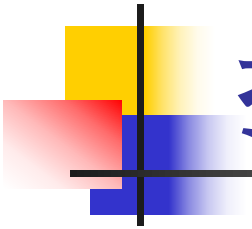
第一讲 程序设计基础

周水庚

计算机学院

2024年9月5日



提要

- 计算机简介
- 程序设计基本概念
- 结构化程序设计
- C语言基础知识
- 高级语言程序开发环境
- 小结

提要

- 计算机简介
- 程序设计基本概念
- 结构化程序设计
- C语言基础知识
- 高级语言程序开发环境
- 小结

计算机的基础与发展规律

- 基础
  - 逻辑代数
  - 图灵机：现代通用数字计算机的数学模型，为存储程序式电子计算机提供了重要的设计思想
  - 冯.诺依曼计算机构架：二进制存储程序并按地址顺序执行
- 规律
  - Moore’s Law (1965)：微处理器内晶体管数每18个月翻一番
  - Bell’s Law (1972)：如果保持计算能力不变，微处理器的价格每18个月减少一半
  - Metcalfe’s Law (1980)：网络价值同网络用户数的平方成正比
  - Gilder’s Law (1996)：主干网的带宽将每6个月增加一倍
  - 半导体存储器发展规律：DRAM密度每年增加60%，每三年翻四倍
  - 硬盘存储技术发展规律：硬盘的密度每年增加约一倍

John von Neumann

- 约翰·冯·诺依曼 （John von Neumann，1903－1957），美籍匈牙利人，20世纪最杰出的数学家之一。在数学、计算机和物理学领域都有开创性贡献
  - 逻辑和集合论
  - 量子力学
  - 计算机
  - 博弈论与经济学
  - 核物理

(1903-1957)

John von Neumann

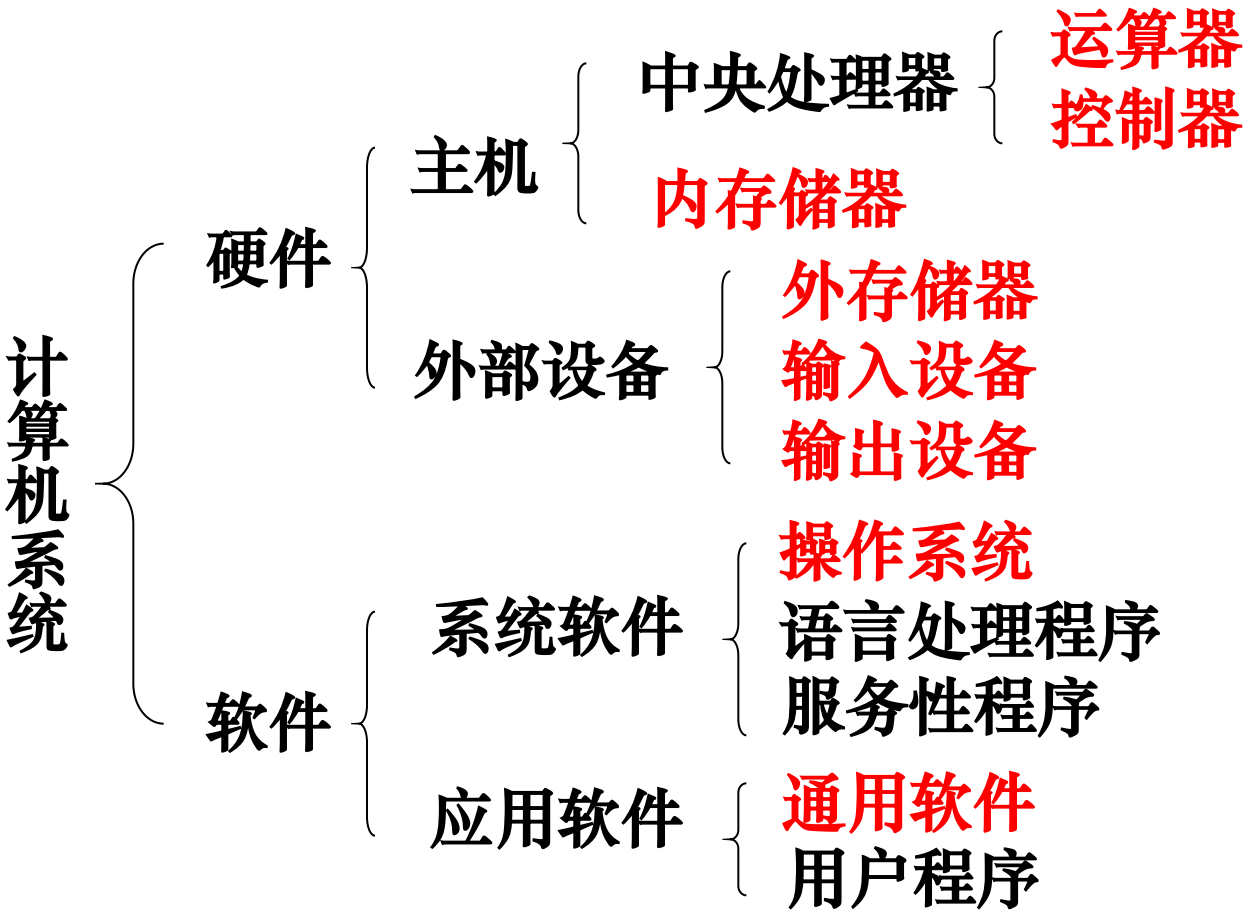
Gravestone of John von Neumann
at Princeton Cemetery in Princeton

EDVAC (*Electronic Discrete Variable Automatic Computer*)
ENIAC (Electronic Numerical Integrator And Computer)

冯•诺依曼体系构架

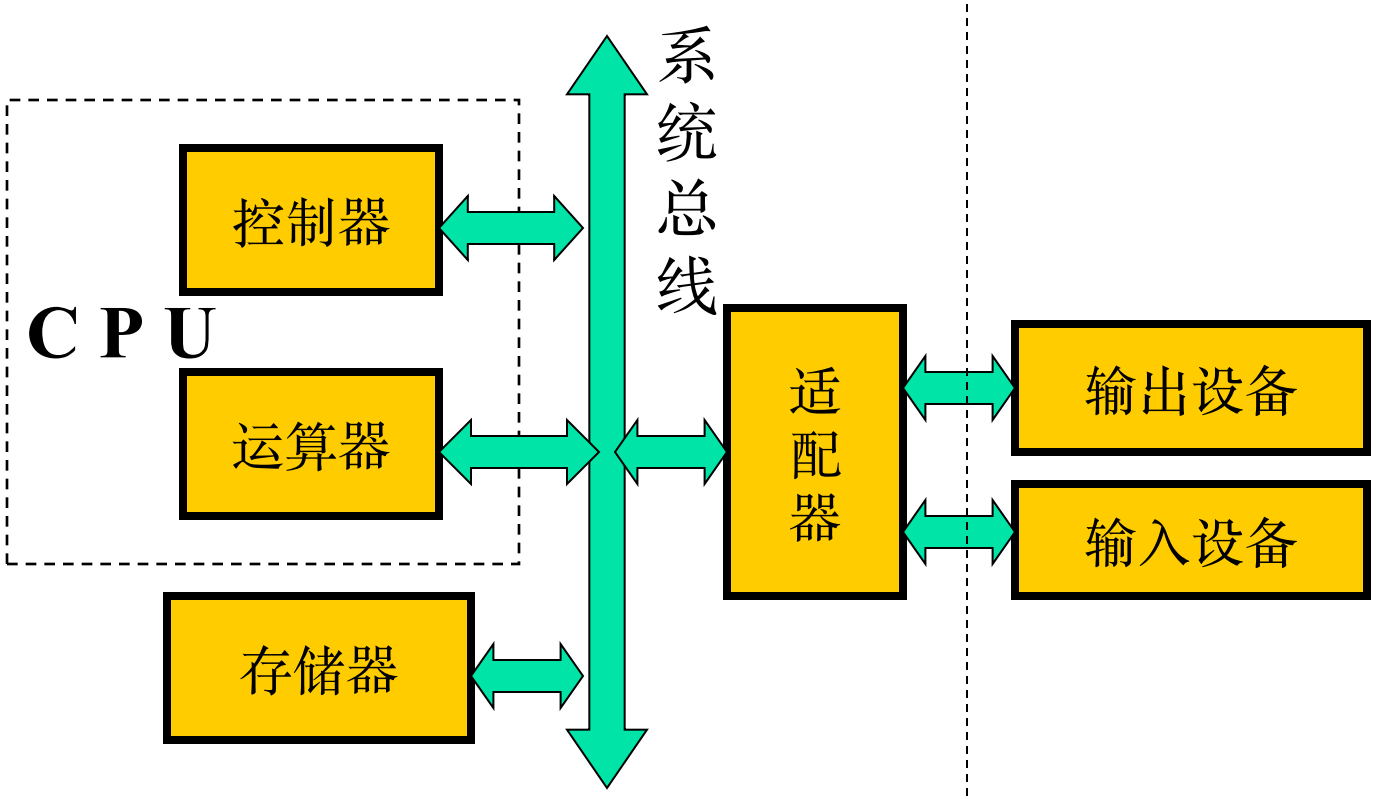
- 由冯·诺依曼等人在1946年6月在一篇报告“*First Draft of a Report on the EDVAC*”中首先明确提出来的，其实这是基于J. Presper Eckert和John Willian Mauchy在UPen开发的ENIAC计算机体系结构。这种计算机体系结构的特点可以简要地概括为以下几点：
  - 计算机（指硬件）应由运算器、控制器、存储器、输入设备和输出设备5大基本部件组成
  - 计算机内部采用二进制来表示指令和数据
  - 将编好的程序的原始数据先存入存储器中，然后再启动计算机工作

计算机系统组成


计算机系统
硬件
主机
中央处理器
运算器
控制器
内存储器
外部设备
外存储器
输入设备
输出设备
软件
系统软件
操作系统
语言处理程序
服务性程序
应用软件
通用软件
用户程序

图形处理器
GPU：Graphics Processing Unit

计算机系统的硬件组成



CPU: Central Processing Unit

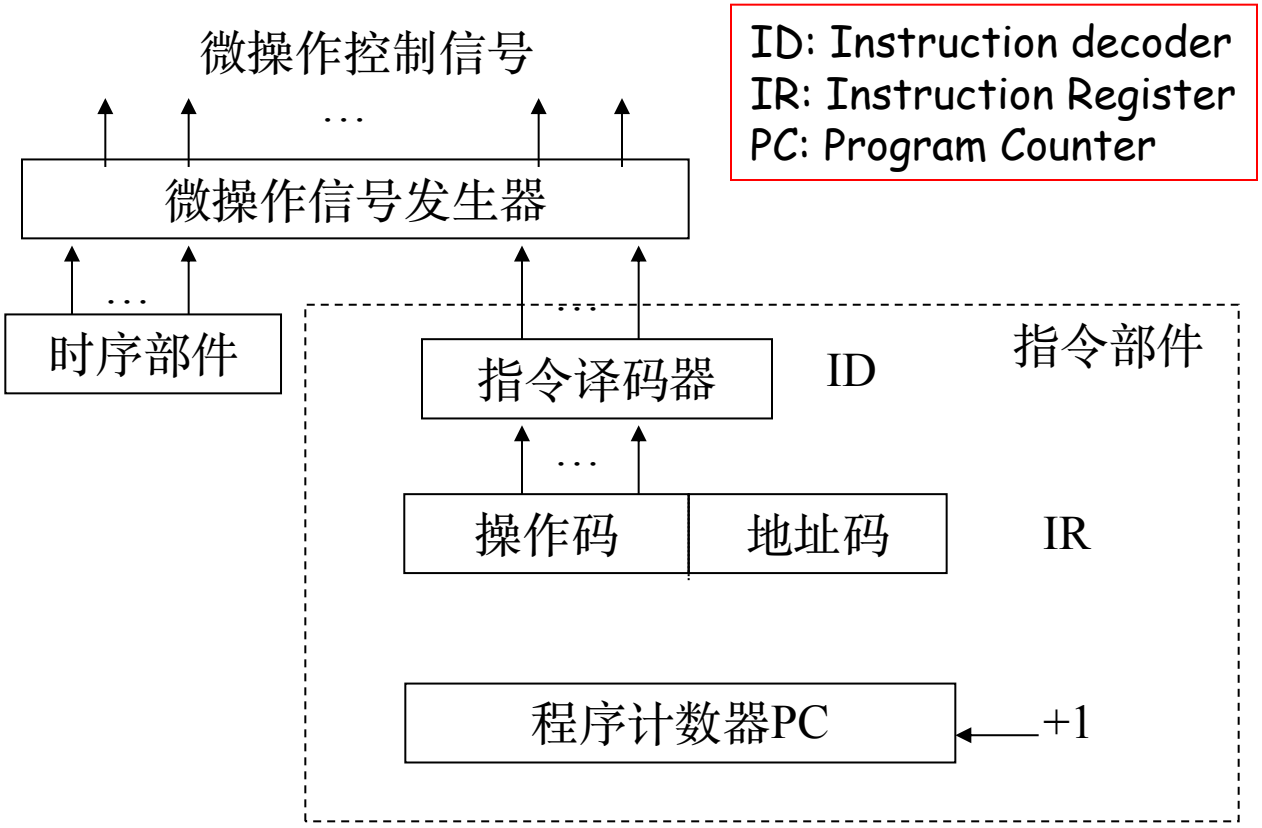
运算器（ALU）

- 功能：实现算术逻辑运算功能
- 构成：全加器(FD: Full-adder)，通用寄存器组(GRS, General Registers)，输入接收门和移位输出门

运算器（算术逻辑单元）:ALU (Arithmetic Logic Unit)
数据总线：DB (Data Bus)
控制总线：CB (Control Bus)
地址总线：AB (Address Bus)
程序计数器：PC (Program counter)

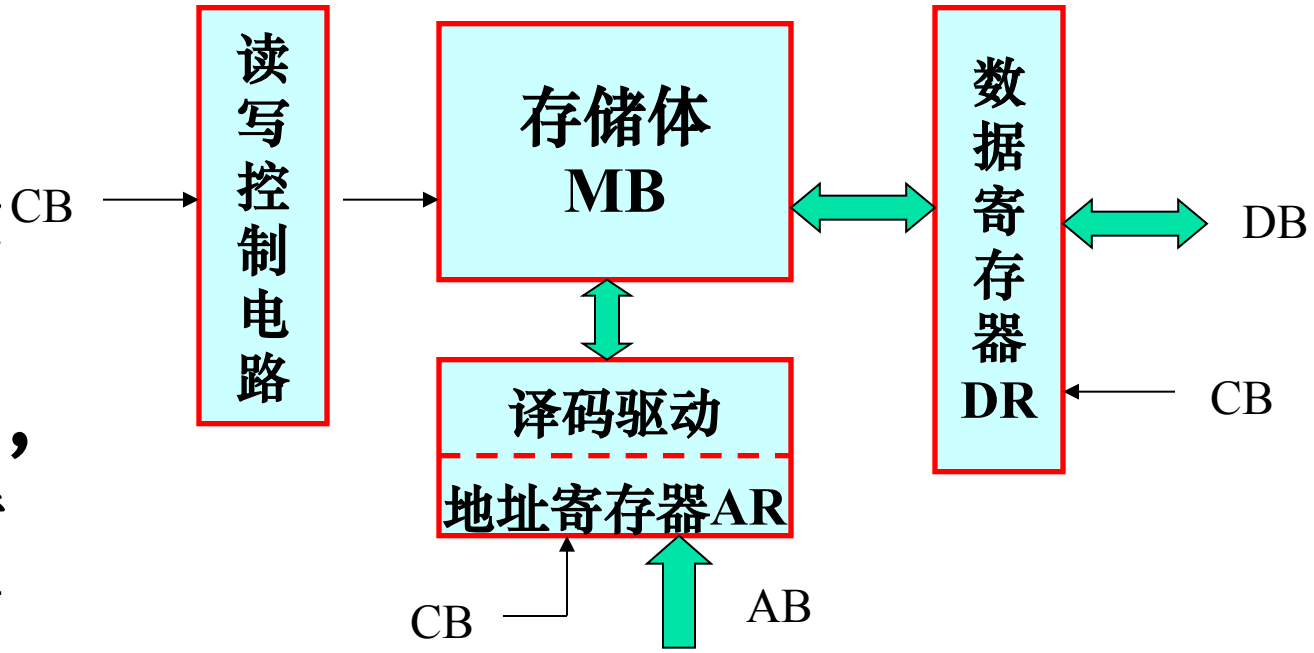
控制器(Controller)

- **功能：根据指令的要求向计算机各部件发出控制信号，使之协调有序工作**



内存储器（Memory）

- 存放信息（程序和数据）
- 内存储器是用于存放计算机运行期间所需要的大量程序和数据的部件，是CPU能直接访问的存储器



存储器的有关术语

- **存储单元**：存储一个字节(8位二进制数)的连续内存空间
- **单元地址**：存储单元的地址编号。地址编号以二（八或十六）进制数表示，从0开始
- **存储容量**：内存储器中存储单元的总数。通常以KB、MB、GB、TB表示

其中，$1TB = 2^{10}\ GB = 2^{40}B$

$1GB = 2^{10}MB = 2^{30}B$

$1MB = 2^{10}KB = 2^{20}B$

$1KB = 2^{10}B$　1B=8bits

Bit: b 计算机中最小的信息单位
Byte: B 计算机中最小存储单位
Kilobyte: KB
Megabyte: MB
Gigabyte: GB
Terabyte: TB

计算机为什么采用二进制?

- 符号个数最少，“0/1” 物理上容易实现
- 用数字电路的两个状态表示(如电压高低)
- 与二值逻辑的真/假两个值对应简单
- 二进制位可以表示任何对象(字符、数值和逻辑值)
- 用二进制码表示数值数据运算规则简单
  - 0+1=1+0=1；1+1=0；0+0=0
  - 仅仅三种运算规则（10进制有55种）
  - 一个异或门即可完成该运算

外围设备（I/O）

- **输入设备**
  - **将各种形式的输入信息，转换为计算机可接收的形式，如：键盘、鼠标等**
- **输出设备**
  - **将计算机的输出信息转换为其它对象可接收的形式：显示器、打印机等**
- **终端设备**
  - **通过通信线路与主机连接的既可输入又可输出的设备**

计算机软件

- 系统软件
  - 管理计算机资源与系统的软件，如操作系统
  - 开发环境软件，如语言程序
  - 共性软件，如数据库管理系统
  - 服务性软件等
- 应用软件
  - 计算机用户为解决各种问题而编制的程序

计算机系统的层次结构

- 计算机系统存在着层次结构，从功能上看，现代计算机系统可分为7个层次级别，每一层都能进行程序设计

计算机的工作过程

- 周而复始地取出指令、解释指令和执行指令的过程
- 具体步骤
  - 取指令：把程序计数器PC中的指令地址送存储器的地址寄存器AR，按该地址取出指令送指令寄存器IR
  - 解析指令：
    - 取操作数：根据IR中的地址码，由地址计算部件形成操作数地址送存储器，取出数据，送到运算器中的寄存器（寄存器组）
    - 取操作码：将IR中的操作码OP送指令译码器进行译码
  - 执行指令：计算机有关部件在控制器发出的操作控制信号的控制下，执行操作码OP规定的操作
  - 程序计数器PC加1，形成下一条指令地址，再取指令。如此重复进行，直到所有指令执行完毕

字长和主频

- 字长
  - 字长是指CPU一次能并行处理的二进制位数。字长是8的倍数
    - 字长标志着机器表示数的精度，位数越多，表示数的精度就越高
    - 在其它指标相同时，字长越大的计算机处理速度越快
- 主频率（主频）
  - 计算机的主时钟不断地产生固定频率的时钟脉冲，时钟脉冲的频率就是CPU的主频率。CPU工作的节拍是由主频率控制，主频率越高，CPU工作节拍越快。是影响计算机运行速度的重要参数

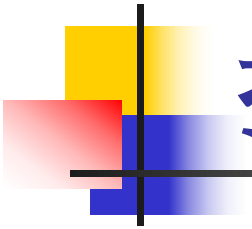
存储容量

- 主存储器的容量
  - 主存是CPU可直接访问的存储器
  - 主存的容量是指主存储器存储单元的总数，例：具有16位地址码的计算机，主存的容量位2^{16}字节，即64K
- 外存储器的容量
  - 外存容量是指计算机系统中联机的外存储器的容量，外存存放暂不参与运行的程序和数据

运算速度

- 计算机的运算速度与许多因素有关，对运算速度的衡量有不同的方法
  - 根据不同类型指令在计算过程中出现的频繁程度，乘上不同的系数，求得统计平均值，这时所指的运算速度是平均运算速度。
  - 以每条指令执行所需时钟周期数CPI来衡量运算速度。
  - 以MIPS和MFLOPS作为计量单位来衡量运算速度

CPI: Clocks Per Instruction
MIPS: Million Instructions Per Second
MFLOPS: Million Floating-point Operations per Second



提要

- 计算机简介
- **程序设计基本概念**
- 结构化程序设计
- C语言基础知识
- 高级语言程序开发环境
- 小结

什么是程序?

- 程序（Program）就是供计算机执行后，能完成特定功能的指令序列（Instructions sequence）
  - 程序=计算机指令序列
- 程序包含两方面的内容
  - 数据：参与运算的数据对象（Objects）
    - 数据结构(Data structure)：数据如何表示、组织?
  - 指令：对这些对象的处理
    - 算法(Algorithm)：指令如何执行?

程序／数据结构／算法

- 程序 = 数据结构 + 算法
- Program = Data Structure + Algorithm

这个公式由Niklaus Wirth首先提出。
Niklaus Wirth 是PASCAL之父和结构化程序设计的首创者，1984图灵奖获得者。

数据结构

- 数据结构**由数据对象及对象中所有数据成员之间的关系组成。记为：**
  Data_Structure = {D, R}
  - **其中，D (Data)是数据对象，R (Relationship)是该对象中所有数据成员之间关系的有限集合**

数据结构（续）

- 程序的处理对象是描述客观事物的数据
- 由于客观事物的多样性，会有不同形式的数据
  - 整数、实数、字符，以及所有计算机能够接收和处理的各种各样符号集合
- 在程序中，形式不同的数据采用数据类型（Data Type）来标识
- 变量的数据类型说明变量可能取值的集合、施于变量的操作集合

数据结构（续）

- 数据类型
  - 一种数据类型定义一组形式和语义相同的数据集，隐含着对这组数据可施行的一组操作集
- 对数据结构的总括
  - 数据结构是指数据成分及其相互关系与构造方法
  - 程序的数据结构描述了程序中的数据的组织形式和结构关系

算法

- 算法即**问题的求解方法**
- 算法由一系列求解步骤组成。算法的描述由明确说明的一组**简单指令**和**规则组成**，计算机按规则执行其中的指令能在有限的步骤内解决一个问题或者完成一个函数的计算
- 组成算法的规则和步骤的意义应是唯一确定的，是没有二义性的

算法（续）

- 算法中的操作是有序的，必须按算法指定的操作顺序执行，能在有限步骤后给出问题的结果
- 求解同一问题可能有多种算法，选择算法主要考虑
  - 正确性
  - 可靠性（鲁棒性）
  - 简单性
  - 易理解性
  - 执行效率
  - 空间（内存和磁盘）代价等

算法（续）

- 描述算法的常用工具有流程图（又称框图）
- 流程图(Flow diagram)是算法的图形描述，流程图往往比程序更直观清晰，容易阅读和理解，它不仅可以作为编写程序的依据，也是交流算法思想的重要工具
- 在逐步求精的结构化程序设计方法中，目前多数采用结构化的伪代码 (pseudo code)来描述算法

数据结构 vs. 算法

- 明确了求解问题的算法，才能较好设计数据结构
- 要选择好算法，又常常依赖于合理的数据结构
- 程序是和数据结构不可分割的。程序在描述算法同时，也必须完整地描述作为算法的操作对象的数据结构
- 对于一些复杂的问题，常有因数据的表示方式和结构的差异，问题的求解算法也会完全不同

程序的性质

- 目的性
  - **程序有明确的目的，程序运行时能完成赋予它的功能**
- 分步性
  - **程序为完成复杂的功能，由一系列计算机可执行的步骤组成**
- 有序性
  - **程序的执行步骤是有序的**
- 有限性
  - **程序是有限的指令序列，程序所包含的步骤是有限的**
- 操作性
  - **有意义的程序总是对某些对象进行操作，使其改变状态**

什么是程序设计?

- 程序设计（Program design）：设计和编制程序的过程
  - 设计程序：数据结构设计和算法设计
  - 编制程序：把设计转化为某种程序语言的代码
- 要进行科学、高效地程序设计，我们需要
  - 程序设计方法 – 软件工程（Software Engineering）方法学
    - 结构化设计方法、面向对象方法
  - 程序设计语言 – 程序语言学（Programming Language）
    - Basic, Pascal, C, C++, Java, C#, Python……

什么是好的程序?

- 高效（Efficient）
- 可靠（Reliable）
- 易读（Easy to read）
- 可维护（Maintainable）
- 可重用（Re-usable）
- 可移植（Portable）
- ……

什么是程序设计语言?

- 程序设计语言是人与计算机对话的工具，是用来书写计算机程序的语言
  - 人机语言（human-machine language）
- 程序设计语言分三类
  - 机器语言（Machine language）
  - 汇编语言（Assembly language）
  - 高级语言（High-level language）

程序设计语言

汇编语言
Assembly Language

自然语言
Natural Language

高级语言
High-level Language

机器语言
Machine language

机器语言

- 机器语言：计算机的指令系统
- 计算机都只能直接执行由其自身机器语言编写的程序
- 机器语言与计算机的硬件密切相关
  - 机器语言中的计算机指令通常用一个二进制形式的代码，由若干位1和0组成
  - 一条计算机指令指示计算机一次完成一个最基本的操作

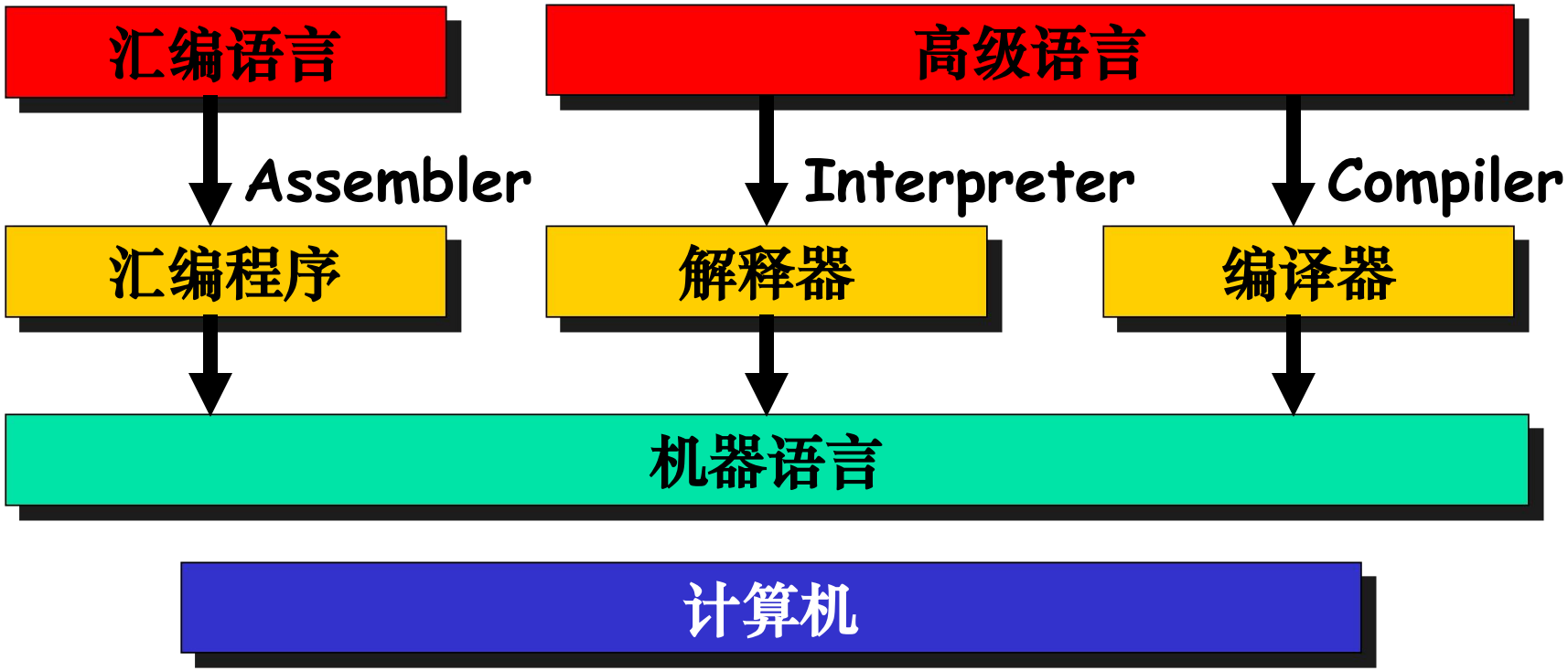
汇编语言

- 汇编语言：用类英语单词缩写的符号指令代替机器语言的二进制代码指令
- 用汇编语言编写的程序在计算机上执行，先要将用汇编语言编写的源程序转换成机器语言程序。完成这个转换功能的程序称为“汇编程序“(assembler)

高级程序

- 最接近自然语言的程序设计语言
  - 主要由语句(statements)构成，语句是要计算机完成任务的命令
  - 有统一的语法，独立于具体机器，便于人们编码，阅读和理解
  - 用高级语言编写的程序（源程序）在计算机上执行，先要由编译器（也称为编译程序）将源程序转换成机器语言程序
  - 既能方便地描述客观对象，又能借助于编译器转换为计算机所接受的语言

机器语言/汇编语言/高级语言



高级语言分类

- 目前的高级语言可分三类
  - 面向过程的语言（Procedural language）
  - 面向问题的语言（Declarative language）
    - 描述语言
  - 面向对象的语言（Object-oriented language）

面向过程的语言

- 早期最流行最常用的程序设计语言为面向过程型的语言
  - Cobol, Fortran, Basic, C, ……
- 可独立于计算机编写程序，但编写程序时，程序不仅要说明做什么(what)，更重要的是非常详细地告诉计算机如何做(how)，程序需要详细描述解题的过程和细节

面向问题的语言

- 不必关心问题的求解算法和求解的过程，只需指出问题是做什么，数据的输入和输出形式，就能得到所需结果
  - 数据库查询和操纵语言：SQL(Structured Query Language)语言
  - 逻辑语言／专家系统语言：Prolog, Lisp等

面向对象的语言

- 目的是克服面向过程语言过分强调求解过程细节，程序不易复用的缺点
- 引入了对象、消息、类、继承、封装、抽象、多态性等机制和概念
- 用面向对象语言进行程序设计时，以问题中的对象为基础，将具有类似性质的对象抽象成类，利用继承机制，对差异进行程序设计
- 是目前的主流编程语言：C++, Java，python等

程序设计语言分类

- 与自然语言亲密关系
  - 机器语言、汇编语言、高级语言
- 描述/求解问题方式
  - 面向过程语言： Basic, Pascal, C
  - 面向问题语言： SQL, Lisp, Prolog
  - 面向对象语言: C++, Java, python
- 程序设计风格
  - 结构化语言
    - Pascal, C, C++, Java, python
  - 非结构化语言
    - Cobol, Fortran, Basic

提要

- 计算机简介
- 程序设计基本概念
- 结构化程序设计
- C语言基础知识
- 高级语言程序开发环境
- 小结

结构化程序设计方法

- 程序结构自顶向下（top-down）模块化设计方法
- 模块算法的逐步求精(stepwise refinement)设计方法
- 用结构化的控制结构描述算法和编写程序

自顶向下模块化设计方法

- 核心思想：限制程序设计的复杂性
  - 把大程序按功能划分成一些较小的部分（模块），每个模块完成明确、独立的功能
  - 分解模块的原则：简单性、独立性和完整性
- 模块化设计方法使程序具有
  - 较高的可靠性和灵活性
  - 便于程序的测试和维护

自顶向下模块化设计方法（续）

- 用模块化方法划分程序时，应尽量让模块具有如下性质
  - 具有单一入口和单一出口
  - 模块不宜过大，模块功能单一
  - 模块的执行不对环境产生副作用
  - 让模块与环境的联系仅限于输入和输出参数，模块的内部结构与调用它的程序无关
  - 用模块的名字调用模块

模块算法逐步求精设计方法

- 基本方法
  - 抽象、枚举、归纳
- 抽象包括算法抽象和数据抽象
  - 算法抽象是指算法的开发采用逐步精化、逐层分解的方法
  - 数据抽象指在算法抽象的过程中逐步完善数据结构和引入新的数据及确定关于数据的操作

逐步求精设计方法（续）

- 算法设计
  - 先设计出一个抽象算法，这是一个在抽象数据上实施一系列抽象操作的算法，由粗略的控制结构和抽象的计算步骤组成。
  - 抽象操作只指明“做什么”，对这些抽象操作的细化就是想方设法回答它“如何做”
  - 这样，由粗到细，将抽象步骤进一步分解成若干子任务。分而治之，对仍不具体的抽象子任务再进行分解。如此反复地一步步细化，算法越来越具体，抽象成分越来越少，直至可以编程为止

逐步求精设计方法（续）

- **枚举**是指在设计算法时，先找出所有可能的情况
- **归纳**是指将所有可能情况和各种可能的处理方法作归纳，找出其中相似或者相近的部分，把相似或者相近的部分归纳为基本操作，以减少程序代码，提高程序的可复用性

程序的基本结构单元

- 顺序结构
- 条件选择结构
  - If c Then op1 Else op2
- 循环结构
  - Do While 循环结构
  - While循环结构
  - For 循环结构
- **任何可计算问题，可用上述三种结构编程解决**

顺序结构

- 把复杂的计算工作分解成一系列逐条执行的操作序列
- 顺序结构为把一个复杂的计算用若干简单计算的顺序执行提供控制手段
- 顺序结构执行时，从序列的第一个操作开始，顺序执行序列中的操作，直至序列的最后一个操作执行完

顺序结构实例

- 实例：交换变量x和y的值
- 可分解为顺序执行的三个操作步骤：

```
{ temp=x;/*将x的值暂存于temp */
  x   = y; /* 将x置成y的值 */
  y=temp;/*将y置成temp的值 */
}
```

条件选择结构

- 条件选择结构由一个**判断条件**和**两个供选择分支操作**组成
- 一般形式：

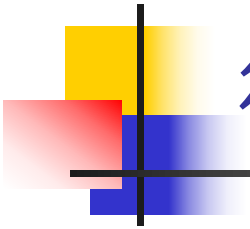
```
if (判断条件)
    分支操作1;
else
    分支操作2;
```

条件选择结构执行过程

- 先计算判断条件，如判断条件的值为真，即条件成立，则执行分支操作1
- 否则，即条件不成立（判断条件的值为假），则执行分支操作2
  - 注意，无论判断条件为何值，条件选择结构只执行分支操作1或分支操作2
- 条件选择结构中的分支可以是任何控制结构。当分支操作是条件选择结构时，就呈现嵌套（nested）的条件选择结构

实例

```
int max(int x，int y) /* 定义max函数，函数值为整型， 形式参数x，y为整型*/
{int z; /* max函数中的声明部分，定义本函数中用到的变量z为整型*/
 if (x>y) z=x;
      else z=y;
return (z);
}
```



循环结构

- 为描述循环操作提供的控制手段
- 在C中，循环结构有以下三种
  - while 循环结构
  - do-while 循环结构
  - for 循环结构

while 循环结构

- 由一个循环条件和一个（组）循环操作语句（称为循环体）组成
- 一般形式：

```
while ( 循环条件 )
        循环体
```

while 循环结构执行过程

- 每次循环前，先求循环条件的值
  - 当条件成立时，就执行循环体
  - 接着再次求循环条件的值，以确定循环体是否再次被执行
- 若循环条件一开始就为假，或者某次循环后循环条件变为假，则结束循环操作

实例

计算5!:

```
int main( ) {
   int i, t;
   t = 1; i = 2;
   while(i <= 5) {t = t * i; i = i + 1;}
   printf("%d", t);
   return 0;
}
```

do-while 循环结构

- 由一个循环条件和一个（组）循环操作语句（循环体）组成
- 一般形式：

```
do
    循环体
while ( 循环条件 );
```

do-while循环结构执行过程

- 每次循环前，先执行循环体，接着求循环条件的值。当条件成立时，再执行循环体
- 如此反复，直到循环条件的值为假，则结束循环操作

实例

求 s = 1 + 2 + 3 + … + 100

用do-while语句可描述成：

```
s = 0;   i = 1;
do {
 s += i;
 i++ ;
} while (i <= 100);
```

for 循环结构

- 由为循环有关**变量赋初值的表达式**、**循环条件**、循环后对**变量的修正表达式**和循环执行的**循环体**组成
- 一般形式

```
for(赋初值表达式；循环条件表达式；修正表达式)
    循环体
```

for 循环结构执行过程

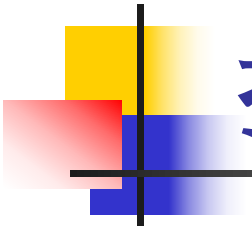
- 循环前，先执行赋初值表达式，为循环中的有关变量赋初值
- 求循环条件的值
  - 若条件不成立，则结束循环
  - 反之，执行循环体
- 求变量修正表达式，更新有关变量的值；接着再次求循环条件
- 如此反复，直到条件为假，结束循环

实例

求 s = 1 + 2 + 3 + … + 100

用for语句可以写成：

```
for(s = 0, i = 1; i <= 100; i++)
  s += i;
```



提要

- 计算机简介
- 程序设计基本概念
- 结构化程序设计
- **C语言基础知识**
- 高级语言程序开发环境
- 小结

Dennis M. Ritchie（丹尼斯•里奇）

Dennis M. Ritchie(续)

- Harvard University获学士和博士学位
- 1967年加入Bell实验室工作至今
- 1972年发明C语言
- 1978年， Brian W.Kernighan和Dennis M.Ritchie 合著《The C Programming Language》
- Turing奖(1983)和IEEE先驱奖(1992)获得者、美国工程院院士

C语言发展史

- 1960年出现的ALGOL 60是一种面向问题的高级语言，它离硬件比较远，不宜用来编写系统程序
- 1963年英国的剑桥大学推出了CPL (combined programming language)语言。CPL语言在ALGOL 60的基础上接近硬件一些，但规模比较大，难以实现
- 1967年英国剑桥大学的Martin Richards对CPL语言做了简化，推出了BCPL (basic combined programming language) 语言

C语言发展史（续）

- 1970年美国贝尔实验室的Ken Thompson 以BCPL语言为基础，设计出了简单且很接近硬件的B语言 (取BCPL的第一个字母)， 并用B语言写了第一个UNIX操作系统。但B语言过于简单，功能有限
- 1972年至1973年间，贝尔实验室的D.M.Ritchie在B语言的基础上设计出了C语言(取BCPL的第二个字母)。C语言既保持了BCPL和B语言的优点(精练，接近硬件)，又克服了它们的缺点(过于简单，数据无类型等)。
- 1973年，K.Thompson和D.M.Ritchie两人合作把UNIX的90%以上用C改写，即UNIX第5版

C语言发展史（续）

- 1983年，美国国家标准化协会(ANSI)根据C语言问世以来各种版本对C的发展和扩充，制定了新的标准，称为ANSI C
- 1987年，ANSI又公布了新标准 — 87 ANSIC，称为C89
- 1990年，国际标准化组织ISO (International Standard Organization) 接受87 ANSI C为ISO C 的标准(ISO 9899—1990)，称为C90
- ISO分别在1999和2011推出C99和C11标准; 2018年6月ISO发布了C18（或C17）；**2022年9月3日ISO发布C23，这是最新的标准**

一个C程序实例

一个只输出一行信息的C程序

```
#include  <stdio.h>
int main( )       /* 主函数 */
{
 printf("This book is <Programming with C and c++ languages>.\n");
 return 0;
 }
```

C程序特点

- 一个C程序有且只有一个名为 main 的主函数
- 主函数前的关键字int表示该函数返回一个整型值
- 在函数名之后要有一对圆括号，里面定义参数
- 函数体用花括号“{ }” 括住。花括号可以用来括起任何一组C代码，从而构成复合语句或分程序
- 简单C语句之后有一个分号“;”
- 程序中的“/* … */”表示程序的注释部分。注释便于人阅读程序，对程序编译和运行都没有作用
- #include <stdio.h>是编译预处理命令行，把有关输入和输出标准函数包含到程序中

【例】读入两个整数，输出它们的和

```
/* 1 */ #include  <stdio.h>
/* 2 */ int main()
/* 3 */ {  /* 变量定义部分 */
/* 4 */    int x, y, sum; /* 定义 x, y, sum */
/* 5 */    /* 以下为语句序列 */
/* 6 */    printf("Input x and y\n");/*提示输入数据*/
/* 7 */    scanf("%d%d", &x, &y); /* 输入x和y的值 */
/* 8 */    sum = x+y ; /* 完成x+y的计算,求sum=x+y */
/* 9 */    printf("x + y = %d\n", sum); /* 输出结果 */
/* 10 */  return 0;
/* 11 */}
```

【例】利用公式：C = （5/9）（F-32）输出F氏温度与C氏温度对照表，设已知F氏温度取0、20、…、200。

```
#include  <stdio.h>
int main()
{  float f, c;          /* 变量定义 */
   int lower, upper, step;
   lower = 0;   upper = 200;
   step = 20;   f = lower;
   while (f <= upper) {  /* 循环计算 */
     c = 5.0/9.0 * (f - 32.0);
     printf("%3.0f   %6.1f\n", f, c);
     f = f + step;
   }
   return 0;
}
```

【例】输入两个实数，输出它们中的小的数

```
#include  <stdio.h>

float min(float a, float b)
{  float temp ;     /* 函数使用的变量的定义 */
   if (a < b) temp = a; else  temp = b;
   return temp ; /* 返回 temp 到调用 min() 函数处 */
}

int main()
{  float x, y, c;          /* 变量定义 */
   printf("输入x和y. \n");
   scanf("%f%f", &x, &y );
   c = min(x, y);          /* 调用函数 min() */
   printf("MIN(%.2f, %.2f) = %.2f\n", x, y, c);
   return 0;
}
```

函数（Function）

- C程序由若干函数组成。每个程序都有一个主函数-main()函数
- 一个函数由函数头和函数体组成。函数头包括函数返回值类型、函数名、函数形式参数名，形式参数类型。
- 函数结构的一般形式

```
函数返回值类型  函数名(参数说明表)
{ 说明和定义部分;
  执行语句序列
}
```

函数（续）

- 函数可以没有参数，但函数名之后的一对圆括号是必须的
- 函数体是函数头之后用一对花括号括住的部分
- 函数体用于描述实现函数功能的代码，包括
  - 说明和定义部分
    - 说明数据结构(类型)和定义函数专用的局部变量等
  - 执行部分
    - 由C语句和控制结构代码组成

【例】统计输入字符行中各数字符、空白符与制表符以及其它字符的出现次数。

```
#include  <stdio.h>
int main()
{ int c, k, nWhite, nOther;
  int nDigit[10]; /* 定义有10个数字符的计数器 */
  nWhite = nOther = 0;
  for(k = 0; k < 10; k++)  nDigit[k] = 0;
  printf("Enter string line\n");
  while ((c = getchar()) != '\n')
    if (c >= '0' && c <= '9')  /* 如是数字符 */
       ++nDigit[c - '0'];/* 对应数字符的计数器增1 */
    else if (c == ' ' || c == '\t')  ++nWhite;
    else  ++nOther;  /* 其它字符计数器增1 */
    for(k = 0; k < 10; k++)  /* 输出对应数字符及其出现次数 */
       printf("%c : %d\n", '0'+k, nDigit[k]);
    printf("white space : %d\n other : %d\n\n", nWhite, nOther);
    return 0;
}
```

C语言词汇

- 基本符号
  - 数字10个(0～9)
  - 英文字母大、小写各26个(A～Z,a～z)
  - 下线字符“_”
  - 其它构成特殊符号的字符集
- 基本词汇
  - 常量
  - 特殊符号（运算符）
  - 关键字
  - 标识符（命名数据对象）

关键词（Keyword）

auto break case char const

continue default do double else

enum extern float for goto

if int long register return

short signed static struct switch

typedef union unsigned void

volatile while

关键词（续）

- 下面几个虽不属于关键字，但建议把它们看作关键字，不要在程序中随便使用。它们用在C程序的预处理命令行中

define　undef　include　ifdef
ifndef
endif　line　elif

标识符（Identifier）

- 作用：标识变量、常量、类型、函数、语句等程序对象，C语言用标识符给它们命名
- 命名规则：在C语言中，一个合理的标识符由英文字母或下线符开头，后跟由字母、下线符、数字符组成的字符序列
- 一般以下线符开头的标识符作系统内部使用

标识符（续）

- **命名要求**：标识符作为程序成分对象的名称，为了便于联想和记忆，建议使用能**反映该对象意义**的标识符
- **限制**：注意不同C系统对标识符的有效字符个数有不同的规定
  - 对于限制标识符最多8个有效字符的系统来说，两个超过8个字符的不同标识符，若最开始的8个字符相同，则系统认为它们是同一标识符

C语言数据类型

- 三种数据类型
  - 基本数据类型、指针类型、复合数据类型
- 基本数据类型三种
  - 整型(short, int, long)
  - 实型(float, double, long double)
  - 字符型(char)

C语言数据类型(续)

- 复合数据类型
  - 数组、结构、联合和枚举
- 指针类型
  - 指针类型用于表示程序对象在内存中的地址

常量（Constant）

- 在程序运行过程中，其值不能改变或不允许改变的数据对象
- 常量按值的表示形式区分它的类型
  - 整型常量：15
  - 浮点型常量：5.0
  - 字符型常量：`a'
  - 指针常量：NULL
  - 字符串常量：“ABC”

常量（续）

- 可用宏定义给常量命名
- 其一般形式是

```
#define 标识符   字符列
```

如：#define PI 3.14159

变量（Variable）

- 在程序运行过程中，其值可以改变的数据对象
- 变量在内存中占据一定的存贮单元，存放变量的值
- 与变量有关的概念
  - 变量名、变量类型、变量在程序中的有效作用范围、变量在程序执行期间的存在时间等

变量（续）

- 程序通过变量定义引入变量，变量定义的一般形式：

类型　变量名列表；

其中，变量名列表由一个或多个变量名组成。

例如：

```
int i,j,sum;/*定义三个int型变量 */
int index = 100, big_int = 10000
```

提要

- 计算机简介
- 程序设计基本概念
- 结构化程序设计
- C语言基础知识
- 高级语言程序开发环境
- 小结

C程序从开发到运行的六个阶段

- C程序从开发到运行大致要经历六个阶段
  - 编辑（Edit）
  - 预处理（Pre-process）
  - 编译（Compile）
  - 链接（Link）
  - 加载（Load）
  - 执行（Execute）

编辑、预处理与编译

- **编辑**：程序员用系统环境提供的编辑器编辑源程序，产生一个源程序文件.c
- **预处理**：编译前，C编绎器先自动调用预处理程序，对源程序文件作转换，产生一个新的**内部程序代码**
- **编译**：若编译过程中发现程序有错误，则输出错误的详细信息；否则，对正确的源程序产生**机器语言程序**，称为源程序的**目标代码**

连接、加载与执行

- **连接**：连接程序将目标代码和一些库函数的目标代码连接起来，产生计算机可直接执行的文件（可执行文件）
  - 静态连接vs.动态连接
- **加载**：将要执行的程序装入内存
- **执行**：装入内存的程序在计算机的操作系统控制下执行

提要

- 计算机简介
- 程序设计基本概念
- 结构化程序设计
- C语言基础知识
- 高级语言程序开发环境
- 小结

小结

- 程序是为解决某一问题、用某种编程语言描述要解决的问题或者解决该问题的方法与过程的语句序列
- 程序设计是基于某种程序语言设计和编制程序的过程
- 结构化程序设计方法是常用的、比较有效的程序设计方法
- C语言是一种曾经的主流过程化高级程序设计语言
- 一个高级语言程序在运行前，要经过预处理、编译、连接和加载等四个过程