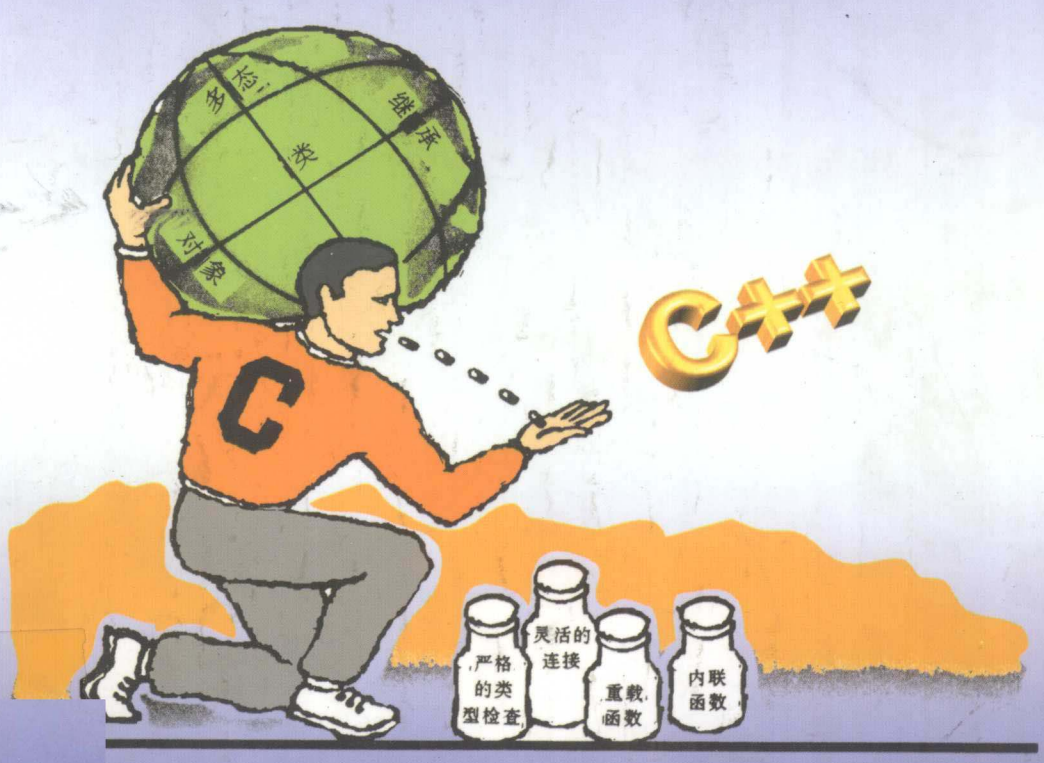
现代测绘科技丛书

面向对象的测量程序设计

马明栋 赵长胜 施群德 杜维甲 编著



教育科学出版社

面向对象的测量程序设计

编著　马明栋　赵长胜
施群德　杜维甲

教育科学出版社出版

责任编辑：徐长发
封面设计：安广军

图书在版编目(CIP)数据
现代测绘科技丛书/武文波　主编
面向对象的测量程序设计/马明栋　赵长胜　施群德　杜维甲　编著
北京.教育科学出版社，2000.05
ISBN 7-5014-1648-6

Ⅰ.①现…②武…
Ⅱ.①面…②马…③赵…④施…⑤杜…
Ⅲ.专业测绘—程序设计
Ⅳ.P.25
中国版本图书馆 CIP 数据核字(20)第 01469 号

面向对象的测量程序设计
编著　马明栋　赵长胜　施群德　杜维甲
教育科学出版社出版
(北京海淀区北三环中路 46 号)

通州印刷厂印刷

2000 年 5 月第 1 版　2000 年 5 月北京第 1 次印刷
开本：787×1092 毫米 1/16　印张：20
字数：480 千字　　印数：1000 册
ISBN 7-5014-1648-6/P.208-2(课)
定价：29.00 元

前言

学习一门语言仅仅学习语言的构成要素是远远不够的。语言需要实践，正如学习一门外语一样，如果你不出国实践一次，你就不会真正理解语言的要义。

面对扑面而来的海洋般的电脑书籍，测量工程技术人员为解决自身要面对的实际工程测量项目，挑选一本真正能解决问题的书籍是很不容易的事情。特别是计算机语言书籍很多，选择哪种语言作为自己今后的主要编程工具也使人常常举棋不定。

有鉴于此，我们编写了本书，书中的源程序全部调试通过并有实例验证，你只需要美化相应的输出部分（有的地方需要增加）就完全能够解决你日常的全部测量计算工作。花费1000 多元买一套测量数据处理系统，真的不如自己动手做一套出来。我们相信，只要读者能够与计算机交友，与 Microsoft Visual C++ 6.0 建立深厚的友谊，读者就能够做出比书中实例更加强大的测量数据处理系统。

本书第一章绪论部分主要介绍 C、C++发生与发展的过程，并详细介绍了 Microsoft Visual C++ 6.0 能够做哪些工作。第二章与第三章纲要性地介绍了 C 与 C++语言。第四章详细介绍了测量数据处理系统的数学模型。第五章以 Microsoft Visual C++ 6.0 提供的 AppWizard 工具建立控制台执行文件（DOS 形式），测试了测量常用的一些函数以及矩阵运算函数。第六章与第七章以 Microsoft Visual C++ 6.0 提供的 AppWizard 工具建立基于 MFC（exe）的小型地形测量数据处理系统和控制测量数据处理系统，其中包括 GPS 基线向量网平差程序，读者可以将这两个系统合二为一。第八章简要介绍测量控制网的可靠性分析与优化设计的基本概念，读者可利用本书提供的源程序结合控制网的可靠性分析与优化设计的专著进行该方向的研究与设计工作；此外本章还提供了三个大地测量方面经常要使用的小程序——大地测量主题正反算以及换带计算程序和新、旧坐标换算程序，读者也可以将其纳入测量数据处理系统之中。

总之，通过本书的学习与实践，相信读者会成长为一名真正的测量工程技术人员和 C++ 编程高手。

编著者　马明栋

2000 年 8 月 30 日星期三

第一章 绪论

§1.1 C语言发展历程及特点

1.1.1 C语言发展历程

C语言是国际上广泛流行的、很有发展前途的计算机高级语言。它适合于作为系统描述语言，即用来写操作系统软件，也可用来写应用系统软件。

以前的操作系统等系统软件主要用汇编语言编写（包括UNIX操作系统）。由于汇编语言依赖于计算机硬件，程序的可读性和可移植性都较差。为了提高可读性和可移植性，最好改用高级语言，但一般高级语言难以实现汇编语言的某些功能（汇编语言可以直接对硬件进行操作，例如，对内存地址的操作、位操作等）。人们设想能否找到一种既具有一般高级语言特性，又具有低级语言特性的语言，集它们优点于一身。于是，C语言就在这种情况下应运而生了。

C语言是在B语言的基础上发展起来的，它的根源可追溯到ALGOL60。1960年出现的ALGOL 60是一种面向问题的高级语言，它离硬件比较远，不宜用来编写系统程序。1963年英国剑桥大学推出CPL（Combined Programming Language）语言。CPL语言在ALGOL 60的基础上接近硬件一些，但规模较大，难以实现。1967年英国剑桥大学的Matin Richards对CPL语言作了简化，推出BCPL（Basic Combined Programming Language）语言。1970年美国贝尔实验室的Ken Thompson以BCPL语言为基础，又进一步作了简化，设计出很简单而且很接近硬件的B语言（取BCPL的第一个字母），并用B语言写了第一个UNIX操作系统，在PDP－7上实现。1971年在PDP－11/20上实现B语言，并写了UNIX操作系统。但B语言过于简单，功能有限。1972年至1973年，贝尔实验室的D.M.Ritchie在B语言的基础上设计出C语言(取BCPL的第二个字母)。C语言既保持了BCPL和B语言的优点(精练，接近硬件)，又克服了它们的缺点（过于简单，数据无类型等）。最初的C语言只是为描述和实现UNIX操作系统提供一种工作语言而设计的。1973年，K.Thompson和D.M.Ritchie两人合作把UNIX的90%以上功能用C改写（即UNIX第5版。原来的UNIX操作系统是1969年是由美国贝尔实验室的K.Thompson和D.M.Ritchie开发成功的，用汇编语言书写）。

后来，C语言多次作了改进，但主要还是在贝尔实验室内部使用。直到1975年UNIX第6版公布后，C语言的突出优点才引起人们普遍注意。1977年出现了不依赖于具体机器的C语言编译版本《可移植C语言编译程序》，使C移植到其它机器时所需做的工作大大简化，这也推动了UNIX操作系统迅速地在各种机器上实现。例如，VAX，AT&T等计算机系统都相继开发了UNIX。随着UNIX的日益广泛使用，C语言也迅速得到推广。C语言和UNIX可以说是一对孪生兄弟，在发展过程中相辅相成。1978年以后，C语言已先后移植到大、中、小、微型机上，已独立于UNIX和PDP了。现在C语言已风靡全世界，成为世界上应用最广泛的几种计算机语言之一。

以1978年发表的UNIX第7版中的C编译程序为基础，Brian W.Kernighan与Dennis M.Ritchie（合称K&R）合著了影响深远的名著《The C Programming Language》，这本书中介绍的C语言成为后来广泛使用的C语言版本的基础，它被称为标准C。1983年，美国国家标准化协会（ANSI）根据C语言问世以来各种版本对C的发展和扩充，制定了新的标准，称为ANSI C。ANSI C比原来的标准C有了很大发展。K&R在1988年修改了他们的经典著作《The C Programming Language》，按照ANSI C标准重新写了该书。1987年，ANSI

又公布了新标准—87 ANSI C，目前流行的 C 编译系统都是以它为基础的。现今广泛流行的各种版本 C 语言编译系统虽然基本部分是相同的，但也存在差异。在微机上使用的有 Microsoft C（Quick C），Turbo C，High C 等，它们的不同版本又略有差异，因此读者应了解所用计算机系统的 C 编译系统的特点和规定。

1．1．2 C 语言的特点

一种语言之所以能存在和发展，并具有生命力，总是有其不同于（或优于）其它语言的特点。C 语言的主要特点如下。

1. 语言简洁、紧凑，使用方便、灵活。C 语言一共只有 32 个关键字，9 种控制语句，程序书写形式自由，主要用小写字母表示，压缩了一切不必要的成分，下面将 C 与 PASCAL 语言作一些比较：

C 语言	PASCAL 语言	含义
{}	BEGIN…END	复合语句
if(e) S;	IF(e) THEN S	条件语句
int i;	VAR i:INTEGER	定义 I 为整型变量
int a[10];	VAR a:ARRAY[10] OF INTEGER	定义 a 为整型一维数组
int f();	FUNCTION f():INTERGER	定义 f 为返回整型值的函数
int *p;	VAR P:↑ INTEGER	定义 P 为指向整型变量的指针变量
i+=2;	i=i+2	赋值语句，使 i＋2->I
i++,++i;	i=i+1	i 自增值 1，i+1->I

学过 PASCAL 的读者可以看到：C 程序比 PASCAL 简练，源程序短，输入程序工作量少。

2. 运算符丰富。C 的运算符包含的范围很广泛，共有 34 种运算符。C 把括号、赋值、强制类型转换等都作为运算符处理。从而使 C 的运算类型极其丰富，表达式类型多样化，灵活使用各种运算符可以实现在其它高级语言中难以实现的运算。

3. 数据结构丰富，具有现代化语言的各种数据结构。C 的数据类型有：整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型等。能用来实现各种复杂的数据结构（如链表、树、栈等）的运算。尤其是指针类型数据，使用起来比 PASCAL 更为灵活、多样。

4. 具有结构化的控制语句（如 if…else 语句、while 语句、do…while 句、switch 语句、for 语句）。用函数作为程序模块以实现程序的模块化。是结构化的理想语言，符合现代编程风格要求。

5. 语法限制不太严格，程序设计自由度大。例如，对数组下标越界不作检查，由程序编写者自己保证程序的正确。对变量的类型使用比较灵活，例如，整型量与字符型数据以及逻辑型数据可以通用。一般的高级语言语法检查比较严，能检查出几乎所有的语法错误。而 C 语言允许程序编写者有较大的自由度，因此放宽了语法检查。程序员应当仔细检查程序，保证其正确，而不要过分依赖 C 编译程序去查错。“限制”与“灵活”是一对矛盾。限制严格，就失去灵活性；而强调灵活，就必然放松限制。一个不熟练的人员，编一个正确的 C 程序可能会比编一个其它高级语言程序难一些。C 语言要求编程者对程序设计更熟练一些。

6. C 语言允许直接访问物理地址，能进行位（bit）操作，能实现汇编语言的大部分功能，可以直接对硬件进行操作。因此 C 既具有高级语言的功能，又具有低级语言的许多功能，可用来写系统软件。C 语言的这种双重性，使它既是成功的系统描述语运，又是通用的程序设计语言。有人把 C 称为“高级语言中的低级语言”，也有人称它为“中级语言”，

意为兼有高级和低级语言的特点。

7. 生成目标代码质量高，程序执行效率高。一般只比汇编程序生成的目标代码效率低 10～20%。

8. 用 C 语言写的程序可移植性好（与汇编语言比），基本上不作修改就能用于各种型号的计算机和各种操作系统。

上面我们只介绍了 C 语言最容易理解的一般特点，由于 C 语言的这些优点，使 C 语言应用面很广。许多大的软件都用 C 编写，这主要是由于 C 的可移植性好和硬件控制能力高，表达和运算能力强，许多以前只能用汇编语言处理的问题现在可以改用 C 语言来处理了。

C 的以上特点，读者现在也许还不能深刻理解，由于本书不是 C 语言教科书，如果读者还没有接触过 C 语言，建议你先看一本 C 语言专著。待学完本书以后各章后，再回顾一下，就会有比较深的体会。

我们从应用的角度出发对 C 语言和其它高级语言作一简单比较，从掌握语言的难易程度来看，C 语言比其它语言难一些。目前，由于 Windows95（98、NT）等视窗操作系统的广泛流行，基于视窗操作系统的语言编译系统也成百花齐放之势，如 Microsoft Visual C++、Visual Basic，Borland C++Build、Delphi（基于 Pascal 语言的可视化开发系统）均有过人之处。

从教学角度，由于 PASCAL 是世界上第一个结构化语言而曾被认为是计算机专业的比较理想的教学语言，目前在“数据结构”等课程中一般用 PASCAL 语言举例。C 语言也是理想的结构化语言，且描述能力强，同样适于教学，而且“操作系统”课程多结合 UNIX 讲解，而 UNIX 与 C 不可分，因此，C 语言有可能取代 PASCAL 而成为被广泛使用的教学语言，而且 C 除了能用于教学外，还有广泛的应用领域，因此更有生命力。PASCAL 和其它高级语言的设计目标是通过严格的语法定义和检查来保证程序的正确性，而 C 则是强调灵活性，使程序设计人员能有较大的自由度，以适应宽广的应用面。

总之，C 语言对程序员要求较高。程序员使用 C 语言编写程序会感到限制少、灵活性大，功能强，可以编写出任何类型的程序。现在，C 语言已不仅用来编写系统软件，也用来编写应用软件。学习和使用 C 的人已越来越多。

§1.2 C++语言及其发展史

C++语言基于 C，是 C 的一个超级集合。C++保留了 C 的灵活性且具有强大的处理硬软件接口编程和低层系统程序设计能力；C++保留了 C 语言的紧凑性和强有力的表达式功能；更重要的是 C++提供了支持面向对象程序设计和高层问题抽象的平台。在支持面向对象程序设计方面 C++比 Ada 更前进了一步。在简洁性和模块化方面，C++类似于 Modula－2，而且保留了 C 的高效率和简洁性。

C++是一种混合性语言。它支持用面向对象的程序设计来求解项目，使之产生项目全新的面向对象解。实际上，C++语言表现出具有过程程序设计方法和新的面向对象设计方法。这种在 C++中的双重性对于 C++初学者提出了一种特殊的挑战，即不仅要学习新语言，而且要学习新的项目求解的方法和掌握新的思维方式。

C++是由 BCPL 和 Simula67 的某些灵感而导致产生的。它从 Simula 引进了子类（导出类）和虚拟函数，借鉴了 Algol68 操作符重载能力和灵活地把说明紧挨着应用程序首次使用的地方的能力。象其他现代程序设计语言一样，C++语言是进化和改良了我们所熟知的高级程序设计语言的某些最佳特征。当然，最接近于 C++的还是 C。

早在 1980 年 C++由贝尔实验室的 Bjarne Stroustrup 创建。最初的动机是打算在效率上改进 Simula67 语言并采用复杂事件驱动，当时的 C 称为带类的 C（C with classes）缺少操作符重载、引用和虚拟函数等许多细节。

在 1983 年，C++首次推广到外界，到 1987 年夏天，C++仍然在演变。AT&T 决定性地提交出与 C 兼容的 C++，这样就保留了完整和完美的 C 的几百万行代码和广泛的 C 库和 C 工具。鼓励 C 程序员学习 C++而不放弃他已经工作多年的过程程序设计语言。这对 C 程序员来说是个大好消息——当从 C 前进到 C++时，他们仍没有放弃他们过去工作的领域，从而使语言在发展中保持了稳定性。事实上，C 语言本身影响了 C++的发展。ANSI C 标准草案中包含了某些 C++的关键特征，诸如函数原型。1990 年通过的 ISO C 中也是加此。

C++语言支持大型软件系统开发。在语言中增加了强类型检测以及与 C 语言比较，减少了在 C++软件系统的开发性错误，并提供了对多用户程序设计。

C++语言最有意义的方面是支持面向对象特征。为了从 C++得到好处，要求程序员在项目求解方法中要进行某些改变：标识对象和对象操作，构造类和子类，有关其实现细节必须推迟，直到所有数据结构和对象之间的相互作用被确定为止。

因为 C 语言允许程序员获得系统的直接控制，许多 C 程序员严格按自底向上方式工作，首先关心低层结构和演变，编排这些低层结构进入到系统高层结构。对转向到 C++的程序员来说，要求他们部分地改变这种实现过程，在软件开发中应同时采用自顶向下和自底向上方法。因为需要标识抽象数据结构和定义与其他抽象对象的相互作用，所以要在抽象对象中开始封装数据结构和算法的实现。为了能够在系统中测试其他类的相关作用，快速原型开发蕴含着对一个类的快速实现。每个类的实现、完善和改进应当推迟到整个系统结构（类的集合和他们的内在联系）确定才能够达到。

所有面向对象语言中，可扩充性是关键，所使用的语言应尽可能自然地反映问题的实体和操作，这是对面向对象软件设计者的挑战，在大多数高级程序设计语言中这是一个困难的任务，因为固定的语言提供有限几种结构和特征。例如，在许多 FORTRAN 软件系统中程序员为表示一个记录结构的集合，必须用多个数组来表示这个数据集合，多个数组则是与一个公共索引紧紧联在一起的，所以如果每个记录有五个数据项，则这些数据将用相同索引位置进行存取，才能得到。

一个最好的解决方法也许是定义一个抽象对象，data_base，它能接收插入、删除、存取或显示（包含对象内的信息）的消息，这样 data_base 对象的操纵就能够实行类似于人的自然方式。如你希望一个新的记录插入到 data_base 对象中，那你至少要做如下事情：

```
database.insert（data）
```

data_base 是一个适当的说明对象，insert 函数是一个相匹配的方法，此方法在支持 data_base 对象的类中定义 data 参数是增加到 data_base 对象中的指定信息，包含 data_base 的对象的类不是作基础语言的一部分。程序员用定义对象的一个新类（修改一个现存类、或建立一个子类）扩充该语言以适应项目需求。这能更自然地从项目解空间映射到程序空间。C++像其他面向对象语言一样，具有更自然的可扩充性。希望读者能尽快掌握这种功能。

§1.3 Visual C++ 6.0 简介

Visual C++是 Microsoft 公司推出的开发 Win 32 环境（Windows 95/98/NT，以及即将广泛使用的 Windows 2000）程序，面向对象的可视化集成编程系统。它不但具有程序框

架自动生成，灵活方便的类管理，代码编写和界面设计集成交互操作，可开发多种程序（应用程序、动态链接库、ActiveX 控件等）等优点，而且通过简单的设置就可使其生成的程序框架支持数据库接口、OLE 2、WinSock 网络、3D 控件界面。因此，它现已成为开发 Win32 程序的主要开发工具。

本节主要对 Visual C++编程的若干要点进行重点讨论，以帮助读者更加深入地理解和掌握 Visual C++编程之精髓。

1.3.1 Visual C++技术主要特征

面向对象程序设计（Object－Oriented Programming 简称 OOP）方法已出现近三十年，90 年代已成为程序设计的主流方向，面向对象程序设计语言是现代程序开发的主要工具，如 C++、Java 是现代程序员必须掌握的编程语言。

程序包含两类基本的元素，即数据和操作数据的指令集（称为代码）。传统的程序设计语言以设计代码为核心，程序设计实际上就是指定程序指令的先后次序，数据表示必须适应代码的设计。模块化程序设计方法将完成某一功能的指令集组成一个相对独立的程序模块（即函数或过程），使得程序的结构清晰，便于有效地维护，对程序设计技术有很大的促进。但由于结构化程序设计方法并不能保证各程序模块之间真正的相互独立，程序设计者在设计一个模块时很难完全排除其他模块的影响。随着程序规模的增大，各模块之间的相互影响导致了一些难于测试、难以定位发现的错误，增加了程序开发和维护的困难。面向对象程序设计方法就是在这种背景下出现和发展起来的。

面向对象程序设计方法主要以数据为中心，代码是围绕着需要处理的数据而设计的，面向对象程序设计语言具有如下的主要特征：

●对象的类描述

面向对象程序设计语言将程序描述的事物看成一个整体，称为对象（Object）。事物的属性基本可以分为两部分，即内部状态（性质）和对数据的操作方法及由此造成的对外部的影响。对象的数据用于描述内部状态，而代码完成对数据的操作。因此，对象就是包含数据和代码的完全独立的实体。类（class）就是具有相同的属性的所有对象的逻辑原型，是对象的规则和设计。同一类的对象具有相同的性质和方法，每一个具体的对象都是类的一个实体，创建对象就是把类实例化。

●封装性（Encapsulation）

封装性是 OOP 的核心技术，是指面向对象程序设计语言将数据和处理数据的方法组合在类中，并具有模块化和信息隐藏的特征。类是一个独立的模块，类的内部状态描述数据对程序的其他部分是不可见的，类只向外界公布其具有 Public 属性的数据和代码，并构成了类与外界的接口。外界不能直接对类的内部状态进行修改，而只能通过这个接口将信息传递给类，并由类定义的对内部数据进行操作的方法进行内部修正，外界不能决定这种修正的结果；只能得到类进行操作所做出的反应。封装性能防止类与外部的非法交互和访问，避免外界对对象内部状态的错误改变，确保类这一模块的真正独立，以保证程序的安全运行。

同时，由于程序的其他部分只能访问类的接口，只要保持类的接口不变，改变类的内部结构、工作方式和实现就不会对整个程序产生非预期的影响，因此，对类的内部做任何的优化都是安全的。

●多态性（Polymorphism）

不同的类或对象对外界传入的相同信息能根据自身的性质做出不同的反应，这就是多态性。通过不同的类或对象都可设计自身的处理外界传入信息的方法实现多态性。在 OOP

中，多态性具有两方面的实际意义，一是具有相同名字的接口（具 Public 属性的公有数据和函数）在不同类中能具有不同的意义和实现；二是具有同一名字的函数可以具有不同的实现代码，在调用时，根据传入的参数不同而调用不同的代码，这叫做函数的重载

●继承性（Inheritance）

继承性是指一个类可以派生出新的类，新类能继承原类定义的性质和方法，还能在原类定义的性质和方法之外加入自身定义的性质和方法。通过继承性能形成类之间的层次结构，在上层中已经定义的性质和方法能被下层直接继承使用，下层就不需重新定义，从而实现了代码的重复利用。这样，下层的类只需专注于自身的新特征描述，提高了程序设计的效率和程序组织的有效性。

不是由其他类派生的类称为基类。由其他类派生的类叫做该类的子类，该类叫做其子类的父类或超类。子类可以利用多态性特征来对继承而来的方法进行重载，使得具有相同名字的方法在子类和父类中实现不同的功能。

C++是运用最广泛的面向对象程序设计语言， Visual C++是一个具有集成、交互和可视化编程的 C++实现，具备上述的所有 OOP 特征。

编写 Visual C++程序实际上就是一个构造类和把类实例化的过程。由于 Windows95 / 98 / NT 是 PC 平台中应用最广泛的操作系统（Microsoft 力图用一个叫做 Win32 的标准的 32 位应用程序接口来作为对这几个操作系统的共同开发接口，所以经常采用 Win32 来代表 Microsoft 的 32 位 Windows 操作系统）， Visual C++主要针对 Win 32 的应用程序开发。

1. Win32 编程

Win 32 具有抢先式多任务、多线程和线性寻址内存管理等特征，Win 32 编程的基本要求包括：

（1）应用程序的执行独立于硬件设备；

（2）应用程序具有图形用户界面；

（3）能在 Wndows 95 和 Windows NT 之间透明移植，并可移植到支持 Windows NT 的 RISC 硬件平台；

（4）高性能的抢先式多任务和多线程管理；

（5）高级的多媒体支持；

（6）通过 OLE 2 技术实现多个应用程序的对象定位。

Microsoft 为进行 Win 32 编程提供了一套名为 Win 32 SDK 的应用程序编程接口，其中包括上千个 Win 32 系统函数。 Visual C++包括一套叫做 MFC（Microsoft Foundation Class Library）的 C++类库，其中定义了进行 Win 32 编程所需要的各种类。有的类封装了大部分的 Win 32 SDK 中应用程序编程接口函数；有的类封装的则是应用程序本身的数据和操作；还有的类封装了 ActiveX、OLE 和 Internet 编程特性，WinSock 网络特性和 DAO（Data Access Objects）、ODBC（Open Database Connectivity）数据访问功能。Win 32 SDK 和 MFC 是实现 Win 32 编程的主要工具。

Visual C++的 APPWizard 工具能自动生成应用程序框架，该框架定义了应用程序的轮廓，并提供了用户接口的标准实现方法。运用 Visual C++的资源编辑器（Resource Editor）能直观地设计程序的用户界面，而 ClassWizard 能把用户界面和程序代码连接起来。程序员要做的就是用 MFC 类实现框架中未完成的应用程序的特定功能部分。所以，使用 Visual C++可以实现 Win 32 的可视化程序设计。

2. 框架和文档一视结构

所谓框架(Framework)，就是应用程序所应具备的软件模块按一定的结构组成的集合。基于 MFC 的应用程序框架是定义了程序结构的 MFC 类库中类的集合，是 Visual C++编程的骨架。运用 MFC 应用程序框架能获得如下的优点：

（1）标准化的程序结构和用户接口：这对具有标准用户界面的 Win 32 程序来说，可以极大地减轻程序员的负担，使程序员不必过多地考虑界面，而把主要精力放在程序设计上，以提高程序设计的效率；

（2）框架产生的程序代码短，运行速度快，具有很大的灵活性：MFC 封装了 Win 32 SDK 中的几乎所有函数，能实现 Win 32 系统的任何功能；

（3）强大的功能：除封装了大部分的 Win 32 SDK 函数外，MFC 还提供了应用程序本身的数据和操作，及 ActiveX、OLE Internet、WinSock、DAO（Data Access Objects）、ODBC（Open Database Connectivity）等操作类。

MFC 框架的核心是文档一视结构（Document一View Architecture），这是一个很有用，但又往往较难以入门的功能。简单地说，文档—视结构就是将数据和对数据的观察或数据的表现（显示）相分离，文档仅处理数据的实际读、写操作，视则是显示和处理数据的窗口，视可以操作文档中的数据。

●框架结构

MFC 框架的基本结构包括应用程序对象、主框窗口、文档、视等，框架通过命令和消息将它们结合在起，共同对用户的操作做出响应。

应用程序管理着一个（SDI）或多个（MDI）文档模板，每个文档模板管理着一个或多个文档。用户通过主框窗口中的视观察和处理数据。

●应用程序对象

这是由 CWinAPP 派生的应用程序类对象。一个应用程序有且仅有一个应用程序对象，它负责应用程序实例的初始化和进程结束时的资源清除，以及创建和管理应用程序所支持的所有文档模板。

任何 Windows 应用程序都包含一个 WinMain（）函数，框架应用程序也一样，只是框架中的 WinMain（）函数由 MFC 类库提供并被隐藏，在框架应用程序启动时调用。应用程序对象的 InitInstance（）函数就是由 WinMain()调用的。

●主框窗口

这是应用程序的主窗口。MFC 框架定义了两种基本的主框窗口类，即单文档接口（Single Document lnterface，简称 SDI）主框窗口类 CFremWnd 和多文档接口（Multiple Document Interface，简称 MDI）主框窗口类 CMDIFrameWnd。应用程序的主框窗口应从其中之一派生出来。对 SDI，视是主框窗口的子窗口；对 MDI，必须从 CMDIChildWnd 派生出主框窗口的子窗口，视是该子窗口的子窗口。

●文档

文档类由 CDocument 类派生而来，文档类对象由框架生成的 File 菜单中的 New 或 Open 命令创建，指定了应用程序数据的实际读、写操作。如想应用程序支持 OLE 功能，则应从 COleDocument 类派生文档类。

文档由应用程序对象创建和维护的文档模板（Document Template）所创建。框架分别使用 CSingleDocTemplate 和 CMultiDocTemplate 文档模板来管理 SDI 和 MDI，前者可以创建和存储一种类型的文档，后者保存了一种类型的多个文档的列表。

●视

视类从 CView 或其子类（CEidtView、CFormView、CRecordView、CScrollView 等）派

生而来，是显示和观察文档数据的窗口类。视类定义了用户以什么方式见到文档的数据，以及如何与其进行交互。一个文档数据可能有多个视。

●文档—视结构

框架在响应它生成的标准用户接口 File 菜单中的 New 和 Open 命令时将创建文档视结构，其创建次序如下：

（1）在程序启动时，WinMain（）函数调用应用程序对象的 InitInstance（）函数，并在其中创建文档模板；

（2）程序运行过程中，用户选取了 File 菜单中的 New 或 Open 菜单项，框架将调用 CWinApp::OnFileNew（）或 CWinApp::OnFileOpen（）函数，并使用已创建的文档模板创建文档；

（3）文档模板同时创建主框窗口（SDI）或子框窗口（MDI）；

（4）主框窗口（SDI）或子框窗口（MDI）创建文档对应的视。

文档—视结构中各对象的交互关系如图 1—2 所示。在文档视结构中各对象创建以后，程序员应覆盖相应类的成员函数，以对各对象做具体的初始化。在视类中覆盖 OnInitialUpdate 是初始化视的最合适方法，覆盖文档类的 OnNewDocument 和 OnOpenDocument 成员函数可以为文档做特定的初始化。

3. 消息映射

Windows 应用程序是消息驱动的，应用程序不能直接得到用户所做的操作事件，如鼠标按键、键盘输入、窗口移动等，这些操作由操作系统管理。操作系统检测到操作事件后，便向相关的应用程序发送消息，应用程序响应这些消息来完成用户的操作。

●消息

Windows 中的消息是操作系统与应用程序之间，应用程序之间，应用程序各对象之间相互控制与信息传递的方式。

消息的基本格式是：

Message wParam lParam

Messnge 是消息名称：wParam 是与消息相关的 WORD 型参数；lParam 是消息相关的 LONG 型参数。

主要有三类消息：

（1）Windows 系统消息：Windows 系统向窗口发送的消息，由窗口或视进行响应处理。这类消息包括除 WM_COMMAND 消息之外的其他名称以 WM 开始的消息；

（2）控制通知消息：控制或子窗口传给父窗口的 WM_COMMAND 通知消息；

（3）命令消息：为响应用户接口操作时，将产生 WM_COMMAND 命令消息。其参数指定了用户接口的标识号，如菜单项、按钮等 ID 号。

●消息映射

MFC 应用程序框架设置了相应的消息处理函数来响应消息，以完成相应的操作、消息处理函数是某些类（通常是窗口类）的成员函数，程序员在其中编写响应消息时应进行操作的代码。

框架将消息和它们的处理函数连接起来就是消息映射、消息映射使应用程序在接收到消息时调用对应的消息处理函数来响应和处理消息。

ClassWizard 在创建新类时将为其创建一个消息映射，并能为每个类能响应的消息和命令增加对应的处理函数。在源代码中，消息影射开始于 BEGIN_MESSAGE_MAP 宏，结束于 ENI_MESSAGE_MAP 宏，中间由一系列预定义的被称为条目宏的宏组成，其基本格式如下：

```
BEGIN_MESSAGE_MAP (Classname,parentclassname)
      //{{AFX_MSG_MAP (classnalne)
条目宏 1
条目宏 2
条目宏 3
.....
    //}}AFX_MSG_MAP
END_MESSAGE_MAP ()
```

其中 classname 为拥有消息影射的当前类名，parentclassname 为当前类的父类名。条目宏定义了类所处理的消息及与其对应的函数，常用的条目宏的类型如表 1.3.1 所示。

表 1.3.1 消息影射条目宏

消息类型	宏格式	说明
Windows 消息	ON_WM_XXXX	WM_XXXX 为 Windows 消息名
命令	ON_COMMAD(ID.FUNCTION)	ID 为命令标识号，Function 为处理函数名
更新命令	ON_UPDATE_COMMAND_UI (ID.Function)	ID 为命令标识号，Function 为处理函数名
控制通知	ON_XXXX (ID.Function)	ID 为控制标识号，Function 为处理函数名
用户定义消息	ON_MESSAGE (ID.Function)	ID 为消息标识号，Function 为处理函数名
用户注册消息	ON_REGISTERED_MESSAG(ID.Function	ID 为消息标识号，Function 为处理函数名

注意：MFC 要求所有消息处理函数声明为 afx_msg 类型。

Windows 消息的处理函数在 CWnd 类中进行了预定义，类库以消息名为基础定义这些处理函数的名称。例如，消息 WM_PAINT 的处理函数在 CWnd 类中声明如下：

```
afx_msg void OnPaint ();
```

通过 ClassWizard 在派生类中用同样的原型定义处理函数并为该函数生成消息影射条目，然后编写处理函数代码，便在派生类中覆盖了其父类的消息处理函数。在有些情况下，必须在派生类的消息处理函数中调用其父类的消息处理函数，使 Windows 和基类能对消息进行处理。ClassWizard 将在生成的处理函数中建议是否应调用父类的消息处理函数及调用的次序。

用户定义和注册的消息、命令和控制通知都没有缺省的处理函数，需要在定义时声明，一般建议根据其 ID 名称来为函数命名。

4. Visual C++可视化编程

Visual C++的资源编辑器能以所见即所得（What you see is what you get）的形式直接编辑程序用户界面，为所有资源分配 ID 标识号。ClassWizard 能把对话框模板与生成类定义或与已有的类代码连接起来，为菜单项、控制等资源生成空的处理函数模板，创建消息影射条目，并将资源 ID）与处理函数联接起来。通过使用 AppWizard，程序员的编程工作便简化为用资源编辑器直观地设计界面，完善对话框类代码，在空的处理函数模板处填写响应用户操作的代码，这是一种完善的可视化编程方法。

用 Visual C++进行 Win 32 可视化编程的基本流程如下：

（1）生成框架：运行 AppWizard，并按需要指定生成应用程序的选项，指定框架中

视类的基类（CView、CEditView、CFormView、CScrollView、CTreeView等）。Appwizard将按指定的选项生成应用程序框架和相关的文件，包括包含项目（project）的工作空间（workspace）文件和源文件，主要是应用程序（application）、文档（document）、视（view）和主框窗口（main frame）的C++代码文件（*.cpp，*.h），以及缺省包含标准界面接口的资源文件（*.rc）。

（2）设计用户界面：利用 Visual C++资源编辑器可视化地直观编辑资源文件，定制菜单、对话框、工具条、字符串、加速键、位图、图标、光标等接口资源。

（3）连接界面和代码：利用 ClassWizard 把资源文件中定义的界面资源标识（如菜单项、工具条和对话框中的控制等）在指定的源文件中映射成相应的函数模板。

（4）编写、修改函数代码：利用 ClassWizard 可以方便地在源码编辑器（source code editor）中跳转到指定的函数代码处。

（5）根据需要创建新类和编写代码：用 ClassWizard 创建新类，并生成相应的源文件。如新类是对话框类，可先用资源编辑器生成对话框模板，然后用 ClassWizard 创建对话框类代码，并与模板连接，编写新类相关的源代码。

（6）实现文档类：在 AppWizard 生成的框架基础上设计文档数据的数据结构，在文档类中增加相应的成员数据、成员函数，实现对数据的操作和文档与数据的接口。

（7）实现框架中标准的文件操作命令，即 Open、 Save 和 Save As 命令：框架已完成标准的文件操作命令的所有接口，程序员要做的仅仅是编写文档类的串行化（Serialize（））成员函数。

（8）实现视类：框架已构造好了文档与视的关系，视能方便地访问文档中的 public 数据成员。可根据文档的需要构造一个或多个视类，通过 ClassWizard 把视的用户接口资源映射成函数模板，并编写函数代码。

（9）如需要，增加分割窗口（splitter window）：在 SDI 的主框窗口类或 MDI 的子窗口类中添加一个 CSplitterWnd 对象，并在窗口类的 ONCreateClient 成员函数中对 CSplitterWnd 对象进行创建和初始化。如果用户分割了一个窗口，框架将给文档创建并增加附加的视对象。

（10）建立、调试、修改程序。如有问题，可根据需要重复步骤2－10。

（11）测试应用程序。如有问题，可根据需要重复步骤（2）～（11）。

（12）结束。

1．3．2 中文程序开发环境的安装

对于 Visual C++6．0 以前的版本，Visual C++安装程序安装的 Visual C++ Developer Studio 不能支持中文的资源编辑，在用 AppWizard 构造应用程序时提供的应用程序语种选项中也没有中文选项。如果强行在资源编辑器中输入中文，得到的将是乱码符号。为了使 Visual C++ Developef Studo 支持中文程序开发，就必须从 Visual C++安装光盘中手动安装 Visual C++的中文环境支撑链接库，安装步骤如下：

（1）退出已运行的 Visual C++ Developer Studio。

（2）拷贝以使 APPWizard 中文支持动态链接库：将 Visual C++安装光盘中 \DEVSTUDIO\SHAREIDE\Bin\Ide 目录下的 Appwzchs.dll 拷贝到硬盘中安装 Visual C++的相应目录中，通常是 C:\Program Files\DevStudio\shareIDE \bin\ide。

（3）拷贝中文 MFC 资源动态链接库：将 Visual C++安装光盘中 \DEVSTUDIO\VC\REDIST 目录下的 Mfc42chs.dll 拷贝到 Windows 95 的系统目录（Windows\System）中，并将其改名为 Mfc42loc.dll。

（4）拷贝中文 MFC 资源静态链接库：将 Visual C++安装光盘中的\DEVSTUDIO\VC\MFC\Src\L.chs 目录和\DEVSTUDIO\VC\MFC\Include\L.chs 目录拷贝到硬盘中安装 Visual C++的相应目录通常是 C:\Program Files\Devstudio\VC\MFC\Src\L.chs 和 C:\Program Flle\DevStudio\VC\MFC\Include\L.chs）下。

经上述安装后，AppWizard 在生成应用程序框架时将提供中文选项。如选择中文，则生成的应用程序的菜单、对话框、控制、字符串中都将显示中文。

用同样的方法能使 Visual C++支持日文、韩文等其他的双字节语种。

1．3．3 编程风格问题

编程风格问题是一个容易引起很大争论的问题，有些人认为自己能适应的风格就是合适的风格。但为了使程序代码易于维护，我们在编写代码时还是应遵循一定的公认准则。

●语句缩进准则

虽然没有任何一种编程语言强制要求使用缩进准则，但在实际编程中，几乎每一个程序员都使用缩进来增强代码的可读性。使用缩进的方式是千差万别的，我们建议使用 Visual C++示例中所采用的缩进方式。

●关于注释

C++同时支持/*....*/和//注释方式，我们建议对于大段注释采用/*....*/方式，对于较少行的注释在每行前加//表示注释。

虽然不要求对每行语句都进行注释，但最好对每一源文件、每个函数、每一特殊操作语 句（或语句段）、重要变量都能有注释，以说明文件、函数、语句或变量的用途和意义。

对源文件的注释应放在文件的最前面，一般包括如下几部分：

（1）文件名；
（2）文件功能、用途；
（3）创建时间；
（4）作者；
（5）修改时间；
（6）修改的目的；
（7）修改人。

●源文件设置

每个类都应有它自己的头文件（*.h 文件）来说明其类定义，而把类代码放在名称相同的*cpp 文件中。如将类 CMyClass 的定义放在 MyClass.h 中，而类代码放在 Myclass.cpp 中。在项目较大时，建议设定一个总头文件来包括其他所有头文件，而在所有代码源文件（*.cpp）中仅包括该总头文件。这样有利于防止互相冲突的定义发生。

使用常数时，应尽可能在头文件中使用常量定义，尽量避免直接在代码中使用常数。

在每一个头文件中应设置防止该头文件被多次包括的预编译条件语句，其形式如下：

```
//==========================
// Sample.h
//该头文件说明
//==========================
#ifndef _SAMPLE_H
#define _SAMPLE_H
头文件内容
```