

21世纪

高等院校计算机系列教材

C++

基础教程

[美] M. T. Skinner 著
英宇 周辉 译

The C++ Primer



中国水利水电出版社
www.waterpub.com.cn

21 世纪高等院校计算机系列教材

C++基础教程

[美] M. T. Skinner 著

英 宇 周 辉 译

中国水利水电出版社

内 容 提 要

C++是一种最通用的编程语言,本书详细讲述了使用C++进行编程的基本知识。全书共分为16章,作者尽量避免赘述比较深奥的C++内容,详细讨论了C++的基础知识、类型、运算符和表达式、语句、函数、数组、指针、结构体、类、成员、继承、友元、类型转换、重载、输入/输出、模块、可变参数函数、信号、处理器、面向对象编程和C++库等内容。本书还包括了大量经过验证的例子,通过练习,读者将可更牢固地掌握所学知识。

本书是一本非常好的C++入门书,适合作为大学低年级学生C++课程的教材,也可供初学C++的读者阅读。

Translations rights arranged with the permission of Silicon Press.

北京市版权局著作权登记号:图字01-2003-0777号

图书在版编目(CIP)数据

C++基础教程 / (美)斯金纳(Skinner, M. T.)著;英宇,周辉译.

—北京:中国水利水电出版社,2003

(21世纪高等院校计算机系列教材)

ISBN 7-5084-1556-6

I. C… II. ①斯…②英…③周… III. C语言—程序设计—高等学校—教材 IV. TP312

中国版本图书馆CIP数据核字(2003)第046568号

书 名	C++基础教程
作 者	[美] M. T. Skinner
译 者	英宇 周辉 等译
出版、发行	中国水利水电出版社(北京市三里河路6号 100044) 网址: www.waterpub.com.cn E-mail: mchannel@public3.bta.net.cn (万水) sale@waterpub.com.cn
经 售	电话: (010) 63202266 (总机)、68331835 (营销中心)、82562819 (万水) 全国各地新华书店和相关出版物销售网点
排 版	北京万水电子信息有限公司
印 刷	北京市天竺颖华印刷厂
规 格	787×1092毫米 16开本 14印张 305千字
版 次	2003年7月第一版 2003年7月北京第一次印刷
印 数	0001—5000册
定 价	20.00元

凡购买我社图书,如有缺页、倒页、脱页的,本社发行部负责调换
版权所有·侵权必究

译者序

作为一种通用的系统程序设计语言，C++如今已经得到了广泛的认可。C++语言的设计目标是提供一种功能强大的系统开发工具，从而让设计者能够使用它解决工程实践中的一些实际问题。C++蕴涵了很多重要的思想，其中包括：数据抽象、对象模型、类型系统、继承框架等。所以，学习和掌握 C++编程语言成为大部分优秀程序员的“必修课”。

Skinner 先生所编写的这套 C++教程共分两个读本，本书为初级读本，主要介绍 C++的基础知识。作者详细讨论基本的 C++功能，让程序员充分体验 C++的强大功能和优点。此外，本书尽量避免赘述比较深奥的 C++功能，而是详细讨论了类型、运算符和表达式、语句、函数、数组、指针、结构体、类、成员、继承、友元、类型转换、重载、输入/输出、模块、可变参数函数、预处理器和 C++库等基础知识。同时，书中还提供了大量经过实际验证了的编程实例，以供读者参考和理解。在掌握本书之后，有兴趣的读者可以继续阅读作者关于 C++的高级读本《C++高级教程》[Skinner 1992]，该书中文版与本书中文版由中国水利水电出版社同步出版。

本书的主要内容如下：第 1 章介绍了 C++程序的编写、编译工具、求二次方程根的例子；第 2 章介绍了字符集、标识符、常量、注释；第 3 章介绍了 C++提供的基本类型及其他类型、类型的定义方法、对象的概念；第 4 章介绍了各种运算符的意义及其结合性、优先级；表达式的构成和计算方式；第 5 章介绍了声明和定义、语句（条件语句、循环语句、跳转语句、赋值语句）；第 6 章介绍了函数的定义、声明、函数的原型、参数传递（传值与传址）、局部变量与全局变量、作用域与生命期；第 7 章介绍了数组、结构体、指针的概念及其用法、链表、字符串、操作符 new 和 delete；第 8 章介绍了类的定义、类的对象、类的成员、堆栈、友元；第 9 章介绍了继承和派生、多重继承、虚函数的运用、一个员工数据的例子；第 10 章介绍了重载的用途、运算符重载；第 11 章介绍了输入与输出、操纵符、流状态；第 12 章介绍了编写一个“对照发生器”程序；第 13 章介绍了模板的定义和声明、函数模板、类模板；第 14 章介绍了可变参数函数；第 15 章介绍了预处理器、宏定义、文件包含、条件编译、避免多重包含；第 16 章介绍了 C++库功能、标准函数。

本书的每一章后面都附有练习题，具有很强的针对性，能够帮助读者复习和巩固所学的内容。本书非常适合于 C++入门的读者阅读。

经过数月的艰苦努力，本书的译稿终于完成并交付出版。如果本书能给广大读者带来益处，那么译者付出的辛苦也就算得到了回报。本书由周辉主译，参与审校和录入排版工作的有欧阳宇、李明、盛海燕、谢小花。在本书出版之际，也对他们表示衷心的感谢。由于时间仓促，译文中难免出现错误和不当之处，敬请广大读者批评指正。

译者

2003 年 6 月

序 言

编程语言是程序员或用户命令计算机执行相应动作的“用户接口”。一个计算机“程序”是用某种编程语言编写的、能够完成某项任务的一系列指令。一方面，“通用”的编程语言可以针对不同的应用（问题）域编写相应的程序。另一方面，专门的编程语言只能用于特定的应用域。

C++是一种通用的编程语言，它由 AT&T 贝尔实验室的 Bjarne Stroustrup[1991]¹设计完成。C++是对应用非常广泛的 C 编程语言[Kernighan & Ritchie 1989]的继承和发展。虽然 C++是基于 C 的，但它绝不是新版本的 C！C++是新一代的编程语言。

C++提供了多种功能，用于：

- 定义常量和变量。
- 变量初始化。
- 赋值。
- 表达式说明。
- 条件执行。
- 循环。
- 函数定义和调用。
- 数据抽象。
- 多重继承。
- 传址方式参数传递。
- 初始化。
- 自动清除。
- 操作符和函数重载。
- 类型转换。
- 写模块。

除了提供比 C 更强大的类型检查功能以外，比如“函数原型”。C++还提供了“面向对象”编程功能。在面向对象的编程中，程序的关注焦点是数据，也就是程序中的对象；而在传统编程中，程序的关注焦点则是数据操作代码。现在，由于面向对象的编程能很好地支持代码重用，进而加速模块和扩展程序的开发，所以 C++被认为是一种强大的编程技术。

面向对象的编程语言，其主要特点就是提供了数据抽象和继承的功能。有了这些功能提供的概念和符号，程序员就能很方便地写出针对某个应用域的应用程序。例如使用数据抽象概念，程序员可以很好地对应用域对象进行建模，比如输入/输出设备、自动装置和员

¹方括号用于引用参考书目中的文献，读者可以在这些参考书目中获得详细信息。

工等。使用继承则可以保留相关对象类型之间的关系，比如员工和管理人员。

1. 关于本书

本书是 C++ 的入门书。虽然本书不要求读者有编程预备知识，但如果有这方面的基础，理解本书内容，掌握 C++ 编程就要更快捷容易得多。本书的内容是基于一本名为《The Advanced C++ Book》[Skinner 1992] 的书。

C++ 提供了很多功能，可以让程序员写出简洁而高效的程序。本书只讨论程序员使用 C++ 编程时经常用到的功能。本书尽量避免赘述比较深奥的 C++ 功能，因为那些知识只有高级程序员才需要掌握。简单介绍 C++ 之后，我们将详细讨论下面这些问题：

- 基。
- 类型。
- 操作符和表达式。
- 语句。
- 函数。
- 数组。
- 指针。
- 结构体。
- 类。
- 成员。
- 继承。
- 友元。
- 类型转换。
- 重载。
- 流输入/输出。
- 模块。
- 可变参数函数。
- 预处理器。
- C++ 库。

书中包括大量的例子程序，这些程序都经过了实际验证。书中超过 5 行的 C++ 头文件或源文件，都在代码的左侧标出相应的行号，便于读者查阅，它们不属于 C++ 代码。

读者可与 Silicon Press 联系，以获得包括本书所有例子源代码的磁盘。书中之所以指定例子源代码所在文件的文件名，是为了：

- 便于查阅，因为 C++ 文件还经常包含其他文件。
- 方便从磁盘上找到所需的程序。

2. 本书中的 C++ 版本描述

C++ 是根据用户需求不断发展的一种编程语言。本书所用的 C++ 版本是最新版本的 C++ [Stroustrup 1991; Ellis & Stroustrup 1990]，这个版本被当作 ANSI 标准化 C++ 的基础，它是《The C++ Programming Language》[Stroustrup 1986] 一书介绍的原始 C++ 的升级版本。

3. 致谢

非常感谢 Dick Differderfer 和其他审稿者的建议，并感谢 Silicon Press 鼓励我最终写成这本书。

M.T.Skinner

目 录

译者序

序言

第1章 简介	1
1.1 用C++编写程序	2
1.1.1 打印输出	2
1.1.2 计算二次方程的根	4
1.2 程序终止	10
1.3 练习	10
第2章 基础知识	11
2.1 字符集	11
2.2 标识符	12
2.2.1 关键字	13
2.3 常数	13
2.4 注释	14
2.5 练习	15
第3章 类型和对象	16
3.1 基本类型	16
3.1.1 字符型	16
3.1.2 枚举类型	17
3.1.3 整型	18
3.1.4 浮点型	19
3.2 其他类型	20
3.2.1 无符号类型	20
3.2.2 空类型	21
3.2.3 派生(复合)类型	21
3.3 对象	21
3.4 类型定义	21
3.5 练习	23
第4章 运算符和表达式	24
4.1 运算符	25
4.1.1 算术运算符	25
4.1.2 比较运算符	27
4.1.3 关系运算符	28

4.1.4	逻辑运算符	29
4.1.5	加 1 和减 1 运算符	30
4.1.6	sizeof 运算符	31
4.1.7	条件运算符	32
4.1.8	位运算符	33
4.1.9	逗号运算符	33
4.1.10	赋值运算符	33
4.1.11	运算符的优先级和结合性总结	35
4.1.12	算术运算符转换	36
4.2	表达式	37
4.3	显式类型转换（强制类型转换）	37
4.4	练习	38
第 5 章	语句	39
5.1	声明和定义	39
5.1.1	声明与定义的比较	40
5.1.2	常量标识符	40
5.1.3	变量	40
5.2	表达式语句	41
5.3	复合语句和块语句	42
5.4	条件执行语句	43
5.4.1	if 条件执行语句	43
5.4.2	switch 条件语句	44
5.5	循环	45
5.5.1	for 循环语句	45
5.5.2	while 语句	48
5.5.3	do-while 语句	49
5.6	跳转语句	50
5.6.1	break 语句	50
5.6.2	continue 语句	50
5.6.3	goto 语句	51
5.6.4	return 语句	51
5.7	空（NULL）语句	51
5.8	练习	52
第 6 章	函数	54
6.1	函数声明和函数原型	56
6.2	函数定义（函数体）	56
6.2.1	默认形参值	58

6.3	函数调用.....	59
6.4	参数传递.....	60
6.4.1	传值方式的参数传递.....	60
6.4.2	传址方式的参数传递.....	61
6.5	局部变量和全局变量.....	63
6.5.1	作用域与生命期.....	63
6.5.2	存储类.....	65
6.5.3	连接.....	65
6.5.4	存储类和连接举例.....	65
6.6	主函数 main().....	66
6.7	递归调用.....	67
6.8	单独编译.....	67
6.9	练习	68
第 7 章	数组、结构体和指针	70
7.1	数组	70
7.1.1	多维数组.....	72
7.1.2	初始化数组.....	76
7.1.3	数组参数.....	77
7.1.4	举例.....	77
7.2	结构体.....	81
7.2.1	联合体.....	83
7.2.2	结构体参数和联合体参数.....	85
7.3	指针	85
7.3.1	动态对象.....	86
7.3.2	void 指针.....	87
7.3.3	指针转换.....	87
7.3.4	指针参数.....	88
7.3.5	指针选取操作.....	88
7.3.6	指针运算.....	88
7.3.7	指针应用的一个例子：链表.....	89
7.4	数组和指针.....	91
7.4.1	数组参数.....	92
7.4.2	动态数组.....	93
7.5	字符串.....	94
7.5.1	举例.....	97
7.6	练习	98

第 8 章 类	100
8.1 复数数据类型	100
8.2 类的定义	105
8.3 类的对象	107
8.4 类的成员	107
8.4.1 特殊变量 this	107
8.4.2 构造函数	109
8.4.3 初始化	110
8.4.4 初始化：初始表达式与成员赋值	111
8.4.5 数组的初始化	113
8.4.6 构造函数调用的顺序	113
8.4.7 析构函数	113
8.4.8 析构函数调用的顺序	115
8.4.9 成员函数	115
8.4.10 成员运算符	115
8.5 堆栈：一个例子	116
8.5.1 stack 的说明	117
8.5.2 堆栈的实现	119
8.6 复制类的对象	120
8.6.1 赋值方式	120
8.6.2 初始化——拷贝构造函数	123
8.6.3 初始化与赋值	124
8.7 动态对象的创建和删除	126
8.7.1 运算符 new	127
8.7.2 删除操作符 DELETE	127
8.8 友元	128
8.8.1 对称接口的例子	128
8.8.2 友元的优点	130
8.9 常量成员与成员函数	130
8.10 执行类型转换	131
8.11 说明类型转换	132
8.11.1 构造函数	132
8.11.2 转换函数（运算符）	133
8.12 内联函数	135
8.13 提前（不完全）类的声明	135
8.14 链表：最后一个例子	136
8.15 练习	138

第 9 章 继承	139
9.1 派生类的说明	141
9.1.1 访问基类成员	142
9.1.2 派生类和基类的相互转换	142
9.1.3 派生类对象和基类对象之间的赋值	142
9.2 多重继承	142
9.3 构造函数/析构函数调用的顺序	143
9.4 派生类的赋值与派生类的构造函数	143
9.4.1 赋值运算符	144
9.4.2 默认（不带参数的）构造函数	144
9.4.3 拷贝构造函数	145
9.5 虚函数	145
9.6 例子	145
9.6.1 虚析构函数	145
9.6.2 员工数据	146
9.7 练习	150
第 10 章 重载	151
10.1 运算符重载	152
10.2 例子	155
10.3 练习	156
第 11 章 输入/输出	157
11.1 流头文件 <code>iostream.h</code>	157
11.2 输入	157
11.3 输出流	160
11.4 操纵符	162
11.5 流的状态	162
11.6 扩展流输入/输出库	163
11.6.1 <code>complex</code> 值的输入/输出	163
11.6.2 打印矢量	166
11.7 定义新流	167
11.8 附加功能	168
第 12 章 一个大型的例子	169
第 13 章 模板	176
13.1 模板的声明和定义	176
13.1.1 函数模板	176
13.1.2 类模板	177
13.2 例子	179

13.2.1	函数模板 SWAP	179
13.2.2	模板类 STACK	179
13.3	练习	181
第 14 章	可变参数函数	182
14.1	例子	183
14.2	练习	184
第 15 章	预处理器	185
15.1	宏定义	185
15.1.1	常量宏定义	185
15.1.2	字符串宏定义	186
15.2	文件包含	186
15.3	条件编译	187
15.3.1	基于常量表达式的值的条件编译	187
15.3.2	基于符号定义的条件编译	187
15.4	避免多重包含	188
第 16 章	库功能	190
16.1	使用 C++ 库	190
16.1.1	标准头文件	190
16.2	标准函数	191
16.2.1	字符处理函数	191
16.2.2	数学函数	192
16.2.3	非局部跳转函数	194
16.2.4	信号处理函数	195
16.2.5	输入/输出	195
16.2.6	通用工具	203
16.2.7	字符串操作	206

第 1 章 简介

编程就是编写一系列有序指令来控制计算机运行的过程，这些有序指令就是“程序”。计算机指令通常称为“语句”。执行特定任务的某些语句又可称为“声明”或“定义”，让计算机执行操作的其他语句称为“执行语句”。

通常为了方便，我们将执行某个特定任务的多行语句捆绑在一起，并赋予一个名称。许多编程语言都通过函数来实现这一捆绑功能。函数（也就是其所代表的指令）在一次简单的函数调用后得到执行。函数也可返回一个值，即执行结果。

程序很像是菜谱。只有当菜谱本身正确，而且我们又按照其要求来烹饪，做出来的才是可口的美食。同样，只有程序写得正确，计算机才能按照它的指令做我们期望它做的事情。

通常计算机只能理解比较低级的“机器代码”，机器代码指令是一系列 0、1 的组合，早期的计算机程序只能用这种机器代码指令，然而机器码不易阅读和理解，于是后来又有了相对人性化的“高级”编程语言，同时也有了将高级语言指令转换成机器代码指令的编译器（又名翻译器）。

编译器本身就是程序。早期的大多数编译器及现有的许多编译器都属于“批处理”类型，在整个程序写完后才“调用”它们来翻译程序。程序本身是在“编辑器”的环境下编写的，然后保存在“文件”中。编辑器并不知道编程语言的语法规则，也就不会标记程序中的错误。编译器在程序中标记错误。这些错误只能使用编辑器更正。

由编译器标记的错误类别称为“句法错误”或“编译时”错误，而由程序中的不恰当指令产生的错误称为“运行时错误”，“运行时错误”只有在程序运行时才能检测出来。修正编译时错误是日常工作，而修正运行时错误却很困难，因为这些错误通常不是显式的指令错误，其错误结果只有在程序的后续运行过程中才能体现出来。

新一代的编译器，如 Borland C++ 编译器，提供了集成的交互式环境，编译器在其中可以：

- 辅助程序员编写程序。
- 在程序员输入程序时检测语法错误。
- 提供“运行”程序的功能。
- 用“调试器”帮助定位运行时错误。

不管编译器是批处理式的还是交互式的，编译过程都可分为两个阶段：首先是编写程序代码，然后是翻译程序，即将程序代码转成计算机可识别的机器码。另一类翻译程序是“解释器”，在程序员看来，有了它计算机就可以理解诸如 C++ 等高级语言了。解释器在程序读入后立即执行，所以程序错误，尤其是运行时错误，在程序开发过程的初期就可以检测出来。然而，对于同一程序，用解释器执行要比用编译器执行速度慢。

1.1 用 C++编写程序

C++程序，尤其是比较大的程序，一般都是注重“数据对象”的面向对象程序。这种编程方式称为“面向对象编程”。“面向对象”程序的目的就是建立（初始化或创建）对象，并对它们进行操作。

学习编程的一个非常好的方法就是试着自己编写程序。本节将介绍两个简单的程序，解释它们要做什么，以及它们是如何做的。第一个程序主要是关于输出问题的，第二个程序则为了说明程序的3个重要功能：输入、运算、输出。示例程序所用的编程功能将在以后的章节中详细介绍。

1.1.1 打印输出

编写一个简单程序将以下几行字输出到显示屏幕上：

```
Hello Programmer!
```

```
Welcome to the World of C++!
```

输出这些行的 C++ 程序如下（文件 `hello.cpp`）：

```
#include <iostream.h>
int main()
{
    cout<<"Hello Programmer!"<<endl;
    cout<<"Welcome to the World of C++!"<<endl;
}
```

程序保存在 `hello.cpp` 文件中，我们称之为“源文件”。第一行是 C++ “预处理器”的 `include` 指令，它包含了使用输入输出功能所必需的所有“声明”。在 C++ 中，所有要用到的项都必须先行“定义”或“声明”。C++ 项的“定义”告诉编译器该项究竟指的是什么，应该如何使用它，并在合适的情况下，在程序使用该项时执行一些 C++ 指令。项的“声明”只是告知 C++ 编辑器如何使用该项。项声明被例行地使用，因为项定义可能在当前源文件的后面给出，或者可能在单独的源文件中给出。只包含声明的文件称为“头”文件。

C++ 编译器实际上包含“预处理器”和真正的编译器两个部分。正如其名，预处理器在编译前先对程序进行处理，而且只对以字符“#”开头的行进行处理。这个例子程序中只有一个预处理行，即 `#include` 指令。该指令执行的结果是将头文件 `iostream.h` 中的内容引入程序中。该头文件包含读入数据和写输出所用到的功能的声明。头文件两边的尖括号告诉 C++ 在“标准”位置查找指定的头文件。

`main` 函数的定义从第 2 行开始，其形式为：

```
int main()
{
    body
}
```

`main` 是函数名，每个程序必须包含 `main` 函数，而且它也是程序执行开始的地方。C++

需要 `main` 函数的详细说明来构造完整的程序。通常情况下，函数必须从程序内部显式地调用，而 `main` 函数却不是这样，它是由诸如 MS-DOS 或 UNIX 等操作系统“调用”来开始执行的。

第二行（函数定义的第一行）指出我们要定义一个名为 `main` 的“函数”，第 3 行的左大括号表明 `main` 函数体开始，第 6 行的右大括号表示函数体结束。第一行中对 `main` 函数定义包含一对圆括号（紧跟在 `main` 标识符后面），这对“空”括号表明在程序开始执行前，`main` 函数不接受任何数值或字符串值。这些值被称为“参数”，由函数的调用者提供（对于 `main` 函数，调用者是操作系统）。为了更加准确，我们还可以这样声明 `main` 函数：

```
int main(void)
{
    body
}
```

`void` 标识符在此处表明 `main` 函数不接受任何参数，该标识符为 C++ 中的关键字，故只能用在 C++ 中使用。由于 C++ 是源于 C 的历史原因，该标识符很容易被忽视。

实际输出结果由 `main` 函数函数体中以下两行指令的执行产生：

```
cout << "Hello Programmer!" << endl;
cout << "Welcome to the World of C++!" << endl;
```

每一行 C++ 代码都输出一行以换行字符结尾的文本串。注意，需要输出的文本内容要用引号括起来，`cout` 指的是显示屏幕。实际上更准确来讲，`cout` 是显示屏幕“输出流”的名称。输出写到“输出流”，输入从“输入流”中读取。“<<”是输出流运算符，就好像字符“+”是加法运算运算符一样。运算符“<<”将其右边的操作数（右边的值）发送到左操作数（左边的值）指定的流中，然后返回该“流”作为结果。

上面两行代码中的每一行都是一个 C++ 语句。除紧跟着大括号的执行语句以外，其他语句都必须以分号结尾，大括号中的语句称为复合语句。

上面两个输出指令也可以写成：

```
cout << "Hello Programmer! ";
cout << endl;
cout << "Welcome to the World of C++!";
cout << endl;
```

`endl` 是一个被称为“操纵符”函数的特殊函数的函数名。如上所述，`endl` 与“<<”搭配使用的作用是打印一个换行符。

与 BASIC 等编程语言不同，在 C++ 中，不能让编译器或解释器只简单地执行上面这样的输出语句（上面示例程序的第 4 行和第 5 行）产生预期结果。这些语句必须要封装在 `main` 程序中。

1. 文件命名习惯

包含 C++ 程序的文件称为源文件。许多 C++ 编译器都要求 C++ 源文件名必须有合适的后缀，如 `.c` 或 `.cpp`。如 UNIX 系统中的 C++ 编译器要求 C++ 源文件以 `.c` 为扩展名。MS-DOS 系统中的 C++ 编译器，如 Zortech 和 Borland C++ 编译器，则要求 C++ 源文件以 `.cpp` 为扩展名，这样编译器可以将其与 C 源文件（必须以 `.c` 为扩展名）区别开来。当然，C++ 源文件

也可以用.c作扩展名,但这样需要设置编译器的选项,让C++编译器知道.c扩展名的源文件中也包含有C++程序。为避免混淆,我们还是简单地用.cpp作为C++源文件的扩展名。

只包含声明的文件称为“头”文件。习惯上头文件以.h为扩展名。

2. 运行程序

运行程序时,屏幕上将打印需要输出的两行内容,但程序在运行前必须先进行编译。如果用的是Borland C++编译器¹,上面的程序可以编译为²:

```
bcc hello.cpp
```

或者

```
bcc hello
```

后一种情况中,C++编译器将会自动搜索名为hello.cpp的文件进行编译(翻译)。编译后产生两个文件:

- hello.obj: 翻译hello.cpp文件中的C++程序得到的机器代码。
- hello.exe: hello.obj的“可执行”版本,计算机可以执行它。hello.exe与hello.obj基本相同,但前者已经与所需的其他程序(存放在库中)进行“连接”结合,从而使程序可以执行。

输入以下内容程序开始执行:

```
hello.exe
```

或者仅输入:

```
hello
```

程序执行后将在屏幕上显示:

```
Hello Programmer!
```

```
Welcome to the World of C++!
```

在UNIX系统中,C++源文件通常以.c为扩展名。假设对应的C++编译器名为CC,则hello.c文件的编译和连接命令如下:

```
cc hello.c
```

然后系统会产生名为a.out的可执行文件,输入以下内容可以运行它:

```
a.out
```

当然,作为UNIX程序员的我们可以给上面的可执行文件起一个便于记忆的名字,而不是默认的a.out。例如,可以用-o参数指定可执行文件的文件名:

```
cc -ohello hello.c
```

执行得到的可执行程序,可以输入以下内容:

```
hello
```

1.1.2 计算二次方程的根

现在要编一个程序计算下面这种形式的二次方程的根:

$$y=ax^2+bx+c$$

¹ Turbo C编译器是Borland公司发布的其他编译器。

² 操作系统提示符不会显示出来。