

《C++入门经典(第4版)》

图书基本信息

书名：《C++入门经典(第4版)》

13位ISBN编号：9787302406286

出版时间：2015-9-1

作者：(美)霍尔顿(Horton,I.)

页数：608

译者：石磊

版权说明：本站所提供下载的PDF图书仅提供预览和简介以及在线试读，请支持正版图书。

更多资源请访问：www.tushu000.com

《C++入门经典(第4版)》

内容概要

《C++入门经典(第4版)》是一本C++初学者指南，讨论了适合初学者的C++功能子集，其语法对应于C++ 14标准。本书的内容适合于任何开发环境，可以在任何操作系统或程序开发系统中使用。读者不需要有任何编程知识。

本书中的所有语言特性都用具体的例子来说明，通过每章最后的练习还可以测试自己对所学知识的掌握情况。文中的示例和习题答案都可以从网上下载，学完本书后，还可以试着完成一个可下载的、更实际的项目。

本书介绍了C++标准库的元素，标准库提供对本书讨论的语法的基本支持。标准模板库(STL)讨论得不多，只介绍并应用了标准模板库中几个对现代C++理念非常重要的元素。

主要内容

- 使用C++基本数据类型进行计算
- 使用循环、选择、决策等建立程序的逻辑
- 使用数组、矢量和字符串
- 使用原指针和智能指针
- 使用函数编程，处理程序文件和预处理指令
- 使用类和类操作定义自己的数据类型
- 使用异常警示并处理错误
- 定义、使用函数模板和类模板
- 用C++处理文件输入输出

《C++入门经典(第4版)》

作者简介

Ivor Horton是世界著名计算机图书作家，独立顾问，帮助无数程序员步入编程殿堂。他曾在IBM工作多年，以优异成绩拥有数学学士学位。他的资历包括：使用大多数语言(如在多种机器上使用汇编语言和高级语言)进行编程，实时编程，设计和实现实时闭环工业控制系统。Horton拥有丰富的面向工程师和科学家的编程教学经验(教学内容包括C、C++、Fortran、PL/1、APL等)。同时，他还是机械、加工和电子CAD系统、机械CAM系统和DNC/CNC系统方面的专家。

书籍目录

第1章 基本概念

1

1.1 现代C++

1

1.2 C++程序概念

2

1.2.1 注释和空白

2

1.2.2 预处理指令和头文件

3

1.2.3 函数

3

1.2.4 语句

4

1.2.5 数据输入输出

4

1.2.6 return语句

5

1.2.7 名称空间

5

1.2.8 名称和关键字

6

1.3 类和对象

6

1.4 模板

7

1.5 程序文件

7

1.6 标准库

7

1.7 代码的表示样式

7

1.8 创建可执行文件

8

1.9 表示数字

9

1.9.1 二进制数

9

1.9.2 十六进制数

11

1.9.3 负的二进制数

12

1.9.4 八进制数

14

1.9.5 Big-Endian和Little-Endian系统

14

1.9.6 浮点数

15	
1.10	表示字符
16	
1.10.1	ASCII 码
16	
1.10.2	UCS和Unicode
17	
1.11	C++源字符
17	
1.11.1	三字符序列
18	
1.11.2	转义序列
18	
1.12	过程化编程方法和面向 对象编程方法
20	
1.13	本章小结
21	
1.14	练习
22	
第2章	基本数据类型
23	
2.1	变量、数据和数据类型
23	
2.1.1	定义整型变量
24	
2.1.2	定义有固定值的变量
26	
2.2	整型字面量
26	
2.2.1	十进制整型字面量
27	
2.2.2	十六进制的整型字面量
27	
2.2.3	八进制的整型字面量
27	
2.2.4	二进制的整型字面量
28	
2.3	整数的计算
28	
2.4	op=赋值运算符
33	
2.5	using声明和指令
34	
2.6	sizeof运算符
34	
2.7	整数的递增和递减
35	
2.8	定义浮点变量

37	
2.8.1	浮点字面量
38	
2.8.2	浮点数的计算
38	
2.8.3	缺点
38	
2.8.4	无效的浮点结果
39	
2.9	数值函数
40	
2.10	流输出的格式化
43	
2.11	混合的表达式和类型转换
45	
2.11.1	显式类型转换
46	
2.11.2	老式的强制转换
48	
2.12	确定数值的上下限
49	
2.13	使用字符变量
50	
2.13.1	使用Unicode字符
51	
2.13.2	auto关键字
52	
2.13.3	lvalue和rvalue
52	
2.14	本章小结
53	
2.15	练习
54	
第3章	处理基本数据类型
55	
3.1	运算符的优先级和相关性
55	
3.2	按位运算符
57	
3.2.1	移位运算符
58	
3.2.2	使用按位与运算符
60	
3.2.3	使用按位或运算符
61	
3.2.4	使用按位异或运算符
63	
3.3	枚举数据类型
67	

3.4 数据类型的同义词

70

3.5 变量的生存期

70

3.5.1 定位变量的定义

71

3.5.2 全局变量

71

3.5.3 静态变量

74

3.5.4 外部变量

75

3.6 本章小结

75

3.7 练习

76

第4章 决策

77

4.1 比较数据值

77

4.1.1 应用比较运算符

78

4.1.2 比较浮点数值

79

4.2 if语句

80

4.2.1 嵌套的if语句

82

4.2.2 不依赖编码的字符处理

84

4.3 if-else语句

85

4.3.1 嵌套的if-else语句

87

4.3.2 理解嵌套的if语句

88

4.4 逻辑运算符

89

4.4.1 逻辑与运算符

90

4.4.2 逻辑或运算符

90

4.4.3 逻辑非运算符

91

4.5 条件运算符

92

4.6 switch语句

94

4.7 无条件分支

98	
4.8 语句块和变量作用域	
99	
4.9 本章小结	
100	
4.10 练习	
100	
第5章 数组和循环	
103	
5.1 数据数组	
103	
5.2 理解循环	
105	
5.3 for循环	
106	
5.3.1 避免幻数	
107	
5.3.2 用初始化列表定义数组的大小	
109	
5.3.3 确定数组的大小	
109	
5.3.4 用浮点数值控制for循环	
110	
5.3.5 使用更复杂的循环控制表达式	
112	
5.3.6 逗号运算符	
113	
5.3.7 基于区域的for循环	
114	
5.4 while循环	
115	
5.5 do-while循环	
119	
5.6 嵌套的循环	
120	
5.7 跳过循环迭代	
123	
5.8 循环的中断	
125	
5.9 字符数组	
128	
5.10 多维数组	
131	
5.10.1 初始化多维数组	
134	
5.10.2 在默认情况下设置维数	
135	
5.10.3 多维字符数组	
136	

5.11 数组的替代品	137
5.11.1 使用array<T,N>容器	138
5.11.2 使用std::vector<T>容器	142
5.11.3 矢量的容量和大小	143
5.11.4 删除矢量容器中的元素	145
5.12 本章小结	145
5.13 练习	146
第6章 指针和引用	149
6.1 什么是指针	149
6.1.1 地址运算符	151
6.1.2 间接运算符	152
6.1.3 为什么使用指针	153
6.2 char类型的指针	154
6.3 常量指针和指向常量的指针	158
6.4 指针和数组	159
6.4.1 指针的算术运算	160
6.4.2 计算两个指针之间的差	162
6.4.3 使用数组名的指针表示法	162
6.5 动态内存分配	165
6.5.1 栈和堆	165
6.5.2 运算符new和delete	166
6.5.3 数组的动态内存分配	167
6.5.4 通过指针选择成员	169
6.6 动态内存分配的危险	169
6.6.1 内存泄漏	

169	
6.6.2 自由存储区的碎片	170
6.7 原指针和智能指针	170
6.7.1 使用unique_ptr<T>指针	172
6.7.2 使用shared_ptr<T>指针	173
6.7.3 比较shared_ptr<T>对象	177
6.7.4 weak_ptr<T>指针	177
6.8 理解引用	178
6.8.1 定义左值引用	179
6.8.2 在基于区域的for循环中使用引用变量	180
6.8.3 定义右值引用	180
6.9 本章小结	181
6.10 练习	181
第7章 操作字符串	183
7.1 更好的string类型	183
7.1.1 定义string对象	184
7.1.2 string对象的操作	186
7.1.3 访问字符串中的字符	188
7.1.4 访问子字符串	190
7.1.5 比较字符串	191
7.1.6 搜索字符串	196
7.1.7 修改字符串	203
7.2 国际字符串	207
7.3 包含Unicode字符串的对象	208
7.4 原字符串字面量	208

7.5 本章小结	209
7.6 练习	210
第8章 定义函数	211
8.1 程序的分解	211
8.1.1 类中的函数	212
8.1.2 函数的特征	212
8.2 定义函数	212
8.2.1 函数体	213
8.2.2 函数声明	215
8.3 给函数传送参数	217
8.3.1 按值传送机制	217
8.3.2 按引用传送	223
8.3.3 main()的参数	227
8.4 默认的参数值	228
8.5 从函数中返回值	231
8.5.1 返回指针	231
8.5.2 返回引用	235
8.6 内联函数	236
8.7 静态变量	237
8.8 函数的重载	239
8.8.1 重载和指针参数	241
8.8.2 重载和引用参数	241
8.8.3 重载和const参数	243
8.8.4 重载和默认参数值	244
8.9 函数模板	

245	
8.9.1	创建函数模板的实例
246	
8.9.2	显式指定模板参数
247	
8.9.3	函数模板的特例
248	
8.9.4	函数模板和重载
249	
8.9.5	带有多个参数的函数模板
250	
8.9.6	非类型的模板参数
251	
8.10	拖尾返回类型
252	
8.11	函数指针
253	
8.12	递归
256	
8.12.1	应用递归
259	
8.12.2	Quicksort算法
259	
8.12.3	main()函数
260	
8.12.4	extract_words()函数
261	
8.12.5	swap()函数
262	
8.12.6	sort()函数
262	
8.12.7	max_word_length()函数
263	
8.12.8	show_words()函数
264	
8.13	本章小结
265	
8.14	练习
266	
第9章	lambda表达式
269	
9.1	lambda表达式简介
269	
9.2	定义lambda表达式
269	
9.3	lambda表达式的命名
270	
9.4	把lambda表达式传递给函数
272	

9.4.1 接受lambda表达式变元的函数模板

272

9.4.2 lambda变元的函数参数类型

273

9.4.3 使用std::function模板类型

274

9.5 捕获子句

277

9.6 在模板中使用lambda表达式

279

9.7 lambda表达式中的递归

281

9.8 本章小结

283

9.9 练习

283

第10章 程序文件和预处理指令

285

10.1 理解转换单元

285

10.1.1 “一个定义”规则

286

10.1.2 程序文件和链接

286

10.1.3 确定名称的链接属性

286

10.1.4 外部名称

287

10.1.5 具有外部链接属性的

const变量

287

10.2 预处理源代码

288

10.3 定义预处理标识符

289

10.4 包含头文件

290

10.5 名称空间

292

10.5.1 全局名称空间

293

10.5.2 定义名称空间

293

10.5.3 应用using声明

296

10.5.4 函数和名称空间

296

10.5.5 未命名的名称空间

299

10.5.6 名称空间的别名	299
10.5.7 嵌套的名称空间	300
10.6 逻辑预处理指令	301
10.6.1 逻辑#if指令	301
10.6.2 测试指定标识符的值	302
10.6.3 多个代码选择	302
10.6.4 标准的预处理宏	303
10.7 调试方法	304
10.7.1 集成调试器	304
10.7.2 调试中的预处理指令	305
10.7.3 使用assert宏	309
10.7.4 关闭断言机制	310
10.8 静态断言	310
10.9 本章小结	312
10.10 练习	313
第11章 定义自己的数据类型	315
11.1 类和面向对象编程	315
11.1.1 封装	316
11.1.2 继承	318
11.1.3 多态性	318
11.1.4 术语	319
11.2 定义类	320
11.3 构造函数	322
11.3.1 在类的外部定义构造函数	324
11.3.2 默认构造函数的参数值	

326	
11.3.3	在构造函数中使用初始化列表
326	
11.3.4	使用explicit关键字
327	
11.3.5	委托构造函数
329	
11.3.6	默认的副本构造函数
331	
11.4	访问私有类成员
332	
11.5	友元
333	
11.5.1	类的友元函数
334	
11.5.2	友元类
336	
11.6	this指针
337	
11.7	const对象和const函数成员
338	
11.8	类的对象数组
340	
11.9	类对象的大小
342	
11.10	类的静态成员
342	
11.10.1	静态数据成员
342	
11.10.2	类的静态函数成员
347	
11.11	析构函数
347	
11.12	类对象的指针和引用
350	
11.13	将指针作为类的成员
351	
11.13.1	定义Package类
353	
11.13.2	定义TruckLoad类
354	
11.13.3	实现TruckLoad类
355	
11.14	嵌套类
360	
11.15	本章小结
363	
11.16	练习
363	

第12章 运算符重载

365

12.1 为类实现运算符

365

12.1.1 运算符重载

366

12.1.2 可以重载的运算符

366

12.1.3 实现重载运算符

366

12.1.4 全局运算符函数

369

12.1.5 提供对运算符的全部支持

369

12.1.6 在类中实现所有的比较运算符

371

12.2 运算符函数术语

373

12.3 默认类成员

374

12.3.1 定义析构函数

375

12.3.2 何时定义副本构造函数

377

12.3.3 实现赋值运算符

377

12.3.4 实现移动操作

379

12.4 重载算术运算符

380

12.4.1 改进输出操作

384

12.4.2 根据一个运算符实现另一个运算符

386

12.5 重载下标运算符

387

12.6 重载类型转换

394

12.7 重载递增和递减运算符

395

12.8 函数对象

396

12.9 本章小结

397

12.10 练习

398

第13章 继承

399

13.1 类和面向对象编程

399	
13.2 类的继承	
401	
13.2.1 继承和聚合	
401	
13.2.2 派生类	
402	
13.3 把类的成员声明为protected	
405	
13.4 派生类成员的访问级别	
405	
13.4.1 在类层次结构中使用访问指定符	
406	
13.4.2 改变继承成员的访问指定符	
408	
13.5 派生类中的构造函数操作	
408	
13.5.1 派生类中的副本构造函数	
412	
13.5.2 派生类中的默认构造函数	
414	
13.5.3 继承构造函数	
414	
13.6 继承中的析构函数	
415	
13.7 重复的成员名	
417	
13.8 重复的函数成员名	
418	
13.9 多重继承	
419	
13.9.1 多个基类	
419	
13.9.2 继承成员的模糊性	
420	
13.9.3 重复的继承	
424	
13.9.4 虚基类	
425	
13.10 在相关的类类型之间转换	
425	
13.11 本章小结	
426	
13.12 练习	
426	
第14章 多态性	
429	
14.1 理解多态性	
429	

14.1.1 使用基类指针	429
14.1.2 调用继承的函数	431
14.1.3 虚函数	434
14.1.4 虚函数中的默认参数值	442
14.1.5 通过智能指针调用虚函数	443
14.1.6 通过引用调用虚函数	444
14.1.7 调用虚函数的基类版本	445
14.1.8 在指针和类对象之间转换	446
14.1.9 动态强制转换	447
14.1.10 转换引用	449
14.1.11 确定多态类型	449
14.2 多态性的成本	450
14.3 纯虚函数	451
14.3.1 抽象类	452
14.3.2 间接的抽象基类	454
14.4 通过指针释放对象	457
14.5 本章小结	458
14.6 练习	459
第15章 运行时错误和异常	461
15.1 处理错误	461
15.2 理解异常	462
15.2.1 抛出异常	463
15.2.2 异常处理过程	465
15.2.3 未处理的异常	466
15.2.4 导致抛出异常的代码	

467	
15.2.5	嵌套的try块
468	
15.3	用类对象作为异常
472	
15.3.1	匹配Catch处理程序和异常
473	
15.3.2	用基类处理程序捕获派生类异常
476	
15.3.3	重新抛出异常
478	
15.3.4	捕获所有的异常
481	
15.4	抛出异常的函数
482	
15.4.1	函数try块
483	
15.4.2	不抛出异常的函数
483	
15.4.3	构造函数try块
484	
15.4.4	异常和析构函数
484	
15.5	标准库异常
485	
15.5.1	异常类的定义
486	
15.5.2	使用标准异常
487	
15.6	本章小结
490	
15.7	练习
491	
第16章	类模板
493	
16.1	理解类模板
493	
16.2	定义类模板
494	
16.2.1	模板参数
495	
16.2.2	简单的类模板
495	
16.2.3	定义类模板的函数成员
497	
16.3	创建类模板的实例
501	
16.4	类模板的静态成员
506	

16.5 非类型的类模板参数	507
16.5.1 带有非类型参数的函数成员的模板	510
16.5.2 非类型参数的变元	514
16.5.3 把指针和数组用作非类型参数	514
16.6 模板参数的默认值	515
16.7 模板的显式实例化	516
16.8 特殊情形	516
16.8.1 在类模板中使用 static_assert()	517
16.8.2 定义类模板特化	518
16.8.3 部分模板特化	519
16.8.4 从多个部分特化中选择	519
16.9 类模板的友元	520
16.10 带有嵌套类的类模板	521
16.11 本章小结	528
16.12 练习	528
第17章 文件输入与输出	531
17.1 C++中的输入输出	531
17.1.1 理解流	531
17.1.2 使用流的优点	533
17.2 流类	534
17.2.1 标准流对象	535
17.2.2 流的插入和提取操作	535
17.2.3 流操纵程序	537
17.3 文件流	540

17.3.1 在文本模式下写入文件	540
17.3.2 在文本模式下读取文件	543
17.4 设置流打开模式	546
17.5 未格式化的流操作	554
17.5.1 未格式化的流输入函数	554
17.5.2 未格式化的流输出函数	557
17.6 流输入输出中的错误	557
17.7 二进制模式中的流操作	559
17.8 文件的读写操作	570
17.9 字符串流	577
17.10 对象和流	578
17.10.1 给对象使用插入运算符	579
17.10.2 给对象使用提取运算符	579
17.10.3 二进制模式中的对象I/O	582
17.10.4 流中更复杂的对象	585
17.11 本章小结	590
17.12 练习	590

1、欢迎使用《C++入门经典(第4版)》。本书修订并更新了上一版（Beginning ANSI C++）。自上一版出版以来，C++语言有了很大的扩展和改进，但不太可能把C++的所有内容压缩到一本书中。本书提供的C++语言基础知识和标准库功能，足以让读者编写自己的C++应用程序。掌握了本书介绍的知识，读者应能毫无困难地扩展C++专业知识的深度和广度。C++要比许多人想象的更容易理解。本书不需要读者具备任何编程知识。如果你非常渴望学习，并具备逻辑思考的能力，掌握C++就会比想象的更容易。开发C++技巧，学习数百万人已在使用的语言，掌握C++技能，它提供了在几乎任何环境下开发应用程序的能力。本书的C++语言对应最新的ISO标准，一般称为C++ 14。C++ 14对以前的标准C++ 11进行了较小的扩展，所以本书的内容大都不专用于C++ 14。本书的所有示例都可以使用目前遵循C++ 11的编译器来编译和执行。使用本书要通过本书学习C++，需要一个遵循C++ 11标准的编译器和一个适合编写程序代码的文本编辑器。目前，有几个编译器兼容C++ 11，其中一些是免费的。GNU Project生产的GCC编译器全面支持C++ 11，是一个开源产品，可免费下载。安装GCC并将它与合适的编辑器一起使用，对新手而言略有难度。安装GCC和合适编译器的一种简单方法是，从<http://www.codeblocks.org>上下载Code::Blocks。Code::Blocks是Linux、Apple Mac OS X和Microsoft Windows的一个免费IDE，它允许使用几个编译器（包括用于GCC、Clang和open Watcom的编译器）开发程序。这表示，安装Code::Blocks会获得C、C++和Fortran的支持。另一种方法是使用在Microsoft Windows下运行的Microsoft Visual C++，它不仅完全兼容C++ 11，而且已经安装好了。其免费版本是Microsoft Visual Studio 2013 Express。编写本书时，它可以编译本书的大多数示例，最终应能编译所有示例。Microsoft Visual C++可以从<http://www.microsoft.com/en-us/download/details.aspx?id=43733>上下载。与GCC相比，Microsoft Visual C++编译器的限制多一些，但根据它对C++ 11的支持力度，这是一个专业的编译器，还支持其他语言，例如C#和Basic。当然，也可以安装这两个编辑器。还有其他支持C++ 11的编辑器，在网上搜索会很快找到它们。本书的内容循序渐进，所以读者应从头开始一直阅读到最后。但是，没有人能仅从一本书中获得所有的编程技巧。本书仅介绍如何使用C++编程，读者应自己输入所有的例子，而不是从下载文件中复制它们，再编译和执行输入的代码，这似乎很麻烦，但输入C++语句可以帮助理解C++，特别是觉得某些地方很难掌握时，自己输入代码就显得非常有帮助。如果例子不工作，不要直接从书中查找原因，而应在自己输入的例子代码中找原因，这是编写C++代码时必须做的一个工作。犯错误也是学习过程中不可避免的，练习应提供大量犯错误的机会，最好自己编几个练习题。如果不确定如何编写代码，应翻看前面的内容。犯的错误越多，对C++的功能和错误的原因认识得就越深刻。读者应完成所有的练习，记住不要看答案，直到肯定不能自己解决问题为止。许多练习都涉及某章内容的一个直接应用，换言之，它们仅是一种实践，但也有一些练习需要多动脑子，甚至需要一点灵感。希望每个人都能成功驾驭C++。—Ivor Horton

《C++入门经典(第4版)》

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:www.tushu000.com