计算机等级考试二级C 各章内容摘要10 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/133/2021_2022__E8_AE_A1_E7_AE_97_E6_9C_BA_E7_c97_133714.htm 第10章 C 流【考点一】 C 流的概念1.C 流的体系结构要利用C 流，必须在程序中包含有关的头文件，以便获得相关流类的声明。为了使用新标准的流，相关头文件的文件名中不得有扩展名。与C 流有关的头文件有：iostream：要使用cin、cout的预定义流对象进行针对标准设备的I/O操作，须包含此文件。fstream：要使用文件流对象进行针对磁盘文件的I/O，操作须包含此文件。strstream：要使用字符串流对象进行针对内存字符串空间的I/O操作，须包含此文件。iomanip：要使用setw、fixed等大多数操作符，须包含此文件。注意，为了使用新标准的C 流，还必须在程序文件的开始部分插入下面这个名字空间声明：using namespace std；2. 预定义流对象C 流有4 个预定义的流对象，它们的名称及与之联系的I/O设备如下：cin 标准输入cout标准输出cerr标准出错信息输出clog带缓冲的标准出错信息输出3.提取运算符和插入运算符输入流类istream重载了运算符，用于数据输入，其原形具有istreamamp.，类型修饰amp. operator（ostream&amp.，类型修饰）；的形式。重载的的功能是把表达式的值插入到输出流中，因此称之为插入运算符（inserter）。当系统执行coutx操作时，首先根据x值的类型调用相应的插入运算符重载函数，把x 的值传送给对应的形参，接着执行函数体，把x 的值（亦即形参的值）输出到显示器屏幕上，在当前屏幕光标位置起显示出来，然后返回ostream流，以便继续使用插入运算符输出下一个表达式的

值。上面格式中的"类型修饰符"是指char、int、double、char*、bool等等C 中固有类型的修饰符。也就是说，只要输入输出的数据属于这些C 固有类型中的一种，就可以直接使用或完成输入输出任务。在完成输入输出任务后，和把第一参数（即流对象的引用）返回，因此这两个运算符可以连续使用，如cinabc；等。4.有格式输入输出和无格式输入输出利用C 流既可进行有格式输入输出，也可进行无格式输入输出。计算机所处理的数据都有内部存储格式和外部表现形式的区分，因此在输入输出过程中必须进行适当的转换，有格式输入输出就是完成这一任务的。有格式输入输出针对的是键盘、显示器、打印机等字符设备以及磁盘中的文本文件。对于有格式输入输出，无论输入输出的数据是什么数据类型，体现在外部设备上都是字符序列。对于无格式输入输出，数据的内部存储存格式与外部存储格式完全相同，因此无格式输入输出只能针对磁盘文件（或磁带、光盘上的文件），而且这样的文件通常不能用一般的文本编辑器查看。进行无格式输入输出需调用流对象的专门的成员函数实现。5.操作符C 流提供了提取运算符和插入运算符，使得输入输出的表达简洁、形象、直观，这最能体现C 流的风格。C 流还提供了很多输入、输出或控制输入输出的成员函数，须通过．或一加以调用，不能与运算符或配合使用，因而与C 流的整体风格很不协调。为此，C 提供了一系列可与运算符或配合使用的特殊函数，称为操作符（manipulator）。每个操作符都与一个具体的函数相联系，使得或可间接地通过它们调用与之联系的函数，完成相应的输入、输出功能或输入输出控制功能。前面经常用到的endl、setw等就是操作符。有了操作符，C 流操

作在风格上就更加统一，输入输出操作也显得更加流畅。【考点二】输入输出的格式控制1.默认的输入输出格式在没有特地进行格式控制的情况下，输入输出采用默认格式。（1）默认的输入格式C 流所识别的输入数据的类型及其默认的输入格式包括：short、int、long（signed、unsigned）：与整型常量同float、double、long double：与浮点数常量同char（signed、unsigned）：第一个非空白字符 char *（signed、unsigned）：从第一个非空白字符开始到下一个空白字符结束void*：无前缀的16进制数bool：把true或1识别为true，其他的均识别为false（vc6.0中把0识别为false，其他的值均识别为true）（2）默认的输出格式C 流所识别的输出数据的类型及其默认的输出格式包括：char（signed、unsigned）：单个字符（无引号）short、int、long（signed、unsigned）：一般整数形式，负数前有－号char *（signed、unsigned）：字符序列（无引号）float、double、long double：浮点格式或指数格式（科学表示法），取决于哪个更短void *：无前缀的16进制数bool： 1或02.格式标志与格式控制在作为流库根类的ios_base中，有一个作为数据成员的格式控制变量，专门用来记录格式标志；通过设置标志，可以有意识地对有格式输入输出的效果加以控制。各种格式标志被定义为一组符号常量。这些作为格式标志的常量与整数的对应关系是精心安排的，每一个标志对应一个二进制位，为1时表示对应标志已设置，为0时表示对应标志未设置。这些作为标志的二进制位保存在格式控制变量的低端的若干位中，每一个流对象都有这样一个作为数据成员的格式控制变量。在外部使用这些格式标志时，必须在标志前加上ios_base：：修饰。格式标志中的

有些关系密切的相邻标志被规定为域，一共有三个：由left、right和 internal组成的域称为adjustfield（对齐方式域）；由dec、oct和hex组成的域称为basefield（数制方式域）；由scientific和fixed组成的域称为floatfield（浮点方式域）。adjustfield、basefield和floatfield也是在ios_base中定义的，因此在外部使用时也必须加上域修饰前缀ios_base：：（如ios_base：：adjustfield）。可以通过调用流对象的下列三个成员函数直接设置格式控制标志：●fmtflags setf（fmtflags fmtfl，fmtflags mask）；其中类型fmtflags实际上就是类型int。参数fmtf1为格式控制标志，参数mask为域。此函数用于设置某个域中的标志，设置前先将该域中所有标志清除。函数返回设置前的格式控制标志。●fmtflags setf（fmtflags fmtf1）；其中参数fmtf1为格式控制标志。此函数用于设置指定的标志，即将指定的标志位置为1，其他标志位不受影响。函数返回设置前的格式控制标志。此函数多用于adjustfield、basefield和floatfield三个域之外的格式控制标志的设置。●void unsetf（fmtflags fmtf1）；其中参数fmtf1为格式控制标志或域。此函数用于清除指定标志或域，即将指定标志位或域清0。除了使用上述函数外，还可以用操作符进行格式控制。对应于上述setf函数的操作符是：setiosflags（＜格式控制标志＞），对应于上述的unsetf函数的操作符是 ：resetiosflags（格式控制标志或域）。3.输入输出宽度的控制宽度的设置可用于输入，但只对字符串输入有效。对于输出，宽度是指最小输出宽度。当实际数据宽度小于指定的宽度时，多余的位置用填充字符（通常是空格）添满；当实际数据的宽度大于设置的宽度时，仍按实际的宽度输出。初始宽度值为0，其含义是所有数

据都将按实际宽度输出。宽度的设置与格式标志无关。有关的操作符是：●setw（int n)：设置输入输出宽度；等价函数调用：io.width（n） 其中n为一个表示宽度的表达式。如果用于输入字符串，实际输入的字符串的最大长度为n-1。也就是说宽度n连字符串结束符也包含在内。函数 width返回此前设置的宽度；如果只需要这个返回值，可不给参数。注意：宽度设置的效果只对一次输入或输出有效，在完成了一个数据的输入或输出后，宽度设置自动恢复为0（表示按数据实际宽度输入输出）。宽度设置是所有格式设置中唯一的一次有效的设置。4.浮点数输出方式的控制在初始状态下，浮点数都按浮点格式输出，输出精度的含义是有效位的个数，小数点的相对位置随数据的不同而浮动；可以改变设置，使浮点数按定点格式或指数格式（科学表示法，如3.2156e 2）输出。在这种情况下，输出精度的含义是小数位数，小数点的相对位置固定不变，必要时进行舍入处理或添加无效0。设置的输出方式一直有效，直到再次设置浮点数输出方式时为止。有关操作符有：●resetiosflags（ios_base：：floatfield）：（此为默认设置）浮点数按浮点格式输出；等价函数调用：o.unsetf（ios_base：：floatfield）●fixed ：浮点数按定点格式输出；等价函数调用：o.setf（ios_base：：fixed，ios_base：：floatfield）●scientific：浮点数按指数格式（科学表示法）输出；等价函数调用：o.setf（ios_base：：scientific,ios_base：：floatfield）。5.输出精度的控制输入输出精度是针对浮点数设置的，其实际含义与浮点数输出方式有关：如果采用的是浮点格式，精度的含义是有效位数；如果采用的是定点格式或指数格式（科学表示法），精度的含义是小数位数。精度

的设置用于输出，默认精度为6，可以通过设置改为任意精度；将精度值设置为0意味着回到默认精度6。设置的精度值一直有效，直到再次设置精度时为止。精度的设置与格式标志无关。有关操作符是：●setprecision（int n）：设置浮点数的精度（有效位数或小数位数）；等价函数调用：io.precision（n）其中n为表明精度值的表达式。函数返回此前设置的精度；如果只需要这个返回值，可不给参数。