计算机程序设计基础

Programming Fundamentals

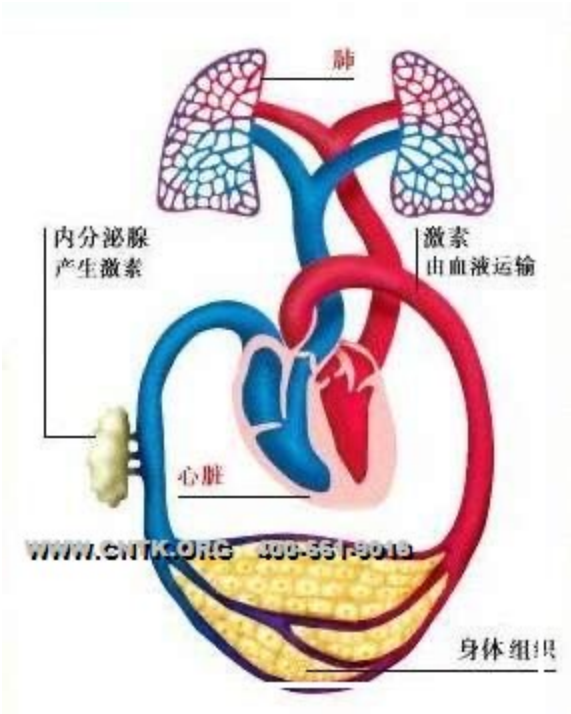
韩文弢

清华大学计算机系

第三章 循环控制

顺序

选择



循环

Why循环结构？

程序员失去循环语句，

世界将会怎么样？

- 计算机的执行速度
  - 100,000,000行代码/秒
- 程序员的编程速度
  - 300行可用的代码/每天
- 编写一个运行时间为1秒的程序，需要多少天？

只要功夫深，铁杵磨成针！

```
int   main( )
{
      磨针;
      磨针;
      磨针;
      ……
      磨针;
      磨针;
}
```

顺序结构

```
int   main( )
{
      while(针未磨好)
      {
            磨针;
      }
}
```

循环结构

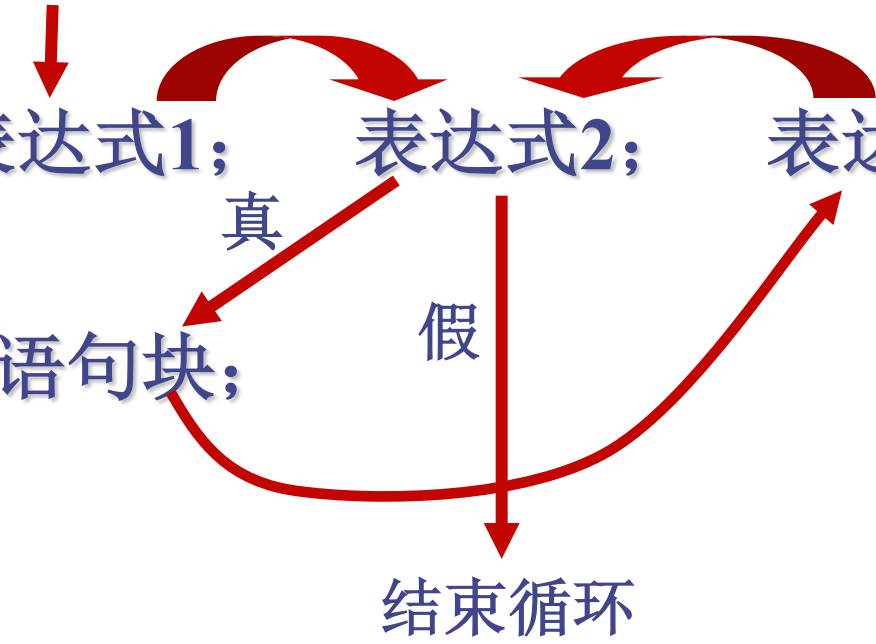
本章结构

1. for语句
2. while语句
3. do-while语句
4. break和continue语句
5. 程序举例

3.1 for语句

一般形式：

```
for (表达式1; 表达式2; 表达式3)
{
    语句块;
}
```



各部分最少被执行几次？

```
for (init; condition; update)
{
    statements;
}
```

- 事前须初始化、事后要更新
- 先条件测试再执行
- 计数驱动

```
/* Frog lifetime */
int days;
for(days = 155; days > 0; days--)
{
     work_all_day();
     sleep_all_night();
}
die_quietly();
```

分析运行结果1

```
sum = 0;
for(i = 1; i <= 10; i = i + 1);
{
    sum = sum + i ;
}
cout << sum << endl;
```

11

分析运行结果2

高斯的难题

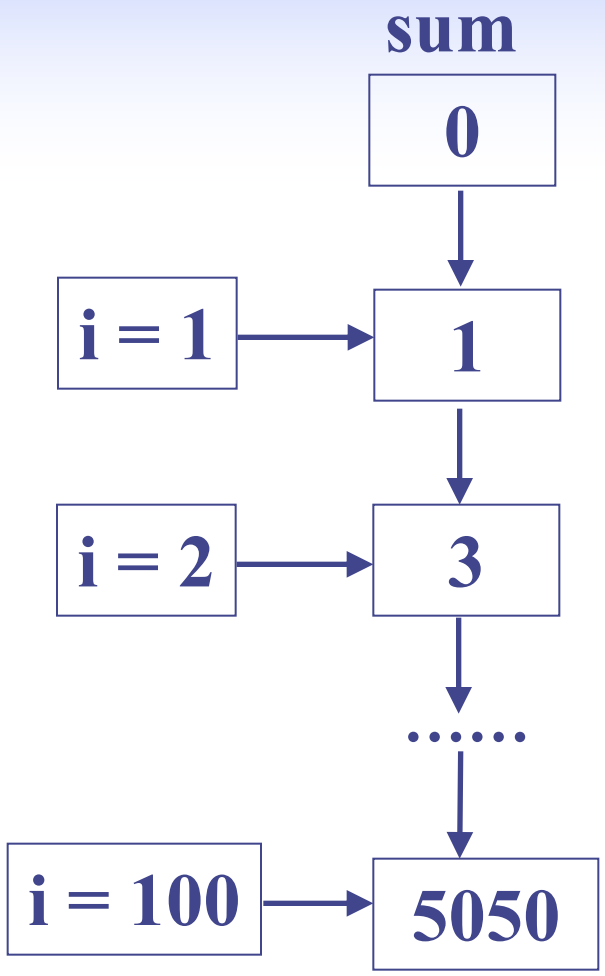
德国数学家高斯，在上小学的时候，老师出了一道难题，计算1+2+3+……+100，高斯很快就在自己的小石板上写出了答案5050，老师非常惊讶，高斯怎么算得这么快呢？原来，高斯不是一个数一个数按部就班地加起出来的，而是发现这些数字有一个规律，一头一尾依次两个数相加，它们的和都是一样的：1+100=101，2+99=101，一直到50+51=101，一共是50个101，所以，他很快就把答案算出来了。

基本思路：

把问题抽象为一种统一的形式，然后采用循环语句来重复地计算。

用一个变量sum来保存总和，对于1、2、3、…、100 中的每一个整数 i，依次把它加入到sum当中，即 sum ＝ sum + i。

```
#include <iostream>
using namespace std;
int main( )
{
      int  i, sum;
      sum = 0;
      for ( i = 1; i <= 100; i++)
      {
            sum = sum + i;
       }
      cout << "sum = " << sum;
}
```

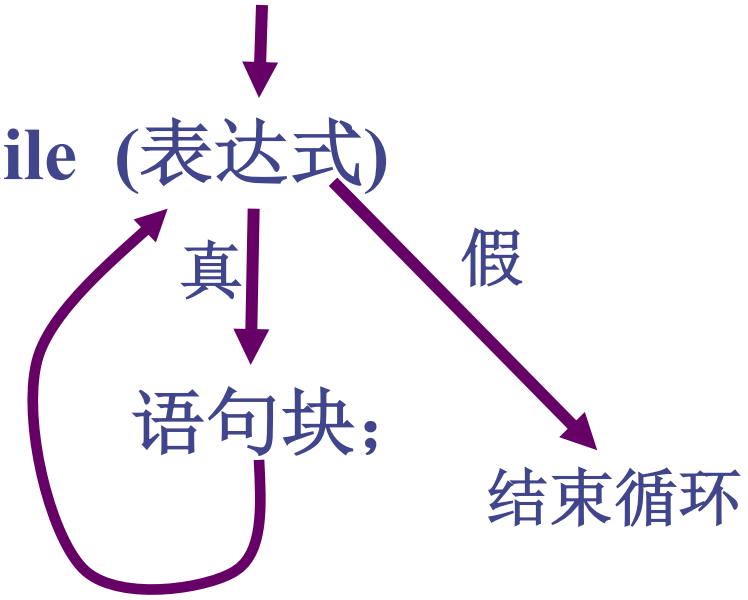


累加编程模式

3.2 while语句

一般形式：

```
while (表达式)
{
    语句块;
}
```



```
while( condition )
{
    statements;
}
```

- 先条件测试再执行
- 事件驱动（比较计数驱动）

```
/* Frog Feeding  */
while (see_fly())
{
    flick_tongue();
    clamp_mouth();

}
```

脆弱的输入方式

```
char  choice;


cout << "你是否想借书? (y/n)";
cin >> choice;
……
```

如果用户输入的是y/n以外的字符呢？

健壮的输入方式

```
char  choice;

cout << "你是否想借书? (y/n)";
cin >> choice;
while(choice  !=  'y'   &&   choice  !=  'n')
{
        cout <<  "你是否想借书? (y/n)";
        cin >> choice;
}
……
```

分析下列程序的输出结果

```
int x = 1536, y = 0;
while(x != 0)
{
    y = y * 10 + x % 10;
    x = x / 10;
}
cout << y << endl;
```

6351

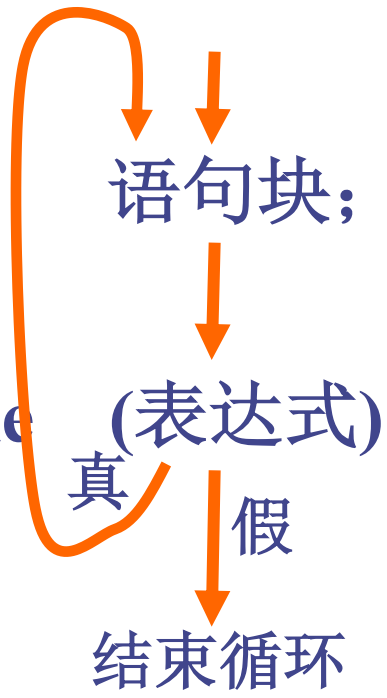
Double Your Money

```
 /* 假定你有1000元，每年投资收益为5%，
 经过多少年后，你的钱会翻番？ */
my_money = 1000.0;
n = 0;
while ( my_money < 2000.0 )
{
        my_money = my_money *1.05;
        n = n + 1;
}
cout << "我的钱将会在" << n << "年内翻番";
```

3.3 do-while语句

一般形式：

```
do
{
}
while (表达式)
```



```
do
{
    statements;
} while( condition );
```

- 先执行再条件测试
- 事件驱动

```
/* Frog Feeding  */
do
{
    chew_and_mash();
    swallow();

} while(!mouth_empty());
```

```
int password;
do {
    cout << "请输入密码：";
    cin >> password;
} while (password != PASSWORD);
```

3.4 break和continue语句

break语句的功能：

1. 用来跳出switch结构；
2. 用来从循环体内跳出循环体，即提前结束循环，接着执行循环下面的语句。

一般形式为：**break;**

```
int days;
double food, fat;
...
for(days=155; days > 0; days--)
{
    work_all_day();
    if(food+fat < 0.01) break;
    sleep_all_night();
}
die_quietly();
```

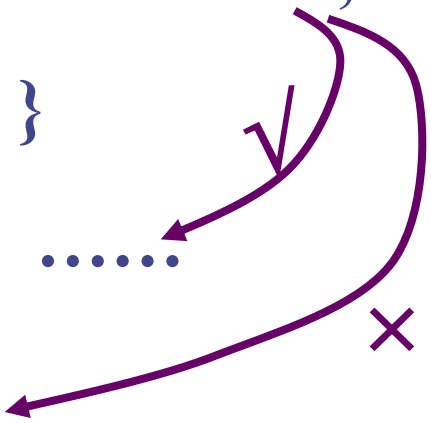
健壮的输入方式 (while again)

```
char  choice;

while( true )
{
        cout << "你是否想借书? (y/n)";
        cin >> choice;
        if ( choice == 'y'  ||  choice == 'n' )  break;
}
……
```

循环条件永真模式

```
while (…)
{
        ….
        while (…)
        {
                ……
                break;
        }
        ……
}
……
```



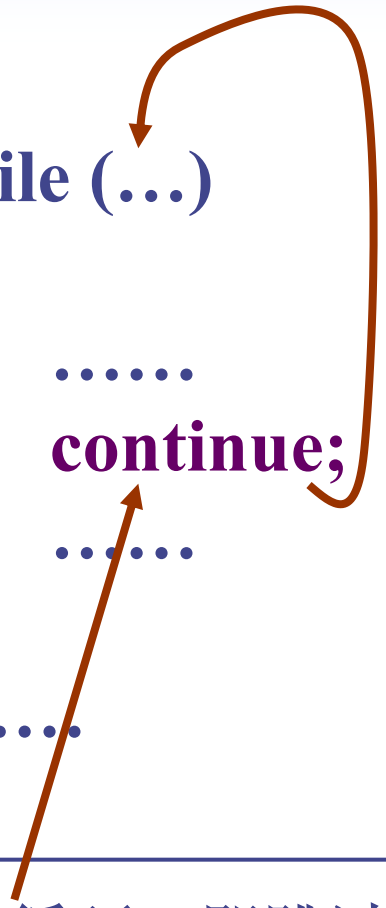
跳出最近的循环。

如何跳出多重循环

- 有几种方法：
  - 设置标志变量，多次 break
  - 将多重循环放入一个函数，用 return 返回
  - 使用 goto 语句

continue语句

```
while (…)
{
        ….
        while (…)
        {
                ……
                continue;
                ……
         }
          ……
}
```



结束本次循环，即跳过循环体中尚未执行的语句，直接回到循环条件的判别。

分析下列程序的输出结果

```
sum = 0;
for(i = 0; i < 10; ++i)
{
    if(i == 4) break;
    if(i == 2) continue;
    sum += i;
}
cout << sum << endl;
```

4

3.5 程序举例

3.5.1 实数解码

问题描述：

编码规则

- 用4个字节（整数）来描述一个实数，低28位为有效数字，高4位为指数部分；
- 如果指数部分大于8，则进行除法；
- 如果指数部分小于8，则进行乘法；

输入一个编码后的数据（十六进制整数），输出经过解码以后的实数。

样例

16进制编码数据	有效数字	指数	解码数据
9D 90 76 04	0x476909D	0x00	74879133
C7 DE 9F 11	0x19FDEC7	0x01	272544710
6A 78 02 95	0x502786A	0x09	8404797.8
4F 42 68 A6	0x668424F	0x0A	1074960.15
32 0F 00 B0	0x0000F32	0x0B	3.89

问题分析

1. 如何把28位的“有效数字”和4位的“指数部分”剥离出来；
2. 如何将一个数乘以10^N或除以10^N？

位运算

1. 按位与（&）：将两个运算量的每一个位进行逻辑与操作；
2. 按位或（|）：将两个运算量的每一个位进行逻辑或操作；
3. 按位异或（^）：相同为0，不同为1；
4. 取反(~)：对一个二进制数按位取反。

```
    0011（整数3）
&   0101（整数5）
-------------------
    0001（整数1）
```

```
    0011（整数3）
|   0101（整数5）
-------------------
    0111（整数7）
```

```
    00111001（整数57）
^   00101010（整数42）
-----------------------
    00010011（整数19）
```

```
int x, count = 0;
int num = 9999;      // 0x270F
x = num;
while(x)
{
    count ++;
    x = x & (x - 1);
}
cout << count << endl;
```

输出结果是：__8__

移位运算

1. 左移运算（<<）：左移后，低位补0，高位舍弃，如a<<2；

2. 右移运算（>>）：右移后，低位舍弃，高位补“符号位”。

问题分析（2）

1. 对一个无符号整数，如0x119FDEC7，如何把28位“有效数字”（0x19FDEC7）剥离出来？

```
value & 0x0FFFFFFF
```

2. 如何把4位“指数部分”(0x1)剥离出来？

```
value >> 28
```

3. 如何将一个数乘以10^N或除以10^N？

```
void main()
{
    unsigned int value;
    int N, i;
    double result;

    cin >> hex >> value;  //十六进制
    N = value >> 28;
    result = value & 0x0FFFFFFF;
    if(N > 8)
    {
        N = N - 8;
        for(i = 1; i <= N; i++) result /= 10;
    }
    else
    {
        for(i = 1; i <= N; i++) result *= 10;
    }
    cout << result << endl;
}
```

测试...

3.5.2 谁做的好事？

问题描述：

清华附中有四位同学中的一位做了好事，不留名，表扬信来了之后，校长问这四位是谁做的好事。

- A说：不是我。
- B说：是C。
- C说：是D。
- D说：他胡说。

已知三个人说的是真话，一个人说的是假话。现在要根据这些信息，找出做了好事的人。

注：本例来自于吴文虎老师的讲义，在此表示感谢。

问题分析

1. 一个人：某个人做了好事，不知是谁；
2. 四句话：“该人是/不是某某”，如何用程序语言描述每个人说的话？
3. 三真一假：如何描述一句话是真是假？
4. 如何找到该人？

在声明变量时，让 thisman 表示要找的人，定义他为字符型变量。

char thisman;

下面，把四个人说的四句话写成关系表达式。

让 “==” 的含义为 “是”

让 “!=” 的含义为 “不是”

A说：不是我。写成关系表达式为（thisman != ‘A’）

B说：是C。　　写成关系表达式为（thisman == ‘C’）

C说：是D。　　写成关系表达式为（thisman == ‘D’）

D说：他胡说。写成关系表达式为（thisman != ‘D’）

相应字符的ASCII码值为：

字符	‘A’	‘B’	‘C’	‘D’
ASCII码值	65	66	67	68

思路分析(1)：

如何找到该人，一定是“先假设某人是做好事者，然后到每句话中去测试看有几句是真话”。“有三句是真话就确定是该人，否则换下一人再试”。比如，先假定是A同学，让thisman=‘A’，代入到四句话中：

A说：thisman != 'A'; 'A' != 'A' 假，值为0。
B说：thisman == 'C'; 'A' == 'C' 假，值为0。
C说：thisman == 'D'; 'A' == 'D' 假，值为0。
D说：thisman != 'D'; 'A' != 'D' 真，值为1。

显然，不是'A'做的好事（3假1真）。

思路分析(2)：

再试B同学，让thisman = 'B'；代入到四句话中

A说：thisman != 'A'; 'B' != 'A' 真，值为1。
B说：thisman == 'C'; 'B' == 'C' 假，值为0。
C说：thisman == 'D'; 'B' == 'D' 假，值为0。
D说：thisman != 'D'; 'B' != 'D'真，值为1。

显然，不是'B'所为（2假2真）

思路分析(3)：

再试C同学，让thisman = 'C'；代入到四句话中

A说：thisman != 'A'; 'C' != 'A' 真，值为1。
B说：thisman == 'C'; 'C' == 'C' 真，值为1。
C说：thisman == 'D'; 'C' == 'D' 假，值为0。
D说：thisman != 'D'; 'C' != 'D'真，值为1。

显然，就是'C'做了好事（3真1假），这时，可以理出头绪，要用所谓的枚举法，一个人一个人地去试，四句话中有三句为真，该人即为所求。

从编写程序的角度看，实现枚举最好用循环结构。

```
// thisman 分别赋值为'A', 'B', 'C', 'D'
for(thisman = 'A';  thisman <= 'D';  thisman++)
{
        sum  =  (thisman  !=  'A')          // A的话是否为真
                + (thisman  ==  'C')        // B的话是否为真
                + (thisman  ==  'D')        // C的话是否为真
                + (thisman  !=  'D');       // D的话是否为真
        if (sum  ==  3)
        {
                cout << "This  man  is: " << thisman << endl;
                break;
        }
}
```

3.5.3 猜数字游戏

问题描述：

电脑随机产生一个数字不重复的四位数，由玩家来猜，每猜一次，电脑将显示形如“*A*B”的结果，A代表位置正确数字也正确，B代表数字正确但位置不正确，例如：2A2B。总共有10次机会

问题分析

1. 如何随机产生一个数字不重复的四位数？
2. 对于玩家猜测的一个四位数，如何计算相应的A和B的数量？
3. 如何实现“总共有十次机会”？

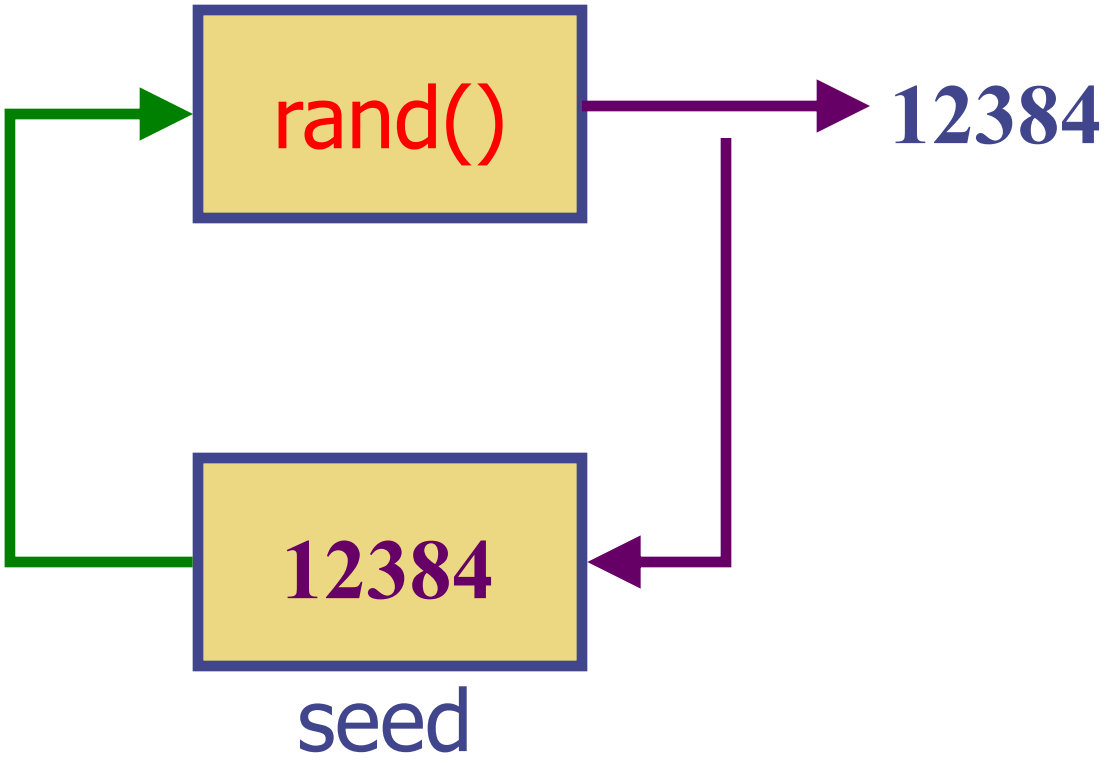
1. 随机数

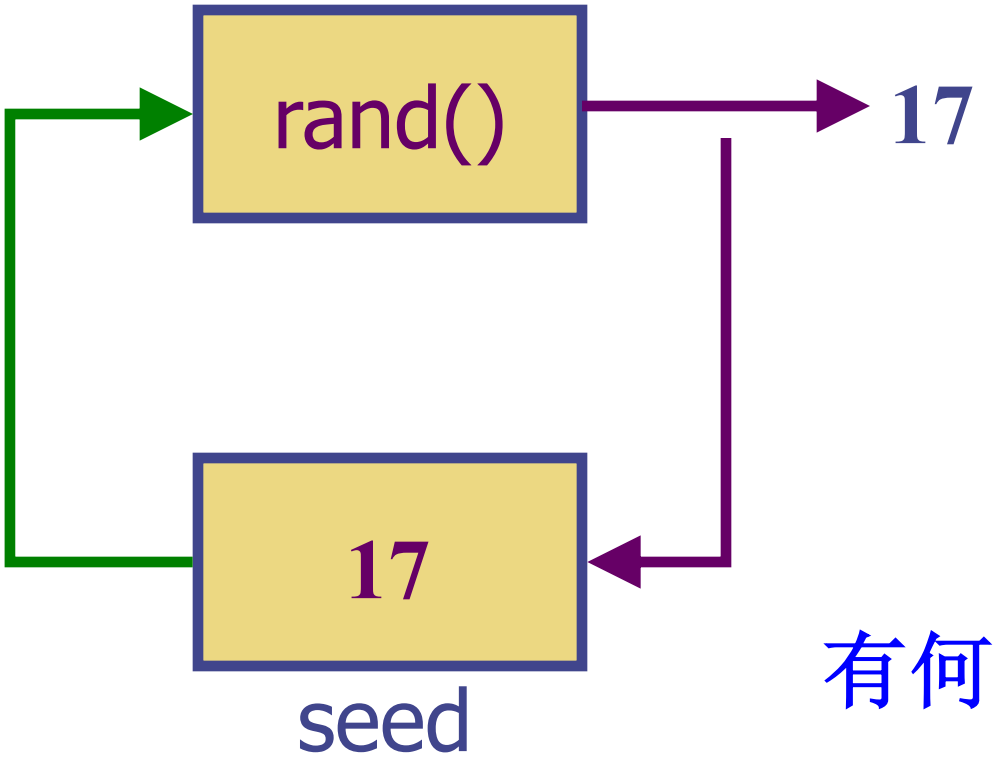
1. 随机数：随机过程，如仍硬币、掷骰子，不确定事件；
2. 计算机：代码的执行是确定的；
3. 伪随机：用一个确定的过程来生成，从统计意义上类似随机数，且难以预测。

在C语言中生成随机数

1、要产生随机数需要在程序开头加入头文件
```
#include <cstdlib>
```

2、rand()是产生随机数的函数，
```
int rand(void);
```
它可生成0至RAND_MAX的整数，RAND_MAX = 0x7FFF = 32767


rand()
12384
12384
seed



有何问题？

随机数序列重复

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main( )
{
    int i, x;
    for(i = 0; i < 10; i++)
    {
        x = rand( );
        cout << x << endl;
    }
}
```

第一次	第二次	第三次
41	41	41
18467	18467	18467
6334	6334	6334
26500	26500	26500
19169	19169	19169
15724	15724	15724
11478	11478	11478
29358	29358	29358
26962	26962	26962
24464	24464	24464

如何改进？

1、需要修改seed的初始值

2、通过srand()函数来实现，

```
void srand(int seed);
```

3、可以用当前时间来作为初始值

```
srand((unsigned)time(NULL));
```

当前时间作为种子

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
void main( )
{
    int i, x;
    srand((unsigned)time( NULL ));
    for(i = 0; i < 10; i++)
    {
        x = rand( );
        cout << x << endl;
    }
}
```

第一次	第二次	第三次
28946	29393	31679
24006	21977	9222
12469	2260	22572
7994	27868	29436
18018	22091	565
16047	29721	25920
31399	31821	15080
19861	12040	27332
8191	9286	296
26646	3561	25290

如何修改取值范围？

1、rand生成的随机数位于[0,RAND_MAX]，如何生成任意范围[a, b]内的随机数？

2、
```
d = (double) rand() / RAND_MAX;
a + d * (b-a);
```

如何随机生成一个数字不重复的四位数？

1、从10个数字中选4个构成排列；

2、逐一生成每一位，且与前面的各位不同；

3、生成一个四位的随机数，再过滤有重复者；

4、.....

<random> 随机数生成库

- 从 C++11 开始，加入了 <random>
- 提供随机数引擎和各种不同的分布

```
#include <random>
#include <iostream>

int main() {
    std::random_device rd;  // 获得随机数种子
    std::mt19937 gen(rd()); // 随机数引擎（发生器）
    std::uniform_int_distribution<> distrib(1, 6);  // 均一分布

    for (int n=0; n<10; ++n)
        std::cout << distrib(gen) << ' ';
    std::cout << '\n';
}
```

2. 计算A和B

已知目标数和猜测数，如何计算相应的A和B的数量？

- 假设目标数为T1 T2 T3 T4，
  猜测数为G1 G2 G3 G4；
- 一种方法：T_i和G_i相比，相同则A加1；不相同则把T_i和剩余三个猜测位相比，若有相同者，则B加1。

3. 总共有十次机会

如何实现？

累加编程模式！

算法思路

1. 随机产生一个数字不重复的四位数，将它拆分为四位数字T1 T2 T3 T4；
2. 让用户输入一个数字不重复的四位数，将它拆分为四位数字G1 G2 G3 G4；
3. 计算相应的A和B的数量；
4. 如果结果为4A0B，则成功；否则，将猜测的次数加1，如果不超过10，则转第2步；否则猜测失败。

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int main()
{
    int Target, T1, T2, T3, T4;
    int Guess, G1, G2, G3, G4;
    int nA, nB, NumGuess;
    double d;
```

```
// 随机产生一个数字不重复的四位数
srand((unsigned)time( NULL ));
while(1)
{
    d = (double) rand() / RAND_MAX;
    Target = 1000 + (d * 8999); //1000~9999

    // 将它拆分为四位数字
    T1 = Target / 1000;
    T2 = (Target / 100) % 10;
    T3 = (Target / 10) % 10;
    T4 = Target % 10;
    if((T1 != T2) && (T1 != T3) && (T1 != T4) &&
       (T2 != T3) && (T2 != T4) && (T3 != T4))
       break;
}
```

```
NumGuess = 0;
while(1)
{
    cout << "输入一个不重复的四位数：";
    cin >> Guess;

    // 将它拆分为四位数字
    G1 = Guess / 1000;
    G2 = (Guess / 100) % 10;
    G3 = (Guess / 10) % 10;
    G4 = Guess % 10;

    // 计算相应的A和B的数量
    nA = 0;  nB = 0;
    if(T1 == G1) nA++;
    else if((T1==G2)||(T1==G3)||(T1==G4)) nB++;
    if(T2 == G2) nA++;
    else if((T2==G1)||(T2==G3)||(T2==G4)) nB++;
    if(T3 == G3) nA++;
    else if((T3==G1)||(T3==G2)||(T3==G4)) nB++;
    if(T4 == G4) nA++;
    else if((T4==G1)||(T4==G2)||(T4==G3)) nB++;
```

```
        // 判断结果
        if(nA == 4 && nB == 0)
        {
            cout << "恭喜你猜对了！答案是：" << Target;
            break;
        }
        else
        {
            NumGuess++;
            if(NumGuess >= 10)
            {
                cout << "十次都没猜中，你玩完了！\n";
                cout << "答案是：" << Target;
                break;
            }
            else
                cout << nA << "A " << nB << "B";
        }
    }
    return 0;
}
```

程序的时间复杂度

- 引入循环后，程序的执行时间就可以很长了。
- 如何估计程序的运行时间？
- 根据问题规模 n
- 用 n 的各种函数来表示
- O(1) O(n) $O(n^2)$ $O(2^n)$
- 假设循环次数都是 n，操作都是 O(1) 的
  - 并列的两个一重循环：O(n)+O(n)=O(n)
  - 两重循环： $O(n^2)$

本讲小结

- 循环结构
- 三种不同的循环语句
- 改变循环执行流程的办法
- 位运算
- 随机数生成