SAMS PUBLISHING CMP

Teach Yourself C++ in 24 Hours

自 学 教 程

全书结构清楚、易于理解，共分24个课时，每课时为一章

简 单 易 懂

可供C++初学者及具有一定C++使用经验的应用程序开发人员使用

(美) Jesse Liberty 著
路 明 等译

C++自学通

机 械 工 业 出 版 社 西蒙与舒斯特国际出版公司

C++自学通

丁·利伯利
（美）Jesse Liberty 著
路 明 等译

机 械 工 业 出 版 社
西蒙与舒斯特国际出版公司

本书是一本自学 C + + 编程的教程，分为 24 个课时，每课时为一章。书中介绍了有关 I/O 处理、循环和数组、面向对象的程序设计、模块和创建 C + + 应用程序等内容。

全书结构清楚、易于理解，无论是初学者还是已经有了一定 C + + 程序设计的经验，本书都将令你受益匪浅。

本书版权登记号：图字：01-97-1247

图书在版编目(CIP)数据

C + + 自学通 （美）利百利(Liberty, J.)著；路明等译．－北京：机械工业出版社，1997.12
书名原文：Teach Yourself C + + in 24 Hours
ISBN 7-111-06013-X

Ⅰ.C… Ⅱ.①利… ②路… Ⅲ.C 语言－程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(97)第 23232 号

出 版 人：马九荣(北京市百万庄南街 1 号 邮政编码 100037)
责任编辑：江 颖
昌平环球印刷厂印刷 · 新华书店北京发行所发行
1997 年 12 月第 1 版第 1 次印刷
787mm × 1092mm1 16 · 22 印张 · 530 千字
印 数：0001－6000 册
定 价：38.00 元

译 者 序

欢迎阅读《C++自学通》。本书提供了一种新的自学C++程序设计的方式。全书共分为24章，可以在24学时内完成，每章1学时。本书中涉及到了目前C++发展中的最新特征，由于把一些复杂的内容拆分为小单元，使读者能够轻而易举的理解并掌握一些新的技巧。它可以作为初学者的C++教程。书中还介绍了有关I/O处理、循环和数组、面向对象的程序设计、模块和创建C++应用程序等内容。在翻译的过程中，译者对于原书中某些概念模糊的地方进行了修正，并对原文做了一些必要的删改。

参与翻译的人员有：路明、曲泽阳、赵劲松、崔军、梅嘉、聂志明等同志，由于水平有限，时间仓促，错误在所难免，欢迎读者提出批评和建议。

译 者

1997年9月

前　言

本书是一本自学C++编程的教程，分为24个课时，每课时为一章。书中介绍了有关I/O处理、循环和数组、面向对象的程序设计、模板和创建C++应用程序等内容。全书结构清楚，易于理解。为了说明主题思想，在每一章中提供了程序清单、完整的输出和对程序的分析。而语法的示例则在段落中被引用。

为了帮助理解，每一章的结尾都提出了一些常见的问题并作了回答。

谁应阅读本书

在阅读本书之前不需要任何C++程序设计的经验。本书从最基本的内容出发，同时介绍了有关C++程序设计的语法和概念。书中所提的供大量语法示例和程序分析对于初学者来说非常有益。无论你是初学还是已经有了一定程序设计的经验，通过本书都能快速而方便的学习C++。

约定

注意：此项信息可以使C++程序的效率更高。

警告：此项内容提示读者注意在某些特殊的情况下可能产生的问题或副作用。

重点概括：此项内容总结了每章中的一些基本原则，不要忽视这些有用的信息。

在本书的程序清单中，每个实际的代码行都被加了行号，在清单中没有行号的行是前面带行号行的继续(有些代码行相对于本书的宽度来说太长)。在这种情况下，实际输入时应把这两行合并为一行，而不要分开。

目　录

第二部分 类

第三部分 内存管理

第四部分 功能强大的工具

第五部分 继承和多态

第六部分 特别主题

第七部分 高级主题

第八部分 附 录

第一部分　介绍C++

第1章　入　　门

欢迎阅读《C++自学通》，在这一章中将介绍：

1）为什么C++会成为软件开发领域最著名的标准？

2）开发一个C++程序的步骤。

3）如何输入、编译和链接第一个C++程序？

1.1　为什么C++是最佳选择

C++是多数专业程序员所选择的开发语言。这是因为它提供了一个具有可移植性的强有力的开发环境用于生成快速简洁的程序。如今的C++工具使创建复杂的、功能强大的商品化应用软件成为相当简单的流水作业。为了充分掌握C++，你需要对这种功能强大的语言有一定的了解。

C++是一种相对较新的语言。当然，程序设计本身只有40多年的历史，在此期间计算机语言经历了几个重要的演变过程。

最早，程序员使用最原始的计算机指令：机器语言。这些指令用一长串由1和0组成的数码表示。不久，发明了汇编语言，它可以将机器指令映射为一些能被人读懂，易于处理的助记符，如ADD和MOV。

新术语：类似BASIC和COBEL的高级语言及时地发展起来了。这些语言使人们可以用一些接近自然语言中单词和句子的表达方式来编制程序，如Let 1 = 100。这些指令被解释程序和编译程序翻译为机器语言。类似BASIC的解释程序，一边读入程序一边翻译，直接执行程序的指令或代码。

新术语：新型的编译程序将程序转换为目标代码。这种转换过程的第一步称为编译（compiling），通过编译程序（compiler）产生目标文件。第二步称为链接（linking），通过链接程序（linker）将目标文件转换为可执行的程序，也就是可以在我们的操作系统上运行的程序。

由于解释程序在输入的同时读取代码，并现场执行，对于程序员来说采用这种方式工作更加容易。而编译程序增加了如编译和链接等额外的步骤，不如解释程序那样方便。但另一方面，编译程序可以产生运行速度很快的程序。

多年以来，计算机程序员的主要目标是力求写出短小的代码使运行速度更快。因为存储器的成本和实现处理功能的成本都很高，所以需要短小而快速的程序。当计算机变得更小、更廉

价、速度更快，存储器的价格也随之降低时，这种首选目标就发生了变化。如今基于程序员工作时间的成本远远超过了用于处理业务的计算机的成本。具有良好的编写风格、易于维护的代码受到欢迎。所谓易于维护是指当业务需求发生变化时，不需要太多的开销就可以扩展和增强程序的功能。

1.1.1 过程化、结构化和面向对象的程序设计

新术语：在过程化的程序设计(procedural programming)中，程序被认为是在一个数据集合上进行的一系列操作。而结构化的程序设计(structured programming)提供了系统的组织这些过程并能够处理大量数据的有效手段。

结构化的程序设计的主要思想其实很简单，就是功能分解并逐步求精。当一些任务十分复杂以至于无法描述时，可以将它拆分为一系列较小的功能部件，直到这些自完备的子任务小到易于理解的程度。

例如，计算一个公司中每一位雇员的平均工资是一项颇为复杂的任务。然而，可以将其拆分为以下的子任务：

1）找出每个人的收入。

2）计算总共有多少雇员。

3）计算工资总额。

4）用雇员人数去除工资总额。

得到每一位雇员记录的任务又可依次拆分为：

1）打开存储雇员记录的文件。

2）找到正确的记录。

3）从磁盘中读入数据。

结构化的程序设计成功的为处理复杂问题提供了有力的手段。

然而，依然存在着一些问题。当数据量增大时，数据与处理这些数据的方法之间的分离使程序变得越来越难以理解。对数据处理能力的需求越强，这种分离所造成的负作用越显著。

采用过程化程序设计方法的程序员发现每一种相对于老问题的新方法都要带来额外的开销。与可重用性相对，通常这被称为重复投入。基于可重用性的思想是指建立一些具有已知特性的部件，在需要时可以插入到程序之中。这是一种依照硬件组合的方式而建立的范型——当一位工程师需要一个新的晶体管时，他不用自己去发明，他只要到储藏晶体管的仓库找到直接或经过改动后符合他工作需要的那种类型。对于软件工程师来说，在面向对象的程序设计出现之前，一直缺乏具备这种能力的工具。

新术语：面向对象的程序设计(object - oriented programming)的实质是把数据和处理这些数据的过程合并为一个单独的“对象”——一个具有确定特性的自完备的实体。

1.1.2 C++和面向对象的程序设计

C++完全支持面向对象的程序设计，它包含了面向对象的开发所具有的四个重要特征：封装、数据隐蔽、继承和多态。

1. 封装和数据隐蔽

当一个工程师制造一台新的设备时，他将各个元件组装在一起。他或许需要用导线连接一个电阻、一个电容和一个晶体管。晶体管具有一定的特性，可以实现一定的功能。他可以直接使

用晶体管，而不必知道它具体是如何工作的。

新术语：为了实现这一点，晶体管必须是自完备(self - contained)的。它必须能够完全实现一些严格定义的功能。这种完全实现某些功能的能力被称为自完备。

C + +通过生成被称为类的用户定义的类型来支持封装和数据隐蔽。一个严格定义的类一旦生成后，就成为一个封装的实体，作为一个整体被使用。而类内部的行为被隐藏起来，用户只要知道如何使用这种严格定义的类，而不必知道它是如何工作的。在第 6 章“基本类”中将介绍如何创建类。

2. 继承与重用

在 80 年代末，我为花旗银行建立一种用于家庭储蓄的设备。我们不希望从零开始，而是希望能够设计出可以迅速向市场推广的产品。因此，我们从增强电话的功能入手。新的增强型电话只是在普通电话之上增加了一些新特点，这样，我们可以利用所有老式普通电话的功能，又可以通过新的功能扩展它的用途。

新术语：C + +通过继承(inheritance)来支持重用。可以说明一个新的类型，此类型是已有类型的扩展。这个新的子类从已有类型派生，有时被称为派生类型。那种具有增强功能的电话就是由老式普通电话派生出来，继承了它所有的特性，但可以根据需要增强它的功能。在第 16 章“继承”中将讨论继承和它的应用。

3. 多态

与普通电话不同，当你接到一个电话时，增强型电话(ET)不会响铃，它的屏幕自动变亮并用声音指示“有你的电话”。电话公司并不了解这些，它只是通过电线向普通电话铃、电子电话铃以及增强型电话通话器发出脉冲信号。每一部电话基于自身对电话公司所发信号的解释做出正确的响应。

新术语：C + + 通过被称为函数多态(function polymorphism)与类多态(class polymorphism)的特征来支持这种“不同类型有各自的响应”的思想，单词 Polymorphism 中，Poly 是许多的意思，morph 的含义是形式，多态是指同名但具有许多种形式的事物。在第 17 章“多态与派生类”，第 18 章“高级多态”中将进行讨论。

1.1.3 C + +是如何发展起来的

当面向对象分析、设计和编程开始流行，Bjarne Stroustrnp 在商品化软件开发中最为通用的 C 语言中加入了便于进行面向对象程序设计的特征，创建了 C + +。在不到 10 年的时间内，它从一种仅仅由少数 AT& T 公司的程序员所使用的工具发展成为全世界约上百万程序员所选择的程序设计语言。

1.1.4 C + +不仅仅是更好的 C

的确，C + +是 C 的超集，事实上，合法的 C 程序也是合法的C + + 程序，但不要被这种表象所迷惑。正如 H. L. Mencken 说 Wagner 的音乐比它听起来更好，从 C 到 C + +的飞跃比字面上看起来要大得多。

多年来，C + +得益于它与 C 语言的关系，一个 C 程序员可以很容易的使用 C + +。然而，许多程序员发现，为了完全发挥C + + 的优点，需要“忘记”以前学过的很多东西，而去学习全新的概念表述方式和解决程序设计中问题的方法。

1.1.5 是否应当首先学习 C

问题不可避免的提出：由于 C + +是 C 的超集，是否应当首先学习 C? Stroustrup 和大多数

C++程序员同意下面的观点：没有必要先学习C语言，甚至可以说这并不是一个好的想法。本书假定你不是一个C程序员。即使你是一个C程序员，也没有关系，反复阅读最初的几章，然后集中注意力往下学习。

1.2 为编程作准备

相对于其他的语言，C++或许更需要程序员在写程序之前进行设计。类似本书最初几章中所出现的小程序，不需要更多的设计工作。但对于专业程序员每天与之打交道的复杂程序来说，则必须进行设计。设计工作进行得越完备，程序越能够在时间和预算的限制范围之内解决设计中期望解决的问题。良好的设计使程序具有相对较少的错误并易于维护。据估计在软件开发中90%的开销是用于调试和维护。良好的设计在某种程度上减少了这种开销，也就显著压缩了项目开发的费用。

当准备编制任何程序之前所提出的第一个问题应该是：我试图解决何种问题？每一个程序都应有一个明确的、一致的目标，在本书中即使是最简单的程序也是如此。

程序员所提出的第二个问题是：这种目标是否可以不通过编写新的软件来实现？与重新编制软件相比，重用旧的程序、使用笔或纸（手工完成）或购买货架之上的商品软件往往是解决一个问题更好的途径。那些可以提供这些替代品的软件开发者永远不会失业，寻找解决现有问题的更为廉价的方法，通常都会在今后产生新的机会。

假定所要解决的问题必须通过编制一个新程序来实现，就可以准备开始设计工作了。

1.3 区分C++、ANSI C++、Windows和其他领域的一些概念

C++是一种语言，DOS、Windows、UNIX和MacOS都是操作系统，当学习C++时你需要知道它是一种具有可移植性的语言，与运行程序时所用的机器和操作系统无关。

新术语：本书没有假定你所用的是什么操作系统，它将介绍ANSI C++，也就是标准C++的另一种称谓。标准C++是指可以移植于任何平台和开发环境之上的国际统一的标准。

因此，在本书中没有关于窗口、列表框等诸如此类的描述。所有都是独立于操作系统的，结果的输出将以标准输出的形式完成。

为了使编译程序正常工作，需要告诉它建立一个应用软件控制平台。在某些用于Windows或Mac或其他类型的视窗化环境的编译程序中，这种控制平台被称为快捷窗口或原始窗口，也许就叫控制平台窗口。

编译程序可能本身具有内置的文本编辑器，也可以使用一个商品化的文本编辑器或能够产生文本文件的字处理软件。无论使用哪种工具输入程序，最重要的是必须以单一的纯文本的格式存储，而不能嵌入字处理软件的格式命令。具有存储功能的编辑器包括DOS中的Edit命令、Brief、Epsilon、EMACS、vi等。很多商品化的字处理软件，如Wordperfect、Word以及其他的一些软件，都提供了存储单一文本文件的方法。

用编辑器生成的文件称为源文件，对于C++源文件，它们通常以.CPP、.CP或.C作为扩展名，但应根据具体编译程序的要求而定。

注意：多数C++编译程序不关心源代码文件的扩展名，但除非指定，否则大多都以.CPP做为缺省扩展名。

重点概括:

1) 用一个简单的文本编辑器生成源代码,或使用编译程序所带的内置编辑器。

2) 不要使用可能存储特殊格式符号的字处理软件。如果使用字处理软件,以 ASCII 文本的形式存储文件。

3) 以 .C、.CP、.CPP 作为扩展名存储文件。

4) 查阅关于编译程序和链接程序的说明文档,以确定如何编译和链接程序。

1.4 编译和链接源代码

虽然文件中的源代码多少有一些晦涩难懂,对于不了解 C++ 的人来说很难理解它们代表什么,但它仍然是一种具有可读性的形式。源代码文件不是程序,不能够像程序一样被执行或运行。

注意:在 C++ 中使用编译程序将源代码转化为程序。如何调用编译程序,以及如何告知它到哪里去寻找源代码,随编译程序的不同而有所区别,查阅有关文档。

如果在操作系统命令行编译源代码,应输入如下的命令:

Borland C++ 编译程序: bc(文件名)

Borland C++ for Windows 编译程序: bcc(文件名)

Borland Turbo C++ 编译程序: tc(文件名)

Microsoft 编译程序: cl(文件名)

1.4.1 在集成开发环境中进行编译

大多数现代的编译程序提供了一个集成开发环境。在这样一个环境中,一般是从菜单中选择 Build 或 Compile 命令,或是通过按下一个功能键来创建应用软件。这也需要查阅关于特定编译程序的文档。

1.4.2 链接程序

源代码被编译后会生成一个目标文件。这个文件通常以 .OBJ 作为扩展名。这仍然不是一个可执行的程序,为了把它转换为可执行程序,必须远行链接程序。

C++ 程序通常是通过同时链接一个或几个目标文件与一个或几个库而创建的。库是由编译程序提供的,或你另外购买的,它是创建程序时所需的可链接文件的集合。所有的 C++ 编译程序都提供了这类库,其中包含一些有用的函数(或过程)和类,在程序中可以使用它们。一个函数是能够实现某种功能的一段代码,如把数字相加或向屏幕输出数据等。一个类是一套数据和与之相关的函数的集合。关于类本书中还要进行更多的讨论。

生成一个可执行文件的步骤是:

1) 生成一个以 .CPP 为扩展名的源代码文件。

2) 编译源代码,生成以 .OBJ 为扩展名的目标文件。

3) 链接目标文件和所需的库以产生一个可执行的程序。

1.5 开发过程

如果从最初阶段开始编制一个程序,就存在着一个开发过程:输入程序、编译原代码、链接程序并运行。可惜,几乎每个程序,不论多么简单,都会存在一些错误和死区。有些错误会导致

编译失败，另一些会导致链接失败，还有一些错误只有在运行时才会显示出来。

无论发现的错误是什么类型，都必须修正它，这包括编辑源代码、重新编译和链接，然后重新运行程序。这种开发的步骤以示意图的形式表示，如图 1-1 所示。

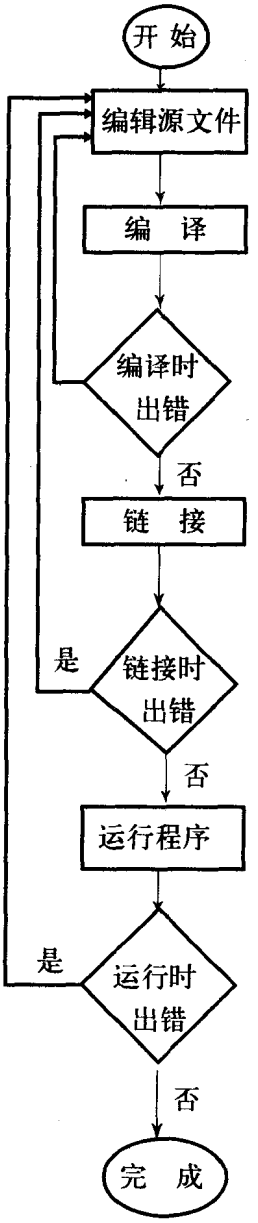

图 1-1 开发 C++程序的步骤

1.6 HELLO.CPP——第一个 C++程序

传统的关于程序设计的书籍通常以一个在屏幕上显示许多的“Hello”或其他语句的程序

作为开始,这次我们也不例外。

准确的按照下面所列出的清单,把第一个程序直接输入编辑器,确认正确以后,将文件存盘,然后编译、链接,并执行它。它将在屏幕上显示"Hello world"这个词组。不要过分关心它是如何工作的,这个程序只是想使你熟悉一下开发过程。在接下来的两章中,有关这个程序的方方面面都会被详细的说明。

注意: 下面的程序中每一行的最左边标有行号,这些数字是在本书中加入的参照行号,不需要被输入到编辑器中。例如,程序清单 1-1 中的第 1 行,应当输入:

#Include iostream. h

程序清单 1-1 HELLO. CPP, Hello World 程序

```
1: #include <iostream. h>
2:
3: int main()
4: {
5:     cout < <"Hello World! \n";
6:     return 0;
7: }
```

按清单所示确保输入的准确性,注意标点符号。第 5 行中的" < <"是重定向符号,在大多数键盘中可以通过按住 Shift 键再按下逗号键来输入,第 5 行以一个分号作为结束,不要遗漏。

确认你按照编译程序的引导进行了正确的操作。多数编译程序都会自动链接,但这需要查阅说明文档,如果发生错误,仔细检查代码并找出它与清单 1 - 1 中的区别。如果发现第 1 行有错误,如"can not find file iostream. h",查阅编译程序的说明文档中关于如何设置 Include 路径和环境变量的部分。如果得到的错误信息是"There is no prototype for main",在第 3 行之前加入一行"int main()"的代码。需要在本书中所有程序的 main()函数开始之前加入这一行代码。大多数编译程序不需要这行代码,但有少数需要。

最终的程序将是这样:

```
1: #include <iostream. h>
2:
3: int main();
4: {
5: cout < <"Hello World! \n";
6: return 0;
7: }
```

试着运行 HELLO. EXE,它将直接在屏幕显示:

Hello World!

如果是这样,那么祝贺你!你成功的完成了输入、编译和链接,并运行了第一个 C + +程序. 这虽然不能意味更多,但每一个专业的 C + +程序员几乎都是从这个程序开始的。