

C++ YING YONG
BIAN CHENG 200 LI

C++应用 200 例 编程

赫阿朋 编著



电子工业出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

<http://www.phei.com.cn>

C++应用编程 200 例

赫阿朋 编著

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 提 要

本书是专为广大学习和使用 C++ 语言的读者设计的,旨在帮助读者更快更有效地掌握 C++ 语言。全书共分 4 部分,通过大量实例全面系统地介绍了 C++ 语言程序设计的基本概念和方法,主要内容包括:基本运算、数据类型、程序结构、函数、类与对象、预处理指令、文件处理、函数模板、类模板和 STL 的使用等。

书中给出的所有实例的代码完整,注释清楚,并附有运行结果和程序说明,同时也包含了许多编程技巧,而对相关概念和方法的说明简单明了。书中所有实例均在 Visual C++ 6.0 开发环境中调试通过。

本书是广大 C++ 语言学习者和程序设计者的优秀参考书。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

图书在版编目(CIP)数据

C++应用编程 200 例 / 赫阿朋编著. —北京:电子工业出版社, 2003.4

ISBN 7-5053-8581-X

I. C... II. 赫... III. C 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2003) 第 016908 号

责任编辑: 祁玉芹

印 刷: 北京市天竺颖华印刷厂

出版发行: 电子工业出版社出版 <http://www.phei.com.cn>

北京市海淀区万寿路 173 信箱 邮编 100036

经 销: 各地新华书店

开 本: 787×1092 1/16 印张: 29.5 字数: 699 千字

版 次: 2003 年 4 月第 1 版 2003 年 4 月第 1 次印刷

印 数: 6000 册 定价: 38.00 元

凡购买电子工业出版社的图书,如有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系。联系电话: (010)68279077

前 言

C++语言是在C语言基础上发展起来的面向对象的程序设计语言，并保持了C语言的特点。它既支持面向对象程序设计，又支持过程化设计，可用于开发各类系统软件和应用软件，因此深受各类程序设计者的喜爱。

本书是为学习和使用C++语言编程读者而设计的，旨在帮助读者快捷高效地掌握C++语言。本书在写法上考虑到不同读者的特点，在全书的整体设计上突出了C++语言的完整性、系统性和连续性，而在某个实例的设计上又考虑到其独立性。因此，读者既可以完整地阅读本书，也可以有选择地阅读书中的某些实例。

全书共分4部分，通过大量应用实例循序渐进地对C++语言程序设计的基本概念和方法进行了全面系统的介绍。

第一部分通过100个实例介绍了C++语言的基础部分，主要包括：基本数据类型、基本运算、基本语句、数组、指针、结构、函数、文件输入输出和预处理指令等。读者掌握好这些内容，可以为C++语言的进一步学习打下基础。

第二部分通过38个实例分别介绍了C++语言与面向对象设计有关的概念和方法，主要包括：类和对象的概念、定义类和创建对象、数据成员与成员函数的设计和使用、构造函数与析构函数的设计和使用、类成员访问性控制、静态数据与静态函数的设计和使用、重载的定义和使用、友元函数和友元类、类继承的概念、成员函数的覆盖、虚函数和抽象类等。这些内容是C++语言为面向对象设计而提供的编程机制，因此，读者要想用C++进行面向对象程序设计，就必须正确地理解和掌握这部分内容。

第三部分通过31个实例分类介绍了C++预定义函数的使用，这些函数是从C语言系统继承过来的，它们的声明都分类存放在C++系统的include子目录下的各个头文件中。熟悉了这些预定义函数的功能和用法，无论对C或C++程序设计都会有很大的帮助。

第四部分通过31个实例系统地介绍了有关C++模板定义和使用的知识，主要包括：函数模板的定义和使用，类模板的定义和使用，STL的容器、迭代器和算法的概念，STL主要容器模板的使用方法，STL一些常用算法的使用特点。这些内容是C++语言的高级部分，理解其概念，熟悉其功能和用法，对提高C++程序设计的效率会有极大帮助。

本书给出的所有实例的代码完整，注释清楚，并附有运行结果和程序说明，同时也包含了许多编程技巧，对相关概念和方法的说明简单明了。书中所有实例均在Visual C++ 6.0(书中简称VC++ 6.0)开发环境中调试通过。

本书写作的立足点是学习标准C++语言，尽管选择了流行的VC++ 6.0作为本书示例和练习的调试运行环境，但其不作为本书的重点。

参加本书编写工作的还有武兴庄、冯微、袁丽、秦绪起、林红、卫东、林静、郝曙光、

王志远、吕峰、李军、钱炜、孙成军及常明亮等同志。

由于编者水平所限，书中错误和疏漏之处在所难免，希望专家和读者朋友及时指正。

读者可从我们的网站下载书中所有实例的源代码，网址为：

<http://txbook.phei.com.cn>

也可给我们发E-mail，地址为：

qyqbook@sohu.com

作者

2003 年 1 月

目 录

第一部分 C++语言基础	1
实例 1 main()函数	2
实例 2 C++程序的两种注释方法	4
实例 3 常量和变量	6
实例 4 cout 输出	8
实例 5 cin 输入	9
实例 6 整型数据的处理	11
实例 7 八进制和十六进制数据的处理	13
实例 8 浮点型数据的处理	15
实例 9 字符型数据的处理	18
实例 10 转义字符及其使用	20
实例 11 布尔类型的使用	22
实例 12 const 变量的使用	23
实例 13 枚举类型的使用	24
实例 14 赋值运算	27
实例 15 自增自减运算	29
实例 16 关系运算	31
实例 17 逻辑运算	32
实例 18 位运算	34
实例 19 组合赋值运算	38
实例 20 sizeof 运算	40
实例 21 顺序运算	42
实例 22 数据类型的转换	43
实例 23 简单条件语句	46
实例 24 if...else 条件语句	47
实例 25 条件语句的嵌套使用	49
实例 26 带 else...if 的条件语句	51
实例 27 if 语句条件设置的几种方法	53
实例 28 条件运算	56
实例 29 短路	57
实例 30 switch 语句	59
实例 31 复合语句中的变量	61

实例 32	for 循环语句.....	62
实例 33	while 循环语句.....	66
实例 34	do-while 循环语句	68
实例 35	多重循环.....	71
实例 36	goto 语句.....	72
实例 37	break 语句.....	74
实例 38	continue 复合语句.....	75
实例 39	数组和下标变量.....	76
实例 40	用顺序法查找数组中的数据.....	80
实例 41	二分法查找数组元素.....	82
实例 42	选择排序法对数组排序.....	84
实例 43	多维数组的使用.....	85
实例 44	字符数组的使用.....	89
实例 45	指针变量与取地址和取内容运算.....	94
实例 46	指向数组的指针.....	99
实例 47	指针变量的运算.....	103
实例 48	字符型指针变量.....	108
实例 49	指针数组的使用.....	112
实例 50	指向指针的指针变量.....	114
实例 51	结构类型和结构变量.....	115
实例 52	结构数组.....	121
实例 53	结构指针变量.....	125
实例 54	new 和 delete 运算	128
实例 55	结构指针变量与结构数组.....	131
实例 56	包含指针的结构类型.....	133
实例 57	嵌套结构.....	135
实例 58	递归结构.....	137
实例 59	单向链表.....	138
实例 60	双向链表.....	140
实例 61	联合类型.....	143
实例 62	联合与结构的比较.....	147
实例 63	用 typedef 定义类型.....	149
实例 64	计算任意一维数组的大小.....	151
实例 65	函数的定义和调用.....	153
实例 66	return 语句在函数中的作用	158
实例 67	函数原型与定义.....	159
实例 68	函数值的传递方式.....	161
实例 69	函数地址的指针传递方式.....	163
实例 70	函数地址的引用传递方式.....	166

实例 71	带默认参数值的函数.....	169
实例 72	使用 <code>const</code> 修饰参数的函数.....	171
实例 73	<code>void</code> 类型的函数参数和返回值.....	172
实例 74	函数的重载.....	174
实例 75	内联函数.....	176
实例 76	递归函数的设计.....	177
实例 77	<code>main()</code> 函数参数和返回值的使用.....	178
实例 78	函数传递数组的方法.....	180
实例 79	函数传递结构的方法.....	183
实例 80	利用引用参数返回函数的多个值.....	185
实例 81	指向函数的指针.....	187
实例 82	用于结构输入输出的函数.....	189
实例 83	全局变量和局部变量.....	191
实例 84	静态变量.....	193
实例 85	函数的存储类别.....	195
实例 86	“流”文件的打开和关闭.....	196
实例 87	以字符方式读写“流”文件.....	200
实例 88	以字符串方式读写“流”文件.....	203
实例 89	以整数方式读写“流”文件.....	205
实例 90	以格式化方式读写“流”文件.....	207
实例 91	以数据块方式读写“流”文件.....	210
实例 92	“流”文件的读写定位.....	213
实例 93	数组的文件读写.....	217
实例 94	结构数据的文件读写.....	220
实例 95	预定义“流”文件指针.....	224
实例 96	“流”文件的错误处理.....	225
实例 97	文件嵌入预处理指令(<code>#include</code>).....	228
实例 98	简单宏替换预处理指令(<code>#define</code>).....	229
实例 99	带参数宏替换预处理指令(<code>#define</code>).....	230
实例 100	条件编译预处理指令(<code>#if</code>).....	232
第二部分 类和对象		235
实例 101	类的定义与对象的建立和使用.....	236
实例 102	构造函数.....	240
实例 103	带参数的构造函数.....	242
实例 104	构造函数的初始化列表.....	244
实例 105	构造函数的重载.....	246
实例 106	带默认值参数的构造函数.....	247
实例 107	析构函数.....	249

实例 108	在类外定义类函数.....	251
实例 109	public:访问模式.....	254
实例 110	private 访问模式.....	256
实例 111	类的静态数据成员.....	259
实例 112	类的静态成员函数.....	261
实例 113	内联成员函数.....	263
实例 114	空类.....	265
实例 115	类与结构.....	266
实例 116	类的组合.....	268
实例 117	成员函数的重载.....	270
实例 118	运算符的重载.....	271
实例 119	友元类.....	275
实例 120	友元函数.....	278
实例 121	插入(<<)和提取(>>)运算符的重载.....	281
实例 122	对象指针.....	283
实例 123	类的继承.....	285
实例 124	privatec 继承模式.....	288
实例 125	public 继承模式.....	290
实例 126	派生类构造函数的调用顺序.....	291
实例 127	派生类构造函数的参数传递.....	293
实例 128	派生类析构函数的调用顺序.....	295
实例 129	protected 成员的继承性.....	296
实例 130	多类继承.....	298
实例 131	多类继承中构造函数和析构函数的执行顺序.....	300
实例 132	覆盖与虚函数.....	303
实例 133	纯虚函数与抽象类.....	306
实例 134	cout 和 cin 对象的成员函数.....	308
实例 135	一个用类实现的双向链表.....	310
实例 136	string 对象的构造和输入输出.....	314
实例 137	string 对象的运算.....	316
实例 138	string 对象常用的成员函数.....	318

第三部分 预定义函数 321

实例 139	字符分类函数.....	322
实例 140	数字转换为字符串函数.....	323
实例 141	字符串转换为数字函数.....	325
实例 142	随机数函数.....	327
实例 143	三角函数.....	328
实例 144	反三角函数.....	330

实例 145	复数绝对值计算函数.....	331
实例 146	对数函数.....	332
实例 147	指数函数.....	333
实例 148	幂函数.....	334
实例 149	平方根函数.....	336
实例 150	获取系统时间函数.....	337
实例 151	系统时间格式化函数.....	338
实例 152	时间转换为字符串函数.....	340
实例 153	计算时间差函数.....	341
实例 154	内存分配和释放函数.....	342
实例 155	控制台输入输出函数.....	344
实例 156	格式化输出函数.....	347
实例 157	格式化输入函数.....	350
实例 158	目录管理函数.....	353
实例 159	获取文件信息函数.....	355
实例 160	字符串设置函数.....	357
实例 161	字符串拷贝函数.....	358
实例 162	字符串连接函数.....	360
实例 163	字符串比较函数.....	361
实例 164	字母大小写转换函数.....	364
实例 165	计算字符串长度函数.....	365
实例 166	字符串查找函数.....	366
实例 167	字符串反转函数.....	368
实例 168	字符串分离函数.....	369
实例 169	格式化输出到字符缓冲区函数.....	371

第四部分 函数模板、类模板和 STL..... 373

实例 170	函数模板的定义和使用.....	374
实例 171	支持多种数据类型的函数模板.....	376
实例 172	函数模板的引用参数.....	378
实例 173	函数模板的指针参数.....	380
实例 174	输入输出函数模板.....	383
实例 175	数组处理的函数模板.....	386
实例 176	二分查找法函数模板.....	388
实例 177	类模板的定义和使用.....	390
实例 178	栈处理的类模板.....	392
实例 179	支持多个不同数据类型的类模板.....	395
实例 180	STL	398
实例 181	序列容器 <code>vector</code>	402

实例 182	序列容器 <code>deque</code>	408
实例 183	序列容器 <code>list</code>	412
实例 184	关联容器 <code>set</code>	415
实例 185	关联容器 <code>multiset</code>	418
实例 186	关联容器 <code>map</code>	421
实例 187	关联容器 <code>multimap</code>	425
实例 188	专用容器 <code>valarray</code>	427
实例 189	专用容器 <code>stack</code>	430
实例 190	迭代器.....	432
实例 191	<code>accumulate()</code> 算法对容器元素的累加计算.....	434
实例 192	<code>count()</code> 算法统计容器元素的个数	437
实例 193	<code>count_if()</code> 算法对容器的统计.....	440
实例 194	<code>fill()</code> 和 <code>fill_n()</code> 算法对容器的填充	442
实例 195	<code>find()</code> 算法对容器的查询.....	444
实例 196	<code>find_end()</code> 算法对容器的查询.....	447
实例 197	<code>generate()</code> 和 <code>generate_n()</code> 算法生成容器元素	450
实例 198	<code>reverse()</code> 和 <code>reverse_copy()</code> 算法对容器的反转和拷贝处理.....	453
实例 199	<code>sort()</code> 和 <code>partial_sort()</code> 算法对容器的排序处理	455
实例 200	<code>max_element()</code> 和 <code>min_element()</code> 算法求容器元素的最大值和最小值	458



第一部分 C++语言基础

本部分通过 100 个实例介绍了 C++ 语言的基础部分，主要内容包括基本数据类型、基本运算、基本语句、数组、指针、结构、函数、文件输入输出和预处理指令等。这些是 C++ 语言的最基本内容，必须很好地掌握它们，以便为 C++ 语言的进一步学习打下基础。

 Start

实例 1 main()函数



语法要点

main()函数也叫主函数，是 C++ 程序中的一个最重要函数。所有完整可运行的 C++ 程序都必须有一个惟一的 **main()** 函数，它也是 C++ 程序中惟一可以直接运行的函数，其他函数都直接或间接地由其调用执行。C++ 程序的执行开始于 **main()** 函数，一个结构良好的 C++ 程序也应结束于该函数。

main()函数的使用要点如下。

(1) 对于一些很简单的问题，只用一个 **main()** 函数即可，程序的全部处理语句都放在其中。

(2) 对于一些复杂的问题，需要进行模块化设计，即把一个复杂问题分解成若干个相对简单一些的子问题。每个子问题由一个或多个函数来处理，而 **main()** 函数负责总控，调用相应的函数。

(3) **main()** 函数是一个特殊的函数，其中“**main**”是函数名。与其他函数一样，该函数也可以有返回值和参数表。在后面有专门的实例讨论 **main()** 函数的参数和返回值的使用问题。以下给出 **main()** 函数的最简单形式：

```
main()           //主函数名
{                //函数体起始符
    变量声明语句
    执行语句
    ...
}                //函数体终止符
```



实例程序

pl-1.cpp: 从键盘上输入任一圆的半径 **R**，计算其周长和面积，并在屏幕上显示计算结果。

```
//根据半径计算圆的周长和面积
#include <iostream.h>
const float PI=3.1416;           //声明常量(只读变量)PI 为 3.1416
float fCir_L(float);             //声明自定义函数 fCir_L() 的原型
float fCir_S(float);             //声明自定义函数 fCir_S() 的原型

//以下是main()函数
main()
{
    float r,l,s;                 //声明 3 个变量
```

```

    cout<<"R=";           //显示字符串
    cin>>r;                //键盘输入
    l=fCir_L(r);           //计算圆的周长,赋值给变量 l
    s=fCir_S(r);           //计算圆的面积,赋值给变量 s
    cout<<"l="<<l;        //显示计算结果
    cout<<"\ns="<<s;
}

//定义计算圆的周长的函数 fCir_L()
float fCir_L(float x)
{
    float z=-1.0;          //声明局部变量
    if (x>=0.0)             //如果参数大于 0, 则计算圆的周长
        z=2*PI*x;
    return(z);              //返回函数值
}

//定义计算圆的面积的函数 fCir_S()
float fCir_S(float x)
{
    float z=-1.0;          //声明局部变量
    if (x>=0.0)             //如果参数大于 0, 则计算圆的面积
        z=PI*x*x;
    return(z);              //返回函数值
}

```

运行结果

```

R=1.5↵
l=9.4248
s=7.0686

```

程序说明

- (1) 本例定义和使用了 main()、fCir_L()和 fCir_S()共 3 个函数。
- (2) 程序运行结果中的“↵”符号表示从键盘输入数据后的回车符。本书一律用“↵”符号表示回车键。
- (3) C++程序中的每个语句都以分号“;”结束。
- (4) 在函数的定义中使用了花括号“{”和“}”，花括号作为函数体的开始和结束标记。使用花括号也是 C++的特点之一，但一定要配对使用。
- (5) main()函数是主函数，其作用是声明变量、接收键盘输入、调用函数计算圆的周长和面积，最后输出结果。
- (6) main()函数中的 cout 是 C++语言系统提供的用于控制显示输出的对象，通过它可以输出各种类型的数据。后面的实例还要专门介绍 cout 对象的使用。
- (7) main()函数中的 cin 是 C++语言系统提供的用于控制数据输入的对象，通过它可以输入各种类型的数据。后面还要专门介绍 cin 对象的使用。

(8) fCir_L()和 fCir_S()函数的作用是按指定半径计算圆的周长和面积,圆的半径由调用函数时的参数给定。例如, s=fCir_S(1.5)指定计算半径为 1.5 的圆的面积,而 s=fCir_S(r)指定计算半径为 r 值的圆的面积。

(9) 在本例中, fCir_L()和 fCir_S()函数的定义放在了 main()函数之后,因此要求在 mian()函数之前要有这两个函数的原型声明。

(10) 程序中以“//”开始的部分为注释。

(11) 程序最开始的一行,即#include <iostream.h>,为 C++的预处理指令。其作用是将头文件 iostream.h 中的内容插入进来,该文件包含了键盘输入和显示输出的有关信息。#include 称为预处理指令,是 C++系统提供的一组指导编译的指令之一。

实例 2 C++程序的两种注释方法



语法要点

注释程序的文字说明,可增强程序的可读性,是程序的重要组成部分。程序中的注释在编译时不产生代码,编译器将其作为“空白符”处理,因此注释不会影响程序的运行。

一般在程序或函数的开始处用注释说明程序或函数的功能,在程序中用注释标注需要说明的问题。

C++语言的注释有以下两种方式。

(1) 沿用 C 语言的注释形式,即由“/*”和“*/”括起的一系列字符组成,其格式如下:

```
/*注释内容*/
```

使用时应注意以下几点。

- 这种注释可以出现在语句之前、之后或之中。

例如,如下注释在语法上都是正确的:

```
s=1.5;    /* 对变量 s 赋值*/  
/* 对变量 s 赋值*/    s=1.5;  
s= /* 对变量 s 赋值*/ 1.5;
```

- 包含在“/*”和“*/”之间的注释内容可以是任何字符包括英文或汉字。

例如,如下注释:

```
cout<<"Hello";    /* 显示字符串 Hello */
```

- 当注释信息为多行时,首行以“/*”开头,尾行以“*/”结束。

例如,以下为多行注释的例子:

```
/* FEOF.C: This program uses feof to indicate when  
it reaches the end of the file FEOF.CPP. It also  
checks for errors with ferror.
```

*/

- “/*...*/”形式的注释不能嵌套。

例如，以下形式的注释是错误的：

```
/* Comment out this routine for testing
   /* Open file */
   fh = _open( "myfile.c", _O_RDONLY );
   ...
*/
```

(2) 为了快速地进行注释，C++新增了一种单行注释方法，即用双斜线“//”开始，直到本行结束，其格式为：

//注释内容

使用时应注意以下几点。

- 这种注释只在一行内有效。

例如，如下注释是合法的：

```
// 以下声明变量
int n;                //声明整型变量
unsigned int s;       //声明无符号整型变量
...
```

- 在执行语句前或执行语句中不能使用这样的注释。

例如，如下注释将产生不同的结果：

```
s=1.5;    // 对变量 s 赋值
// 对变量 s 赋值    s=1.5;
```



实例程序

p1-2.cpp: 测试 C++程序的两种注释方法。

```
/* Program: P1-2.CPP
   Written by: Hap
   Date written: 02:11:10
*/
#include <iostream.h>
void main(void)
{
    double s1,s2,s3;
    s1=1.5;    /* 对变量 s1 赋值*/
    cout<<"s1="<<s1<<endl;
    /* 对变量 s2 赋值*/    s2=2.5;
    cout<<"s2="<<s2<<endl;
    s3=    /* 对变量 s3 赋值*/    3.5;
    cout<<"s3="<<s3<<endl;
```

```
cout<<"s1+s2+s3="<<s1+s2+s3<<endl;    //计算并显示
//计算并显示: cout<<"s1+s2+s3="<<s1+s2+s3<<endl;
}
```

运行结果

```
s1=1.5
s2=2.5
s3=3.5
s1+s2+s3=7.5
```

程序说明

- (1) 在程序的开头是`/* */`形式的多行注释。
- (2) 在`main()`中,使用了C++的两种注释方法对语句进行注释。
- (3) 最后一行没有输出结果是因为输出语句也成为注释的一部分。

实例 3 常量和变量

语法要点

程序中处理的数据从形式上可分为常量和变量两大类。

(1) 常量是指在程序中固定不变的数据量,在C++程序中可作为常量的数据有整型数、浮点型数、单个字符和字符串等。例如,下面是一些常量的例子。

- 整型常量: 10, 0, -100, 32 767。
- 浮点类型(实型)常量: 3.141 6, 1.414, 1.7E4。
- 字符常量: 'A', '\$', 's'。
- 字符串常量: "This is My book.", "How are you."。

(2) 变量是内存存储单元名。存储单元类似旅馆的房间,而变量相当于房间号。在程序运行过程中,变量中的数据可以改变。程序中的变量与数学中的变量类似,用标识符表示。

C++语言规定,变量必须先声明后使用。在声明变量时要指定变量名和变量的数据类型,也可以同时为声明的变量赋初值。变量名是程序引用的依据,而变量的数据类型则决定了编译系统为其分配的空间大小,以及它所能保存哪种类型的数据。例如,以下变量声明语句:

```
int m,n=100;    //声明整型变量m和n,并为n赋初值
float x;        //声明浮点型变量x
char c;         //声明字符型变量c
```

Visual C++ 编译程序为`int`类型变量`m`和`n`分配的空间为4个字节(32位),为`float`类型变量`x`分配的空间也为4个字节(32位),为`char`类型变量`c`分配的存储空间为1个字节