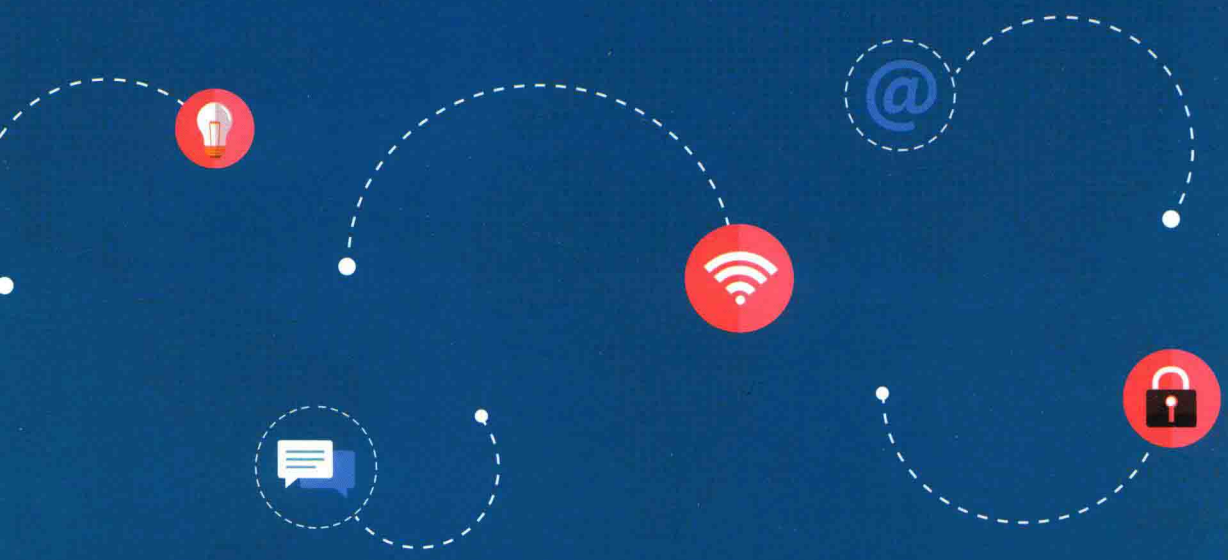

Easy to Understand C++

通俗易懂C++

鲍 钰◎编著

华东师范大学出版社

高校教材

Easy to Understand C++

通俗易懂C++

鲍 钰◎编著

华东师范大学出版社

图书在版编目(CIP)数据

通俗易懂C++/鲍钰编著. —上海:华东师范大学出版社,2017
ISBN 978-7-5675-7246-1

Ⅰ.①通… Ⅱ.①鲍… Ⅲ.①C++语言—程序设计—高等学校—教材 Ⅳ.①TP312.8

中国版本图书馆CIP数据核字(2017)第296172号

通俗易懂C++

编　　著　鲍　钰
策划编辑　赵建军
项目编辑　孙小帆
特约审读　周佳清
装帧设计　俞　越

出版发行　华东师范大学出版社
社　　址　上海市中山北路3663号　邮编200062
网　　址　www.ecnupress.com.cn
电　　话　021-60821666　行政传真021-62572105
客服电话　021-62865537　门市(邮购)电话021-62869887
地　　址　上海市中山北路3663号华东师范大学校内先锋路口
网　　店　http://hdsdcbs.tmall.com

印 刷 者　上海市崇明裕安印刷有限公司
开　　本　787×1092　16开
印　　张　10.5
字　　数　187千字
版　　次　2018年6月第1版
印　　次　2018年6月第1次
书　　号　ISBN 978-7-5675-7246-1/TP·121
定　　价　31.80元

出 版 人　王　焰

序

C++语言是目前流行最广，功能最强大的语言，没有之一。以前程序员圈里有这么一句话，真正的程序员用C++，聪明的程序员用DELPHI。当然，VB圈里则会说，真正的程序员用C++，聪明的程序员用VB。无论聪明的是谁暂且不争论，真正的程序员用的永远都是C++。C++就好像武林至尊，在编程语言中的地位不可撼动。

C++语言是在C语言的基础上发展起来的，其产生是因为C语言函数分解FD的思想在软件大作业中遇到了致命的问题，这个问题很有名，叫软件危机。于是，科学家们指路，面向对象OOD语言C++轰轰烈烈的到来了，这标志着一个新的时代。

可以说，面向对象思想解决了大型软件开发中精细分工的问题，程序员可以各司其职，只关注自己的对象，即类的实现即可。C++语言是科班出身软件专业学生的必修之路，可是C++涉及的编程思想非常全面，要全面掌握它是需要长时间深厚积累，慢慢体会的，而且里面有很多内容理解起来比较枯燥难懂，所以本书采用了通俗易懂的口语化书写方式，将复杂的内容形象化，通过打比方的叙述方式，将抽象的内容生动化。

本书结合笔者十多年的一线教学经验，从学生的感受和易于理解的特性出发，从科班出身人员的综合编程素质培养入手，介绍了很多编程思想的概念和要点。书中涵盖了面向对象程序设计C++语言方面比较全面的知识体系，每个知识点都有详细生动的介绍和例子说明。书中一共有13章内容，介绍了130多个知识点，通过形象的比喻和简明扼要的例子代码，将难懂的知识概念变得浅显易懂。本书特别适合高等学府学习面向对象编程语言的大学生，可以快速将学生们领进门。书中还介绍了很多科班出身的学生需要明白的编程素养知识，在潜移默化中培养学生真正的编程思维。

本书的作者是华东师范大学软件学院资深的C++课程讲席正教授，有十多年的语言类课程讲授经验，其轻松风趣的授课方式深受大学生喜爱，曾获得IBM大学合作部评选的全国优秀教师、华东师范大学十大“学生心目中最优秀教师”等荣誉称号。

最后，感谢李琪琦、李梦芸、章文利、李凌云、王路遥等同学为本书做了部分文字录入工作，感谢华东师范大学出版社孙小帆老师给予的帮助，在此表示衷心地感谢！

笔者

2017年9月1日于上海

目录

第 3 章　C++ + 数据类型介绍　27

第 4 章　输入和输出　41

第 10 章　结构体和类　97

第 11 章　数组　111

1

第 1 章

四种控制结构

第1章

四种控制结构

本章开门见山，我们先介绍C++程序中的四种控制结构，分别是顺序控制结构、选择条件控制结构、循环控制结构和子函数子过程控制结构。这四种控制结构如果弄明白的话，一些简单的编程题就可以开始写了。

现在来看一看C++程序中主要的这四种控制结构。

1.1 顺序控制结构

顺序结构是一个什么概念呢，举一个例子。比如有如下的一段代码(先不用管具体的语法)：

```
int a=10;
cout<<a<<endl;
a=20;
cout<<a<<endl;
```

这是一段非常简单的代码，C++总是由一句语句接着一句语句组成的，每个语句的最后有一个分号“;”作为语句的结束符。缺省的情况下，C++一条语句做完，再做下一条语句。按照这个顺序依次往下做，这是C++缺省的控制结构，是一种顺序结构。C++顺序结构是一个最重要的串行执行的语法结构。有的同学一开始不明白顺序结构，把代码按随便次序放在一起，比如说：

```
int a=10;
cout<<a<<endl;
a=20;
```

上面的代码第一行先做a=10，即对一个整型变量a初始化为10，然后输出a的数值10到显示器屏幕上，再用操作符endl输出一个换行符。接着第二行a被赋值成了20。但是这个语句要是把它的顺序变一变的话，整段程序的运行结果就完全不一样了，比如修改成下面的顺序：

```
int a=10;
a=20;
cout<<a<<endl;
```

这时按照从上到下的执行顺序，a先被初始化为10，然后被重新赋值为20，最后输出a的数值20和一个换行符到电脑屏幕上。这就是C++语言的第一种控制结构，叫顺序控制结构。C++通常都是按照此执行顺序，就像平时开车去旅游，车子沿着一条笔直的公路一步一步往前开，是一个过程化的从上到下串行顺序执行结构。

1.2 选择条件控制结构

打个比方，当我们的车开到一个分岔路口，这个分岔路口有一个条件，也就是指示

牌，标明了往左边开是往北方北京方向，往右边开是往江浙一带南方走。这个指示牌有条件地控制程序选择哪一条分支。这样一个判断的控制结构一般带一个布尔类型的条件(true 或者 false)，你是要往北走吗？是，那就是 true；否，就是 false。这就是一个选择条件控制结构，在 C++语句中，它通常是通过 if... else... 的语句来实现的。下面这段程序首先定义了一个字符类型的变量 sex 并初始化为字母 g，然后是一个选择控制条件判断 sex 是否等于 b(该 bool 类型的表达式结果是 false)，所以执行的是第二个分支 else 里的语句，输出 girl 喜欢的活动 go to Global Harbor 即去环球港逛街。若开始 sex 被初始化为 b 即 boy 的时候，程序就会选择第一个分支最终输出的是 play football 即去踢球。

```
char sex='g';
if(sex==' b')
{
    play football;
}
else
{
    go to Global Harbor;
}
```

注意，如果 if 语句后面只有一行语句的时候(一个分号作为一行语句的结束标志)，那大括号{}可以有也可以省略，保留{}是最标准的写法，是一个 if 语句块，但有时这个 if 语句块里面只有一句代码，{}就可以省略不写。

还有一个注意点，语句 sex==‘b’，这里用的是“==”，前面赋值语句采用的是“=”，“=”表示的是一个初始化或者赋值，但是如果要去判断某个条件是否成立的时候，我们需要用“==”，“==”代表判断左右两个值是否相等，如果这两个值是相等的，那就返回 true，否则就返回 false。对于这个 if 条件如果为真的话，那就执行 if 下的语句块；如果这个条件不成立(返回 false，也就是为假)，会执行 else 后面的语句块。

有同学会问，如果三条路的话，情况会是什么样呢？

回答：三条路可以在 if 里面再套 if... else，或者还有 switch，case 的语句，这个我们后面会讲。这些都是条件控制结构。

1.3 循环控制结构

我们有时候会在现实生活中碰到一种循环逻辑，比如上体育课，体育老师说今天天气不错，同学们跑 10 圈去。一般大学校园的操场一圈是 400 米，10 圈就是 4 公里，我们阳光的大学生应该很轻松就能跑完。同样事情做了十次，然后出来结束，那么这个程序该怎么写？

我们来看这个 while 循环。上课了，铃声响了，体育老师说跑十圈，怎么跑？

```
int count=0;
while(count<10)
{
  Running();
  cout<<"这是第"<<count+1<<"圈";
  count++;
}
```

这里要跑十圈，所以要有一个计数的变量 count。同学们一定要区分，cout 是输出，count 是定义的计数变量。还有，有同学把主函数 main 写成 mian，这里是主函数，而不是面条。C++是一个很严谨的语言，哪怕敲错一个字符，一字之差可能会出现很奇怪的错误。比如将 count 敲成 Count，C++是区分大小写的语言，所以 count 和 Count 是完全不同的两个变量。首先 int count=0；这里定义了一个计数器变量 count 并将其初始化为整数 0。有的同学如果有类的知识基础，上述语句也可以写成 int count(0)；这种写法是用类的构造函数来写的，说明该同学已经知道 class 了。

count=0，跑十圈，那么这里要有一个 while 语句，当后续()内的判断条件为 true 时，不停的循环执行{}内的代码。这里 count<10，为什么不是<=10？因为 count 是从 0 开始计数的。count<10，从 0 开始执行，0，1，2，3，4，5，6，7，8，9 正好十次。但是如果 count 是从 1 开始，那么就应该是 count<=10，1 执行到 10 也是十次。

所以我们在写程序的时候一定要按照逻辑来，0～9 是 10 次。1 执行到 10 也是十次。每跑完一圈的时候心里数一下：count++，count++什么意思？其等价于 count=count+1。但一般我们在增 1 的时候会用 count++，它的效率会高一些，书写起来会方便点，没必要写 count=count+1。增 1 之后应该怎么办，肯定要说一句嘛，我又跑了一圈了，我们用一个 Running()函数来表示。

图 1-1　操场跑圈

当然我们还有 do...while，for 等循环语句。所以涉及逻辑上有这种循环往复的时候，就不妨用这些循环语句。如果有条件分支的时候，就用 if...else 语句。

1.4 子过程结构

子过程结构是一种分而治之的控制结构。什么叫分而治之？举一个简单的例子，假设要把一个习题库所有的题目都做一遍，这个任务非常庞大，有三千多道题，全部做完这个实现起来非常艰难。但是，我们可以每天做十道题，经过一年就完成了三千多道，这就是分而治之的思想。把一个复杂艰巨的任务，通过日积月累分成很多时间段来完成。写程序有时候也会遇到这种问题，可能突然间遇到一个很复杂的问题，那么就可以把这个问题分成若干多个小问题，一个个小问题解决了，然后慢慢地大问题也就解决了。举一个例子，现在上数学课，数学课老师说，我们来计算一下 10!。你可能会觉得太烦了，要算 10 次，但是我们可以通过分而治之的思想做，并根据这个思想实现一个函数，这个函数就是用来算阶乘的。

```
int factorial(n){
  if(n==0)
    return 1;
  else
    return n*factorial(n-1);
}
```

形参中传一个参数 n 进来，factorial(n)就是求 n! 很简单，0! =1，所以通过语句 if (n==0) return 1;实现这个逻辑。C++支持递归，这实际上是一种递归的写法。求阶乘，感觉挺复杂的，但是现在通过递归的思想，写起来发现原来这么简单，就这么两行话。按照数学的定义，n 如果等于 0，就 return 1；否则 else return n * factorial(n-1)，就是返回 n 乘以 n-1 的阶乘。

有同学不懂 return，return 就是终止当前函数并返回一个结果值给调用者。有时一个函数是没有返回值的，比如 void main，这代表这个函数没有返回值，写完代码之后就不能写 return 0 了。因为 void 是没有带返回值的意思，要么一个空的 return 语句，要么不写 return。空的 return 语句代表程序结束了，但没有任何返回值。

这个时候数学老师突然说，请你们计算 1～10 的阶乘。联想到刚刚跑十圈的题目，那个跑十圈的程序再加上这个阶乘函数，你会发现这个程序很好写了。怎么写？

```
int count=1;
while(count<=10){
  cout<<count<<"的阶乘是"
  <<factorial(count)<<endl;
  count++;
}
```

首先 count=1，当 count<=10 的时候进入这个循环，输出 count 的阶乘(调用 factorial 函数，返回结果)，一轮做完之后 count++。这段程序分出去一个子函数，叫 factorial()，它

是用来求阶乘的。可以把子函数想象成一个功能模块，一个黑盒子，可以往这个黑盒子里面丢一些东西（通过实参向形参的参数传递），然后通过函数内部的逻辑运算生产出结果。比如这个阶乘函数，我丢1个5进去，通过它的逻辑生产加工，它输出的是5!。

本章小结

本章介绍了C++程序中的四种控制结构。我们用非常详细的代码分别解释了顺序结构、条件结构、循环结构、函数子过程结构，所有的C++程序都离不开这四种结构。我们以后还会学其他编程语言，比如Java，c#等也是非常主流的程序设计语言，这些语言中也是同样的这四种控制结构。所以学习好了C++语言中的编程思想和相关知识点概念，以后再学习其他编程语言都会觉得似曾相识，游刃有余了。

要记住，写程序就像用C++语句写作文，好的C++程序看起来是非常舒服的，是需要循序渐进不断练习的，这是一个过程，千万不要想着能一下子把程序写得很漂亮，同学们在书写的过程中逐渐积累经验，等到程序写多了，自然就发现写得很漂亮了。