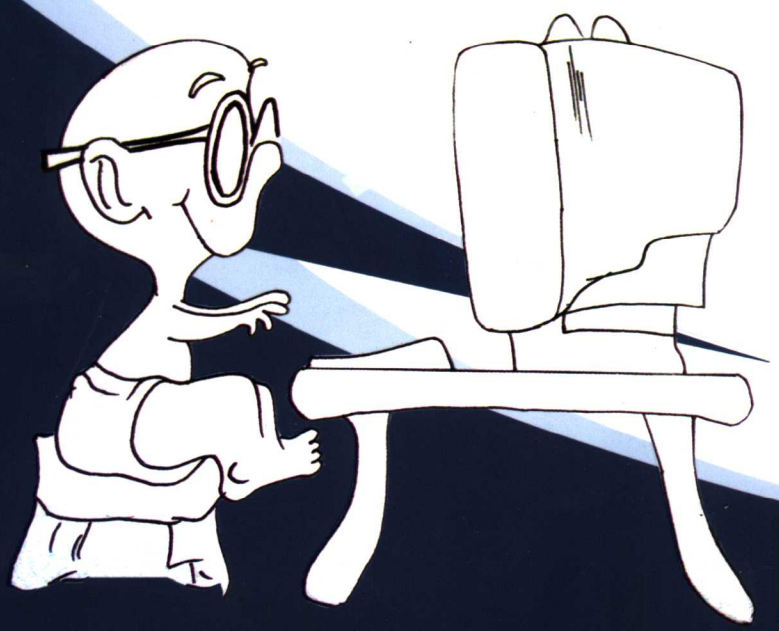
内附光盘

C++ 语言编程基础教程

宋振会 编著



清华大学出版社

C++语言编程基础教程

宋振会　编著

清华大学出版社
北　京

内 容 简 介

标准 C++是 ANSI 和 ISO 在 1998 年正式推出的国际化标准版本。本书便是按照此标准为基础，对 C++进行了全面、详细的介绍。

本书主要内容包括编程逻辑和技术，编辑和编译 C++程序，常量、变量和指针，运算符、优先级和结合律，函数、函数指针和指针函数，面向对象的编程方法（类），成员函数、构造函数和析构函数，条件判定、循环和跳转，数组、数组指针和指针数组，静态多态，类的关系，动态多态，多重继承、歧义性和虚基类，文件的输出和输入，类指针、单链表和双链表，用链表实现栈和队列。

本书编写时参考了大量的国际软件工程师培训教程，又借鉴了作者多年的编程经验和教学经验，采用符合国际性标准的编程方法和惯例，将一些高深、抽象的理论，通过大量的程序案例进行讲述。

本书是学习 C++语言编程的优秀教程，内容丰富，讲述清楚，通俗易懂，实例典型而丰富，适用于 C++培训学员、高等院校及职业院校的学生、其他 C++编程爱好者。

图书在版编目（CIP）数据

C++语言编程基础教程/宋振会编著. —北京：清华大学出版社，2005.5

ISBN 7-302-10767-X

Ⅰ.C…　Ⅱ.宋…　Ⅲ.C 语言-程序设计-教材　Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2005）第 028275 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969

责任编辑：欧振旭
封面设计：姜凌娜
版式设计：崔俊利
印 刷 者：北京四季青印刷厂
装 订 者：北京市密云县京文制本装订厂
发 行 者：新华书店总店北京发行所
开　　本：203×260　**印张**：19.75　**字数**：440 千字
版　　次：2005 年 5 月第 1 版　2005 年 5 月第 1 次印刷
书　　号：ISBN 7-302-10767-X/TP·7168
印　　数：1～5000
定　　价：29.00 元（附光盘 1 张）

前言

编写目的

目前，由于软件产业的美好前景和诱人薪酬，使软件培训和学习如火如荼，热遍大江南北。诸如 NIIT（印度国家信息技术学院）、北大青鸟等知名的培训机构在国内就有 1000 多家，培训学员达 200 多万。本书作者为 NIIT 特聘教师，在 C++培训和教学过程中，深切体会到需要一本内容翔实，通俗易懂的教程作为 C++培训学员或自学人员的教材或参考书。为此编写了本书。

编写本书的另一个目的是，作者试图以新的角度来探讨培训和自学教程的写作模式，给现有相关书籍的千篇一律带来一些新思想。作者希望通过本书，能将复杂的问题简单化，能使读者掌握 C++编程的捷径。

主要内容

本书主要内容包括编程逻辑和技术，编辑和编译 C++程序，常量、变量和指针，运算符、优先级和结合律，函数、函数指针和指针函数，面向对象的编程方法（类），成员函数、构造函数和析构函数，条件判定、循环和跳转，数组、数组指针和指针数组，静态多态，类的关系，动态多态，多重继承、歧义性和虚基类，文件的输出和输入，类指针、单链表和双链表，用链表实现栈和队列。

读者对象

本书明确定位于学习 C++编程的入门与提高人员。主要适合以下读者：

- 参加 C++培训的学员。
- 高等院校学习 C++编程的学生。
- 中职、高职学校学习 C++编程的学生。
- 其他 C++爱好者或自学人员。

特色提示

本书是一本与众不同的书籍，它将带给读者耳目一新的感觉。本书具有以下特色：

- 结合 NIIT 软件培训教材的特点编写，在内容的选取、讲解、实例及课后实践等方面都

力求有代表性。

- 探讨了快速授课的教学模式，是一本很好的C++培训教材和自学教材。
- 选材准确，不讲废话。本书的内容是任何一个初学 C++编程的读者所必需掌握的最基本、最常见、最实用的内容。不符合C++初学者的内容一概不涉及。
- 形式新颖，适于阅读。本书摒弃了传统图书编排方式的呆板，代之以活泼而清新的风格，让读者阅读时有一种轻松感。

阅读建议

作为一本基础教程，作者建议读者按照章节的顺序阅读，并且注意阅读每章后的小结，完成独立实践。对于书中所有范例程序的源代码，希望读者能在读懂代码的基础上，亲自上机调试，看能否正常运行。只有这样，才能真正深刻理解所学内容。

配书光盘

本书附带一张光盘，内容为本书所有范例程序的源代码及书中练习和独立实践的源代码。其中，部分代码为片段代码，部分代码是需要读者更正的错误代码，这些代码在书中都有详细讲述，并不能完全编译运行。其他完整的范例程序都是在 Microsoft Visual C++编译器编译，能正常运行，无任何错误。

配书光盘中的源代码按章收录，且文件名和编号与书中完全对应。例如，书中第2章的“程序 2.1：我的第一个 C++程序.cpp”对应的源代码在光盘中的目录为：/第 2 章编辑和编译C++程序/程序 2.1：我的第一个 C++程序.cpp。请读者按此方法在本书的配书光盘中检索本书范例程序的源代码。

作者情况

本书主要由 NIIT 特聘教师宋振会教授编写。作者长期讲授 C++、Java、SQL 等课程，在教学过程中积累了丰富的经验，了解初学人员的学习特点，熟悉教学的重点与难点。NIIT 软件培训基地的李秋芬、张瑾、孔祥国等老师在素材提供、代码调试、后期审阅等方面做了大量工作。在此一并表示衷心的感谢！

由于作者水平所限，加之写作时间比较仓促，书中可能还有不足和疏漏，恳请广大读者批评与指正。

E-mail：0532-2906053@tom.com
tel2906053@tom.com

编著者

2005 年 2 月

目 录

第1章

编程逻辑和技术

目标：

本章中，你将学习：

- ☑ 引入框图
  - ☑ I-P-O 周期
  - ☑ 使用框图表示程序流程
- ☑ 变量、常量和内存
- ☑ 循环迭代和条件判定
  - ☑ 预检和预检表
  - ☑ 循环迭代
  - ☑ 条件判定
- ☑ 理解编程的模块化方法
- ☑ 养成一个好的习惯

开始

我们在开始学习任何一门编程语言之前，首先需要知道和了解编程所使用的技术。也许，你学过 BASIC、FORTRAN 和 COBOL 等编程语言，在编程之前，首先要使用框图绘制程序流程图。绘制程序流程图的过程就是程序设计的过程，它包含着编程逻辑和编程技术，就好像在写这本书之前，首先要写好大纲和目录，确定要编写的内容和深度以及所面向的读者群体，架构起本书的框架一样。

本课提供对编程技术的理解，使用这种技术来表示使用程序求解问题所需要的编程逻辑。主要使用框图技术开发基本的编程逻辑。

本章你将学习如下技术：

- 使用框图绘制程序流程图。
- 用条件表示程序逻辑。
- 理解循环和迭代。
- 实现模块化编程。
- 学会一题多解，反向思维和超常规思维。

引入框图

I-P-O 周期

计算机执行的一个活动周期也遵循着“输入—处理过程—输出”周期，或称 I-P-O 周期（In-Process-Out）。计算机由几个部分组成，如键盘、鼠标、中央处理器（CPU）、存储器、显示器、打印机等。每个部件参与输入阶段、处理过程阶段、输出阶段的某一个活动。例如，键盘和鼠标用于输入阶段；计算机系统内部的中央处理器和存储器用于处理过程阶段；显示器和打印机用于输出阶段，如图 1-1 所示。

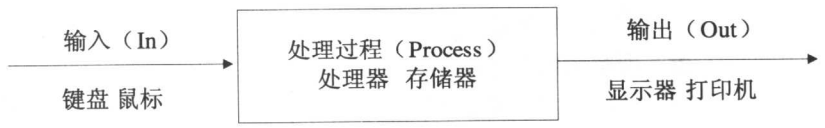


图 1-1　一个 I-P-O 周期

例如：从键盘输入两个数，并求两个数的和。

1. 输入阶段：从键盘输入两个数，并将两个数存储在计算机的两个内存单元中。
2. 处理过程：中央处理器从指定的两个内存单元中取出两个数据，并将两个数据相加，将结果存储在另一个内存单元中。
3. 输出阶段：取出所指定的内存单元中的数据，输出显示在显示器上。

使用框图表示程序流程

框图

框图是求解程序问题要遵循的步骤的图形符号，它由一组标准专用的图形符号组成，每个图形符号表示一个特定的活动内容。一个典型的程序处理过程涉及接受输入、处理输入及显示输出。处理过程涉及求解程序问题所采用的条件判定、循环迭代、程序跳转等结构。

框图中使用的图形符号

让我们看一下框图中表示 I-P-O 周期所用的某些图形符号。

- 任何问题的输入用平行四边形▱表示，包括从键盘、鼠标、文件的输入。
- 长方形▭表示对输入的处理过程，以产生结果和输出。
- 图形⬭表示输出，包括输出到显示器、打印机及文件。

类似地，还有一些图形符号表示开始、结束以及程序中求解问题所需的结构。表 1-1 列出了框图中所使用的图形符号和活动。

表 1-1　框图使用的图形符号

序　号	符　号	活　动
1		输入框，用于表示所有的输入活动
2		处理框，用于表示对数据的所有处理活动
3		输出框，用于表示对结果的所有输出活动
4		判定框，用于求值一个条件的真和假的活动
5		子例程框，在程序流程中调用一个独立的程序模块
6		流线图，连接框图的步骤，指出要执行的顺序。顺序应是自顶向下、从左向右
7		终结符，指出框图的开始和结束
8		页连接符，用于同一页中框图的一个步骤连接到同一页中的另一个步骤
9		离开页的连接符，用于不同页中框图的一个步骤连接到另一页中的另一个步骤
10		注释框，用于插入到框图的注解，插入注解提供对步骤的解释，便于阅读

画出简单的框图

既然知道了框图中使用的不同符号，下面我们考察使用框图获得解决实际程序问题的方案。例如，从键盘输入两个数，并求两个数之和的框图如图 1-2 所示。

图中用语言描述的形式，画出了从键盘输入两个数，并求两个数之和的流程图。那么，从键盘输入两个数和两个数求和结果存放在什么地方？为了使程序流程图更详细，更接近程序源代码，需要了解变量、常量和内存。

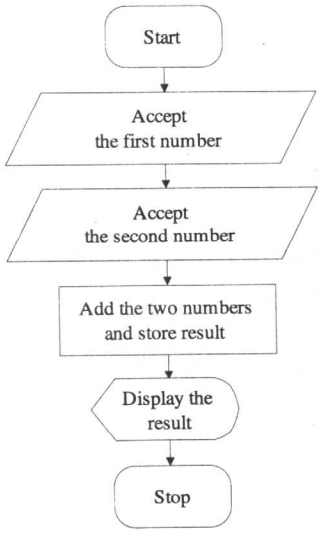
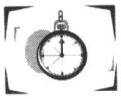


图 1-2　从键盘输入两个数，并求两个数之和的框图



问题与思考 1

以正确的顺序重新安排下面给出的步骤，从而把两个数相加，并画出框图。

Accept the first number
Display result
Stop
Add the two numbers
Accept the second number
Start

变量、常量和内存

计算机内有存储器，也就是通常所说的内存。内存用来存储用户提供的输入数据，处理输入的指令及处理结果。存储器由不同的存储数据的单元组成。为理解计算机如何处理数据，还是看前面的例子。当执行输入指令时，第一个数据被接受并存储在存储器的一个内存单元中，第二个数据被接受并存储在存储器的另一个内存单元中。当执行相加处理命令时，中央处理器从两个内存单元中取出数据相加，并将结果存储在一个内存单元中。当执行输出命令时，中央处理器从内存单元中取出数据并显示到屏幕。

在编程中，如果要使用变量则必须提前声明。声明变量事实上就是申请了一个内存单元。变量的名称即内存单元的名称。使用数据类型声明变量，例如 int（整型），第一个变量为 Num1，第二个变量为 Num2，两个数相加的结果存放的变量为 Sum。

从键盘输入两个数，并求两个数的和，显示声明变量并使用变量的框图如图 1-3 所示。

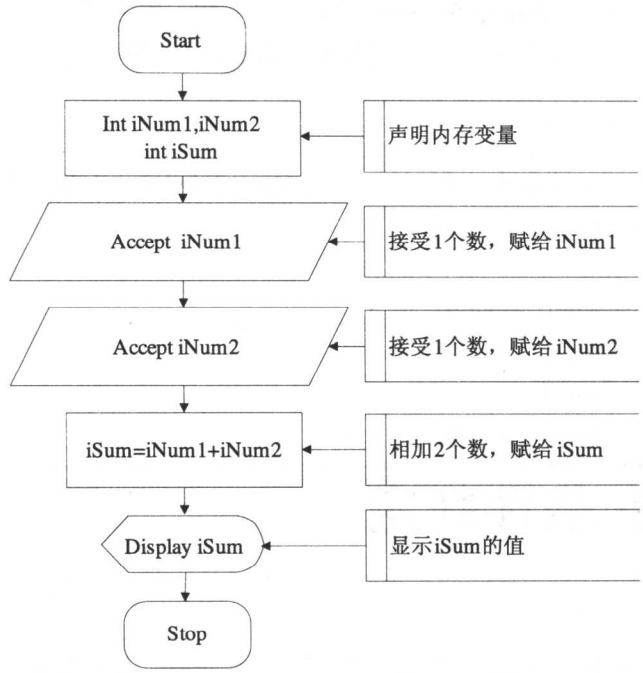


图 1-3　从键盘输入两个数，并求两个数的和

在程序执行过程中，iNum1、iNum2 和 iSum 的数值是变动的，这依赖于用户的输入和程序指令的执行。例如下面这条程序指令：

```
iNumber =iNumber + 1
```

在数学上，这条指令无论如何也无法理解。但在计算机中它完成了读操作和写操作，中央处理器读取 iNumber 存储单元的数据，执行加 1 操作，然后将结果重新存储在 iNumber 存储单元中。这里“=”是赋值操作符，而不是数学中的等号。

在上面的程序流程中，将数据 10 赋值给 iNum1 存储单元，将数据 15 赋值给 iNum2 存储单元，将数据相加的结果赋值给 iSum 存储单元，如图 1-4 所示。

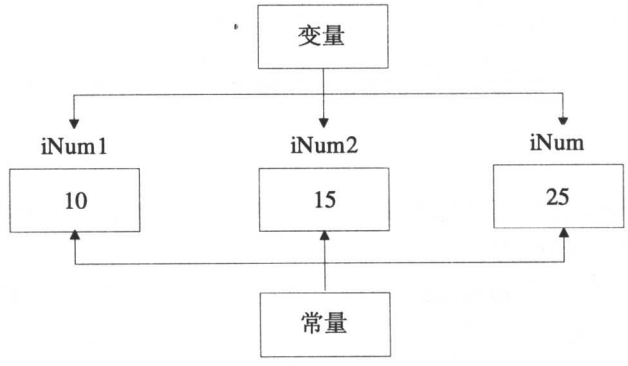


图 1-4　变量和常量

内存单元 iNum1、iNum2 和 iSum 的值是可变的，称为变量；而内存单元中存放的数据 10、15 和 25 是不变的，因此存储在变量中的值被称为常量。

问题与思考 2

考察下列程序指令，预测 iNum1、iNum2 和 iNum3 存储单元中的数据。

```
iNum1 = 1;
iNum2 = 2;
iNum3 =iNum1 +iNum2
iNum1=iNum2
iNum2=iNum3;
iNum3 =iNum1 +iNum2
```

循环迭代和条件判定

预检和预检表

编程语言能使编写的程序可以计算连贯的复合条件，这些复合条件在框图中有时导致复合的流程图。预检的概念将帮助你执行逻辑检查和理解框图中的控制流。也可以使用预检表，通过一组样本值对程序可能的输出求值。预检表提供了一步步地计算程序中变量的值。下面使用预检表，求问题与思考 2 中 Num1、Num2 和 Num3 存储单元中的最终数据，预检表将帮助你理解这段程序指令的含义，如表 1-2 所示。

表 1-2　预检表

序　号	程序指令	Num1	Num2	Num3
1	Num1 = 1;	1（赋值）	未知	未知
2	Num2 = 2;	1（不变）	2（赋值）	未知
3	Num3 = Num1 + Num2	1（不变）	2（不变）	3（赋值）
4	Num1= Num2	2（赋值）	2（不变）	3（不变）
5	Num2= Num3;	2（不变）	3（赋值）	3（不变）
6	Num3 = Num1 + Num2	2（不变）	3（不变）	5（赋值）

循环迭代

计算机的一个重要特征就是重复地执行一串指令的能力。计算机的这种能力，可让你具有控制重复执行任务的能力。

例子：要求产生一个 Fibonacci 数列，该数列的前两个数为 1 和 2，从第三个数开始，每个数为前两个数之和。

1．声明三个变量 Num1、Num2 和 Num3，分别用于保存数列中的连续三个数。

2．赋值给数列第一个数 Num1=1 并显示第一个数。

3．赋值给数列第二个数 Num2=2 并显示第二个数。

4．将前两个数相加赋值给数列的第三个数 Num3 = Num1 + Num2，并显示第三个数。

接下来应该是第三个数和第二个数相加赋值给数列的第四个数 Num4 = Num2 + Num3，并显示第四个数。这样 Fibonacci 数列有多少个数，就要声明多少个变量，相加并赋值的指令代码就要重复写多少遍。很显然这是不现实的。怎样将上述问题简化为可以重复执行的数学模型呢？由于内存的数值是可变、可重复赋值的，考察下列两行程序指令。

- Num1 = Num2；数列第二个数变成第一个数。
- Num2 = Num3；数列第三个数变成第二个数。

经过上面两条程序指令，交换了数列的值，这样就简化为一个数学问题：Num3=Num1+Num2，执行程序第四条指令，就可以求出数列第四个数，再重复执行程序第五、六、四条指令，就可以依次求出数列的各个数。

产生 Fibonacci 数列的程序流程图如图 1-5 所示。

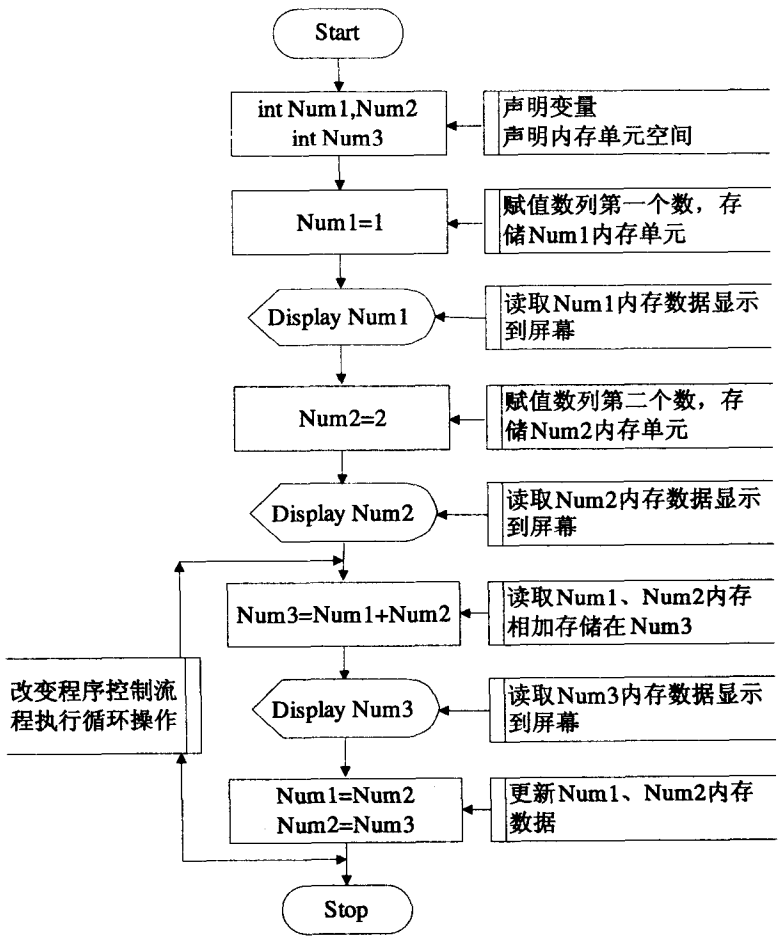


图 1-5　产生 Fibonacci 数列的程序流程图

下面使用预检表求一个 Fibonacci 数列的输出，参见表 1-3。

表 1-3　用预检表求一个 Fibonacci 数列的输出

序　号	程序指令	Num1	Num2	Num3	Display
1	Num1 = 1　Display	1	未知	未知	1
2	Num2 = 2　Display	1	2	未知	2
3	Num3 = Num1 +Num2　Display	1	2	3	3
循环 1	Num1 = Num2;Num2=Num3 Num3 = Num1 +Num2　Display	2	3	5	5
循环 2	Num1=Num2;Num2=Num3 Num3 = Num1 +Num2　Display	3	5	8	8
循环 3	Num1=Num2;Num2=Num3 Num3 = Num1 +Num2　Display	5	8	13	13
循环 4	Num1=Num2;Num2=Num3 Num3 = Num1 +Num2　Display	8	13	21	21
循环 5	Num1=Num2;Num2=Num3 Num3 = Num1 +Num2　Display	13	21	34	34
循环 6	Num1=Num2;Num2=Num3 Num3 = Num1 +Num2　Display	21	34	55	55
循环 7	Num1=Num2;Num2=Num3 Num3 = Num1 +Num2　Display	34	55	89	89
循环 8	Num1=Num2;Num2=Num3 Num3 = Num1 +Num2　Display	55	89	144	144

条件判定

在上面求一个 Fibonacci 数列输出的程序中，通过数据交换，始终让 Num2 和 Num1 值总是数列最后的两个值，然后重复执行 Num3 = Num1 +Num2。这里有一个问题，循环到什么时候为止呢？或者说在最后的丁字路口处，程序怎样知道是继续返回循环还是退出终止循环呢？这里需要有条件判定。循环有两类：

- 固定的循环（重复的次数已知）
- 可变的循环（重复的次数未知）

例如，在上面的 Fibonacci 数列中，如果要求创建 Fibonacci 数列的前 10 个数，则为固定的循环（重复的次数已知）；如果要求创建 Fibonacci 数列的最大数小于 100，则为可变的循环（重复的次数未知）。

这两种循环，产生 Fibonacci 数列的程序流程图如图 1-6 所示。

第一种循环以 Fibonacci 数列的最大数 Num3（或 Num2，因为 Num2=Num3）为判定条件，如果 Num3<100 条件判定为真，则重复执行循环体，否则，Num3<100 条件判定为假，则退出循环体，执行循环体后面的语句（Stop）。

第二种循环以 Fibonacci 数列的个数为判定条件，这里需要声明一个新的变量 Counter（计数器），用来累加循环的次数。由于 Fibonacci 数列已经有了前两个数，所以变量 Counter 初始化为 2（Counter=2），每执行一次循环体 Counter 变量加 1，其中“++”是 C++增量运算符（加 1 运算符），Counter++等价于 Counter = Counter+1。如果 Counter <10 条件判定为真，则重复执