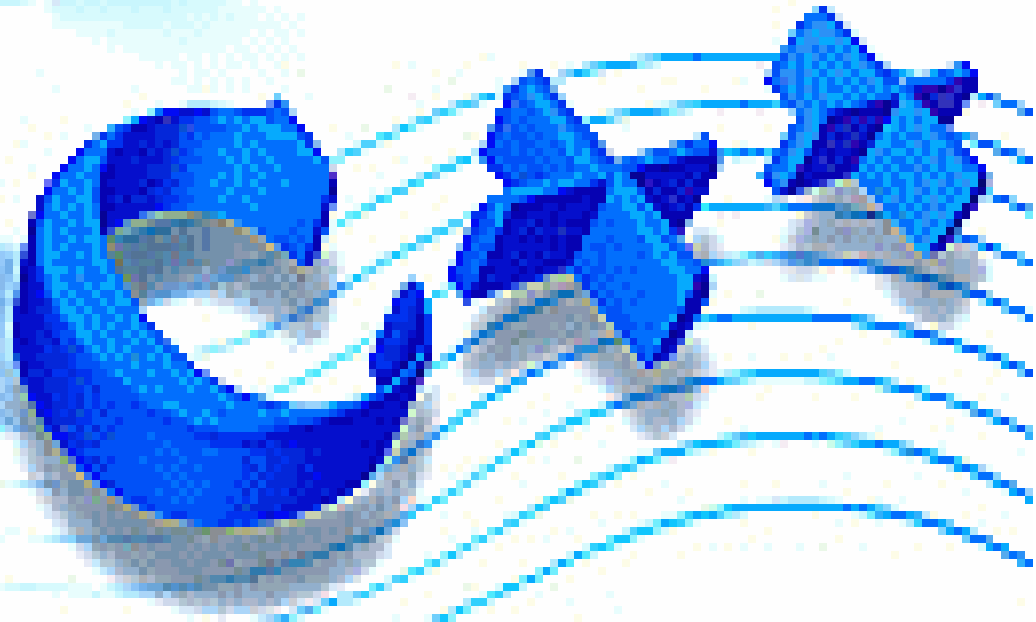

中国高等学校计算机科学与技术
应用系列教材

C++程序设计

[illegible] [illegible] 编

面向21世纪
高等学校系列教材

西安电子科技大学出版社
http://www.xdupt.com

□中国高等职业技术教育研究会推荐

高职系列教材

C++程序设计

陈圣国 阎会昌 编

西安电子科技大学出版社

2000

内 容 简 介

本书首先介绍了 C++语言与 C 语言的关系，介绍了面向对象的程序设计思想及基本概念。第 3 章~第 8 章围绕类与对象，系统介绍了 C++语言面向对象的语言成分。第 9 章~第 13 章为 Visual C++的使用。首先介绍 Visual C++开发环境的使用，然后围绕 Windows 环境下开发应用程序常见的几个方面的问题分别进行了介绍，包括 Windows 用户界面设计、数据库访问技术、ActiveX 控件使用和 Internet 编程。

本书语言简洁流畅，附有较多的实例，适合高职、高专类院校计算机专业 C++程序设计课程使用，也可以作为学过 C 语言并想进一步自修 C++语言的读者使用。

图书在版编目（CIP）数据

C++程序设计/陈圣国等编.

—西安：西安电子科技大学出版社，2000．8

高职系列教材

ISBN 7–5606–0892–2

Ⅰ. C… Ⅱ. 陈… Ⅲ. C 语言–程序设计–高等学校：技术学校–教材 Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2000)第 34481 号

责任编辑 马乐惠

出版发行 西安电子科技大学出版社(西安市太白南路 2 号)

电 话: (029)8227828 邮编 710071

http: //www.xduph.com E-mail: xdupfxb@pub.xaonline.com

经 销 新华书店

印 刷 西安市高陵县印刷厂

版 次 2000 年 8 月第 1 版 2001 年 3 月第 2 次印刷

开 本 787 毫米×1092 毫米 1/16 印张 13.25

字 数 307 千字

印 数 4 001～10 000 册

定 价 14.00 元

ISBN 7–5606–0892–2/TP・0474

前　言

随着计算机软件技术的发展，计算机软件应用范围越来越广，复杂程度也越来越高，传统的结构化程序设计思想越来越不能适应软件开发工作的需要。面向对象技术是计算机应用领域近几年迅速发展起来的软件开发技术，与传统的结构化程序设计思想相比有许多优点。采用面向对象的观点看待所要解决的问题，并将其抽象为系统是极其自然的，因为它更符合人类的思维习惯。

面向对象的程序设计方法易于编程、修改和维护，代码的可重用性很好，可以大幅度提高软件开发的生产率。特别是近几年来，面向对象的程序设计思想扩展到许多传统的程序设计语言，包括 Windows 环境下各种流行的开发平台，如 Visual Basic、Visual FoxPro、Delphi 等。因此有必要让学生了解面向对象的程序设计思想，并能够较熟练地使用一种面向对象的程序设计语言。

C++语言是一种影响较大的面向对象的程序设计语言，它充分体现了面向对象的程序设计思想。C++是 C 语言的超集，适合在 C 语言课程的基础上学习面向对象的基本思想和程序设计方法。

本书首先介绍了 C++语言与 C 语言的关系及 C++编译器的使用，以便于学生能够很快上机实习。第 2 章介绍了面向对象的程序设计思想及基本概念，同时简单介绍了面向对象的分析与设计的基本步骤。第 3 章~第 8 章围绕类与对象，系统介绍了 C++语言面向对象的语言成分。第 9 章~第 13 章为 Visual C++的使用，以使读者在学完 C++语法之后对 Windows 环境下应用程序的开发有一定的了解。首先介绍 Visual C++开发环境的使用，然后本着实用的宗旨，围绕 Windows 环境下开发应用程序常见的几个方面的问题分别进行了介绍，包括 Windows 用户界面设计、数据库访问技术、ActiveX 控件使用和 Internet 编程。

目录中带*的章节在教学中可以根据需要选学，其中部分章节需要相关课程的知识，如数据库编程需要对关系数据库模型和客户/服务器模式有一定的了解，Internet 编程需要具备网络基本概念和网络协议层次模型方面的知识基础。

本书由陈圣国主编，阎会昌编写了第 9 章、第 10 章和第 12 章的大部分内容，其余由陈圣国编写并统稿。深圳职业技术学院的余苏宁老师担任主审。本书在编写过程中得到了西安电子科技大学出版社的大力支持，在此谨表谢意。

作者

2000 年 6 月

目　　录

第 1 章　C 与 C++ 1
1.1　C语言与C++的关系 1
1.1.1　C++发展历史简介 1
1.1.2　C++与C语言 1
1.2　C++语言的开发环境 4
1.2.1　常见C++编译器简介 4
1.2.2　Borland C++ 3.1的使用 5
习题 6
第 2 章　面向对象的系统分析与设计方法 8
2.1　面向对象的基本思想和基本概念 8
2.1.1　面向对象的基本思想 8
2.1.2　对象 8
2.1.3　类与实例 9
2.1.4　继承性 10
2.1.5　多态性 10
2.1.6　封装 11
*2.2　面向对象的系统开发方法 12
2.2.1　概述 12
2.2.2　分析与设计的基本步骤 13
习题 15
第 3 章　类和对象 16
3.1　最简单的类定义 16
3.1.1　类定义 16
3.1.2　函数成员的实现 17
3.1.3　函数成员的隐含参数 18
3.1.4　C++中的结构 19
3.1.5　编程示例：集合的实现 19
3.2　构造函数与析构函数 23
3.2.1　构造函数 23
3.2.2　析构函数 25
3.2.3　编程示例：可变大小的集合类 25
3.3　类的派生 28
3.3.1　派生类的定义 28
3.3.2　基类与派生类的关系 29
3.3.3　构造函数与析构函数 31
3.3.4　虚函数 32
3.3.5　protected成员 34
3.3.6　编程示例：集合的实现 35
*3.3.7　多继承简介 39
3.4　友元 42
3.4.1　友元函数 42
3.4.2　友元成员 43
3.4.3　编程示例：矩阵与向量的乘积 44
3.5　静态成员 46
3.5.1　静态数据成员 46
3.5.2　静态函数成员 48
习题 48
第 4 章　重载 50
4.1　函数的重载 50
4.1.1　函数重载的定义 50
4.1.2　一个简单的例子 51
4.1.3　在C++中编译C程序 51
4.2　运算符重载 52
4.2.1　运算符重载的基本方法 52
*4.2.2　类型转换运算符 54
4.2.3　下标和函数调用运算符 56
习题 58
第 5 章　指针与引用 59
5.1　动态对象 59
5.1.1　动态对象与new、delete运算符 59
*5.1.2　重载new与delete 61
5.1.3　编程示例 62
5.2　引用 63
5.2.1　什么是引用 63
5.2.2　引用的定义 64
5.2.3　引用作函数参数 64
5.2.4　引用返回值 65
5.2.5　对象的复制 66
习题 69

第 6 章 模板 70
6.1 模板的概念 70
6.1.1 什么是模板 70
6.1.2 模板的基本语法 70
6.2 函数模板 71
6.2.1 函数模板的定义 71
6.2.2 重设模板函数 73
6.2.3 显式和隐式的模板函数 74
*6.3 类模板 75
6.3.1 类模板的定义与使用 75
6.3.2 类模板参数 76
6.4 编程示例:栈模板 77
习题 80
第 7 章 流 81
7.1 输入输出流 81
7.1.1 流的概念 81
7.1.2 输出流 81
7.1.3 输入流 83
7.1.4 格式控制 85
7.2 文件流 87
7.3 编程示例: 文本数据文件的读写 90
习题 92
第 8 章 异常处理 93
8.1 异常处理的传统方法 93
8.1.1 异常的概念 93
8.1.2 基于C的结构化异常 94
8.2 C++中的异常处理 95
8.2.1 C++异常的语法 95
8.2.2 异常种类的识别 97
8.2.3 异常信息的获取 98
8.2.4 异常处理的方法 98
*8.2.5 Visual C++中的异常语法 98
8.3 编程示例 99
习题 100
第 9 章 Visual C++集成环境的使用 101
9.1 Visual C++集成开发环境 101
9.1.1 集成开发环境的启动及其主窗口简介 101
9.1.2 正文窗口及其操作 102
9.1.3 Workspace窗口及其操作 103
9.2 菜单功能介绍 104
9.2.1 File菜单 104
9.2.2 Edit菜单 104
9.2.3 View菜单 105
9.2.4 Insert菜单 105
9.2.5 Project菜单 106
9.2.6 Build菜单 107
9.2.7 Tools菜单 107
9.2.8 Window菜单 108
9.2.9 Help菜单 108
9.3 工具栏的使用 109
9.3.1 标准(Standard)工具栏 109
9.3.2 显示/隐藏工具栏 109
9.3.3 工具栏的定制 109
9.4 资源与标识符 110
9.4.1 资源 110
9.4.2 资源编辑器 111
9.5 应用程序的建立 112
9.5.1 概述 112
9.5.2 利用AppWizard创建一个新的项目 112
9.5.3 程序分析 114
习题 121
第 10 章 Windows用户界面 122
10.1 Windows消息与命令 122
10.1.1 消息驱动机制 122
10.1.2 应用程序菜单 123
10.1.3 快捷键和加速键 130
10.1.4 工具栏和状态栏 131
10.2 单文档与多文档程序 133
10.2.1 文档视－图结构 133
10.2.2 单文档应用程序的建立 134
10.2.3 多文档应用程序的建立 141
10.3 对话框与常用组件 143
10.3.1 对话框 143
10.3.2 常用控件 152
10.4 通用对话框与Windows 95控件 157

10.4.1 通用对话框.....157
10.4.2 Windows 95控件.....162
习题.....166
第 11 章 数据库编程.....167
11.1 关系数据库模型.....167
11.1.1 数据结构.....167
11.1.2 完整性规则.....168
11.1.3 关系数据库管理系统.....168
11.1.4 结构化查询语言(SQL).....168
11.2 使用ODBC.....169
11.2.1 ODBC工作原理.....169
11.2.2 MFC ODBC类简介.....170
11.2.3 创建ODBC应用程序.....171
11.2.4 遍历、添加、修改和删除记录.....177
*11.2.5 数据库异常.....179
*11.2.6 记录的筛选和排序.....179
*11.2.7 统计函数使用.....180
*11.2.8 多表的连接.....181
*11.2.9 直接使用SQL语言.....183
*11.3 使用DAO.....184
11.3.1 DAO概述.....184
11.3.2 MFC DAO类.....184
11.3.3 创建DAO应用程序.....185
习题.....186
第 12 章 ActiveX.....187
12.1 COM、OLE、ActiveX简介.....187
12.1.1 OLE/COM.....187
12.1.2 OLE自动化.....187
12.1.3 OLE/ ActiveX控件.....188
12.2 编程示例: ActiveX控件的使用方法.....188
习题.....191
第 13 章 Internet编程概述.....192
13.1 Internet编程概述.....192
*13.2 WinInet类的使用.....192
13.2.1 WinInet简介.....192
13.2.2 一个简单的浏览器程序.....193
13.2.3 建立WinInet类应用程序的一般步骤.....194
13.3 ISAPI的使用.....196
13.3.1 ISAPI程序工作原理.....196
13.3.2 ISAPI类简介.....197
13.3.3 编程示例：建立一个ISAPI应用程序.....198
习题.....201
参考文献.....202

第1章 C与C++

1.1 C语言与C++的关系

1.1.1 C++发展历史简介

随着面向对象程序设计思想的日益普及，很多支持面向对象程序设计方法语言也相继出现了，C++就是这样一种语言。C++是Bjarne Stroustrup于1980年在AT&T的贝尔实验室开发的一种语言，它是C语言的超集和扩展，是在C语言的基础上扩充了面向对象的语言成分而形成的。最初这种扩展后的语言称为带类（class）的C语言，1983年才被正式称为C++语言。

Bjarne Stroustrup在设计和实现C++语言时，既保留了C语言的有效性、灵活性、便于移植等全部精华和特点，又添加了面向对象编程的支持，具有强大的编程功能，编写出的程序具有结构清晰、易于扩充等优良特性，适合于各种应用软件、系统软件的程序设计。

C++语言由C语言扩展而来，同时它又对C语言的发展产生了一定的影响，ANSI C语言在标准化过程中吸收了C++语言中某些语言成分。

1.1.2 C++与C语言

C++语言是C语言的超集，与C语言具有良好的兼容性，使用C语言编写的程序几乎可以不加修改直接在C++语言编译环境下进行编译。

C++语言对C语言在结构化方面做了一定程度的扩展。例如：

1．函数原型

C语言中，在使用一个函数之前可以不对其加以说明，编译器缺省认为其返回值为整型数据。但在C++编译环境中，任何一个函数在使用前，必须有函数的原型说明，声明函数的返回值类型及参数的类型。如下面这段在C语言教材中广泛使用的程序:

```
main( )
{
printf("Hello，World！\n");
}
```

在 C++编译器中编译时必须在前面加上：

```
#include <stdio.h>
```

以将函数 printf 的原型声明包含进来。使用函数原型，可以避免程序在调用其它函数时，错误引用其返回值或传递错误的实参给被调用的函数，发生这样的错误可以由编译器在编译时发现；如果使用 C 语言的函数声明习惯，则不能在编译阶段检查出此类错误。现代 C 语言标准已经引进了函数原型说明，但不作为一种强制标准。

2．函数重载

C++语言可实现函数重载，即多个函数在同一作用域可以用相同的函数名，编译器在编译时可以根据实参的类型来选择应该调用的函数。例如，在 C 语言的数学函数库中，求绝对值的函数有 abs、fabs 等，分别用于不同类型的参数；而在 C++中，对相同功能但参数类型不同的函数可以使用相同的函数名，在调用时无需记忆多个函数名，而由编译器根据参数类型选择。有关函数重载的具体内容可参看后面相关章节的介绍。

3．缺省参数

在 C++语言中，函数参数允许使用缺省值。当函数调用时，若给出的参数个数少于函数表中参数的总数时，则所缺参数自动取函数参数表中设置的缺省值。

下面是一个参数缺省的例子：

```
void f(int x, int y=10)
{
…//此处省略函数体内容
}
```

该程序定义了一个函数 f( )，它有两个整形参数，第二个参数缺省值为 10。如调用语句：

```
f(2);
```

就相当于 f(2,10)。函数可以有多个缺省参数，但应注意只能从右往左缺省，例如：

```
int f1(int x, int y=0, int z=0);//正确
int f2(int x, int y=0, int z);          //错误
```

4．注释

C++语言保留了 C 语言中以/*开始，*/结束的注释，这种方式适用于多行的注释，同时 C++语言中还提供行注释符//，该注释在它的行结束处结束，适用于短注释。

5．枚举名与结构名

C++中枚举可以命名，一个枚举名就是一个类型名字，因此不必在枚举类型名前加标识符 enum；同样，定义的结构就是一个用户定义的数据类型，在结构名前也不必加标识符 struct。例如下面定义一个结构类型：

```
struct student{
char name[10];
int number;
int page;
};
```

则在定义该结构类型变量时可以使用如下方式：

```
student s1, s2;
```

6．作用域标识符

在 C++语言中增加了作用域标识符（或称为名字解析运算符）::，用以解决局部变量名与全局变量的同名重复问题。在局部变量的作用域内可用作用域标识符 :: 对被其隐藏的同名全局变量进行访问。下面是一个简单的例子：

```
int x=0;
void test(int x)
{
x=5;                //此处引用局部变量
::x=9;                  //此处引用全局变量
}
```

7．程序块中的变量声明

将几个语句用大括号{、}组合起来就构成了一个程序块。在 C 语言中，变量的声明只允许出现在程序块的开始，而在 C++语言中变量声明可以出现在任何位置。如下面的程序片断：

```
for(int i=0; i<10; i++){
    sum+=i;
}
```

在 for 语句的第一个表达式中定义变量 i 作循环变量。

8．常量

C++语言中增加了常量类型，或称为只读变量。常量用标识符 const 声明，它的值在作用域内保持不变，ANSI C 标准现在已引进了 const 修饰符。例如：

```
const   int maxSize=128；
const int intArray[ ]={1, 2, 3, 4, 5, 6};
```

也可以声明常量指针，此时该指针不能指向其它对象，例如：

```
char * const str="Hello, world! ";
```

作了以上声明后，下面的赋值是非法的:

```
str="Hi, there! ";
```

另外，还可以声明一个指向常量，例如：

```
const char *str2="Hello, world";
```

此时指针 str2 所指向的对象不能被改变，但 str2 本身可以被改变，有了以上声明后下面的运算是非法的：

```
str2[1]='a';
```

常量必须在它被定义时初始化成某个值，一旦定义后，常量的值不能被改变。

函数的形参也可以声明为常量，这对于用指针作形式参数的函数可以防止在函数中修改指针形参所指向的对象。例如，C 语言头文件 string.h 中 strcmp 函数的原型声明为：

```
int strcmp(const char *, const char *);
```

9．内置函数

内置函数用标识符 inline 来定义。当程序编译时，就对内置函数在其调用处进行内置替换，将内置函数的代码插入在调用处，从而消除执行过程中的函数调用开销。

内置函数通常用于小而常用的函数，它与带参数的宏替换相似，但其处理方法不同。宏替换由预处理程序进行简单的字符串替换，在替换过程中不进行语法的检查；而内置函数由 C++编译器进行处理，在插入代码之前即进行语法检查。下面是一个简单的例子：

```
inline int max(int x, int y)
{
if(x>y)
      return x;
else
       return y;
}
```

在使用内置函数时应该注意：如果该函数由多个源文件引用，则应将该函数的实现代码放在头文件中，而不能仅仅将该函数的原型声明放入头文件中。

C++语言与 C 语言最显著的区别是它的面向对象的特征，引进了类与对象的概念。类封装了一组数据结构和作用于该数据结构的一组方法，下面对 C++语言的介绍将着重围绕类来进行介绍。

1.2 C++语言的开发环境

1.2.1 常见 C++编译器简介

使用 C++语言开发应用程序的过程与 C 语言相同，首先输入源程序，然后使用编译器编译生成目标代码文件，再由连接程序生成可执行的二进制文件。

目前可以选择的 C++编译器很多，在 PC 平台上，常见的操作系统环境下都有可使用的编译器，比较普遍的有微软公司与 Borland（ Inprise）公司的产品。

DOS 环境下，由于操作系统本身功能的限制，C++编译器提供的功能相对简单，如微软的 MS C 7.0、Borland C++ 3.1 等产品。相比而言，Borland C++的集成环境使用起来更方便。

目前，PC 操作系统越来越多地使用图形用户界面的 Windows 95 和 Windows NT。Windows 环境下，微软公司与 Borland 公司都有自己的 C++编译器，如微软的 Visual C++ 6.0、Borland 公司的 Borland C++ 5.0、C++ Builder 等，它们所提供的不仅仅是 C++编译器，而

是一个完整的开发平台。

在学习 C++基本语法的过程中，建议读者使用 DOS 系统下的 Borland C++ 3.1。它的集成环境简单易用，而且提供了强大的调试功能。下面先简单介绍一下 Borland C++3.1 的使用，本书的第 9 章将详细介绍 Visual C++开发平台的使用。

1.2.2 Borland C++ 3.1 的使用

1．安装

(1) 插入安装盘（第一张盘）到驱动器 A，在 DOS 提示符下键入：

A:install

(2) 在安装屏幕上按回车键，顺序回答所有的提示即可完成安装。Borland C++3.1 支持 DOS 及 Windows 3.1 应用程序的开发。如果不想安装 Windows 3.1 支持，可在安装过程中出现图 1.1 所示的屏幕时，将有关 Windows 的选项去掉。

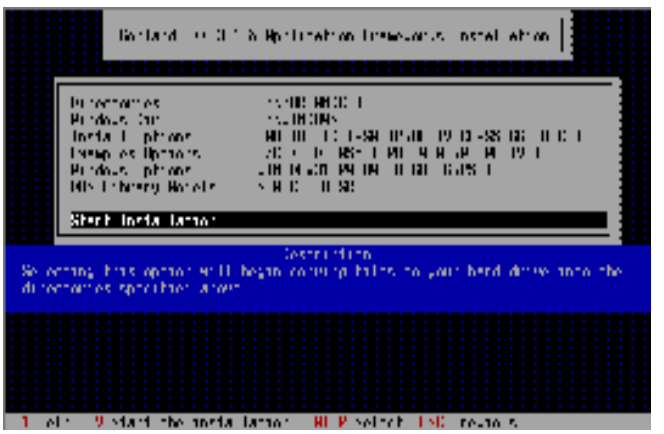

图 1.1 Borland C++3.1 的安装

2．用 Borland C++ 3.1 编译 C++程序

Borland C++3.1 安装完成后，启动安装目录下 bin 子目录下的 bc.exe，即可进入 Borland C++ 3.1 的集成环境。

Borland C++3.1 集成环境可以编译执行独立的源程序文件，也可以编译执行完整的由多个源程序文件组成的项目。操作方法是： 选择 File 菜单下的 New，Borland C++3.1 打开一个新的源程序编辑窗口如图 1.2 所示，输入源程序，然后按 F2 保存源程序（也可以使用 File 菜单下的 Save），源程序文件的扩展名使用 CPP。选择 Run 菜单下的 Run 即可编译执行程序。如果程序在编译连接过程中有错误，错误信息可在 Message 窗口中看到。如果是编译已存在的源程序文件，可用 File 菜单下的 Open 功能装入该程序文件。

如果要编译执行多个源程序文件构成的项目，需要建立一个项目文件，即选择 Project 菜单下的 Open 功能，输入项目文件名，如果该文件存在，则打开该项目文件，否则将建立一个新的项目文件。图 1.2 中的 Project VCIRC 窗口，即为一个已打开的项目文件窗口，在

该窗口中按 Insert 键弹出一个对话框，可选择要加入该项目文件的源程序文件。

图 1.2　Borland C++3.1 运行画面

3．Borland C++3.1 调试功能简介

Borland C++3.1 提供了较为强大的源程序级调试功能，可以通过 Run 菜单下的各项命令来跟踪程序的执行情况：

Goto Cursor（F4）命令运行用户的程序，从程序开始到光标所在行，如果光标所在行没有可执行语句，将出现警告信息。该命令不设置永久性的断点，如果光标所在行前有永久性的断点存在，程序将会中断执行。如果想在某一行设置断点，可以将光标移动至该行按 Ctrl+F8 设置断点，重复按将取消断点。

Trace Into（F7）命令逐行语句运行用户的程序，如果遇到一个函数调用将跟踪进入该函数，执行函数的每条语句。

Step Over（F8）命令功能与 Trace Into 相似，但是它不跟踪进入被调用的函数。

Program Reset 命令终止当前的调试状态，释放分配给用户程序的内存空间，关闭用户程序打开的所有文件。

Argument 命令用来输入用户程序的命令行参数，如果当前调试的程序需要命令行参数，可使用此功能输入。

使用 Borland C++3.1 提供的调试功能需通过 Options 菜单功能进行设置，使得在编译时目标代码中包含调试信息。详细的设置请参考 Borland C++3.1 的用户手册。

习　　题

1．简述 C 语言与 C++语言之间的关系。

2．C++语言对 C 语言结构化方面主要作了哪些扩展?

3．采用函数原型说明有什么好处?

4．下面的程序正确吗? 如果有错误请指出原因。

```
int max(int x=0, int y)
{
    if(x>y)
        return x;
    return y;
}
void main(void)
{
 int x=5, y=100;
 printf("The max is %d\n", max(x, y);
}
```

5．内置函数与带参数的宏定义有什么不同?

第2章 面向对象的系统分析与设计方法

2.1 面向对象的基本思想和基本概念

2.1.1 面向对象的基本思想

与传统的结构化程序设计方法用过程化的方式描述应用系统不同，面向对象的方法认为，客观世界是由各种各样的对象组成的，每个对象都有各自的内部状态和运动规律，不同对象之间通过消息传送相互作用和联系就构成了各种不同的系统。

采用对象的观点看待所要解决的问题，并将其抽象为系统是极其自然与简单的，因为它符合人类的思维习惯，使得应用系统更容易理解。同时，由于应用系统是由相互独立的对象构成的，使得系统的修改可以局部化，因此系统更易于维护。

本节首先介绍几个面向对象方法的基本概念。

2.1.2 对象

对象是客观世界中事物在计算机领域中的抽象，是一组数据和施加于该组数据上的一组操作（行为）组成的集合体。

对象是面向对象方法的主体。当一个对象映射为软件实现时由三个部分组成：

(1) 私有的数据结构。它用于描述对象的内部状态。

(2) 处理。也称为操作或方法，它施加于数据结构之上。

(3) 接口。这是对象可被共享的部分，消息通过接口调用相应的操作。接口规定哪些操作是允许的。它不提供操作是如何实现的信息。

C++语言中的对象由描述对象状态的数据结构和作用于这个数据结构上的方法（或称为操作）构成，它们都可以分为私有的和公有的两个部分，私有部分从对象的外部不可直接访问，而公有部分可以由对象的外部访问。C++语言中对象之间的相互联系和作用通过

对公有数据和方法（操作）的访问来实现。

例如对于某个人，它的状态可能是：

- 身高: 175 cm
- 年龄: 20
- 性别: 男
- 肤色: 黄色

它的操作可能是：

- 回答身高
- 回答年龄
- 回答性别
- 回答肤色

其它对象不能直接访问它的内部状态信息，只能通过调用这些公共操作来访问该对象。

客观世界的同一对象在不同的应用系统中，由于考察对象的角度不同，对其抽象的数据结构和操作都可能是不同的。例如对于一个学生，在学籍管理系统与户籍管理系统两个不同的应用系统中，抽象出的表示内部状态的数据结构和对数据结构进行的操作都是不同的。因此，在对实际应用系统中的对象进行分析时，应注意该系统的要求，区分哪些是该对象的本质特征。

2.1.3 类与实例

采用面向对象方法进行系统分析与设计时，对于一个具体的系统而言，可能存在很多具有相同特征的对象。例如，对于一个学籍管理系统，存在许多学生对象，它们具有相同的结构特征和行为特征，只是表示内部状态的数据值不同。为了描述这种相同结构特征和行为特征的对象，面向对象方法引入了类的概念。这一点与人们在认识客观世界的事物时所采取的分类思想相同。人们在认识事物时总是将具有相同特征的事物归为一类，属于某类的一个事物具有该类事物的共同特征。

类是对一组具有相同特征的对象的抽象描述，所有这些对象都是这个类的实例。对于学籍管理系统，学生是一个类，而一个具体的学生则是学生类的一个实例。一个类的不同实例具有相同的操作或行为的集合和相同的信息结构或属性的定义，但属性值可以不同；不同的实例具有不同的对象标识。对于学生类中的每一个对象，描述它们所使用的数据结构相同，但是其值不同。在程序设计语言中，类是一种数据类型，而对象是该类型的变量，变量名即是某个具体对象的标识。

因此，一个类的定义至少包含以下两个方面的描述：

(1) 该类所有实例的属性定义或结构的定义。

(2) 该类所有实例的操作（或行为）的定义。

在 C++语言中，一个类的定义包含数据成员和函数成员两部分内容。数据成员定义该类对象的属性，不同的对象属性的值可以不同；函数成员定义了该类对象的操作。

在一个系统中，每一个对象均属于某个类，类是对象的属性和操作的定义模板，而实

例是某个具体的对象。

2.1.4 继承性

人们在对客观世界的事物进行描述时，经常采取分类的方法。类是有层次的，即某个大类的事物可能分为若干小类，而这些小类可能又分为若干个更小的类。

面向对象思想采纳了事物分类的层次思想，在描述类的时候，某些类之间具有结构和行为的共性。例如教师类与学生类，在结构方面均具有姓名、年龄、身高、体重等，在行为（或操作）方面均具有回答身高、回答体重等操作。将这些共性抽取出来，形成一个单独的类——人，描述教师类和学生类中的共性。类人的结构特征和行为特征可以被多个相关的类共享，例如教师类和学生类继承了类人的结构和行为特征。在 C++语言中，通过类的派生机制来实现类的继承，可以从一个类中派生出一个新的类，这个类称为派生类的基类或父类，派生出的新类称为基类的派生类或子类。派生类的对象具有基类对象的特征，同时又有其自身特有的特征。一个教师类的对象与一个学生类的对象都具有类人所描述的特征，同时又具有各自所属教师类和学生类特有的特征。

利用类之间的继承关系，可以简化类的描述。在类人中描述教师类和学生类的共性，而在学生类和教师类中只需描述各自的个性。利用继承机制可以提高软件代码的可重用性。在设计一个新类时，不必从头设计编写全部的代码，可以通过从已有的具有类似特性的类中派生出一个类，继承原有类中的部分特性，再加上所需的新的特性。这一点与面向过程的设计语言中的过程或函数不同，要使用具有相似功能的过程或函数，必须修改源程序代码以使其适应新系统的功能需求，而类的派生机制无须原有类的源代码即可派生出新的类。

利用类及其继承性描述系统时，由于类之间的继承关系，可能会形成一种具有层次性的类结构。在使用类的层次结构描述系统时，某些类之间的层次关系可以有多种实现方案。例如中学生类，既可以直接从类人派生出来，也可以从类人的派生类——学生类派生出来。在设计类的层次结构时，应注意建立的类层次结构是否易于理解以及组织类结构的费用等方面的问题。设计出来的类层次结构是否合理，往往取决于系统分析员的经验等因素。

另外，人们在对客观世界的事物分类时，一个事物可能属于多个类，具有多个类的特性。例如一个黑人学生，他既属于学生类，又属于黑人类。这种情形在面向对象方法中称为多继承，即一个类同时从多个类中派生出来，此时类的层次结构是网状的。多继承在有些面向对象的程序设计语言中是不允许的，C++语言允许多继承。只允许派生类有一个基类称为单继承，单继承的类层次结构是树状的。

2.1.5 多态性

多态性是面向对象系统的又一重要特性。所谓多态即一个名词可具有多种语义。

在 C++语言中，多态性主要表现在下述几个方面：

(1) 重载：在 C++语言中，同一函数名以及运算符可以具有不同含义的多种实现版本，编译器将根据函数调用的参数形式决定调用哪一种实现版本。