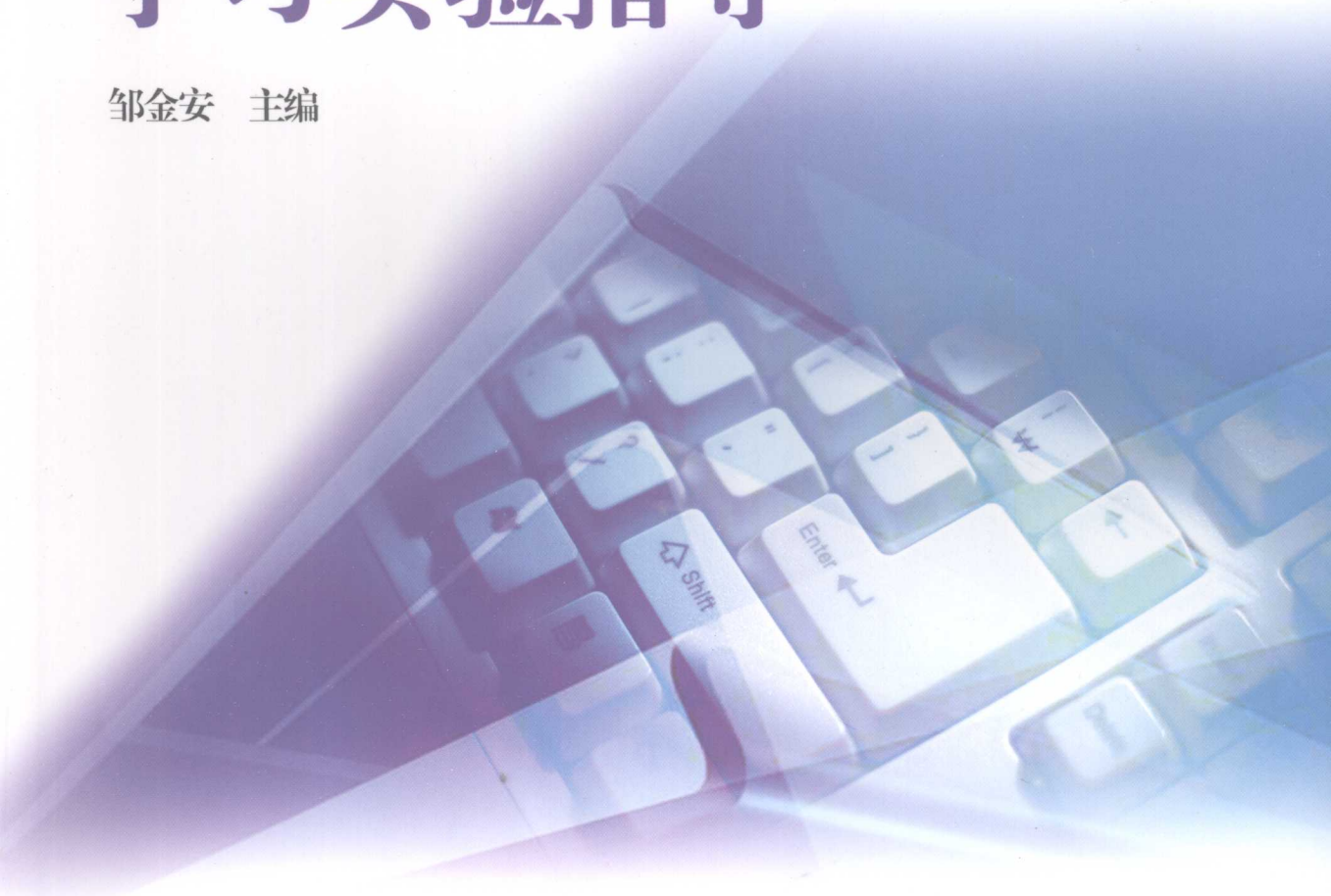
面向21世纪应用型本科计算机规划教材

面向对象程序设计与 *Visual C++ 6.0* 学习实验指导

邹金安　主编



厦门大学出版社
XIAMEN UNIVERSITY PRESS

面向对象程序设计与Visual C++6.0学习实验指导

主　编　邹金安

副主编　赵少卡　杨剑炉

厦门大学出版社

图书在版编目(CIP)数据

面向对象程序设计与 Visual C++ 6.0 学习实验指导/邹金安主编. —厦门:厦门大学出版社,2009.7
(面向 21 世纪应用型本科计算机规划教材)
ISBN 978-7-5615-3217-1

Ⅰ.面… Ⅱ.邹… Ⅲ.C 语言-程序设计-高等学校-教学参考资料 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2009)第 106813 号

厦门大学出版社出版发行
(地址:厦门市软件园二期望海路 39 号 邮编:361008)
http://www.xmupress.com
xmup @ public.xm.fj.cn
厦门市明亮彩印有限公司印刷
2009 年 7 月 1 版 2009 年 7 月第 1 次印刷
开本:787×1092 1/16 印张:15.75
字数:373 千字 印数:1～2000 册
定价:24.00 元
如有印装质量问题请与承印厂调换

前 言

面向对象程序设计(Object-Oriented Programming,OOP)是一种程序开发的方法论。它将对象作为程序的基本单元,将程序和数据封装其中,以提高软件的重用性、灵活性和扩展性。

目前已经证实面向对象程序设计增强了程序的灵活性和可维护性,并且在大型项目设计中广泛应用。面向对象程序设计能够让人们更简单地设计并维护程序,使得程序更加便于分析、设计与理解。

为配合《面向对象程序设计与 Visual C++ 6.0 教程》的学习与实验,我们编写了此书。全书正文分四部分,第一部为各章知识点,对教程各章的重点知识点进行了回顾;第二部分为实验,共设计了 12 个实验,其中包括两个综合实验;第三部分为课程设计,包括“课程设计指导书”,以及“图书出版管理系统”和“复数计算器”的设计;第四部分为模拟试卷,包括 4 套模拟试卷,并提供参考答案和评分标准。

全书各部分的编写分工如下:

莆田学院杨剑炉老师编写第一部分;福建师范大学福清分校赵少卡老师编写第二部分的实验 1～5、第三部分、第四部分试卷 1～2;莆田学院邹金安副教授编写第二部分实验 6～11、第四部分试卷 3～4 及附录。第二部分的实验 12 由邹金安和赵少卡合作编写。最后由邹金安负责统稿。

莆田学院计算机专业 2005 级学生陈俊雨和曾琼两位同学帮助本书的录入,在此表示感谢!

编 者

2009 年 6 月

目 录

前言
第一部分 各章知识点 …… 1
第 1 章 Visual C++集成开发环境 …… 1
知识点 1:Visual C++常用功能键及其意义 …… 1
第 2 章 程序设计概述 …… 2
知识点 1:编程操作流程 …… 2
知识点 2:IDE 集成开发环境 …… 2
知识点 3:主函数的返回类型 …… 2
知识点 4:C++的头文件 …… 3
知识点 5:结构化程序设计 …… 3
知识点 6:面向对象程序设计 …… 3
第 3 章 C++语言基础 …… 4
知识点 1:向量的定义 …… 4
知识点 2:向量的常用方法 …… 4
知识点 3:内联函数 …… 5
知识点 4:函数重载 …… 5
知识点 5:引用 …… 5
知识点 6:名空间的定义 …… 5
知识点 7:名空间成员的访问 …… 6
知识点 8:编译预处理 …… 6
知识点 9:#include 指令 …… 6
知识点 10:#define 和#undef 指令 …… 6
第 4 章 类 …… 7
知识点 1:类的定义 …… 7
知识点 2:类的三种访问类型 …… 7
知识点 3:类的成员函数 …… 7
知识点 4:对象的定义 …… 8
知识点 5:类外访问成员的方法 …… 8
知识点 6:静态数据成员 …… 8
知识点 7:静态成员函数 …… 9
知识点 8:友员函数 …… 9

知识点 9：友员类 …… 9
知识点 10：运算符重载 …… 10
第 5 章　对象 …… 10
知识点 1：构造函数 …… 10
知识点 2：构造函数的重载 …… 11
知识点 3：拷贝构造函数 …… 11
知识点 4：析构函数 …… 11
知识点 5：构造顺序 …… 12
知识点 6：静态对象 …… 12
第 6 章　继承 …… 12
知识点 1：继承和派生的概念 …… 12
知识点 2：派生类的声明 …… 13
知识点 3：多继承 …… 13
知识点 4：派生类的构造函数 …… 13
知识点 5：类的继承方式 …… 14
知识点 6：虚拟继承 …… 14
第 7 章　面向对象程序设计的方法与步骤 …… 15
知识点 1：抽象编程 …… 15
知识点 2：过程化分析步骤 …… 15
知识点 3：基于对象的分析步骤 …… 15
第 8 章　多态与抽象类 …… 16
知识点 1：多态的概念 …… 16
知识点 2：多态的分类 …… 16
知识点 3：虚函数 …… 16
知识点 4：虚函数与重载的关系 …… 17
知识点 5：虚函数的限制 …… 17
知识点 6：纯虚函数 …… 17
知识点 7：抽象类 …… 17
第 9 章　模板 …… 18
知识点 1：模板的概念 …… 18
知识点 2：函数模板 …… 18
知识点 3：类模板 …… 19
知识点 4：类模板的成员函数 …… 19
知识点 5：类模板的对象 …… 19
第 10 章　异常 …… 20
知识点 1：异常处理的语法 …… 20
知识点 2：关于异常的一些注意点 …… 21

知识点 3:异常处理的执行过程 …… 21
第 11 章 IO 流 …… 22
知识点 1:输入/输出标准流类 …… 22
知识点 2:常用的 I/O 流类库控制符 …… 22
知识点 3:文件流类 …… 23
知识点 4:打开磁盘文件 …… 23
知识点 5:文件打开模式 …… 24
知识点 6:关闭磁盘文件 …… 24
知识点 7:对 ASCII 文件的操作 …… 24
知识点 8:对二进制文件的操作 …… 24
知识点 9:文件定位 …… 25
第 12 章 创建应用程序框架 …… 25
知识点 1:projects 类型 …… 25
知识点 2:创建 MFC AppWizard[exe]应用程序流程 …… 26
知识点 3:ClassWizard 类向导 …… 26
第 13 章 MFC 应用简介 …… 27
知识点 1:MFC(Microsoft Foundation Class) …… 27
知识点 2:MFC 的类体系 …… 27
第 14 章 对话框编程 …… 29
知识点 1:对话框的常用成员函数 …… 29
知识点 2:消息对话框 …… 30
知识点 3:静态控件(CStatic) …… 30
知识点 4:按钮控件(CButton) …… 31
知识点 5:编辑框控件(CEdit) …… 31
知识点 6:列表框控件(CListBox) …… 31
知识点 7:滚动条控件(CScrollBar) …… 31
知识点 8:滑动条控件(CSliderCtrl) …… 31
知识点 9:组合框控件(CComboBox) …… 31
第 15 章 文档类与视图 …… 32
知识点 1:文档/视图概念 …… 32
知识点 2:文档类(CDocument) …… 32
知识点 3:视图类(CView) …… 33
知识点 4:文档、视图框架(Document Frame,View Frame) …… 33
知识点 5:文档模板(Document Template) …… 33
第 16 章 高级应用程序 …… 33
知识点 1:ODBC MFC 类 …… 33
知识点 2:CDatabase 类 …… 34

知识点 3:CRecordSet 类 …… 34
知识点 4:多线程 …… 34
知识点 5:CWinThread 类 …… 34
第二部分　实验 …… 36
实验一　C++程序的编辑、调试与运行 …… 36
实验二　数组、指针的使用 …… 41
实验三　类的定义和使用 …… 45
实验四　类的继承、多态与运算符重载 …… 50
实验五　标准设备与文件流的输入输出 …… 56
实验六　综合实验一:学生选课小系统 …… 60
实验七　创建简单的 MFC 应用程序 …… 73
实验八　滚动条的设计与实现 …… 81
实验九　Windows 标准控件在可视化编程中的应用 …… 85
实验十　幸运 52 游戏模拟程序的设计与实现 …… 91
实验十一　数据库应用程序的开发 …… 97
实验十二　综合实验二:小型学生信息管理系统 …… 107
第三部分　课程设计 …… 128
课程设计指导书 …… 128
课程设计 1:图书出版管理系统 …… 133
课程设计 2:复数计算器 …… 169
第四部分　模拟试卷 …… 188
模拟试卷 1 …… 188
模拟试卷 1 参考答案及评分标准 …… 193
模拟试卷 2 …… 198
模拟试卷 2 参考答案及评分标准 …… 206
模拟试卷 3 …… 209
模拟试卷 3 参考答案及评分标准 …… 216
模拟试卷 4 …… 219
模拟试卷 4 参考答案及评分标准 …… 226
附录 A　Visual C++ 6.0 程序调试 …… 229
附录 B　MFC 库简介 …… 239
附录 C　Visual C++开发 Windows 程序的步骤 …… 242

第一部分
各章知识点

第 1 章　Visual C++集成开发环境

知识点 1：Visual C++常用功能键及其意义

表 1-1　Visual C++常用功能键及其意义

操作类型	功能键	对应菜单	含义
文件操作	Ctrl+N	File|New	创建新的文件、项目等
	Ctrl+O	File|Open	打开项目、文件等
	Ctrl+S	File|Save	保存当前文件
编辑操作	Ctrl+X	Edit|Cut	剪切
	Ctrl+C	Edit|Copy	复制
	Ctrl+V	Edit|Paste	粘贴
	Ctrl+Z	Edit|Undo	撤销上一个操作
	Ctrl+Y	Edit|Redo	重复上一个操作
	Ctrl+A	Edit|Select All	全选
	Del	Edit|Del	删除光标后面的一个字符
建立程序操作	Ctrl+F7	Build| Compiler current file	编译当前源文件
	Ctrl+F5	Build|Run exe	运行当前项目
	F7	Build|Build exe	建立可执行程序
	F5	Build|Start Debugging	启动调试程序
调试	F5	Debug|Go	继续运行
	F11	Debug|Step into	进入函数体内部
	shift+F11	Debug|Step out	从函数体内部运行出来
	F10	Debug|Step over	执行一行语句
	F9		设置/清除断点
	Ctrl+F10	Debug|Run to cursor	运行到光标所在位置
	shift+F9	Debug|QuickWatch	快速查看变量或表达式的值
	Shift+F5	Debug|Stop Debugging	停止调试

第 2 章　程序设计概述

知识点 1：编程操作流程

一般的编程操作流程为：编辑(edit)→编译(compile)→连接(link 或 make 或 build)→调试(debug)，该过程循环往复，直到编译成功。

程序员编辑的程序，也称源程序，或称源代码(source code)，简称代码，以文本形式存放在以 .cpp(在 Windows 环境中)为扩展名的文件中。在比较少的情形下，机器指令集代码也称源代码。程序被编译后，生成目标代码(object code)，存放在目标文件中。Windows 中的 C++编译器通常将目标文件以 .obj 作为文件扩展名。目标代码即机器代码，是计算机能够识别的指令集合。但是，目标指令(也称目标代码)还不能在具体的计算机上运行，因为目标代码只是一个个独立的程序段，程序段之间还没有相互呼应，程序段中用到的 C++库代码和其他资源还没有挂上，需要相互衔接成适应一定操作系统环境的可执行程序整体。为了把成组的程序段转换为可执行程序，必须进行链接(link)，链接的过程就是将目标代码整合(或称转换)成可执行文件。可执行文件通常以 .exe 作为文件扩展名。

知识点 2：IDE 集成开发环境

较早期程序设计的各个阶段都要用不同的软件来进行处理，如先用字处理软件编辑源程序，然后用链接程序进行函数、模块连接，再用编译程序进行编译，开发者必须在几种软件间来回切换操作。现在的编程开发软件将编辑、编译、调试等功能集成在一个桌面环境中，这样就大大方便了用户。

集成开发环境(Integrated Develop Environment，IDE)是用于提供程序开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面工具，就是集成了代码编写功能、分析功能、编译功能、调试功能等一体化的开发软件服务平台。所有具备这一特性的软件平台都可以叫作集成开发环境，如微软的 Visual Studio C++便是本书所用的集成开发环境。

知识点 3：主函数的返回类型

在 ISO C++标准中规定 C++的主函数的返回类型必须是整型：

```
int main(){/*...*/}
int main(int argc,char* argv[]){/*...*/}
```

主函数的形式可以不同，但返回值一定都是整型。主函数要向系统的调用者返回整型值。系统不能忽略这个返回值，所以 void main()在 C++和 C 中都是非法的。即使有的编译器能

够编译 void main()，也不要使用。在 C++里，main()函数并不一定显式地返回某个整数。此时，它缺省返回 0，表示运行正常。

知识点 4：C++的头文件

现在所有的标准 C++头文件都是没有 .h 的，如：

<iostream.h>(老版本)对应<iostream>(新版本)

C 中头文件在 C++中标准为：前面加 c，后面没有 .h，如：

<stdio.h>(c 头文件)变成<cstdio>(C++标准)

使用 C++标准中的头文件时要注意所有的标准头文件都是放在 namespace std 中，所以在 INCLUDE 完 C++标准头文件后要加上：

using namespace std;

知识点 5：结构化程序设计

结构化程序设计是建议采用有含义的变量名，在程序的全局和局部范围，面向过程、自顶向下的编程方法。它的基本思想是：自顶向下，逐步求精，将整个程序结构划分成若干个功能相对独立的子模块，并要求这些子模块间的关系尽可能简单；子模块又可继续划分，直至最简；每一个模块最终都可用顺序、选择、循环三种基本结构来实现。程序基本控制结构有以下特点：

- 有一个入口；
- 有一个出口；
- 结构中每一部分都应当有被执行到的机会，也就是说，每一部分都应当有一条从入口到出口的路径通过它(至少通过一次)；
- 没有死循环(无终止的循环)。

知识点 6：面向对象程序设计

面向对象最基本的观点是认为一切都是对象，客观世界是由“对象”构成的，对象都有一定的状态和功能。面向对象不仅以这样的观点分析和看待客观世界，而且认为程序也同样应该以这样的方式来构造。在面向对象程序设计中，将对象作为构成软件系统的基本单元，并从相同类型的对象中抽象出一种新型的数据结构——类。对象是类的实例。类是一种区别于其他各种一般数据类型的特殊类型。类的成员中不仅包含描述类对象属性的数据，还包含对这些数据进行处理的程序代码，称之为对象的行为(或操作)。对象将其属性和行为封装在一起，并将其内部大部分的实现细节隐藏起来，仅通过一个可控的接口与外界交互。

第 3 章　C++语言基础

知识点 1:向量的定义

vector 可以有四种定义方式:

(1)vector<int>a(5);

(2)vector<int>b(5,1);

(3)vector<int>c(b);

(4)vector<int>d(a. begin(),a. begin+5);

vector<int>是模板形式,尖括号中为元素类型名,它可以是任何合法的数据类型。

- 第一种形式定义了 5 个整型元素的向量,但并没有给出初值,也就是其值是不确定的。
- 第二中形式同样的也定义了 5 个整型元素的向量,且给出其每个元素的初值为 1。这种形式是数组所不能匹配的,而数组只能通过循环来给每个元素赋初值。
- 第三种形式用一个已有的向量创建另一个向量。
- 第四种形式定义了一个向量,其值依次为 a 向量中从第 0 个元素到第 4 个(总共 5 个)元素的值。

知识点 2:向量的常用方法

- size();　　//返回元素个数
- empty();　　//return size()==0;
- begin();　　//返回第一个元素的位置
- end();　　//返回最后一个元素再后一个位置
- []　　//下标运算符
- push_back(p);　　//在末尾添加元素 p
- push_front(p);　　//在表头添加元素 p
- pop_back();　　//删除向量最后一个向量
- insert(p,x);　　//在 p 前插入 x
- insert(p,n,x);　　//在 p 前插入 n 个 x
- insert(p,first,last);　　//把从 first 到 last 的元素插入到 p 前
- resize(n);　　//向量元素个数为 n 个
- a. resize(n,x);　　//向量元素个数为 n,且每个赋值为 x
- erase(p);　　//删除指定位置的元素
- clear();　　//向量中元素清空

知识点 3：内联函数

内联函数不是在调用时发生转移，而是在编译时将函数体嵌入到每一个调用语句处。这样就相对节省了参数传递、系统栈的保护与恢复等的开销。

内联函数在定义时使用关键字 inline 以区别于一般函数，其语法形式如下：

```
<inline><类型标识符><被调函数名>(含类型说明的形参表)
{
  函数体
}
```

知识点 4：函数重载

C++编译系统允许为两个或两个以上的函数取相同的函数名，但是形参的个数或者形参的类型不应相同，编译系统会根据实参和形参的类型及个数的最佳匹配，自动确定调用哪一个函数，这就是所谓的函数重载。

注意点：

1. 重载函数的参数匹配

- 对于 int 形参，char 和 short int 为严格匹配；
- 对于 double 形参，float 也是严格匹配；
- C++允许 int 到 long，int 到 double 的转换。

2. 函数的返回值不用于区分重载函数

例如，下面的声明是错误的：

```
void func(int);
int func(int);
```

3. 调用省略默认参数的函数时可能与调用另一重载函数相同，调用时会发生语法错误。例如：

```
int func();
int func(int a=1);
```

知识点 5：引用

引用是另一个变量的别名，必须在声明时就初始化。声明引用的语法如下：

```
类型 &引用名=变量名;
```

知识点 6：名空间的定义

名空间是 C++为解决变量、函数名和类名等标识符的命名冲突服务的，它的基本方法是

将变量等标识符定义在一个不同名的名空间中。名空间的定义由 C++关键字 namespace 引导，其定义格式如下：

```
namespace  名空间标识符名
{
  成员的声明;
}
```

其中，名空间标识符名在所定义的域中必须是唯一的，名空间内的成员既包括变量，也包括函数。

知识点 7：名空间成员的访问

在使用名空间的成员时需用名空间名进行标识，从而有效解决了标识符冲突。名空间成员的访问方式如下：

```
名空间标识符名::成员名
```

知识点 8：编译预处理

预处理程序又称预处理器，它包含在编译器中。预处理程序提供了一组编译预处理指令和预处理操作符。编译器在对源程序进行编译之前，首先要由预处理程序对程序文本进行预处理，然后编译器接受预处理程序的输出，并将源代码转化成用机器语言编写的目标文件。

使用预处理指令需要注意以下几点：

①所有的预处理指令在程序中都是以“#”来引导的。

②每一条预处理指令单独占用一行，不需要用分号结束。

③预处理指令可以根据需要出现在程序中的任何位置。

知识点 9：#include 指令

#include 指令也称文件包含指令，其作用是将另一个源文件嵌入到当前源文件中该点处。在前面各章中已经用过#include 指令来嵌入头文件。文件包含指令有两种格式：

格式 1：#include<文件名>

功能：按标准方式搜索 C++系统目录下的 include 子目录。

格式 2：#include "文件名"

功能：首先在当前目录中搜索，若没有，再按标准方式搜索。

知识点 10：#define 和#undef 指令

在 C 语言中，#define 用来定义符号常量和带参数的宏。例如：

```
#define PI 3.14                    //定义了一个符号常量 PI
#define MAX(a,b) ((a)>(b)? (a):(b))    //定义一个带参数的宏 MAX(a,b)
```

在 C++中，虽然仍可以这样定义，但是定义符号常量已经被 const 所替代；定义带参数的宏也被 inline 内嵌函数所代替。#undef 的作用是删除由 #define 定义的宏，使之不再起作用。

第 4 章　类

知识点 1：类的定义

类(class)是一种用户自定义的数据类型。声明一个类时，以关键字 class 开始，接着是类的名字，其语法结构为：

```
class <类名称>
{
  private:
    [<私有数据和函数>]
  protected:
    [<保护数据和函数>]
  public:
    [<公有数据和函数>]
};
```

知识点 2：类的三种访问类型

- 私有类型(private)：私有类型的数据只允许类本身声明的函数对其进行存取，而类的外部任何函数都不能访问。
- 保护类型(protected)：保护类型用于类的继承，当类的成员被声明为 protected 时，从类的外部就不能对它们进行访问。
- 公有类型(public)：它们是类与外部的接口，任何外部函数都可以访问公有类型的数据和函数。

知识点 3：类的成员函数

类的成员函数用于对数据成员处理，又称为“方法”。程序中通过类的成员函数来访问其内部的数据成员。成员函数是类与外部程序之间的接口。在类的声明体中只需声明成员函数

的原型，成员函数的具体实现可放在类外定义。通常采用下面的形式定义成员函数：

```
<类型标识符><类名>::<成员函数名>(<形参表>)
{
  <函数体>
}
```

知识点 4：对象的定义

与结构体一样，类也是一种数据类型。要想使用类，还必须声明对象(Object)。对象是类的实例。

对象定义的一般格式为：

```
类名  对象名;
```

知识点 5：类外访问成员的方法

声明了类及其对象，就可以访问对象的公有成员。类外访问成员的方法为：

```
对象.公有成员函数名(实参表)
对象.公有数据
```

或定义一个指向对象的指针来访问公有段的成员。方式为：

```
指针->公有成员函数名(实参表)
指针->公有数据
```

或者：

```
(*指针).公有成员函数名(实参表)
(*指针).公有数据
```

知识点 6：静态数据成员

- 静态数据成员用于解决同一个类的不同对象之间的数据共享问题。静态数据成员声明时，应在前面加 static 关键字来说明。例如：

static int n;

- 静态数据成员必须初始化，并且一定要在类外进行。其初始化的形式如下：

<类型标识符><类名>::<静态数据成员名>=<值>

例如：int point::n=0;

- 静态数据成员属于类，而不属于任何一个对象，所以，在类外只能通过类名对它进行引用。静态数据成员引用的一般形式如下：

```
<类名>::<静态数据成员名>;
```

知识点 7：静态成员函数

• 所谓静态成员函数，就是使用 static 关键字声明的函数成员。同静态数据成员一样，静态函数成员也属于整个类，由同一个类的所有对象共同维护，为这些对象所共享。

• 静态成员函数可以直接引用该类的静态数据成员，而不能直接引用非静态数据成员。

• 公有类型的静态成员函数在类外的调用格式有两种：

格式一：

```
类名::静态成员函数(实参表)
```

格式二：

```
对象.静态成员函数(实参表)
```

若采用格式二调用，则对象会自动转换为所对应的类，然后还是按格式一调用。

知识点 8：友员函数

• 友员函数不是当前类的成员函数（这个函数可以是不属于任何类的非成员函数，也可以是其他类的成员函数），而是独立于当前类的外部函数，但它可以访问该类的所有对象的成员，包括私有成员和公有成员。

• 普通函数声明为友员函数的形式如下：

```
friend<类型标识符><友员函数名>(参数表)
```

• 成员函数声明为友员函数的形式如下：

```
friend<类型标识符><类名>::<友员函数名>(参数表)
```

• 友员函数的声明可以在类声明中的任何位置，既可在 public 区，也可在 protected 区，意义完全一样。

知识点 9：友员类

如果 A 是 B 的友员类，则 A 中的所有成员函数可以像友员函数一样访问 B 类中的所有成员。友员类的声明同样可以在类声明中的任何位置，其声明形式如下：

```
friend class<友员类名>
```

关于友员类，还有两点需要注意：

①友员关系是不能传递的。B 类是 A 类的友员，C 类是 B 类的友员，C 类和 A 类之间如果没有声明，就没有任何友员关系，不能进行数据共享。

②友员关系是单向的。如果声明 B 类是 A 类的友员，B 类的成员函数就可以访问 A 类的私有和保护数据，但 A 类的成员函数不能访问 B 类的私有和保护数据。