目 录

第一部分 C++ 基 础

第一章 面 向 对 象

1.1 简 介

面向对象(Object-Oriented)已经成为90年代重要主题词之一，可用如下说法简单地表达面向对象的整体思路：先对特定问题建立通用的解决方案，然后再改制这些方案以适应特定需要，这样的设计方法内在的优点是：

- 能够直接地再利用他人已经设计并编写成功的程序，而不需要再作广泛的重新测试。
- 能够由基本而通用的解决方案派生出新的解决方案。
- 能够建立一个由模块组成的系统。

用实物作比拟会有助于弄清模块化系统的概念。比如一种组合型长椅，是按这样的结构设计制造：适合某种特定需要，其中的一块或几块既可以加到长椅的椅架上且可以从椅架上拿开，又不影响长椅本身的性能和设计上的美观，又比如一个模块化的房间，可以设计得既容易得到扩展又不影响房子的基本结构。模块化系统与上述所列的比拟很相似，是由包含系统基本功能的固定基组成的。这个固定基是待完善系统的构件，它容易扩展，新的功能可从这个基派生出来，无需进行广泛的重新测试和编码。

面向对象的系统的主要缺点（如果称之为缺点的话），就是无法正确地设计一个足够通用的解决方案来构成系统的基，因此，要建立一个强大而有用的面向对象的系统，使系统具有长期的可用性是至关重要的。

C++ 是一种面向对象的编程语言，但如果系统没有采用面向对象的技术，C++ 只能作为面向过程的语言来使用。为了将 C++ 带入生活，读者必须考虑面向对象。在编写第一行程序之前，必须尽力地花时间去思考系统的设计、系统的基以及期望由系统基所派生出来的解决方案。如果没有考虑到面向对象和模块化，那 C++ 这种工具的面向对象强大之处将变得毫无用处，因为没有去正确使用它。本章将教会读者在编程之前如何构思。

1.2 抽 象

抽象是面向对象设计的核心思想，抽象的艺术在于抓住一个系统最核心的特性，这些特性是能够表述以后发生的任何事情的构件。虽然抽象是一个简单易懂的概念，但读者会惊讶地发现，在设计一些系统的时候，使用这种方法的频度几乎微乎其微！本章提供了一个极复杂系统的极简单的设计。我们的目的是要说明怎样把一个极复杂的问题更好地分解成一些最简单的组成部分，又如何建立并修改一个已经用模块化风格分解产生的系统。

为简化起见，我们只考虑系统最通用的特性。

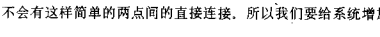
1.3 设计一个电讯系统

当您拿起话筒给别人打电话时，在您和另一方之间就建立了一个连接。简单说，这就是两点间的连接，您的目标就是要设计一个能获得这种连接的系统。

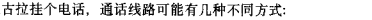
当然，实际生活中不会有这样简单的两点间的直接连接，所以我们要给系统增加一些复杂性。

例如，假设您要给尼古拉挂个电话，通话线路可能有几种不同方式：

- 您非常幸运地有一条直达热线接到尼古拉那里。在这种情况下，电话线路直接联接读者和尼古拉：

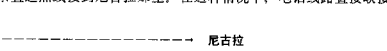


- 您不得不经过另外一个地点的线路才可最终与尼古拉相接通，即：



在这种情况下，您到太德那里有一条直达线，而太德到尼古拉那里又有一条直达线，为了将您和尼古拉连通，必须先经过太德。

- 如果太德的线路处于忙的状态又会怎样呢？那就不能通过它来与尼古拉相连，这就必须考虑其复杂性。读者的呼叫经由另外一条绕过太德的线路到达尼古拉那里，如：



- 现在，看看最复杂的情况。在现实生活中，象我们刚才谈到的这些不同的连接（您，太德，汤姆，尼古拉），实际上可认为是物理的硬连接的机器。用某种方式对这些机器编程，使之能接受某种类型的电话呼叫，再将呼叫导向目的地。这种逻辑被编入机器（从现在起，我们称这些机器为设备），以便能理解那些对它们的呼叫的类型。然而，如果这些呼叫类型不同，那么设备就不知道如何处理，因而拒绝这些呼叫。从技术上讲，不同类型的设备具有理解不同类型协议的能力，如果一个电话呼叫在逻辑上不遵守这个设备能理解的协议，则该呼叫就会遭到拒绝。

这样，在为某一呼叫进行路由选择的时候，必须鉴别一下要通过设备的性质，以确保该设备确实有能力处理该呼叫并为之选择路径。

1.4 电讯系统的再思考

如果读者再花点时间去重读上一段落，会发现在设计该系统的时候需要涉及以下几个问题：

- 怎样获得可能毗连或非毗连的两点间的连接？
- 怎样获知不同设备的不同行为特及这些设备能确定连接的不同类型？
- 如果在正常情况下选择的逻辑连接不可用，那么如何将该呼叫导向另外的设备？

下一步的任务就要把眼光集中在为实现本系统而要明确表达的各种不同类型对象的基本特性上。我们试图简化上述问题的复杂性，简化结果如下：

- 两点间最简单的连接就是彼此间毗连的两点的连接，它们之间就可以直通。因而要确定的第一个对象是获得这种连接的本质。
- 所有的设备的都要求最终能实现一种主要功能：将一个呼叫向前传递给下一点。因而要确定的另一个对象是获得一个基本设备这一特征的本质。
- 反呼叫从一个设备导向另一个设备之前，第一个设备必须向下一个设备发送信息以询问将呼叫发给下一个点是否可行。下一个设备也必须给这个设备回送一个适当的应答信息。如果该设备可行，发送呼叫的设备就将这个呼叫传递给它。如果不可行，那么发送呼叫的设备就要试图找到另一条路径。这样，就可以归纳成一种机制，在这种机制下，上面所确定的对象彼此间能进行通讯。

刚才所罗列的简单概括可以构成整个系统的骨架。一旦一个系统设计得能够满足所列的要求，就有足够能力处理以后的复杂问题，在下一小节再详述。

1.5　在基本抽象基础上建立系统

现在总结一下分析结果。在上一小节里，我们列举了两种对象的功能。在C++ 中，对象的功能用一种称作类（Class）的形式来表达。用来构成其它类的类叫作基类（Base Class）。在上一小节中，我们认定了两种用来确定该系统骨架的基类。这两种基类将构成所有由它们派生出来的其它类。对这两种基类规定如下：

- 获取在毗邻两点间传递呼叫所需规则的类，称之为 Class A。
- 获取用作呼叫路由选择的最基本设备的功能的类，称之为 Class B。

此外，我们还认定了这些类的对象彼此间进行通讯的机制，通过这些机制，完成呼叫的完整路程。

有了这些基本构成，我们就可以使系统更复杂些，并看到所设计的这些基类怎样成功地处理这些情况。下面是该系统所要求的另外一些特性。

- 一个呼叫可以在非毗连的两点间漫游，这是经过了其它连接才链在一起的。

我们从 Class A 派生出一个类来完成这种功能。这个派生类继承了明确表达两点间一对一连接所要求的规则。但是，它包含有另外的方法，此方法中有明确表达更复杂的连接所要求的规则。

- 各种不同类型的设备都可以加入到该系统中，用不同的方式来为一个呼叫选择路径。

然而，为呼叫进行路由选择的基本机制总是相同的。

我们通过从Class B中派生出一些类的办法来完成该特性。这些派生类将继承最基本设备的功能。然而，每一个派生类都应该增加一些独特的属性或方法来将该设备与其它设备区分开。类之间彼此通讯的机制也就被确定下来。

这就应该是在设计一个大系统时所用的基本思维模式。本系统将能足够灵活地处理完整的路由选择和改变设备的属性所需的变化规则，这些变化将简单地通过修改派生类的应用代码来实现。

每个设备的方法都封装（ encapsulate ）在它所属的类中。如果系统中加入了新的设备，而该设备没有明确描述的属性，那么它可以利用基类中说明的功能。这种特性通过虚函数（ virtual function ）来表达。一旦这个新设备的属性通过多态性（ polymorphism ）被确定下来，那么这些虚函数就可被废弃（ override ）。

现在我们强调一下几个关键的短语和词汇，它们是：对象，类，基类，派生类，虚函数以及多态性。

1.6 结 论

尽管这里对一个复杂系统作了极其简单的表达，但这正是我们所要阐明的观点。尽可能地将复杂问题简化。在初始设计阶段把精力集中在主要的特性和需求上，而不是陷入到不重要的细节中。一定要从系统的最根本处人手。一旦当读者能并用这种方法分析并获得所需系统，读者会觉得，在系统的强大的基上构造系统是一件多么容易的事。系统的复杂性往往掩盖了潜在的，用经建立和实现系统时的简单性。需要设计者花时间努力地思考如何反复杂问题分解成最简单的形式，一旦能这样做，就会觉得其它什么事情都可按部就班了。

另一方面，如果设计中有缺陷，那么就会感到，试图在这样的设计上构造系统会混乱不堪。一个好的系统总是遵循正确的规则；而一个设计得不好的系统最终将混乱得几乎无法收拾，通常的结果是完全重来。

尽管本章是预备性的，建议读者在学习第二章之前能把第一章重读一遍（最好要慢一些）。我们在第二章讨论语言本身。

第二章　C++ 及编程基础

2.1　简　介

在编写程序之前，必须懂得程序如何存储以及如何在计算机上运行。本章将描述什么是程序，程序如何被计算机处理和执行，然后，我们将编写并执行一个简短的C++ 程序。

2.2　什么是程序设计?

■ **程序是用能够被计算机理解的一种语言编写的语句的集合。**

程序的正确执行通常会通过计算机得到某种输出结果。

C++ 程序包含对预先写好的代码的引用。这些预先写好的代码以库（libraries）的形式存储，可通过特殊的语句把这些库包含到C++ 代码中。

■ **程序要经过编译器、链接器和装载器的处理。**

这些个别的实体只不过是存储在计算机中的不同类型的软件，是专门用来处理程序代码的。所有这些处理的最终结果是产生能被计算机所认识并能被计算机运行的代码。

程序设计从开始到完成所需的步骤如下:

第一步　程序员进行计算机登录。

第二步　程序员用喜爱的系统的编辑器编辑源程序。

第三步　程序员把编辑好的源程序存入文件中，这个文件叫做源代码文件。

第四步　程序员用适当的命令指示计算机编译源代码文件。

用来编译程序的命令因编程语言的不同而不同。例如，用来编译 C++程序的命令与用来编译 BASIC 程序的命令不同。如果使用的是菜单驱动系统，那么编译和运行程序的处理只不过是在适当的菜单选项上敲击一下。

我们再继续前面的步骤，这就是编译开始后将会发生什么事情。

第五步　编译器将源程序代码翻译成汇编语言格式，然后传给计算机汇编器。

第六步　汇编器将汇编语言代码翻译成可重定位的目标码，并将这些目标码传送给链接器。

第七步　链接器将程序引用的所有支持例程连接在一起。这些例程存在于运行库中或早已存在于被程序所引用的预编译程序中，然后链接器产生源代码的最终版本，我们把它叫做可执行代码。该代码被传给执行程序的装载器。

第八步　运行程序。

虽然，整个处理过程显得很长，但是这些步骤对于程序员来说大多数是透明的。需要程序员做的只是以下几个步骤:

第一步　产生一个包含源代码的文件。

第二步　输入编译命令，得到编译结果。

第三步　运行程序。

下面将要讨论源代码文件命名的约定，在保存源代码文件时，要遵守这些约定。

2.3 源代码文件的命名约定

不同的操作系统有各自不同的C++ 程序源文件命名规则，应参照系统手册来决定。一般来讲，C++ 程序用“.C”作后缀（大写C；如果后缀是小写字母 c，编译器将认为该程序是一个C 语言程序）。

2.4 编 译 器

本书中所有的程序都是在 IBM PC 机的 MS Windows 环境下用 Turbo C++ 编译器（V3.1版）编译并运行通过的。

本书中所用的程序都是可移植的。实际上，可移植性也正是使C++ 语言颇具实力的原因之一。这些程序可以用任何一种 C++ 的编译器编译并运行，可以在 Windows 环境下，也可以不在 Windows 环境下进行。

Borland C++ 的编译器用 .c 后缀来识别 C语言程序，用 .cpp 后缀来识别 C++ 程序。在本书中采用同样的约定，若使用其它的操作系统或编译器，那么只需参阅相关资料中的适用说明。一般说来，程序能象在本系统上一样正确地编译并运行。代码和输出都会是相同的，尽管出错信息的措辞会因使用不同的编译器而变化，但这些信息的含义是相同的。

2.5 程序的产生和执行

要运行一个程序，必须先用程序员所喜爱（或不甚喜爱）的编译器将程序敲进去，并保存在文件中。一旦文件建立之后，就可以按照正在使用的特定工具的要求去编译、链接和运行。有些工具允许在操作系统命令行提示符下键入命令来完成上述步骤，而另一些系统则允许通过菜单和对话框的形式来完成整个过程。本书所用的系统，全部是由菜单驱动的，使用户可在一个集成的环境里编辑、编译并运行程序。对于其它类型的系统只需根据系统资料提供的说明来编译并运行。

2.6 Borland C++ 3.1 编译器

下面的章节将逐步讲述 Borland 公司的 Turbo C++ V3.1 编译器提供的集成开发环境中的编辑、编译和运行过程。读者可以通过定位并单击鼠标左边按钮来选择选项，或简单地用键盘上的方向键移动高亮光条到相关选项并按 Enter 键。本书采用后一种方法。选项也可以通过与之相关联的热键或敲击标明该命令的字母来选择。

集成开发环境（IDE）是 Borland 的程序员开发平台。对其作简短的介绍以开始我们的讨论。

2.7 Borland 的程序管理器

在使用编译器之前，必须将它安装到 PC 上，必须执行编译器提供的INSTALL程序将所有可用文件装入计算机。该程序要求 PC 机中已经装入了 Microsoft Windows，当启动 Windows 后，进入程序管理器。安装成功后，先退出，再重新进入 Windows 环境，Borland C++ 将自动建立一个新的程序管理器组，组中包含与该平台相关联的一组图标。在以 Borland C++ 3.1 为标题的窗口中共有 20 个图标，如图 2.1 所示。

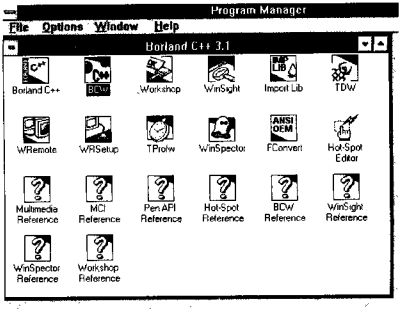


图 2.1 Borland C++ 3.1 图标组

我们只探讨用于 Borland C++ for Windows 编译器的菜单选项（左起第二个图标）。我们鼓励读者探索用于其余功能的图标。详细的说明，请参阅其它 C++ 资料。

在图 2.1的窗口中，将光标移到标有BCW的图标（最上一行左起第二个图标）上，双击鼠标左按钮，就进入了 IDE 平台。这个环境包括窗口、菜单、对话框、加速条、卷滚条、编辑状态行。

2.8 IDE 平 台

在IDE平台的顶部是菜单选项，后面将详细讨论。在菜单选项下面是加速条。加速条包含一组图标，这些图标为频繁使用的任务提供捷径。将鼠标箭头定位在其中任何一个图标上，都会在状态域中显示出该图标功能的简短描述。当鼠标箭头定位在？（帮助）图标（左起第一个）时，可见到图 2.2 的信息显示（Display Context Sensitive Help）。

窗口底部的箭头钮允许左右滚动，窗口右边的箭头钮允许上下滚动。

窗口本身可极大化（覆盖终端 屏幕的整个区域）；或极小化（变成一个图标），当单击窗口左上角的按钮时，从显示出的菜单中选择相关的项目即可实现极大化和极小化。

2.8.1 编 辑 窗 口

再次观察编辑窗口时，就会注意到窗口顶部的标题行写着：

C: \ borlandc\bin\noname00.cpp。

编译器自动打开一个编辑会话窗口，并把要创建的 C++ 文件取名为noname00.cpp。可以使用这个默认文件名，也可以通过 File 菜单选项改变该文件名。这一点我们后面会解释。

关于基础就讲这些。下面让我们浏览一下可显示各种选项的菜单条。

将鼠标箭头放在 File 选项（左起第一个菜单选项）上，单击它（或通过键盘上的方向键移动光标定位并敲回车键），File菜单条将显示出来，如图 2.3 所示。

可以用下面三种方式之一选择菜单条中的任一选项。

- **把鼠标移到要选择的菜单项上，然后单击鼠标。**
- **如果菜单项中含有带下划线的字母，则可键入该字母。**

例如要选 File 菜单条中的 New 菜单项，键入 N 即可。

- **如果存在热键（hotkey），可键入菜单项的热键。**

例如对于File菜单条中的Exit，可按住 Alt 键的同时敲 F4 键。

后面带有省略号（…）的每个菜单命令隐含着一个对话框，如果选择该命令，将激活对话框。该对话框叠加在当前窗口上。例如，如果读者选择 File 菜单条中的 Open 命令，Open a File（打开一个文件）对话框就显示出来。如图2.4 所示。

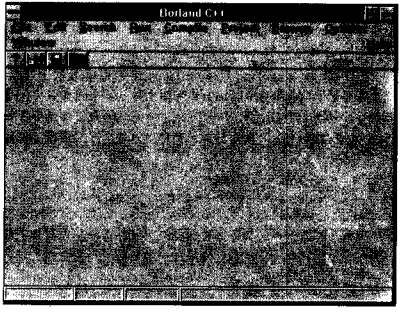


图2.2 Borland C++ 3.1编译器的 IDE 环境

对每个打开的对话框，键入适当的信息，然后单击 OK 按钮使系统接受所敲入的信息，或者单击 Cancel 按钮使系统废弃所键入的信息。单击 Help 按钮可以得到本窗口相关的帮助信息。

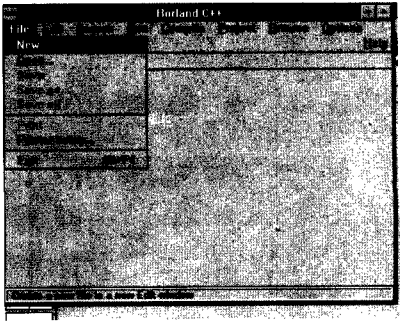


图2.3 IDE中的 File （文件）菜单条

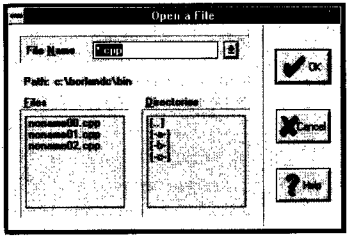


图2.4　Open a File（打开一个文件）对话框

图2.5～图2.13显示了所有可用菜单项的菜单条。

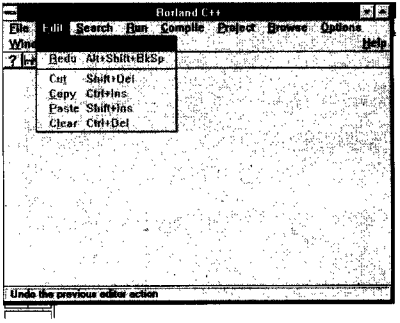


图2.5 IDE中的 Edit 菜单条

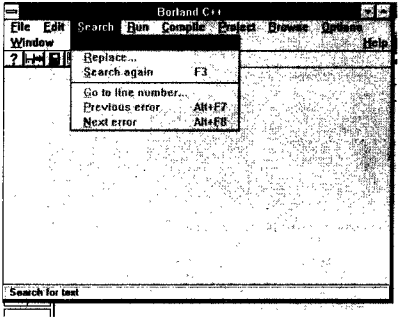


图2.6 IDE中的 Search 菜单条

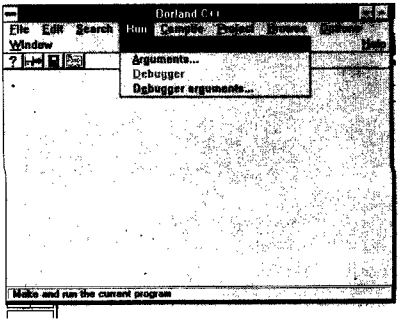


图2.7　IDE中的 Run 菜单条

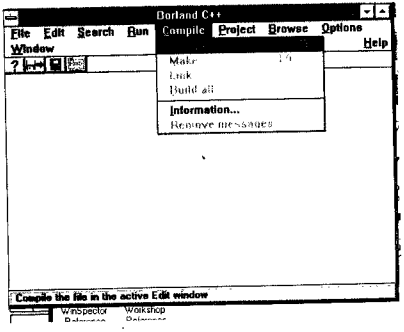


图2.8　IDE中的 Compile 菜单条