

程序设计与C语言

程 淼

E-mail: mew_cheng@outlook.com

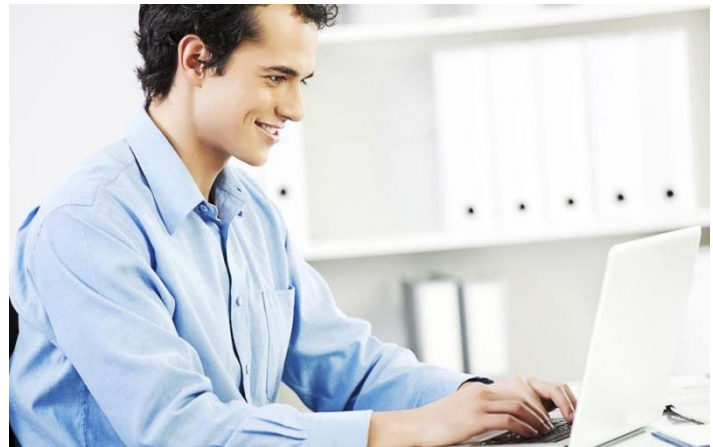


语言与程序

- ◆ 一个例子:
 - ◆ 我是中国人
 - ◆ I am Chinese
 - ◆ たしは 中国人です
 - ◆ 저는 중국사람 입니다

语言与程序

💧 作用

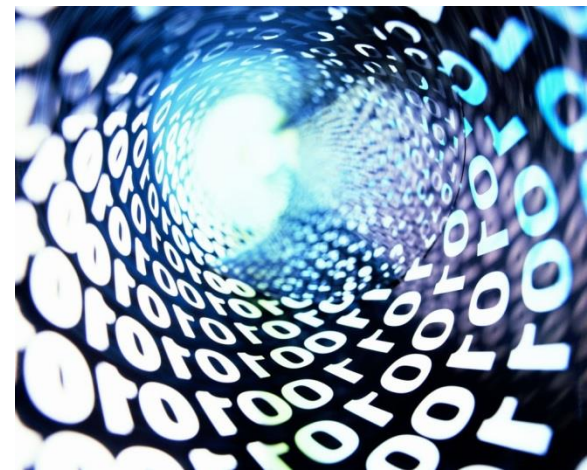


计算机程序

- ◆ 所谓**程序**，就是一组计算机能识别和执行的指令。
- ◆ 每一条**指令**使计算机执行特定的操作。

计算机语言

- 💧 **机器语言**：计算机工作基于二进制
 - 💧 在计算机发展初期，一般计算机的指令长度为16，即以16个二进制数(0或1)组成一条指令，16个0和1可以组成各种排列组合
- 💧 计算机能直接识别和接受的二进制代码称为**机器指令**
- 💧 机器指令的集合就是该计算机的**机器语言**



计算机语言

- ◆ **符号语言**用一些英文字母和数字表示一个指令
 - ◆ ADD A, B
- ◆ 需要用一种称为**汇编程序**的软件，把符号语言的指令转换为机器指令
 - ◆ 一般，一条符号语言的指令对应转换为一条机器指令

计算机语言

- ◆ **高级语言**接近于人们习惯使用的自然语言和数学语言
 - ◆ 程序中的语句和指令都是用英文单词表示的，运算符和运算表达式和人们日常所用的数学式子差不多
- ◆ 功能很强，且不依赖于具体机器，故称为计算机高级语言

计算机语言

- 💧 计算机不能直接识别高级语言程序，也要进行“翻译”
- 💧 用一种称为**编译程序**的软件把用高级语言写的程序(称为**源程序**)转换为机器指令的程序(称为**目标程序**)

计算机语言

- 高级语言经历了不同的发展阶段：
 - 非结构化的语言：初期的语言属于非结构化的语言
 - 编程风格比较随意，**程序中的流程可以随意跳转**
 - 早起的BASIC，FORTRAN和ALGOL等都属于非结构化的语言
 - 结构化语言：规定程序必须由具有良好特性的基本结构(顺序结构、分支结构、循环结构)构成
 - 程序中的流程不允许随意跳转，总是由上而下顺序执行各个基本结构
 - QBASIC，FORTRAN 77和C语言

计算机语言

- ◆ 处理规模较大的问题时，开始使用面向对象的语言
 - ◆ C++，C#，Visual Basic和Java等语言

C语言的发展

BCPL语言

B语言

C语言

1967年英国剑桥大学的 Martin Richards推出了没有类型的BCPL语言

1970年美国 AT&T 贝尔实验室的 Ken Thompson

1972—1973年间，美国贝尔实验室的 D. M. Ritchie

C语言的发展

1973年, Ken Thompson和D. M. Ritchie 合作把UNIX的90%以上用C语言改写, 即UNIX第5版

1978年, Brian W. Kernighan 和 Dennis M. Ritchie 合著了影响深远的《The C Programming Language》

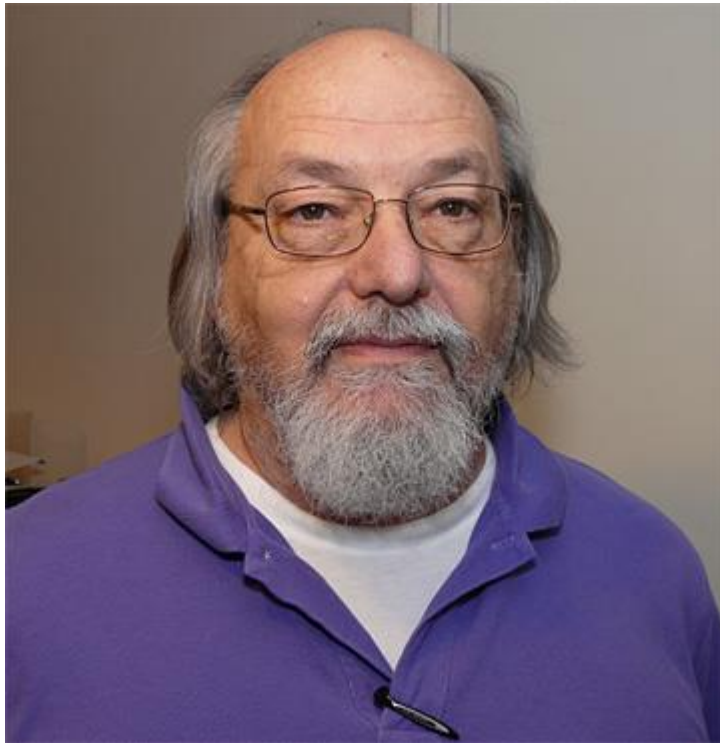
1983年, 美国国家标准协会(ANSI)成立了一个委员会, 制定了第一个C语言标准草案('83 ANSI C)

1989年, ANSI公布了一个完整的C语言标准—ANSI X3(常称ANSI C或C89)

1990年, 国际标准化组织ISO接受C89作为国际标准ISO/IEC

1995年和1999年, ISO分别对C90和C语言标准进行修订。1999、2001和2004年先后进行了技术修正。

Ken Thompson和D. M. Ritchie



C语言的特点

- 语言简洁、紧凑，使用方便、灵活
 - 一共只有37个关键字、9种控制语句
- 运算符丰富：34种运算符
- 数据类型丰富
- 具有结构化的控制语句

C语言的特点

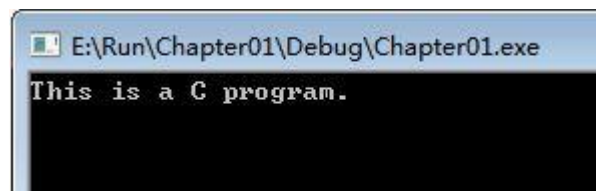
- 语法限制不太严格，程序设计自由度大
- C语言允许直接访问物理地址，能进行位(bit)操作
 - 能实现汇编语言的大部分功能，可以直接对硬件进行操作
- 可移植性好
- 生成目标代码质量高，程序执行效率高

C语言程序

💧 最简单的C语言程序

```
#include <stdio.h>

int main()
{
    printf("This is a C program. \n");
    return 0;
}
```



E:\Run\Chapter01\Debug\Chapter01.exe
This is a C program.

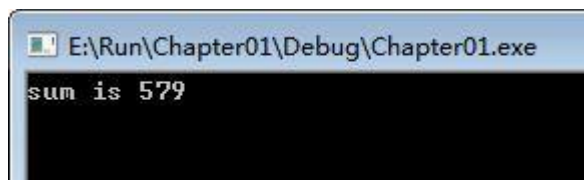
C语言程序

- ◆ 另一个例子：求两个整数之和
- ◆ 解题思路：设置3个变量，a和b用来存放两个整数，sum用来存放和数。

```
#include <stdio.h>

int main()
{
    int a, b, sum;
    a = 123;
    b = 456;
    sum = a + b;
    printf("sum is %d\n", sum);

    return 0;
}
```



```
E:\Run\Chapter01\Debug\Chapter01.exe
sum is 579
```

C语言程序

- 一个例子：求两个整数中的较大者
- 解题思路：用一个函数来实现求两个整数中的较大者

```
#include <stdio.h>

int main()
{
    int max(int x, int y);
    int a, b, c;
    scanf("%d, %d", &a, &b);
    c = max(a, b);
    printf("max=%d\n", c);

    return 0;
}
```

```
int max(int x, int y)
{
    int z;
    if (x > y)
        z = x;
    else
        z = y;

    return (z);
}
```



C语言程序

- 在UNIX操作系统中，首先必须在某个文件中建立这个源程序，例如hello.c，然后通过命令进行编译：
 - cc hello.c
- 如果源程序没有什么错误，编译过程将顺利进行，并生成一个可执行文件
 - a.out
- 即可运行a.out，打印出下列信息：
 - hello, world

C语言程序的结构

- 一个程序由一个或多个源程序文件组成
 - 预处理指令**, #include <stdio.h>
 - 全局声明**, 即在函数之外进行的数据声明
 - 函数定义**
- 函数是C程序的主要组成部分
- ald

C语言程序的结构

- 一个函数包括两个部分

- 函数首部

int max (int x, int y)

- 函数体

- 声明部分

- 执行部分

C语言程序的结构

- 在某些情况下也可以没有声明部分，甚至可以既无声明部分也无执行部分

- `void dump()`
 `{ }`

C语言程序的结构

- ◆ 程序总是从main函数开始执行的，而不论main函数在整个程序中的位置如何
- ◆ 程序中对计算机的操作是由函数中的C语句完成的
- ◆ 在每个数据声明和语句的最后必须有一个分号
- ◆ C语言本身不提供输入输出语句
- ◆ 程序应当包含注释

运行C程序

语言的沟通过程

发出语音



听闻接收

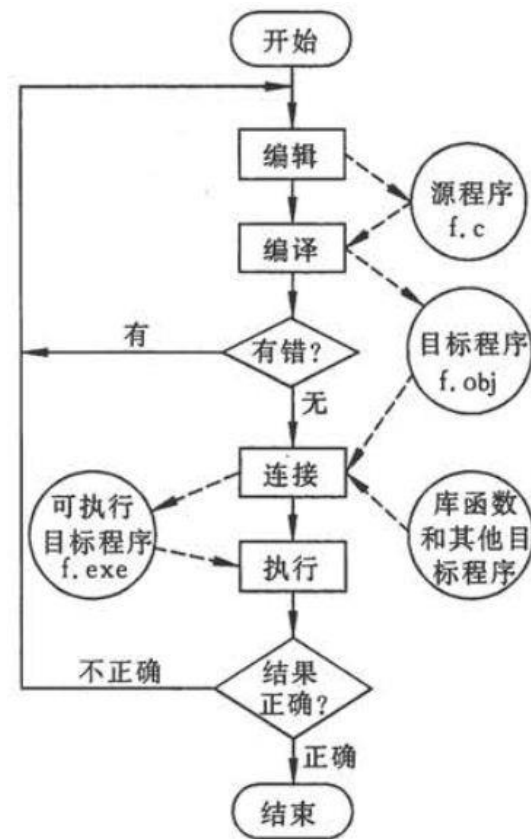


大脑理解



运行C程序

- 上机输入和编辑源程序
- 对源程序进行编译，先用C编译系统提供的“预处理器”(又称“预处理程序”或“预编译器”)
- 进行连接处理
- 运行可执行程序，得到运行结果



程序设计的任务

- ◆ 问题分析
- ◆ 设计算法
- ◆ 编写程序
- ◆ 对源程序进行编辑
- ◆ 运行程序，分析结果
- ◆ 编写程序文档

第2章 算法



算法

- 💧 程序的灵魂
 - 💧 对数据的描述：数据结构
 - 💧 对操作的描述
- 💧 算法 + 数据结构 = 程序

算法

- ◆ 事务处理的**顺序性**
- ◆ 为解决一个问题而采取的方法和步骤，就称为**"算法"**
 - ◆ 不是只有“计算”的问题才有算法

算法

💧 典型问题： $1+2+3+\dots+100 = \sum_{n=1}^{100} n$

算法

- 计算机算法 $1 \times 2 \times 3 \times 4 \times 5$ ，或将100个学生的成绩按高低分数的次序排列，**是可以做到的**
- 让计算机去执行“替我理发”或“煎一份牛排”，**是做不到的**
 - 也是可以做到的

算法

- ◆ 数值运算算法：求数值解、数值分析
- ◆ 非数值运算算法：排序算法、查找搜索算法

算法举例



算法举例

● 例2.1 求 $1 \times 2 \times 3 \times 4 \times 5$

算法举例

- ◆ 例2.2 有50个学生，要求输出成绩在80分以上的学生的学号和成绩

算法举例

- ◆ 例2.3 判定2000-2500年中的每一年是否为闰年，并将结果输出
- ◆ 闰年的条件：
 - ◆ 能被4整除，但不能被100整除的年份都是闰年，如1996年、2008年、2012年、2048年是闰年
 - ◆ 能被400整除的年份是闰年，如1600年、2000年是闰年

算法举例

例2.4 求 $1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{1}{99} - \frac{1}{100}$

算法举例

- ◆ 例2.5 给出一个大于或等于3的正整数，判断它是不是一个素数

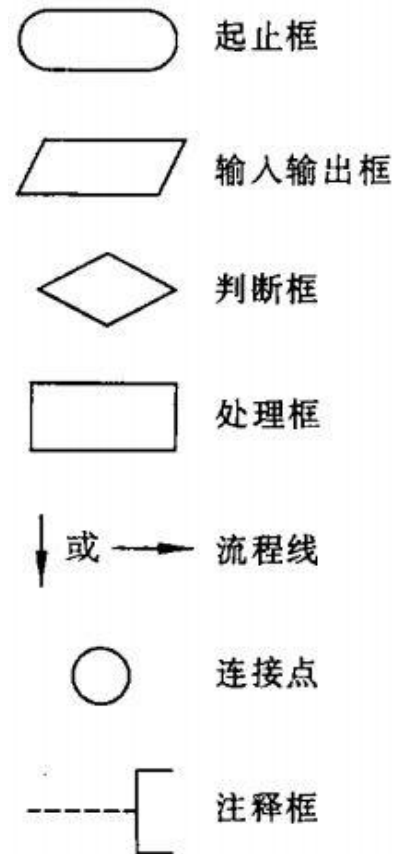
算法的特性

- **有穷性**：一个算法应包含有限的操作步骤，而不能是无限的。
- **确定性**：算法中的每一个步骤都应当是确定的，而不应当是含糊的、模棱两可的。
- **有零个或多个输入**
 - 所谓输入是指在执行算法时需要从外界取得必要的信息。
- **有一个或多个输出**
- **有效性**：算法中的每一个步骤都应当能有效地执行



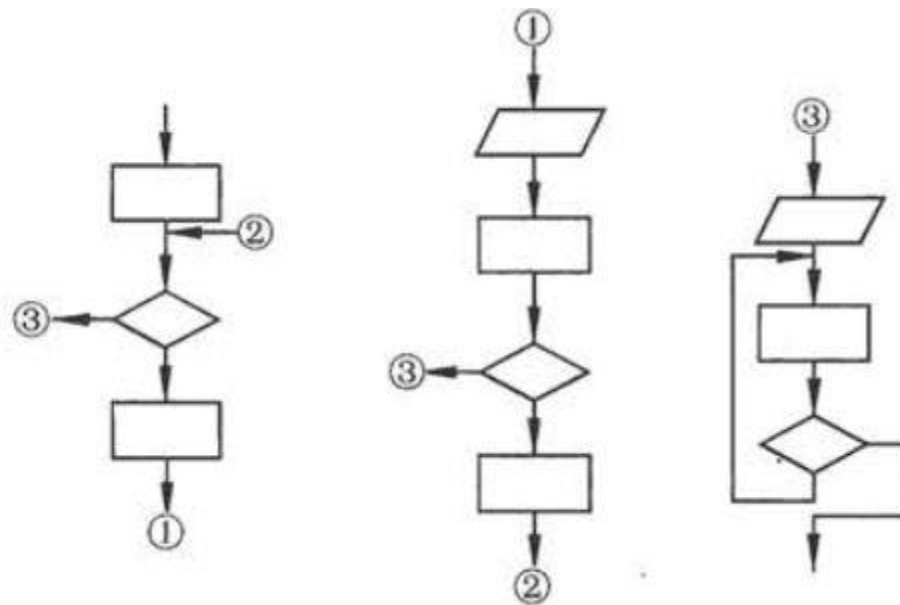
算法的表示

- 用自然语言表示算法
- 用流程图表示算法



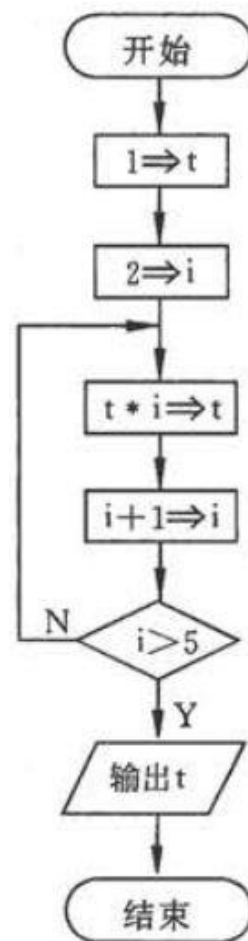
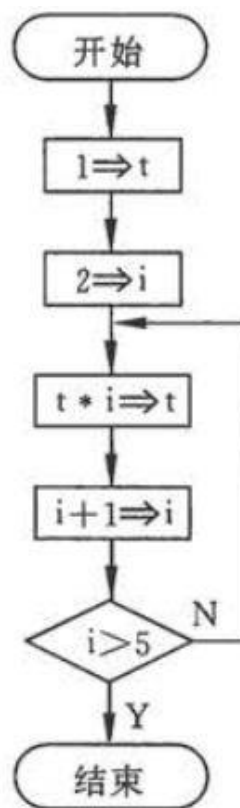
算法的表示

- ◆ **菱形框**的作用是对一个给定的条件进行判断，根据给定的条件是否成立决定如何执行其后的操作
- ◆ **连接点**(小圆圈)是用于将画在不同地方的流程线连接起来



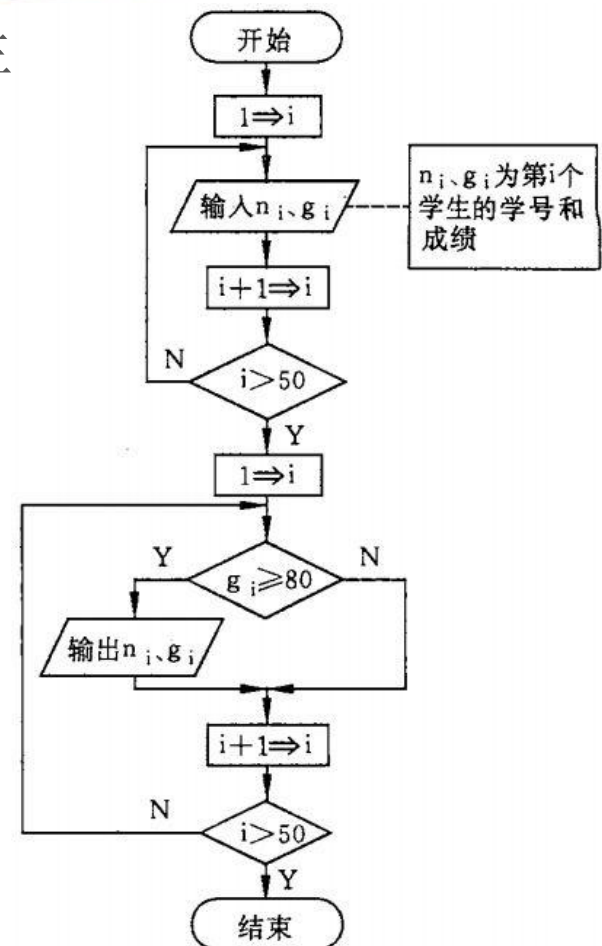
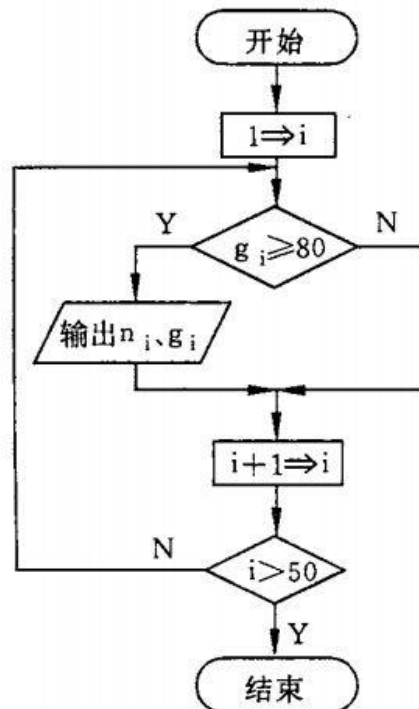
算法的表示

- 将例2.1的算法用流程图表示。求 $1 \times 2 \times 3 \times 4 \times 5$



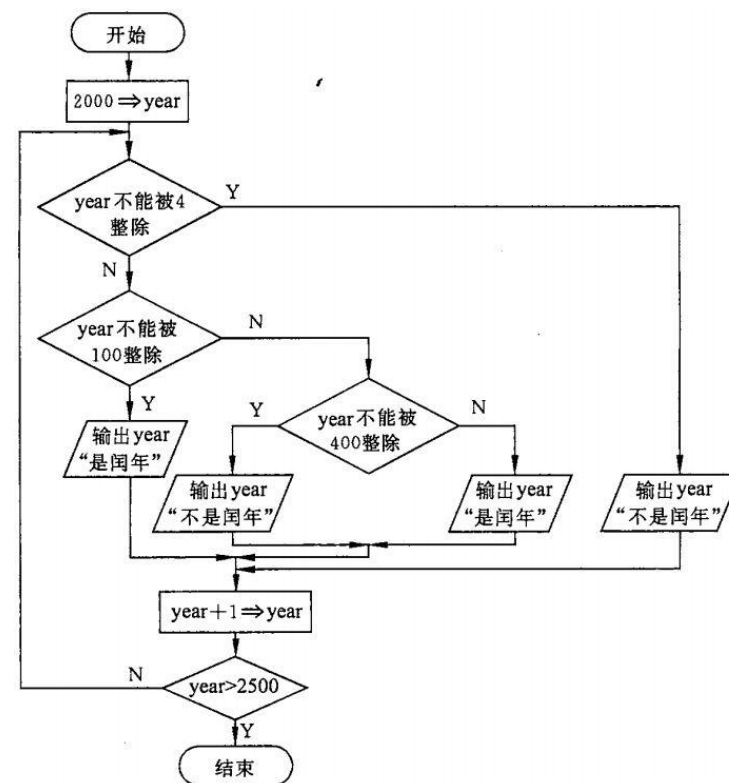
算法的表示

- 例2.2的算法用流程图表示。有50个学生要求输出成绩在80分以上的学生的学号和成绩



算法的表示

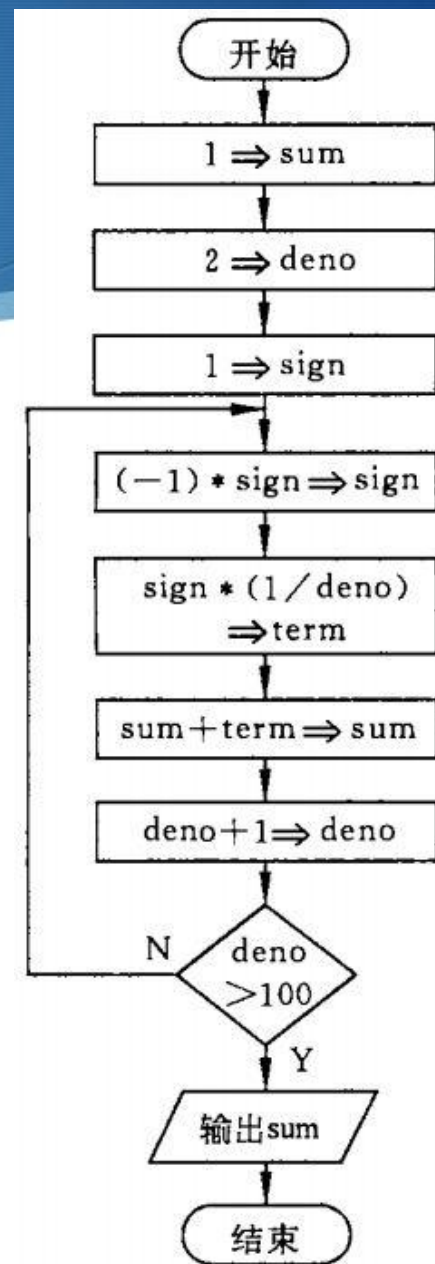
- 例2.3判定闰年的算法用流程图表示。判定2000-2500年中的每一年是否为闰年，将结果输出。



算法的表示

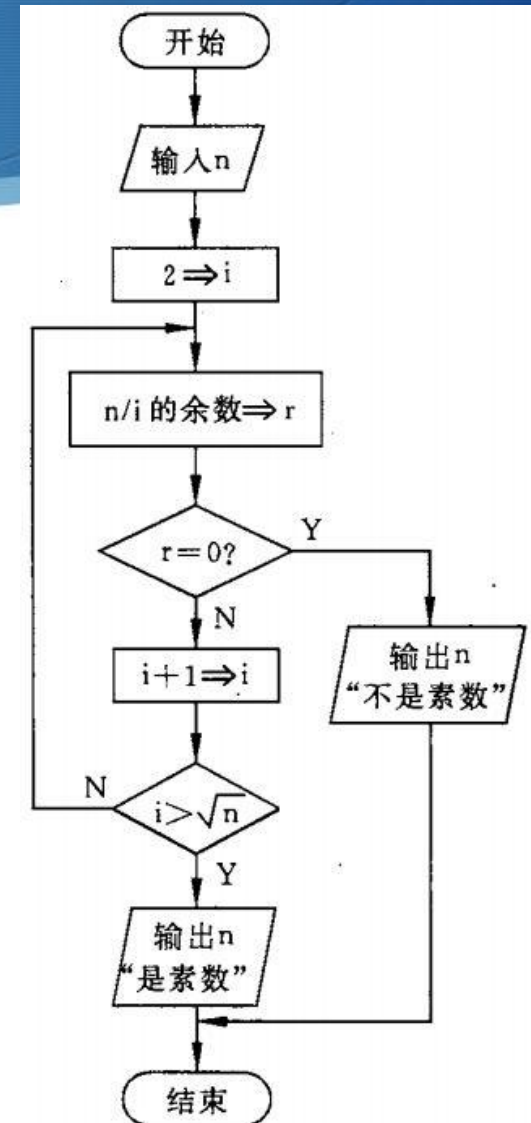
将例2.4的算法用流程图表示。

求 $1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{1}{99} - \frac{1}{100}$



算法的表示

- 例2.5判断素数的算法用流程图表示。对一个大于或等于3的正整数，判断它是不是一个素数



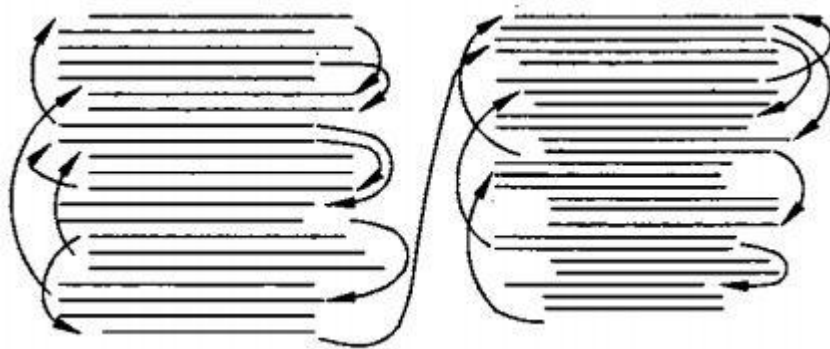
算法的特性

- 不要以为任意写出的一些执行步骤就构成了一个算法
- 一个有效算法应该具有的特点
 - 有穷性
 - 确定性
 - 有零个或多个输入
 - 有一个或多个输出
 - 有效性



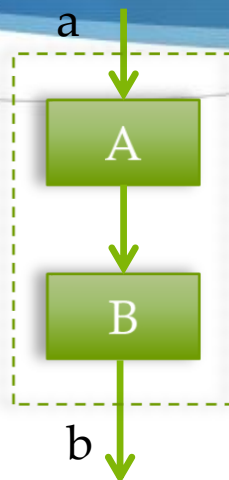
传统流程图的弊端

- ❖ 使用者可以不受限制地使流程随意地转来转去，称为**BS型算法**
- ❖ 算法执行的**随意性**和**无顺序性**

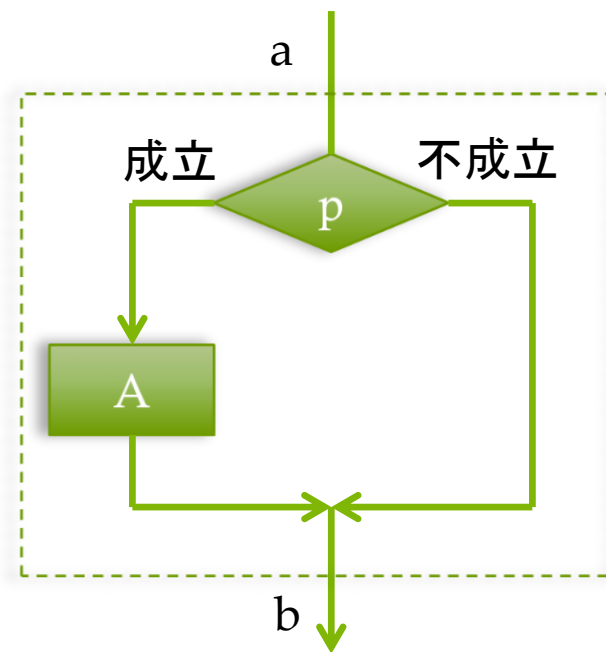
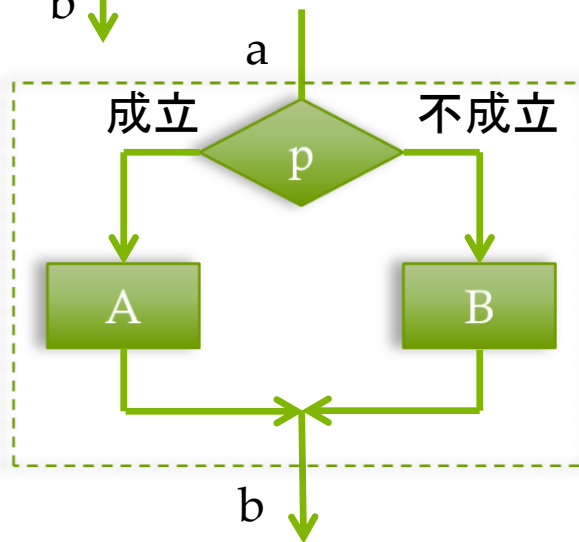


基本结构

顺序结构



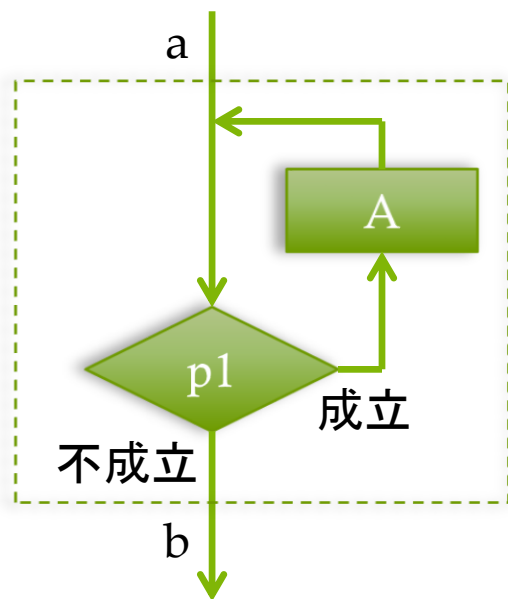
选择结构



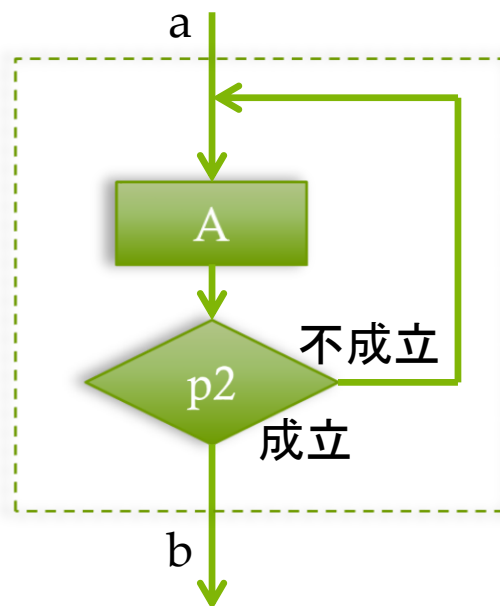
基本结构

循环结构（重复结构）：反复执行某一部分的操作

当型(while型)循环结构

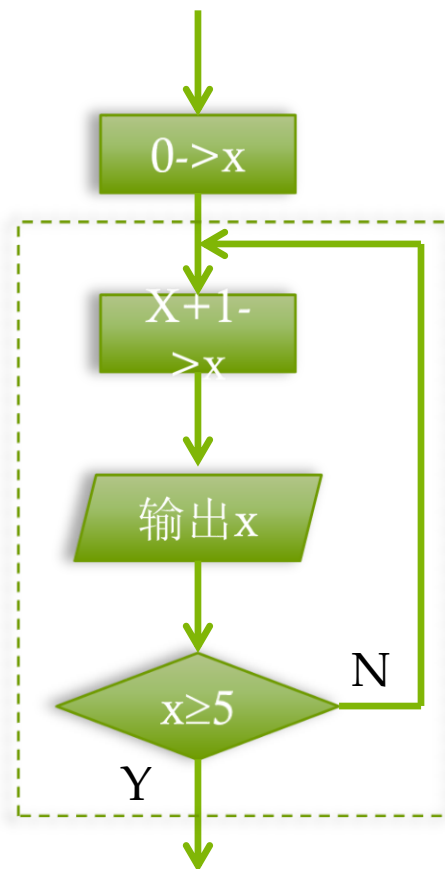
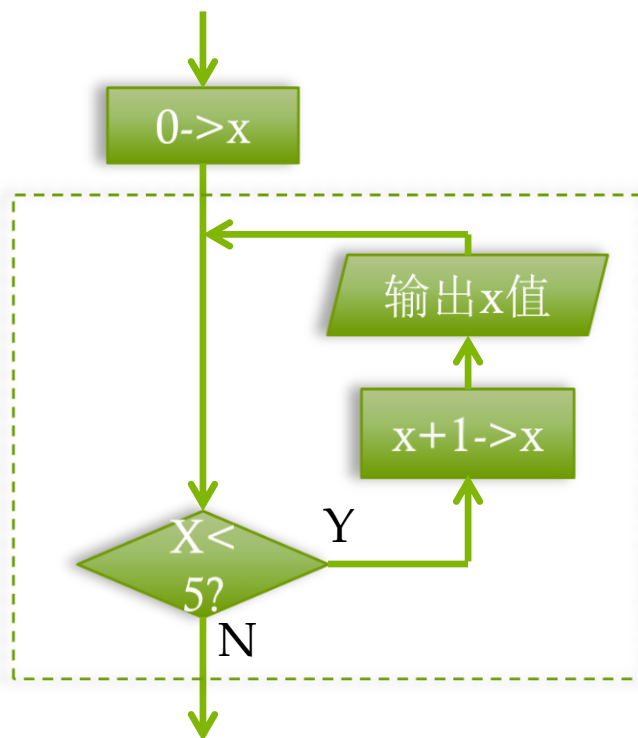


直到型(until型)循环结构



基本结构

- 对同一个问题既可以用当型循环来处理，也可以用直到型循环来处理
- 输出5个数：1, 2, 3, 4, 5



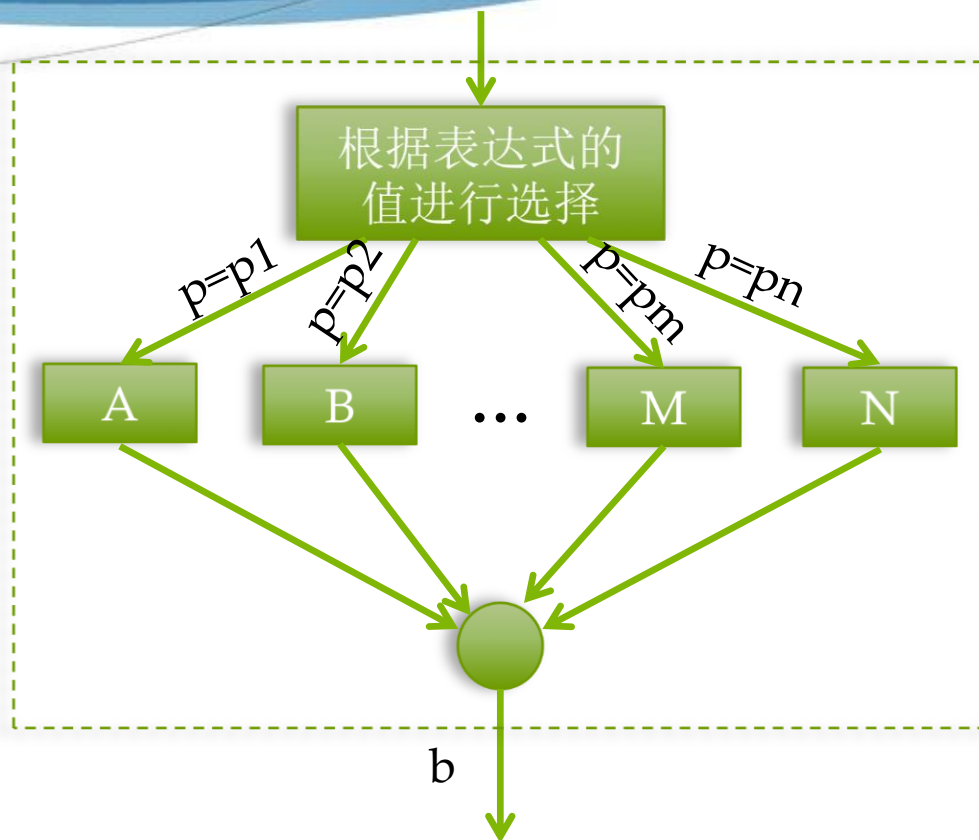
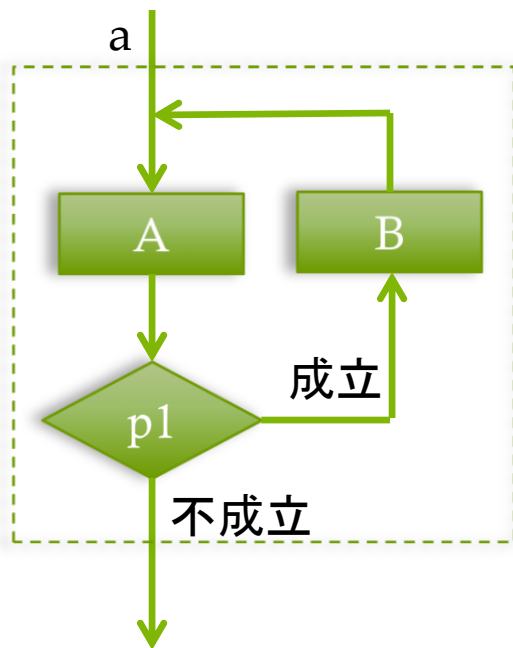
基本结构

- ◆ 共同特点

- ◆ 只有一个入口
- ◆ 只有一个出口
- ◆ 结构内的每一部分都有机会被执行到
- ◆ 结构内不存在“死循环”（无终止的循环）

基本结构

自定义的基本结构



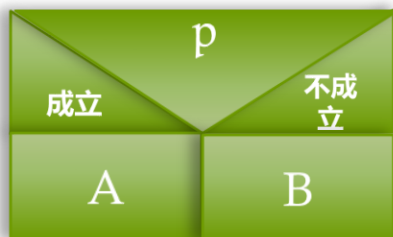
N-S流程图

- 1973年，美国学者I. Nassi和B. Shneiderman提出了一种新的流程图形式
- 在这种流程图中，完全去掉了带箭头的流程线

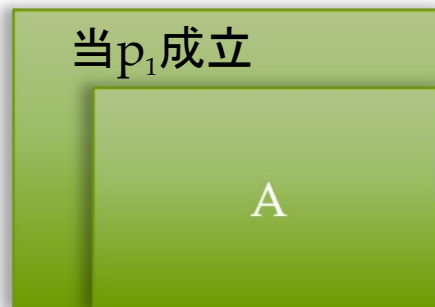
N-S流程图



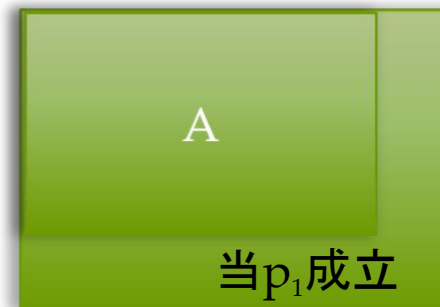
顺序结构



选择结构



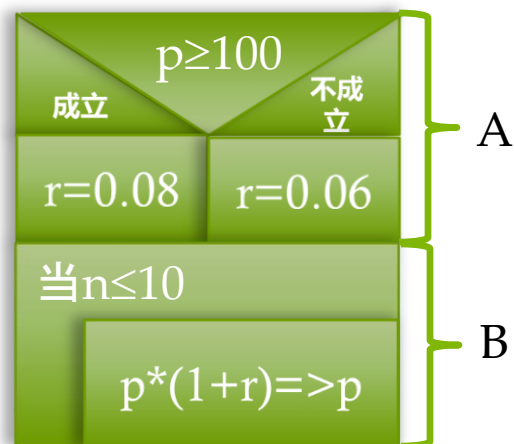
循环结构



直到型结构

N-S流程图

- 由两种结构组成的一个顺序结构
 - 其中的A框可以又是一个选择结构，B框可以又是一个循环结构



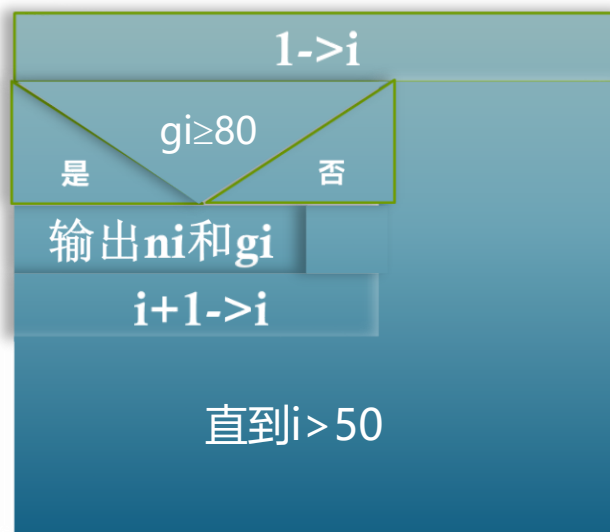
N-S流程图

- 例2.11 将例2.1的求5! 算法用N-S图表示



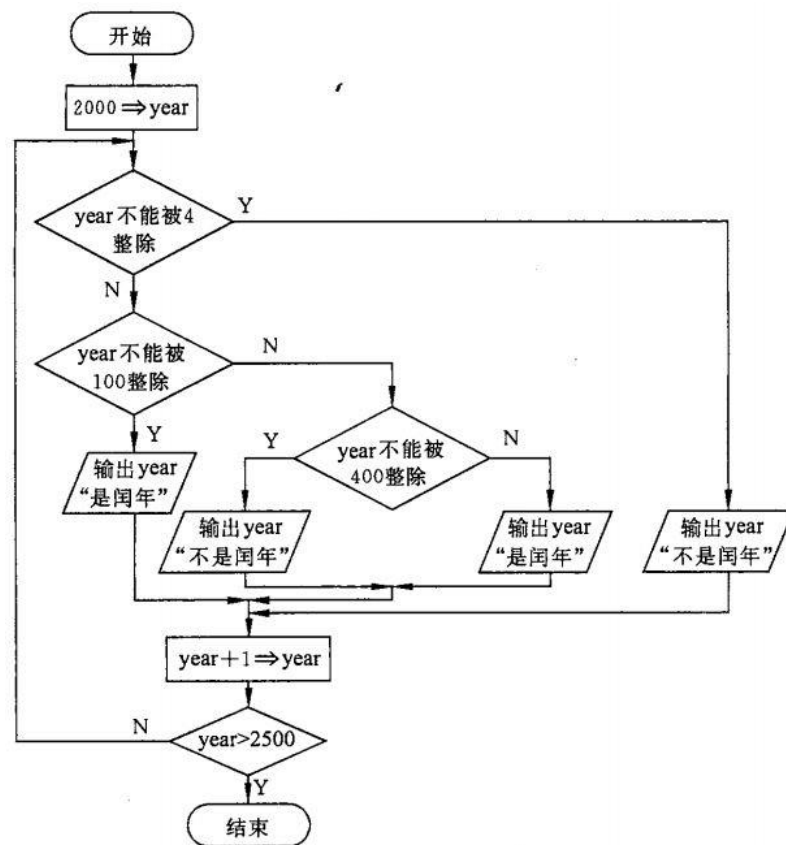
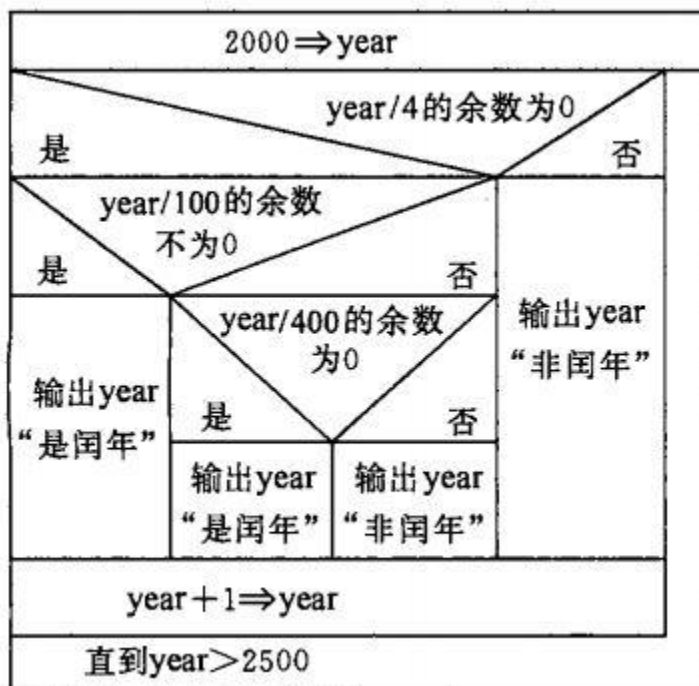
N-S流程图

- 例2.12 将例2.2的算法用N-S图表示。输出50名学生中成绩高于80分者的学号和成绩



N-S流程图

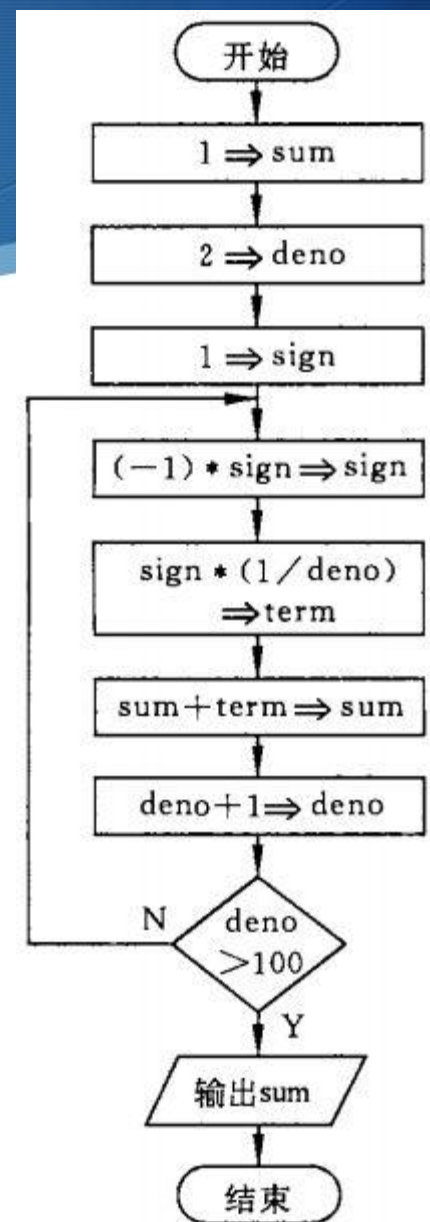
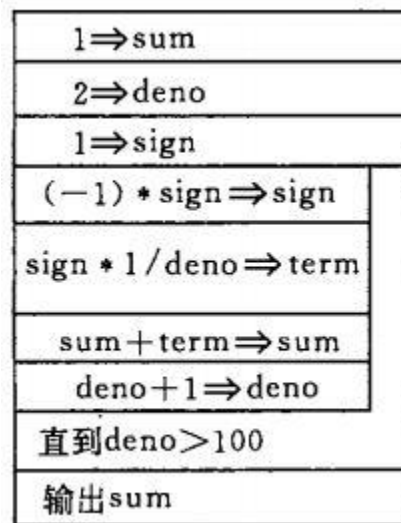
例2.13 将例2.3判定闰年的算法用N-S图表示



N-S流程图

💧 例2.14 将例2.4的算法用N-S图表示。

💧 求 $1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{1}{99} - \frac{1}{100}$

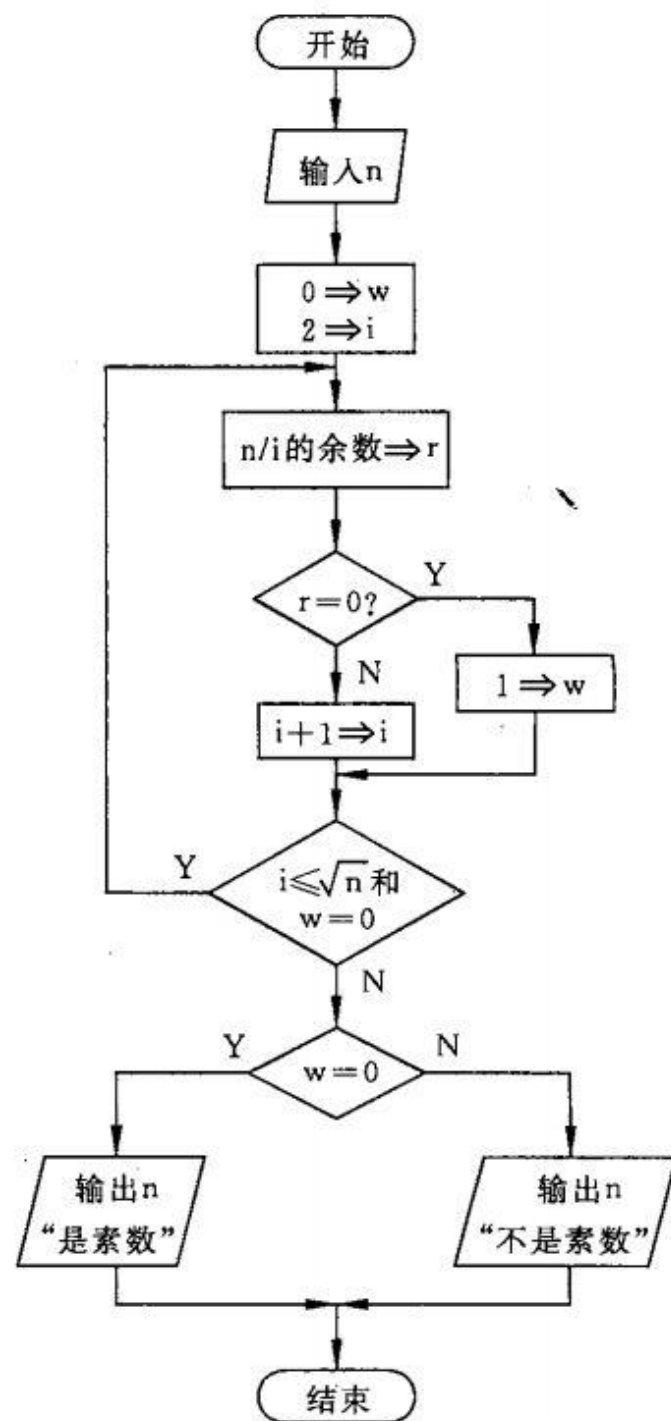
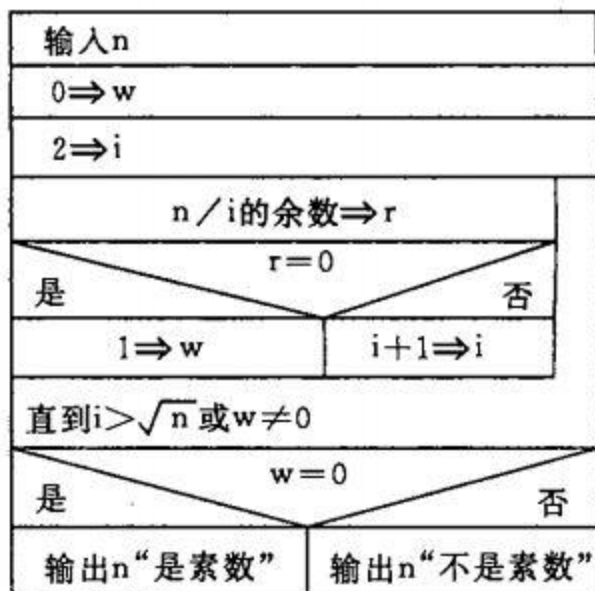


N-S流程图

- 一个结构化的算法是由一些基本结构顺序组成的
- 如果一个算法不能分解为若干个基本结构，则它必然不是一个结构化的算法

N-S流程图

- 例2.15 将例2.5判别素数的算法用N-S流程图表示



伪代码表示算法

- 伪代码是用介于自然语言和计算机语言之间的文字和符号来描述算法
 - 每一行(或几行)表示一个基本操作
 - 可以用英文，也可以中英文混用

- 例2.16 求5!

```
begin
  1 → t
  2 → I
  while i ≤ 5
  {
    t*i → t
    i+1 → i
  }
  print t
end
```

伪代码表示算法

求 $1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{1}{99} - \frac{1}{100}$

```
begin
  1 → sum
  2 → deno
  1 → sign
  while deno ≤ 100
  {
    (-1)*sign → sign
    sign*1/deno → term
    sum + term → sum
    deno + 1 → deno
  }
  print sum
end
```

结构化程序设计

- 结构化程序：用计算机语言表示的结构化算法
- 结构化程序设计方法的基本思路：把一个复杂问题的求解过程分阶段进行
 - 自顶向下
 - 逐步细化
 - 模块化设计
 - 结构化编码

结构化程序设计

◆ 设计方法

- ◆ 自顶向下，逐步细化：逐步分解各个实现步骤，直到可以完整地将各功能步骤加以全面表达
- ◆ 自上而下，逐步积累：依照客观要求，逐步完成相关实现步骤

结构化程序设计

- **结构化编码**：根据已经细化的算法正确地写出计算机程序

