HZ BOOKS

高等院校计算机教材系列

面向对象程序设计

C++版

刘振安 主编

机械工业出版社
China Machine Press

为教师提供教学课件

面向对象程序设计

C++版

高等院校计算机教材系列

面向对象程序设计
C++版

刘振安 主编

机械工业出版社
China Machine Press

本书系统介绍面向对象程序设计方法，并用C++语言描述了具体实现方法。本书假设读者具有基本的面向过程编程知识，所以直接通过使用对象和STL库，建立对象行为及实例的概念，并强调C++中重要的概念和编程思想。本书密切结合案例建立对象和类的概念，突出应用，旨在提高使用面向对象方法解决实际问题的能力，并进一步强调多文件编程方法，以便为可视化编程打下基础。

本书取材新颖、结构合理、概念清楚、实用性强，易于教学，适合作为高等院校的教材，也可以作为培训班教材、自学教材及工程技术人员的参考书。

图书在版编目（CIP）数据

面向对象程序设计：C++版/刘振安主编. －北京：机械工业出版社，2006.10
（高等院校计算机教材系列）
ISBN 7-111-19714-3

Ⅰ.面…　Ⅱ.刘…　Ⅲ.①面向对象语言－程序设计－高等学校－教材　②C语言－程序设计－高等学校－教材　Ⅳ.TP312

中国版本图书馆CIP数据核字（2006）第089994号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：朱　劼
北京诚信伟业印刷有限公司印刷·新华书店北京发行所发行
2006年10月第1版第1次印刷
184mm×260mm·18.25印张
定价：28.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

前　言

计算机科学的每一步发展几乎都在软件设计和程序设计语言方面得到充分体现。软件是一个发展的概念，随着软件开发规模的扩大和开发方式的变化，人们开始将程序设计语言作为一门科学来对待。程序设计方法和技术的发展不仅直接导致了一大批风格各异的程序设计语言的诞生，而且对计算机理论、硬件、软件以及计算机应用技术等都产生了深远的影响。

目前，大多数语言类教材都是把重点放在基本词法、语法和简单的程序上，学完教材之后，很难编出实用的程序。本书是在自 1994 年以来开设的几门课程的基础上，对内容进行合理组合与取舍，并力求反映学科发展，展现它们的最新特征。本书把重点放在程序设计方法上，为了方便学习，每一章均配有相应的实验和习题。在写作中力求达到取材新颖、结构合理、概念清楚、语言简洁、通俗易懂、实用性强、易于教学的目标，所以既适合作为高等院校的教材，也适合作为培训班教材、自学教材及工程技术人员的参考书。如果配合机械工业出版社已经出版的课程设计，则学习效果更佳。

本书假设读者学过 C 语言或者C++ 语言的面向过程部分的知识，所以直接通过使用对象和 STL 库，建立对象行为及实例的概念。在介绍面向对象程序设计时，密切结合案例，突出应用，提高解决实际问题的能力。

本书将进一步强调多文件编程方法，以便为可视化编程打下基础。第 10 章的综合实验也可以作为可视化编程的课程设计。其实，本书的门槛并不高，只要自学一点 C 或C++ 面向过程部分的知识，就可以很快入门。

本书共有 11 章。第 1 章介绍C++ 语言的新面貌，包括动态内存分配、引用、指针、使用 const、范型算法用于普通数组及数据的简单输出输入格式等。第 2 章是从结构到类的演变，通过实例简要说明结构如何向类变化，从而引入对象和类的知识。第 3 章阐述函数和函数模板，除介绍函数调用、递归调用以及函数调用中的参数替换和返回值等问题之外，还将结合软件编程技术的发展，讨论函数指针、内联函数、函数重载、函数模板及算法等知识，但不涉及类的构造函数和成员函数。第 4 章将建立类和对象的概念，将重点放在C++ 中定义类、建立和使用对象的基本方法及面向对象编程的文件规范上。第 5 章介绍特殊函数和成员，包括静态成员、友元、const 对象、volatile 对象、数组、类成员指针和转换函数。第 6 章是面向对象编程实例，通过求解一元二次方程的根和出圈游戏两个程序，演示如何使用面向对象的思想来解决实际问题。第 7 章讨论继承和派生，涉及公有继承的赋值兼容性规则，给出包含及派生两种实现方法的实例以加深对这两种方法的理解。第 8 章介绍类模板与向量容器，将简要概述使用向量容器类的基础知识，并引入 STL 库和范型算法的基础知识。第 9 章介绍多态性、虚函数及其他类型，重点介绍运行时的多态性，并结合图解和程序实例帮助读者更好地理解多态性，也简要介绍结构、联合和枚举等。第 10 章涉及运算符重载及流类库，将简要介绍运算符重载的基础知识、流类库的概念及使用流类库进行文件存取的基本方法。第 11 章是面向对象专题的讨论，主要涉及在实际应用中需要考虑的几个专题，目的是加深对知识的理解并锻炼解决实际问题的能力。

中国科学技术大学软件学院院长陈国良院士，原安徽大学副校长、计算机系教授程慧霞及南京大学计算机系陈本林教授在百忙之中审阅了书稿并提出许多宝贵意见，许多使用过原教材的兄

弟院校老师也提出许多宝贵意见。本书在定稿之前，曾为多个软件工程硕士班及本科生班讲授，他们也提出了一些好的建议并验证了书中的程序及实验，特此感谢。写作中还参考了大量资料，有的收入参考文献之中，还有些没有收入其中，在此对这些作者表示感谢。

刘燕君、周军、潘剑锋、王文涛、孙忱等参加了本书的编写工作。由于时间有限，不妥之处在所难免，敬请同行及读者不吝赐教。

作者联系方式：zaliu@ ustc. edu. cn。

刘振安

于中国科学技术大学

目　录

第 1 章　C++ 语言的新面貌

C 语言是结构化程序设计语言，其程序设计特点就是函数设计。同样，C++ 语言的类也离不开函数。本章将通过一个实际的例子，介绍C++ 如何兼容面向过程设计、它的基本程序结构以及在函数设计上与 C 语言的异同。然后给出使用函数重载的实例，以便为从结构引入类的知识打下基础。最后介绍C++ 语言的动态内存分配、引用、指针使用 const、范型算法用于普通数组及数据的简单输出输入格式。

1.1　似曾相识燕归来

本节将给出一个使用结构和函数的典型的C++ 程序。通过这个程序，既复习 C 语言的知识，又引入C++ 语言的新特征，并为学习设计类打下基础。

【例 1.1】演示使用结构的示例程序。

```
/* 功能：将结构的两个域值相加，乘以 2 再减去 50* /
#include < iostream >                          //包含头文件
using namespace std;                           //使用命名空间
int result(int,int);                           //result 函数的原型声明
const int k =2;                                //定义常量
struct Point                                   //定义结构 Point
{
     int x,y;                                  //定义结构成员 x 和 y
};

int main()                                     //主程序
{
    int z(0),b(50);                            //初始化整数对象
    Point a;                                   //定义结构对象 a
    cout << "输入两个整数(以空格区分):";        //输出提示信息
    cin >>a.x >>a.y;                           //接受输入数值
    z = (a.x + a.y)*k;                         //计算结果
    z = result(z,b);                           //计算结果
    cout<< "计算结果如下:" << endl;             //输出信息并换行
    cout<< "((" <<a.x << " + " <<a.y           //输出信息
        << ")*" <<k << ") - " <<b              //输出信息
            << " = " << z                      //输出结果
        <<endl;                                //换行
        return 0;                              //主函数 main 的返回值
}                                              //主函数结束
//* * * * * * * * * * * * * * * * * * * * * * * * *
//* 函数:result                                  *
//* 参数:整型对象 a 和 b                         *
//* 返回值:整型对象                              *
//* * * * * * * * * * * * * * * * * * * * * * * * *
int result(int a,int b)
```

```
{
    a = a - b;
    return a;                                    //返回 a - b
}
```

1. 主函数造就混合型语言

用C++语言编写的程序称为C++语言源程序，简称C++程序并以“.cpp”作为文件扩展名。和C语言一样，一个C++程序必须有一个且只能有一个名为main的主函数。main并不是C++语言定义的关键字，但C++编译系统假设程序中定义有main函数。如果没有定义该函数，程序将无法执行。因为C++保留了这个面向过程的主函数，所以称之为混合型语言。

2. 灵活的注释方式

程序中以“//”开始的内容是注释，有效范围至本行结束，注释的内容在编译时不产生目标代码。所谓目标代码，就是程序可以执行的代码。

C++语言的另外一种注释形式沿用了C语言的语法，即从“/*”开始，直到“*/”结束，它更适合注释占多行的情况。

目前使用“//”注释多行的现象越来越多，例如程序中对函数result的注释。只要不遗忘左边的注释符号“//”，就不会出错。

一个好的程序设计者应该在程序中正确使用注释来说明整个程序的功能、注意事项及有关算法等。有人认为，注释越多，程序的可读性和可维护性越好，其实不然，只在必要的地方进行注释即可。也就是说，应该添加的是程序的注释，不是对程序的说明。

3. 使用输出和输入对象

C++将数据从一个对象流向另一个对象的流动抽象为“流”。从流中获取数据的操作称为提取操作，向流中添加数据的操作称为插入操作。cin用来处理标准输入，即键盘输入。cout用来处理标准输出，即屏幕输出。它们都能自动按照正确的默认格式处理数据。语句

```
cout << " 输入两个整数 (以空格区分):";
```

的作用是输出信息“输入两个整数（以空格区分):”，用来提示用户输入两个整数。它就是C++的“语句”（statement），语句以分号结束。和自然语言中的句子一样，语句是C++程序的最小独立单元。因为cin和cout都不是C++语言本身定义的一部分，而是由C++的一套面向对象类体系（classes hierarchy）提供支持，并作为C++标准程序库（standard library）的一员。要使用这个类体系，必须先在主函数之前使用#include语句将其包含，以便让程序知道该类体系的定义。C++标准输入输出库的头文件是iostream，所以在程序中使用下述语句：

```
#include < iostream >                              //包含头文件
```

符号“<<”由连按两次“<”键产生，表示将信息输出到显示屏上，符号“>>”表示接受键盘输入，语句“cin >> a. x;”将输入赋给对象a的域x。符号“>>”和“<<”形象地表示了数据流动的方向。可以在一条语句中多次使用流的符号，也可以分为多个语句，例如可将程序中的输出改写为如下形式：

```
cout<< "((" <<a.x << " + " <<a.y;                //以分号结束
cout<< ")* " <<k << ") - " <<b;                  //以分号结束
cout<< " = " <<z;                                //以分号结束
cout<<endl;                                      //与语句“cout <<" \n";”功能相同
```

“cout<< endl;”与“cout<<" \n";”的功能都是换行，即将光标位置换到下一行的起点。同样，如果要输入a. x和a. y的值，既可将语句分成两行，也可以使用一行。例如：

```
cin >>a.x;                                       //使用两行
```

```
cin >> a.y;
```

endl 可以插在流的中间，下面的语句是将 a. x 和 a. y 分为两行输出的例子。

```
cout<< a.x << endl<< a.y << endl;
```

4. 使用命名空间

C 语言一直使用后缀“. h”标识头文件，但在例 1. 1 中没有使用后缀，原因是新的C++ 标准引入了新的标准类库的头文件载入方式，即省略“. h”。不过，这时必须同时使用下述语句：

```
using namespace std;                    //使用命名空间
```

所谓命名空间（namespace）是一种将程序库名称封装起来的方法，它提高了程序的性能和可靠性。目前无需深入了解它的含义，只要记住C++ 标准中的标准类库的变量与函数都属于命名空间 std，若要在程序中使用 cin 和 cout 这两个 iostream 类的对象，不仅要包含 iostream 头文件，还要让命名空间 std 内的名称曝光，这条语句的作用就是让命名空间中的名称曝光。

C++ 新标准就是将标准类库的头文件与一般的头文件（需要使用后缀“. h”）区分开来。当然也可以自己定义符合标准库的头文件，使用这种头文件时，也必须同时使用命名空间语句。

由以上分析可见，一般的程序都要具有如下两条语句：

```
#include < iostream >                  //包含头文件
using namespace std;                    //使用命名空间
```

当使用其他头文件时，有的不需要使用后缀“. h”，例如 < string > ，但有的仍然需要使用后缀形式，例如 < math. h > 。

5. 对象的定义及初始化

定义对象包括为它命名并为它指定数据类型。每个对象都属于某个特定的数据类型。对象名称如果设计得好，能让人直接联想到该对象的属性。

本程序使用两个整数对象 z 和 b。int 是C++ 的关键字，用来定义整数对象。可以在同一行使用如下两条语句：

```
int z = 0;int b = 50;                   //一行有多条语句，为了可读性一般不这样做
```

也可以在一个语句中定义多个对象，其间以逗号隔开，即

```
int z = 0,b = 50;
```

一般来说，即使初值只用来表示该对象尚未具有真正意义的值，也应将每个对象初始化。本程序使用了一种不同的初始化语法，称为构造函数语法。目前先不解释原因，只要记住它们的作用相同即可：

```
int z(0);                               //等同于 int z = 0;
int b(50);                              //等同于 int b = 50;
```

6. 函数和函数原型

C++ 要求所有函数都要有类型说明，主函数也不例外。本例中的 int main( ) 指出 main 函数是整数类型。函数返回值由 return 后面的表达式决定。这个表达式的值必须与声明函数的类型一致。这个程序确实不需要使用返回值，所以使用语句

```
return 0;
```

表示 main 函数结束，返回“0”值。如果函数确实不需要返回值，还可以用 void 标识，一旦使用 void 标识，函数体内就不再需要使用 return 语句。如果再使用“return 0;”，则编译出错，但可使用“return;”语句。本书以后对无需返回值的函数将使用 void 的形式且不使

用 return 语句。

C++ 函数有库函数（标准函数）和自定义函数两类，本例中的函数 result 就是自定义函数。C++ 程序使用变量的基本规则是：必须先声明，后使用，对函数调用也是如此。如果没有语句“int result(int,int);”，那么当编译主函数 main，扫描到语句

```
z=result(z,b);
```

时，尚没有见到 result 函数，这时系统就会报错。所以在主函数之前应使用语句

```
int result(int,int);                    //函数 result 的原型声明
```

对 result 函数进行原型声明。这个语句声明了 result 的函数原型，即只列出参数的数据类型。它向编译系统声明，后面有一个 result 函数，该函数有 2 个整型类型的参数，函数返回整型值。编译系统记录下调用这个函数所需要的信息，然后根据函数原型对程序中调用函数的合法性进行全面检查。因为要检查调用函数的类型是否和声明的类型一致，所以声明时不需要给出参数的变量名称。如果使用下述方式声明：

```
int result(int a,int b);
```

因为编译系统不检查参数名，所以效果一样。为了严格检查调用函数是否匹配，应该养成使用函数原型声明的好习惯。在C++ 中，每一个函数都有基本相同的形式：

```
函数类型声明  函数名（形式参数列表）
{
    变量声明
    语句部分
}
```

函数可按任何顺序出现，且可出现在一个源程序文件或多个源程序文件中。函数定义中不可缺少的部分是：

```
函数类型声明  函数名 ()
{}
```

其他部分则根据需要来确定有无。分析函数的定义形式，可以把定义分为两部分，即函数类型声明部分和函数体。

函数类型声明用来定义函数返回值的数据类型，可使用基本数据类型和自定义类型。C++ 编译系统要求必须指定类型，无返回值的函数的类型为 void。

函数体是处理需要完成功能的部分，它从花括号“{”开始，直到对应的花括号“}”为止。变量声明通常放在“{”的后面，变量声明后的是语句部分。函数体的最后是“}”，表示该函数到此结束。另外，C++ 中也常使用空的函数体函数。例如：

```
void tmpc() { }
```

这里 tmpc 是函数名。因为“{}”内没有任何可供执行的语句，所以该函数一旦被调用，就什么也不做而立即返回到调用它的函数里去，这是C++ 程序的最小函数形式。这种函数有两种用途：一是在程序开发时，为将来要设置的函数事先安排一个位置。通常是先给函数起个暂时的名字，待以后再设计这个函数。当设计正确之后，再将它改为合适的名字。另一种用途是用在继承中，基类声明一个函数作为接口，由派生类根据需要去定义它的功能。

7. const 修饰符和预处理程序

C 语言一般使用宏定义“#define”定义常量，在C++ 中，建议使用 const 代替宏定义。注意，const 是放在语句定义之前的，因此可以进行类型判别。用关键字 const 修饰的标识符是一类特殊的常量，称为符号常量，或 const 变量。const 修饰符的使用方法也很简单。事实

上，对于基本数据类型的变量，一旦加上 const 修饰符，编译器就将其视为一个常量，不再为它分配内存，并且每当在程序中遇到它时，都会用在声明时给出的初始值取代它。使用 const 可以使编译器对处理内容有更多的了解，从而允许对其进行类型检查，同时避免对常量的不必要的内存分配，以及改善程序的可读性。

常量定义也可以使用构造函数的初始化方法，即

```
const int k(2);          //定义常量并使用构造函数方法初始化
```

不过，为了强调常量的不变性，仍然沿用“=”号进行初始化。

C++ 语言仍然可以使用宏定义。无参数的宏作为常量，而带参数的宏则可以提供比函数调用更高的效率。但预处理只是进行简单的文本代替而不进行语法检查，所以会存在一些问题。例如：

```
#define BUFSIZE 100
```

这里的 BUFSIZE 只是一个名字，并不占用存储空间并且能被放在一个头文件中。在编译期间，编译器将用“100”来代替所有的 BUFSIZE。这种简单的置换常常会隐藏一些很难发现的错误，并且这种方法还存在类型问题。比如这个 BUFSIZE 究竟是整数还是浮点数？而使用 const，把值代入编译过程即可解决这些问题。和上面宏定义等效的语句如下：

```
const int BUFSIZE = 100;
```

这样就可以在任何编译器需要知道这个值的地方使用 BUFSIZE，并且编译器在编译过程中可以通过必要的计算，把一个复杂的常量表达式缩减成简单的表达式。对于某些更复杂的情况，宏定义往往不如常量来得简洁清楚，用 const 完全可以代替无参数的宏。

因为被 const 修饰的变量的值在程序中不能改变，所以在声明符号常量时，必须对符号常量进行初始化，除非这个变量是用 extern 修饰的外部变量。例如：

```
const int i = 8;
const int d;              //错误!
extern const int d;       //可以
```

const 的用处不仅仅是在常量表达式中代替宏定义。如果一个变量在生存期中的值不会改变，就应该用 const 来修饰这个变量，以提高程序的安全性。

C++ 语言的预处理程序不是C++ 编译程序的一部分，它负责分析处理几种特殊的语句，这些语句称为预处理语句。顾名思义，预处理程序对这些特殊语句的分析处理是在编译程序的其他部分之前进行的。为了与一般的 C ++ 程序语句相区别，所有预处理语句都以位于行首的符号“#”开始。预处理语句有 3 种，它们分别是宏定义、文件包含和条件编译。

C++ 预处理程序和有关语句能够帮助程序员编写易读、易改、易移植并便于调试的程序，对于模块化程序设计也提供了很大的帮助。例如语句

```
#define PI 3.14159
```

用名字 PI 来代替数字 3. 14159，又例如：

```
#define  YES  1
#define  NO   0
```

则定义 YES 和 NO 分别是 1 和 0。当然，在这些场合下，最好是使用 const 语句。

预处理程序把所有出现的、被定义的名字全部替换成对应的“字符序列”。#define 中的名字与C++ 中的标识符有相同的形式，为了区别，往往将#define 中的名字用大写字母来表示（标识符用小写字母）。这也适合 const 语句。

例 1. 1 中的语句 1 是文件包含语句，它指出一个程序把另一个指定文件的内容包含进

来。书写时，可以使用引号也可以用尖括号。例如：

```
#include"filename"
```

或者

```
#include <filename>
```

都是在程序中把文件 filename 的内容（引号或尖括号是一定要的）包含进来。

另外还要注意，用双引号和用尖括号括起文件名的含义并不一样。使用尖括号时，C++ 编译系统将首先在C++ 语言系统设定的目录中寻找包含文件，如果没有找到，就到指定的目录中去寻找，这是引用系统提供的包含文件所采用的方法。自定义的包含文件一般放在自己指定的目录中，所以在引用它们时，应采用双引号以通知C++ 编译器在用户当前目录下或指定目录下寻找包含文件。指定的目录不必位于同一个逻辑盘中。例如自定义的包含文件 myfile.h 位于 E 盘的 prog 目录中，则引用形式为

```
#include "e:\prog\myfile.h"
```

在程序设计中，文件包含语句是非常有用的。一般来说，C++ 系统中带有大量的.h 文件，用户可根据不同的需要将相应的.h 文件包含起来。标准输入输出是定义在标准库 iostream 中的，所以要同时用到如下两条语句：

```
#include <iostream>
using namespace std;
```

一般的C++ 程序都离不开这两条语句，C++ 语言的初学者也最容易遗漏这两条语句。

8. *变量对象命名及其初始化*

C++ 语言是区分字母大小写的语言，例如 name 和 Name 就代表不同的标识符。C++ 规定标识符的长度不限，在选取时不仅要保证正确性，还要考虑容易区分，不易混淆，例如数字 1 和字母 l 放在一起就不易辨认。在取名时，应使名字的含义清晰，例如使用 area 作为求面积函数的名字，因为 area 的英文含义就是面积，所以可以很容易地从名字猜出函数的功能。对于一个可读性好的程序来说，必须选择恰当的标识符，取名应统一规范，使读者一目了然。在C++ 语言中，以下划线（“_”）字符开头的标识符一般由系统内部使用，最好不要将它作为标识符的第 1 个字符。习惯上把使用宏定义的标识符用大写字母表示，例如将圆周率定义为 PI。

在内部名字中至少前 31 个字符是有效的，所以应该采用直观的名字。但名字的第 1 个字母不能是数字或运算符，也不能包含运算符。例如，下面的字符串都是不合法的标识符：

```
2a,-xy,name* ,no#,a/b,a* b[6]
```

为标识符命名时，一般可以遵循如下规律：

1）使用能代表数据类型的前缀。

2）名称与变量的作用尽量接近。

3）如果名称由多个英文单词组成，每个单词的第 1 个字母应大写。

4）由于库函数通常使用以下划线开头的名字，因此不要将这类名字用作变量名。

5）局部变量应使用比较短的名字，尤其是循环控制变量（又称循环位标）的名字。

6）外部变量应使用比较长的名字，且名字应与所代表的变量含义接近。

书中的程序一般都较短，变量的特定意义不明显，所以有时就使用 x、y、z、a、b、c 等简单的变量名。编制实用程序时建议不要这样做，应该从现在开始就养成良好的命名习惯。

简单变量对象可以在定义时明确地加上初始值，方法是在定义变量对象的后面加上赋值

运算符“=”和一个数学表达式，如

```
int x=1,y=25;
char squote='\";
long day=60*24;
```

对外部和静态变量对象只能做一次初始化工作，从概念上看应该在编译时进行。自动型和寄存器型变量对象每进入一次函数或复合语句，就被初始化一次，而且初值不限于常数，可以包含以前已定义过的值，甚至包含函数调用的合法表达式。例如：

```
void binary(int x,int n)
{
    int high=n-1;
    ...
}
```

C++中的对象就是C语言中的变量。一定要养成在声明对象的同时进行初始化的好习惯。另外，人们已经习惯于称这些对象为变量，所以在进行面向对象设计时，用户仍然喜欢称其为变量。从现实世界来看，它们确实映射着一个确定的对象。对于这些简单而基本的对象，虽然可以称其为“变量”，但为了尽快养成使用对象思考问题的习惯，建议改称为对象。

9. 程序的书写格式

C++语言的格式和C语言一样，都很自由，在一行中可以写几条语句。不过，使用恰当的格式对于充分理解C++语言非常重要。一个格式恰当的程序和一个格式不恰当的程序就像一封写得很漂亮的信和一封写得非常凌乱的信，给人的印象是大不一样的。应该使源代码易于理解，特别是容易被输入程序的程序员所理解，这有助于复杂程序的调试及修改以前输入的代码。

应使用缩进格式和必要的空行的书写风格，并使源代码具有层次性和逻辑性，以增加程序的可读性和可操作性。一般来讲，每次最好缩进5个字符的位置，按程序特性设置空行。在本书中，为了节省篇幅，有意识地减少空行甚至不留空行。读者在输入程序时，不要模仿，应注意养成良好的书写风格。

在书写程序语句时，一般应遵循如下规则：

1）括号应紧跟在函数名的后面，但在for和while后面，应用一个空格与左括号隔开以增加可读性。

2）数学运算符的左右两边各留一个空格，以与表达式区别。

3）在表示参数时，逗号后面留一个空格。

4）在for、do…while和while语句中，应合理使用缩进、一对花括号和空行。

10. 程序运行结果

输入两个整数（以空格区分）：

```
119 125 <CR>
```

计算结果如下：

```
((119+125)*2)-50=438
```

带下划线的数字“119 125”表示该数字是从键盘输入的，符号<CR>表示按键盘上的“回车”（Enter）键，其作用是通知程序，输入结束。在没有按回车键之前，可以反复修改输入值。一旦按下回车键，就表示确认输入，不能再修改。这种交互方式称为命令行交互方式。当使用这种方式与程序交互时，均以回车键作为信息结束符。以后除非特殊需要，一般

不再使用符号 <CR> 。

1.2 使用函数重载

C++ 允许为同一个函数定义几个版本，从而使一个函数名具有多种功能，这称为函数重载（详细机理见 3.6 节）。假设有一个函数 max，分别具有如下函数原型：

```
double max(double,double);          //2 个实型参数的函数原型
int max(int,int);                   //2 个整型参数的函数原型
int max(int,int,int);               //3 个整型参数的函数原型
```

只要分别为带有不同参数的 max 编制相应的函数体，就可以实现各自的功能。

【例 1.2】函数重载产生多态性的例子。

```
#include <iostream>
using namespace std;
double max(double,double);          //2 个实型参数的函数原型
int max(int,int);                   //2 个整型参数的函数原型
int max(int,int,int);               //3 个整型参数的函数原型
void main()
{
  cout <<max(2.5,17.54) << " " <<max(56,8);
  cout << " " << "max(5,9,4) = " <<max(5,9,4) <<endl;
}
double max(double m1,double m2)
{return(m1 >m2)? m1:m2;}
int max(int m1,int m2)
{return(m1 >m2)? m1:m2;}
int max(int m1,int m2,int m3)
{
    int t =max(m1,m2);
    return max(t,m3);
}
```

C++ 能够正确调用相应函数，程序输出结果如下：

```
17.54 56 max(5,9,4) =9
```

从函数原型可见，这几个 max 函数的参数类型不同，参数个数也不同。max（2.5，17.54）的参数是 double 型，所以调用 max(double,double)。max(56,8)则调用 max(int,int)。对于求三个整数中最大值的 max(5,9,4)形式，显然它必须调用 max(int,int,int)。

1.3 C++ 的基本数据类型

C++ 语言与 C 语言的基本数据类型类似，所以本节仅仅简要介绍一下C++ 语言在基本数据类型方面的特别之处及注意事项。

1.3.1 关键字

下面是部分常用关键字：

auto	const	else	goto	new	short	this	unsigned
break	continue	enum	if	operator	sizeof	throw	using
bool	default	extern	int	private	struct	true	virtual
case	delete	false	inline	protected	signed	try	void