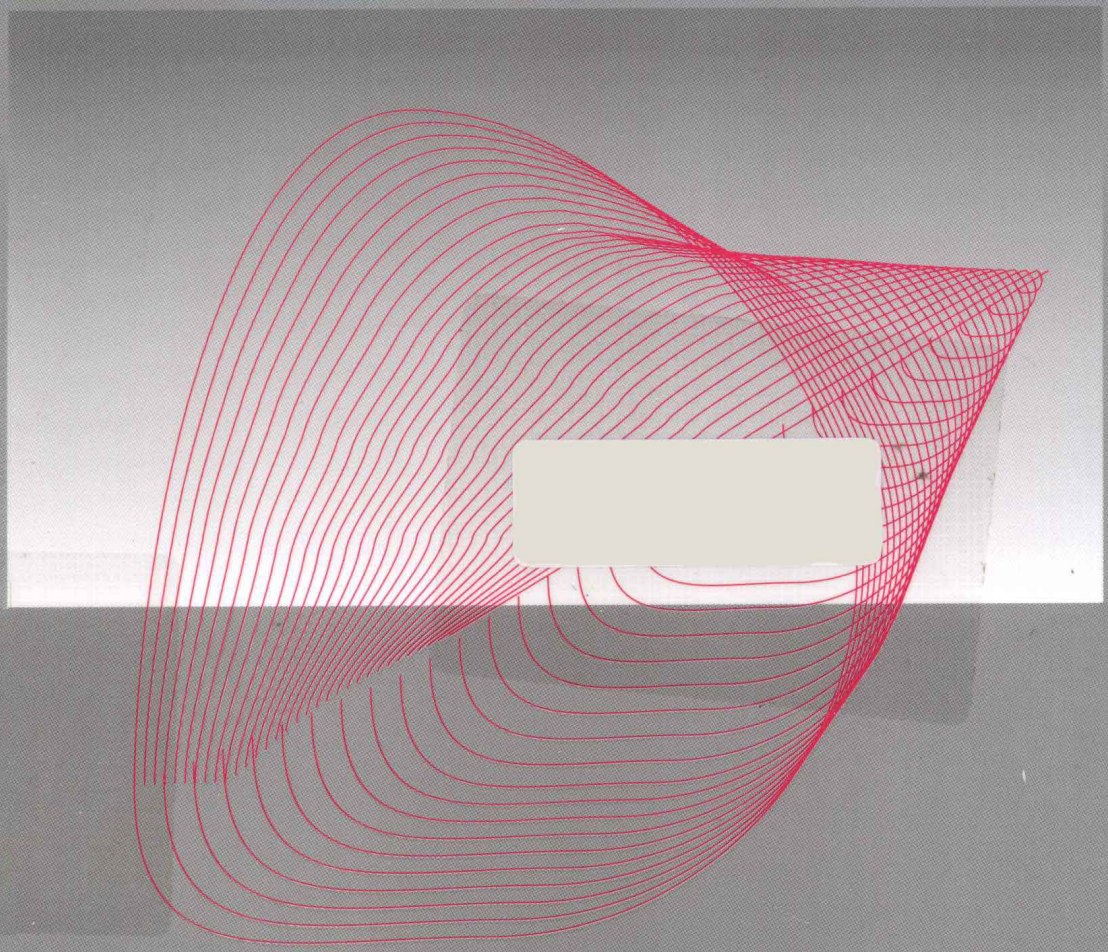
21 世纪高等学校计算机教育实用规划教材

数据结构与应用教程

（C++版）

马石安 魏文平 编著



清华大学出版社

21 世纪高等学校计算机教育实用规划教材

数据结构与应用教程
(C++版)

马石安 魏文平 编著

清华大学出版社
北京

内 容 简 介

全书采用面向对象的观点讨论数据结构技术，并以C++类模板作为算法描述工具。

本书在简要回顾C++程序设计概念的基础上，全面系统地介绍了线性表、栈和队列、串、数组和广义表、树和二叉树及图等数据结构，讨论了常用的查找和排序技术。对每一种数据结构，除了详细阐述其逻辑结构、存储结构和相关算法外，并对所有算法进行了C++语言实现和评价，最后通过实例来了解它的基本应用。对查找和排序算法，还着重在时间上作出了定量或定性的分析比较。本书最后还精心设计了8个上机实验。

全书条理清晰，语言通俗，图文并茂，可操作性强，列举实例丰富、典型。每章后面提供的练习题和附录部分的实验内容与教学要求一致，并提供全方位的教学资源。

本书可作为高等院校计算机及相关专业的教材或参考书，也可供自学者使用。

图书在版编目(CIP)数据

数据结构与应用教程：C++版/马石安，魏文平编著. --北京：清华大学出版社，2012.9
(21世纪高等学校计算机教育实用规划教材)
ISBN 978-7-302-29109-1

Ⅰ. ①数… Ⅱ. ①马… ②魏… Ⅲ. ①数据结构－高等学校－教材 ②C语言－程序设计－高等学校－教材 Ⅳ. ①TP311.12 ②TP312

中国版本图书馆CIP数据核字(2012)第131025号

责任编辑：魏江江　薛　阳
封面设计：傅瑞学
责任校对：焦丽丽
责任印制：张雪娇

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62795954
印 刷 者：北京世知印务有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：20.25　　**字　　数**：496千字
版　　次：2012年9月第1版　　**印　　次**：2012年9月第1次印刷
印　　数：1～3000
定　　价：33.50元

产品编号：038632-01

前言

当今的程序设计员不但要掌握一般的程序设计技巧，而且要对计算机加工处理的对象进行系统的研究。数据结构正是一门研究非数值计算的程序设计问题中计算机的操作对象以及它们之间的关系和操作等内容的学科。

数据结构在计算机科学的各领域中应用十分广泛，如编译系统中要使用栈、语法树等；操作系统中要使用队列、存储管理表、目录树等；数据库系统中要使用线性表、链表、索引树等；人工智能中要使用广义表、检索树、图等；同样，在面向对象的程序设计、计算机图形学、多媒体技术、软件工程等领域，都会用到各种不同的数据结构。因此，学好数据结构，可以掌握更多的程序设计技巧，并能评价算法的优劣，为以后学习计算机专业课程及走上工作岗位从事计算机大型软件开发打下良好的基础。目前，数据结构不仅是计算机专业的核心课程，而且是其他非计算机专业的主要选修或必修课程。

数据结构随着程序设计的发展而发展，面向对象技术是目前最流行的程序设计技术。为了适应形势的需要，有必要开设结合面向对象技术的数据结构课程。本书采用面向对象的观点讨论数据结构技术，并以 C++语言作为算法描述工具。

全书包括 10 章和一个附录：第 0 章介绍 C++程序设计语言预备知识，让不熟悉 C++的读者迅速掌握 C++的基本要点，为后续章节打下基础；第 1 章为绪论，介绍数据结构的发展、研究的内容、基本概念和术语及算法的定义和分析；第 2～7 章讲解线性表、栈和队列、串、数组和广义表、树和二叉树及图等数据结构的逻辑结构、存储结构和相关算法，并对每种算法进行了 C++语言实现和评价，最后通过实例来了解它们的基本应用；第 8 章和第 9 章介绍在计算机中非常广泛的两种运算：排序和查找，结合图解和实例描述常用的排序和查找算法，并给出了 C++语言实现，还着重在时间上作出了定量或定性的分析比较。附录部分是实验内容，供上机选用。

本书浓缩了作者多年来软件开发和教学实践的经验和体会，写作中注意借鉴近年出版的多种数据结构领域优秀著作的长处。本书有以下特色：

1. 采用 C++语言作为算法描述工具

一方面数据结构可以理解为“数据＋操作”，以往用 C 语言描述其算法不能很好地表达这一思想，而 C++语言正好可以满足这一要求；另一方面使用 C++的类模板，可以提高算法中数据类型的通用性，支持高效的代码重用；再者只要读者学过 C 语言，就能容易阅读和理解。

2. 不需要先有扎实的 C++语言基础

首先书中第 0 章介绍了学习数据结构时，设计算法所需要的 C++程序设计语言预备知识，另外为方便读者对算法的理解和验证，书中尽量避免采用复杂的 C++机制，让读者更多

关注算法本身的设计思想。

3. 所有算法都给出了实现代码

所有算法的代码均在 Visual C++ 6.0 环境下调试通过,读者只需通过主函数来调用它们,就可以边看书,边上机看效果,这样有助于读者更好地理解、把握算法的精髓。

4. 注重培养应用能力

在讲解每种数据结构时都给出了应用实例。

5. 习题完整、系统且典型

习题的选择和设计是教材编写的一个重要方面。每章后面的练习题内容全面,形式多样,有选择题、填空题、简答题、算法应用和程序设计题等。通过这些练习题,读者可以及时地检查和考核对本章内容学习和掌握的情况,教师也可以从中选出一些题作为作业题。

6. 实验内容与教学要求一致

安排并指导学生上机实习对于学好本课程具有重要意义。本书提供的实验内容既有验证性的,也有设计性的。验证性实验的主要内容是将书上的重要数据结构进行上机实现,深化理解和掌握理论知识,这部分实验不需要读者自己设计,只需将给定的方案实现即可。设计性实验的主要内容是针对具体问题,应用已学知识点,自己设计方案,并上机实现,目的是培养读者对数据结构的简单应用能力。

本书第 3～9 章由马石安编写,第 0～2 章和附录由魏文平编写,所有习题和图片由魏文平配置,浙江大学魏梦捷同学参加了资料收集、部分算法实现和程序调试等工作。全书由马石安统一修改、整理和定稿。

为方便教师教学和学生学习,我们还编写了配套的教学用书《数据结构与应用教程(C++版)题解与实验指导》,并提供书中所有源代码和课堂教学的课件等资源(从清华大学出版社网站 http://www.tup.tsinghua.edu.cn 上免费下载),构成一个完整的教学系列。

本教材的编写得到了江汉大学教材建设立项项目的支持。在编写过程中,本书参考和引用了大量书籍和文献资料。在此,向被引用文献的作者及给予本书帮助的所有人士表示衷心感谢,尤其感谢江汉大学领导和同事以及清华大学出版社领导和编辑的大力支持与帮助。

由于作者水平有限,加之时间仓促,书中难免存在缺点与疏漏之处,敬请读者及同行予以批评指正。作者联系方式:wenpingwei@163.com,欢迎各位同仁探讨数据结构教学中的相关问题。

编　者

2012 年 4 月

目 录

第0章 C++程序设计语言预备知识

数据结构的先行课程有高等数学、离散数学、程序设计语言等。本章介绍学习数据结构时，设计算法所需要的C++程序设计语言预备知识，重点是本教材后续章节中涉及的一些内容，用于复习和参考。

0.1 一个简单C++语言程序

C++是在C语言基础上为支持面向对象程序设计而研制的一种编程语言，它对C语言最主要的扩充是引入了面向对象的概念及相应的处理机制，与此同时，还对C语言进行了一系列有效的扩充。由于C++语言兼顾了C语言中的内容，因此在程序结构上与C语言基本相同，只是增加了类的定义和对象。下面我们来看一个典型的C++程序。

【例0.1】 输入矩形的长和宽，计算矩形的面积并输出矩形的详细信息。

```
//Li0_1.cpp
//定义矩形类
#include <iostream>
#include <iomanip>
using namespace std;
//类的声明
class Rectangle
{
public:
     Rectangle(){}
     ~Rectangle(){}
     void SetData(float L,float W);          //设置长、宽值
     float ComputeArea();                    //计算面积
     void PrintArea();                       //输出面积
     void PrintDetails();                    //输出信息
private:
     float length,width,area;                //矩形长、宽、面积
};
//类的实现
void Rectangle::SetData(float L,float W)
{
     length = L;
     width = W;
}
float Rectangle::ComputeArea()
```

```
{
    area = length * width;
    return area;
}
void Rectangle::PrintArea()
{
    cout <<"矩形面积为:"<< area << endl;
}
void Rectangle::PrintDetails()
{
    cout << setw(30)<<" ---- 矩形详细信息 ---- "<< endl;
    cout << setw(15)<<"矩形长:"<< setw(8)<< length << endl;
    cout << setw(15)<<"矩形宽:"<< setw(8)<< width << endl;
    cout << setw(15)<<"矩形面积:"<< setw(8)<< area << endl;
}
//主函数
int main()
{
    int length,width;
    Rectangle rect;                          //声明对象
    cout <<"请输入矩形的长和宽:";
    cin >> length >> width;
    rect.SetData(length,width);
    rect.ComputeArea();
    rect.PrintArea();
    rect.PrintDetails();
    return 0;
}
```

0.1.1 C++程序基本结构

一般情况下,面向对象程序都是由三个部分来构成:类的声明、类的成员的实现和主函数。上述例程 Li0_1.cpp 所示的就是这样的结构。在这种结构中,C++语言中有一个特殊的函数称为主函数(main function)。每一程序都从主函数开始执行,由主函数去激活一个对象的行为,通过这一对象的行为又去激活其他对象的行为。程序中的众多对象共同协作,完成某一任务。

此外,C++程序的基本结构还有两种退化的情形。

一种退化情形是程序中仅有类而没有函数(包括没有主函数)。这些程序通常不是为了直接运行,而是用来构造 C++程序库,然后供编写其他程序时重用。

另一种退化情形是程序中仅有函数而没有类。除主函数外,还可能有一些游离的函数,这些游离的函数不属于任何类。

0.1.2 C++程序基本组成

程序 Li0_1.cpp 是一个最简单的程序,下面分析程序 Li0_1.cpp,以了解 C++程序的基本组成。

1. 文件包含命令

文件包含命令,即 # include 指令,其作用是将某一个源文件的代码并入当前源程序。

其形式有两种：

- #include <文件名>

这种形式一般用于C++提供的库函数。C++编译程序按标准方式搜索，即系统到存放C++库函数头文件的include子目录中寻找要包含的文件。

- #include "文件名"

这种形式一般用于程序员自己开发的模块。C++编译程序首先在当前工作目录中搜索，若没有，再按标准方式搜索。

程序Li0_1.cpp中的第1、2行代码

```
#include <iostream>
#include <iomanip>
```

是编译预处理中的文件包含命令，它的作用是在编译之前将头文件iostream.h和iomanip.h的内容增加到源程序Li0_1.cpp该命令所在的地方。文件iostream.h设置了C++的I/O相关环境，定义了输入/输出流类对象cout与cin等，文件iomanip.h包含了一些格式控制函数，用来对输入/输出进行格式控制，比如程序中所使用的setw()函数就是用来设置输出数据项的域宽的。

C++编译系统提供的头文件有两类，一类是标准的C++库头文件，这些头文件不带".h"；这种写法也适合标准的C库头文件，但是必须使用前缀字符"c"。例如：

```
#include <cmath>                          //相当于#include <math.h>
```

另一类是非标准的C++库头文件，这些头文件带".h"。在连接时，编译系统会根据头文件名自动确定连接哪一个库。但是一定要注意，同一个文件，两种头文件不能混用。

当包含了必要的".h"头文件后，就可以使用其中预定义的内容了。但是使用标准C++库时却不同，在紧接着所有的include指令之后，需要加入下面这一条语句

```
using namespace std;
```

2. 针对名字空间的指令

一个软件往往由多个模块组合而成，其中包括由不同的程序员开发的组件及类库提供的组件，这样不同模块间在对标识符命名时就有可能发生命名冲突，简单地说，就是在不同的模块中使用相同名字表示不同的事物，这样当然会引起程序错误。C++提供名字空间(namespace)来防止命名的冲突。

程序Li0_1.cpp中的语句

```
using  namespace std;
```

是针对名字空间的指令。告诉编译程序此程序中所有的标识符都在std名字空间中，标识符都可以直接使用而不会发生命名的冲突。std涵盖了标准C++所有标识符，本书将例行地使用std。

3. 类的声明与成员的实现

代码段

```
class Rectangle
{
...
};
```

定义了一个名为 Rectangle 的类,类中设置了 3 个数据成员 length、width 和 area 分别用来存储矩形的长、宽和面积值,另外还定义了 4 个函数 SetData()、ComputeArea()、PrintArea()和 PrintDetails(),实现数据的输入、面积的计算、面积的输出与矩形详细信息的显示等功能。

紧接着类的定义后面的代码,是 Rectangle 类中 4 个成员函数的实现。

4. 主函数部分

代码段

```
int main()
{
    ...
    return 0;
}
```

定义了一个名为 main()的主函数,每个 C++程序都必须有且只能有一个主函数 main(),它是程序执行的起点。一般来说,所有函数,包括 main()函数,都必须指明其返回类型。现在编译程序都支持程序 Li0_1.cpp 形式的主函数原型:

```
int main();
```

函数名之前的 int 表示函数需要一个整型返回值,一般用返回 0 表示程序正常结束。本教材中的程序均采用这种原型的主函数。

另外,主函数原型还有如下形式:

```
void main();
```

主函数名之前的 void 表示函数没有返回值,但并不是所有的编译程序都支持这种格式。

5. 注释部分

C++提供了两种注释方式。一种注释方式是从"//"开始,直到行尾,都将被计算机当作注释。另一种是使用"/* …… */",将要注释的部分括起。一般情况下,多行注释使用"/* …… */",而短的注释较多使用"//"。

0.1.3 简单的输入/输出

C++本身没有定义输入/输出操作,而是由一个 I/O 流类库提供的。流类对象 cin 和 cout 分别代表标准的输入设备和输出设备,它们在文件 iostream 中声明。

1. 键盘输入

在 C++中输入操作可理解为从输入流对象中提取数据,故称为提取操作。键盘输入是标准输入,其一般形式可表示为

```
cin>>变量 1>>变量 2>>…>>变量 n;
```

其中，cin 是预定义的标准输入流对象，>>是输入操作符，也称提取运算符。使用这种方法可以从 cin 的输入流中通过提取运算符“>>”获取各种不同类型的数据，给相应类型的变量赋值。这里，输入流的数据项用默认分隔符——空格符进行分隔。

2. 屏幕输出

在 C++中输出操作可理解为将数据插入到输出流对象中，故称为插入操作。屏幕输出是标准输出操作，用来将表达式的结果输出到显示器的屏幕上。其一般形式可表示为：

```
cout <<表达式 1 <<表达式 2 <<…<<表达式 n;
```

其中，cout 是预定义的标准输出流对象，<<是输出操作符，也称插入运算符，用它可以输出各种不同类型的数据。在输出时若要换行，可使用控制符 endl。

运行例 0.1 中的程序，得到如下结果：

```
请输入矩形的长和宽:20 10
矩形面积为:200
            ---- 矩形详细信息 ----
      矩形长:                20
      矩形宽:                10
    矩形面积:               200
```

程序分析：

(1) 语句

```
cin>> length>> width;
```

用来输入矩形的长和宽，可以写成

```
cin>> length;
cin>> width;
```

(2) 语句

```
cout << setw(15)<<"矩形面积:"<< setw(8)<< area << endl;
```

用来输出矩形的面积。其中控制符 endl 用来换行，与 C 语言中的“\n”等价。setw()用来设置输出数据项的域宽，字符串"矩形面积:"输出域宽为 15，矩形面积 area 输出域宽为 8，默认对齐方式为右对齐。

0.2 指针与引用

在 C++中，指针和引用是非常重要的。指针不仅可以在任何需要的时候指向动态分配的对象，而且指针还可以用于对象的共享存取，此外指针对于面向对象编程中多态性的实现也是十分重要的。引用主要用于函数参数传递和返回中。

0.2.1 指针

指针(pointer)是一种特殊的对象，指针的类型是它所指向对象的类型，它的值是它所指向对象的地址值。

1. 指针的值和类型

指针具有一般对象的三个要素：名字、类型和值。指针的命名与一般对象的命名是相同的，它与一般对象的区别在于类型和值上。

指针是一种用来存放某种对象地址值的对象。一个指针存放了哪个对象的地址值，我们就说该指针指向那个对象。可见，指针的值是它所指向那个对象的地址值。指针的类型是它所指向对象的类型。指针的内容便是它所指向对象的值。

2. 指针的定义格式

如同使用一般对象之前必须先用定义语句定义一样，使用一个指针之前也要先定义。具体格式如下所示：

```
<类型> *<指针名1>, *<指针名2>, …;
```

其中，<类型>为指针的基类型，可以是系统提供的基本类型，也可以是用户自定义类型。例如：

```
int *p1;                    //定义一个指向 int 型的指针 p1
char *p2;                   //定义一个指向 char 型的指针 p2
float *p3;                  //定义一个指向 float 型的指针 p3
```

定义一个指针后，系统将为该指针分配一个内存单元。不同类型的指针被分配的内存单元的大小是相同的，因为类型不同的指针所存放的数据值都是内存地址值。

0.2.2 引用

所谓引用(reference)，就是给对象起一个别名，使用该别名可以存取该对象。换句话说，是使新对象和原对象共用一个地址。这样，无论对哪个对象进行修改，其实都是对同一地址的内容进行修改。因而原对象和新对象(规范地说是对象和它的引用)总是具有相同的值。

引用的建立格式如下：

```
<类型说明符> & <引用名> = <对象名>;
```

例如：

```
int a;
int &ta = a;
```

其中，ta 是一个引用名，即 ta 是对象 a 的别名，ta 和 a 都是 int 型的。在这里，要求 a 已经声明或定义。

声明引用语句中的符号 & 是引用运算符，它用在引用名前，说明 ta 是一个引用名。它与地址运算符 & 不同，地址运算符 & 是取地址，它作用在对象名前面。

引用运算符 & 符号与<类型说明符>和<引用名>之间可有空，也可没有空，即下述三种使用方法都是等价的：

```
int& ta = a;
int &ta = a;
int & ta = a;
```