# 第一章　Visual C++ 2.0 介绍

Microsoft Visual C++编译器软件包为所有的 Windows 和 MS-DOS 应用程序提供了最高级、最新开发的环境。出自 Microsoft 的最新版本(2.0)综合了一些新的和升级的特点。其中最重要的进步是支持 AT&T C++2.1,同时还有些新特点,比如预编译头文件、自动直接插入以及 P 代码(压缩代码)。Microsoft 还可以直接混合位图、图标、光标和对话框编辑器以进入集成环境。

Microsoft Visual C++编译器软件包还可以为建立 Windows 程序提供工具,C++编译器包括所有的头文件、库文件以及用于生成一个真实稳定可靠的 Windows 应用程序所必需对话框及资源编辑器。

这一章中读者将了解C++编译器的多种组件、系统需求及怎样建立开发环境。本章还解释了 Microsoft Visual C++系统,并且指导读者如何正确调试以满足特定的需求。

在这本书的剩余部分详细地介绍了所要讨论的一些主题,所有的章节将介绍这一章中提所及的先进工具。

## 1.1　硬件需求

这一章提供关于硬件和软件的建议,有助于获得大量的 Microsoft Visual C++编译器的信息。许多建议只为了提高整个系统的性能,而另外一部分则是为了使产品更易于使用。

### 1.1.1　最小硬件和软件需求

Microsoft 的 Visual C++编译器软件包将运行在基于 Intel 计算机这样一个较宽范围之上。这里列出了运行 Microsoft Visual C++编译器软件包所必需的"最小"硬件和软件要求。

(1) 一个基于 80386 的 PC

(2) 16M 的 RAM

(3) 一个高密度的软磁盘驱动器

(4) 一个 CD-ROM 驱动器(用于联机文档)

(5) 11,744K 的内存(用于最小 16 位设置)或者 70,304K 的内存(用于典型的 16 位设置),或是 63,888K 的内存(用于最小 32 位设置),或 125,856K 内存(用于典型 32 位设置)

(6) 用于更高版本的 MS-DOS 5.0,Microsoft Windows 3.1 或更高版的 Win32s

(7) 用于 Win32 开发的 Microsoft Windows NT 3.5 或更高版

(8) 一个 VGA 监视器

### 1.1.2　硬件和软件需求

对于使用简便、性能及整套产品的受欢迎程度而言,最小的硬件和软件要求并非总是最

优的选择。下面推荐系统需求来优化C++程序的开发周期。

(1) 一个以 50MHZ 运行(或者更高)基于 80486 或 Pentium 的 PC

(2) 20MB 的 RAM

(3) 一个 300MB 的硬盘

(4) MS-DOS 6.2 和 Windows 3.1(或更高),或者是 Microsoft Windows NT 3.5(或更高)

(5) 一个 VGA 或者更高分辨率的监示器

(6) 一个高密度软盘驱动器(3.5")

(7) 一个 CD-ROM 驱动器(用于联机文档)

还可以需要一个处理高级 Windows 应用程序大小和复杂性的快速微处理器。并且因为有大量的内存就能得到最大的 Microsoft Visual C++及 Windows 环境的全部性能(当然还可以通过大量的自由磁盘空间来获得性能来增强)。

MS-DOS5.0 或者更高版本提供一些新特征,使内存结构更加简单。如果还未升级到 5.0版本或更高,在为 Microsoft Visual C++安装系统文件前最好先这样做。在这本书中贯穿的几项性能推荐要求系统具有这些新的 MS-DOS 来增强。

Windows3.1 和 Windows NT 3.5 的增强特征和性能可让用户生成完美的 Windows 应用程序。在 Windows 图像环境中工作时会最好使用 VGA 或高分辨率监示器。

## 1.2 选择正确的安装选项

这部分描述了 Visual C++开发系统及 Win32s 文件的基本安装。Visual C++程序所提供的安装程序执行全部安装 Visual C++组件所必需的工作。

下面是针对 Visual C++的两个安装程序:

(1) WIN32s\SETUP.EXE,Win32s 安装程序

(2) \MSVC20,SETUP.EXE,Visual C++安装程序

Visual C++安装程序可被用来安装 Visual C++,从 CD-ROM 驱动器或网络中运行。

Win32s 安装程序被用于安装:

(1) Visual C++远程调试工具

(2) Windows OLE DLLs

(3) Wind32s 系统 DLLs

(4) Win32s 工具

### 1.2.1 使用何种设置

如果开发 Win32s 应用程序,从 Windows 中运行 Win32s\WIN32S\SETUP.EXE程序,在目标计算机中安装 Win32s。

如果仅开发针对 Windows NT 的目标应用程序,运行 Visual C++\MSVC20\SETUP.EXE 程序。必须从 Windows NT 中运行这种安装程序。

### 1.2.2 目录

下面列出了 Visual C++目录\MSVC20 子目录的内容：

| | |
|---|---|
| \MSVC20\BIN | 可执行文件及建立 32 位应用程序所需的工具 |
| \MSVC20\DEBUG | 所需调试文件用以再分配给接受该软件的用户 |
| \MSVC20\HELP | Visual C++帮助文件 |
| \MSVC20\INCLUDE | C++运行时间及 Microsoft Win32 软件开发工具(SDK)头文件 |
| \MSVC20\LIB | C++运行时间及 Win32 SDK 库文件 |
| \MSVC20\MFC | 针对 Microsoft Foundation Class(MFC)库文件的子目录 |
| \MSVC20\REDIST | 再分配给接受该软件的用户所需的文件 |
| \MSVC20\SAMPLES | 样本程序 |

Win32s 文件位于下面列出和描述的\WIN32S 子目录中：

| | |
|---|---|
| \WIN32S\BIN | Win32s 的调试工具，用于 Win32s 的 Microsoft Profiler 和远程调试文件。 |
| \WIN32S\DEBUG | Win32s DLLS 的调试版本 |
| \WIN32S\NLS | 国内语言支持(NLS) |

## 1.3 典型的 Widnows 安装

可以配置 Microsoft Visual C++在 Windows NT 环境、MS-DOS 和 Windows 环境以及二者之下运行。充分利用这本书中提供的全部样本程序来安装在两种环境下操作的 Visual C++(使用\MSVC20\SETUP.EXE 程序)。

如果有足够的磁盘空间则应该选择"典型"安装。这将提供所有帮助文件、例子及所需的支持文件，以及尽可能不费力气地向 Microsoft Visual C++过渡。

确保在安装前使 Windows 运行。通过运行 Widnows Program Manager 来开始安装，选择 File|Run 菜单项，并输入：

```
drive:\MSVC20\SETUP.EXE
```

这是用于典型的 Windows NT 安装，或者是

```
drive:\WIN32S\SETUP.EXE
```

这是针对典型的 Windows3.1 安装。

整个过程大约需要 45 分钟，所以在开始前不妨准备一杯咖啡。下面列出了选择缺省系统安装所必需的步骤：

(1) 按下 ENTER 键继续安装。

(2) 按下 ENTER 键继续编译器安装。

(3) 选择"典型"安装(定制安装所允许增加的附加库文件及选择不同的系统配置)。

(4) 按下 ENTER 键以接受这一选项然后继续。

(5) 此时，提示告知 README.TXT 文件在安装过程中被自动装入。这个提示敦促程序员读取该过程其余部分中的信息。

(6) 重引导用户系统。

## 1.4 文档

Visual C++联机文档由 Quick Reference 和 BooksOnline 组成。Quick Reference 允许用户在编程时快速查阅信息。Books Online 是为 Visual C++在联机格式下设制的文档。每一个 Quick Refence 主题目都联接 Books Online，在此可以获得完整的信息。

Visual C++将 Quick Reference 文件安装在磁盘上，这时 Books Online 驻留在 CD-ROM 上。可以定制安装文件或者获得信息的区域，或直接进入 Books Onlone 上下文有关的帮助(F1)。

所覆盖的主题如下：

(1) 怎样使用 BooksOnline
(2) 用户指南
(3) Microsoft Foundation Classes(MFC)
(4) 用 Microsoft Foundation Class 库程序设计
(5) Class 库参考
(6) MFC 样本
(7) MFC 技术说明
(8) C/C++
(9) 程序设计技术
(10) C 语言参考
(11) C++语言参考
(12) Run-time 库参考
(13) Iostream 参考
(14) 预处理器参考
(15) C/C++样本
(16) Win32 Software Developmet Kit(SDK)
(17) API 32 功能
(18) 开放 GL 功能
(19) Win32s 程序员参考
(20) Windows 接口
(21) OLE 2.0 软件开发工具
(22) OLE 2 程序员参考，卷 1
(23) OLE 2 自动程序员参考，卷 2

## 1.5 开发系统

Microsoft Visual C++开发系统对于 Windows 和 Windows NT 合成了新的、全面集成的 Widnows 开发工具及可视性接口。例如，Microsoft 的 CodeView 的调试能力可以直接从开发环境的集成调试器内得到。下面部分列出那些可以被直接集成到 Microsoft Visual

C++ Workbenck 的独立实用工具。

### 1.5.1 新型集成调试器

Microsoft 运用新的集成调试器把 CodeView 的功能直接放入到 Visual C++ Workbench。集成调试器允许用户单步执行应用程序，观察及改变变量甚至追溯代码。

### 1.5.2 新型集成 Dialog Image 和 Hotspot 编辑器

Dialog、Image 以及 Hotspot 编辑器允许用户定制一个应用程序接口。允许用户创造具有视觉吸引力的、色彩丰富及带鼠标的资源，比如对话框、图标及光标。

1. DIALOG、EDITOR

Dialog Editor 是一种直接的图表式开发工具，允许用户简单而迅速地生成具有专业处理的对话框。事实上 Microsoft 所做的是提供一种自由指令，另外还有优良的开发环境---Visual Basic。Dialog Editor 允许用户定制对话框标签、框架、选项及复选框选择、文本窗口和滚动条。

如果使用过 Visual Basic，则能 90%的了解 Dialog Editor 是怎样工作的。工具框为程序员提供了 14 种使用在应用程序中的控件。一个控件把表示性能的可视图形和由用户定制的一组预定义特性结合起来。

例如，一些对话框使用水平或垂直的滚动条。Dialog Editor 允许程序员用鼠标从工具框中选择控件，并且放置于对话框中，同时还可以用鼠标改变滚动条的尺寸及位置。再用鼠标的单击，还可以从以下所列各项中选择滚动条类型：Visuable、Disabled、Group 或 Tabstop。

2. IMAGE EDITOR

图形 Imamge Editor 允许程序员很容易地生成定制的位图、图标及光标。位图是这样一些东西的图形——例如，使用在一条警告信息中的注释点。图标是一个小的彩色图像，被用来代表一个已被最小化的应用程序。Microsoft C++甚至允许使用 Image Editor 来生成定制光标。例如，可以设计一个带有光标的经济软件包，这个光标看上去要像一个美元符号。定制图标、光标及位图可以存储为带.RC 文件扩展名，并且用于资源脚本的文件中。

3. HOTSPOT EDITOR

Hotspot Editor 允许生成和编辑含有一个热点的光标。热点是在保存带 Windows 图形活动位置的定制光标图形中的确切像素或相关坐标。这个坐标用以指示可视界面中鼠标的位置。

### 1.5.3 Spy++和 DDESpy

Spy++是一种工具，它可以给出系统进程、线程、窗口、及窗口消息的图形视图。DDESpy 被用于 Microsoft Windows 环境中动态数据交换活动的监视程序。除了监视特定的消息类型，DDESpy 也能跟踪特定的串句柄、会话、连接及服务，通过击 Track 菜单上特定的对象类型可以激活跟踪。

### 1.5.4 MC

MC 是 Miscellaneous Tools Qutck Reference 工具的缩写，Quick Reference 可以在编程时提供快速参考信息。例如，当一个关键字或函数在源文件中以高亮度显示时按下 $F_1$，这时就打开了 Quick Reference 标题，它提供了有关关键字或功能的基本信息。这个标题可以提供以下信息：

(1) 跳向相关功能组或父类
(2) 从 Quick Reference 中的 Example 按钮中打开一个例子
(3) 如果适用，则兼容性怎样
(4) 参数描述
(5) 类型或语法
(6) 返回值

Quick Reference 也提供"怎样"信息及有关大量附加条目的的上下文敏感帮助：

| 标题 | 选择条目 |
|---|---|
| 建立错误 | Build Errors |
| C/C++ 语言 | C/C++ Langluage |
| C/C++ 运行时间库 | Run - Time Routines |
| 数据库类 | Foundation Classes |
| iostrecam 类库 | iostream Classes |
| Microsoft Foundafion Classes | Foundation Classes |
| 模块定义文件语句 | Miscellaneous Tools |
| ODBC 应用程序接口 | ODBC API(仅对 Intel) |
| OLE 2.0 应用程序接口 | OLE API |
| OLE 2.0 类 | Foundat ion Classes |
| 资源文件语句 | Miscellaneous Tools |
| 使用联机文档 | Using Online Dolumentation |
| Visual C++ 工具 | Visual C++ |
| Win 32 应用程序接口 | Windows API |

### 1.5.5 剖析程序

Visual C++ Profiler 是一个有用的分析工具，用于估计应用程序的运行时行为。Profiler 的输出允许程序员找到高效率运行的代码部分及那些需要精炼的代码部分，另外，Profiler 标志那些不被执行的代码部分。

Profiler 被用来让程序运行得更好而不是发现错误，一旦程序相当稳定，则应该去搜索那些需要引起注意以优化代码的地方。用 Profiler 来判定一条算法是否有效、一个函数是否被频繁调用(或者反之)、或者一个代码是否未达到或根本未执行。

可以从 Visual C++ 开发环境或者命令行中运行 Profiler。对于从命令行中使用 Profiler 的有关信息可以参看 Books Online 帮助参考。关于 Win32s 中剖析应用程序的一些信息，参见 Books Online 中的 Profling Under Win32s。

### 1.5.6 PortTool

这个工具是在 C:\MSVC20\BIN 下,是将程序从 Windows3.1 移植到 Windows NT 的一个工具。

考虑 Win32 应用程序,它允许调用 Win32 函数。在 Windows 中不被支持的 Win32 函数通常返回错误值。

移植工具和数据文件,WIN32S.DAT 列出 Win32S 不支持的所有 Win32 函数,并且允许程序员判断应用程序是否已经不小心引用了不被支持的 Win32 函数。把 WIN32S.DAT 读入移植工具,取出应用程序的源代码,然后让移植工具检查源。

### 1.5.7 Process Viewer

Process Viewer 对话框允许快速设置及观察所有的追踪当前过程,以及分处理器时所必需的选项。要启动 Process Viewer,只需在 Visual C++组中双击 PView 图标。

UProcess Viewer 可以帮助回答一些问题,比如在程序执行中不同点上能给该程序分配多少的内存,多少内存被调出页面?"哪一些进程和线程使用了最多的 CPU 时间?""程序在不同的系统优先级中,进程在响应 DDE、OLE 或管道 I/O 时而暂停,将会发生什么情况?"

### 1.5.8 WinDiff

在 Visual C++组下的 WinDiff 工具,允许程序员图形化地比较和修改两个文件或两个目录。在 WinDiff 中的所有选项的操作很像它们在 Windows File Manager 中的命令。

## 1.6 重要的编译器特性

Microsoft 已经用一些有用的提高、新特征和选项来压缩 Visual C++编译器软件包。下面章节简要介绍了这些进步,用解释其用法。

### 1.6.1 P-Code

Microsoft 的一个最新技术就是 P 代码(是"Packed Code"的缩写),它集中于优化代码的速度和尺寸。P 代码可以明显地减少程序大小以及提高执行速度,达 60%之多。更好的是,要完成这些只需简单地打开一个特定的编程选项。这就意味着写在 C 或C++中的任何代码既可以用通常形式也可以用 P 代码形式。

这项技术把一个应用程序的源代码编成"解释性目标代码",这是一个高标准且更具压缩性的目标代码的表示。要完成该过程只须把一个小的解释模块连接进应用程序中去。

不过要更加有效地使用这项技术确实需要一些经验,因为解释程序在运行时间生成目标代码。P 代码运行速度比本来的目标代码要慢得多。仔细使用 #pragma 指令,应用程序可以生成针对空间临界的 P 代码,并且切换回速度临界以生成源代码。

对 P 代码生成的最佳候选者是那些处理用户接口的例程,因为一些 Windows 应用程序要花去 50%的时间处理用户接口,P 代码则提供了优化的处理特性。

### 1.6.2 预编译头文件和类型

C++在被称为头文件的特殊文件中放置通用类型、函数原型、外部引用以及成员函数声明。这些头文件包含一些多个源文件所需的关键定义。多个源文件汇总在一起来生成程序的可执行版本。对于包含头文件的每个模块,部分头文件被典型地重编译。不妙的是,部分代码的重复编译会造成编译器速度缓慢。

Microsoft Visual C++可以通过预编译头文件加速这一过程。预编译头文件使用并不新,而 Microsoft 补充的方法是一种新方法。预编译存储编译状态到一个给定点,并且再现建立在源文件及预编译头文件中的关系,可以为每个源文件生成不止一个预编译头文件。

这项技术最好的应用,包括一个具有常用代码转换但无常用基类定义的应用程序的开发周期。如果头文件被预编译,编译器可以把时间集中在源代码转换上。预编译头文件也为带头文件的应用程序提供一个编译时间帮助,这个头文件包含了常同C++程序同时出现的一个给定模块的大部分代码。

Microsoft Visual C++编译器假定编译器环境的当前状态同任何预编译头文件被编译时相同,编译器如果判别出任何非一致性就会发出警告。这种非一致性将引起内存模式的变化、已定义常量的状态的变化,或者是不同的调试项或者代码生成选项的变化。

不像一些平常的C++编译器,Microsoft Visual C++编译器不限制对头文件的预编译。因为这一过程允许预编译程序到设定点为止,因而甚至还能预编译源代码。对于在头文件中包含大部分成员函数定义的C++程序这是特别有用的。通常预定义保留在被认为是稳定的算法部分;它是设计来最小化编译程序所需时间的。

### 1.6.3 基础类库

几乎每个人都同意这一点:一个书写正确的 Windows 应用程序是易于使用的。然而,这类程序并不易于开发。为此许多程序员就不得不精通超过 500 个的 Windows 应用程序界面函数,这就是写一个 Windows 应用程序所必需的。

Microsoft 用来解决上面难题的方法是使用 MFC。可重用的类理容易掌握和使用。MFC 充分利用了C++提供的数据抽象的特性,它的使用简化了 Windows 编程。初级程序员可以像用"烹调全书"抄菜那样照搬照抄,有经验的C++程序员可以扩展该类或者把它们混合进自己的类层次中。

MFC 库提供用于管理 Windows 对象的类,并提供许多可以同时用于 MS-DOS 和 Windows 的通用类。比如,创建和管理文件、字符串、一致的存储和异常处理都是通用的类。

实际上,Microsoft Foundation Class 库虚拟表示了每个 Widnows API 的功能,包含了用于简化消息处理、诊断及所有 Windows 应用程序都需要重复处理的理的一般问题的高级代码。Windows API 函数的逻辑组合和增强的优点主要有以下九个方面:

(1) Windows API 的封装是逻辑的、完整的。MFC 库为所有的经常使用的 Windows API 性能提供支持,包括窗口功能、消息、控件、单、对话框、GDI(图形设置接口)对象(字体、画刷、画笔及位图),对象链接以及多文档界面(MDI)。

(2) MFC 很容易学。Microsoft 做了最大的努力以保证 MFC 函数和相关的参数的名字与 Windows API 父类相一致。这极大的减少了有经验的 Windows 程序员想利用能简化工作

的 MFC 平台的优点时所可能带来的混乱。它也可以帮助初级 Windows 程序员利用 MFC 这个 Windows API 的超集来完成自己的工作。

(3) C++代码更加有效。当在小内存模式下编译 MFC 库中的类时，一个应用程序只耗用少许的额外 RAM。一个 MFC 应用程序的执行速度几乎同使用标准 Windows API C 语言所编写的程序相等。大部分的 MFC 应用程序的运行速度仅比相应的 Widnows API C 语言程序慢 5%——考虑到 MFC 有助于缩短程序设计开发周期，这是非常值得的。

(4) MFC 库提供了自动消息处理。Microsoft Foundation Class 库消除了最易产生编程错误的来源，即 Windows API 的消息循环。MFC 库被设计来自动处理每一条 Widnows 消息。代替使用标准 switch-case 语句，每一条 Windows 消息被直接映射到一个进行处理成员函数。

(5) MFC 库允许自诊断。MFC 库实现了自诊断的能力，这就意味着可以在一种易于理解的格式下把有关多个不同对象的信息转储到文件中去，并且验证一个对象的成员变量。

(6) MFC 库具有一个稳定的体系结构。因为事先考虑到 ANSIC 的 throw/catch 标准，Microsoft Foundation Class 库已经实现了一个扩展的异常处理体系结构。这允许一个 MFC 对象可从标准错误状态恢复过来，这些标准包括"内存溢出"错误、无效选项、文件或资源加载问题，在体系结构中的每个成员同 ANSI C 建议标准能提供向上兼容。

(7) MFC 库提供动态对象类型。这一特别有用的性能把动态分配对象类型确定延迟到运行时才进行。这样便可以不必担心一个对象的数据类型而进行处理。因为关于对象类型的信息在运行时被返回，程序员可以减少许多细节处理。

(8) MFC 库可以和协地同以 C 语言为基础的 Windows 应用程序共存。Microcoft Foundation Class 库最重要的特征是同以 C 语言为基础的使用 Windows API 的 Windows 应用程序共存的能力。在同一个程序中，程序员可以同时使用 MFC 类和 Windows API 调用。这样，就可按经验和需要在一个使用 MFC 的应用程序加进真正的C++面向对象代码。上面透时的环境成为可能，是因为在两个体系结构使用了公共的命令协议。这就意味着 MFC 头文件、类型及全局定义与相应的 Windows API 名字不冲突。透明的内存管理是达成上面和谐关系的关键因素之一。

(9) MFC 库可以和 MS-DOS 应用程序一起使用。Microsoft Foundation Class 库是专门用来设计 Windows 应用程序的。然而，许多类提供了使用频繁的 I/O 文件和串操作所需要的对象。因此，这些通用类既可在 Windows 应用程序中使用，也可在被 MS-DOS 应用程序中使用。

### 1.6.4 函数嵌入

Microsoft Visual C++编译器支持完整的函数嵌入，意味着任何类型的函数或者指令组合可以被扩展成行。一些常用的C++编译器对某种类型的语句或表达式限制嵌入——例如，嵌入选项会被包含 switch，While 或 for 语句的函数所忽视。Microsoft 的C++编译器允许嵌入速度优化的例程(包括很少有使用的类成员函数或构造函数)，并且不限定其内容。通过 Project 菜单选择 Settings...，然后是 C/C++文件夹，最后从 Category 列表中选择 Optimizations 来设置这个选项。

## 1.7 编译器选项

Microfosft Visual C++是一种全局优化编译器，允许利用几种速度或代码尺寸选项进行每类程序的开发。下面的编译器选项允许针对可执行尺寸、速度或建立时间来优先代码。如果没有看见一个可观的性能改善，则有可能测试应用程序并不包含足够的代码。所有的选项都是选择 Settings...项中的 Project 菜单来设置的；做完这些之后，将会显示 Project Settings 对话框。

### 1.7.1 C 语言

下面是一些条目，它们在 C 语言应用程序中常有益于优化。从 C/C++文件夹，选择 C Language Category。

(1) 禁止语言扩展(Disable Language Extensions)

(2) 削除重复串(Eliminate Duplicate Strings)

(3) 形成函数级联结(Enable Function - Level Linking)

(4) 工程文件、源文件及普通选项(Project, Source file, and Common Options)

(5) 复位(Reset)

(6) 抑制启动标题(Suppress Startup Banner)

(7) 作为错误警告(Warning as Errors)

(8) 警告级(Warning Level)

### 1.7.2 C++语言

从 C/C++语言文件夹，选择C++ Language Category。在应用程序中，C++选项标识类成员指针的继承性描述、控制异常处理、控制带虚拟基的类中隐藏虚拟构造函数/析构函数的生成。在以下类别中的C++语言数据库中将发现专门C++选项的讨论：

(1) 禁止结构替换(Disable construction Displacements)

(2) 允许异常处理(Enable Exception Handling)

(3) 通常目的表示(General - Ourpose Representation)

(4) 成员指针表示(Pointer - to - Member Representation)

(5) 工程文件、源文件及普通选项(Project, Source File, and Common Options)

(6) 表示方法(Representation Method)

### 1.7.3 代码生成

从 C/C++文件夹，选择代码生成(Code Generation)类。代码生成项选择 CPU、运行时库、调用协议及结构对齐。在以下类目中的代码生成数据库中将发现特定代码生成讨论：

(1) 调用协议(Calling Convention)

(2) 处理器(Processor)

(3) 工程文件、源文件及普通选项(Project, Source File, and Common Options)

(4) 复位(Reset)

(5) 结构成员字节对齐(Struct Member Byte Augnment)
(6) 使用运行时库(Use Run - Time Library)

### 1.7.4 General

从General文件夹可以检查常用选项。所有编译器常用选项,除去Debug Info选项,也可以在其他选项类目设置中获得。在以下类目中的General库中将发现特定通用编译器选项的讨论。

(1) 调试信息(Debug Info)
(2) 生成浏览信息(Generate Browse Info)
(3) 优化(Optimizations)
(4) 处理器定义(Preprocessor Definitions)
(5) 工程文件、源文件及普通选项(Project, Source File, and Common Options)
(6) 复位(Reset)
(7) 作为错误警告(Warning as Errors)
(8) 警告级(Warning Level)

### 1.7.5 列表文件

选择C/C++文件夹,然后选择Listing Files Category。列表文件选项生成浏览信息文件及代码列表文件。特定列表文件信息在以下条目的Listing Files数据库中可以找到:

(1) 不要压缩信息(Don't Pack Info)
(2) 排除局部变量(Exclude Local Variables)
(3) 生成浏览信息(Genarate Browse Info)
(4) 中间浏览信息文件名(Intermediate Browse Info File Name)
(5) 列表文件名(Listing File Name)
(6) 列表文件类型(Listing File Type)
(7) 工程文件、源文件及普通选项(Project, Source File, and Common Options)

### 1.7.6 优化

选择C/C++folder,然而选择优化(Optimization)类。这些选项指定编译器怎样很好地配合于应用程序执行。五个优化条目(Default,Disable,Maximize Speed 和 Minimize Size)中的四个并不要求用户方面更进一步的优化。如果选择第五个优化条目,Customize,还可以用Category Settings中列表框中的选项设定特别的优化。特定优化讨论在下面类目中的Optimization数据库中可以找到:

(1) 嵌入函数扩展(Inline - Function Expansion)
(2) 优化(Optimization)
(3) 工程文件、源文件及普通选项(Project, Source File, and Common Options)

### 1.7.7 预编译头文件

选择C/C++文件夹,然后选择Precompiled Headers Category。预编译头文件加速编译

时间，它们还可以预编译任何 C 或C++代码(包括嵌入代码)。另外支持的主题在 Precompiled Headers 数据库中可以找到，包括：

(1) 预编译头文件的自动使用(Automatic Use of Precompilod Headers)
(2) 生成. PCH 文件(Create . PCH File)
(3) 预编译头文件的 Per 文件的使用(Per - File Use of Precompiled Headers)
(4) 工程文件、源文件及普通选项(Project, Source File, and Common Options)
(5) 使用. PCH 文件(Use . PCH File)

### 1.7.8 预处理器

选择 C/C++文件夹，然后选择 Preprocessor Category。预处理器选项定义特定控制字符、宏和用于C++预处理器的包含路径。用 Preprocessor 数据库以及下面关于怎样使用 Preprocessor 选项附加信息的条目：

(1) 附加包括目录(Additonal Include Directories)
(2) 忽视标准包括路径(Ignore Standard Include Paths)
(3) 预处理器定义(Preprocessor Definitions)
(4) 工程文件、源文件及普通选项(Project, Source File, and Common Options)
(5) 解除符号定义(Symbols to Undefine)
(6) 解除全部符号定义(Undefine All Symbols)

# 第二章　Visual C++ Workbench 初步

Microsoft Visual C++ Workbench 是一种集成环境，允许简易地生成、打开、观察、编译、存储、编译和调试所有的 C 语言及 C++语言应用程序。Visual C++ Workbench 也包含了为很好地协调工作环境而依据各人喜好及配合应用程序特定硬件要求而设计的选项。

## 2.1　开始 Visual C++ Workdench

执行 Visual C++ Workbench 是较容易的。如果使用鼠标，用户可以双击 Visual C++图标，它可以在 Microsoft Visual C++组中找到。另外，还可以从键盘上访问 Windows Run 命令，输出以下命令：C:\MSVC.EXE

## 2.2　访问上下文敏感性帮助

Microsoft 已经通过使所有产品文档联机提供了每种 Visual C++ Workbench 的介绍。获得这些有价值资源只要把光标放在不了解的性能上，按下 F1。

然而，上下文敏感性帮助并不限制 Visual C++ Workbench 的性能。如果在一个 C/C++语言结构上放置光标，并按下 F1(图 2.1 Visual C++ Workbench 初始屏)帮助工具将自动展示结构语法的描述，它通常是一个简洁的可执行例子。

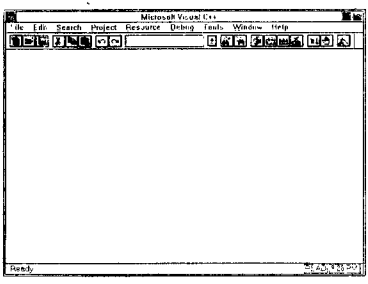


图 2.1　显示了 Visual C++ Workbench 的初始屏幕

这章的设计是要给出一个关于每个 Visual C++ Workbench 选项的综述。不要让可获

得选项及性能的数量弄得灰心丧气。还可以使用一些 Visual C++ Workbench 能力的缺省设置，这可以较容易的让一个应用程序通过并运行。

随着经验的增长和应用程序要求在复杂性方面提高，就会逐渐获得这个功能强大的环境的亲身体会。在阅读这一章时，拿着铅笔记录一下本章所介绍的 Visual C++ Workbench 性能。当需要使用这性能时，则可以从解释怎样使用选项的段落中方便地加以参考。

## 2.3 理解菜单

在讨论每一个 Visual C++ Workbench 性能前，应解释一下所有菜单项共有的一些特点。例如，有两种方法访问菜单项。最通用的方法是把鼠标箭头指向所选项上，然后单击鼠左按钮。第二种方法是使用带下划线的热键。例如，可以通过同时按下 ALT 键和字母 F 键访问 File 菜单。

菜单项可以使用上面描述的相同次序来选择，通常还有另外一种选择的办法。可以通过使用特定的热键组合在集成环境中的任何地方直接激活一些菜单项。如果一个菜单项有这种能力，这个选项的特定热键组合将展示在菜单上菜单项的右端。例如，在 File 菜单中列的第一个选项是 New...这一选项可以被立即激活，避开了首次选择 File 菜单的必要性，只需简单按一下 CTRL-N。

有关菜单项还有解：首先，如果菜单是灰的，集成环境在警告这一事实即那个特殊选项当前不能获得。基本上，这意味着集成环境缺乏一些使特殊选项有效的必要的预要求。例如，如果编辑窗口空，File 菜单的 Save 选项将是灰色的。这一选项知道不可能存储不存在的东西，然而通过不激活以及变灰 Save 命令来表明这点。

第二，紧跟着三个点的任何菜单，表明选项被选中时，将自动展示一个对话框或者一个小菜单。例如，File 菜单的 Open 命令，被选中时，出现 Open 对话框。

最后，还可以通过单击工具条上的相关按钮激活一些菜单项。工具条在主菜单条下。把笔拿在手里，准备标记感兴趣的 Visual C++ Workbench 的性能，下面逐个介绍。

## 2.4 File 菜单

Visual C++ Workbench File 菜单实现了对许多 Windows 应用程序都应用的文件操作命令标准集。图 2.2 显示了可以从 File 菜单中获得的命令项。

### 2.4.1 New....

New 命令打开一个新编辑窗口。一般是在这点上开始设计应用程序。集成环境自动命名和记数所打开的每一个窗口(图 2.2 Visual C++的工作菜单)。记数从 1 开始，因此第一个窗口名称通常是 xxx1，第 2 个窗口命令为 xxx2 依此类推。xxx 是一项标签，表明所工作文件的类型(代码、工程文件、资源、位图、二进制、图标和光标)。

如果已经让命令为 xxx1 到 xxx6 的窗口打开，然后又决定关闭 xxx2 窗口，下一次调用 New 命令时，这项命名(这里是 xxx2)将不会被再使用。窗口将自动补充下一个最高数码(例如，xxx7)。

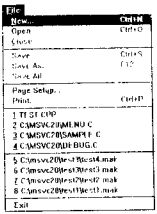


图 2.2 Visual C++的工作菜单

打开一个的编辑窗口最快速的方法是单击工具条上最左边的按钮。这个按钮上有一个文件图像。可以单击这项控制直接调用 New 命令。

### 2.4.2 Open...

不像 New...所打开的编辑窗口是先前不存在的文件,Open...命令打开的是先前被存储的文件。选择这个选项时,Visual C++ Workbench 展示了标准 Open File 对话框,显示了缺省驱动、路径和文件搜索参数,并且允许选择自己的。

对话框具有时间节约的特征,可以自动记录用户的喜好,每一次该用户使用 Open...命令时都能自动把这些作为缺省。试图打开一个已经打开的文件时会自动调用一个报警和警告消息。这种有用的记忆防止偶然地打开两个或更多的同一文件的复制本,只编辑其中之一,以及再存储非最新的版本!

工具条上左数第二个按钮,上面有一个带打开箭头的文件夹的图形,这可以用来调用 Open...命令。

### 2.4.3 Close

Close 命令被用来关闭一个打开的文件。如果有多个文件打开。这个命令将关闭活动或者是被选窗口。通过观察窗口边界,可以判断那个窗口是活动的。活动或被选窗口有键盘和鼠标焦点,并且用系统所选颜色来显示。这些选项常常包括彩色标题条以及深色窗口边界。非活动窗口通常使标题条及窗口边界变灰。

如果偶然试图关闭一个未存储文件,不必担心。集成环境会自动地警告这一文件未被存储以便保护文件不致产生破坏性结果,并且发出访问是否在此时存储这个文件。

### 2.4.4 Save

Save 命令把当前被选的内容或活动窗口存储到指定文件中去。要区别一个窗口先前已被存储内容同未存储内容时,只需简便地检测窗口标题条。如果看见一个缺省标题,例如 xxxx1,就可以知道窗口内容还未被给予一个有效的文件名,没有存储。要存储没有保存过

的文件将自动调用 Save 作为对话框。

还可以用工具条上的 Save 按钮,左起第三个按钮有软磁盘图像在上面。如果文件打开在只读状态(见 EditProperties 命令的描述),控制图像将变灰,表明这一选项当前不可获得。

### 2.4.5 Save As...

Save As... 选项允许以新名称存储活动窗口内容的复本。如果考虑为什么可以选择这个选项,这儿有一个可能的答案。已经完成了一个工程文件,有一个工作程序。然而,有可能要尝试一些变化。为了安全起见,并不想要覆盖当前版本,通过选择 Save As... 选项可以用一个新名字存储文件内容,然后要处理新文件。如果一旦出现什么不测,还可以返回原来的文件。

### 2.4.6 Save All

如果从未书写过一个 C,C++ 语言的 Microsoft Windows,或 Microsoft Windows NT 应用程序,就可能会在包括于生成一个工程文件的可执行文件中的文件确切数量上卡壳。Save 选项出现的问题是存储活动窗口的内容。Save All 选项则存储每个窗口的内容。如果有窗口包含了先前未存储的文本,Save All 命令将自动调用 Save As 对话框,提示给每个窗口一个有效的文件名。

### 2.4.7 Page Setup...

Page Setup... 选项最常的用法是文档化及格式化硬拷贝。Page Setup 对话框允许选择每个被打印页的页眉和页脚,可用它来设置顶端、底部和左、右打印边界。

表 2.1 列出选择页眉和页脚可用的格式代码。

**表 2.1 页眉和页脚的格式代码**

| 格式化代码 | 相关用法 |
|---|---|
| &c | 文本对中 |
| &d | 添加当前系统日期 |
| &f | 使用文件名 |
| &l | 文本左对齐 |
| &p | 添加页号 |
| &r | 文本右对齐 |
| &t | 添加当前系统时间 |

### 2.4.8 Print...

要得到的活动窗口内容的硬拷贝只需简单地选择 Print... 命令。Print 对话框提供了几个选项。

首先,可以单击合适的单选按钮,来选择打印整个窗口内容或者仅打印被选文本。还可以通过选择 Setup 选项,来挑选使用何种打印机以及被选打印机的格式。

如果只希望打印窗口内容的一部分,就首先要选择所希望的文本。选择文本只要简单地把鼠标箭头置于所要打印文本的第一个字符上,然后按住鼠标左按钮同时向右或向下拖动鼠标穿过该文本。这会造成被选文本将被反向显示。当文本被选中,Print 对话框将以正常类

型(非变灰色)显示 Print Range Selection 单选按钮,表明该项当前可用。

### 2.4.9 近期文件列表

在 Print... 命令的右方是一个菜单选项,列出了四个最近被编辑的文件。关于这些列表(常称为历史列表)的优越性是它们具有上下文敏感性,历史列表由于记忆了为某一特定选项所选的最新几个条目,所以可以节省时间。就这个菜单而言,这些被记忆条目是先前已打开文件。第一次使用 Visual C++ Workbench,这一部分的 File 菜单是空白,因为没有被打开文件可以作历史记录。这个菜单在前面显示的图 2.2 中,它展示了带有四个文本文件的近期文件历史列表。

### 2.4.10 近期工程文件列表

近期文件列表在菜单上紧靠于近期文件列表之下。这个历史列表同近期文件列表相似。只是近期工程列表仅包含工程文件。想要打开在这两个列表中的任意文件。只需在所选条目上双击鼠标左按钮。图 2.2(见前面部分)显示了有四个 MAKE 文件的 File 菜单。

### 2.4.11 Exit

Exit 选项允许退出 Visual C++ Workbench。不要担心在选择 Exilt 前是否忘记存储窗口内容了。集成环境会自动为每个窗口包含的未存储文本显示一条警告消息,以便在退出之前存储这些信息。

## 2.5 Edit 菜单

Edit 菜单选项允许快速编辑或者搜索一个活动窗口的内容,方法同使用任何标准字处理器一样。图 2.3 显示了 Visual C++ Workbench Edit 菜单。

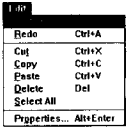


图 2.3 Visual C++ Workbench Edit 菜单

### 2.5.1 Undo

Undo 命令允许用户取消最近所做的编辑变动。这可以从工具条上使用 Undo 选项。在工具条上,Undo 选项是左指向箭头,这是本系统上左起第七个图标。