C++primer 第一章笔记(P1~25)

by Henry Huang 2021-01

目录

0.第一章简述

因为第一章比较简短，因此有个简述。

第一章就是通过一个需求：写一个书店的小程序，可以保存书的ISBN、售出册数、售出单价，

引出了下列概念：

- 定义变量
- 输入输出
- 使用数据结构
- 判断语句
- 循环语句

因此下面只记录一些概念性和之前不会的东西。

1.初识输入输出

1.1 标准库

C++没有输入输出语句，而是使用标准库来提供IO机制。

> The C++ language does not define any statements to do input or output (IO). Instead, C++ includes an extensive standard library that provides IO (and many other facilities).

常用的输入输出使用iostream库，有**两个输入输出类型**，如下：

- 输入：istream**类型**
- 输出：ostream**类型**

1.2 标准的IO对象

有4个IO对象：

- cin,istream类型的**对象**，**标准输入 standard input**
- cout,ostream类型的**对象**，**标准输出 standard output**
- cerr,ostream类型的**对象**，用于输出警告/错误
- clog,ostream类型的**对象**，用于输出运行时的一般性信息

1.3 输入输出运算符

- <<：输出运算符，将右侧的值写到左侧的ostream对象中
- >>：输入运算符，将左侧的istream对象写入到右侧对象

2. 注释 comments

2.1 单行注释

单行注释：

- 开始：“//”
- 结束:换行符

例子：

```
// 单行注释
```

2.2多行注释

也叫界定符注释，使用一对“/* */” 符号。

多行注释不能嵌套，而单行注释中的所有内容都会被忽略,因此嵌套的其他注释也会被忽略**

书上建议：针对跨越多行的注释，建议/*中的其他行，以* 开头，来显示指出此为注释部分。

例子：

```
/*多行注释
* 注释
* 注释
*/
```

2.3 习题1.8

```
    std::cout <</*"*/"*/;
    std::cout <</*"*/"/*"/*"*/;
```

第一个是报错，第二个是对的，可以看到，左右是两个被注释的双引号，而中间是一个被双引号括起来的/*字面值常量。

3.不定量输入(P13)

因为while、for、if我都熟悉了。所以细节留到后面记录。

这里记录一下怎么进行不定量的输入。

```
int value=0;
while(std::cin >> value)
{
    语句块;
}
```

把输入流作为条件，则是检测流输入的情况，输入成功则不报错继续输入。

当出现以下两种情况，输入停止：

- 遇到EOF(end of file)文件结束符(windows 中使用 cltr+z在键盘键入文件结束符)
- 无效输入，如非整形输入

4.编译器可检测的错误

编译器一般可以检测出以下**形式(form)**错误：

- **语法错误(syntax error)**，如漏掉分号
- 类型错误(type error)，如向int类型变量赋予字面值常量
- **声明错误(declaration error)**，如变量使用之前未被声明

5. C++的格式自由(P16)

缩进、哪里放置花括号等，其实是不影响程序的语义的。

唯一的格式要求：左花括号必须是main的形参列表后第一个**非空非注释**的字符。

6. 类(class)初见

- 在 类名.h 的头文件中定义类
- **类其实就是类型，新定义一个类，其实就是自定义一个新的类型。类型名就是类名。**因此前文提到的类型的对象，其实就是拥有某一类型的变量，其实就是类的一个实例。
- 类定义了其对象可以进行的所有操作
- 每个类都有数据和方法(成员函数)

7.概念总结：

7.1 类型 type

类型定义了**数据的元素内容**，以及数据上**可以进行的运算**

程序的所有数据都保存在变量中，而每个变量都应该有他的类型。

7.2 函数 function

一个函数由四部分组成：

- 返回类型 return type（return若包含值，则其类型必须与函数类型相同）
- 函数名 function name
- 形参列表 parameter list（可以为空）
- 函数体 function body（在花括号中的语句块）

7.3 IDE(Intergrated Develoed Environment)

集成开发环境

7.4 流 stream

流，即字符串序列，从IO设备读入/写入IO设备。

流，反映随时间推移，字符串是顺序产生/消耗的。

7.5 buff刷新

使用endl操纵符(manipulator)，可以结束当前行，并刷新缓冲区。

刷新缓冲区的操作可以使得内存中的数据真正的被写入流中，而不是停留在内存上。

7.6 命名空间 namespace

- 好处：可以避免命名冲突（避免定义的名字与标准库名字冲突等）
- 副作用：使用某个命名空间的名字时，必须显示指定命名空间，如std::cin。表面使用来自std命名空间的cin
- ::：双冒号是**作用域运算符**
- std：是标准库的命名空间。如果不想每次std::，**有以下两种方案**：
    - using namespace std; 这会导入所有标准库的名字
    - using std::cin; 只导入std中的cin，以后直接cin就可以使用

7.7 术语表

术语表

参数（实参，argument）向函数传递的值。

赋值（assignment）抹去一个对象的当前值，用一个新值取代之。

程序块（block）零条或多条语句的序列，用花括号包围。

缓冲区（buffer）一个存储区域，用于保存数据。IO 设施通常将输入（或输出）数据保存在一个缓冲区中，读写缓冲区的动作与程序中的动作是无关的。我们可以显式地刷新输出缓冲，以便强制将缓冲区中的数据写入输出设备。默认情况下，读 cin 会刷新 cout；程序非正常终止时也会刷新 cout。

内置类型（built-in type）由语言定义的类型，如 int。

Cerr 一个 ostream 对象，关联到标准错误，通常写入到与标准输出相同的设备。默认情况下，写到 cerr 的数据是不缓冲的。cerr 通常用于输出错误信息或其他不属于程序正常逻辑的输出内容。

字符串字面值常量（character string literal）术语 string literal 的另一种叫法。

cin 一个 istream 对象，用来从标准输入读取数据。

类（class）一种用于定义自己的数据结构及其相关操作的机制。类是 C++中最基本的特性之一。标准库类型中，如 istream 和 ostream 都是类。

类类型（class type）类定义的类型。类名即为类型名。

clog 一个 ostream 对象，关联到标准错误。默认情况下，写到 clog 的数据是被缓冲的。clog 通常用于报告程序的执行信息，存入一个日志文件中。

注释（comment）被编译器忽略的程序文本。C++有两种类型的注释：单行注释和界定符对注释。单行注释以//开始，从//到行尾的所有内容都是注释。界定符对注释以/*开始，其后的所有内容都是注释，直至遇到*/为止。

条件（condition）求值结果为真或假的表达式。通常用值 0 表示假，用非零值表示真。

cout 一个 ostream 对象，用于将数据写入标准输出。通常用于程序的正常输出内容。

花括号（curly brace）花括号用于划定程序块边界。左花括号（{）为程序块开始，右花括号（}）为结束。

数据结构（data structure）数据及其上所允许的操作的一种逻辑组合。

编辑-编译-调试（edit-compile-debug）使程序能正确执行的开发过程。

文件结束符（end-of-file） 系统特定的标识，指出文件中无更多数据了。

表达式（expression）最小的计算单元。一个表达式包含一个或多个运算对象，通常还包含一个或多个运算符。表达式求值会产生一个结果。例如，假设 i 和 j 是 int 对象，则 i+j 是一个表达式，它产生两个

int 值的和。

for 语句（for statement）迭代语句，提供重复执行能力。通常用来将一个计算反复执行指定次数。

函数（function）具名的计算单元。

函数体（function body）语句块，定义了函数所执行的动作。

函数名（function name） 函数为人所知的名字，也用来进行函数调用。

头文件（header）使类或其他名字的定义可被多个程序使用的一种机制。程序通过 #include 指令使用头文件。

if 语句（if statement）根据一个特定条件的值进行条件执行的语句。如果条件为真，执行 if 语句体。否则，执行 else 语句体（如果存在的话）。

初始化（initialize）在一个对象创建的时候赋予它一个值。

iostream 头文件，提供了面向流的输入输出的标准库类型。

istream 提供了面向流的输入的库类型。

库类型（library type）标准库定义的类型，如 istream。

main 操作系统执行一个 C++程序时所调用的函数。每个程序必须有且只有一个命名为 main 的函数。

操纵符（manipulator）对象，如 std::endl，在读写流的时候用来“操纵”流本身。

成员函数（member function）类定义的操作。通常通过调用成员函数来操作特定对象。

方法（method）成员函数的同义术语。

命名空间（namespace）将库定义的名字放在一个单一位置的机制。命名空间可以帮助避免不经意的名字冲突。C++标准库定义的名字在命名空间 std 中。

ostream 标准库类型，提供面向流的输出。

形参列表（parameter list）函数定义的一部分，指出调用函数时可以使用什么样的实参，可能为空列表。

返回类型（return type）函数返回值的类型。

源文件（source file）包含 C++程序的文件。

标准错误（standard error）输出流，用于报告错误。标准输出和标准错误通常关联到程序执行所在的窗口。

标准输入（standard input）输入流，通常与程序执行所在窗口相关联。

标准库（standard library）一个类型和函数的集合，每个 C++编译器都必须支持。标准库提供了支持 IO 操作的类型。C++程序员倾向于用“库”指代整个标准库，还倾向于用库类型表示标准库的特定部分，例如用“iostream 库”表示标准库中定义 IO 类的部分。

标准输出（standard output）输出流，通常与程序执行所在窗口相关联。

语句（statement）程序的一部分，指定了当程序执行时进行什么动作。一个表达式接一个分号就是一条语句；其他类型的语句包括语句块、if 语句、for 语句和 while 语句，所有这些语句内都包含其他语句。

std 标准库所使用的命名空间。std::cout 表示我们要使用定义在命名空间 std 中的名字 cout。

字符串常量（string literal）零或多个字符组成的序列，用双引号包围（"a string literal"）。

未初始化的变量（uninitialized variable）未赋予初值的变量。类类型的变量如果未指定初值，则按类定义指定的方式进行初始化。定义在函数内部的内置类型变量默认是不初始化的，除非有显式的初始化语句。试图使用一个未初始化变量的值是错误的。未初始化变量是 bug 的常见成因。

变量（variable）具名对象。

while 语句（while statement）迭代语句，提供重复执行直至一个特定条件为假的机制。循环体会执行零次或多次，依赖于循环条件求值结果。

()运算符（() operator）调用运算符。跟随在函数名之后的一对括号“()”，起到调用函数的效果。传递给函数的实参放置在括号内。

++运算符（++ operator）递增运算符。将运算对象的值加 1，++i 等价于 i=i+1。

+=运算符（+= operator）复合赋值运算符，将右侧运算对象加到左侧运算对象上；a+=b 等价于 a=a+b。

.运算符（. operator）点运算符。左侧运算对象必须是一个类类型对象，右侧运算对象必须是此对象的一个成员的名字。运算结果即为该对象的这个成员。

::运算符（:: operator）作用域运算符。其用处之一是访问命名空间中的名字。例如，std::cout 表示命名空间 std 中的名字 cout。

=运算符（= operator）将右侧运算对象的值赋予左侧运算对象所表示的对象。

—运算符（— operator）递减运算符。将运算对象的值减 1，--i 等价于 i=i-1。

<<运算符（<< operator）输出运算符。将右侧运算对象的值写到左侧运算对象表示的输出流：cout << "hi"表示将 hi 写到标准输出。输出运算符可以连接：cout << "hi" << "bye"表示将输出 hibye。

>>运算符（>> operator）输入运算符。从左侧运算对象所指定的输入流读取数据，存入右侧运算对象中：cin >> i 表示从标准输入读取下一个值，存入 i 中。输入运算符可以连接：cin >> i >> j 表示先读取一个值存入 i，再读取一个值存入 j。

#include 头文件包含指令，使头文件中代码可被程序使用。

==运算符（== operator）相等运算符。检测左侧运算对象是否等于右侧运算对象。

!=运算符（!= operator）不等运算符。检测左侧运算对象是否不等于右侧运算对象。

<=运算符（<= operator）小于等于运算符。检测左侧运算对象是否小于等于右侧运算对象。

<运算符（< operator）小于运算符。检测左侧运算对象是否小于右侧运算对象。

>=运算符（>= operator）大于等于运算符。检测左侧运算对象是否大于等于右侧运算对象。

>运算符（> operator）大于运算符。检测左侧运算对象是否大于右侧运算对象。