

在线服务：视频库、源代码库、专业论坛、专家实时支持

Visual C++ 2008 完全学习手册

张水波 等编著



45段全程配音语音教学视频

全书实例源代码，使学习、分析、调试程序更方便

在线服务方式

在线服务网站：www.itzcn.com

QQ群在线服务：45368980、33925615、107423140

清华大学出版社

在线服务：视频库、源代码库、专业论坛、专家实时支持

Visual C++ 2008 完全学习手册

张水波 等编著



45段全程配音语音教学视频

全书实例源代码，使学习、分析、调试程序更方便

在线服务方式

在线服务网站：www.itzcn.com

QQ群在线服务：45368980、33925615、107423140



清华大学出版社
北 京

内 容 简 介

本书介绍 Visual C++ 2008 支持的两种版本的 C++，详细介绍两种风格的 C++ 语言，使用微软基本类 MFC 开发本地的 ISO/ANSI C++ Windows 应用程序，使用 Windows Forms 开发 C++/CLI Windows 应用程序。本书共分 4 篇，分别介绍 Visual C++ 2008 基础知识、C++ 面向对象的程序开发、Windows 应用程序开发和数据库访问等内容。配套光盘提供了教学视频和实例源文件。

本书可以作为 Visual C++ 2008 的基础学习书籍，也可以帮助中级读者提高技能，顺利掌握编程应用实践知识。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。

版权所有，侵权必究。侵权举报电话：010-62782989 13701121933

图书在版编目 (CIP) 数据

Visual C++ 2008 完全学习手册 / 张水波等编著. —北京：清华大学出版社，2011.1
ISBN 978-7-302-23684-9

I. ①V… II. ①张… III. ①C 语言—程序设计—手册 IV. ①TP312-62

中国版本图书馆 CIP 数据核字 (2010) 第 162964 号

责任编辑：夏兆彦

责任校对：徐俊伟

责任印制：何 芊

出版发行：清华大学出版社

地 址：北京清华大学学研大厦 A 座

<http://www.tup.com.cn>

邮 编：100084

社 总 机：010-62770175

邮 购：010-62786544

投稿与读者服务：010-62795954, jsjic@tup.tsinghua.edu.cn

质 量 反 馈：010-62772015, zhiliang@tup.tsinghua.edu.cn

印 刷 者：清华大学印刷厂

装 订 者：三河市新茂装订有限公司

经 销：全国新华书店

开 本：190×260 印 张：41.75 字 数：1042 千字

附光盘 1 张

版 次：2011 年 1 月第 1 版

印 次：2011 年 1 月第 1 次印刷

印 数：1~5000

定 价：79.50 元

产品编号：034171-01

前 言

微软开发的 Visual Studio 2008 支持两种截然不同但又紧密相关的 C++ 语言：ISO/ANSI 标准 C++ 和 C++/CLI。ISO/ANSI 是标准 C++，它是在 20 世纪 80 年代早期开发的，是一种基于 C 的面向对象语言，所以这两种语言有许多共同的语法和功能，但是 C++ 要比其前身丰富得多，用途也广泛得多。标准 C++ 主要用于开发本地计算机上运行的高性能应用程序。C++/CLI 由微软公司开发，现在是 ECMA 标准，它是专门为 .NET Framework 开发的。

本书使用 Visual C++ 2008 支持的两种技术，讲授 C++ 编程的基础知识。详细介绍了两种 C++ 语言，使用微软基本类 MFC 开发本地的 ISO/ANSI C++ Windows 应用程序，使用 Windows Forms 开发的 C++/CLI Windows 应用程序等。

由于当前信息化时代中大量地使用数据库，本书还介绍了用于 MFC 和 Windows Forms 应用程序中访问数据源的技术。与 Windows Forms 应用程序相比，MFC 应用程序需要编写大量的代码来实现希望的功能，而 Windows Forms 程序经常只需要编写很少的代码，有时甚至不用编写代码。因此，本书重点介绍了 MFC 编程，而非 Windows Forms 编程。

1. 本书内容

本书介绍 Visual C++ 2008 支持的两种不同版本的 C++，详细介绍了两种风格的 C++ 语言，使用 Microsoft 基本类 MFC 开发本地的 ISO/ANSI C++ Windows 应用程序，使用 Windows Forms 开发 C++/CLI Windows 应用程序等。本书共分为 4 篇，分别是 Visual C++ 2008 基础知识、C++ 面向对象的程序开发、Windows 应用程序开发、数据库访问。各篇主要内容如下。

第 1 篇：Visual C++ 2008 基础知识（第 1~7 章）。首先介绍使用 C++ 编写本地应用程序和 .NET Framework 应用程序所需要理解的基本概念，以及在 Visual C++ 2008 开发环境中体现的主要思想。第 2~7 章专门讲授两个 C++ 语言版本。

第 2 篇：C++ 面向对象的程序开发（第 8~12 章）。本篇介绍了什么是面向对象的程序开发，以及面向对象的程序设计的主要特性。最后介绍如何使用标准模板库 STL。

第 3 篇：Windows 应用程序开发（第 13~20 章）。本篇是本书的重点之一，首先介绍创建 Windows 应用程序的几种方式，然后重点介绍如何使用 MFC 开发本地的 Windows 应用程序，并使用 C++/CLI 创建了 Windows 应用程序。

第 4 篇：数据库访问（第 21~22 章）。本篇首先使用 MFC 创建访问数据库的 Windows 应用程序，然后使用 Windows Forms 进行高度的可视化数据库编程。

2. 本书特色

本书中采用大量的实例进行讲解，力求通过实际操作使读者更容易地掌握 Visual C++ 2008 的知识。本书难度适中，内容由浅入深，实用性强，覆盖面广，条理清晰。

□ 知识点全 本书紧紧围绕 Visual C++ 2008 的实际应用、管理与开发展开讲解，具有很强的逻辑性和系统性。



- **基于理论，注重实践** 在讲述过程中，不仅仅只介绍理论知识，而且在合适位置安排具有代表性、操作性强的综合应用，将理论应用到实践当中，来加强读者的实际应用能力，巩固 Visual C++ 的理论知识。
- **快速掌握** 注重技术原理和实际应用的高度融合，通过循序渐进的内容组织，以及大量来自工作现场的应用案例，帮助读者快速掌握和应用 Visual C++ 2008 编程技术。
- **随书光盘** 本书为实例配备了视频教学文件，读者可以通过视频文件更加直观地学习 Visual C++ 2008 的使用知识。
- **网站技术支持** 读者在学习或者工作的过程中，如果遇到实际问题，可以直接登录 www.itzcn.com 与我们联系，作者会在第一时间内给予帮助。

3. 读者对象

本书具有知识全面、实例精彩、指导性强的特点，力求以全面的知识及丰富的实例来指导读者透彻地学习 Visual C++ 2008 各方面的知识。本书可以作为 Visual C++ 2008 的入门书籍，也可以帮助中级读者提高技能，对高级读者也有一定的启发意义。

本书适合以下人员阅读学习。

- 希望学习 C++ 语言的初学者。
- 了解 C++ 并希望使用 C++ 开发 Windows 应用程序者。
- 各大中专院校的在校学生和相关授课老师。

除了封面署名人员之外，参与本书编写的还有胡家宏、于永军、张秋香、李乃文、张仕禹、夏小军、赵振江、李振山、李文才、吴越胜、李海庆、何永国、李海峰、陶丽、吴俊海、安征、张巍屹、崔群法、王咏梅、康显丽、辛爱军、牛小平、贾栓稳、王立新、苏静、赵元庆、郭磊、徐铭、李大庆、王蕾、张勇、郝安林、郭新志、牛丽平、唐守国等。在编写过程中难免会有疏漏，欢迎读者与我们联系，帮助我们改正提高。

编者
2010年6月

目 录

第一篇 Visual C++ 2008 基础知识

第 1 章 开始使用 Visual C++ 20081	2.3.4 浮点类型.....25
1.1 .NET Framework 3.5.....1	2.3.5 定义数据类型的别名.....26
1.1.1 什么是.NET Framework.....1	2.4 输入与输出表达式.....26
1.1.2 公共语言运行库 CLR.....2	2.4.1 标准 I/O 流.....27
1.1.3 .NET Framework 类库.....2	2.4.2 格式化输出.....27
1.1.4 .NET Framework 3.5 的新功能.....3	2.5 运算符与表达式.....31
1.2 C++应用程序.....3	2.5.1 算术运算.....31
1.3 Visual C++ 2008 与 Visual Studio 2008.....4	2.5.2 赋值运算.....36
1.4 控制台应用程序.....7	2.5.3 逗号运算.....37
1.4.1 Win32 控制台应用程序.....7	2.5.4 关系运算.....38
1.4.2 创建 CLR 控制台程序.....10	2.5.5 逻辑运算.....39
1.5 创建 Windows 应用程序.....11	2.5.6 位运算.....41
1.5.1 创建 MFC 应用程序.....12	2.6 C++/CLI 的应用（托管 C++）.....43
1.5.2 创建 Windows Forms 应用程序.....13	2.6.1 C++/CLI 的基本数据类型.....43
第 2 章 变量和表达式15	2.6.2 C++/CLI 格式化输出.....45
2.1 C++的基本语法.....15	2.6.3 C++/CLI 的键盘输入.....47
2.1.1 程序的结构.....15	2.6.4 safe_cast 安全类型转换.....47
2.1.2 注释.....18	第 3 章 流程控制结构48
2.1.3 #include 指令.....19	3.1 条件语句.....48
2.1.4 命名空间.....19	3.1.1 if 语句.....48
2.1.5 程序块.....20	3.1.2 条件运算符.....53
2.2 变量和常量.....21	3.1.3 switch 语句.....54
2.2.1 定义变量.....21	3.2 循环语句.....56
2.2.2 常量和符号常量.....22	3.2.1 while 语句.....57
2.3 基本数据类型.....23	3.2.2 do-while 语句.....59
2.3.1 整型.....23	3.2.3 for 语句.....60
2.3.2 字符类型.....24	3.2.4 嵌套循环.....62
2.3.3 布尔类型.....25	3.3 跳转结构.....63
	3.3.1 break 语句.....63

3.3.2	continue 语句	65	5.8	变量的生存期	114
3.3.3	goto 语句	67	5.8.1	自动变量	115
3.4	C++/CLI 的应用	68	5.8.2	静态变量	116
第 4 章	数组和字符串	71	5.9	函数的重载	117
4.1	数组概述	71	5.10	函数模板	119
4.1.1	定义数组	71	5.11	C++/CLI 的应用	122
4.1.2	引用数组元素	72	5.11.1	接收数量可变的实参	122
4.1.3	初始化	73	5.11.2	泛型函数	123
4.2	多维数据	74	第 6 章	指针与引用	126
4.2.1	多维数组的定义	75	6.1	指针的概念	126
4.2.2	多维数组的初始化	76	6.2	声明指针变量	126
4.3	数组的应用	78	6.3	指针运算符	127
4.3.1	排序	78	6.3.1	取地址运算符	127
4.3.2	倒置矩阵	84	6.3.2	间接运算符	128
4.4	字符数组和字符串	85	6.4	指针变量的初始化	129
4.4.1	字符数组	85	6.5	指针和数组	130
4.4.2	string 字符串	86	6.5.1	使用指针法引用数组元素	130
4.4.3	使用 string 字符串	87	6.5.2	使用指针处理多维数组	133
4.4.4	Unicode 字符串	92	6.6	指针和函数	135
4.5	C++/CLI 的应用	92	6.6.1	作为形参的指针	135
4.5.1	CLR 数组	92	6.6.2	返回指针的函数	137
4.5.2	CLR 字符串	95	6.6.3	函数指针	139
第 5 章	函数	97	6.7	常量指针和指针常量	140
5.1	函数的定义	97	6.7.1	指向常量的指针	140
5.2	函数的参数和返回值	98	6.7.2	指针常量	142
5.2.1	实参与形参	98	6.8	指向 char 类型的指针变量	143
5.2.2	函数的返回值	101	6.9	动态内存的分配	145
5.2.3	使用 const 修饰符的函数形参	102	6.9.1	堆与 new 和 delete 运算符	145
5.3	函数的调用	102	6.9.2	数组的动态分配内存	146
5.3.1	函数调用机制	103	6.10	引用	149
5.3.2	函数原型	104	6.11	C++/CLI 中的指针与引用	151
5.4	函数的递归调用	105	6.11.1	跟踪句柄	151
5.5	函数的默认参数值	108	6.11.2	跟踪引用	153
5.6	函数和数组	109	第 7 章	程序文件和预处理指令	154
5.7	函数与变量的作用域	111	7.1	使用头文件	154
5.7.1	局部变量	111	7.2	外部名称	156
5.7.2	全局变量	113	7.3	命名空间	157

7.3.1	声明命名空间	157	7.4.1	文件包含	160
7.3.2	未指定名称的命名空间	159	7.4.2	宏定义	161
7.3.3	嵌套的命名空间	160	7.4.3	条件编译	162
7.4	预处理指令	160	7.4.4	#error 和#pragma 指令	164

第二篇 C++面向对象的程序开发

第 8 章	自定义数据类型	167	9.2.1	对象的指针	205
8.1	结构	167	9.2.2	对象的引用	207
8.1.1	定义结构	167	9.3	this 指针	207
8.1.2	访问结构成员	168	9.4	类的静态成员	209
8.1.3	结构与指针	169	9.4.1	静态数据成员	209
8.2	联合	171	9.4.2	静态成员函数	211
8.2.1	定义联合	172	9.5	复制构造函数	212
8.2.2	匿名联合	173	9.6	类的友元	215
8.3	枚举	174	9.6.1	类的友元函数	216
8.4	类的概念与定义	175	9.6.2	友元类	219
8.4.1	定义类	176	9.7	重载运算符重载	220
8.4.2	创建类的对象	176	9.7.1	重载运算符的概述	220
8.4.3	访问类的数据成员	177	9.7.2	重载双目运算符	221
8.4.4	为类添加成员函数	178	9.7.3	重载增量运算符	223
8.5	类的构造函数	180	9.8	类模板	224
8.5.1	构造函数的使用	180	9.8.1	定义类模板	225
8.5.2	类的默认构造函数	182	9.8.2	根据类模板创建对象	227
8.5.3	默认的初始化值	184	9.9	C++/CLI 的应用	231
8.5.4	构造函数中的初始化列表	186	9.9.1	在数值类中重载运算符	231
8.6	类的析构函数	186	9.9.2	重载数值类的递增和递减运算符	233
8.7	类的私有成员	189	9.9.3	重载引用类运算符	234
8.8	C++/CLI 中的枚举	191	第 10 章	继承与多态性	236
8.9	C++/CLI 中的类和对象	193	10.1	继承的基本概念	236
8.9.1	数值类类型	193	10.1.1	类的层次关系	236
8.9.2	引用类类型	195	10.1.2	派生类对象的结构	238
8.9.3	类属性	197	10.2	继承机制下的访问控制	238
第 9 章	类的更多功能	203	10.2.1	访问父类成员	239
9.1	对象数组	203	10.2.2	继承方式	243
9.2	对象的指针与引用	205	10.3	派生类的构造函数	246

10.4	派生类的析构函数	248	11.3.1	队列容器	292
10.5	派生类的复制构造函数	249	11.3.2	优先级容器	294
10.6	多重继承	252	11.3.3	堆栈容器	296
10.7	在相关的类类型之间转换	254	11.4	关联容器	298
10.8	理解多态性	256	11.4.1	使用映射容器	298
10.8.1	虚函数	257	11.4.2	使用多重映射容器	301
10.8.2	使用虚函数的原则	260	11.5	迭代器	303
10.9	纯虚函数和抽象类	263	11.5.1	输入流迭代器	303
10.9.1	纯虚函数	263	11.5.2	输出流迭代器	304
10.9.2	抽象类	263	11.5.3	插入迭代器	305
10.10	C++/CLI 的应用	265	11.6	算法	307
10.10.1	装箱与拆箱	265	11.6.1	fill、fill_n、generate 与 generate_n	307
10.10.2	C++/CLI 中的继承	266	11.6.2	数学算法	308
10.10.3	接口类	268	11.7	C++/CLI 中的 STL	309
10.10.4	委托和事件	270	11.7.1	STL/CLR 顺序容器	309
第 11 章	标准模板库	277	11.7.2	STL/CLR 关联容器	311
11.1	标准模板库概述	277	第 12 章	异常处理	314
11.1.1	STL 容器	277	12.1	异常的概念	314
11.1.2	STL 迭代器	280	12.2	捕获异常	315
11.1.3	STL 算法	281	12.2.1	try-catch 块	316
11.1.4	STL 头文件	281	12.2.2	捕获所有的异常	318
11.2	顺序容器	282	12.3	抛出异常的函数	319
11.2.1	vector 顺序容器	282	12.4	标准异常类	320
11.2.2	deque 顺序容器	289	12.5	MFC 异常	321
11.2.3	使用 list 容器	290			
11.3	容器适配器	292			

第三篇 Windows 应用程序开发

第 13 章	创建 Windows 应用程序	323	13.2.2	入口函数	327
13.1	认识 Windows 应用程序	323	13.2.3	窗口过程函数	335
13.1.1	窗口	323	13.2.4	一个简单 Windows 程序	337
13.1.2	Windows 程序的工作过程	324	13.3	使用 MFC	339
13.1.3	Windows API	325	13.4	使用 Windows Forms	342
13.2	Windows 程序结构	325	第 14 章	使用 MFC 创建 Windows 程序	344
13.2.1	Windows 数据类型	325	14.1	MFC 概述	344

14.1.1	MFC 和 Win32	344	16.1.4	菜单命令范围	393
14.1.2	MFC 类库	345	16.1.5	更新菜单项	394
14.1.3	MFC 应用程序的运行	346	16.1.6	菜单更新范围	396
14.1.4	MFC 中的全局函数	347	16.2	快捷键	397
14.2	创建 MFC 应用程序	347	16.3	动态更改菜单	398
14.2.1	创建 SimpleEditor 应用程序	347	16.3.1	手工编辑创建菜单	398
14.2.2	管理项目	351	16.3.2	修改现有菜单	399
14.3	代码分析	352	16.3.3	系统菜单	401
14.3.1	应用程序类 CSimpleEditorApp	352	16.4	快捷菜单	401
14.3.2	主框架窗口类 CMainFrame	355	16.5	在 C++/CLI 程序中使用菜单	403
14.3.3	文档类 CSimpleEditor	356	16.5.1	理解 Windows Forms 应用程序	403
14.3.4	视图类 CSimpleEditorView	357	16.5.2	MDI 窗口	406
14.3.5	预编译头文件 stdafx.h	359	16.5.3	活动子窗体	408
第 15 章	消息和命令	360	16.5.4	排列子窗体	411
15.1	与 Windows 进行通信	360	16.5.5	合并菜单	412
15.1.1	了解消息映射	360	16.5.6	替换和删除菜单	415
15.1.2	处理不同类型的消息	362	16.5.7	快捷菜单	417
15.2	获取鼠标输入	363	第 17 章	在窗口中绘图	419
15.2.1	客户区鼠标消息	363	17.1	Windows GDI	419
15.2.2	非客户区鼠标消息	368	17.1.1	MFC 设备描述表类	419
15.2.3	WM_NCHITTEST 消息	370	17.1.2	设备描述表的属性	420
15.2.4	鼠标滚轮消息	370	17.1.3	绘图模式	421
15.2.5	捕获鼠标	372	17.1.4	映射模式	422
15.3	获取键盘输入	373	17.1.5	移动坐标原点	423
15.3.1	输入焦点	373	17.1.6	获取设备信息	424
15.3.2	按键消息	373	17.2	用 GDI 绘图	425
15.3.3	字符消息	375	17.2.1	应用程序中的视图类	425
15.3.4	插入符	376	17.2.2	绘制直线和曲线	426
15.4	定时器消息	380	17.2.3	绘制椭圆和矩形	429
15.4.1	WM_TIMER 消息	380	17.2.4	画笔	429
15.4.2	回调函数设置定时器	382	17.2.5	画刷	431
15.5	自定义消息	383	17.2.6	绘制文本	432
第 16 章	菜单	386	17.2.7	字体	434
16.1	菜单	386	17.2.8	预定义对象	438
16.1.1	利用资源编辑器编辑菜单	386	17.3	使用鼠标绘图	439
16.1.2	菜单消息	389	17.3.1	绘制图形分析	439
16.1.3	添加消息处理代码	390	17.3.2	创建绘图类	440
			17.3.3	完成基类 CShape	442

17.3.4	绘制直线	443	18.7.3	颜色对话框类	504
17.3.5	完成其他派生类	444	18.7.4	打印对话框类	504
17.3.6	创建绘图类对象	447	18.8	Windows Forms 窗体	505
17.3.7	捕获鼠标消息	452	18.9	Windows Forms 控件	506
17.3.8	保存绘图对象	453	18.9.1	控件类	506
17.4	使用 GDI+绘图	454	18.9.2	Button 控件	508
17.4.1	了解 GDI+	454	18.9.3	RadioButton 和 CheckBox 控件	509
17.4.2	图形操作	455	18.9.4	GroupBox 控件	510
17.4.3	坐标和区域	458	18.9.5	Label 和 LinkLabel 控件	510
17.4.4	使用 Font 类绘制文本	459	18.9.6	TextBox 控件	511
17.4.5	图像操作	462	18.9.7	RichTextBox 控件	515
18.9.8	Listbox 和 CheckedListBox 控件	517	18.9.8	Listbox 和 CheckedListBox 控件	517
第 18 章	对话框和控件	464	第 19 章	文件和串行化	519
18.1	理解对话框和控件	464	19.1	操作文件的几种方式	519
18.2	传统控件	465	19.1.1	使用 CRT 函数 std::fxxx()	519
18.2.1	CButton 类	465	19.1.2	使用标准 C++库 std::fstream	521
18.2.2	CListBox 类	469	19.1.3	使用 Windows API 函数	523
18.2.3	CStatic 类	471	19.2	CFile 类	525
18.2.4	CEdit 类	472	19.2.1	打开、关闭和创建文件	525
18.2.5	CComboBox 类	475	19.2.2	文件的读和写	528
18.2.6	CScrollBar 类	477	19.2.3	CFile 类的派生类	529
18.3	高级控件编辑	478	19.3	串行化	530
18.3.1	添加键盘接口	478	19.3.1	串行化基础	530
18.3.2	修改控件行为	479	19.3.2	创建可串行化类	532
18.4	对话框	480	19.3.3	为 AddressBook 程序添加串行化 功能	532
18.4.1	对话框模板	480	19.4	访问 XML 文件	535
18.4.2	CDialog 类	483	19.4.1	XML 文档	535
18.4.3	创建模式对话框	484	19.4.2	解析 XML 文档	540
18.4.4	创建无模式对话框	486	19.4.3	选择节点	546
18.4.5	对话框数据交换与校验	487	19.5	CLR 中的文件操作	552
18.4.6	与对话框控件的交互	490	19.6	路径、目录和文件	553
18.5	AddressBook 程序	491	19.6.1	Directory 和 File 类	553
18.6	基于对话框的程序	495	19.6.2	DirectoryInfo 和 FileInfo 类	555
18.6.1	计算器程序分析	496	19.7	流和存取文件	557
18.6.2	设计 MyCalculator 类	497	19.7.1	FileStream 对象	557
18.6.3	设计对话框类 CCalculatorDlg	498	19.7.2	StreamReader 类	560
18.7	通用对话框	502			
18.7.1	文件对话框类	502			
18.7.2	字体对话框类	503			

19.7.3 StreamWriter 类	561	20.3.5 导出变量	575
19.8 访问二进制文件	562	20.3.6 导出类	576
第 20 章 动态链接库与钩子	564	20.3.7 DLL 的入口函数	579
20.1 了解 DLL	564	20.4 MFC 规则 DLL	580
20.2 静态链接库	566	20.4.1 静态链接 MFC DLL 的规则 DLL	580
20.3 常规 DLL	568	20.4.2 共享 MFC DLL 的规则 DLL	582
20.3.1 一个简单的常规 DLL	569	20.5 MFC 扩展 DLL	583
20.3.2 声明导出函数	571	20.6 Windows 钩子	586
20.3.3 DLL 的调用方式	573	20.6.1 认识 HOOK 钩子	586
20.3.4 调用约定与名称改编	574	20.6.2 钩子的安装与卸载	587
		20.6.3 键盘钩子	588

第四篇 数据库访问

第 21 章 使用 ODBC 数据源	593	21.7.1 添加订单的逻辑处理	616
21.1 数据库访问技术	593	21.7.2 创建记录集视图	617
21.1.1 ODBC API	593	21.7.3 对话框切换	620
21.1.2 ODBC 的 MFC 类	594	21.7.4 创建订单编号	621
21.1.3 DAO 与 RDO	594	21.7.5 订单数据	623
21.1.4 OLE DB 与 ADO	595	21.7.6 为订单选择产品	624
21.2 MFC ODBC 类	596	21.7.7 添加新订单	625
21.2.1 CDatabase 类	596	第 22 章 数据库和 ADO.NET	628
21.2.2 CRecordSet 类	597	22.1 ADO.NET 概述	628
21.3 注册数据源和示例数据库	597	22.1.1 ADO.NET 的设计目标	628
21.4 创建数据库应用程序	599	22.1.2 ADO.NET 体系结构	629
21.4.1 了解记录集	600	22.1.3 System::Data 命名空间	630
21.4.2 了解记录视图	603	22.2 使用 DataReader	632
21.4.3 创建视图对话框	605	22.3 使用 DataSet	636
21.4.4 对记录排序	606	22.3.1 读取数据	636
21.5 多表操作	607	22.3.2 修改数据	639
21.5.1 添加记录集	607	22.3.3 保存 DataSet 对数据的修改	643
21.5.2 添加记录视图类	608	22.4 在 DataSet 中访问多个表	645
21.5.3 过滤数据	609	22.4.1 ADO.NET 中的关系	645
21.5.4 显示多个记录视图	610	22.4.2 导航关系	645
21.6 更新操作	613	22.5 使用数据控件	648
21.6.1 更新 CRecordSet 记录集	614	22.5.1 DataGridview 控件	649
21.6.2 事务	615	22.5.2 数据绑定	654
21.7 向表添加数据	616		

第一篇 Visual C++ 2008 基础知识

第 1 章 开始使用 Visual C++ 2008

在 Visual C++ 2008 中可以采用多种方式编写 Windows 应用程序，既可以编写基于本地 C++ 的 Windows 程序，也可以在托管环境下开发 Windows 程序。在接触 Windows 编程之前，必须熟悉 C++ 编程语言，特别是 C++ 语言的面向对象功能。

本章主要概述了 C++ 编程涉及到的一些基本概念，以及 C++ 的集成开发环境 Visual Studio 2008 的使用。

- 了解 .NET Framework 3.5 的概念
- 熟悉 C++ 应用程序
- 创建控制台应用程序
- 编译、连接并执行 C++ 控件台应用程序
- 创建并执行基本的 Windows 应用程序

1.1 .NET Framework 3.5

.NET Framework 是 Visual C++ 2008 以及微软所有其他 .NET 开发产品的核心，它是微软近年来主推的应用程序开发框架，该框架提供跨平台和跨语言的特性。使用 .NET Framework，并配合其集成开发环境 Visual Studio，开发人员可以比以往更轻松地创建出功能强大的应用程序。

1.1.1 什么是 .NET Framework

现在的计算机编程语言的执行方式分为两种，一种是编译执行，一种是解释执行。编译执行是指源程序代码先由编译器编译成可执行的机器码，然后再执行；解释执行是指源代码程序被解释器直接读取执行。

上面这些都是比较传统的程序代码执行方式，从 Java 语言开始，一种新的程序语言执行方式产生了，这就是“中间码+虚拟机”执行机制。在这种执行方式中，程序语言源代码需要被编译成一种特殊的中间码，这种中间码是不能直接在机器上执行的，它需要一个叫“虚拟机”的装置来管理和执行，虚拟机可以是解释执行，也可以是编译执行。因为“虚拟机”可以参与和管理程序代码的执行，因此解决了很多传统编译语言一些致命的缺点，如垃圾内存回收、安全性检查和跨平台等。正因为 Java 有如此优点，微软也采用了这种执行方式，.NET

Framework 类似于管理和执行中间码的“虚拟机”。

需要注意，.NET Framework 和 Java 的虚拟机 JVM 是不完全相同的，Java 的虚拟机是解释执行的，而 .NET Framework 是编译执行的。另外，.NET Framework 作为开发应用程序的一个框架，它对操作系统进行封装，需要使用 .NET Framework 开发的应用程序与操作系统特性隔离开来。这样，.NET Framework 开发的应用程序就可以移植到许多不同的硬件和操作系统上。

事实上，.NET Framework 的主要特色在于简化应用程序开发的复杂性。它提供了一致的开发模型，开发人员可以选择任何支持 .NET 的编程语言来进行多种类型的应用程序开发，例如 Visual Basic.NET、C# 和 C++.NET。

.NET Framework 由两个主要部分组成：CLR（Common Language Runtime，公共语言运行库）和一组供使用的 .NET Framework 类库。

1.1.2 公共语言运行库 CLR

公共语言运行库 CLR 是标准化的程序执行环境。这些可以在 CLR 中执行的程序可以用各种高级语言编写的，比如 Visual Basic、C#，以及 C++。现在 CLR 规范收录在 ECMA（欧洲计算机制造）的 CLI（通用语言基础结构）标准——ECMA-335 中，并且 CLI 已经通过 ISO 认可，作为 ISO/IEC 23271 的标准。也就是说，CLR 是微软对 CLI 的实现。这也就是为什么将支持 CLR 的 C++ 称为 C++/CLI。

CLI 本质上是一种虚拟机环境规范，这种环境规范使各种高级编程语言编写的应用程序能够在不同的系统环境中执行，而不用修改或重新编译原来的源代码。CLI 规定了一种供虚拟机使用的标准中间语言，高级语言源代码首先将被编译为这种语言。在 .NET Framework 中，这种中间语言称为 MSIL（Microsoft 中间语言）代码。执行程序时，中间语言代码最终由 JIT（Just-In-Time，实时）编译器编译为本地机器代码。当然，CLI 中间语言代码可以在任何实现 CLI 标准的环境中执行。

在开发应用程序时，使用一种高级语言编写的程序很难与使用另一种高级语言编写的程序进行数据交换，这是许多编程语言存在的一个共同问题。为此，CLI 定义了一组通用的数据类型——CTS（Common Type System，通用类型系统），在使用由 CLI 实现的编程语言编写程序时都应该使用该类型系统。这样，CLI 中的各种高级语言都会使用相同的数据类型系统，从而使得不同编程语言编写的组件以相同的方式处理数据，也可以使得将不同语言编写的组件集成到单个应用程序中。

CLR 大大增强了数据的安全性和程序的可靠性。因为在 CLR 中运行的程序是托管的，即 CLR 管理着应用程序，它会对程序的内存进行管理，并对程序执行的安全性进行检查等。与此相反，不在 CLR 控制下运行应用程序是非托管的。非托管程序可以直接访问操作系统的低级功能，对于托管程序则是通过 CLR 与操作系统通信。

1.1.3 .NET Framework 类库

关于类库的概念一直就存在，以前的 Visual C++ 为 MFC 类库，Delphi 的类库为 VCL，

Java 的类库为 Swing、AWT 等。这些类库封装了系统底层的功能并提供更好的操作方式。.NET Framework 中的类库封装了对 Windows、网络、文件、多媒体的处理功能，是所有 .NET Framework 语言都必须使用的核心类库。并且，为了便于语言之间进行交互操作，.NET Framework 类库中的类型都是符合公共类型系统 CLS 的。使用类库可以创建多种类型的应用程序，极大简化了开发人员的学习曲线，提高了软件开发生产力。

1.1.4 .NET Framework 3.5 的新功能

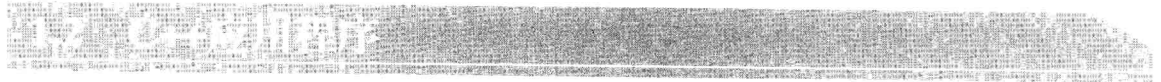
从微软公司发布第一个 .NET Framework 以来，已经陆续发布了 1.0 版、1.1 版、2.0 版和 3.0 版。.NET Framework 3.2 版是目前最新的版本，也是功能最强大和完美的一个版本。开发人员可以使用 .NET Framework 创建 Web 程序、Windows 以及智能设备应用程序等。

Visual C++ 2008 所使用的 .NET Framework 为 3.5 版。.NET Framework 3.5 版是以 .NET Framework 2.0 版和 .NET Framework 3.0 版为基础的，包括 .NET Framework 2.0 和 3.0 版的 Service Pack。

.NET Framework 2.0 改进了 .NET Framework 1.1 中的许多问题，并增加了如泛型、可空类型、匿名方法等新特性。.NET Framework 3.0 则以 .NET Framework 2.0 为基础，增加了 3 种全新的技术。

- **Windows Presentation Foundation(WPF)** Windows 表现层技术。
- **WCF** Windows 通信技术。
- **WF** Windows workflow 开发技术。

.NET Framework 3.5 则以 3.0 为基础，增加了对 ASP.NET AJAX 的直接支持，并提供了语言集成查询 LINQ 技术以及一些附加的类库。



使用 Visual C++ 2008 可以开发两种类型的应用程序，即在 CLR 中执行的托管程序，也可以编写直接编译为机器代码的非托管程序。就针对 CLR 的、基于窗口的托管应用程序而言，可以使用 .NET Framework 类库提供的 Windows Forms 作为 GUI 基础。使用 Windows Forms 可以快速开发 GUI 程序，因为 Visual C++ 2008 开发环境提供了直观的组件来开发 GUI，而代码是完全自动生成的。

对于本地执行的非托管代码，则有多种方式可用。一种是使用 MFC (Microsoft Foundation Classes, 微软基本类) 来编写 Windows 应用程序的图形界面。MFC 封装了 Windows 操作系统提供的用于创建 Windows 应用程序的 API (Application Programming Interface, 应用程序编程接口)，因此大大简化了程序的开发过程。如果应用程序需要最佳性能，还直接使用 Windows API 开发 Windows 应用程序。

由于托管 C++ 是在 CLR 中执行的，其数据和代码都由 CLR 管理，所以在其中为数据动态分配的内存是自动释放的，这就消除了本地 C++ 应用程序中常见的错误。对于非托管的 C++ 程序，因为 CLR 不参与这种代码的执行过程，因此程序员必须自己处理程序执行过程中分配

与释放内存的各个方面。另外，还得不到 CLR 提供的安全性检查。非托管 C++ 也被称为本地 C++，因为其代码被直接编译为本地机器代码。各种 C++ 程序的特点如图 1-1 所示。

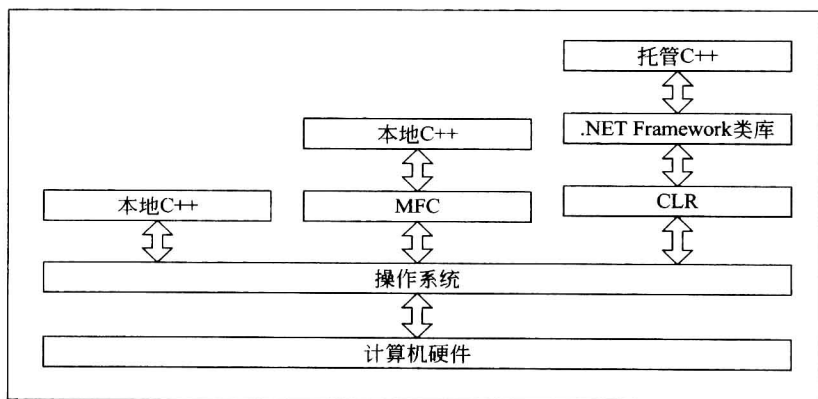


图 1-1 C++ 程序类型

图 1-1 列出的程序类型之间并不是完全独立的。应用程序可以部分代码采用托管 C++，而另一部分则使用本地 C++ 代码。当然，混合使用代码会造成一些损失，因为托管代码组件与非托管代码组件之间通信可能会产生相当大的系统开销。因此，只有当需要开发或扩展非托管代码，而又希望使用托管代码的优点时，混合两种代码才有意义。对于新的应用程序而言，在最初就应该决定是否采用托管 C++ 来开发程序。

为了以后学习的方便，需要全面理解 C++，包括语言的本地 C++ 和 C++/CLI 版本。以下是 Visual C++ 2008 支持的两个独立标准定义的 C++ 版本。

- **ANSI/ISO C++** 也称为标准 C++，用于实现本地非托管的 C++ 应用程序。标准 C++ 很早就出现了，多数计算机平台都支持该版本的 C++。ANSI/ISO C++ 一直是许多专业程序开发者的首选，而且目前仍然是功能最强大的编程语言之一。
- **C++/CLI** 运行在 CLR 中的托管 C++，它是 ISO/ANSI C++ 的扩展，以便更好地支持 ECMA-335 标准定义的 CLI。该标准是在 2003 年微软公司为了支持 C++ 程序而在 .NET Framework 内执行而提出的。虽然 C++/CLI 是 ISO/ANSI C++ 的扩展，但是如果希望程序完全在 CLR 控制下执行，那么有些 ISO/ANSI C++ 的功能绝对是不能使用的。

1.3 Visual C++ 2008 与 Visual Studio 2008

本书使用 Visual Studio 2008 进行所有的开发，包括简单的命令行应用程序，以及比较复杂的项目类型。尽管 Visual Studio 2008 不是开发 C++ 应用程序所必需的，可以使用基本的文本编辑器（如常见的记事本程序）编辑 C++ 源代码，再使用 C++ 编译器对代码进行编译。但使用 Visual Studio 2008 可以使任务更简单一些。

Visual Studio 2008 是一个完整的开发环境。Visual C++ 2008 只是 Visual Studio 2008 的一部分，在 Visual Studio 2008 中还可以使用 Visual Basic.NET 和 Visual C# 语言进行开发。

如果是第一次运行 Visual Studio 2008，则屏幕上会显示一个如图 1-2 所示的“选择默认环境设置”对话框。

在其中选择“Visual C++ 开发设置”选项，表示使用 Visual C++ 的开发环境。然后单击“启动 Visual Studio”按钮，系统将花几分钟的时间使配置环境生效。

如果不是第一次运行 Visual Studio 2008，并且选择了另一个选项，为了将设置重置为 Visual C++ 开发环境，可以在打开 Visual Studio 之后，选择“工具”菜单中的“导入和导出设置”选项，打开配置文档管理向导，如图 1-3 所示。在该对话框中可以选择导出当前的环境设置，或导入选定的环境设置，这样通过导出配置文档，可以建立并共享自己的配置文件。

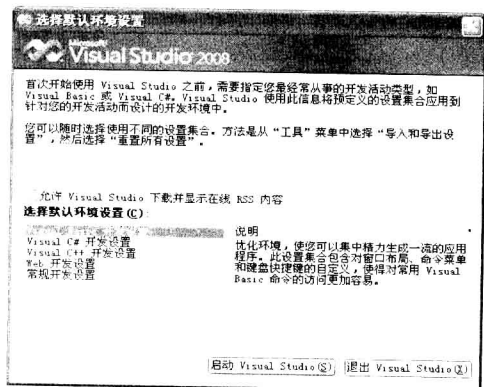


图 1-2 “选择默认环境设置”对话框

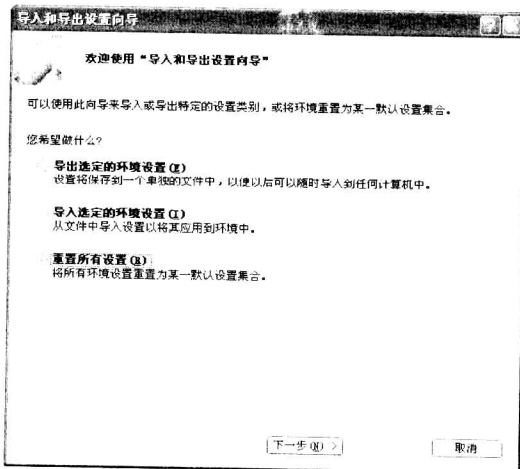


图 1-3 配置文档管理向导

为了重新设置当前的开发环境，选择“重置所有设置”选项，然后单击“下一步”按钮，屏幕上会出现如图 1-4 所示的对话框，这个画面有两个选项，一是保存现有设定，二是放弃保存现有设定。用户可以按照自己的需要任意选择，然后单击“下一步”按钮。

接下来就会出现如图 1-5 所示的对话框，在其中可以重新选择默认的环境配置。

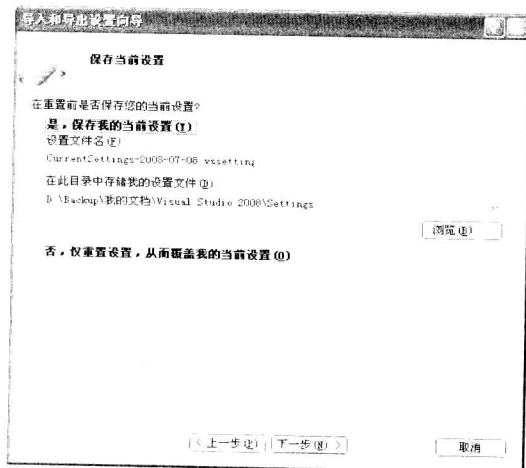


图 1-4 导入和导出设置

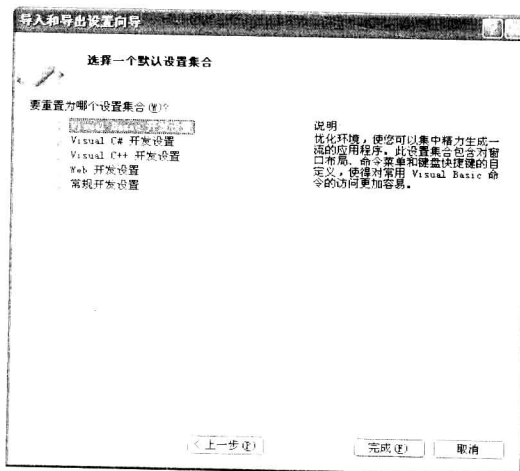


图 1-5 重新选择默认环境配置