

目 录

21 天工作日历	I
如何使用本书	I
本书的特点	I
定义引用	I
致谢	IV
关于作者	V

第一周

第一周一瞥	1
欢迎进行第一周的学习	1
第一天 C++ 概览	3
1.1 欢迎来到面向对象的 C++ 世界	3
1.1.1 C++ 的历史	3
1.1.2 非 OOP 编程方法	4
1.1.3 OOP 程序设计方法的优越性	5
1.1.4 保持轻松愉快的心境来阅读本书	6
1.2 究竟什么是对象	6
1.3 OOP, C++ 和 C	7
1.4 Visual C++ 和本书	8
1.5 键入和编译 C++ 程序	9
1.5.1 准备用 C++ 进行编程	11
1.5.2 运行程序	11
1.6 现在读者已经学到的知识	14
1.7 读者将要学习的内容	15
1.8 小 结	15
1.9 问题和解答	15
1.10 习 题	16
1.10.1 问 答	16
1.10.2 练 习	16
第二天 性能卓越的 C++ 语言	17
2.1 在程序中插入注释行	17
2.1.1 [句法]C++ 注释: //	18
2.2 Visual C++ : 代码更少, 功能更强	18
2.3 C++ 的函数原型	20
2.4 函数原型中未说明参数意味着参数表为空	22

2.5	C++和数据	22
2.5.1	字符占用一字节内存	23
2.5.2	进行变量定义的位置	23
2.5.3	只在好莱坞才进行类型转换	25
2.5.4	作用域操作符——::	26
2.5.5	全局常量的作用域	27
2.6	摒弃#define	29
2.6.1	const:代替#define 来定义简单常量	29
2.6.2	内联函数:宏定义的改进	31
2.7	几个有用的 Visual C++符号常数	34
2.8	小 结	37
2.9	问题与解答	37
2.10	习 题	38
2.10.1	问 答	38
2.10.2	练 习	38
第三天	简单的 I/O	40
3.1	输出数据用 cout 和 <<	40
3.2	用 cin 和 >> 作输入也很容易	43
3.3	怎样显示错误信息	44
3.3.1	句法: I/O 对象:cout ,cin 和 cerr 的用法	45
3.4	路向何方	46
3.5	输出的控制	48
3.6	输入的考虑	55
3.7	小 结	59
3.8	问题和解答	59
3.9	习 题	60
3.9.1	问 答	60
3.9.2	练 习	60
第四天	功能强大的指针	61
4.1	void 类型的指针(void pointer)的功能	61
4.2	关于数据的引用	65
4.3	const 中的常数	68
4.4	关键字 const 用于指针和引用	69
4.5	指向常数的指针	69
4.6	常数指针	71
4.7	指向常数的常数指针	71
4.8	只读别名	72
4.9	小 结	73
4.10	问题和解答	74

4.11 习 题	75
4.11.1 问 答	75
4.11.2 练 习	75
第五天 内存分配:在需要的时候进行	77
5.1 new 用于分配,delete 用于释放	77
5.2 内存分配的概念	78
5.3 为一内存堆说明 new	79
5.4 在分配时进行初始化	82
5.5 用 delete 释放内存	84
5.6 多维数组	92
5.7 如果堆出现问题	94
5.8 小 结	96
5.9 问题和解答	96
5.10 习 题	97
5.10.1 问 答	97
5.10.2 练 习	97
第六天 函数通信	98
6.1 引用传递	98
6.2 传递引用变量代替按地址传递	101
6.3 返回一个引用	104
6.4 缺省参数加快了程序的开发	106
6.5 使用多个缺省参数	107
6.6 Visual C++和命令行参数	110
6.7 使用参数数目可变的参数表	113
6.8 小 结	116
6.9 问题与回答	116
6.10 习 题	117
6.10.1 问 答	117
6.10.2 练 习	118
第七天 能减轻工作量的重载函数	119
7.1 重载函数	119
7.2 调用用户自己的 C 函数	123
7.3 简单的运算符重载	125
7.4 提高重载的地位	127
7.5 operator...()函数	127
7.6 扩大运算符重载的作用	129
7.7 最后的警告	130
7.8 小 结	132
7.9 问题和解答	133

7.10 习 题.....	134
7.10.1 问 答.....	134
7.10.2 练 习.....	135
第一周回顾	136

第二周

第二周一瞥	141
第二周学习的中心目标是“对象”.....	141
第八天 在数据中增加“类”	142
8.1 抽象数据类型	142
8.1.1 建立一个抽象数据类型	142
8.1.2 嵌套的抽象数据类型	145
8.2 从 struct 提高到 class	148
8.3 学习“对象”概念	149
8.4 类和结构之间唯一的不同之处	151
8.5 Public 和 Private	155
8.6 混合使用和匹配访问说明符	157
8.7 小 结	161
8.8 问题和解答	161
8.9 习 题	162
8.9.1 问 答	162
8.9.2 练 习	163
第九天 成员函数激活类变量	164
9.1 成员函数和数据成员的联编	164
9.2 使用公用成员函数	165
9.3 关于对象的全部内容	168
9.4 现在可以控制数据的访问了	170
9.5 main()可以给对象传送一个数据	171
9.6 从成员函数返回	174
9.7 使类变得清晰简捷	176
9.8 为效率起见使用“inline”(内联)	179
9.9 使类自我保护	180
9.10 一个更加有用的例子.....	185
9.11 一个绕口的词:封装	188
9.12 隐式指针 * this	189
9.13 小 结.....	190
9.14 问题和解答.....	190
9.15 习 题.....	191
9.15.1 问 答.....	192

9.15.2 练习	192
第十天 友元	193
10.1 为什么使用友元	193
10.2 友元函数	194
10.3 使用 friend 定义友元函数	194
10.4 两个类的友元	197
10.5 友元类	204
10.6 小结	206
10.7 问题与解答	206
10.8 习题	207
10.8.1 问答	207
10.8.2 练习	207
第十一天 操作符重载	208
11.1 小复习	208
11.2 用 OOP 思想处理操作符重载	210
11.3 简单的算术操作符重载	217
11.4 关系和逻辑操作符重载	222
11.5 复合操作符重载	226
11.6 混合类与内部数据类型	232
11.7 重载++和--	234
11.8 扩充性	237
11.9 小结	240
11.10 问题与解答	241
11.11 习题	241
11.11.1 问答	242
11.11.2 练习	242
第十二天 扩展操作符重载	243
12.1 输入输出操作符重载	243
12.2 重载<<完成输出的细节问题	245
12.3 重载输入操作符>>的几点细节	251
12.4 建立自己的 I/O 操作算子	256
12.5 下标重载	259
12.6 小结	263
12.7 问题与解答	263
12.8 习题	264
12.8.1 问答	264
12.8.2 练习	264
第十三天 构造函数与析构函数	265
13.1 定义构造函数	265

13.2	定义析构函数.....	266
13.3	为什么需要构造函数和析构函数.....	267
13.4	时机是关键问题.....	269
13.5	带参构造.....	273
13.6	重载类型转换符.....	277
13.7	何时显式调用构造函数.....	278
13.8	缺省构造函数注意事项.....	279
13.9	建立对象数组.....	280
13.10	函数 Operator=()与拷贝构造函数	282
13.11	小 结	287
13.12	问题与解答	287
13.13	习 题	288
13.13.1	问 答	288
13.13.2	练 习	289
第十四天	轻松结束:静态及较大型的程序	290
14.1	关于 static(静态)	290
14.1.1	static:保护数据	290
14.1.2	静态全局变量的不同之处.....	293
14.1.3	静态函数.....	293
14.1.4	对象并不是例外.....	295
14.2	在类中 static 的特殊应用	298
14.3	多文件处理.....	302
14.4	温习:编译及连接	304
14.4.1	工程文件.....	306
14.5	小 结.....	313
14.6	问题与解答.....	314
14.7	习 题.....	315
14.7.1	问 答.....	315
14.7.2	练 习.....	315
第二周回顾		316

第三周

第三周一瞥.....	321
复用代码提高编程速度.....	321
第十五天 数据继承.....	322
15.1 派生结构.....	322
15.2 深入学习 Visual C++的继承	326
15.3 利用受保护的访问使用私有成员.....	328
15.4 派生类如何使用继承来的成员.....	330

15.5	学习的方向	334
15.6	小 结	335
15.7	问题与解答	335
15.8	习 题	336
15.8.1	问 答	336
15.8.2	练 习	337
第十六天	继承限制与扩展	338
16.1	为什么需要用到构造初始化表	338
16.2	构造初始化风格	340
16.3	首先建立基类	342
16.4	关于析构函数	353
16.5	小 结	354
16.6	问题与解答	355
16.7	习 题	355
16.7.1	问 答	355
16.7.2	练 习	356
第十七天	数据合成	358
17.1	合成和继承	358
17.2	使用合成方法编程	360
17.3	简化合成	372
17.4	用合成的对象彼此赋值	374
17.5	小 结	380
17.6	问题和解答	380
17.7	习 题	381
17.7.1	问 答	381
17.7.2	练 习	381
第十八天	虚拟函数	383
18.1	类家族	383
18.2	什么时候执行静态和动态联编	384
18.3	虚拟函数	390
18.4	说明虚拟函数	394
18.5	多态性：“多种形式”	395
18.6	小 结	397
18.7	问题和解答	397
18.8	习 题	398
18.8.1	问 答	398
18.8.2	练 习	399
第十九天	异常控制	400
19.1	异常处理的需要	401

19.1.1	异常控制的起源	401
19.1.2	错误种类	402
19.2	没有异常控制的世界	402
19.3	一些术语	408
19.4	为异常控制编制程序	409
19.5	小 结	412
19.6	问题和解答	412
19.7	习 题	413
19.7.1	问 答	413
19.7.2	练 习	413
第二十天	简单的文件 I/O	414
20.1	顺序 I/O: 读、写和追加	414
20.2	准备好	415
20.3	向顺序文件中写入字符数据	415
20.4	从顺序文件中读取字符数据	419
20.5	有什么问题吗	421
20.6	向顺序文件中追加数据	422
20.7	类数据和磁盘文件	423
20.8	随机存取文件	428
20.9	小 结	432
20.10	问题和解答	432
20.11	习 题	433
20.11.1	问 答	433
20.11.2	练 习	433
第二十天	Visual C++ 工具	434
21.1	程序员水平的提高过程	434
21.2	研究 Visual C++	436
21.2.1	Visual 工作台编辑器	436
21.2.2	工具栏	437
21.2.3	状态栏	438
21.2.4	多文件和 Windows	440
21.3	Visual C++ 调试器	442
21.3.1	跟踪代码执行过程	443
21.3.2	设置断点	443
21.3.3	附加控制	445
21.4	余下的工具有更强的功能	446
21.4.1	App Studio	446
21.4.2	App Wizard: 超凡的功能	447
21.4.3	用 Class Wizard 建立应用程序	449

21.4.4 监控应用程序.....	450
21.4.5 Visual C++ 1.5 的新特性	451
21.5 小 结.....	452
21.6 问题及解答.....	453
21.7 习 题.....	453
21.7.1 问 答.....	453
21.7.2 练 习.....	453
第三周回顾	454

附 录

附录 A Windows 程序设计与 Microsoft 基本类(MFC)简介	458
A.1 Windows 程序概览	458
A.2 设置编程环境	460
A.3 从 AppWizard 开始	460
A.4 建立工程	463
A.5 运行程序	463
A.6 发生了些什么	464
A.6.1 应用文件	464
A.6.2 应用程序的类	465
A.7 类的描述	480
A.8 小 结	482
A.9 问题和解答	482
A.10 习 题	483
A.10.1 问 答	483
A.10.2 练 习	483
附录 B MFC 类:用户的有力工具	485
B.1 添加一些细节	485
B.2 轻松的第一步:About 框.....	486
B.3 用 MFC 添加功能说明	496
B.4 MFC 串类(String 类)	502
B.5 读取键盘输入	504
B.6 小 结	507
B.7 问题和解答	508
B.8 习 题	508
B.8.1 问 答	509
B.8.2 练 习	509
附录 C 文件和更多的 MFC 类	510
C.1 加入打印和模拟显示功能	510
C.2 对文件进行操作	512

C.2.1 产生一个新的应用程序	512
C.3 加入关于文件的存储、打开和建立的代码	519
C.3.1 写入另外一些数据	522
C.4 MFC 异常处理简介	524
C.5 小 结	529
C.6 问题和解答	530
C.7 习 题	531
C.7.1 问 答	531
C.7.2 练 习	531
附录 D 图形和 Visual C++	532
D.1 从一些图形开始	532
D.2 打开和关闭指针	532
D.2.1 熟悉坐标	535
D.2.2 进一步的学习	536
D.3 提高图形功能	540
D.4 画矩形	543
D.5 填充矩形	545
D.6 绘制光滑的椭圆	548
D.7 小 结	549
D.8 问题和解答	550
D.9 习 题	551
D.9.1 问 答	551
D.9.2 练 习	551
附录 E 下一步要做什么	552
E.1 现在还需要用 C 进行 Windows 程序设计吗	552
E.2 其它环境	553
E.3 Visual C++ 和 Windows NT	554
E.4 Visual Basic 概略	557
E.5 保持一致	558
E.6 推荐书目	558
E.7 小 结	559
E.8 问题与解答	559
E.9 习 题	560
E.9.1 问 答	560
E.9.2 练 习	560
附录 F ASCII 字符表	561
附录 G Visual C++ 关键字	568
附录 H 操作符优先级	569

附录 I 答案	571
I.1 第一天 C++概览(答案).....	571
I.1.1 问 答	571
I.1.2 练 习	571
I.2 第二天 性能卓越的 C++语言(答案)	571
I.2.1 问 答	571
I.2.2 练 习	572
I.3 第三天 简单的 I/O(答案)	572
I.3.1 问 答	572
I.3.2 练 习	573
I.4 第四天 功能强大的指针(答案)	573
I.4.1 问 答	573
I.4.2 练 习	574
I.5 第五天 内存分配,在需要的时候进行(答案)	575
I.5.1 问 答	575
I.5.2 练 习	575
I.6 第六天 函数通信(答案)	576
I.6.1 问 答	576
I.6.2 练 习	576
I.7 第七天 能减轻工作量的重载函数(答案)	578
I.7.1 问 答	578
I.7.2 练 习	578
I.8 第八天 在数据中增加“类”(答案)	581
I.8.1 问 答	581
I.8.2 练 习	581
I.9 第九天 成员函数激活类变量(答案)	582
I.9.1 问 答	582
I.9.2 练 习	583
I.10 第十天 友元(答案)	587
I.10.1 问 答	587
I.10.2 练 习	587
I.11 第十一天 操作符重载(答案)	589
I.11.1 问 答	589
I.11.2 练 习	589
I.12 第十二天 扩展操作符重载(答案)	592
I.12.1 问 答	592
I.12.2 练 习	593
I.13 第十三天 构造函数与析构函数(答案)	595
I.13.1 问 答	595

I. 13.2 练习	595
I. 14 第十四天 轻松结束:静态及较大型的程序(答案)	598
I. 14.1 问答	598
I. 14.2 练习	598
I. 15 第十五天 数据继承(答案)	599
I. 15.1 问答	599
I. 15.2 练习	599
I. 16 第十六天 继承、限制与扩展(答案)	601
I. 16.1 问答	601
I. 16.2 练习	601
I. 17 第十七天 数据合成(答案)	606
I. 17.1 问答	606
I. 17.2 练习	607
I. 18 第十八天 虚拟函数(答案)	607
I. 18.1 问答	607
I. 18.2 练习	608
I. 19 第十九天 异常控制(答案)	611
I. 19.1 问答	611
I. 19.2 练习	611
I. 20 第二十天 简单的文件 I/O(答案)	612
I. 20.1 问答	612
I. 20.2 练习	612
I. 21 第二十天 Visual C++ 工具(答案)	616
I. 21.1 问答	616
I. 21.2 练习	617
I. 22 附录 A Windows 程序设计与 Microsoft 基本类(MFC)简介(答案)	617
I. 22.1 问答	617
I. 22.2 练习	617
I. 23 附录 B MFC 类:用户的有力工具(答案)	618
I. 23.1 问答	618
I. 23.2 练习	618
I. 24 附录 C 文件和更多的 MFC 类(答案)	620
I. 24.1 问答	620
I. 24.2 练习	621
I. 25 附录 D 图形和 Visual C++(答案)	624
I. 25.1 问答	624
I. 25.2 练习	624
I. 26 附录 E 下一步做什么(答案)	628
I. 26.1 问答	628

I.26.2 练习	628
附录 J C 概念的复习	629
J.1 C 的不同	629
J.2 C 程序的格式	629
J.3 C 的注释	630
J.4 预处理指令	630
J.5 C 数据	631
J.6 首先考虑变量	632
J.7 输入/输出	633
J.8 操作符	636
J.9 特殊算术操作符	637
J.10 关系操作符与逻辑操作符	638
J.11 测试数据	639
J.12 指针	641
J.13 进一步学习提示	642
J.13.1 C 入门指导	642
J.13.2 C 语言程序设计 21 日通	642
J.13.3 高级 C 语言	642
附录 K 术语表	643

第一周一瞥

现在,请准备好开始进行 Visual C++ 的学习。在今后的三个星期中,本书将介绍如何使用 Visual C++ 进行面向对象的程序设计(object-oriented programming,即 OOP)。如果读者对 OOP 是什么一无所知,这二十一天的学习将揭开它神秘的面纱,这本书由课文、练习、说明以及供复习用的问答题和练习构成,全部习题的答案在附录 D 中给出。所有这些部分完美地组合在一起,每一部分对于理解 OOP 的概念,以及掌握使用 Visual C++ 编写面向对象的程序都有极大帮助。

为了巩固当天所学的知识,在每一天课程的末尾都配有复习和小测验。在学完一天的课程之后,读者应该能够回答测验中的题目,并且编写出符合练习要求的程序(在练习的过程中,务必记住:同一程序往往有若干种实现方法,因此,如果读者编写的程序和本书提供的答案不一样,该程序也可能是正确的)。

欢迎进行第一周的学习

在第一个七天中接触到的 Visual C++ 并不涉及到 OOP 概念。读者买本书就是为了学习 OOP,而本书的主旨也是为了介绍 OOP,但是,掌握面向对象的程序设计方法的过程可以明确地分为三个阶段,这和本书三个星期式的结构正好相符,在这第一周的学习中,读者将通过本书对 Visual C++ 中 OOP 概念以外部分的介绍,理解 OOP 方法的基础(由 Visual C++ 改进了的这部分语言表面上看起来和 OOP 无关,事实上却正是用来支持 OOP 的)。

读者一旦接触到 OOP,就会发现 Visual C++ 提供了一种比 C 语言更好的程序设计方法,即使只掌握了第一个星期所教的知识,而没有学习 OOP,读者仍将发现 C++ 比 C 语言更加先进(确实如此)。

注意: 本书假定读者已具备了一定的 C 语言基础,如果读者想要复习一下 C 语言的基础知识,可以参看本书的附录 E。

C++ 的设计者并不仅仅是为了改进 C 语言,他们设计这种语言是为了获得一种支持面向对象的程序设计方法的语言,结果 C++ 比 C 语言更加先进,但是所有这些改进不仅支持了 OOP 设计方法,还使非 OOP 程序设计有了长足的发展。

在第一个星期中,读者将对如何使用 Visual C++ 的编辑器来编辑或者编译 C++ 的程序有所了解,并且马上就会意识到 C++ 对程序的数据输入输出进行了怎样的改进,也将知道为什么 C++ 的程序员从不使用 printf() 和 scanf() (恰恰在读者以为自己熟练掌握了它们的时候)。

第一个星期的课程还提出了改进设计和编写函数的技巧,C++ 增加了许多省时省力方便快捷的方法,在头两天的学习中读者就会用到其中一些方法。通过对注释语句的一个简单的改变,读者可以重写一些内部函数使它们完成所需要的功能。C++ 中含有一些不涉及 OOP 思想的命令,同时还对 C 语言进行了改进。在这个星期里,本书提供了这两方面内容的有关知识。

另外,用 Visual C++ 编程的一个优点是读者所掌握的有关 C 语言的知识仍旧有用, Visual C++ 支持几乎所有 C 的特性, C 的命令、语法并且读者已经知道并掌握的所有的头文件和函数在 Visual C++ 的程序中都能正常运行。通常情况下,当学习一种新的语言时,一切都必须从头开始学,如数据的类型、赋值语句等,而学习 Visual C++, 却可以一开始就编写大段的程序。

完全掌握了第一星期的课程后,就可以准备运用这些知识用 Visual C++ 编写面向对象的程序了。

第一天 C++ 概览

欢迎阅读本书！第一章旨在激发读者学习 C++ 的兴趣，在本天，读者将学习以下三个内容：

- ☐ C++ 的历史。
- ☐ 面向对象的程序设计方法(OOP)的功能。
- ☐ OOP 程序员使用 C++ 的优越性。

1.1 欢迎来到面向对象的 C++ 世界

很多人认为 C++ 是 C 语言的扩展，但实际上并非如此——C++ 是一种独立的语言。C++ 大大提高了 C 的性能，不过它的基础仍然是 C。一个 C++ 程序员必须认识到这一点，否则将会产生许多问题。学习 C++ 的读者中有许多在此之前已经是 C 程序员，这一点是显而易见的，因为毕竟 C 在 1980 年公布，而 C++ 在 1990 年才公布。

因为 C++ 面向对象的特性，许多程序员正在由 C 或者别的编程语言转向 C++。面向对象的程序设计，即 OOP，如同众所周知的那样，把 1980 年以来的以过程为核心概念的传统程序设计方法所支撑的软件产业向前推进了一步。

面向对象的程序设计方法提供了许多强有力的工具，帮助程序员达到预期目标(编制和调试应用程序)，这种方法比以往的面向过程的设计方法更有效率。面向对象的程序设计把数据封装在一个安全的外壳中，并使它们具有活动性，变量能够对自己进行主动的操作而非被动地等待程序中的代码对它们进行操作。当需要打印一个变量的内容时，你无须去打印它，而只需要告诉变量去打印自己！这种魔术般的编程方法正是一种最自然的编程方法。

1.1.1 C++ 的历史

建立起 C++ 应归功于 Bjarne Stroustrup。他发展了 C++，使程序中的事件成为现实中事件的模拟，从而使 C++ 几年后风靡海内外。Stroustrup 揭示出了普通的不具备 OOP 思想的程序设计方法的缺陷，即它不能像面向对象的程序设计那样在程序中模拟现实中的事件来完成某项任务。

Stroustrup 为 AT&T 公司的 Bell 实验室工作了许多年，致力于发展和提高 C++。美国国家标准学会(ANSI, American National Standards Institute)作为一个制定所有和计算机有关的语言标准的组织也接受了 C++ 的标准。但是他们还在致力于制定一个更完备的标准，这是因为 ANSI 还没有为 C++ 制定出一个实际可行的标准来和 AT&T 的标准竞争。

Microsoft 公司是 Visual C++ 软件的制作者。AT&T 公司需要一个基准测试程序来确定一个编译器是否足够先进可以用作 C++ 的正式编译器。除了不支持样板之外，Microsoft 的 C++ 编译器严格遵循了 AT&T 的标准。样板是一个被激烈辩论的问题，一些程序员坚持认为应该使用样板，但另一些程序员则从不用样板编程。许多 Visual C++ 程序员无论以前是否对样板感兴趣都编制出了许多功能强大的 PC 机程序，而且也并没有感到需要使用

别的支持样板的 C++ 编译器。之所以出现这种情况,也许是 Visual C++ 提供的丰富的实用工具使程序员不需要使用样板,也许是样板的作用一向被高估了。

注意:多年来,AT&T 公司发展了 C++ 的非面向对象方面的性能,使它优越于 C。许多增强的功能已被 ANSI 委员会加进他们的 C 标准以发展 C。例如,函数原型的概念并不是 C 提出的,但现在它作为 C 的组成部分已经有好几年了。函数原型概念最早是由 C++ 提出的,因为它非常有利于程序设计,所以 ANSI 把它加入到 C 中去了。

1.1.2 非 OOP 编程方法

当程序员不使用 OOP 方法时,编程的核心概念是过程。一个面向过程的程序完成一个应用目标的手段是顺序地执行某种指令序列。这种程序在带参量的指令引导下达到某个状态,也就是说是指令决定了程序运行所有可能的方向。C, Pascal, QBasic 和 COBOL 都是这种面向过程的程序设计方法的例子。

今天的计算机发展已不能满足于面向过程的程序设计方法。比如,Windows 这样的图形用户接口(GUI)可以使用面向过程的程序设计方法,但它只有在使用面对对象的程序设计方法时才更充分地体现出它的优越性能。

图标、对话框、选择框等 GUI 的屏幕窗口当诸如鼠标或键盘被按下等等的事件发生时被激活。当使用面向过程的程序设计方法编制 GUI 的应用程序时,用户必须在程序结构中利用数量庞大的 C 开关量来不停地测试这些事件是否发生。但前面我们已经提到过(而且在本书的其余部分将做更为详尽的解释),面向对象的程序设计无须这样做,而只需激活数据就可以了。在程序中所有显示在屏幕上的 GUI 元素都可被激活。

现实世界的程序设计

当用户阅读完本书,将会发现 OOP 思想比起非 OOP 思想更像是现实世界的一面镜子。OOP 的专家认为 OOP 从程序中“剔除抽象层”。

剔除抽象层的意思简单来说就是使程序能够更像它所映像的应用目标那样去工作。程序越接近映像的应用目标,编程就越快,调试也就越简单。比如如果用户想用 C++ 编制一段 CD-ROM 控制程序,以便在计算机的 CD-ROM 驱动器上放 CD 来欣赏所喜爱的音乐,用户必须为 play(播放),reward(倒带),fast forward(快进)和 eject(停止)等键定义相应的变量,当 play 键被按下,play 变量将被激活并接管以下的工作来放出一段音乐。

在面向过程的程序设计方法中,每种变量都有它自己的特性。一个 Sales 变量必须是唯一的,它精确地代表一个数字;一个 FirstName 变量必须是一个串变量或字符数组,它代表一个人的姓。在 OOP 语言中,变量不仅有特性,还有行为,行为是指对数据的操作。当用户给出一个变量的行为(通过编写代码),就使这个变量具有了对自身进行初始化、显示、运算等等的功能。

当用户的变量像拥有特性一样拥有了行为,编程工作就会变得非常简单,用户也不必在使用这些数据时都编写代码对它们进行操作。通过激活一个变量,C++ 对用户自定义数据的操作远比 C 更为清晰。当用户使用关键字 struct 定义自己的数据类型时,有许多不同的方法向 C++ 中添加这种新的类型。在 C 中,当用户声明了一个结构后,必须重复使用关键字 struct 来定义这个结构类型的变量,而在 C++ 中,用关键字 struct 声明了一个结构后