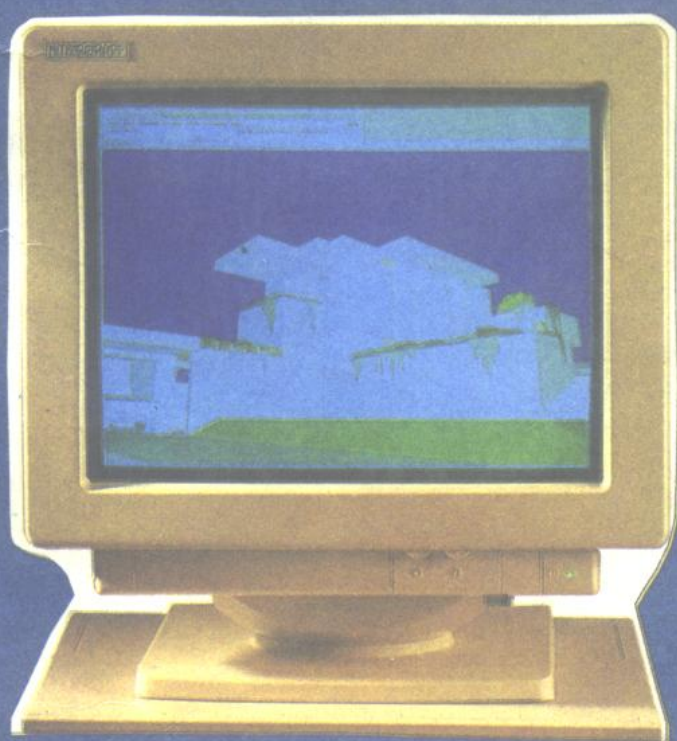




# C<sup>++</sup> 技术学习与指导

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。

JS305/28

# 目 录

|       |                          |    |
|-------|--------------------------|----|
| 第 0 章 | 介绍 .....                 | 1  |
| 0.1   | 所需设备 .....               | 1  |
| 0.1.1 | 所必需硬件 .....              | 1  |
| 0.1.2 | 可选硬件 .....               | 1  |
| 0.1.3 | 所必需软件 .....              | 1  |
| 0.1.4 | 可选软件 .....               | 1  |
| 0.2   | 安装 Zortech C++ 编译器 ..... | 2  |
| 0.2.1 | 硬盘安装 .....               | 2  |
| 0.2.2 | 大容量软盘安装 .....            | 3  |
| 0.2.3 | 360K 软盘安装 .....          | 3  |
| 0.2.4 | 测试所做安装 .....             | 4  |
| 0.3   | 使用 VDE 编辑器 .....         | 4  |
| 0.4   | 程序编译 .....               | 5  |
| 第 1 章 | 了解 C++ .....             | 6  |
| 1.1   | C++ 的构成 .....            | 6  |
| 1.1.1 | 流 .....                  | 7  |
| 1.1.2 | 关于分号 .....               | 8  |
| 1.1.3 | 关于注释 .....               | 9  |
| 1.1.4 | 标识符 .....                | 11 |
| 1.1.5 | 大小写问题 .....              | 11 |
| 1.1.6 | 关键字 .....                | 11 |
| 1.1.7 | 标点符号 .....               | 12 |
| 1.1.8 | 分隔符 .....                | 12 |
| 1.1.9 | 头文件 .....                | 12 |
| 1.2   | 变量 .....                 | 13 |
| 1.2.1 | 用定义初始化变量 .....           | 14 |
| 1.2.2 | 用赋值语句对变量进行初始化 .....      | 15 |
| 1.2.3 | 关于作用域的范围 .....           | 16 |

|              |                               |           |
|--------------|-------------------------------|-----------|
| 1.2.4        | 初始化全局变量 .....                 | 18        |
| 1.3          | 输入和输出 .....                   | 19        |
| 1.3.1        | 输出流 .....                     | 20        |
| 1.3.2        | 旧类型输出 .....                   | 21        |
| 1.3.3        | 格式化输出 .....                   | 21        |
| 1.3.4        | 流输入 .....                     | 25        |
| 1.3.5        | 旧类型输入 .....                   | 26        |
| 1.3.6        | 格式化输入 .....                   | 27        |
| 1.3.7        | 一个简单的十进制——十六进制——八进制转换程序 ..... | 28        |
| 1.4          | 常量 .....                      | 28        |
| 1.4.1        | 常量类型 .....                    | 29        |
| 1.4.2        | 文字常量 .....                    | 29        |
| 1.4.3        | 定义常量 .....                    | 32        |
| 1.4.4        | 说明常量 .....                    | 33        |
| 1.4.5        | 枚举常量 .....                    | 35        |
| 1.4.6        | 对枚举元素赋值 .....                 | 37        |
| 1.5          | 操作符 .....                     | 38        |
| 1.5.1        | 操作符和优先级 .....                 | 38        |
| 1.5.2        | 递增和递减操作符 .....                | 38        |
| 1.5.3        | 表达式 .....                     | 40        |
| 1.6          | 问题与练习 .....                   | 41        |
| <b>第 2 章</b> | <b>语句处理和结构建立 .....</b>        | <b>43</b> |
| 2.1          | 高级操作符 .....                   | 43        |
| 2.1.1        | 关系操作符 .....                   | 43        |
| 2.1.2        | if/else 语句 .....              | 45        |
| 2.1.3        | 逻辑操作符 .....                   | 46        |
| 2.1.4        | 逻辑非 .....                     | 48        |
| 2.1.5        | 位运算操作符 .....                  | 49        |
| 2.1.6        | 移位操作符 .....                   | 53        |
| 2.1.7        | 赋值合并——表达式的简写形式 .....          | 56        |
| 2.2          | 程序流向 .....                    | 56        |
| 2.2.1        | Exit .....                    | 56        |
| 2.2.2        | 减少复杂性 .....                   | 57        |
| 2.2.3        | 筛选程序和 while 语句 .....          | 59        |

|              |                           |           |
|--------------|---------------------------|-----------|
| 2.2.4        | 用 if/else 作判断 .....       | 61        |
| 2.2.5        | switch 语句 .....           | 63        |
| 2.2.6        | 使用 switch 语句 .....        | 66        |
| 2.2.7        | do/while 语句 .....         | 67        |
| 2.2.8        | C++ 中最常用的语句: for 循环 ..... | 69        |
| 2.2.9        | 多重 for 循环元素 .....         | 70        |
| 2.2.10       | 无休止循环 .....               | 71        |
| 2.2.11       | break 语句 .....            | 71        |
| 2.2.12       | continue 语句 .....         | 73        |
| 2.2.13       | goto 语句 .....             | 74        |
| 2.3          | 数据结构 .....                | 75        |
| 2.3.1        | 用于安全保护的结构 .....           | 75        |
| 2.3.2        | 嵌套结构 .....                | 81        |
| 2.3.3        | 联合: union .....           | 82        |
| 2.3.4        | 数组 .....                  | 84        |
| 2.3.5        | 使用指导——计算分数程序 .....        | 85        |
| 2.3.6        | 数组的初始化 .....              | 87        |
| 2.3.7        | 多维数组 .....                | 87        |
| 2.3.8        | 字符串 .....                 | 90        |
| 2.3.9        | 字符数组初始化 .....             | 91        |
| 2.3.10       | 数组和指针 .....               | 91        |
| 2.3.11       | 位域 .....                  | 92        |
| 2.3.12       | 计算变量的长度 .....             | 95        |
| 2.4          | 问题和练习 .....               | 97        |
| <b>第 3 章</b> | <b>函数</b> .....           | <b>98</b> |
| 3.1          | 保持其简单 .....               | 98        |
| 3.1.1        | 写自己的函数 .....              | 98        |
| 3.1.2        | 自顶向下设计 .....              | 100       |
| 3.2          | 函数和变量 .....               | 103       |
| 3.2.1        | 局部变量 .....                | 104       |
| 3.2.2        | 外部变量 .....                | 106       |
| 3.2.3        | 寄存器变量 .....               | 107       |
| 3.2.4        | 静态变量 .....                | 109       |
| 3.3          | 具有返回值的函数 .....            | 110       |
| 3.4          | 参数传递 .....                | 113       |

|              |                       |            |
|--------------|-----------------------|------------|
| 3.4.1        | 值传递和引用传递 .....        | 114        |
| 3.4.2        | 缺省参数 .....            | 120        |
| 3.5          | 递归 .....              | 124        |
| 3.6          | 内联函数 .....            | 128        |
| 3.7          | 总结 .....              | 130        |
| 3.8          | 问题和练习 .....           | 137        |
| <b>第 4 章</b> | <b>指针 .....</b>       | <b>139</b> |
| 4.1          | 指针说明 .....            | 139        |
| 4.1.1        | 指针间接寻址 .....          | 140        |
| 4.1.2        | 指针和类型检查 .....         | 141        |
| 4.1.3        | NULL 和 void 指针 .....  | 142        |
| 4.1.4        | 指针的使用——造型 .....       | 143        |
| 4.1.5        | 近程和远程指针 .....         | 146        |
| 4.1.6        | 指向系统位置的指针 .....       | 146        |
| 4.1.7        | far 指针和键盘 .....       | 147        |
| 4.2          | 内存管理 .....            | 151        |
| 4.2.1        | 结构指针 .....            | 153        |
| 4.2.2        | 内存溢出 .....            | 155        |
| 4.2.3        | 为什么使用动态变量 .....       | 155        |
| 4.2.4        | 建立动态链表 .....          | 157        |
| 4.2.5        | 动态数组 .....            | 164        |
| 4.2.6        | 用 malloc 保留内存空间 ..... | 164        |
| 4.3          | 指针作为函数参数 .....        | 170        |
| 4.4          | 函数指针 .....            | 174        |
| 4.5          | 指针和数组 .....           | 177        |
| 4.5.1        | 指针算法 .....            | 180        |
| 4.5.2        | 指针数组 .....            | 180        |
| 4.6          | 返回字符串指针的函数 .....      | 182        |
| 4.7          | 字符串函数 .....           | 183        |
| 4.7.1        | 普通字符串函数 .....         | 184        |
| 4.7.2        | 字符串联接 .....           | 185        |
| 4.7.3        | 字符串比较 .....           | 186        |
| 4.7.4        | 字符串搜索 .....           | 186        |
| 4.8          | 命令行参数 .....           | 187        |
| 4.8.1        | 字符参数 .....            | 189        |

|              |                                    |            |
|--------------|------------------------------------|------------|
| 4.8.2        | 数字参数 .....                         | 191        |
| 4.9          | 问题和练习 .....                        | 193        |
| <b>第 5 章</b> | <b>类目标 .....</b>                   | <b>195</b> |
| 5.1          | 引入类概念 .....                        | 196        |
| 5.1.1        | 建立类 .....                          | 197        |
| 5.1.2        | 内联成员函数 .....                       | 201        |
| 5.1.3        | 类是数据类型 .....                       | 203        |
| 5.1.4        | 构造程序 .....                         | 206        |
| 5.1.5        | 类用法 .....                          | 207        |
| 5.1.6        | 模块和类 .....                         | 210        |
| 5.2          | 编译电梯模拟程序 .....                     | 212        |
| 5.3          | 使用头文件 .....                        | 214        |
| 5.4          | person 类 .....                     | 216        |
| 5.4.1        | person 数据域 .....                   | 218        |
| 5.4.2        | perscollection 类 .....             | 218        |
| 5.4.3        | person 和 perscollection 成员函数 ..... | 218        |
| 5.5          | floor 类 .....                      | 227        |
| 5.5.1        | 实现 floor 类 .....                   | 229        |
| 5.6          | elevator 类 .....                   | 235        |
| 5.6.1        | elevator 类的实现 .....                | 237        |
| 5.7          | 引入继承 .....                         | 244        |
| 5.7.1        | 使用继承 .....                         | 244        |
| 5.8          | building 类 .....                   | 249        |
| 5.8.1        | 替换成员函数 .....                       | 250        |
| 5.8.2        | 实现 building 类 .....                | 251        |
| 5.9          | 完成此电梯模拟 .....                      | 254        |
| 5.10         | 问题和练习 .....                        | 257        |
| <b>第 6 章</b> | <b>提高你的 C++ 知识 .....</b>           | <b>259</b> |
| 6.1          | 好的朋友和邻居 .....                      | 259        |
| 6.1.1        | friend 类 .....                     | 259        |
| 6.1.2        | 相互友元类 .....                        | 263        |
| 6.1.3        | 友元函数——部分 1 .....                   | 263        |
| 6.1.4        | 友元函数——部分 2 .....                   | 265        |
| 6.2          | 函数重载 .....                         | 267        |

|        |                              |     |
|--------|------------------------------|-----|
| 6.2.1  | 名字规整 .....                   | 267 |
| 6.2.2  | overload 关键字 .....           | 269 |
| 6.2.3  | 常规函数重载 .....                 | 269 |
| 6.2.4  | 类成员函数重载 .....                | 270 |
| 6.3    | 操作符重载 .....                  | 270 |
| 6.3.1  | 重载操作符成员函数 .....              | 274 |
| 6.3.2  | 单目操作符重载 .....                | 276 |
| 6.3.3  | 对于操作符重载的一些要点 .....           | 277 |
| 6.4    | 数组下标重载 .....                 | 277 |
| 6.5    | 重载和用户定义类型转换 .....            | 280 |
| 6.6    | 赋值操作符重载 .....                | 283 |
| 6.6.1  | 考备类事例 .....                  | 284 |
| 6.6.2  | 成员对成员初始化 .....               | 285 |
| 6.6.3  | 指针域考备 .....                  | 287 |
| 6.6.4  | 考备构造程序 .....                 | 288 |
| 6.6.5  | 成员对成员赋值 .....                | 290 |
| 6.6.6  | 从一个考备构造程序中调用 operator= ..... | 291 |
| 6.6.7  | 考备导出类事例 .....                | 292 |
| 6.7    | 重载和内存管理 .....                | 293 |
| 6.7.1  | 对 this 指针的赋值 .....           | 293 |
| 6.7.2  | 对 new 重载 .....               | 295 |
| 6.7.3  | 对 delete 重载 .....            | 296 |
| 6.8    | 流重载 .....                    | 297 |
| 6.8.1  | 输出流重载 .....                  | 297 |
| 6.8.2  | 输入流重载 .....                  | 298 |
| 6.9    | 碎片 .....                     | 300 |
| 6.9.1  | 其它 I/O 流 .....               | 300 |
| 6.9.2  | 条件表达式 .....                  | 301 |
| 6.9.3  | 全局函数冲突的解决 .....              | 302 |
| 6.9.4  | 继承类的缺省状态 .....               | 303 |
| 6.9.5  | 成员函数指针 .....                 | 304 |
| 6.9.6  | 虚基类 .....                    | 307 |
| 6.10   | 总结 .....                     | 308 |
| 6.10.1 | C 和 C++ 之间的区别 .....          | 308 |
| 6.10.2 | 使用其它的 C++ 编译器 .....          | 309 |
| 6.11   | 问题与练习 .....                  | 309 |

|              |                       |            |
|--------------|-----------------------|------------|
| <b>第 7 章</b> | <b>库函数 .....</b>      | <b>311</b> |
| 7.1          | 如何使用此参考 .....         | 311        |
| 7.2          | 函数参考 .....            | 312        |
| <b>附录 A:</b> | <b>保留关键字 .....</b>    | <b>361</b> |
| <b>附录 B:</b> | <b>操作符优先级 .....</b>   | <b>362</b> |
| <b>附录 C:</b> | <b>窗口设计工具示例 .....</b> | <b>363</b> |

# 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 语言在言的阴影.

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，决不可以再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 README.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：

```
lepp
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 vdc.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:\  
vdc / 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  
ztcpl.exe
```

考备下列文件到 EDITER 盘上：

```
examples.vdk  
vdc.com  
vdc.doc  
vdef54.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:\LCPD 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.

不要担心如何牢记住这些术语, 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. 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