

( 第5版 )

# C++ Primer 习题集

Stanley B. Lippman  
[美] Josée Lajoie 著  
Barbara E. Moo  
王刚 杨巨峰 李忠伟 改编

# 第 1 章

## 开始

### 导读

在本章，我们初识 C++ 语言，学习了如何编译、运行简单的 C++ 程序。因此，本章习题的一个主要训练内容是熟悉编译工具和集成开发环境。本章还介绍了 C++ 程序的基本结构，如何定义变量，如何进行输入输出，以及如何编写 if、for 和 while 语句，用这些知识来练习编写简单的 C++ 程序是本章练习的另一个重要内容。本章最后简要介绍了如何使用其他人定义的类，因此，本章最后一部分习题是利用书中的 Sales-item 类进行类对象的创建、使用方面的练习。

**练习 1.1：**查阅你使用的编译器的文档，确定它所使用的命名约定。编译并运行第 2 页的 main 程序。

**【出题思路】**

熟悉编译工具和集成开发环境。

**【解答】**

首先利用编辑器（如 Linux 系统中的 vim 或 Windows 系统中 Visual Studio 自带的编辑器）输入 main 程序，保存为 .cpp 或 .cc 后缀的源程序文件。然后按书中说明运行 GNU 或微软编译器，将源文件编译为可执行文件。最后执行程序，观察执行结果。

尝试使用命令行方式编译程序，尝试不同编译选项，可以帮助我们更好地了解编译过程，掌握生成所需目标程序的方法。

**练习 1.2：** 改写程序，让它返回-1。返回值-1 通常被当作程序错误的标识。重新编译并运行你的程序，观察你的系统如何处理 main 返回的错误标识。

**【出题思路】**

了解 C++ 程序与操作系统间的交互。

**【解答】**

Windows 7 操作系统并不处理或报告程序返回的错误标识，直观上，返回-1 的程序与返回 0 的程序在执行效果上并无不同。但环境变量 ERRORLEVEL 记录了上一个程序的返回值。因此，在控制台窗口执行修改后的程序，接着执行 echo %ERRORLEVEL%，会输出-1。在 Linux 系统中，执行 echo \$? 有类似效果。

**练习 1.3：** 编写程序，在标准输出上打印 Hello, World。

**【出题思路】**

C++ 程序基本结构和简单输出的练习。

**【解答】**

```
#include<iostream>
int main()
{
    std::cout << "Hello, World" << std::endl;
    return 0;
}
```

**练习 1.4：** 我们的程序使用加法运算符+来将两个数相加。编写程序使用乘法运算符\*，来打印两个数的积。

**【出题思路】**

简单运算和简单输入输出的练习。

**【解答】**

```
#include <iostream>

int main()
{
    std::cout << "请输入两个数" << std::endl;
    int v1, v2;
    std::cin >> v1 >> v2;
    std::cout << v1 << "和" << v2 << "的积为"
    << v1 * v2 << std::endl;
    return 0;
}
```

**练习 1.5：** 我们将所有输出操作放在一条很长的语句中。重写程序，将每个运算对象的打印操作放在一条独立的语句中。

**【出题思路】**

对输出语句进行不同形式的练习，同时让读者体会良好的程序格式。

### 【解答】

```
#include <iostream>

int main()
{
    std::cout << "请输入两个数";
    std::cout << std::endl;
    int v1, v2;
    std::cin >> v1 >> v2;
    std::cout << v1 << "和" << v2 << "的积为"
        << v1 * v2 << std::endl;
    return 0;
}
```

### **练习 1.6.** 解释下面程序片段是否合法。

```
std::cout << "The sum of " << v1;
    << " and " << v2;
    << " is " << v1 + v2 << std::endl;
```

如果程序是合法的，它输出什么？如果程序不合法，原因何在？应该如何修正？

### 【出题思路】

延续练习 1.5，让读者体会输出语句分段形式容易出现的错误。

### 【解答】

这段代码不合法。

前两行的末尾有分号，表示语句结束，第 2、3 两行为两条新的语句。而这两条语句在“<<”之前缺少了输出流，应在“<<”之前加上“std::cout”，即得到正确的程序。

### **练习 1.7.** 编译一个包含不正确的嵌套注释的程序，观察编译器返回的错误信息。

### 【出题思路】

一方面了解非法嵌套注释，另一方面体会编译器对较为复杂的错误如何给出错误信息，程序员应如何利用这些信息迅速定位、修改错误。

### 【解答】

对不正确的嵌套注释，不同编译器给出的错误信息可能是不同的，而且通常很难理解。例如，用 tdm-gcc 4.8.1 编译器编译 1.3 节中错误嵌套的程序：

```
#include <iostream>

/*
 * 注释对/* */不能嵌套。
 * “不能嵌套”几个字会被认为是源码,
 * 像剩余程序一样处理
 */

int main()
```

```
{
    return 0;
}
```

编译器会报告：

```
4 error: stray '\262' in program
4 error: stray '\273' in program
```

(篇幅所限，后续错误信息略。)

原因是编译器将第一个“\*/”看作注释结束，之后的中文文字看作下一条语句，从而给出非法字符的错误信息。如果“\*/”之后是英文文字，或是使用其他编译器进行编译，给出的可能是完全不同的错误信息。而且这些错误信息都很难直接与注释错误嵌套挂上钩，程序员需要有一定的经验才能快速定位错误，确定错误原因。

### **练习 1.8：**指出下列哪些输出语句是合法的（如果有的话）：

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

预测编译这些语句会产生什么样的结果，实际编译这些语句来验证你的答案（编写一个小程序，每次将上述一条语句作为其主体），改正每个编译错误。

#### 【出题思路】

进一步熟悉更复杂的正确和不正确的注释语句。

#### 【解答】

第一条和第二条语句显然是合法的。

在第三条语句中，第一个双引号被注释掉了，因此<<运算符后真正被编译的内容是“ \*/”，编译器认为这是一个不完整的字符串，所以会报告：

```
7 error: missing terminating " character
```

即，缺少结束的双引号。在分号前补上一个双引号，这条语句就变为正确的了。

第四条看起来很混乱，但它是正确的。第一个双引号被注释掉了，第四个双引号也被注释掉了，第二个双引号和第三个双引号之间的“ /\* ”被认为是字符串的文字内容。但是，这样的程序风格显然是不好的。

### **练习 1.9：**编写程序，使用 while 循环将 50 到 100 的整数相加。

#### 【解答】

```
#include <iostream>

int main()
{
    int sum = 0;
    int i = 50;
    while (i <= 100) {
        sum += i;
        i++;
    }
}
```

```

    std::cout << "50 到 100 之间的整数之和为"
    << sum << std::endl;
    return 0;
}

```

**练习 1.10:**除了++运算符将运算对象的值增加 1 以外，还有一个递减运算符(--)实现将值减少 1。编写程序，使用递减运算符在循环中按递减序打印出 10 到 0 之间的整数。

### 【出题思路】

递减循环较之递增循环用得较少，应有意进行这方面的练习，对提高编程能力是有益的。

### 【解答】

```

#include <iostream>

int main()
{
    int i = 10;
    while (i >= 0) {
        std::cout << i << " ";
        i--;
    }
    std::cout << std::endl;
    return 0;
}

```

**练习 1.11:** 编写程序，提示用户输入两个整数，打印出这两个整数所指定的范围内所有整数。

### 【出题思路】

编写一个简单但完整的依据用户输入进行处理的实例。让读者体会：用户的输入可能有各种各样的情况，我们编写的程序必须全面地考虑各种情况，避免由于考虑不周使得程序在某些用户输入下产生错误结果甚至更严重的后果。

### 【解答】

```

#include <iostream>

int main()
{
    std::cout << "请输入两个数";
    std::cout << std::endl;
    int v1, v2;
    std::cin >> v1 >> v2;
    if (v1 > v2) // 由大至小打印
        while (v1 >= v2) {
            std::cout << v1 << " ";
            v1--;
        }
}

```

```

    else          // 由小至大打印
        while (v1 <= v2) {
            std::cout << v1 << " ";
            v1++;
        }
    std::cout << std::endl;
    return 0;
}

```

**练习 1.12:** 下面的 for 循环完成了什么功能? sum 的终值是多少?

```

int sum = 0;
for (int i = -100; i <= 100; ++i)
    sum += i;

```

**【解答】**

此循环将 -100 到 100 之间（包含 -100 和 100）的整数相加，sum 的终值是 0。

**练习 1.13:** 使用 for 循环重做 1.4.1 节中的所有练习（第 11 页）。

**【出题思路】**

让读者体会：对同样的目标，C++语言提供的不同解决方法。

**【解答】**

练习 1.9 的循环版本：

```
#include <iostream>
```

```

int main()
{
    int sum = 0;
    for (int i = 50; i <= 100; i++)
        sum += i;
    std::cout << "50 到 100 之间的整数之和为"
        << sum << std::endl;
    return 0;
}

```

练习 1.10 的循环版本：

```
#include <iostream>
```

```

int main()
{
    for (int i = 10; i >= 0; i--)
        std::cout << i << " ";
    std::cout << std::endl;

    return 0;
}

```

练习 1.11 的循环版本：

```
#include <iostream>
```

```
int main()
```

```

{
    std::cout << "请输入两个数";
    std::cout << std::endl;
    int v1, v2;
    std::cin >> v1 >> v2;
    if (v1 > v2) // 由大至小打印
        for (; v1 >= v2; v1--)
            std::cout << v1 << " ";
    else // 由小至大打印
        for (; v1 <= v2; v1++)
            std::cout << v1 << " ";
    std::cout << std::endl;
    return 0;
}

```

**练习 1.14:** 对比 `for` 循环和 `while` 循环，两种形式的优缺点各是什么？

**【解答】**

在循环次数已知的情况下，`for` 循环的形式显然更为简洁。

而循环次数无法预知时，用 `while` 循环实现更适合。用特定条件控制循环是否执行，循环体中执行的语句可能导致循环判定条件发生变化

**练习 1.15:** 编写程序，包含第 14 页“再探编译”中讨论的常见错误。熟悉编译器生成的错误信息。

**【出题思路】**

继续熟悉编译器对不同错误给出的信息。

**【解答】**

对于复杂程序中的错误，编译器给出的错误信息很可能无法对应到真正的错误位置并给出准确的错误原因。这是很正常的，因为某些时候我们人类都无法准确判断程序员到底犯了什么错误，在当前人工智能技术发展水平下，要求编译器有超越人类的智能是不现实的。

而且，不同的编译器对同一个程序给出的错误信息有可能是有很大差别的。一方面是因为如前所述，很多时候并不存在“唯一正确”的错误原因，编译器（甚至我们人类也是）只能给出它认为最有可能的错误原因；另一方面，不同编译器对同样的错误原因也可能有自己不同的解释方式。

因此，使用几种不同的编译器，编译一些错误的程序，观察编译器给出的错误信息。对今后在大型软件中查找、修改编译错误是很有帮助的。

**练习 1.16:** 编写程序，从 `cin` 读取一组数，输出其和。

**【出题思路】**

练习不定次数的循环，以及输入流结束判断。

**【解答】**

```
#include <iostream>
int main()
{
    int sum = 0, value = 0;
    std::cout << "请输入一些数, 按 Ctrl+Z 表示结束"
    << std::endl;
    for (; std::cin >> value;)
        sum += value;
    std::cout << "读入的数之和为" <<
    sum << std::endl;
    return 0;
}
```

显然, 对于循环次数无法预知的情况, for 循环比 while 循环稍累赘一些。

**练习 1.17:** 如果输入的所有值都是相等的, 本节的程序会输出什么? 如果没有重复值, 输出又会是怎样的?

**【出题思路】**

练习程序分析和手工执行程序。

**【解答】**

如果输入的所有值都相等, 则 while 循环中的 else 分支永远不会执行, 直到输入结束, while 循环退出, 循环后的输出语句打印这唯一的一个值和它出现的次数。

若没有重复值, 则 while 循环中的 if 语句的真值分支永远不会执行, 每读入一个值, 都会进入 else 分支, 打印它的值和出现次数 1。输入结束后, while 循环退出, 循环后的输出语句打印最后一个值和出现次数 1。

**练习 1.18:** 编译并运行本节的程序, 给它输入全都相等的值。再次运行程序, 输入没有重复的值。

**【解答】**

输入:

1 1 1 1 1

程序输出:

1 occurs 5 times

输入:

1 2 3 4 5

程序输出:

1 occurs 1 times

2 occurs 1 times

3 occurs 1 times

4 occurs 1 times

5 occurs 1 times

注意，不要忘了用 `Ctrl+Z` 表示输入结束。

**练习 1.19：**修改你为 1.4.1 节练习 1.10（第 11 页）所编写的程序（打印一个范围内的数），使其能处理用户输入的第一个数比第二个数小的情况。

#### 【解答】

练习 1.10 的解答已经包含了此功能。

**练习 1.20：**在网站 <http://www.informit.com/title/0321714113> 上，第 1 章的代码目录中包含了头文件 `Sales_item.h`。将它拷贝到你自己的工作目录中。用它编写一个程序，读取一组书籍销售记录，将每条记录打印到标准输出上。

#### 【出题思路】

练习如何使用其他人定义的类来创建、使用对象。

#### 【解答】

```
#include <iostream>
#include "Sales_item.h"
int main()
{
    Sales_item book;
    std::cout << "请输入销售记录: "
    << std::endl;
    while (std::cin >> book) {
        std::cout << "ISBN、售出本数、销售额和平均售价为 "
        << book << std::endl;
    }
    return 0;
}
```

**练习 1.21：**编写程序，读取两个 ISBN 相同的 `Sales_item` 对象，输出它们的和。

#### 【出题思路】

类对象的更复杂的使用，利用类接口进行运算。

#### 【解答】

```
#include <iostream>
#include "Sales_item.h"
int main()
{
    Sales_item trans1, trans2;
    std::cout << "请输入两条 ISBN 相同的销售记录: "
    << std::endl;
    std::cin >> trans1 >> trans2;
    if (compareIsbn(trans1, trans2))
        std::cout << "汇总信息: ISBN、售出本数、销售额和平均售价为 "
        << trans1 + trans2 << std::endl;
    else
```

```

    std::cout << "两条销售记录的 ISBN 不同" << std::endl;
    return 0;
}

```

**练习 1.22:** 编写程序，读取多个具有相同 ISBN 的销售记录，输出所有记录的和。

### 【出题思路】

练习在处理数据流的过程中“状态”（是否是相同的 ISBN）的保存和变迁。

### 【解答】

```

#include <iostream>
#include "Sales_item.h"

int main()
{
    Sales_item total, trans;
    std::cout << "请输入几条 ISBN 相同的销售记录："
    << std::endl;
    if (std::cin >> total) {
        while (std::cin >> trans)
            if (compareIsbn(total, trans)) // ISBN 相同
                total = total + trans;
            else { // ISBN 不同
                std::cout << "ISBN 不同" << std::endl;
                return -1;
            }
        std::cout << "汇总信息：ISBN、售出本数、销售额和平均售价为 "
        << total << std::endl;
    }
    else {
        std::cout << "没有数据" << std::endl;
        return -1;
    }
    return 0;
}

```

**练习 1.23:** 编写程序，读取多条销售记录，并统计每个 ISBN（每本书）有几条销售记录。

### 【解答】

```

#include <iostream>
#include "Sales_item.h"

int main()
{
    Sales_item trans1, trans2;
    int num = 1;
    std::cout << "请输入若干销售记录："
    << std::endl;
    if (std::cin >> trans1) {
        while (std::cin >> trans2)

```

```
if (compareIsbn(trans1, trans2)) // ISBN 相同
    num++;
else { // ISBN 不同
    std::cout << trans1.isbn() << "共有"
    << num << "条销售记录" << std::endl;
    trans1 = trans2;
    num = 1;
}
std::cout << trans1.isbn() << "共有"
<< num << "条销售记录" << std::endl;
}
else {
    std::cout << "没有数据" << std::endl;
    return -1;
}
return 0;
}
```

**练习 1.24：**输入表示多个 ISBN 的多条销售记录来测试上一个程序，每个 ISBN 的记录应该聚在一起。

#### 【解答】

在网站 <http://www.informit.com/title/0321714113> 上，第 1 章的代码目录中包含了一些数据文件，可以将这些文件重定向到此程序进行测试，也可自己创建销售记录文件进行测试。

**练习 1.25：**借助网站上的 Sales\_item.h 头文件，编译并运行本节给出的书店程序。

#### 【出题思路】

练习编译、运行稍大规模的程序。

#### 【解答】

在网站 <http://www.informit.com/title/0321714113> 上下载 Sales\_item.h 头文件，编译书店程序即可。