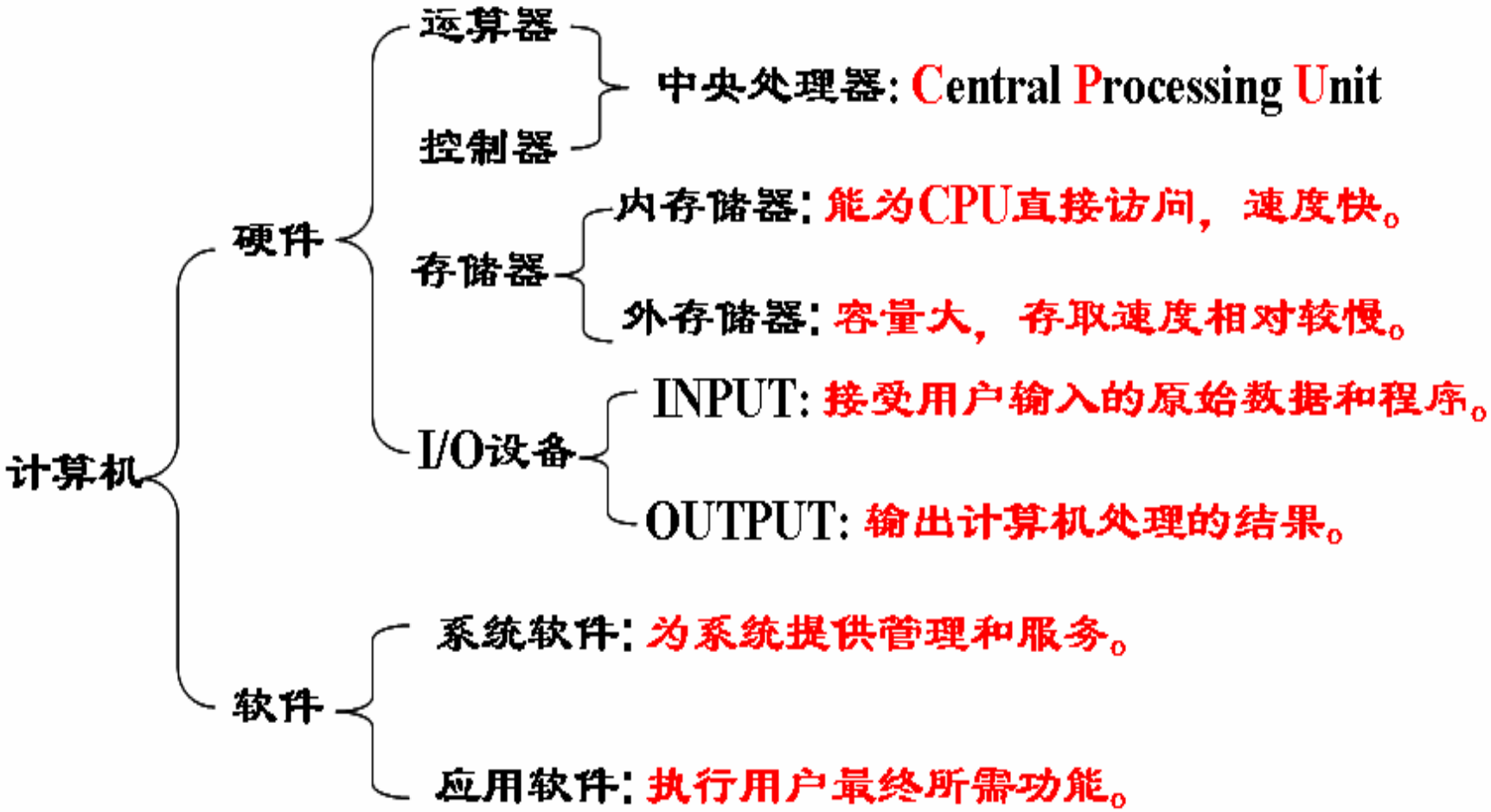
第1章 计算机基础知识

§1 计算机的组成


计算机
硬件
运算器
控制器
中央处理器: Central Processing Unit
存储器
内存储器: 能为CPU直接访问，速度快。
外存储器: 容量大，存取速度相对较慢。
I/O设备
INPUT: 接受用户输入的原始数据和程序。
OUTPUT: 输出计算机处理的结果。
软件
系统软件: 为系统提供管理和服务。
应用软件: 执行用户最终所需功能。

§2 进位计数制及其转换

一、常用计数制

1.十进制

基本数码：0、1、2…9

以10为基，逢10进1

2.二进制

基本数码：0、1

以2为基，逢2进1

3.八进制

基本数码：0、1、2…7

以8为基，逢8进1

4.十六进制

基本数码：0、1…9、A (a)、B(b)、C(c)、D(d)、E(e)、F(f)

以16为基，逢16进1

§2 进位计数制及其转换

二、十进制数转化为二进制数

整数部分：

对整数部分不断除2，以其商再除2，直至商为0。

所有的余数即为所求的二进制结果。

小数部分

对小数部分不断乘2，以乘积的小数部分再乘2，直至小数部分全0或达精度要求为止。

所有乘积的整数部分为结果。

§3 原码、反码和补码

—背景：用1字节存放1个整数

一、数的原码表示

以最高二进位作为数的符号位（0表+，1表-），
其余二进位用作数值位的表示方法。

eg1: $7_{原码}$ → 00000111

eg2: $-7_{原码}$ → 10000111

eg3: $12_{原码}$ → 00001100

eg4: $-12_{原码}$ → 10001100

eg5: $127_{原码}$ → 01111111　128 溢出

eg6: $-127_{原码}$ → 11111111　-128 溢出

eg7: $+0_{原码}$ → 00000000

eg8: $-0_{原码}$ → 10000000

数的表示范围：$-(2^7-1) \sim (2^7-1)$

0的表示不唯一

§3 原码、反码和补码

—背景：用1字节存放1个整数

二、数的反码表示

1.正数的反码:同正数的原码

2.负数的反码：符号位用1表示-号,其余数值位为原码逐位求反

—— 0的表示不唯一

eg1: $+7_{反码}=+7_{原码}=$ 00000111

eg2: $-7_{原码}=$ 10000111　则　$-7_{反码}=$ 11111000

eg3: $+0_{反码}=+0_{原码}=$ 00000000

eg4: $-0_{原码}=$ 10000000, $-0_{反码}=$ 11111111

§3 原码、反码和补码

—背景：用1字节存放1个整数

三、数的补码

- 正数的补码：同正数的原码
- 负数的补码：原码数值位求反(获得反码),加1

补码表示

- 取值范围：

eg1: $+7_{补码} = +7_{反码} = +7_{原码} =$ 00000111

eg2:求$-7_{补码}$ ⟶ 10000111 ⟶ 11111000 ⟶ 11111001

eg3:$+0_{补码}$=00000000

eg4:$-0_{补码}$ ⟶ 10000000 ↓ 11111111 ↓ 00000000

§3 原码、反码和补码

—背景：用1字节存放1个整数

三、数的补码

正数的补码：同正数的原码

补码表示

负数的补码：原码数值位求反(获得反码),加1

取值范围：$-2^7 \sim 2^7-1$

-0	00000000
-1	11111111
-2	11111110
-3	11111101
…	……
-127	10000001
-128	10000000

正数域：00000000 ~ 01111111

负数域：10000000 ~ 00000000

§4 计算机语言

一、机器语言(低级语言)

1.机器指令：硬件能直接识别并产生有效动作的二进制数字串

2.机器语言：某种硬件平台支持的机器指令集合即该计算机的机器语言

3.机器语言程序：用机器指令所编写的解题程序

—— 是可执行程序(executing program)，执行速度最快

—— 直观性差、不便于调试

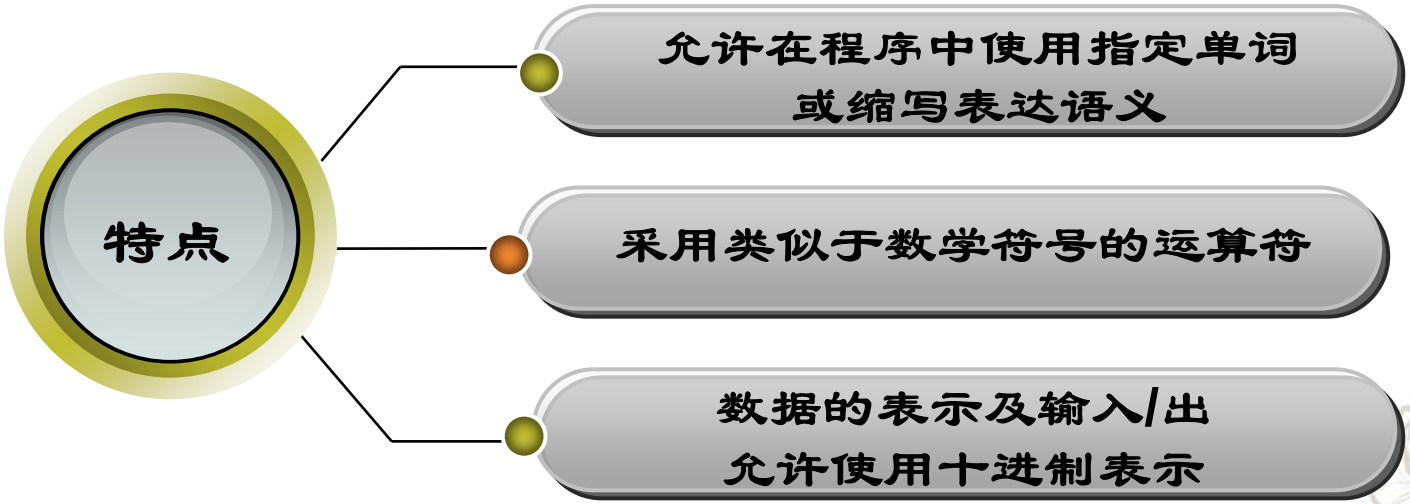
—— 移植性差、程序无兼容性

二、汇编语言(中间语言)

- 引入助记符,形成与机器指令对应的汇编指令,加强程序直观性
- 需汇编，运行速度较机器语言程序有所下降
- 与硬件一一对应，无兼容性

§4 计算机语言

三、程序设计语言(高级语言)



1.源程序(.c或.cpp)：用高级语言所编写的解题程序

2.编辑程序：帮助用户输入、修改、保存源程序的程序

3.编译程序：将源程序转换成更接近于机器表达方式的，功能和源程序等价的目标程序(Object program)

4.链接程序对目标程序及其库函数实施链接装配，形成可执行程序(Execute Program)

§4 计算机语言

三、程序设计语言(高级语言)

C程序的执行步骤

1.编辑
通过编辑程序将源程序输入计算机
eg: a.c或a.cpp

2.编译
通过编译程序生成功能与源程序等价的目标程序
eg:a.obj

3.链接
通过链接程序生成可独立运行的可执行程序
eg:a.exe

4.运行
启动可执行程序(.exe)的运行,获得结果

1.源程序(.c或.cpp)：用高级语言所编写的解题程序

2.编辑程序：帮助用户输入、修改、保存源程序的程序

3.编译程序：将源程序转换成更接近于机器表达方式的，功能和源程序等价的目标程序(Object program)

4.链接程序对目标程序及其库函数实施链接装配，形成可执行程序(Execute Program)

§5 算法(Algorithm)

一、算法定义

为解决一个问题所采取的方法以及步骤，称为解决该问题的算法

算法特征

- **有穷性**：算法应包含有限步骤,而不能是无限的
- **确定性**：算法的每个步骤应是确定无歧义的
- **有效性**：算法的每个步骤应能有效地执行并获得结果
- **有0或多个输入**：输入是指执行算法时需从外界获取的信息
- **有1或多个输出**：输出即算法的“解”,无解的算法无意义

例1:已知a、b ,设计算法交换它们之值

算法I：设置中间变量temp

```
a => temp
b => a
temp => b
```

算法II：

```
a+b => a
a-b => b
a-b => a
```

§5 算法(Algorithm)

二、怎样表示算法

1.自然语言：方便，但不严格，有歧义

2.伪码：介于自然语言和计算机语言间的文字和符号。便于书写及修改，但无严格的语法规则，不便于交流

例1:已知a、b ,设计算法交换它们之值

算法I：设置中间变量temp

a => temp

b => a

temp => b

算法II：

a+b => a

a-b => b

a-b => a

§5 算法(Algorithm)

二、怎样表示算法

1.自然语言：方便，但不严格，有歧义

2.伪码：介于自然语言和计算机语言间的文字和符号。便于书写及修改，但无严格的语法规则，不便于交流

3.图形语言（标准、形式化的方法）

—流程图 (FC图：Flow Chart)

—改进的流程图(NS流程图)

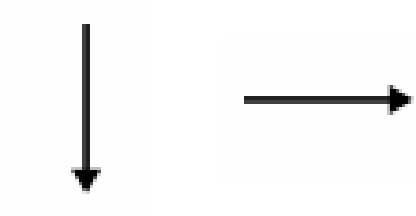
例2:求出a、b中的大者

算法伪码：

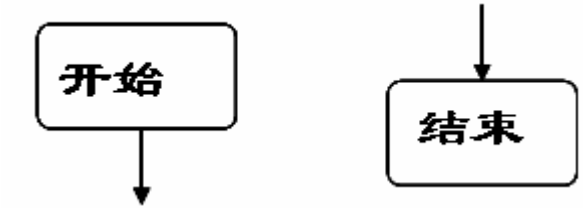
```
输入a、b;
若a>b
    max=a
否则
    max=b;
输出max;
```

§5 算法(Algorithm)

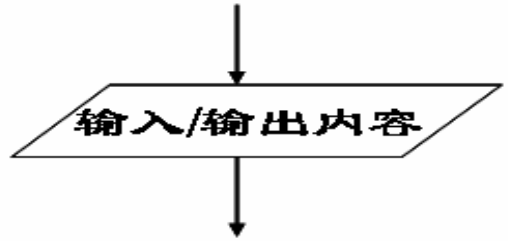
三、流程图符号(ANSI)



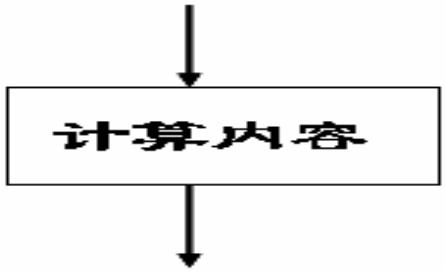
(a)流线



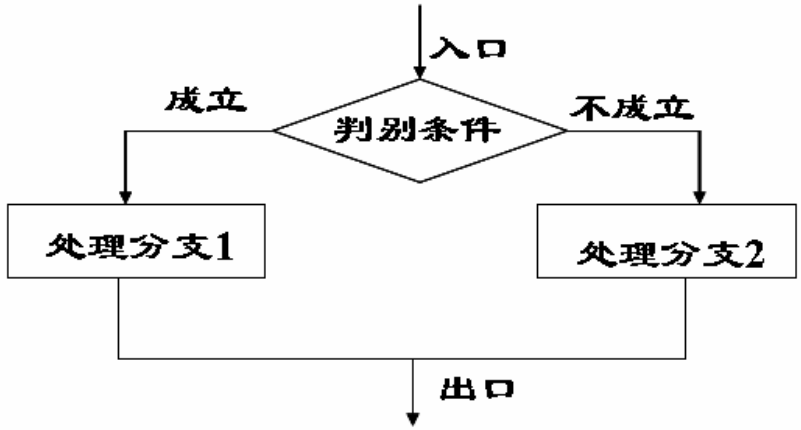
(b)起止框



(c)输入输出框

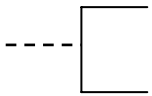


(d)处理框



(e)判别框

(f)连接点



(g)注释框

§5 算法(Algorithm)

例1: 已知a、b ,设计算法交换它们之值。

算法的伪码：

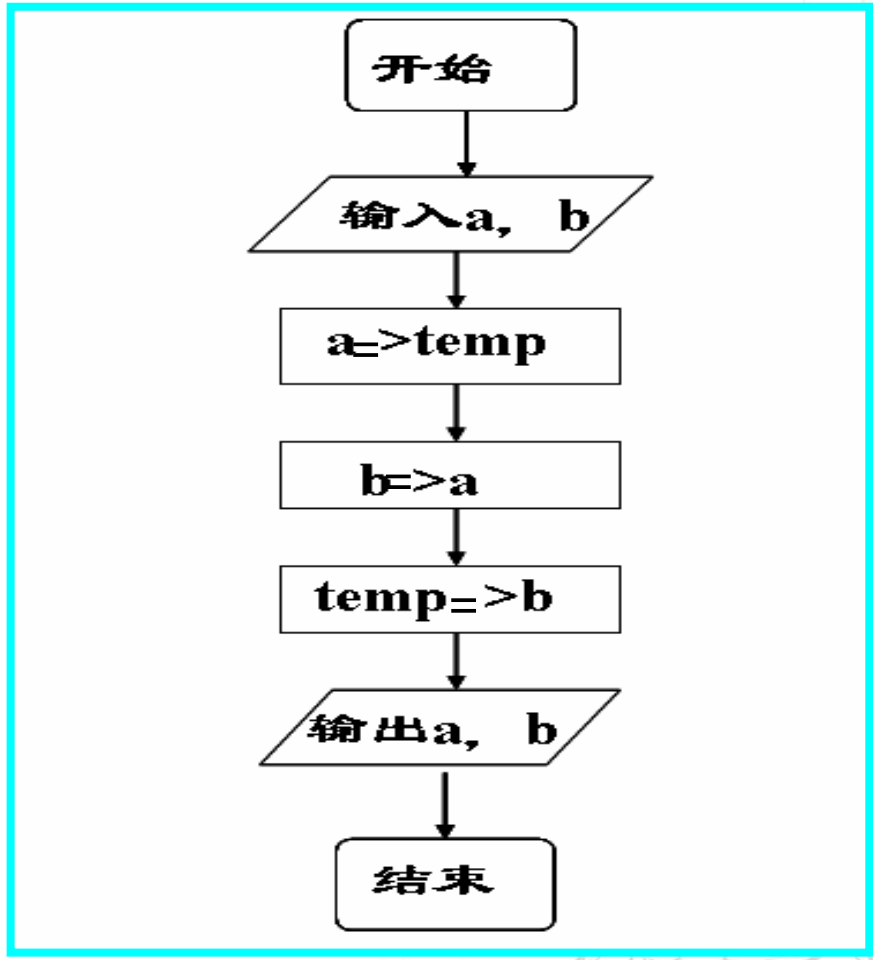
输入a、b；

a => temp；

b => a；

temp => b；

输出a、b；

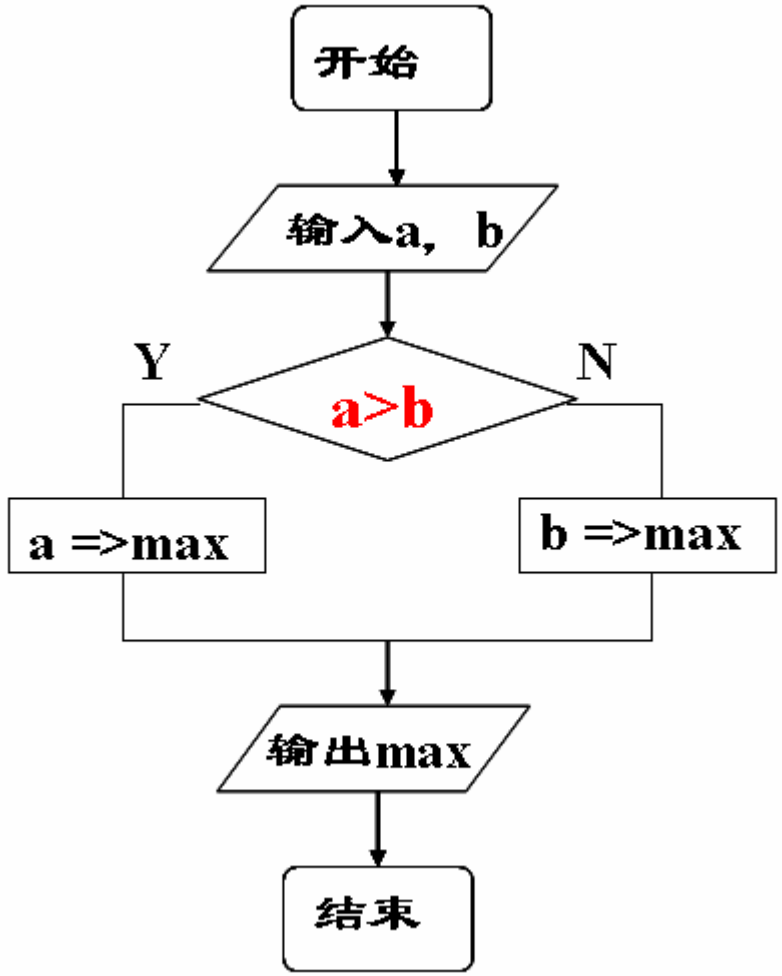


§5 算法(Algorithm)

例2:编程求a、b中的大者

算法的伪码:

输入a、b;
若a>b
　　max=a
否则
　　max=b;
输出max;



§5 算法(Algorithm)

例3:编程求1+2+3+……+10之值

算法的伪码：

s1: sum=0,i=1；

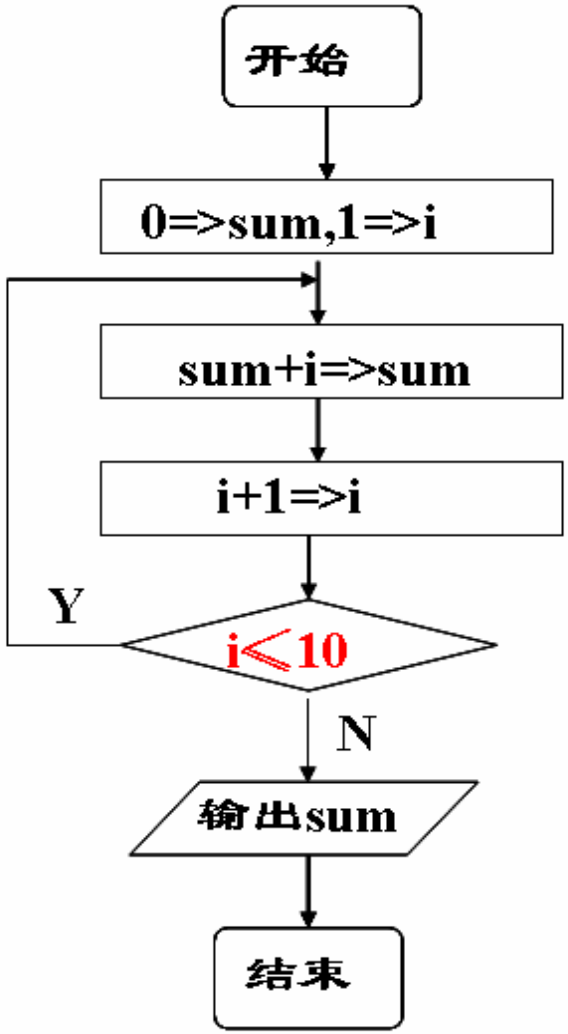
s2: sum=sum+i；

s3: i=i+1；

s4: if i≤10 返回s2;

else 累加结束；

s5:输出sum；

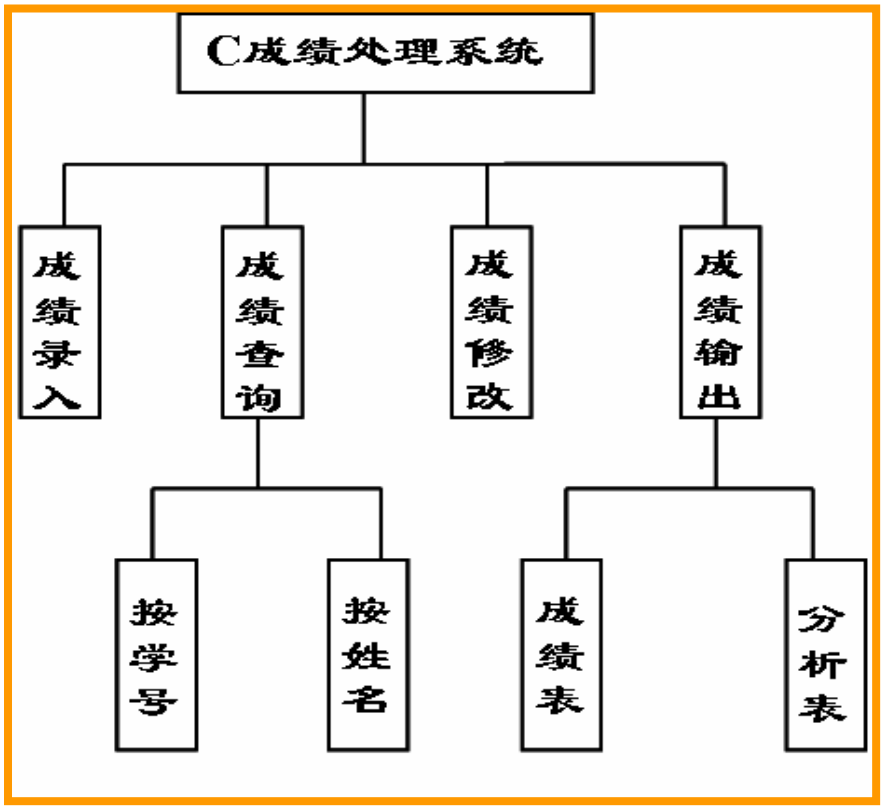


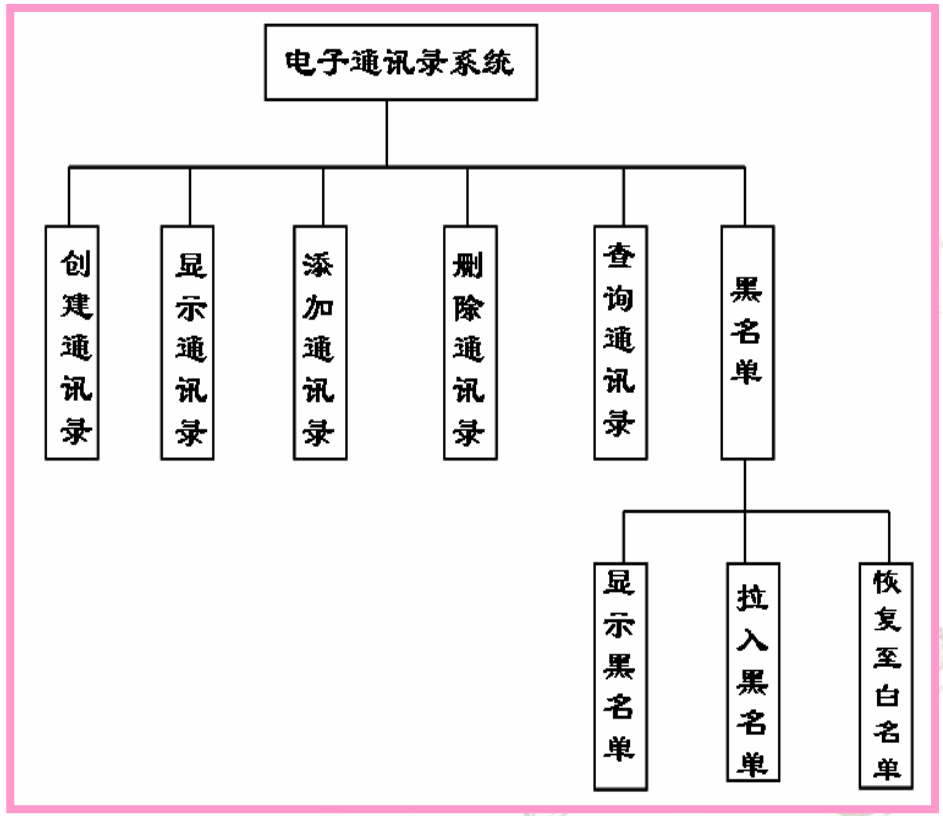
§6 结构化程序设计方法

1.软件总体设计 —— Top-Down模块化设计方法

—— 自顶向下(Top-Down)，以功能为基础实施分解，逐步细化

—— 进行模块化(Module)设计



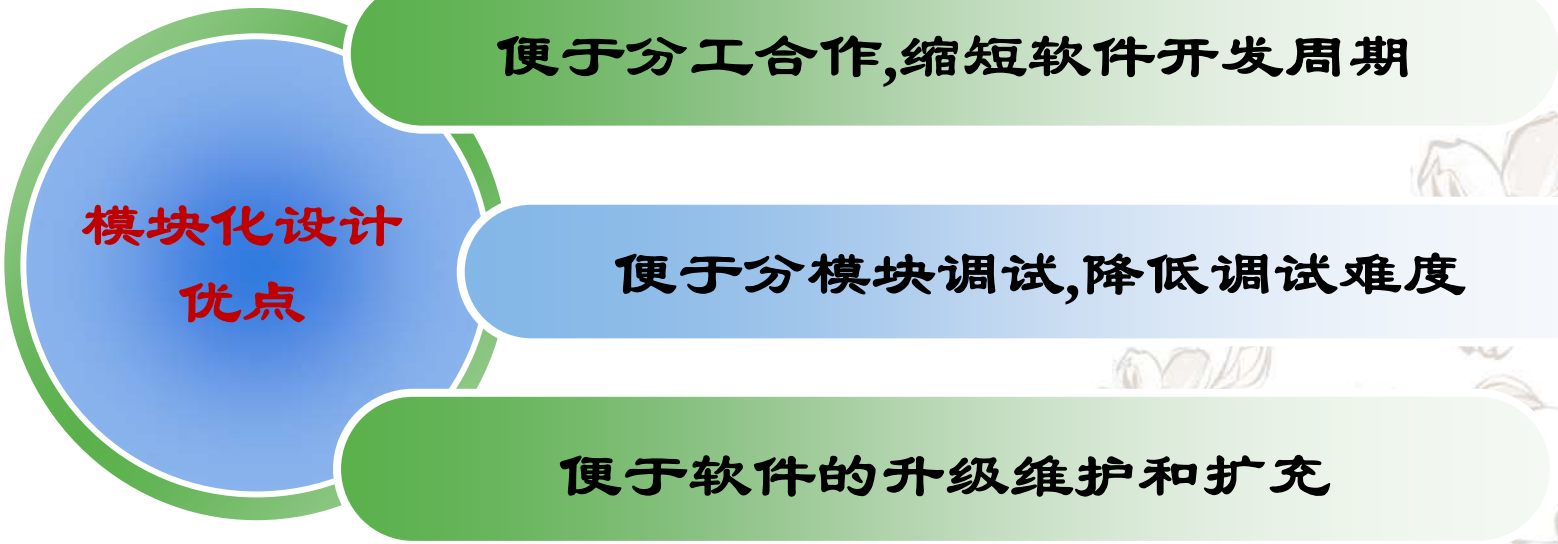


§6 结构化程序设计方法

1.软件总体设计 —— Top-Down模块化设计方法

—— 自顶向下(Top-Down)，以功能为基础实施分解，逐步细化

—— 进行模块化(Module)设计



§6 结构化程序设计方法

1.软件总体设计 —— Top-Down模块化设计方法

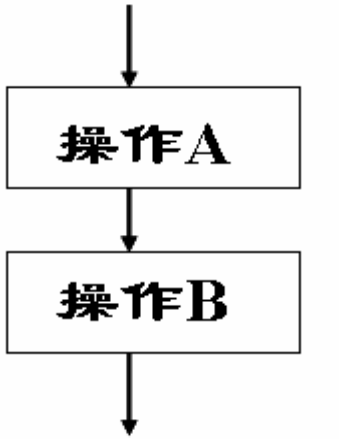
—— 自顶向下(Top-Down)，以功能为基础实施分解，逐步细化

—— 进行模块化(Module)设计

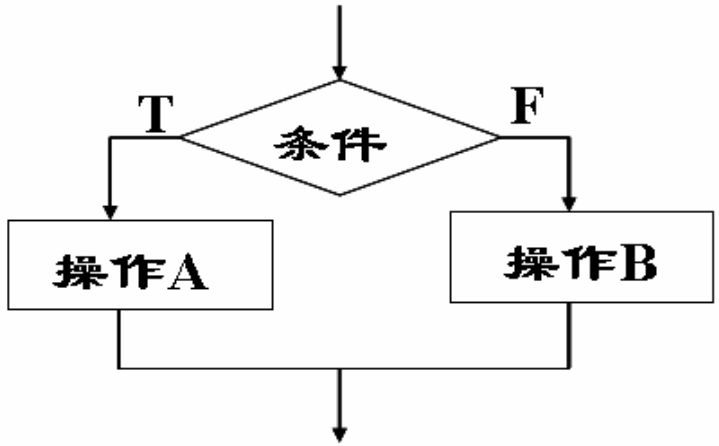
2.软件详细设计 —— 结构化编码

——节制对无条件转移语句GOTO的使用

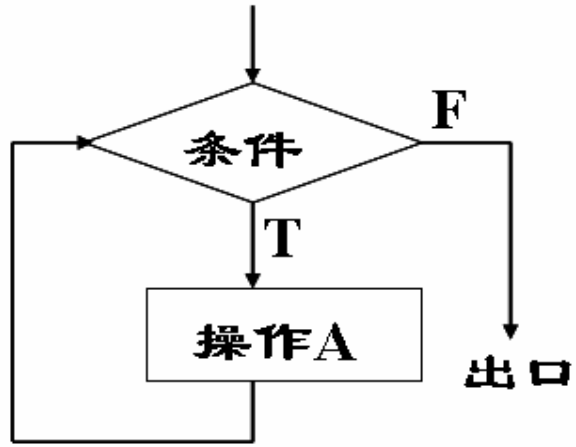
——使用三种基本控制流结构进行编码



(1) 顺序结构



(2) 分支结构



(3) 循环结构

学习指导

《C语言》课程教学设计

C语言程序设计

3学分，48学时，课堂教学和研讨相结合。
百分计分制,研究性教学,七、三评分机制

C语言实验

1学分，32实验时数，研讨和自主开发相结合。
五级记分制

C语言课程设计

2学分，为期1周，独立完成课题研发。
五级记分制