普通高等教育“十二五”规划教材

C++

程序设计实践教程

主　编　王　芳　张晓如
副主编　华　伟　祁云嵩　束　鑫

科学出版社

普通高等教育“十二五”规划教材

C++程序设计实践教程

主　编　王　芳　张晓如
副主编　华　伟　祁云嵩　束　鑫

科学出版社
北京

内 容 简 介

本书是将C++程序设计的基础知识和实践操作相结合的实践教材，以满足不同层次的教学需求。全书共6章，内容包括课程实践概述、课程实践预备知识、常用算法介绍、课程实践示例、课程实践题选，以及课程实践报告。书后附有全国计算机等级考试模拟试卷。

本书可以作为普通高等学校C++上机实践课程教材使用，还可供计算机爱好者阅读参考。

图书在版编目(CIP)数据

C++程序设计实践教程 / 王芳，张晓如主编. —北京：科学出版社，2013

普通高等教育“十二五”规划教材

ISBN 978-7-03-038357-0

I. ①C… II. ①王… ②张… III. ①C语言–程序设计–高等学校–教材 IV. ①TP312

中国版本图书馆CIP数据核字(2013)第192458号

责任编辑：相 凌 高 远 / 责任校对：刘小梅
责任印制：阎 磊 / 封面设计：华路天然工作室

科 学 出 版 社出版
北京东黄城根北街16号
邮政编码：100717
http://www.sciencep.com

北京通州皇家印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2013年8月第 一 版 开本：787×1092 1/16
2013年8月第一次印刷 印张：9 3/4
字数：231 000

定价：25.00元

(如有印装质量问题，我社负责调换)

前　言

计算机程序设计是一门实践性很强的课程，学习程序设计最重要的环节就是上机实践，要提高 C++程序设计的能力，需要做大量的上机实践题。为了配合课程教学，我们编写了本书，旨在帮助学习程序设计的学生，从看懂教科书，到具有一定的编程能力，使学生在上机实践的过程中不断进步，体验成功的乐趣。

本书力求编撰成一本提高学生程序设计能力、指导 C++程序设计课程实践的教材，尝试解决学生看书听课时对所学内容可以基本理解，上机编程时却无从下手的问题。全书共 6 章，第 1 章和第 6 章是课程实践的过程和要求以及实践报告的书写；第 2 章和第 3 章提炼了程序设计中算法的知识和常用算法以及程序调试中的常见错误；第 4 章和第 5 章分层次精心设计了一系列实践题目，有基础题、提高题和拓展题 3 个层次，共 75 题，通过从题目说明、分析、设计和测试几方面进行例题详解，引导学生在程序设计时学会思考、运用所学的知识进行编程。此外，书后还附有课程实践报告封面格式样例和计算机等级考试模拟试卷及参考答案，供学生在学过 C++程序设计后对自己的学习情况进行测试。

本书既可与同时出版的《C++程序设计教程》及《C++程序设计习题与实验教程》配合使用，也可独立使用。

在本书的编写过程中，参考了已经出版的书籍和网络资料，我们对这些书籍的作者以及网络资料的提供者表示由衷的感谢。此外，编写工作得到了学校及学院领导的关心和大力支持。教研室各位老师积极参与，提出了许多宝贵的意见，在此一并表示感谢。

感谢读者选择使用本书，欢迎您提出批评和修改意见，我们将不胜感激。

编　者

2013 年 6 月

目　录

第 1 章　课程实践概述

1.1　课程实践的性质与目的

“C++课程实践”是在学习了“计算机程序设计语言 C++”课程后进行的实践教学环节，该环节为学生提供一个既动手又动脑，独立实践的机会，使学生将课本上的理论知识和实践有机结合起来，有利于巩固、提高和融合所学的理论知识，提高学生运用所学知识解决实际问题的能力。课程实践主要目的如下。

(1) 进一步培养学生程序设计的思想，加深对 C++语言的理解。

(2) 针对 C++中的重点和难点内容进行训练，强调良好的程序设计风格。

(3) 掌握程序设计的常用算法。

(4) 进一步熟悉 C++的编程技巧和上机调试程序的方法。

(5) 初步掌握可视化编程的方法，进行界面设计。

1.2　课程实践的要求

首先，要求学生仔细阅读本书，认真完成程序设计实践的要求。其次，要发挥自主学习的能力，合理安排课程实践的时间，并在课程实践过程中不断检查自己的计划完成情况。根据系统的功能要求，学生必须在教师的指导下认真完成应用程序的设计。具体要求如下。

(1) 系统功能模块分析、控制模块分析正确。

(2) 系统设计要实用。

(3) 编程简练、功能全面。

(4) 说明书、流程图要清晰。

1.3　课程实践的选题原则

课程实践题目从第 5 章中选择，只选基础题应不少于 6 题，提高题应不少于 3 题，创新性题目可以只选择 1 题；也可以基础题和提高题组合选题，例如，基础题选 4 题，提高题选 1 题；还可以自行选择感兴趣的题目(需经指导老师审定)。对于创新题可组成团队开发，但应制定详细的项目分工说明。改错题为选做题型。

1.4 课程实践的基本步骤

课程实践是一个从分析到设计，再到总结的过程，具体工作可按照如下步骤依次进行。

(1)确定问题要求，充分分析和理解问题本身，给出解决方案框架。

(2)在确定解决方案框架过程中，考虑怎样使程序结构清晰、合理、简单和易于调试，并确定每个函数的简单功能，以及函数之间的调用关系。

(3)详细设计和编码。确定算法的主要流程，在此基础上进行代码设计。

(4)上机前编写程序与检查。可用两种方法检查程序，有效提高调试效率。方法一，用一组测试数据手工执行程序；方法二，通过阅读或给别人讲解自己的程序，深入全面地理解程序中的逻辑关系，将程序中的明显错误事先排除。

(5)上机调试程序。

(6)完成课程实践报告。

1.5 组织与管理

良好的组织是课程实践质量的重要保证，在课程实践期间，主要组织工作安排如下：每人独立完成所选任务；创新性题目可 2~3 人组成小组，小组应指定一名组长；班长负责考勤；指导教师负责指导学生。

1. 时间及地点安排

本课程实践按照教学要求在 1 周(按 5 天计算)内完成，每天至少上机 3~4 小时进行程序调试，总上机时间不少于 16 个小时。题目布置后，所有同学应提前开始查找资料，做好准备。具体时间安排如下。

(1)分析设计准备阶段(前一周的周六、周日)。

(2)编程调试阶段(周一~周四)。

(3)总结及书写课程实践报告阶段(周五)。

(4)检查验收阶段(时间由指导老师确定)。

验收地点：机房。

2. 考核评价

根据学生完成情况，结合所选题目的难度及分析解决问题的能力和创新精神，确定成绩等级。

考核标准包括如下几点。

(1)所设计程序的正确性、通用性，全面完成题目的要求(占总成绩的 60%)。

(2)课程实践报告(占总成绩的 20%)。按照所选题型，撰写实践报告。报告包括系统设计要求，设计思路，系统功能模块图，系统流程图，类的层次图(包括类成员列表)，调试过程，关键程序代码，程序设计实践总结等，最后附源程序代码。不符合以上要求者，本次实践成绩不及格。

(3) 平时考勤(占总成绩的 20%)。

提交材料包括如下几点。

(1) 源程序。按照课程实践的要求所选题目的所有源程序。

(2) 程序的说明文件(保存在文本文档中)。在说明文档中应该写明上交程序所在的目录，以及主程序文件名，如果需要安装，要有程序的安装使用说明等。

(3) 课程实践报告(保存在 Word 文档中)。文档的文件名要求按照“姓名-学号-实践报告”起名，如文件名为“张三-1140308112-实践报告.doc”。

注意事项：程序及课程实践报告的电子稿，发送至指导老师的邮箱(由教师提供)；且每位学生需交一份课程实践报告的打印稿。

第 2 章　课程实践预备知识

2.1　算法与算法分析

一般情况下程序员在进行程序设计时，应该考虑两个方面的问题。

(1) 对数据的描述，即数据的类型和数据的组织形式。

(2) 对操作的描述，即操作步骤。

因此，著名的瑞士计算机科学家尼古拉斯·沃斯(N.Wirth)教授提出一个公式：

程序=数据结构+算法

其中，数据结构解决数据的组织形式问题，而算法即为操作步骤，它需要依靠程序来完成功能。可见，算法在程序设计中的重要性，可以说它是一个程序的核心。

2.1.1　算法的概念

简单地说，算法就是解决问题的方法与步骤。例如，发送一封电子邮件，首先要登录网络，再根据你的邮箱地址登录电子邮件服务器，然后填写邮件内容、接收方地址，最后发送。这些步骤应该按照一定的顺序执行，否则无法实现发送邮件的功能。这里发送邮件的步骤就是一个算法。

一个算法具有如下 5 个重要特性。

(1) 有穷性。一个算法的步骤是有限的，在执行有穷个步骤后结束，并且，每一个步骤都可在有穷时间内完成。

(2) 确定性。即无二义性，算法的每一个步骤都必须有确切的含义。

(3) 可行性。算法中描述的操作，都是可以执行的，并得到确定的结果。

(4) 输入。执行算法时从外部得到的必要信息，即算法的输入，一个算法有零个或多个输入。

(5) 输出。一个算法得到的结果就是算法的输出，一个算法有 1 个或多个输出，没有输出的算法是没有意义的。

【例 2-1】写出求 1+3+5+7+9 的方法步骤。

步骤如下。

第一步：先求 1+3，得到结果 4。

第二步：将第一步得到的结果 4 再加 5，得到结果 9。

第三步：将第二步得到的结果 9 再加 7，得到结果 16。

第四步：将第三步得到的结果 16 再加 9，得到最后的结果 25。

这样的算法是正确的，但若求 1+3+5+7+9+…+999，上述算法显然不可取。可以将算法进行改进。

设两个变量，一个变量 i 表示要相加的数，另一个变量 s 保存累加和，用循环求结果。改写的算法如下。

第一步：使变量 i 的值为 3。

第二步：使变量 s 的值为 1。

第三步：使 i+s，将累加和放在 s 中。

第四步：使 i 的值加 2。

第五步：如果 i 的值小于等于 999，返回到第三步，再顺序执行其后的步骤，直到 i 的值大于 9，算法结束。最后变量 s 中的值即 1+3+5+7+9+…+999 的结果。

可见，一个问题的解决经常有多种不同的算法，为了提高解题的效率，选择算法时，不仅要保证算法正确，还要考虑算法的质量。

2.1.2　算法的表示

描述算法的工具很多，除了例 2-1 所用的自然语言描述外，还有流程图描述、N-S 结构图描述、伪代码描述、程序设计语言描述等。下面简单地介绍几种常用工具。

1. 自然语言描述法

自然语言描述方法是用人们日常使用的语言，如中文、英文等来描述算法的方法。这种方法主要用在较简单问题上。

【例 2-2】设计一个求 s=1+2+…+100 的算法，并用自然语言描述出来。

算法设计如下。

①设 s=0，i=1；

②如果 i<=100 执行③，否则转向⑤；

③执行 s=s+i；

④计算 i=i+1，返回②；

⑤输出 s 的值。

用自然语言描述算法通俗，利于理解。但用于描述含分支、循环等复杂问题时会显得较繁琐，且容易出现二义性。

2. 流程图描述法

流程图描述法是使用几何图形的图框，代表各种不同的操作或算法运行走向的一种图示描述方法。相对于自然语言来说更简洁直观。美国国家标准协会(American National Standard Institute，ANSI)规定了一些常用的流程图符号，如图 2-1 所示。

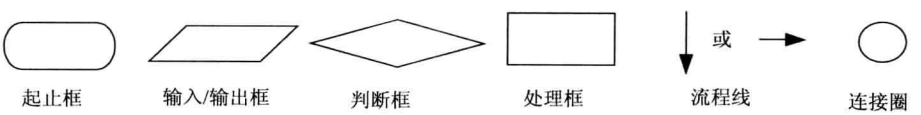
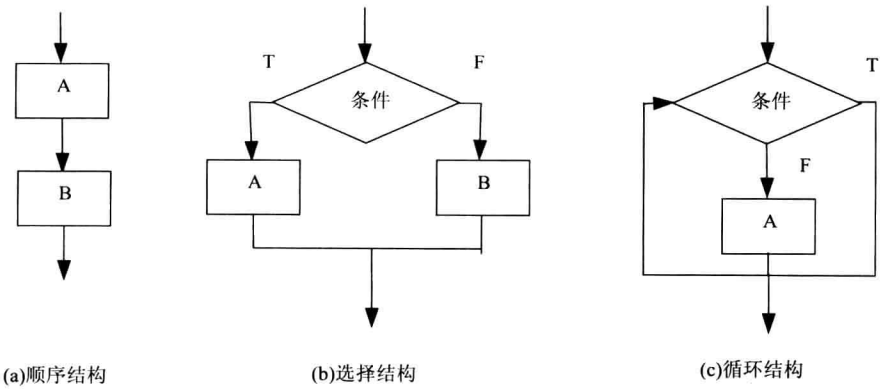


图 2-1　流程图常用表示符号

其中，“起止框”用在流程图的开始和结束位置，标志算法的开始和结束；“判断框”是对一个给定的条件进行判断，根据条件是否成立来决定如何执行其后的操作。画流程图时还可以使用圆圈，表示流向本流程图外某个地方的出口点，或表示从流程图外的某个地方进入的入口点。图 2-2 是程序的 3 种结构流程图。



(a)顺序结构　　(b)选择结构　　(c)循环结构

图 2-2　3 种程序结构流程图

【例 2-3】画出求 s=1+2+…+100 算法的流程图描述。

本题算法流程如图 2-3 所示。

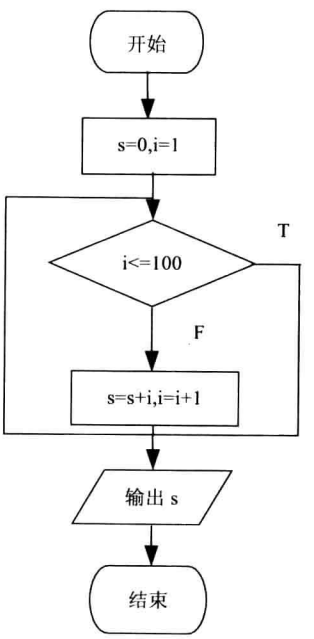


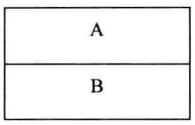
图 2-3　求 s=1+2+…+100 算法的流程图

流程图虽然直观、易懂，但由于可以根据需要不受限制地使用流程线，可能造成设计出的算法结构性不好，从而编写出来的程序结构性不好。

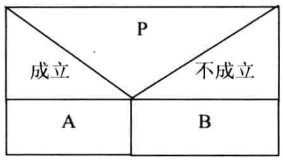
3. N-S 结构图描述法

N-S 结构图是在 1973 年由美国学者 I.Nassi 和 B.Shneiderman 提出的一种新型流程图，又称为 N-S 结构化流程图，适用于结构化程序设计。

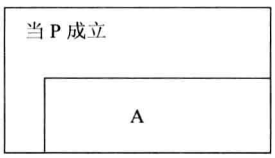
用 N-S 结构图表示 3 种程序结构如图 2-4 所示。



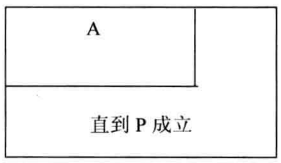
(a)N-S 顺序结构



(b)N-S 选择结构



(c)N-S 当型循环结构



(d)N-S 直到型循环结构

图 2-4　3 种程序结构的 N-S 结构图表示

【例 2-4】将例 2-3 的流程图用 N-S 结构图表示。

N-S 结构图如图 2-5 所示。

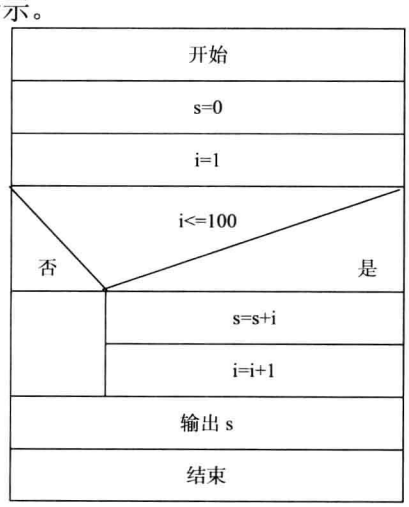


图 2-5　例 2-3 的 N-S 结构图表示

N-S 结构图省去了流程图中的流程线，使得图形更直观紧凑，并且结构性强，但修改较困难。

4. 其他表示法

自然语言、流程图及 N-S 结构图表示法不需要掌握相应计算机语言的语法规则，其

共同点是简单、直观、易读、逻辑关系清楚，但存在结构不清，画起来较费事、修改较困难的问题。同时，它们又与源程序差异较大，不利于较快地转化成源程序。因此，经常还会用其他一些方法作为描述算法的工具。

(1)伪代码表示。伪代码又称为虚拟代码，它使用计算机语言和自然语言相结合来描述算法，是介于自然语言与计算机语言之间的一种用文字和符号相结合的算法描述工具，形式上跟计算机语言比较接近，但又没有严格的语法规则限制，通常借助某种高级语言的控制结构进行描述，中间的操作可以用自然语言，也可以用程序设计语言描述。其特点是结构清晰、代码简单、可读性好，并且由于类似于自然语言，不用拘泥于具体的实现，因此比画流程图等更省时省力，且更容易转化为源程序。

(2)PAD。PAD(Problem Analysis Diagram)又称问题分析图，1973 年由日本日立公司提出。它用二维树形结构的图来表示程序的控制流，这种图翻译成程序代码比较容易。图 2-6 给出了 PAD 的基本符号。

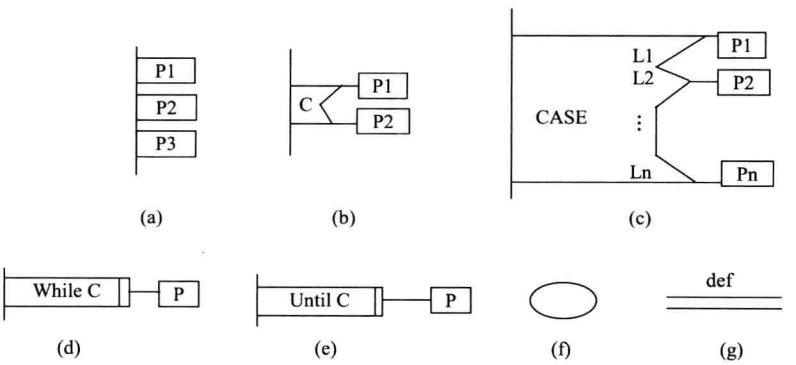


图 2-6 PAD 的基本符号

(a)为顺序结构；(b)为选择结构；(c)为多分支选择结构；(d)为当型循环；(e)为直到型循环；(f)为语句标号；(g)为定义

在 PAD 中，最左边的竖线是程序的主线，即第 1 层控制结构。随着程序层次的增加，PAD 逐渐向右延伸，每增加一个层次，图形向右扩展一条竖线。PAD 中竖线的总条数就是程序的层次数。由于每种控制语句都有一个图形符号与之对应，显然将 PAD 转换成与之相对应的高级程序语言比较容易。

2.1.3 算法分析

通常对于同一个问题，有不同的解决方法，即可以有不同的算法。例如，对 N 个数进行排序，可以用选择法排序，也可以用冒泡排序、归并排序等。那么到底应该选择哪一种算法呢？

一般而言，算法的选择在保证正确的前提下，还应考虑执行一个算法所要占用的计算机资源是多少，包括时间资源和空间资源两方面，这两方面都与问题的规模有关。例如，对 10 个数排序与对 1000 个数排序所花费的运行时间和存储空间显然是有差别的。

1. 算法的时间复杂度

一个算法执行时所耗费的时间需要由依据算法所编写的程序在计算机上运行时所花费的时间来度量。同一个算法用不同的语言编写，用不同的编译系统进行编译，或在不同的计算机上运行，所消耗的时间均不相同。计算算法的执行时间不考虑这些与计算机软件、硬件相关的因素，只考虑程序中每条语句的执行时间之和。

【例 2-5】求两个 N 阶方阵乘积的算法的时间耗费。

```
#define n 4
void matrix(int a[][n], int b[][n], int c[][n])
{
    int i,j,k;
    for(i=0;i<n;i++)                                    //A  n+1
        for(j=0;j<n;j++)                                //B  n(n+1)
        {   c[i][j]=0;                                  //C   n^2
            for(k=0;k<n;k++)                            //D   n^2(n+1)
                c[i][j]+=a[i][k]*b[k][j];               //E    n^3
        }
}
```

注释行列出的是各语句执行的次数。其中 A 行的循环变量 i 从 0 增加到 n，当 i≥n 时，循环才会终止，所以执行次数为 n+1，但循环体只执行 n 次，所以 B 行执行次数为 n(n+1)。同理，语句 C 行、D 行、E 行的执行次数为 n^2、$n^2(n+1)$ 和 n^3。该算法所有语句执行次数之和，即时间耗费为 $2n^3+3n^2+2n+1$。

一个算法的时间复杂度(Time Complexity)是执行该算法时所耗费的时间，它是问题规模 n 的函数，记作 T(n)。若有某个辅助函数 f(n)，使得当 n 趋近于无穷大时，T(n)/f(n) 的极限值为不等于零的常数，则称 f(n) 是 T(n) 的同数量级函数。记作

$$T(n)=O(f(n))$$

称 O(f(n)) 为算法的渐进时间复杂度，简称时间复杂度。如：

$$\lim_{n\to\infty} T(n)/n^3 = \lim_{n\to\infty}(2n^3+3n^2+2n+1)/n^3 = 2$$

这表明，当 n 充分大时，T(n) 和 n^3 是同阶的，可记作 $T(n)=O(n^3)$。即例 2-2 算法的时间复杂度为 $O(n^3)$。

【例 2-6】计算以下算法的时间复杂度。

```
#define n 10
void m(void)
{
    int x=0,y=0;int i,j,k;
```

```
    for(i=1;i<=n;i++)   x++;
    for(j=1;j<=n;j++)
        for(k=1;k<=n;k++)
          y++;
}
```

本算法时间复杂度的计算可以类似于例 2-2，依据例 2-2 得到 $T(n)=O(n^2)$。但一般情况下，对循环语句只需考虑循环体中语句执行的次数，忽略语句中循环变量加 1、循环结束判断等因素。因此，以上算法中执行次数最多的语句是 y++，计算得 $f(n)=n^2$，所以 $T(n)=O(n^2)$。

按数量级递增排列，常见的算法时间复杂度有常数阶 $O(1)$、对数阶 $O(\log_2 n)$、线性阶 $O(n)$、线性对数阶 $O(n\log_2 n)$、平方阶 $O(n^2)$、立方阶 $O(n^3)$、k 次方阶 $O(n^k)$ 以及指数阶 $O(2^n)$ 等。随着问题规模 n 的不断增大，上述时间复杂度不断增大，算法的执行效率越低。如时间复杂度为指数阶的算法效率很低，当 n 稍大时，这样的算法就没有实际意义了。

2. 算法的空间复杂度

空间复杂度(space complexity)是指一个算法实现时所占用的系统存储空间的大小，记作

$$S(n)=O(f(n))$$

表示随着问题规模 n 的增大，算法运行时所需系统存储空间的增长率与 $f(n)$ 的增长率相同。

一个根据算法写好的程序所占用的存储空间，包括存储程序本身所占用的空间，算法的输入数据所占用的空间和程序在运行过程中临时占用的存储空间。存储程序本身所占用的空间与程序代码的长短有关；算法的输入数据所占用的空间主要取决于问题本身，与算法无关。一般算法的空间复杂度只考虑程序在运行过程中所占存储空间的大小。例如，递归算法每次递归时都要存储返回信息，所以算法的空间复杂度是 $O(n)$，这里 n 为问题的规模。

对于同一个问题，人们都希望选用一个运行时间短、所占存储空间小，并且易于理解、编码、调试的算法。然而，实际的算法可能做不到十全十美，因为这些要求有时会相互牵制。因此，选择算法时，应根据实际情况有所侧重。

2.2 程序调试中的常见错误

程序调试中的常见错误一般为语法错误和逻辑错误，下面对编程时经常出现的错误进行了归纳，并且给出改正方法。

2.2.1 语法错误

所谓语法错误是指程序中包含了违反 C++语法规则的代码语句。在编译程序时将自

动检测出这种类型的错误。相比于逻辑错误，语法错误更容易被发现和解决。常见的语法错误有变量未定义、括号不匹配、遗漏了分号等。

使用 Visual C++ 6.0 进行程序编译时，语法错误会在“输出”窗口中显示。双击“输出”窗口中“错误”行，光标就会停留在“编辑”窗口中该错误所在的行，同时有一“箭头”指向该行。下面给出常见语法错误的种类及改正方法。

1. “头文件”错误

主要表现为程序中使用了“库函数”而没有相应的“头文件”，或“库函数”和“头文件”不一致。如程序中使用了函数“abort( )”，而没有写编译预处理命令“#include<stdlib.h>”，或该命令书写错误，此时，显示的错误在调用函数“abort( );”行(显示没有定义该函数)，此时只要将正确的编译预处理命令写出即可。

2. 宏定义错误

如：

```
#define PAI 3.14159;          //A
flaot r=2;
cout<<"圆的面积为: "<<PAI*r*r<<'\n';  //B
```

显示的错误在宏使用处(B 行)，而不在宏定义处(A 行)。

3. 变量使用错误

包括使用了非法标识符，如“C++”；变量作用域错误，如使用了不可见的变量或变量重复定义；将常量当做变量操作；定义数组时没有确定数组的大小等。

【例 2-7】求数组 a 偶数下标和奇数下标元素的和。指出下列程序中的错误。

```
#include <iostream.h>
void main(void)
{
    int a[10]={1,2,3,4,5,6,7,8,9,10};
    for(int i=0;i<10;i+=2){static int s1;s1+=a[i];}  //A
    for(int i=1;i<10;i+=2){static int s2;s2+=a[i];}  //B
    cout<<"s1="<<s1<<"\ts2="<<s2<<'\n';   //C
}
```

本例中 A 行和 B 行的循环变量 i 处在同一个作用域内，不能重复定义，故要去掉 B 行的 int；变量 s1 和 s2 在循环体中定义，C 行使用了不可见的变量 s1 和 s2。

4. 函数使用错误

包括返回值类型错误，如返回值类型和函数体内 return 语句带回的值类型不一致或返回值类型和 return 语句使用形式不一致；函数名错误，如使用非法标识符作函数名；参数错误，如定义函数时没有列出所有参数的类型或函数体内所用形参名和定义不一致；函数调用错误；函数原型说明错误，如函数原型说明和函数定义不一致或函数原型说明未以分号结束；在类体外定义类的成员函数时未使用作用域运算符“::”说明该函数属于