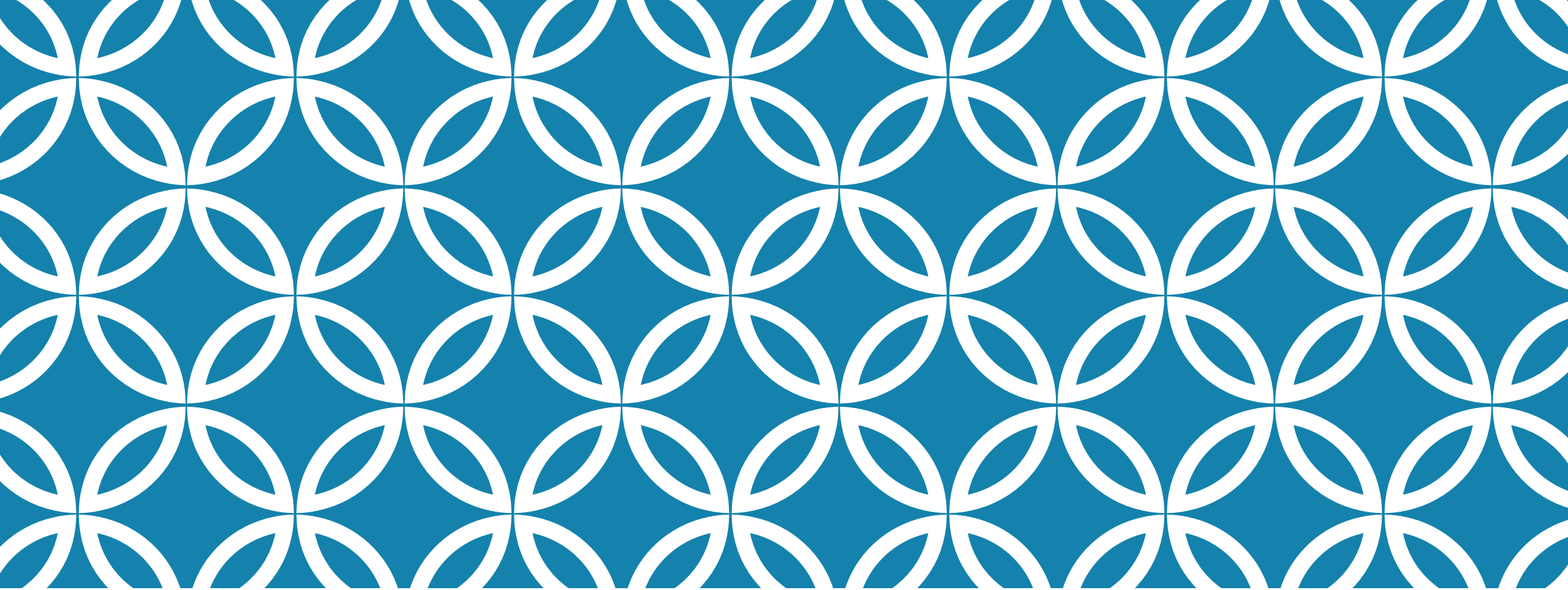


C++ 直播答疑课

潘祖江
清华大学计算机系

知识点综述-函数的声明和使用

- 函数定义的语法形式：

  类型说明符　函数名（形式参数表）

  {　语句序列}

  调用函数之前先要在主调函数中声明函数原型，形式如下：

  类型说明符　被调函数名　（含类型说明的形参表）

  声明了函数原型之后，便可以按如下形式调用子函数：

  函数名（实参列表）

- 内联函数不是在调用时发生控制转移，而是在编译时将函数体嵌入在每一个调用语句处。这样就节省了参数传递、控制转移等开销。内联函数在定义时使用关键字 inline，语法形式如下：

  inline　类型说明符　被调函数名　（含类型说明的形参表）

函数的调用过程及内联函数的优势?

函数调用的过程：①程序存储当前现场及返回地址；②进行参数传递；③执行被调用函数中函数体的语句；④返回主调函数并将函数返回值传回主调函数。

使用内联函数在编译时将所调用函数的代码直接嵌入到主调函数中，可以节省运行时间。一般只将规模很小而使用频繁的函数声明为内联函数。

内联函数体内一般不能有循环语句和switch语句；内联函数的定义必须出现第一次被调用之前；对内联函数不能进行异常接口声明。

如何理解形参与实参？

(1)在定义函数时声明形参，在未执行函数调用时，形参并不占用内存单元。只有在发生函数调用时，函数中的形参才被分配内存单元，并接收从实参传来的数据。在调用结束后，形参所占的内存单元也被释放。

(2)实参可以是常量、变量或表达式。在调用函数时用实参的值初始化形参。

(3)在定义函数时，必须在函数首部指定形参的类型和个数。

(4)实参与形参的类型应相同或可以进行隐含转换。

(5)实参变量对形参变量的数据传递是“值传递”，即单向传递，只由实参传给形参，而不能由形参传回来给实参。

示例

- 函数的不同调用形式
- 函数返回值
- 函数原型声明

知识点综述-函数的参数传递

函数调用因参数传递的不同可以分为传值调用、引用调用和传址调用。

- C++默认的参数传递方式是，当发生函数调用时，给形参分配内存空间，并用实参来初始化形参，成为传值调用。
- 引用调用是指用引用作为形参。
- 传址调用是指用指针作为形参，将在第六章介绍。

值传递与引用传递

值传递：

形参是实参的拷贝，改变形参的值并不会影响外部实参的值。从被调用函数的角度来说，值传递是单向的（实参->形参），参数的值只能传入，不能传出。当函数内部需要修改参数，并且不希望这个改变影响调用者时，采用值传递。

引用传递：

形参相当于是实参的“别名”，对形参的操作其实就是对实参的操作，在引用传递过程中，被调函数的形式参数虽然也作为局部变量在栈中开辟了内存空间，但是这时存放的是由主调函数放进来的实参变量的地址。被调函数对形参的任何操作都被处理成间接寻址，即通过栈中存放的地址访问主调函数中的实参变量。正因为如此，被调函数对形参做的任何操作都影响了主调函数中的实参变量。

示例

- 交换函数
- 通过引用返回多于一个的值

知识点综述-函数的嵌套和递归调用

- 函数允许嵌套调用。如果函数1调用了函数2，函数2再调用函数3，便形成了函数的嵌套调用。
- 函数可以直接或间接地调用自身，称为递归调用。
- 递归算法的实质是将原有的问题分解为新的问题，而解决新问题时又用到了原有问题的解法。按照这一原则分解下去，每次出现的新问题都是原有问题的简化的子集，而最终分解出来的问题，是一个已知解的问题。这便是有限的递归调用。只有有限的递归调用才是有意义的，无限的递归调用永远得不到解，没有实际意义。

如何理解递归的两个阶段

递归的过程有两个阶段：

第一阶段：递推。将原问题不断分解为新的子问题，逐渐从未知向已知推进，最终达到已知的条件，即递归结束的条件，这时递推阶段结束。

例如，求5!，可以这样分解：

$5!=5\times4 \rightarrow 4!=4\times3! \rightarrow 3!=3\times2! \rightarrow 2!=2\times1! \rightarrow 1!=1\times0! \rightarrow 0!=1$

如何理解递归的两个阶段

递归的过程有两个阶段：

第二阶段：回归。从已知的条件出发，按照递推的逆过程，逐一求值回归，最后达到递推的开始处，结束回归阶段，完成递归调用。

例如，求5!的回归阶段如下：

$5!=5\times4=120\leftarrow4!=4\times3!=24\leftarrow3!=3\times2!=6\leftarrow2!=2\times1!=2\leftarrow1!=1\times0!=1\leftarrow0!=1$

递归模型

递归模型是递归算法的抽象，它反映一个递归问题的递归结构。例如前面的递归算法对应的递归模型如下：

$$fun(1)=1 \qquad (1)$$

$$fun(n)=n*fun(n-1) \quad n>1 \quad (2)$$

其中，第一个式子给出了递归的终止条件，第二个式子给出了fun(n)的值与fun(n-1)的值之间的关系，我们把第一个式子称为递归出口，把第二个式子称为递归体。

一般地，一个递归模型是由递归出口和递归体两部分组成，前者确定递归到何时结束,后者确定递归求解时的递推关系。递归出口的一般格式如下：

$$f(s1)=m1 \qquad (1)$$

这里的s1与m1均为常量,有些递归问题可能有几个递归出口。递归体的一般格式如下：

$$f(sn+1)=g(f(si),f(si+1),\ldots,f(sn),cj,cj+1,\ldots,cm) \qquad (2)$$

其中，n、i、j和m均为正整数。这里的sn+1是一个递归“大问题”，si、si+1、…、sn为递归“小问题”，cj、cj+1、…、cm是若干个可以直接（用非递归方法）解决的问题，g是一个非递归函数，可以直接求值。

递归思路

把一个不能或不好直接求解的“大问题”转化成一个或几个“小问题”来解决；

再把这些“小问题”进一步分解成更小的“小问题”来解决；

如此分解，直至每个“小问题”都可以直接解决（此时分解到递归出口）。但递归分解不是随意的分解，递归分解要保证“大问题”与“小问题”相似，即求解过程与环境都相似。

示例

最小公倍数函数

函数LCM与函数GCD的嵌套调用过程

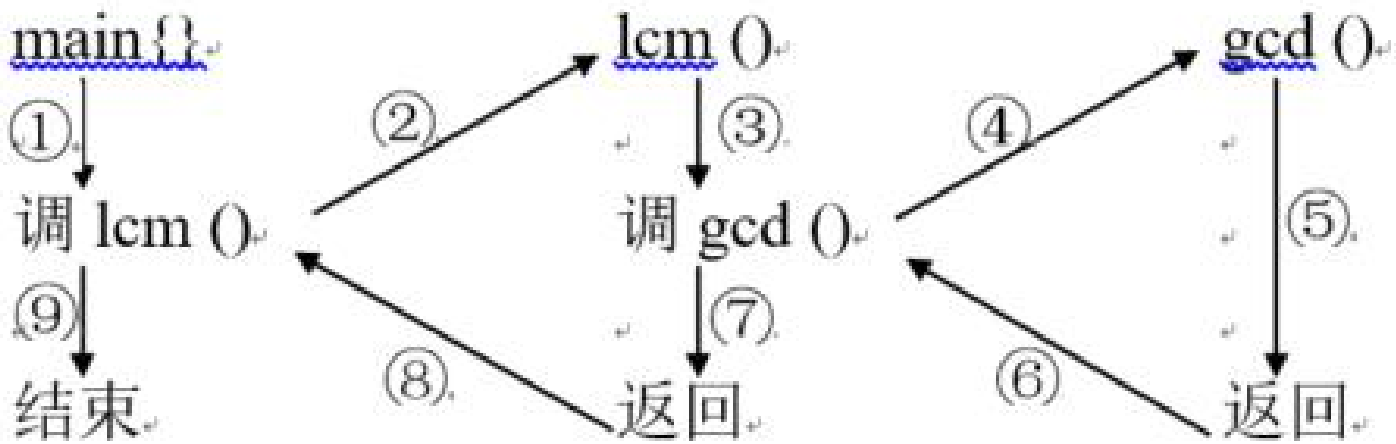


图 3-1 函数的嵌套调用过程

知识点综述-带默认形参的函数

函数在定义时可以预先声明默认的形参值。调用时如果给出实参，则用实参初始化形参，如果没有给出实参，则采用预先声明的默认形参值。

默认形参声明时该注意什么？

- 默认形参值必须按从右向左顺序声明，在有默认值的形参右面，不能出现无默认值的形参。因为在调用时，实参初始化形参是按从左向右的顺序。

- 在相同的作用域内，默认形参值的说明应保持唯一，但如果在不同的作用域内，允许说明不同的默认形参。有关作用域的概念，详见第5章。

示例

显示函数的参数-带默认参数值的函数

知识点综述-函数的重载

- 两个以上的函数，具有相同的函数名，但是形参的个数或者类型不同（也就是说函数的签名不同），编译器根据实参和形参的类型及个数的最佳匹配，自动确定调用哪一个函数，这就是函数的重载。

- 不要将不同功能的函数定义为重载函数，以免出现对调用结果的误解、混淆。

示例

类型可变的最值函数-参数类型不同的重载函数

知识点综述-C++系统函数

- C++不仅允许根据需要自定义函数，而且C++的系统库中提供了几百个函数可供程序员使用。系统函数的原型声明已经全部由系统提供了，分类存在于不同的头文件中。程序员需要用include指令嵌入相应的头文件，然后便可以使用系统函数。

- 充分利用系统函数，可以大大减少编程的工作量，提高程序的运行效率和可靠性。要使用系统函数应该注意以下两点：

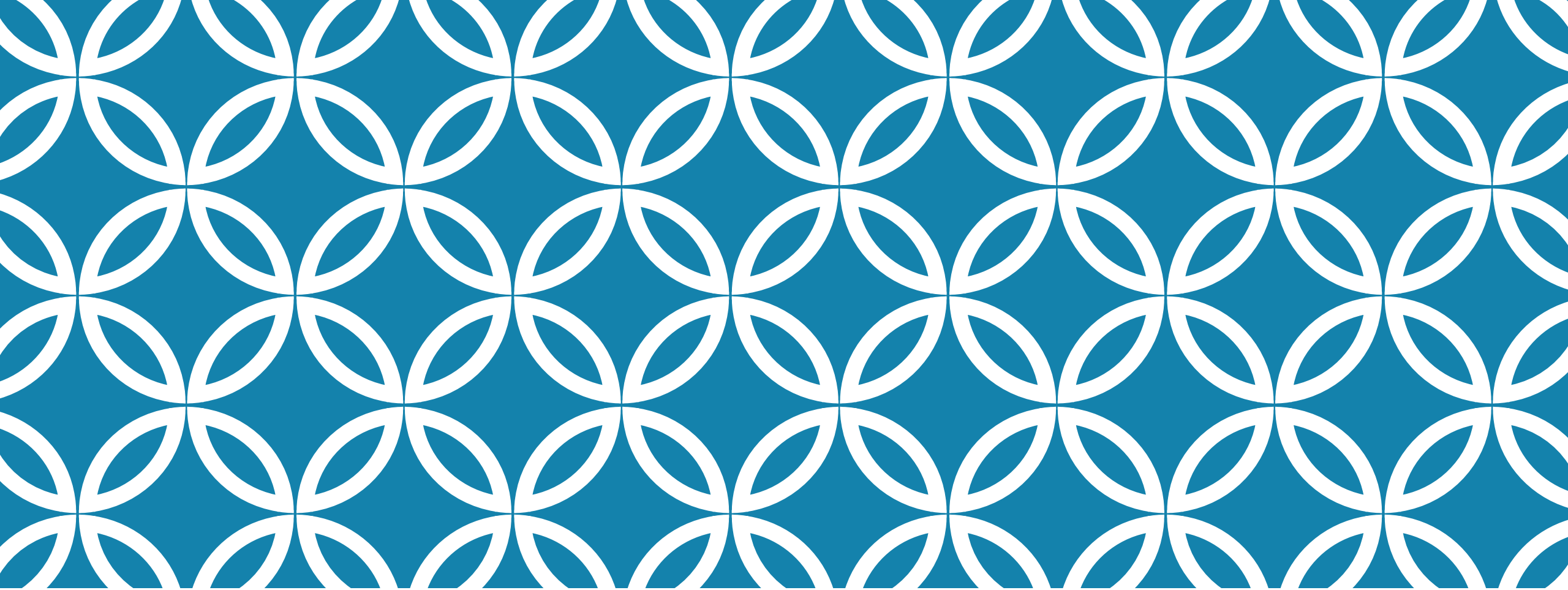
  （1）了解所使用的C++开发环境提供了哪些系统函数。编程者需要查阅编译系统的库函数参考手册或联机帮助，查清楚函数的功能、参数、返回值和使用方法。

  （2）知道要使用的系统函数声明在哪个头文件中。这也可以在库函数参考手册或联机帮助中查到。

示例

取整数-使用系统函数

求随机数-使用系统函数



LAB题

潘祖江
清华大学计算机系

C3-1 直角三角形

题目描述

输入一个三角形的3边长度，判断该三角形是否为直角三角形，若是则输出True，若不是则输出False。推荐做法：定义一个函数，接受三个int参数，返回bool，再用主函数调用之。

输入描述

每行输入三个由空格隔开的整数 a, b, c ，表示三角形的 3 条边长

1 <= a,b ,c <= 10000

输出描述

对于每一行输入，输出True或者False表明是否为直角三角形

样例输入

```
3 4 5
```

样例输出

```
True
```

样例输入

```
6 7 8
```

样例输出

```
False
```

C3-2 斐波那契数列

题目描述

斐波那契数列f(n)满足以下定义：

f(0) = 1, f(1) = 1, f(n) = f(n-1) + f(n-2) (n >= 2)。

请用递归的方法编写函数，对于给定的n，求出斐波那契数列的第n项f(n)

输入描述

每行输入一个整数 n

0 <= n<= 30

输出描述

对于每一行输入，输出斐波那契数列第n项的值f(n)

样例输入

```
1
10
25
```

样例输出

```
1
89
121393
```

C3-3 丑数

题目描述

只包含因子2,3,5的正整数被称作丑数，比如4,10,12都是丑数，而7,23,111则不是丑数，另外1也不是丑数。请编写一个函数，输入一个整数n，能够判断该整数是否为丑数，如果是，则输出True，否则输出False。

输入描述

每行输入一个正整数 n

1 <= n<= 1000000

输出描述

对于每一行输入，输出其是否为丑数，是则输出True，否则输出False

样例输入

```
4
7
12
```

样例输出

```
True
False
True
```

C3-4 完美数

题目描述

我们定义完美数为一个正整数，其值等于所有因子(不包含其本身)之和。

给定一个整数，判断是否为完美数

输入描述

输入一个整数c, 0 < c <= 10000;

输出描述

输出为一行, 如果是完美数，输出True; 否则输出False.

样例输入

```
28
17
```

样例输出

```
True
False
```

C3-5 开灯关灯

题目描述

有n盏灯，编号依次为1,2,3,...,n；初始化时，所有灯都是关闭状态；

小明第一次将编号为1,2,3,...,n都打开了；第二次将编号为2,4,6,....都关闭了（1~n之间的2的整数倍）；第三次将编号为3,6,9,...的灯进行操作（1~n之间的3的整数倍），若是之前关闭状态则打开，若是打开状态，则关闭；第四次将4,8,12,....的灯进行操作（1~n之间的4的整数倍）。这样操作了n次，问最后亮了几盏灯。

输入描述

第一行输入整数T，表示有T组测试用例；1<=T<=10000

接下来T行，每行有一个整数n，表示当前测试用例有n盏灯；1<=n<=10000

提示：总结开灯关灯规律。

输出描述

输出n行，每次测试用例最后亮灯的数目。

样例输入

```
2
1
5
```

样例输出

```
1
2
```

C3-6 出现次数最多的数字

题目描述

给定一串已经从小到大排好顺序的数字，其中有些数字出现了多次，请输出出现次数最多的那个数字。

如果有两个数字出现的次数一样，那么其中较小的那个。

输入描述

输入一个整数n，表示有n个已经排好顺序的数字，其中(0 < n < 1000).

接下来n行，每行一个数字。

输出描述

输出为一行，表示出现次数最多的那个数字。

样例输入

```
4
1
2
2
3
```