C++实例教程

易忠亮 周建辉 编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL: http: // www. phei. co. cn

C++实例教程

易忠亮 周建辉 编著

電子工業出版社
Publishing House of Electronics Industry

内 容 提 要

90年代是面向对象的时代，从操作系统、数据库到形形色色的应用软件，到处都有它的影子，C++作为主要的面向对象开发语言也得到了空前的发展。用C++开发面向对象应用软件需要新的思想和方法，本书试图通过实例来引导读者迅速掌握这些方法，并平滑地从C过渡到C++、DOS环境过渡到Windows环境。

本书共分十章，前三章介绍面向对象与C++语言的基本知识和技术；第四、五章结合实例深入浅出地讨论了Windows应用软件的设计方法；第六章到第十章讨论作者近年用C++开发的两个科技应用软件simbase和MATCLIB，前者是一个用于仿真的基类，后者为一个用于向量和矩阵运算的可复用类库，本书结合作者的开发体会分析了在simbase和MATCLIB设计过程中涉及到的关键技术，并提供了大量源于实践的编程经验。本书全部实例程序存放在一张3.5英寸的软盘上，通过上机实践读者可迅速走完从初始概念到软件成品的整个开发过程。

本书适合于对C++和Windows软件开发有兴趣的大学生、研究生及各类工程技术人员。

书　　名：C++实例教程
编　　著：易忠亮　周建辉
责任编辑：崔国荣
排版制作：电子工业出版社排版室
印 刷 者：北京大中印刷厂
出版发行：电子工业出版社出版、发行　URL：http://www.phei.co.cn
　　北京市海淀区万寿路173信箱　邮编　100036　发行部电话68214070
经　　销：各地新华书店经销
开　　本：787×1092 1/16　印张：25.75　字数：650千字
版　　次：1997年9月第一版　1997年9月第一次印刷
书　　号：ISBN 7-5053-3995-8/TP·1747
定　　价：30.00元

前　言

面向对象技术是90年代软件开发的主流，它通过对象概念支持软件复用，为克服长期困扰工业界的软件危机提供了有效的途径。C++作为主要的面向对象开发语言之一，同时具备面向对象语言的灵活性和常规语言的效率，在各行各业均有极为广泛的应用前景。

由于面向对象是一种认识客观世界的新方法，因此，用C++开发面向对象应用软件将面临许多挑战，新的概念、新的思维方式、新的软件设计方法，这一切贯穿于软件系统构思、分析、设计、实现、维护各个阶段。仅仅掌握C++语言本身的语法结构是远远不够的，只有学会一套完整的解决问题方法，包括从现实问题中抽象和提炼对象、设计适当的类层次结构予以编程实现、再反过来解决现实问题并对原来的设计进行修正和完善等等，才算真正掌握了C++面向对象软件开发方法。

本书的基本目标是通过实例引导读者掌握用C++语言开发一个面向对象应用软件的基本方法和技术，通过分析应用软件simbase和MATCLIB设计中的关键技术，阐述了用面向对象思想解决实际问题的基本途径，并提供了大量源于实践的编程技巧。由于Windows已成为当今台式计算机操作系统的事实标准，取代DOS的趋势已不可逆转，因而，本书的应用软件均设计为DOS和Windows双平台，并用一定的篇幅介绍了Windows应用软件的基本设计方法，以帮助不熟悉Windows环境的读者从DOS平滑过渡到Windows。

本书共分为十章，第一章概要介绍面向对象技术和C++语言的特点；第二章描述Borland C++集成开发环境的使用方法；第三章用实例说明C++语言的重要特征；第四章介绍Windows环境的特点及基本Windows应用软件的设计技术；第五章阐述如何用Borland C++提供的OWL类库开发具有面向对象风格的Windows应用软件；第六、七两章全面分析了仿真基类simbase的设计思想、体系结构、实现代码以及在DOS和Windows环境中的应用实例；第八章到第十章对向量与矩阵运算类库MATCLIB的演化过程和各时期的关键技术作了较全面的描述，侧重分析与内存共享、类模板应用、动态连接库设计相关的问题；附录A列出了常用的常微分方程数值积分方法；附录B和C分别提供MATCLIB类库中向量类和矩阵类的定义。本书中的所有实例程序均可在所附软盘中找到，通过上机实践读者可迅速走完从初始概念到软件成品的整个开发过程。

为了运行书中的实例程序，需要下述硬软件的支持：

- 386以上的微机
- 与Microsoft兼容的鼠标
- VGA或更好的图形适配器
- Borland C++ 3.1及其应用框架
- MS-DOS 5.0或更高版本
- MS Windows 3.1或更高版本

由于C++标准头文件和库文件在每台计算机中的安装位置均不相同，读者在使用本书所附盘中的工程文件时，需要重新设置路径，以便编译及连接系统能正确定位头文件和库文件。

只要您对C语言有一定的了解，就能比较顺利地通过本书掌握Windows应用软件的基本设计方法，并学会用C++面向对象特性开发中等规模的科技应用软件。此外，本书中介绍的许多实际编程技巧对具有一定经验的C++软件开发人员也有重要参考价值。

本书第一章到第四章由周建辉编写，第五章到第十章、以及附录由易忠亮编写。由于我们水平有限，书中若有不当和错误之处，敬请读者批评指正。

目　录

第一章 绪 论

本章主要介绍面向对象程序设计、C++语言、Borland C++编译器和应用软件工具包等方面的基本概念及其相互关系，为读者进一步阅读和学习其它章节理清思路打好基础。

1.1 面向对象程序设计

1.1.1 什么是面向对象程序设计

面向对象程序设计(Object-Oriented Programming 即 OOP)是一种新的程序设计方法，其基本原理是充分利用人们在建立现实世界模型的实践中形成的概括、分类和抽象的能力，将复杂问题划分为模块并形成等级，分块逐级求解，编制出更加结构化模块化、可重用(Reuse)、可扩充和易维护的程序。面向对象程序设计已被公认为是解决大型复杂软件(如并行计算、人工智能和图形用户界面等)开发的有效途径，是九十年代程序设计发展的主流。

面向对象程序设计的基本概念如下：

(1) 对象

对象(Object)不外乎是一个变量。常规的变量具有一定的属性(如整型变量用于存放整数)，而面向对象程序设计中的对象既包括数据(属性)也包括作用于数据的操作(行为或方法)，所以一个对象把属性和行为封装(Encapsulation)成一个整体。对象知道如何管理自己，一个对象可以自己初始化，自己打印或完成其它任务。一般地，一个对象的数据是隐蔽的私有的因而是抽象的(即一个字串可以用数组、指针或文件来实现)，使用对象的程序常常不知道或者不关心对象的物理实现，对象的私有数据只能由对象自己的行为来操纵；应用程序只要知道对象的行为(调用接口)就能用对象进行工作，因此编程过程实现了模块化和流水线化，并且设计一个对象的程序员可以修改对象的数据实现方式和数据方式，这种修改很少影响到使用对象的程序，因而提供了软件的易维护性。

从程序设计者来看，对象是一个程序模块；从用户来看，对象为他们提供了所希望的行为。

(2) 方法

方法(methods)是对象内数据的操作接口，是应用程序使用一个对象的途径。因为只有指定的少量方法操作私有数据，因而程序调试也十分容易。

(3) 消息传递

消息传递(Message Passing)是指对象之间通信的机制。当一个消息发送给某个对象时，包含要求接收对象执行某些活动的信息，接收到消息的对象经过解释后予以响应。发送消息的对象不需要知道接收消息的对象如何对请求予以响应，只要知道接收消息的方法即可。

(4) 类

类(Class)是对一组具有相同数据和方法的对象的描述。把一组对象的共同特性再加以抽象并存贮在一个类中的能力是面向对象程序设计最重要的一点，是否建立了一个丰富的类库是衡量一个面向对象程序设计语言成熟与否的重要标志。

类是在对象之上的抽象。有了类以后，对象则是类的具体化，是类的实例。类可以有子类，同样也可以有父类，形成层次结构。

(5) 继承性

继承性(Inheritance)是类的层次结构之间共享数据和方法的机制。继承性是类之间的一种关系，在定义和实现一个类时可以在已存在的类的基础上进行，把已存在的类所定义的内容作为自己的内容，并扩充和加入新的具体的内容。继承性是面向对象程序设计语言不同于其它语言(如面向过程的语言)的最主要的特点。

有了类和继承性，就可以将较通用和公共的行为与属性定义在较高层次的类中，这样较低层次的类可以继承和重用较高层次类的特性与行为并可增加自己特有的特性与行为，可以说，面向对象程序设计是建立类的等级的过程。这种思信维方式与我们建立现实世界模型中形成的分类学十分相似，它为程序设计提供了概括、分类与抽象能力，减少了代码冗余度，增强了可重用性和易维护性，提高了软件生产效率。

例如，可以定义雇员(Employee)和经理(Manager)两个类，经理类继承了雇员类的特性与行为日增加了自己特有的级别(level)：

```
class    Employee{
         char *Aname;                    //数据
         int age;
         public:
         void DispName (int x,int y);    //方法
         Employee (char *name,int age);
         ~Employee();
         }
class    Manager:public Employee{
         int level;                      //增加的数据
         public:
         void DispLevel (int x,int y);   //增加的方法
         Manager(char *name,int age,int level);
         ~Manager(0;
         };
```

接着可以定义两个类的实例，即建立雇员对象 el 和经理对象 ml：

```
Employee el("张三",23);
Manager ml("李四",46,2);
```

有了对象及其方法，应用程序就可以向其发送消息，调用方法进行工作：

```
el. DispName(20,10);        //在(20,10)处显示 el 的名字
ml. DispLevel(40,50);       //在(40,50)处显示 ml 之级别
```

由于经理类自动继承了雇员类的行为，故可以显示 ml 的名字(尽管 ml 中没有定义显示名字的

方法)：

```
ml. DispName(40,10);      //在(40,10)处显示 ml 之名字
```

由上述例子可知，我们可以修改 Employee 类的方法 DispName(int x,int y)的代码(如用写屏或调中断等方式)，则这样修改可以完全被其子类继承，即在类层次结构之顶部修改可以做到完全自动传播；而且如果只修改方法的实现方式而保证其名字和形参不变则调用此方法的所有代码均无需改动(而面向过程的程序设计改动一处就会引起许多不一致的问题)；另外如果要增加其它对象(如保管员和司机等)，则仅需增加一些子类即可，Employee 类的代码可自动被新加的类和对象继承。

(6) 多态性

多态性(Polymorphism)是指在类的等级中同一方法(返回值、函数名、参数个数与种类均相同)的多种形态。在收到消息时对象要予以响应，不同对象收到同一消息可以有完全不同的处理方式和结果，如将坐标值消息发往不同的对象(如西文、汉字、图形)其处理方式(如打印方法)是不同的，这一现象叫做多态。在使用多态时用户可以发一个通用消息，而处理方法的实现细节则由接收对象自行决定，这样同一消息就可以调用同一方法的不同版本。由于方法(如打印)与具体对象(如西文)的关系不是固定相连，而是在运行时再决定所用的特定版本，这项技术叫动态联编(Dynamic Binding)。有了动态联编就很容易对已有工具库进行扩充(如增加汉字打印方法)，这样用面向对象思想设计的工具就很容易扩充汉化，且无需分析或知道库源代码。

与多态性相连的另一概念是多义性，有时也称之为重载(Overloading)，重载有运算符的重载与函数重载两种，重载也可用于不要库源代码而扩充其功能。多态性指不同的对象接收同一消息(如坐标)调用同一方法(如打印)之不同版本，而多义性则是指不同消息自动识别的调用同名方法的不同版本(参数个数或类型不同)；进一步说多态性是在运行时发生的，而多义性则是在编译时发生的(产生不同的内部名字)，因此，多态性更灵活。

以上列举了面向对象程序设计的几个重要的概念(有时也归纳成封装性、继承性和多态性)，其中有的在其它语言中已经采用，并不是面向对象程序设计最早定义的。例如我们所指的对象是在抽象数据类型(具有数据结构和算法的模块类型，即 ADT)基础上定义的，它在其它语言已经采用；消息也已在并发程序设计中普遍使用；运算符和函数的重载也在许多语言中使用，加法运算符就可以适用于多种数据类型；动态联编在一些解释性语言(如 Lisp)中也已采用。面向对象程序设计语言增加的是类及其继承性，另外则是把以上那些概念和特点加以扩充和提高，并把它们结合起来形成了更加特有的能力。例如对象虽然在其它语言中使用但在那里主要着重于对象之内部实现，而在面向对象程序设计语言中则着重于对象与外部的通信以及接收消息时如何响应；更为重要的是面向对象程序设计中的对象是类的特例，而类具有继承性，故有人形象的以下列公式：

```
OOP=ADT+Inheritance
```

来表示面向对象程序设计。

类的概念最早在 Simula 语言中出现，因而面向对象程序设计语言被认为发源于 Simula；

Smalltalk 则把类和子类的共享机制称之为继承性。现在普遍把具有类的继承性作面向对象程序设计语言与其它语言区别的标志。

面向对象程序设计是一种全新的思维方法，它与传统的面向过程程序设计方法(如模块化结构化)有什么本质的区别呢？其一是模块的划分和结构的组织均以数据为中心的思想，数据引导代码流，而传统面向过程序设计中是以过程为中心，以功能分析为主，在那里以操作(或方法)为主，数据处于从属地位。其二数据抽象和封装性，数据被方法封装起来，应用程序只能通过严格定义的方法操作数据，且类和继承性可将应用组成层次结构并进行访问控制；而一般面向程序设计中代码与数据被松散地组合在一起，抽象数据类(ADT)的通用特性和它的特例特性之间也没有什么区别，其抽象，分类和概括问题的能力仍不能满足应用的需要。其三，面向对象程序设计主要是可重用类库的设计与应用，且类库中类同使用语言的数据类型相似，还可以利用继承性与多态性对类进行扩充；而面向过程程序设计中的子程序库是以代码和子程序(函数)的形式出现，对它的修改与扩充就很困难。

在面向对象程序设计中，出发点是数据分析，开始的元素是对象，然而把具有相同特性的对象归纳成类并归成类库，也可把某种特定应用(人工智能、CAD 和图形用户界面)中的若干基本类组合在一起形成子类库或称为构架(Frameworks)。应用时则在已有的或自己定义的构架或类库中选择类，实例化类以后得到对象，向对象发送消息来调用对象以组成应用程序。因此，而向对象程序设计可划分成以下几个基本步骤：

a、鉴别并定义对象和类；

b、组织类之间的等级或层次关系；

c、建立可重用的类库；

d、在类库基础上建立应用构架；

e、应用上述自定义的类库或已有的类库。

以上所述的面向对象程序设计方法一般称之为数据驱动法，其它面向对象程序设计方法还有责任驱动法和事件驱动法等，在此不再赘述。

1.1.2 为什么需要面向对象程序设计

九十年代计算机硬件发展和更新速度越来越快，RISC 技术、工程工作站(WS)、网络、并行处理、多媒体技术和虚拟现实(VR)技术等新技术新结构层出不穷；应用由计算，商务处理和计算机辅助设计发展到并行分布式处理，人工智能应用，图形用户接口。应用系统规模愈来愈大，复杂程序愈来愈高，因而对软件设计提出了越来越高的要求：

a、采用更加复杂的数据结构，例如用对象描述一个并行分布式系统中的实体的特性、通信方式以及它们之间的交互作用(即竞争与合作)；用对象和消息传递来表达知识；用类与继承性来找出图形用户接口中的图形间关系，用消息传递来触发事件等等。

b、采用更加复杂的软件结构。现代的软件已不再是程序加文档简单形式，而是数据库、多种语言、网络和多个操作系统相互作用的整体，要进行如此复杂的大规模系统设计需要多层次抽象，唯有采用面向对象的技术。它所提供的从对象到类到类库，以致于专用的构架，能够支持复杂软件的开发。

c、要求提供理更高的可扩充性，软件产品要求根据用户需要进行配置，自由更新，软件要具有良好的可维护性，并且软件行业的竞争要求高速高质量地进行软件设计。这就必然要求软件划分为一系列标准的可重用的模块，用生产线方式进行组装，使软件产品真正由工艺品发展

为工业品。

软件业的梦想一直是：象用集成电路(IC)构造硬件系统那样，由软件 IC 搭成软件系统，以彻底改变软件生产重复性差低下的顽症，跟上硬件发展的速度。而传统的软件工程方法(如结构化方法，工程管理方法，功能分解和需求分析，开发工具环境支持等)均建立在行为分析与过程抽象的基础上，而行为是不稳定的，系统的需要也是变化的，以行为和过程作为出发点难以适应系统的变化，是形不成软件 IC 的！只有面向对象程序设计中的以数据为中心，以系统中相对稳定的存在为中心，进一步提高抽象性和模块性，隐蔽细节，加上独特继承性才可以开发出模块化，可重用，可扩充和易维护的代码。可以说，有了面向对象程序设计，软件 IC 不再是梦想。

1.1.3 面向对象程序设计的发展

面向对象程序设计发源于 60 年代末，典型代表是由 Algol60 发展而来的 Simula 67 语言，在 Simula 67 中首先引入了类的概念。但由于面向对象程序设计要求内存越来越大，这在当时还难以满足；并且当时软件工程概念的提出与兴起使人们忽略了面向对象程序设计思想的研究与发展。

80 年代初，人工智能研究的发展导致 Smalltalk 语言产生。Smalltalk 的很多内容借鉴了 Lisp 语言，而它的核心概念——类则取自于 Simula67，Smalltalk 最重要的贡献是发展了类的继承性。因此，真正的面向对象程序设计是 Smalltalk 语言奠定基础的，“面向对象”一词也是 Smalltalk 首先采用的，因而 Smaltalk 被认为是“纯粹”的面向对象语言。但是由于 Smalltalk 属性解释型系统，学习它需要掌握全新的知识、概念和开发环境；加之当时日本人提出“第五代计算机”大出风头，使人们又一次忽略了面向对象程序设计方法的研究。

直到 80 年代中期，Bell 实验室的 B. Stroustrup 着手在 C 语言的基础上引入类的概念并加以扩充，使之成为面向对象程序设计语言，并定名为 C++，才导致面向对象程序设计最终风行起来。C++不是纯粹的面向对象语言，但由于 C 语言在 80 年代已经成为通用的可编译程序设计语言，容易学习并有良好的开发工具，因此在 C 语言基础上扩展而成的 C++虽不是纯粹的面向对象语言，但 C++一经问世就受到普遍欢迎和广泛使用。加之 C++可用于开发 90 年代图形界面 Windows 应用程序。可以预计，C++唯一有可能成为标准化的面向对象程序设计语言。其它面向对象的语言，如 Smalltalk，Objective-C，面向对象的 Pascal，CLOS，Eiffel 和 Ada9x 等虽在某些领域和某些地区拥有一定的用户，但他们还不如 C++流行。以致于人们谈到面向对象程序设计就要联系到 C++语言，就如同人们谈到人工智能联系到 Lisp 语言一样。而实际上面向对象程序设计主要是一种新的程序设计方法，C++只是支持面向对象程序设计的语言工具而已，它实现了许多(但并非全部)OPP 方法；另外面向对象程序设计和 C++中的一些名词叫法也有区别，如 OPP 中方法在 C++叫成员函数，而 OPP 消息传递在 C++中是调用成员函数等等。

总之，面向对象程序设计在 80 年代兴起的主要原因是由软件工程发展的需要，高性能的硬件系统又提供了可能，特别是由 C 发展而来的 C++本身良好的性质等诸多因素决定的。现在面向对象程序设计语言(OOPL)特别是 C++已很流行，面向对象程序设计(OOP)也已深入人心，面向对象分析(OOA)、面向对象设计(OOD)和面向对象软件设计(OOSD)等深层次问题也逐渐引起了人们的兴趣与重视，新一代面向对象数据库(OODB)和面向对象操作系统(OOOS)也斩露头角，面向对象程序设计全面超越 80 年代全盛的结构化程序设计的时期不

远了。

1.2 C++语言

1.2.1 C++语言的主要特点

C++是以C语言为基础的支持面向对象的通用可编译程序设计语言。C++保持了C的紧凑、灵活、高效和可移植性等优点，并且增加了封装、继承和多态性等优良的面向对象特性，可支持大规模复杂软件开发。C++主要由类支持封装和继承性，而C++多态性是由虚拟函数支持。

C++对C语言的扩充部分绝大多数来自许多著名语言中的很优秀的特性。C++语言从Simula 67中吸取了类；从Algol 68中吸取了运算符重载，引用和在任何地方申明变量；综合了Ada的类属和Clu的模块特点形成了C++的模板(Template)；从Ada，Clu和ML中吸取了异常事件处理；多BCPL中吸取了用//表示注释。

由于C++既有面向对象的能力，又比其它面向对象的语言(如Smalltalk，Eiffel和CLOS等)的运行性能高得多；加上C语言的普及，由C过渡到C++较为平滑；C++与C的兼容可使数量巨大(占全部软件的60%以上)的C程序能方便地在C++环境中应用；有大量的优秀的C++编译器产品(如MSC7.0和Borland C++等)；C++支持Microsoft Windows编程等等。所有这一切使得C++能在短短的几年内迅速流行，成为当今OOP中的主流语言。

1.2.2 C++语言的产生和发展

C++语言最早是由AT&T的Bell实验室Bjarne Stroustrup于1980年设计和实现的，最初称之为C with Class(其预处理程序为Cpre)，它在C的基础上增加了如下设施：

a、类(Class)；

b、派生类(derived classes)；

c、公用/私有(public/private)访问控制；

d、构造函数(constructor)和析构函数(destructor)；

e、友类(friend class)；

f、内联函数(inline function)；

g、函数的默认参数(functions with default values)；

h、赋值运算符重载。

在C with Class取得成功之后，Stroustrup对之进行了进一步的改进与扩充，并在1983年11月定稿的论文(Data Abstraction in C，Bell Labs Technical Journal Vol. 63，No. 8，October 1984)中正式使用了C++的名称。

1985年10月，发布了C++编译程序前端处理器Crfont 1.0，它在C with Class的基础上主要增加了以下设施：

a、虚拟函数(virtual function)；

b、函数名和运算符重载；

c、引用(reference)；

d、常量(const)。

1986 年 6 月和 1987 年 2 月相继发表了 Cfront 1.1 和 Cfront 1.2,1986 年在 Stroustrup 的第一本书《The C++ Programming Language》中给出了 1.x 版 C++语言的详细定义。

1989 年 6 月发表了 Cfront 2.0,它与 1.x 的主要差别是增加了多重继承(Multiple inheritance)和保护成员(Protected).1991 年 9 月发表了 Cfront 3.0,正式引入了模板(Template)。93 年发布的 Cfront 4.0 具有异常处理(exception handing)功能。预计今年将推出 cfront 5.0 引入运行时类型的识别。

1.2.3 C++编译器

自 1987 年后,在 AT&T 完善 C++的同时,许多厂商开始提供 C++编译器产品,主要有:

a、1987 年 1 月,GNU 发布了 C++1.13;

b、1988 年 1 月,Tau Metric C++(oreGon software C++)发布;

c、1988 年 6 月,Zortech 发布了自行开发的 C++编译器,在此之前所有 PC 机上的 C++编译器均是 Cfront 移值;

d、1990 年 5 月,Borland C++1.0 发布;

e、1992 年 3 月,Microsoft C/C++7.0 发布;

另外,DEC,IBM,SUN 和 HP 等国际知名计算机公司也相继推出了自行开发的或从 Cfront 移值的 C++编译器产品。

我们说 C++语言时着重于描述其词法与语义的定义,而 C++编译器则更多是指支持 C++语言编译,连接和生成执行程序的软件产品。一般地,C++编译器均能基本实现标准 C++语言定义并加进自己的一些改变或扩充。我们用 Borland C++更多地指 Borland 公司的 C++编译器产品,它是 C++语言的一个实现。本书主要以 Borland C++来介绍 C++语言及其应用编程。

1.3 Borland C++编译器

Borland 国际公司早在 1990 年 5 月就推出了一个优秀的 C++编译器产品 Borland C++1.0,不适应 Microsoft Windwos 3.0 应用程序开发的需要于 1991 年 2 月推出了 Borland C++2.0,其主要特色有:

a、C++:Borland C++C++2.0 提供了 C++语言的全部功能(AT&AT C++2.0 的实现),并提供了一个 C++类库和对 C++ 1.2 版流的支持;

b、ANSI C:Borland C++ 2.0 提供了最新的 ANSI C 标准的实现并增加了对 IBM PC 环境的支持(如混合模式和混合语言编程等);

c、支持 Microsoft Windows 应用编程,特别增加的资源开发管理工具和资源编译器,可以方便地生成 Windons 3.0 应用程序(EXE 或 DLL 等);

d、独特的予编译头文件可以缩短程序编译时间;

e、提供的实模式和保护模式的命令行编译器和集成编译器(共四种编译器);

f、类库中提供了集合和数组等类型;

g、新的集成开发环境(IDE):支持鼠标操作,支持多文件编辑,内嵌的汇编器和集成的调试器;

h、大量精心设计的 DOS 和 windows 样本程序

i、新的联机帮助,且可以将联机帮助中的实例程序拷贝到编辑器中运行或拷贝到用户程