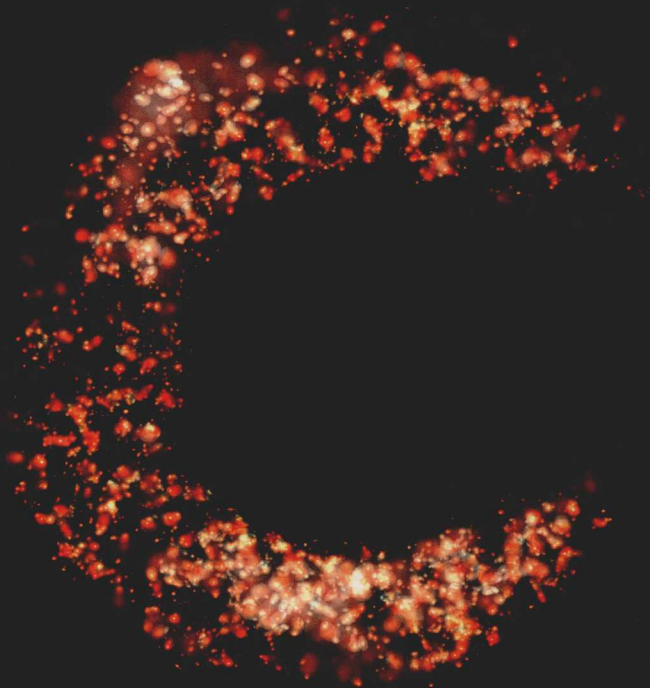




C++ 语法详解

本书全方位阐述了C++语法知识点,语法示例短小精悍,案头必备,方便速查
细致的图解分析,让读者更易理解语法原理,确保读者知其然更知其所以然

黄勇 / 编著



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

C++ 语法详解

黄勇 / 编著

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书对C++的语法进行了全面介绍和深入讲解,内容包括:C++整型、字符型、浮点型、声明、定义、typedef、运算符、表达式、左值、选择语句、循环语句、指针、数组、函数和标识符的作用域、类基础、类作用域及相关运算符、构造函数、复制构造函数、析构函数、名称空间、类中的成员、运算符(操作符)重载、继承、虚函数、多态性、对象模型、虚函数表、模板、I/O、异常、预处理器、typeid、强制类型转换和string类等。本书层次分明,由浅入深,各章节相对独立,语法示例短小精悍,方便对有疑惑的语法进行速查。学习完本书,读者不会再对C++的各种语法感到困惑。

本书适合有一定C++基础、对C++的语法有疑惑、想深入了解C++语法细节的人员阅读。本书同时也可以作为解决C++语法问题的参考书;对于学习过C++或已精通C++的人员,也是一本不错的资料查阅手册。

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++语法详解 / 黄勇编著. —北京:电子工业出版社, 2017.7
ISBN 978-7-121-31655-5

I. ①C… II. ①黄… III. ①C 语言—程序设计 IV. ①TP312.8

中国版本图书馆 CIP 数据核字(2017)第 119321 号

策划编辑: 安娜

责任编辑: 葛娜

印刷: 三河市良远印务有限公司

装订: 三河市良远印务有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编: 100036

开本: 787×980 1/16 印张: 31.25 字数: 697 千字

版次: 2017 年 7 月第 1 版

印次: 2017 年 7 月第 1 次印刷

定价: 89.00 元

凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888, 88258888。

质量投诉请发邮件至 zltz@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式: 010-51260888-819, faq@phei.com.cn。

投稿联络：安娜

微信&QQ：80303489

邮箱：anna@phei.com.cn



前言

本书具有如下特点：

(1) 本书是专门讲解 C++语法规则的书籍，因此书中不会介绍任何有关程序设计的内容（比如编写一个计算规形面积的程序等）。本书将语法问题分离出来，避免既不像写语法的书，也不像写程序设计的书。

(2) 书中的示例程序都使用简短的名字，比如 a, b, A 等，以便于记忆，而不会使用很长的名字。

(3) 一个知识点能用一段话讲解清楚的，尽量不使用两段话。一个知识点一个标号，方便查阅和增补。

(4) 一个知识点列举一个单独的简短易懂的程序作为示例。大多数教材都喜欢在第 1 章开头定义一个变量，然后一直到章尾都在使用那个变量作为示例。本书打破传统，一个知识点就是一个单独的示例，不与上一个知识点的示例拉上关系，更不会与上一章的内容拉上关系，让读者能够随时独立复习每个知识点，而不用再去复习不必要的章节内容。

(5) 本书的示例程序主要是针对语法问题的，示例程序每行都有注释，尽量做到把每个语法问题都反映出来。

(6) 本书引用了大多数教材上没有提到的一些概念，并对这些概念做了深入介绍。

(7) 本书对某些难点内容做了细致的图解分析，让读者更容易明白难点的原理。书中的图是专门针对语法问题的，尽量做到让读者看图就能明白其原理。

(8) 本书对指针和数组的理解有独到的见解，学完数组和指针章节会给读者耳目一新的感觉。

(9) 本书尽量做到用最少的文字、最少的篇幅描述清楚知识点，是一本真正的含金量高的图书。

由于能力有限，书中难免有错漏之处，望广大读者指出更正，不胜感激。

读者服务

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- **下载资源：**本书提供的附录 A 和附录 B 文件，可在 下载资源 处下载。
- **提交勘误：**您对书中内容的修改意见可在 提交勘误 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动：**在页面下方 读者评论 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：<http://www.broadview.com.cn/31655>



目 录

第 1 章 C++快速入门	1
第 2 章 整型、字符型和浮点型专题	6
2.1 基础	6
2.2 整型和 sizeof 操作符	8
2.3 char (字符型)	11
2.4 bool (布尔型)	16
2.5 浮点型	17
2.6 符号常量和#define 预处理指令简介	22
第 3 章 声明、定义、复杂声明和 typedef 专题	23
3.1 声明与定义	23
3.2 复杂声明和 typedef 的使用	31
3.2.1 复杂声明	31
3.2.2 typedef	34
第 4 章 运算符、表达式和左值专题	40
4.1 赋值、左值和右值	41
4.2 表达式和运算符	42
4.2.1 基础	42
4.2.2 表达式的副作用和顺序点	44
4.2.3 运算符的优先级、结合性和操作数的求值顺序	45
4.2.4 运算符性质总结	46
4.3 运算符	47

4.3.1	二元算术运算符.....	47
4.3.2	关系运算符.....	49
4.3.3	逻辑运算符.....	50
4.3.4	赋值运算符.....	51
4.3.5	复合赋值运算符.....	52
4.3.6	递增和递减运算符.....	54
4.3.7	位运算符.....	56
4.3.8	条件运算符.....	58
4.3.9	逗号运算符.....	60
4.3.10	sizeof 运算符.....	61
4.4	类型转换.....	61
4.4.1	基础.....	61
4.4.2	各种类型转换.....	61
4.4.3	转换溢出处理.....	64
4.4.4	强制类型转换运算符.....	65
第 5 章	选择语句和循环语句专题	68
5.1	语句概念	68
5.2	if 语句.....	69
5.3	switch 语句.....	72
5.4	while 和 do-while 语句	77
5.5	for 语句.....	79
5.6	continue 和 break 语句.....	81
5.7	循环语句头定义的变量的作用域.....	82
5.8	goto 跳转语句简介	83
第 6 章	指针和数组专题.....	84
6.1	指针	84
6.1.1	指针的概念.....	87
6.1.2	&与*运算符.....	88
6.1.3	指针（变量）的声明、初始化.....	92
6.1.4	各种指针.....	93
6.1.5	指针的简单运算.....	98

6.2	数组	100
6.2.1	一维数组	100
6.2.2	多维数组	104
6.3	指针与数组	106
6.3.1	理解数组名	106
6.3.2	指针与数组的混合运算	109
6.3.3	数组指针(*p)[]和指针数组*p[]	112
6.4	动态分配内存 new 关键字	115
6.4.1	内存管理基础	115
6.4.2	使用 new 动态分配单个对象	116
6.4.3	使用 new 动态创建数组	118
6.4.4	使用 new 动态分配内存的类型分析	121
6.4.5	使用多级指针动态创建多维数组	122
6.4.6	delete 常见错误及内存错误	125
6.4.7	使用 malloc/free 动态创建和释放内存简介	127
6.5	C 风格字符串	128
6.5.1	C 风格字符串	128
6.5.2	C 风格字符串的标准库函数	131
6.5.3	C 风格字符串的输入/输出	132

第 7 章 函数和标识符的作用域专题..... 133

7.1	函数基本语法规则	133
7.1.1	函数声明、定义及形参的语法规则	133
7.1.2	函数调用、实参、返回值、return 语句语法规则	137
7.2	函数参数传递	141
7.2.1	指针形参和引用形参	141
7.2.2	数组形参	144
7.2.3	函数指针	148
7.2.4	默认参数与可变形参	150
7.2.5	内联函数、main 函数、extern"C"链接指示符	151
7.3	函数重载	154
7.4	函数匹配（或函数重载解析）	155
7.4.1	函数匹配的过程	155

7.4.2	候选函数的确定方法	156
7.4.3	确定最佳匹配函数的方法	156
7.4.4	完全匹配详解	159
7.5	作用域、存储持续期、链接性和存储类区分符	164
7.5.1	作用域	164
7.5.2	存储持续期、链接性与作用域	167
7.5.3	将程序写在多个文件中	175

第8章 类基础、类作用域及相关运算符专题.....177

8.1	面向对象程序设计基本概念	177
8.2	类的声明/定义、类成员简介及相关运算符	181
8.2.1	类和对象的声明、定义	181
8.2.2	类成员简介、成员运算符、作用域解析运算符、访问控制符	182
8.3	类作用域	187
8.3.1	类作用域中的名称	187
8.3.2	类作用域中的名称解析	189

第9章 构造函数、复制构造函数和析构函数专题.....192

9.1	构造函数与析构函数简介	192
9.1.1	构造函数、默认构造函数、单形参构造函数、explicit 关键字	192
9.1.2	析构函数	195
9.2	对象初始化	198
9.2.1	使用构造函数、默认构造函数初始化对象	198
9.2.2	使用成员初始化表初始化数据成员	200
9.2.3	使用复制构造函数初始化对象及临时对象	202

第10章 名称空间专题.....207

10.1	名称空间基础	207
10.2	名称空间的分类	209
10.3	访问名称空间中的名称	211
10.4	名称空间中的名称解析	214

第 11 章 类中的成员专题	217
11.1 静态成员	217
11.1.1 静态数据成员	217
11.1.2 静态成员函数	221
11.2 const 成员、mutable 关键字、this 指针	222
11.3 对象数组、对象成员、数组成员和对象数组成员	225
11.3.1 对象数组	225
11.3.2 对象成员、数组成员和对象数组成员	226
11.4 嵌套类、局部类、友元	228
11.4.1 嵌套类	228
11.4.2 局部类	232
11.4.3 友元	232
11.5 指向类成员的指针	237
11.6 枚举、联合（共用体）、位段（域）	240
11.6.1 枚举类型	240
11.6.2 联合（共用体）类型	244
11.6.3 位段（域）	246
第 12 章 运算符（操作符）重载专题	249
12.1 运算符重载基本概念	249
12.2 运算符重载示例	252
12.3 转换函数和重载解析	260
12.3.1 转换函数	260
12.3.2 有转换函数时的函数重载解析	263
12.3.3 带有类类型实参和在类作用域中调用函数时函数重载解析	266
12.3.4 重载运算符函数时的重载解析	268
12.3.5 仿函数与重载解析	270
12.4 重载 new/delete 运算符和定位 new/delete	271
12.4.1 重载 new/delete 运算符	271
12.4.2 定位（布局）new 和 delete	277
12.4.3 new 表达式和 new 运算符函数总结	282

第 13 章 继承、虚函数与多态性专题	284
13.1 继承	284
13.1.1 继承基础及继承后的访问级别	284
13.1.2 继承下的构造函数与复制控制	289
13.1.3 父类与子类间的转换	291
13.1.4 继承下的名称解析、名称隐藏及函数重载解析	294
13.1.5 多重继承与虚基类	297
13.2 虚函数与多态性	302
13.2.1 多态性原理	302
13.2.2 虚函数	305
第 14 章 对象模型与虚函数表专题	314
14.1 对象模型与虚函数表基础、内存对齐、函数内部转换	314
14.1.1 对象模型简介	314
14.1.2 类成员的存储次序与内存对齐	318
14.1.3 编译器对函数的内部转换与名称改编	322
14.1.4 指向虚成员函数的指针	325
14.1.5 对成员函数的各种转换总结	325
14.2 各种 C++ 对象模型	326
14.2.1 指针与类型的关系	326
14.2.2 VC++ 2010 访问虚函数表的三种方法	327
14.2.3 单继承下的对象模型	330
14.2.4 多重继承下的对象模型与 this 指针调整	332
14.2.5 虚继承下的对象模型	339
14.3 编译器合成的各种构造函数和析构函数	342
14.3.1 编译器合成的默认构造函数	342
14.3.2 编译器合成的复制构造函数与按成员初始化	346
14.3.3 编译器合成的复制赋值操作符函数	349
14.3.4 编译器合成的析构函数	349
14.4 类对象创建和销毁时编译器实现原理	349
第 15 章 模板专题	354
15.1 模板基础	354

15.2	模板形参与模板实参详解	359
15.2.1	类型形/实参与非类型形/实参	359
15.2.2	默认模板实参	363
15.2.3	模板模板形/实参	364
15.3	模板实参推演与显式模板实参	365
15.3.1	基础	365
15.3.2	模板实参推演	368
15.3.3	显式模板实参	374
15.4	名称的识别与依赖实参的查询	375
15.4.1	依赖实参的查询 (ADL)	375
15.4.2	typename 前缀和 template 前缀	380
15.5	实例化	383
15.5.1	实例化基本规则	384
15.5.2	实例化的时机和位置点及两段式名称查询	384
15.5.3	显式实例化	393
15.6	类模板中的成员	396
15.7	模板特化	401
15.7.1	全局特化与局部特化	401
15.7.2	类模板成员的特化及定义	406
15.8	有模板时的函数重载解析	410
15.9	模板与友元	416
15.9.1	基础	416
15.9.2	把模板或其实例声明为友元	417
15.10	模板与继承	419
第 16 章	I/O 专题	421
16.1	I/O 流模型及 I/O 类组织结构	421
16.1.1	I/O 流模型	421
16.1.2	I/O 类组织结构	422
16.2	标准输出流 (ostream 类)	425
16.2.1	使用 ostream 类的成员函数进行输出	425
16.2.2	控制输出时的格式	426
16.3	标准输入流 (istream 类)	432

16.3.1	流状态	432
16.3.2	使用 istream 类的成员函数进行输入	434
16.4	文件流	438
16.5	字符串流	445
16.6	C 风格字符串流	447
第 17 章	异常专题	448
第 18 章	预处理器、typeid 和强制类型转换专题	464
第 19 章	string 类专题	472
参考文献		488

第 1 章

C++快速入门

- (1) C++是大小写敏感的语言，即在程序中大写字母 C 和小写字母 c 是不同的。
 - (2) 关键字(保留字)：关键字是 C++使用的名字，任何自定义的名字都不能与关键字相同。
 - (3) 注意：C++中的所有符号都应在英文状态下输入，比如分号应使用“;”，而不是“;”。
- 示例如下：

```
/*此符号是C++的多行
注释符号*/
#include<iostream>
using namespace std;    //双斜杠符号是C++的单行注释符号
int main()
{cout << "hy" <<endl;    //将右边的表达式输出到由<<指向的左边的设备cout上。cout一般表示控制台
    cin.get();            //暂停等待输入，或按下回车键继续执行，此句可以省略
    return 0;
}
```

程序说明如下。

1. 解决执行后的结果一闪而过

有些编译器在运行 C++程序时，会在独立的窗口中运行，程序执行完之后窗口就关闭了(程序执行得很快，一般情况下，窗口都是一闪而过)，这样就无法看到程序的执行结果。解决方法就是在程序中想要暂停的地方加上 `cin.get();`语句，这样程序就会停在这里等按下回车键。一般情况下，只需将 `cin.get();`放在 `main` 函数的结尾或者 `return` 语句之前就行。

2. 注释

(1) 注释：注释的内容是不会被编译器编译的，注释就是程序中对代码的说明性文字，这些说明性文字是为提高程序的可读性而写的，可以在注释中写入任何内容，当然也可以没有注释。

(2) 单行注释使用“//”符号：C++使用“//”作为程序注释的开头，“//”后面的内容为程序的注释。但使用“//”只能注释单行的程序，若注释超过一行，到第二行又必须以“//”作为

注释的开头。

(3) 多行注释使用`/*...*/`符号：这是 C 风格的注释，但 C++ 同样支持，`/*`和`*/`之间的内容为注释，以`/*`开头，以`*/`结束，这种注释方法可以跨越多行。

3. 预处理器和`#include<iostream>`

(1) `#include` 是预处理器指令，预处理器就是编译器在把代码编译为机器指令之前执行的一个过程，它会在编译程序的时候自动运行，所有的预处理器都是以“`#`”开头的，其中 `include` 是预处理器的一个指令。

(2) `#include<iostream>`的作用：将 `iostream` 文件的内容添加到程序中。在将代码编译为机器指令之前，会将 `iostream` 文件的内容添加到程序中，这时 `iostream` 文件的内容将取代代码行 `#include<iostream>`。

(3) `iostream` 文件的作用：这里的 `io` 指的是输入 (input) / 输出 (output)，使用 `cout` 进行输出或使用 `cin` 进行输入的程序必须包含 `iostream` 文件，因为 `cout`、`cin` 这些内容是在 `iostream` 文件中定义的。

4. 头文件（或包含文件）和 `iostream`

(1) 头文件（或包含文件）：像`#include<iostream>`中的 `iostream` 文件，被包含在其他文件起始处的文件被称为头文件或包含文件。

(2) 可以将任何合法的 C++ 文件包含进其他文件，只需像上面那样使用`#include<...>`即可。

(3) 库文件：C++ 编译器本身自带了很多头文件，这些头文件都有一定的功能，例如 `math.h`、`iostream.h`，其中 `math.h` 文件有很多关于数学计算方面的函数（函数见后文解释）。C++ 自带的这些标准头文件被称为库文件。

(4) 在 C 语言中头文件使用扩展名 `.h`，而 C++ 则不再使用扩展名了，因此老式的 C++ 编译器应使用`#include<iostream.h>`并将后面的 `using namespace std;`去掉，若还有更老的编译器，则应使用`#include<stream.h>`。因此，建议使用比较新的编译器。

(5) C++ 转换了 C 中的一些头文件，有些头文件还做了一些修改，并将这些文件重新命名，新命名的文件去掉了扩展名 `.h`，并在文件名前面加上了前缀 `c`，比如 `math.h` 的 C++ 版本为 `cmath`，转换后的文件可以使用 C 所不具有的 C++ 特性。

5. 名称空间

(1) 名称空间：就是提供一个声明名称的区域，比如在某个区域声明了一些名称，那么拥有这些名称的这个区域就是名称空间。当然，你可以为这个区域取一个名字，有了名称空间就能更好地控制名称的作用范围，同时一个名称空间中的名称不会与另一个名称空间中声明的相同名称相冲突。比如重庆市有一个合川市，广西也有一个合川市，在这里“重庆市”和“广西”

就是两个名称空间。若直接说合川市并不能知道它的具体位置，但说广西合川市就能知道它是广西的，这时并不会与重庆市的合川市相冲突。因此，在使用名称空间之前应先告诉程序，将使用哪一个名称空间中的名称，这样当以后出现这个名称时就知道它来自哪个名称空间了。

(2) 名称空间的创建：名称空间使用 `namespace` 来创建，比如 `namespace sss{...}` 就创建了一个名称空间，该名称空间的名称为“sss”，在大括号 `{}` 中可以声明名称。

(3) 使用名称空间中的名称。

- 可以使用 `::`（作用域解析运算符）来完全限定名称空间中的名称，比如 `std::cout` 就表示使用的是 `std` 名称空间中的名称 `cout`。
- 使用 `using` 编译指令：其语法格式为“`using namespace 名称空间的名称;`”，表示指定名称空间中的所有名称都可用。比如使用 `using namespace std;` 语句，就表示位于 `std` 名称空间中的所有名称都可直接使用。

6. std 名称空间和 C++标准库文件的关系

(1) `std` 是 C++ 的标准名称空间，所有 C++ 标准库文件中的名称都是在 `std` 名称空间中声明的。

(2) 凡是需要使用 C++ 标准库文件（需要使用 `#include` 将库文件包含进来）的地方都必须使用 `std` 名称空间，也就是应使用 `using namespace std;` 语句。当然，也可以通过上面讲的作用域解析运算符“`::`”来使用 `std` 名称空间中的名称。

7. 函数

(1) 函数基础。

① 函数：函数就是一种操作或一种功能。比如使用某函数实现对一些数据进行排序，再用另一个函数进行求和运算等。

② 函数的形式为：返回类型 函数名(形参列表){函数体}。比如 `int f(int a, int b){...}`，表示定义了一个名称为 `f`，返回类型为 `int`（整型），且带两个 `int` 类型的形参变量 `a` 和 `b` 的函数。

③ 函数若没有返回值，则只需使用 `void` 表示函数的返回类型即可。比如 `void f(){...}`，关键字 `void` 表示没有返回类型，也就是函数没有返回值。

④ 函数的形参列表也可以为空，比如 `int f(){...}`。注意，即使函数的形参列表为空，也不能省略函数名后面的小括号。

⑤ 函数由两部分组成：函数头和函数体。比如 `int main(){...}`，函数头是 `int main()`，函数体则是大括号 `{}` 中的内容，在函数体中可以编写程序以使函数实现一种操作或一种功能。

(2) 函数的调用。

① 在一个函数内部可以调用其他函数。

② 在一个函数中调用另一个函数只需使用另一个函数的函数名即可，当程序执行到被调用