# 第 4 讲

# 结构化程序设计

# 主要内容

4.1 结构化程序设计方法

4.2 顺序结构程序设计

4.3 选择结构程序设计

4.4 循环结构程序设计

# 4.1 结构化程序设计方法

- 结构化方法是一种传统的程序设计方法，组成：
  - 结构化分析（Structured Analysis，SA）
  - 结构化设计（Structured Design，SD）
  - 结构化编码（Structured programming，SP）
- 基本思想：“分而治之”

自 顶 向 下　逐 步 求 精

模块化设计　结构化编码

- 结构化程序设计的概念最早由狄克斯特拉（E. W. Dijikstra）于1965年提出，他获得1972年度图灵奖
- 1972年，狄克斯特拉与英国科学家、1980年图灵奖获得者霍尔（C. A. Hoare）合著了《结构化程序设计》（Structured Programming, Academic Press），系统阐述了结构化程序设计的思想，并提出了著名的论断：“Program testing can be used to show the presence of bugs, but never to show their absence!”（程序测试只能用来证明有错，绝不能证明无错 ）

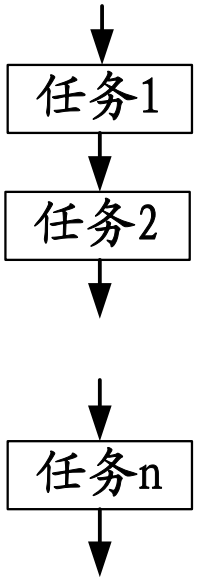
- 此外，狄克斯特拉还编写初步了大量关于程序设计的论著和论文
  - **Algol to程序设计入门**
  - **程序设计的训练方法**
  - **程序设计的教学就是思维方法的教学**
  - **程序设计方法**
  - **程序与证明的形式开发**
  - **…**

## 结构化编码

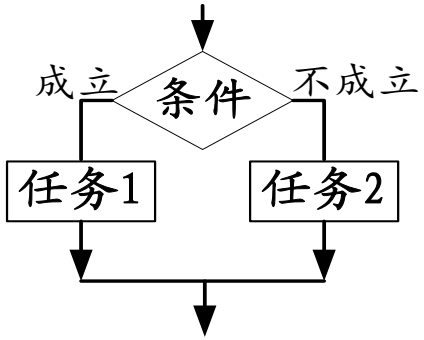
- 就是借助于具体的程序设计语言，编写程序代码，以实现预定的功能
- 结构化编码主张使用顺序、选择和循环三种基本结构及其嵌套来构造复杂层次的结构化程序，杜绝或严格控制goto语句的使用
- 结构化程序以控制结构为单位，一个入口，一个出口，程序可读性好，容易理解

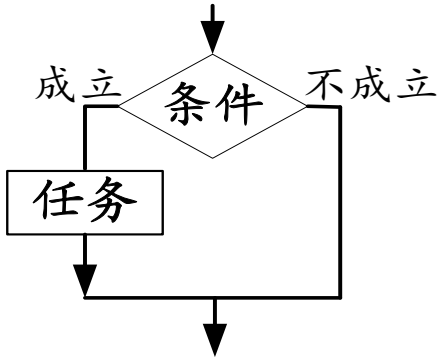
# 结构化程序的基本控制结构

- 顺序结构
  - ✓ 一种线性的、有序的结构。使计算机按从上到下的顺序依次执行各语句，直至结束
  - ✓ 通常，复杂问题无法只用一条语句就能表达清楚，需要进行任务分解，各个小任务分别用不同的语句来表达，从而构成顺序结构

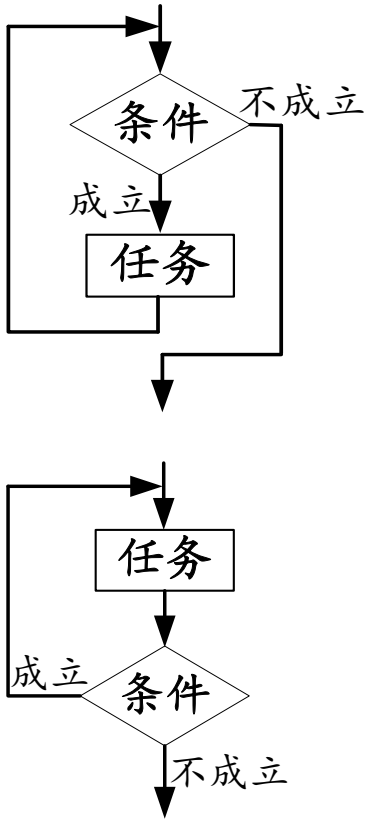


- 选择结构
  - ✓ 根据条件成立与否有选择地执行某个任务
  - ✓ 可以是二选一，如：开灯与否？开车还是坐校车上班…
  - ✓ 或多选一，如：电风扇的不同档位、不同购买量时的折扣率…



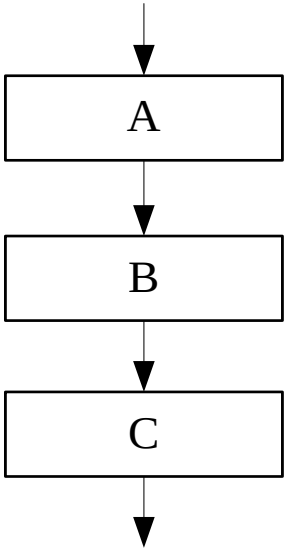
- 循环结构
  - ✓ 控制一个任务重复执行多次，直到满足一定的条件为止
  - ✓ 分为“当型”循环和“直到型”循环
  - ✓ 如：累加、累乘、针对全体学生重复相同的处理…



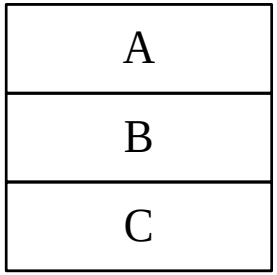
# 4.2 顺序结构程序设计

- **顺序结构**（Sequence Structure）——最简单的控制结构，也是程序设计中使用最多的结构
- 执行特点：所有语句按照书写顺序自上而下逐条执行，每条语句被执行一次。即流程是固定的，不能跳转，不能重复执行

## 流程图




## N-S图



- 顺序结构是其他控制结构的设计基础
- **顺序结构的主体**：包括一系列语句、变量等的定义及声明部分
  - ✓ 如果是复合语句，则用“{}”括起来，如：函数的函数体、选择结构的分支部分、循环结构的循环体
  - ✓ 顺序结构中常用的语句包括赋值语句、表达式语句、输入/出语句等

- 顺序结构的应用
  - ✓求值计算
  - ✓输出文本信息
  - ✓输出图案
- 顺序结构的特点
  - ✓逻辑简单，结构清晰
  - ✓任务复杂时算法较繁琐

- 顺序结构的程序设计

（1）设计数据结构：数据的组织与处理形式较为简单，基于需求分析需要处理的数据对象，并根据其值选定数据类型

（2）设计解题算法：给要处理的数据对象赋初值，然后设计任务的详细解题步骤，描述：输入各对象的初始值（I）、基于对象进行计算和处理数据（P）、输出结果（O）

**例：** 清明巡园，共坐八船，大船满六，满四小船，三十八童，满船坐观。请问客家，大小几船？

- 分析问题
  - ✓ 已知：大船乘坐6人、小船乘坐4人，共8只船，坐满了38人
  - ✓ 求解：大船需要多少只？小船需要多少只？
  - ✓ 条件：大船数*6+小船数*4 = 总人数

- 设计算法
  - ✓ 首先设计数据结构：处理的对象包括“大船”、“小船”、“船只数”以及“总人数”，共4个对象。指定4个对象的名称及相应数据类型：
    - 大船：$x$，整型
    - 小船：$y$，整型
    - 船只数：$p$，整型
    - 乘船人数：$q$，整型

- 设计算法
  - ✓设计解题方法和步骤
    - 解题方法

      根据已知条件：$x+y=p$，$6x+4y=q$

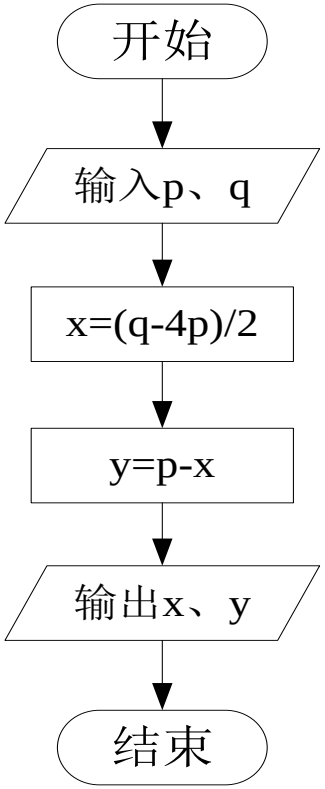
      推导计算公式：$x=(q-4p)/2$ → $y=p-x$
    - 解题步骤

      **Step 1:** 为对象p、q赋初值，或从键盘上输入

      **Step 2:** 计算公式的值

      **Step 3:** 输出对象x、y的计算结果

## 流程图




## N-S图

| 输入p、q的值 |
| --- |
| 计算x的值 |
| 计算y的值 |
| 输出x、y的值 |

- 编写代码
  - ✓ 将公式中：以名称出现的对象，定义为变量（x、y、p、q）；以确定值出现的对象，如4，处理为常量
  - ✓ 输入I：p、q
  - ✓ 输出O：x、y
  - ✓ 处理P：根据输入值，计算大小船只数量

```
#include"stdio.h"
int main( )
{
  int x, y, p, q;
  printf(" Please input the total boats (p) and People (q):");
  scanf(“%d%d”, &p, &q);      // 输入船数和人数
  x=(q-4*p)/2;                // 大船数
  y=p-x;                      // 小船数
  printf(" The number of big boats is %d.\n The number of
          small boats is %d.\n", x, y);
  return 0;
}
```

- 运行求解

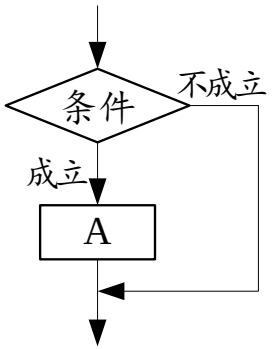
```
Please input the total boats (p) and People (q):8 38
The number of big boats is 3.
The number of small boats is 5.
```

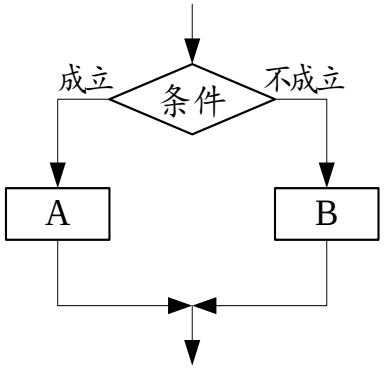
```
Please input the total boats (p) and People (q):15 76
The number of big boats is 8.
The number of small boats is 7.
```

# 4.3 选择结构程序设计

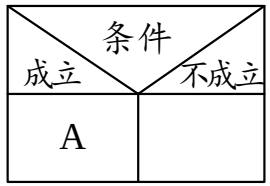
- **选择结构**（Selection Structure）——顺序结构无法完成所有类型的任务，有时需要通过选择结构来确保语句只在某种情况下执行,即在任务实现中，具体执行的操作依赖条件测试的结果
- 执行特点：根据条件的成立与否，选择执行不同的语句

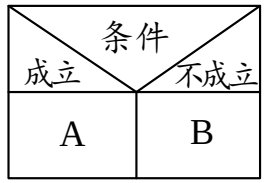
## 流程图




## N-S图

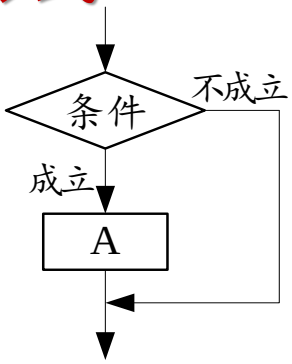


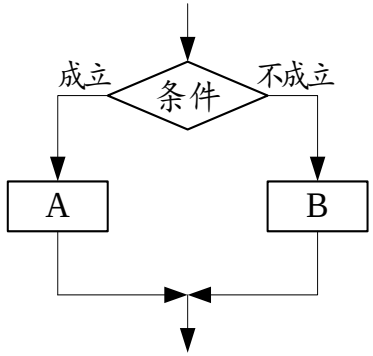



- **选择结构的主体**：包括条件的判断和一系列语句
  - ✓ 所谓“条件”就是一个可以被判断为“真”或“假”的表达式（布尔表达式，为纪念英国数学家乔治·布尔，他发明了可以把“真”与“假”这种抽象概念用于计算的布尔数学），该表达式一般由关系表达式、逻辑表达式、条件表达式或者数值表达式构成
  - ✓ 语句部分可以是单个语句，也可以是多个语句构成的语句块（复合语句）。如果是语句块，需用“{}”括起来

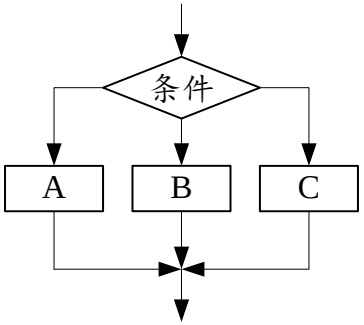
- 选择结构的形式

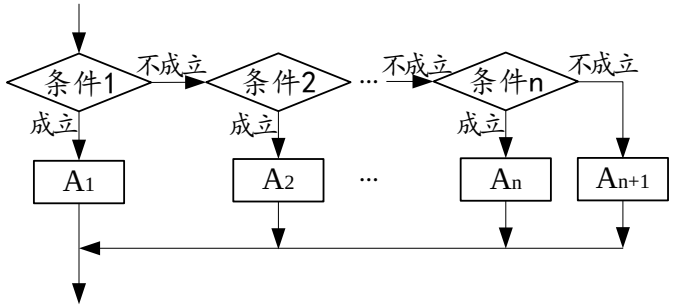
✓两路选择




✓多路选择



- 选择结构的程序设计

（1）如何区分各种情况：用关系表达式、逻辑表达式、条件表达式或数值表达式来实现

（2）如何根据不同的情况执行不同的语句：用不同的控制语句来实现

# C语言中的选择结构

- 布尔值
  - ✓C中没有布尔值
  - ✓“真”：用1表示
  - ✓“假”：用0表示

- **关系表达式**
  - ✓ 用于比较两个值的大小
  - ✓ 关系运算符：>、>=、<、<=、==、!=

    优先级：高于赋值运算符，低于算术运算符

    结合性：自左至右
  - ✓ 关系表达式：用关系运算符将两个表达式连接起来形成的表达式，值为1或0

如：“3>2”的值为 1　　“5>8”的值为 0

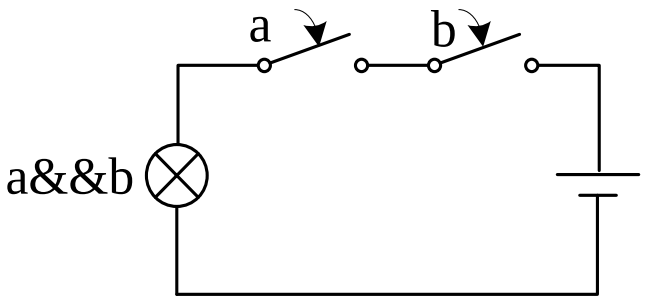
“f=8>5>-3”执行后f的值为 1

“x=8==5+8>5-8<5”执行后x的值为 0

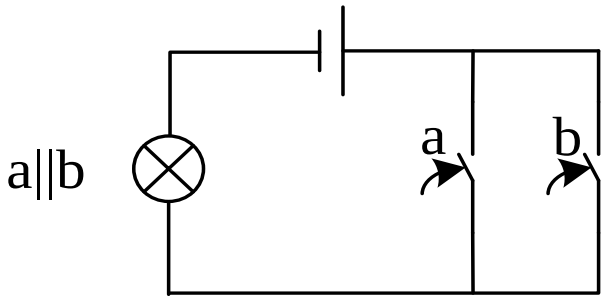
- 逻辑表达式
  - ✓ 用于复杂情况的判定，如：“5>x≥3”
  - ✓ 逻辑运算符：&&、||、！

    优先级：！、算术、关系、&& ||、赋值（高→低）

    结合性：&&和||（自左至右）、！（自右至左）
  - ✓ 逻辑表达式：用逻辑运算符将两个表达式连接起来形成的表达式，值为1或0
  - ✓ 逻辑表达式的运算规则：




| a | b | a&&b |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |




| a | b | a\|\|b |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

- 数值表达式

✓ 数值表达式的运算规则：

运算结果非0时，按“真”处理，值为1

运算结果为0时，按“假”处理，值为0

- 条件表达式
  - ✓ 条件运算符： ？ ：

    优先级：！、算术、关系、&& ||、条件、赋值（高→低）

    结合性：自右至左
  - ✓ 条件表达式：表达式1？表达式2：表达式3
  - ✓ 条件表达式的运算规则：执行表达式1；表达式1值为真执行表达式2；表达式1值为假执行表达式3
  - ✓ 应用：多用于向同一变量赋值，如：z=x>y? x:y;

- 选择结构的形式（控制语句）

✓ 两路选择结构：

```
if ( 条件 )
    语句;
```

```
if ( 条件 )
  { 复合语句; }
```

```
if ( 条件 )
    语句1;
else
    语句2;
```

```
if ( 条件 )
  { 复合语句1;}
else
  { 复合语句2;}
```

✓ 多路选择结构：

利用嵌套技术，形成在if子句或者else子句中又包含有选择结构

```
if(条件1)
   if（条件2） 语句1;
   else       语句2;
else
   if（条件3） 语句3;
   else       语句4;
```

```
if ( 条件1 )  语句1;
else  if ( 条件2 ) 语句2;
      else  if ( 条件3 ) 语句3;
                  …
                        else  语句n;
```

## 多路选择结构：switch语句

- ✓ **switch语句的执行规则：**

  首先计算表达式的值，判断其是否与case后面某个常量表达式的值相等；相等则执行该case后面的语句；全部不相等时，执行default后面的语句

- ✓ **常量表达式**：int值或char值
- ✓ **各case与default次序**：不定
- ✓ **break的使用**：将导致程序的执行立即从switch语句中退出

```
switch (表达式)
{
    case  常量表达式1: 语句1;
    case  常量表达式2: 语句2;
                ...
    case  常量表达式n: 语句n;
    default      :  语句n+1;
}
```

**例：** 某地车辆实施限行政策，规定：车牌尾号为0、1的，周一限行；尾号为2、6的周二限行；尾号为3、7的周三限行；尾号为4、8的周四限行；尾号为5、9的周五限行，周六周日不限行。请编写程序以便车主查询某天的限行情况。

- 分析问题
  - ✓已知：查询时间、车牌尾号
  - ✓求解：某天自己车辆的限行情况？
  - ✓条件：周一~周五的限行规定

- **设计算法**
  - ✓ 设计数据结构：涉及的对象：
    - 查询时间： *week*，整型
    - 车牌尾号： *num*，整型
  - ✓ 设计解题方法和步骤
    - 解题方法

      用switch语句实现多路选择结构
    - 解题步骤

      **Step 1**：输入查询时间和车牌尾号

      **Step 2**：将每周分为6种情况

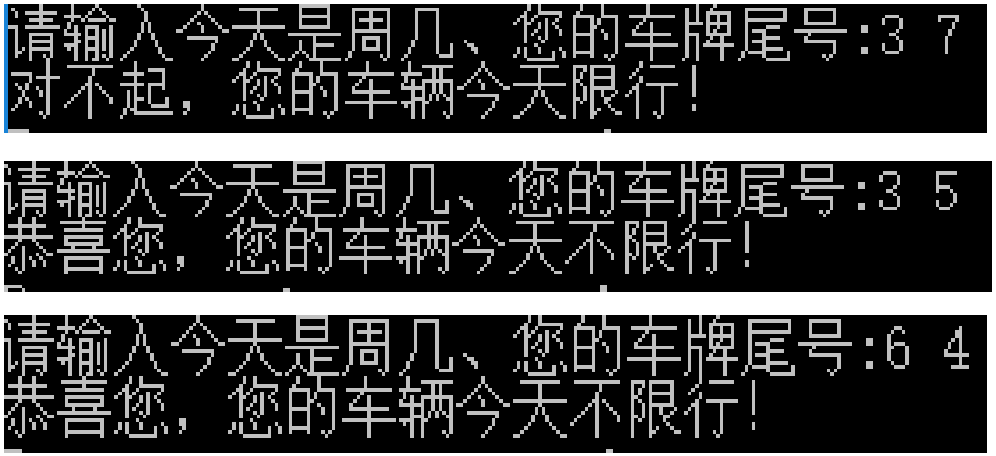
      **Step 3**：输出自己车辆这天的限行情况

- 编写代码
  - ✓ 输入I：week、num
  - ✓ 输出O：查询结果
  - ✓ 处理P：根据输入值，分别选择不同的路径执行，如果全部不匹配，执行default后面的语句

```
#include"stdio.h"
int main( )
{
  int week, num;
  printf("请输入今天是周几、您的车牌尾号:");
  scanf("%d%d", &week, &num);       // 输入时间和车牌尾号
  switch(num)
  {
    case 0:
    case 1:   if(week==1) printf("对不起，您的车辆今天限行！\n");
              else printf("恭喜您，您的车辆今天不限行！\n");
              break;
    case 2:
    case 6:   if(week==2) printf("对不起，您的车辆今天限行！\n");
              else printf("恭喜您，您的车辆今天不限行！\n");
              break;
```

```
    case 3:
    case 7:   if(week==3) printf("对不起，您的车辆今天限行！\n");
              else printf("恭喜您，您的车辆今天不限行！\n");
              break;
    case 4:
    case 8:   if(week==4) printf("对不起，您的车辆今天限行！\n");
              else printf("恭喜您，您的车辆今天不限行！\n");
              break;
    case 5:
    case 9:   if(week==5) printf("对不起，您的车辆今天限行！\n");
              else printf("恭喜您，您的车辆今天不限行！\n");
              break;
    default:  printf("您输入的车牌尾号不正确，请重新输入0-9之间的数字！\n");
  }
  return 0;
}
```

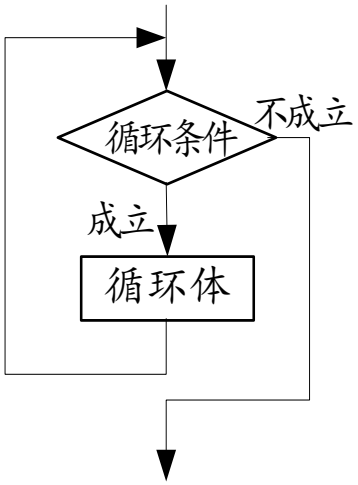
- 运行求解

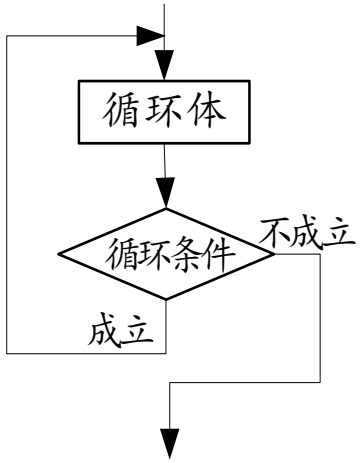

请输入今天是周几、您的车牌尾号:3 7
对不起，您的车辆今天限行!
请输入今天是周几、您的车牌尾号:3 5
恭喜您，您的车辆今天不限行!
请输入今天是周几、您的车牌尾号:6 4
恭喜您，您的车辆今天不限行!

# 4.4 循环结构程序设计

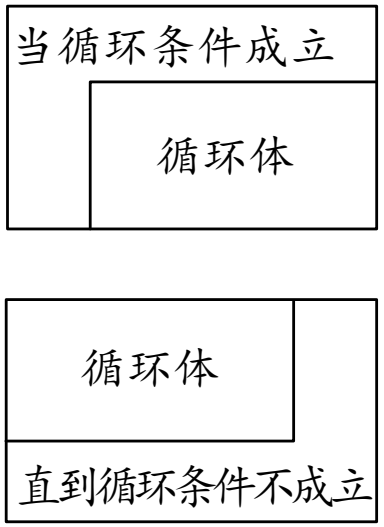
- 循环结构（Loop Structure）——在解决涉及大量数据的问题时，往往需要对每个数据多次执行相同的操作。重复执行某个操作的结构称为循环结构
- 执行特点：在一定条件（循环条件）下，让一条或一段程序（循环体）重复地执行

## 流程图




## N-S图



- 循环结构的形式
  - ✓条件控制循环：通过使用true/false条件控制循环的次数
  - ✓计数控制循环：循环特定的次数
  - ✓随机循环：无法预知循环的次数

- 循环结构的主体：
  - ✓循环初值：确定循环的次数，以及循环体中工作单元的初值等
  - ✓循环控制：对循环次数进行计数，判断循环执行的条件是否成立
  - ✓循 环 体：需要多次执行的程序的主体，可以是一条语句或复合语句

- 循环结构程序的设计：确定
  - ✓重复执行的内容是什么？
  - ✓循环控制的条件是什么？
  - ✓循环控制的方式是什么？
  - ✓是否需要通过嵌套实现？

# C语言中的循环结构

- 条件控制循环
  - ✓ while循环
  - ✓ do-while循环
  - ✓ goto语句实现循环
- 计数控制循环
  - ✓ for循环
- 随机循环

- while循环
  - ✓ 一般格式：`while(循环条件) 循环体;` 或
    `while(!(停止条件)) 循环体;`
  - ✓ 工作方式：当循环条件非0时执行循环体
  - ✓ 循环/停止条件：同于选择结构中的选择条件

实例：求 10！

```
result=1;
i=1;
while(i<=10)
{
   result *= i;
   i++;
}
```

```
result=1;
i=1;
while(!(i>10))
{
   result *= i;
   i++;
}
```

- do-while循环
  - ✓ 一般格式：`do {循环体} while(循环条件);` 或
    `do {循环体} while (!(停止条件));`
  - ✓ 工作方式：先执行循环体，再判断循环条件。如果循环条件表达式的值为非0继续循环，直到为0。循环体至少要执行一次
  - ✓ 循环条件：同于前面

实例：求 10!

```
result=1;
i=1;
do{
   result *= i;
   i++;
}while(i<=10);
```

```
result=1;
i=1;
do{
   result *= i;
   i++;
}while(!(i>10));
```

- 用goto语句实现循环
  - ✓ goto语句：无条件转向语句
  - ✓ 一般格式：

```
goto 标号;
…
标号：语句;
```

  - ✓ 工作方式：执行goto语句后，流程跳转到标号处并执行其后面的语句
  - ✓ 实例：求 10!

```
#include<stdio.h>
void main()
{
  int result=1, i=1;
ABC:
  if(i<=10)
  {
    result *= i;
    i++;
    goto ABC;
  }
 printf("10!=%d",result);
}
```

```
10!=3628800
```

- for循环
  - ✓ 一般格式：

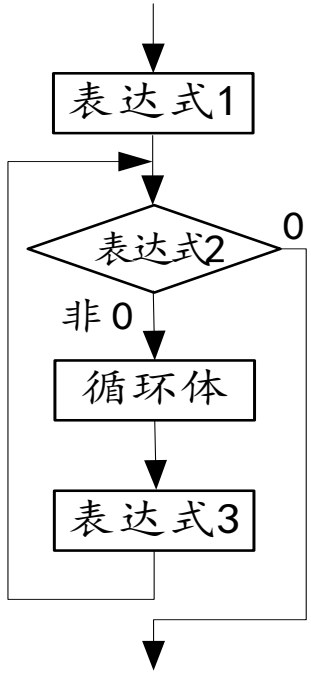
    for(表达式1 ;表达式2 ;表达式3) 循环体;

  - ✓ 工作方式：

    1)求解表达式1（只执行一次）；

    2)求解表达式2（可能执行若干次），值为非0则执行循环体，为0则结束循环；

    3)如表达式2的值为非0，执行完循环体后，求解表达式3（可能执行若干次），再重新求解表达式2 …



- 表达式1：为循环控制变量赋初值，或其他操作
- 表达式2：循环条件
- 表达式3：调整循环控制变量的值

✓说明：

- 表达式1、3可以是逗号表达式，如

  for(sum=0, i=1, j=100; i<j; i++, j--)  sum+=i+j;

- 表达式1~3均可以省略，但“;”不能省略，即可以是for(;i<10;i++)…，for(i=0; ; i++)…, for(; ; i++)…，还可以是for(; ; )…。但要注意对循环控制变量及循环条件进行调整，否则：不循环、死循环

- 随机循环
  - ✓ 随机循环：事先无法预知循环何时会停止，需要根据一定的条件决定是否继续进行循环
  - ✓ 实例

```
scanf("%d", &n);
while(n>0)
{
   if(n%3==0) printf("这个数能被3整除!\n");
   scanf("%d", &n);
 }
```

- 循环的提前终止
  - ✓ **continue**：提前结束本次循环，接着进行下一次循环的判断，如满足循环条件，继续循环
  - ✓ **break**：提前结束整个循环，接下来执行循环之后的语句

（注：break语句只能用于switch语句和循环结构中）

```
sum=0;
for(i=0;i<10;i++)
{
   scanf("%d", &x);
   if(x<0)  break;
   sum+=x;
}
printf("%d\n",sum);
```

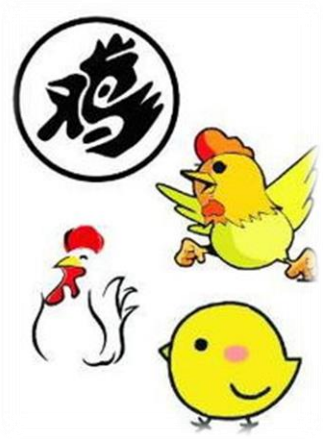
```
3 7 4 6 5 -1 7 8 9 10
25
```

```
sum=0;
for(i=0;i<10;i++)
{
   scanf("%d", &x);
   if(x<0)  continue;
   sum+=x;
}
printf("%d\n",sum);
```

```
3 7 4 6 5 -1 7 8 9 10
59
```

- 循环的嵌套
  - ✓ for循环与for循环嵌套
  - ✓ for循环与while循环嵌套
  - ✓ for循环与do-while循环嵌套
  - ✓ while循环与while循环嵌套
  - ✓ while循环与do-while循环嵌套
  - ✓ do-while循环与do-while循环嵌套

**例：百钱买百鸡**“鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一。百钱买百鸡，问鸡翁、母、雏各几何？”



- 分析问题
  - ✓已知：公鸡一只5元，母鸡一只3元，小鸡三只1元
  - ✓求解：所买公鸡数？母鸡数？小鸡数？
  - ✓条件：100元钱买100只鸡

- 设计算法
  - ✓ 设计数据结构：涉及的对象：
    - 公鸡：$x$，整型
    - 母鸡：$y$，整型
    - 小鸡：$z$，整型
  - ✓ 设计解题方法和步骤

$5x+3y+z/3=100$ //百钱

$x+y+z=100$ //百鸡

// 不定方程组，用循环结构+枚举技术实现

- 编写代码、运行求解

```
#include<stdio.h>
void main()
{
  int x, y, z;
  for(x=1; x<=20; x++)          // 100元钱最多买20只公鸡
    for(y=1; y<=33; y++)        // 100元钱最多买33只母鸡
      for(z=3; z<=99; z++)      // 按要求最多可买99只小鸡
        if(x+y+z==100&&5*x+3*y+z/3==100)
          printf("公鸡=%d, 母鸡=%d, 小鸡=%d\n", x, y, z);
}
```

```
公鸡=3,  母鸡=20,  小鸡=77
公鸡=4,  母鸡=18,  小鸡=78
公鸡=7,  母鸡=13,  小鸡=80
公鸡=8,  母鸡=11,  小鸡=81
公鸡=11,  母鸡=6,  小鸡=83
公鸡=12,  母鸡=4,  小鸡=84
```

```
#include<stdio.h>
void main()
{
  int x, y, z;
  for(x=1; x<=20; x++)          // 100元钱最多买20只公鸡
    for(y=1; y<=33; y++)        // 100元钱最多买33只母鸡
      for(z=3; z<=99; z++)      //按要求最多可买99只小鸡
        if(x+y+z==100&&5*x+3*y+z/3==100&&z%3==0)
          printf("公鸡=%d, 母鸡=%d, 小鸡=%d\n", x, y, z);
}
```

```
公鸡=4,  母鸡=18,  小鸡=78
公鸡=8,  母鸡=11,  小鸡=81
公鸡=12,  母鸡=4,  小鸡=84
```

```
#include<stdio.h>
void main()
{
  int x, y, z;
  for(x=1; x<=20; x++)
    for(y=1; y<=33; y++)
    {
      z=100-x-y;
      if(5*x+3*y+z/3==100&&z%3==0)
          printf("公鸡=%d, 母鸡=%d, 小鸡=%d\n", x, y, z);
    }
}
```

```
公鸡=4,  母鸡=18,  小鸡=78
公鸡=8,  母鸡=11,  小鸡=81
公鸡=12,  母鸡=4,  小鸡=84
```

**思考：** 从键盘输入一个三位正整数N，对于从最小的三位数到该数之间的所有数值，如果其各位数字阶乘之和是一个个位数，则输出该数。请编写程序实现。

```
#include<stdio.h>
void main( )
{
  int N, n, sum, pol, t, k, i;
  scanf("%d", &N);
  for(i=100; i<=N; i++)
  {
     sum=0;     n=i;
     while(n!=0)
     {
        t=n%10; //取各位上的数字
        n=n/10;

        k=1;
        pol=1;
        while(k<=t)  //求阶乘
        {
          pol=pol*k;
          k++;
        }
        sum+=pol;
     }
     if(sum<10)
        printf("%5d", i);
     else continue;
  }
  printf("\n", i);
}
```

```
999
100  101  102  103  110  111  112  113  120  121  122  123  130  131  132  200  201  202  203  210  211  212  213  220
221  222  230  231  300  301  302  310  311  312  320  321
```

# Homework

- 实践

  实验2 结构化程序设计

  要求：提交纸质实验报告

- 作业

  《教材》：P107-109　9、10、12

  P137-138　7、10、14