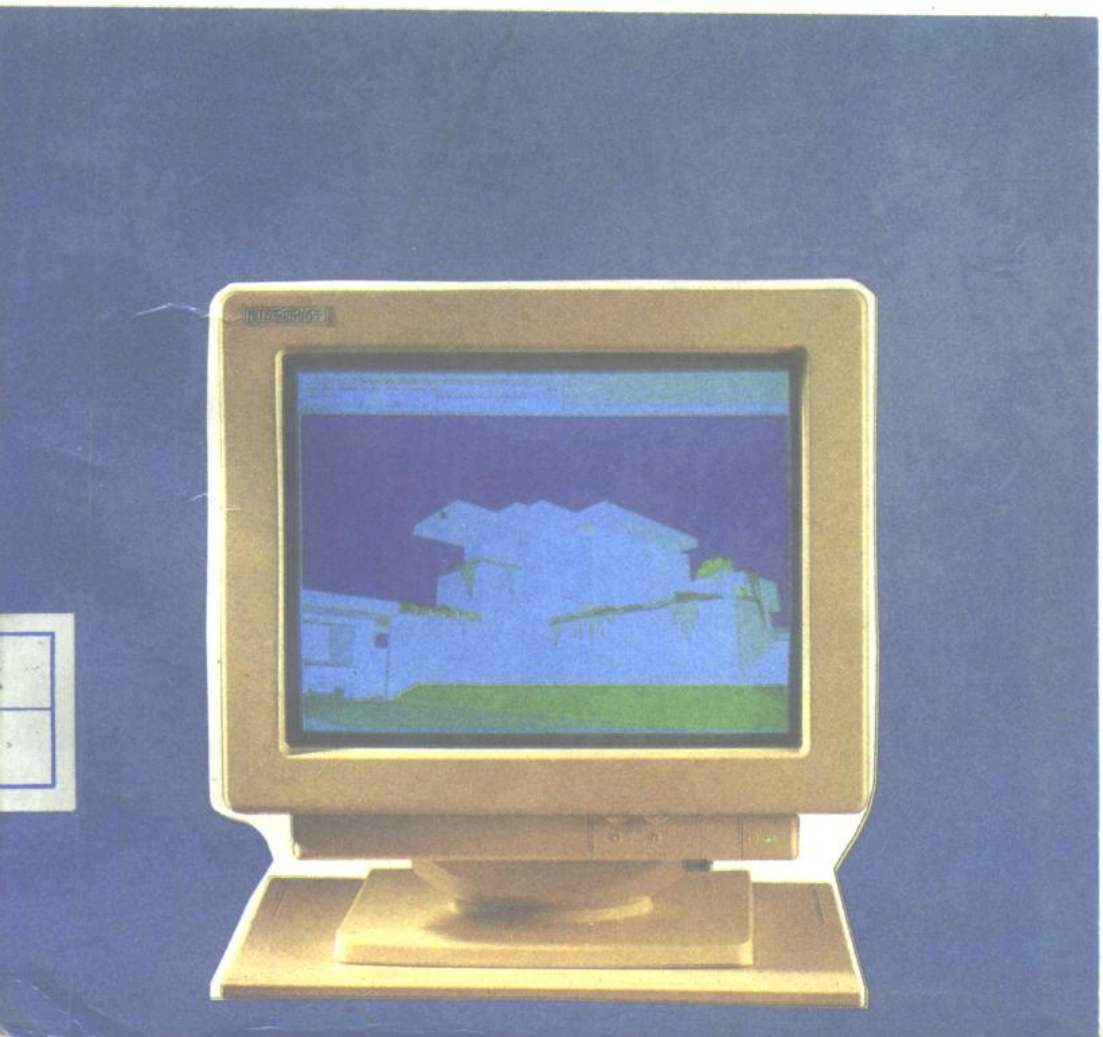
C++技术学习与指导

广聚　　玉萍　编
希望　　审校



北京希望电脑公司

C++ 技术学习与指导

广聚　　玉萍　编
希望　　审校

北京希望电脑公司

前　言

C++ 已越来越成为广受人们欢迎的一种程序设计语言。它继承了 C 语言的传统，且增加了一整套全新的功能，包括：

数据抽象，

继承和多态的支持，

名字规整，

类型安全联接，

内联函数和函数重载等等。

可以这样说，C++ 是已有的最强大的通用程序设计语言。它把 C 语言推进到面向目标程序设计的世界。

本书除了面对有 C 经验的程序员之外，没有 C 语言基础的程序员也照样可以看懂这本书。本书直接从 C++ 语言的基础入手，直接将读者引入到 C++ 的环境中。

本书详细地讲解了在何时使用 OOP 技术，在何时不使用 OOP 技术，在何时将二者兼容共存。同时还列出了大量的已编译通过的示例，供读者参考。

目　　录

第 0 章　介绍

在这里，可以坦率地说，这是一本概括很全的书. 此书从一开始，就直接引入 C++ 的概念. 不象其它一些 C++ 参考书那样，先解释 C 语言，然后，再引入 C++. 那样做虽然对读者(特别是熟悉 C 语言的读者)易学，但是，C++ 和 C 是两种不同的语言，C++ 绝不是 C 语言的简单升级，二者的设计思想完全不同，所以这样做，不易使读者摆脱 C 语在言的阴影.

在此书中，有大量的示例，供读者参考和学习. 从第一章到第六章给出了关于 C++ 的大量指导，你可以学到写 C++ 程序所需要的常规和目标化技术. 你还可以学到关于类，单继承和多继承，友元，操作符重载，流和一些其它特性. 这些特性激发了大量有经验的程序员和初学者使用和学习 C++. 在第七章中，你可以发现至少有 150 个函数参考. 这些函数都存放在 C++ 库中. 看完此书后，你就可以使用这些函数写自己的 C++ 程序.

在此还解释和介绍了如何安装 C++ 编译器，以及所必需的最少设备. 从第二章开始，你就可以编译你自己的 C++ 程序.

0.1　所需设备

下面列出了所必需的和可选的硬件和软件:

0.1.1　所必需硬件

1). IBM PC, XT, AT, PS／2, 或百分之百的兼容系统.

2). 384K——512K 有效 RAM.

3). 硬盘驱动器(建议).

4). 5 吋软盘驱动器(仅用于安装).

0.1.2　可选硬件

1). 打印机.

2). 彩显.

3). 对系统来说，如果没有硬盘驱动器，至少应有一个 1.2M 或 1.44M 的软盘驱动器，建议使用硬盘驱动器.

0.1.3　所必需软件

1). MSDOS 或 PCDOS 2.0 以上版本(大多数程序需要 3.0 或更高版本).

2). 所有其它必需软件包含在此书中.

0.1.4　可选软件

1). 你所喜爱的文本编辑器. 如果你没有较好的编辑器，请使用本书中所推荐的编辑器 VDE.

2). Zortech C++ 2.1. 如果你已拥有 Zortech 完整的开发系统，你就可以直

接使用其编译你的程序. 当然, 你可没必要非使用此开发系统编译和运行此书中所包含的示例.

0.2 安装 Zortech C++ 编译器

你应保证你的机器有 3M 有效硬盘空间. 如果你想安装在高容量软盘上, 那么, 你先格式化一张软盘, 然后把标号为 #1 的磁盘插入到驱动器中. 键入下面两个命令, 每一命令后打回车.

```
a:
install
```

根据屏幕提示, 一直操作下去. 如果想停止 INSTALL, 你可以在任何时候打 <ESC> 命令. 如果想重新安装此软件, 只需重复上述命令即可.

在安装过程中, 你需要选择驱动器. 但最好选择硬盘驱动器. 你还需要提供一个目录名, 其缺省名为 \LCPP. 你可将其改为任意合法的名字. 但你不能把此软件安装在根目录上. 所以, 你最好打回车, 保证原来的缺省目录名.

对于软盘安装, 你可以只用(\)来说明一个路径名. 这样将节省软盘空间, 且将编译器, 编辑器, 联接程序安装在软盘的根目录中. 但注意, 这种情况只可用于高密软盘, 决不可以在硬盘安装时这样做.

安装结束后, INSTALL 将运行 READ.EXE 程序, 此程序用于显示名字为 README.TXT 的文本文件. 你也可以在 DOS 提示符下运行 READ. 此文件在第 #1 片盘上, 未采用压缩格式存储. 然后, 在当前目录中使用 READ.EXE, 键入 read, 选择 README>TXT 或其它文本文件. 使用 <Cursor> 和 <Page> 键进行浏览. 打 <ESC> 返回到目录显示或 DOS (如果你在 DOS 提示符下键入 read readme.txt, 读取一个文件, READ 不列出目录).

0.2.1 硬盘安装

当把此系统安装在硬盘上, 且读取了 READ.TXT 文件之后, C:\LCPP 为当前目录, 如果不是, 打 <ESC> 返回 DOS. 然后键入 C: 和 CD\LCPP. 如果你使用了其它路径名, 或安装在别的盘上, 使用它们代替 C\LCPP.

使用 DIR 查看目录内容, 这样, 除此文件之外, 还可看到三个文件:

```
LCPP.EXE
LIB.EXE
VDE.EXE
```

这三个文件是压缩文件. 它们是把一些文件以压缩格式存储在一起. 为将这三个文件还原, 执行下列步骤:

```
lcpp
lib
vde
```

如果你有自己的文本编辑器, 不需要使用 VDE 编辑器, 可不键入第三条命令. 但你可以把 VDE.EXE 考备到别的盘上保存起来. VDE 是一个功能非常强的文本编辑器和

字处理器.

还原后，你可以删除这三个源文件，因为你已不需要它们. 请键入:

```
del lcpp.exe
del lib.exe
del vde.exe
```

0.2.2 大容量软盘安装

在把系统安装到 1.2 或 1.44M 软盘上后，你需要使用一特别命令来还原那些被压缩为三个文件的文件. 把安装好的盘插入到 A 驱动器中，把第二张已格式化好的空盘插入到 B: 盘中. 然后键入下列命令，每命令后打回车:

```
a:
lcpp / cb:\
lib / cb:\
vde / cb:\
```

这些命令将把上述三个文件还原. 你可把还原后的文件考备到软盘上，编译和编辑其它程序. 从所安装的盘片上，删去 LCPP.EXE, LIB.EXE, VDE.EXE 以腾出空间.

0.2.3 360K 软盘安装

首先，不赞成你使用两张 360K 软盘运行 C++ 编译器. 如果你决定这样做，下列是一些如何使用的建议:

先把这些文件安装在一个硬盘上，或一个大容量的软盘上，格式化好三张软盘，且分别标上 LINKER, COMPILER 和 EDITER. 考备下列文件到 LINKER 盘上:

```
blink.exe
pls.exe
zls.exe
```

考备下列文件到 COMPILER 盘上:

```
*.h (加上 page.h 和 cmm.h)
*.hpp
ztc.com
ztc2.exe
ztcpp1.exe
```

考备下列文件到 EDITER 盘上:

```
examples.vdk
vde.com
vde.doc
vde154.upd
vinst.com
vinst.doc
wp.vdf
```

ws4.vdf

你还需要两张磁盘来存放所列出的源代码. 从 SOURCE, ANSWER, LIB 目录中把所有文件和子目录考备到盘上. 现在你就可以插入相应磁盘编译和联接你的程序. 显然, 这样做是非常麻烦的, 如果可能的话, 建议你使用硬盘. 如果没有硬盘, 将不能编译比较大的程序.

0.2.4 测试所做的安装

在安装和还原所有文件后, 你还需要改变你的 PATH 语句, 设置两个环境变量. 这样做, 你就可以把编译的文件存入任何子目录中. 且在联接和编译过程中, 编译器和联接程序可以找到各个文件.

在根目录下, 编辑或建立 AUTOEXEC.BAT 文件, 插入如下 PATH 语句:

```
path = c:\dos; c:\lcpp
```

PATH 语句可以列出其它路径名, 用分号隔开. 还需要把下列两行加到 AUTOEXEC.BAT 文件中:

```
set include = c:\lcpp\include; c:\lcpp\lib
set lib = c:\lcpp
```

这些命令建立两个环境变量: INCLUDE 和 LIB. 它们告诉编译器和联接程序在何处找
到各种文件. 注意, \LIB 目录跟在 INCLUDE 变量之后, 存储在 \LIB 中的文件为类库.
(class library).

保存所修改 AUTOEXEC.BAT 文件, 重新启动计算机. 如果出现 out of environment space
错误, 则你还必须在根目录下修改或建立 CONFIG.SYS 文件. 把下列命令加入到 CONFIG. SYS 命令中:

```
shell = command.com / E : 512 / p
```

0.3 使用 VED 编辑器

VDE 编辑器是为了那些没有使用其它编辑器的用户准备的. 你可以使用 VDE 修改, 浏览源代码, 或建立新文件.

使用编辑器之前, 先进入 C:\LCPP 目录, 打入命令 vinst. 根据屏幕提示, 把软件装入到你的机器中. 不管你改变不改变其配置, 打 S 保存其配置于盘中. 然后, 返回 DOS 状态, 再把目录改变为保存你要修改的文件的子目录 (如 C:\LCPP\SOURCE\C01), 打入如下命令:

```
vde welcome.cpp / n
```

文件 welcome 可以是一个已存文件, 也可以是一个准备建立的新文件. / n 选项告诉 VDE 以 ASCII 码格式读写此文件. 如果你在保存文件时, 没有用 / n 选项, 你将不能编译此文件. 如果你仅想用 VDE 编辑一个程序, 你可以使用 vinst.com 实用程序改变其缺省文件类型为非文本格式. 关于如何使用及处理 VDE 的更详细的资料, 请参阅文本

文件 VDE.DOC 和 VINST.DOC. 如果要打印, 则键入如下命令:

```
type vinst.doc > prn
type vde.doc > prn
```

注: 不要使用 READ 程序来检查 VDE 的 DOC 文件, 这些文件对于 READ 来说太大, 不能处理.

如果你使用过 wordstar, 或 Borland International 公司的 Turbo Pascal, Turbo C 或 SideKick, 你将会体会到, 使用 VDE 非常容易, 其中的命令与它们完全相似. 如果要退出 VDE, 请键入 <Ctrl>—KQ. 如果改变了当前文件, 打 Y 保存你所做修改, 打 N 恢复原状. 如果想保存所做改变, 且不退出, 可键入 <Ctrl>—KS. 如果要保存当前文件且退回 DOS, 打 <Ctrl>—KX.

0.4 程序编译

测试你所做的安装是否正确, 最好的方法是试着编一些程序. 下面是编译的正确过程, 你还可以用其编译后面章节中的示例:

. 在 DOS 状态下, 保证你的 PATH, INCLUDE, LIB 变量被正确设置. 在 DOS 提示下键入 set, 且查看变量列表, 你将发现 C:\LCPP 目录在 PATH 中, 在其中还有前面讲到的另两个目录.

. 键入 ztc welcome, 用于编译和联接 WELCOME.CPP 程序, 此程序在第一章中, 你将在屏幕上看到如下显示:

```
ztcpp1 -oztc___APC.tmp welcome
Zortech C++ Demo Compiler
ztc2 ztc___APC.tmp -owelcome.obj
BLINK welcome/noi;
```

. 这样, 在当前目录中完成了代码文件 WELCOME.EXE. 键入 welcome 运行此程序.

. 键入相似的 ztc 命令. 编译本书中的其它程序, 其中有些复合程序需要一些特殊方法来处理.

. 你如果接收到一个错误信息, 特别是: ERROR: 'ztc1 not found, 那么, 请把正确的 .cpp 文件放到当前目录中(ztc1 是 zortech 的 C 编译器, 有时, ztc 也运行 C 编译器, 由于没有安装 ztc1, 所以产生错误信息, 但这不是大问题).

第 1 章　了解 C++

这一章将帮助你了解：什么是 C++？如何方便地使用它. 如果你知道 Pascal, C 更好. 你可以浏览此章，此章会告诉你 C++ 与 Pascal 或 C 等语言的区别，以及相似之处. 在此假设你只是一个语言的初学者，只懂得位和字节是什么，如何键入 DOS 命令，和如何运行程序.

首先从所有程序所必需的元素讲起，然后讲流，常量，变量，输入和输出，操作符. 几乎所有的 C++ 程序都或多或少地使用这些基本元素构件，所以花费一些时间对其进行了解，在以后将是非常有帮助的.

1.1　C++ 构成

所有的 C++ 程序都是相关联的，即它们都有相同的骨架. 了解这些骨架结构的一个好办法是编写一些小程序，将它们进行编译，运行. 如程序 welcome.cpp，在 DOS 提示符下，键入 ztc welcome，建立 WELCOME.EXE 文件，然后运行它.

列表 1.1　WELCOME.CPP

```
1: #include <stream.hpp>
2:
3: main()
4: {
5:     cout << "Welcome to C++ programming!\n";
6: }
```

跳过第一行，看 3--6 行，它们构成了这个程序的内容. 把第 5 行去掉，程序则成为

```
main()
{
}
```

这称为函数. 它是 C++ 中最重要的构件. 在这里，只有一个函数，名字为 main. 在其它 C++ 程序中，可能有几十个，几百个甚至更多个函数，各具有各自的名字，但具有相同的格式. 无论一个程序有多少个函数，它必须有且只有一个 main 函数. 当 C++ 程序运行时，总是从 main 处开始.

注：不要把此处的术语"函数"和数学中的函数相混淆. 在 C++ 中的函数可以执行一些数学运算，但也可以不这样. 在 C++ 中，函数是执行一个或多个动作的构件组合. 正象你所学到的那样，一个函数的动作完全取决于你所做的描述.

函数名后面的空花括号告诉编译器：此函数从外面不接受信息. 后面你将学到如何在花括号内表示信息，允许函数处理所输入的信息. 例如，你可以传给 main 一个命令行选项或一个文件名，以被此程序使用. 函数名后面的左右括号和其花括号构成了函数实体.

上列表 1 中, main 函数的实体所包含的程序为:

```
{
    cout << "Welcome to C++ programming !\n";
}
```

花括号的作用是告诉编译器: 其中的行属于 main 函数, 这些行称为语句. 通常来说, 一条语句在一个程序中表示程序运行时的一个特定动作. 而其它的如: 说明, 定义, 表达式或结构, 用于告诉编译器在编译时所做的一些事情. 典型地说: 说明告诉编译器一些信息, 如一个新数据类型的格式等等. 定义为某值在内存中申请一个空间, 如一个以前说明的数据类型的变量. 表达式如 1 + m 用于得到一个单一值. 其它的结构或改变编译器的工作, 或根据不同的情况在程序的不同节段上进行选择性的编译.

不要担心如何雇记住这些术语, 即使是作者和有经验的程序员也经常搞混说明和定义. 在此处, 你所要知道的是 C++ 花括号的目的: 用于括起一个或多个语句, 表达式, 说明或定义, 将它们作为一个单位. 花括号和其中内容一般称为块. 编译时作为一个整体处理.

1.1.1 流(stream)

当你运行列表 1.1 程序时, 你将看到, 它在屏幕上显示信息: Welcome to C++ programming! 有很多种方法实现这一功能, 但在 C++ 中, 最简便的方法是使用输出流 (output stream). 请看列表 1.1 中的输出流语句:

```
cout << "Welcome to C++ programming!\n";
```

首先是输出流目标的名字 cout, 它是"字符输出"(character output)的缩写. (在此提示一句: 最好把 cout 发音为 "see out", 不是 "kout"). 目标 cout 是一个输出指定点, 你把程序显示或打印的内容送到此处.

双角符号 << 表示流的输出符号. 它是一个由两个字符 '<' 组成的符号. 此符号指向 cout. 暗示以此方向从左到右把内容流向目标. 程序流的源是字符串 " Welcome to C++ programming!\n". 流的意思来源于水的流向.

双引号告诉编译文本的内容范围, 且不把其中内容作为语句处理. \n 符号也由两个字符组成, 称为换行符. 在一个字符串中插入 \n (不必必须放在最后)可使程序在屏幕或打印机上从一新行开始.

学习输出流的一个好方法是亲自做几次. 编一些小程序, 加入各种输出行. 如把下列语句加在 main 的实体中:

```
cout << "Welcome";
cout << "to";
cout << "C++ programming !\n";
```

上述三行语句和列表 1.1 产生相同的结果. 因为头两行未用 \n 换行符结束. 程序把它们显示在相同的行上. 你可试着在每一行中加入 \n 换行符. 看看发生什么结果.

另一种产生相同结果的方法是使用一个单一的输出流, 但使用多个输出部分:

```
cout << "Welcome " << "to\n"
     << "C++ programming!\n";
```

仔细检查上述示例, 运行之且观查其结果. 在第一个例子中, 有三条语句, 每一个用