::ios::left);
        std::cout << std::setw(4) << id << std::setw(8) << name;
        if (gender == M)
            std::cout << std::setw(4) << "男";
        else
            std::cout << std::setw(4) << "女";
    }
};

void main()
{
    Student stu[3] = { Student(1, "Zhang", M), Student(2, "Li", F), Student(3, "Wang", M) };
    std::vector<Student> stu_vec;
    std::cout.setf(std::ios::left);
    std::cout << std::setw(4) << "ID" << std::setw(8) << "Name" << std::setw(4) << "Gender" << std::endl;
    for (Student si : stu) {
        si.print();
        std::cout << std::endl;
    }
}
```

```
#define _CRT_SECURE_NO_WARNINGS

#include <iostream>
#include <cstring>
#include <iomanip>
using namespace std;
enum Gender {M, F}; //枚举类型
class Student
{
    int id;
    char* name;
    Gender gender;
public:
    Student(int i, char* n, Gender g)
    {
        id = i;
        name = new char[strlen(n) + 1];
        strcpy(name, n);
        gender = g;
    }
    void print()
    {
        cout.setf(ios::left);
        cout << setw(4) << id << setw(8) << name;
        if (gender == M)
            cout << setw(4) << "男";
        else
            cout << setw(4) << "女";
    }
};

void main()
{
    Student stu[3] = { Student(1, "Zhang", M), Student(2, "Li", F), Student(3, "Wang", M) };
    cout.setf(ios::left);
    cout << setw(4) << "ID" << setw(8) << "Name" << setw(4) << "Gender" << endl;
    for (Student si : stu) {
        si.print();
        cout << endl;
    }
}
```



命名空间

也称为名字空间（namespace）

C++语言的新标准引入的概念

由用户命名的作用域，解决大型程序中标识符重名的问题

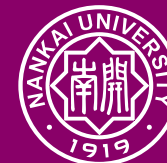
说明语句格式

- namespace <标识符>{<若干说明或定义>}

使用方式

- using namespace <命名空间名>;





命名空间解决的问题

```
//student1.h
#include<iostream>
using namespace std;
class Student
{
    char* name;
    int id_num;
    int age;
public:
    Student(int n,char* sname,int num)
    {age=n;name=sname;id_num=num;}
    get_data()
    {cout<<id_num<<" "<<name<<" "<<age<<endl;}
};
```





命名空间解决的问题

```
//student2.h
#include<iostream>
using namespace std;
class Student
{
    int id_num;
    char* name;
    char sex;
public:
    Student(int n,char* sname,char ssex)
    {id_num=n;name=sname;sex=ssex;}
    void get_data()
    {cout<<id_num<<" "<<name<<" "<<sex<<endl;}
};
```





命名空间解决的问题

```
//student.cpp
#include<iostream>
#include"student1.h"
#include"student2.h"
using namespace std;
int main()
{
    Student stu(101,"wang",18);/*有两个Student类定义，无法确定*/
    stu.get_data();
};
```

编译程序报错：
类Student重复定义





命名空间解决的问题

```
//student1.h
#include<iostream>
using namespace std;
namespace np1
{
    class Student
    {
        char* name;
        int id_num;
        int age;
    public:
        Student(int num,char* sname,int n)
        {id_num=num;name=sname;age=n;}
        void get_data()
        {cout<<id_num<<" "<<name<<" "<<age<<endl;}
    };
}
```





使用命名空间

```
//student2.h
#include<iostream>
using namespace std;
namespace np2
{
    class Student
    {
        int id_num;
        char* name;
        char sex;
    public:
        Student(int n,char* sname,char ssex)
        {id_num=n;name=sname;sex=ssex;}
        void get_data()
        {cout<<id_num<<" "<<name<<" "<<sex<<endl;}
    };
}
```





使用命名空间

```
//student.cpp
#include<iostream>
#include"student1.h"
#include"student2.h"
using namespace std;
int main()
{
    np1::Student stu1(101,"wang",18);
    stu1.get_data();
    np2::Student stu2(101,"wang",'f');
    stu2.get_data();
};
```





标识符和关键字

标识符是给程序的一些“元素”起的名字

关键字是系统预定义的，具有某些特殊用途，也称为保留字

```
#define _CRT_SECURE_NO_WARNINGS

#include<iostream>
#include<cstring>
#include<iomanip>
using namespace std;
enum Gender{M,F}; //枚举类型
class Student
{
    int id;
    char* name;
    Gender gender;
public:
    Student(int i, char* n, Gender g)
    {
        id = i;
        name = new char[strlen(n) + 1];
        strcpy(name, n);
        gender = g;
    }
    void print()
    {
        cout.setf(ios::left);
        cout << setw(4) << id << setw(8) << name;
        if (gender == M)
            cout << setw(4) << "男";
        else
            cout << setw(4) << "女";
    }
};

void main()
{
    Student stu[3] = { Student(1, "Zhang", M), Student(2, "Li", F), Student(3, "Wang", M) };
    cout.setf(ios::left);
    cout << setw(4) << "ID" << setw(8) << "Name" << setw(4) << "Gender" << endl;
    for (Student si : stu) {
        si.print();
        cout << endl;
    }
}
```



类和对象

类是定义数据类型的代码块

- 系统预定义类型
- 用户自定义类型

类的名称是数据类型的名称

类类型的数据项称为对象

```
#define _CRT_SECURE_NO_WARNINGS

#include<iostream>
#include<cstring>
#include<iomanip>
using namespace std;
enum Gender {M, F}; //枚举类型

class Student
{
    int id;
    char* name;
    Gender gender;
public:
    Student(int i, char* n, Gender g)
    {
        id = i;
        name = new char[strlen(n) + 1];
        strcpy(name, n);
        gender = g;
    }
    void print()
    {
        cout.setf(ios::left);
        cout << setw(4) << id << setw(8) << name;
        if (gender == M)
            cout << setw(4) << "男";
        else
            cout << setw(4) << "女";
    }
};

void main()
{
    Student stu[3] = { Student(1, "Zhang", M), Student(2, "Li", F), Student(3, "Wang", M) };
    cout.setf(ios::left);
    cout << setw(4) << "ID" << setw(8) << "Name" << setw(4) << "Gender" << endl;
    for (Student si : stu) {
        si.print();
        cout << endl;
    }
}
```



模板

几个类或者函数，代码中只是数据类型有区别

编译器可以使用给定的类型自动生成类或函数的代码实例

标准模板库（STL）

```
#define _CRT_SECURE_NO_WARNINGS

#include<iostream>
#include<cstring>
#include<iomanip>
#include<vector>
using namespace std;
enum Gender{M,F};
class Student
{
    int id;
    char* name;
    Gender gender;
public:
    Student(int i, char* n, Gender g)
    {
        id = i;
        name = new char[strlen(n) + 1];
        strcpy(name, n);
        gender = g;
    }
    void print()
    {
        cout.setf(ios::left);
        cout << setw(4) << id << setw(8) << name;
        if (gender == M)
            cout << setw(4) << "男";
        else
            cout << setw(4) << "女";
    }
};

void main()
{
    Student stu[3] = { Student(1, "Zhang", M), Student(2, "Li", F), Student(3, "Wang", M) };
    vector<Student> stu_vec;
    cout.setf(ios::left);
    cout << setw(4) << "ID" << setw(8) << "Name" << setw(4) << "Gender" << endl;
    for (Student si : stu) {
        si.print();
        cout << endl;
    }
}
```

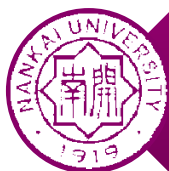
计算机与程序设计语言 ☐
数制转换与数据存储 ☐
C++语言的基本概念 ☐
C++语言的词汇 ☒

☐ 关键字
☐ 标识符
☐ 字面值常量
☐ 运算符

☐ 分隔符



计算机与程序设计语言



数制转换与数据存储



C++语言的基本概念



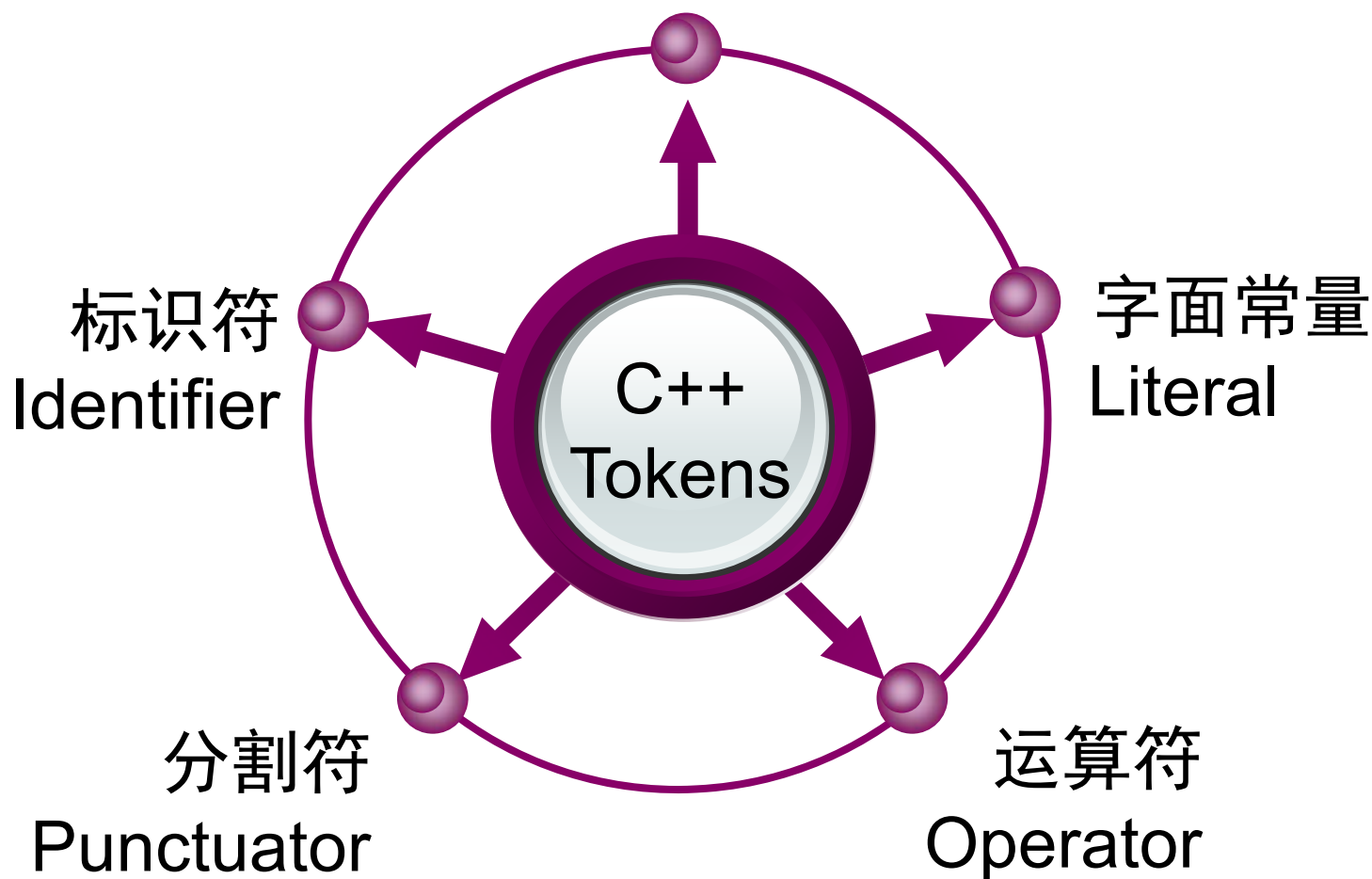
C++语言的词汇



C++程序的结构



关键字Keyword





关键字 (keyword)

一类特定的具有专门含义的单词
又称为保留字 (reserved word)
例如 (见P34表2.3)

- void
- int
- if
- else
- for
- while
-





标识符 (identifier)

为程序中使用的数据（常量或变量）、函数、类、对象、文件等起的“名字”

组成规则

- 以字母或下划线“_”开头，由字母、数字、下划线组成的字符串
- 不能与关键字重名
- 标识符区分大小写
- 有效长度有规定





字面值常量 (literal)

整型常量 (int) :

- 整数

浮点型常量 (double) :

- 小数

字符常量 (char) :

- 一个字符, 由单引号标识

字符串常量:

- 0个、1个或多个字符, 以字符'\0'结尾, 由双引号标识

布尔型常量

指针常量

用户自定义常量





整型常量

int 型常量即整型常量，实际上就是整数。C++程序可以处理一般的十进制整数，以及：

- 二进制整数（加前缀0b或0B）
- 八进制整数（加前缀0）
- 十六进制整数（加前缀0x或0X）

注意：在C++程序中，各种进制的整数都自动转换为十进制输出

- 例如：`cout<<023<<" "<<23<<" "<<0x23<<0b10;`
将输出不同的十进整数：`19_23_35_2`





浮点型常量

小数点表示法：4.75，2.0

科学记数表示法：1.2e4，-7.37e-3

注意，在C++程序中，浮点数以十进制的形式输入和输出，浮点数的存储格式与类型有关

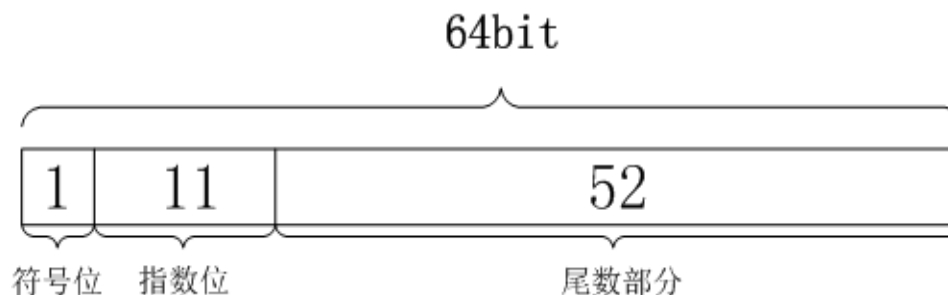
- 例如：C++中float型浮点数占4个字节，double型浮点数占8个字节，long double型浮点数占10个字节。
- 浮点型常量按double型处理，占8个字节
 - 加后缀f或F可按float型处理，占4个字节
 - 加后缀l或L可按long double型处理，占10个字节
- 浮点型常量有表示范围，见P63表3.2



浮点数的存储

以double型浮点数为例

- 占8个字节（64bit）
 - 符号位
 - 0代表正，1代表为负
 - 指数位
 - 用于存储科学计数法中的指数数据，并且采用移位存储，即以 $2^{10}-1$ 为基准，加减指数
 - 尾数





字符型常量

用单引号括起的基本符号

- 'A', 'g', '3', ' ' ,
- 基本字符集（96个）
 - 大小写字母52个
 - 数字10个
 - 空白字符1个（空格）
 - 控制字符4个（水平和垂直制表符、换页符、换行符）
 - 其它可见字符29个
- 转义序列（P38表2.4）
 - 基本字符的一符多义
 - 以反斜杠\开头
 - 一般用于输出字符串中的字符





字符型常量

ASCII字符

- 占1个字节
- 表示范围：-128~127

ASCII码将字符型常量与整型常量建立对应关系

- 7位二进制编码，有128不同的编码值，每个值对应一个ASCII字符
 - 0~127
- 扩展：8位二进制编码
 - -128~127
 - 0~255





字符型常量

Unicode字符

- Unicode是标准，定义了一组字符及其代码点，可以表示更大范围的字符
 - UTF-8、UTF16和UTF-32
- UTF-8将字符表示为1个字节或者4个字节的变化序列
- UTF-16将字符表示1个或2个16位值
- UTF-32将字符表示为32位值
- 字符常量加前缀L、u8、u或U，分别对应宽字符wchar_t、UTF-8、UTF16和UTF-32





字符型常量

- 字符型常量（char）：一个字符，由单引号标识
 - ✓ char型：'a'
 - ✓ 宽字符型，加前缀L：L'a'
 - ✓ UTF-8字符型，加前缀u8：u8'a'
 - ✓ UTF-16字符型，加前缀u：u'a'
 - ✓ UTF-32字符型，加前缀U：U'a'





字符串常量

用双引号括起来的字符序列

- 以字符‘\0’结尾，包含0个、1个或多个字符的序列
- 字符串长度为字符数+1，如“string constant”长度为16

前缀：

- R, u8, u8R, u, uR, U, UR, L, LR
 - U8: UTF-8字符串
 - u: UTF-16字符串, char16_t类型
 - U: UTF-32字符串, char32_t类型
 - L: 宽字符串, wchar_t类型
 - R: 表示该字符串常量为原始字符串字面值常量



字符串常量

原始字符串字面值常量 (raw string literal)

- 不包含转义字符，全部按照原始字符理解字符串
- 格式：R"(.....)"
- 例如：

```
cout<< R"(C++ \nProgramming)";
```

输出：C++ \nProgramming

```
cout<<"C++ \nProgramming";
```

输出：C++
Programming



布尔型常量

只有两个值

- true
 - 对应非0整数
- false
 - 对应整数0

指针字面值常量

nullptr

- 空指针，用于初始化“悬挂”状态的指针变量





用户定义字面值常量

用户定义字面值常量

- 由operator定义后缀

<返回值类型> **operator**""<后缀名> (参数表)

- 将后缀的功能用函数描述，该后缀为用户自定义的字面值常量

- 例如：

```
double operator ""km(long double n)
{
    return n*1000;
}
cout<<100.0km<<endl;
```

支持用户自定义字面值常量的参数类型包括：

unsigned long long
char
wchar_t
char16_t
char32_t
long double
以及相应的复合类型



运算符 (operator)

由字母、数字之外的第三类基本符号组成
个别关键字如sizeof、new、delete，也被认为是运算符

其余运算符为：

$+, -, *, /, \%, =, !=, <, <=, >, >$
 $=, !, \&\&, ||, \&, ^, |, \sim, ++, --, +=, -=, *=, /=, \%$
 $=, <<=, >>=, \&=, ^=, |=, ?:, =, (), [], ., ->, <<, >>, ,, ::$

- 某些运算符还有其它用途，如乘运算符*





分隔符 (Punctuator)

没有明确的含义，但在程序中必不可少
用来界定或分隔程序中的语法成分，类似于
“标点符号”

常用分隔符有：

`_ , " # () /* */ // ' ; { }`





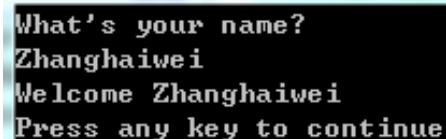
【例1.1】最简单的C++程序

```
#include<iostream>
using namespace std;
int main()
{
    cout<<"This is a C++ Program."<<endl;
    return 0;
}
```



【例1.2】简单的输入输出交互

```
#include<iostream>
using namespace std;
void main()
{
    char name[20];
    cout<<"What's your name?"<<endl;
    cin>>name;
    cout<<"Welcome " <<name<<endl;
}
```

A screenshot of a terminal window showing the execution of the C++ program. The output is as follows:

```
What's your name?
Zhanghaiwei
Welcome Zhanghaiwei
Press any key to continue
```

【例1 3】输出本学期的课程表

```
#include <iostream>
using namespace std;
void
```

```

                                课程表
=====
时间      星期一      星期二      星期三      星期四      星期五
-----
8:00      高等数学          高等数学      线性代数
-----
10:00     政治      线性代数          习题课      英语
-----
14:00          体育      英语          习题课
-----
16:00          实验课
-----
18:30     实验课
=====
Press any key to continue
```

```
"<<setw(15)<<"<<setw(15)<<"<<setw(15)<<"<<setw(15)<<"<<endl;
```

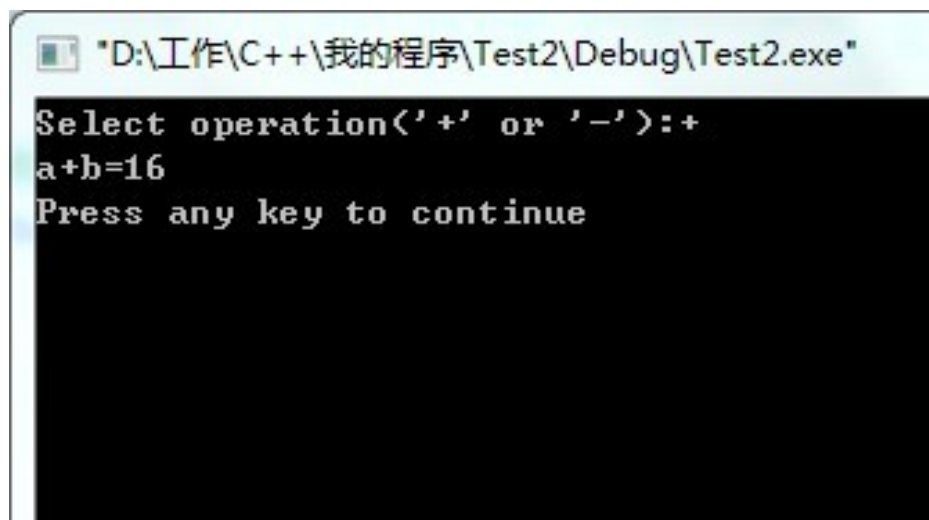
```
cout<<"===== "<<endl;
```

```
}
```



【例1.4】简单的计算程序

```
#include<iostream>
using namespace std;
void main()
{
    char op;
    int a=10,b=6;
    cout<<"Select operation("
    cin>>op;
    switch(op){
    case '+':
        cout<<"a"<<op<<"b="<<a+b<<endl;
        break;
    case '-':
        cout<<"a"<<op<<"b="<<a-b<<endl;
        break;
    default:
        cout<<"No such operation!"<<endl;
    }
}
```



```
"D:\工作\C++\我的程序\Test2\Debug\Test2.exe"
Select operation(<'+' or <'-'>):+
a+b=16
Press any key to continue
```

计算机与程序设计语言 ☐
数制转换与数据存储 ☐
C++语言的基本概念 ☒
C++语言的词汇 ☐

☐ 过程型结构
☐ 函数型结构
☒ 面向对象型结构



计算机与程序设计语言



数制转换与数据存储



C++语言的基本概念



C++语言的词汇



C++程序的结构



C++程序的结构

过程型结构

- 根据解决问题的步骤组织程序

函数型结构

- 将待解决的问题分解为若干子问题，用函数进行组织

面向对象型结构

- 使用类和对象

过程型结构

【例1.5】过程型程序

```
#include<iostream>
using namespace std;
void main()
{
    int a,b;
    cout<<"a=";
    cin>>a;
    cout<<"b=";
    cin>>b;
    int c = a+b;
    cout<<"a+b="<<c<<endl;
}
```



函数型结构

【例1.6】函数型程序

```
#include<iostream>
using namespace std;
void input(int& x, int& y)
{
    int a,b;
    cout<<"a=";
    cin>>a;
    cout<<"b=";
    cin>>b;
    x=a;
    y=b;
}
int add(int x, int y)
{
    return x+y;
}
```

```
void output(int z)
{
    cout<<"a+b="<<z<<endl;
}
void main()
{
    int a,b;
    input(a,b);
    int c = add(a,b);
    output(c);
}
```





面向对象型结构

【例1.7】面向对象型程序

```
#include<iostream>
using namespace std;
class Calculator
{
    int a;
    int b;
public:
    Calculator();
    int Add(int,int);
    int Sub(int,int);
};
Calculator::Calculator()
{
    a=0;
    b=0;
}
```

```
int Calculator::Add(int x,int y){
    return x+y;
}
int Calculator::Sub(int x,int y){
    return x-y;
}
void main(){
    Calculator cal;
    int m,n;
    cout<<"m=";
    cin>>m;
    cout<<"n=";
    cin>>n;
    cout<<"m+n="<<cal.Add(m,n)<<endl;
    cout<<"m-n="<<cal.Sub(m,n)<<endl;
}
```



```
int poker=4;  
char poker='\♥';
```

如何将不同类型数据组合在一起，描述现实世界中的复杂“数据”

类

- 过程型结构
- 函数型结构
- 面向对象型结构



大学学生注册卡片

学号: _____

姓名(用汉语拼音写)

院(系):

专业:

校徽号:

学号 为 单数贴 照片处	姓名(汉字):		性别:	出生地:	省	县	学号 为 双数贴 照片处		
	政治面目:		民族:	出生年月:	年	月		日(周岁)	
	入学	①	年	中学毕业,校址:					
	程度	②							
来源		参加	年	考区录取第__志愿或经过			保送		
高考成绩		语文	数学	外语(语种)	史化	物地	政治	生物	总分
家庭通讯地址		省 县 邮编:							
家庭 成员 及 主要 社会 关系	姓	名	与本人关系	政治面目	工作单位及职务			电 话	
身份证号					备 注				

填写时必须认真写清姓名和出生年月日,且须与户口本或身份证相一致。

年 月 日





ResultofGroupA.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

France 2:1 Romania
Switzerland 1:0 Albania
France 2:0 Albania
Switzerland 1:1 Romania

C:\WINDOWS\system32\cmd.exe

```
=====
Pos  Team          Pld  W  D  L  GF GA GD  Pts
-----
1    France         2    2  0  0  4  1  3    6
2    Switzerland    2    1  1  0  2  1  1    4
3    Romania        2    0  1  1  2  3  -1    1
4    Albania        2    0  0  2  0  3  -3    0
=====
```

请按任意键继续. . .

搜狗拼音输入法 全 :



第一章 结束

