国家示范（骨干）高职院校
重点建设专业优质核心课程系列教材

C++程序设计基础教程

主　编　刘志宝　朱伟华　谢利民
副主编　曹建峰　刘金明　闫　淼

中国水利水电出版社
www.waterpub.com.cn

国家示范（骨干）高职院校重点建设专业优质核心课程系列教材

C++程序设计基础教程

主　编　刘志宝　朱伟华　谢利民

副主编　曹建峰　刘金明　闫　淼

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

本书以 Visual C++典型案例为载体，针对典型任务明确知识目标和技能目标，通过任务分析、知识学习、任务实现、任务拓展等体现“教学做”的教学理念，采用全程导入、全程渐进的方式，由易到难，由仿真到实战组织教学内容。

全书共 11 章，将 C++的基础编程知识、面向对象设计方法、文件操作、异常处理等内容通过案例解析实现。

本书可以作为高职高专计算机及相关专业的基础课教材，也可以作为相关工程技术人员的自学参考书。

本书配有免费电子教案，读者可以从中国水利水电出版社网站以及万水书苑下载，网址为：http://www.waterpub.com.cn/softdown/或 http://www.wsbookshow.com。

图书在版编目（CIP）数据

C++程序设计基础教程 / 刘志宝，朱伟华，谢利民主编. -- 北京 : 中国水利水电出版社，2015.12
国家示范（骨干）高职院校重点建设专业优质核心课程系列教材
ISBN 978-7-5170-3996-9

Ⅰ. ①C… Ⅱ. ①刘… ②朱… ③谢… Ⅲ. ①C语言－程序设计－高等职业教育－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2015)第321334号

策划编辑：石永峰　　责任编辑：张玉玲　　封面设计：李　佳

书　名	国家示范（骨干）高职院校重点建设专业优质核心课程系列教材 C++程序设计基础教程
作　者	主　编　刘志宝　朱伟华　谢利民 副主编　曹建峰　刘金明　闫　淼
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路 1 号 D 座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（发行部）、82562819（万水）
经　售	北京科水图书销售中心（零售） 电话：（010）88383994、63202643、68545874 全国各地新华书店和相关出版物销售网点
排　版	北京万水电子信息有限公司
印　刷	三河市铭浩彩色印装有限公司
规　格	184mm×260mm　16 开本　11 印张　282 千字
版　次	2015 年 12 月第 1 版　2015 年 12 月第 1 次印刷
印　数	0001—2000 册
定　价	24.00 元

前　言

C++是一种使用非常广泛的程序设计语言，是在C语言的基础上发展演变而来的。它是一种静态数据类型检查的支持多范型的通用程序设计语言。C++支持过程化程序设计、数据抽象化、面向对象程序设计、泛型程序设计、基于原则设计等多种程序设计风格。

C++语言既保留了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，可以方便地构造出模拟现实问题的实体和操作，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。用C++编写的程序可读性好，生成的代码质量高。

本书是作者在总结了多年教学经验的基础上编写的，每章既有理论部分又有实践内容，而且以大量的典型案例为载体，让读者巩固知识、消化理解，以达到强化技能培养的目标。本书具有以下特色：

（1）以典型案例为载体。书中各章都含有大量典型案例，而且每个案例都是大家所熟知的经典问题，容易理解；另外，解决问题对应的代码详尽、复用性高。

（2）层次递进的组织结构。本书整体由浅入深、从易到难，依次将C++的基础编程知识、面向对象设计方法、文件操作、异常处理等内容引入，并通过案例解析实现强化学习；针对每章的任务设置也是由易到难依次纵深展开，而且有的任务之间具有紧密的联系。

（3）教学做一体化的教学理念。以本书为载体进行教学时，可以将理论教学和实践教学有机地结合起来，融“教学做”为一体。针对典型任务明确知识目标和技能目标，通过任务分析、知识学习、任务实现、任务拓展等体现“教学做”的教学理念。

本书由刘志宝、朱伟华、谢利民任主编，曹建峰、刘金明、闫淼任副主编。刘志宝编写提纲并统稿，第1章和第5～11章由刘志宝、朱伟华（吉林电子信息职业技术学院）、谢利民（无锡机电高等职业技术学校）编写，第2～4章由曹建峰（无锡职业技术学院）、刘金明（吉林电子信息职业技术学院）、闫淼编写，另外参加本书部分编写工作的还有罗大伟、陈巍（吉林电子信息职业技术学院）等。

在本书编写过程中编者参阅了相关著作、教材和电子资料，在此谨向相关作品的作者表示衷心的感谢。由于时间仓促及编者水平有限，书中错漏之处在所难免，恳请广大读者批评指正。

编　者

2015年12月

目　　录

1 C++程序设计概述

面向对象程序设计是针对开发较大规模的程序而提出的，目的是提高软件开发的效率。学习C++，既要会利用C++进行面向过程的结构化程序设计，又要会利用C++进行面向对象的程序设计。本书既介绍C++在面向过程程序设计中的应用，又介绍C++在面向对象程序设计中的应用。

1.1 C++程序样例

例1.1 简单C++程序，输入一个整数，扩大2倍后输出结果。

程序如下：

```
#include <iostream>                    //引用头文件
using namespace std;                   //使用命名空间
int main()                             //主函数
{
    int n=0;                           //定义整型变量
    cin>>n;                            //输入整数值，存储到变量n中
    n=n*2;                             //值扩大2倍
    cout<<n<<endl;                     //输出n的值
    return 0;                          //主函数返回值
}
```

在运行时会在屏幕上输入一个整数，然后会输出该整数扩大2倍后的数值。

用main代表“主函数”的名字。每一个C++程序都必须有一个main函数，而且只能有一个main函数。main前面的int的作用是声明函数的类型为整型。程序中的“return 0;”语句的作用是向操作系统返回一个零值。如果程序不能正常执行，则会自动向操作系统返回一个非零值，一般为-1。

函数体是由花括号（{}）括起来的，里面的所有C++语句最后都应当有一个分号。

再看程序的第1行“#include <iostream>”，这不是C++的语句，而是C++的一个预处理命令，它以“#”开头以与C++语句相区别，行的末尾没有分号。#include <iostream>是一个“包含命令”，作用是将文件iostream的内容包含到该命令所在的程序文件中，代替该命令行。文件iostream的作用是向程序提供输入或输出时所需要的一些信息。iostream是i、o、stream三个词的组合，从它的形式就可以知道它代表“输入输出流”的意思，由于这类文件都放在程序单元的开头，所以称为“头

文件”（head file）。在程序进行编译时，先对所有的预处理命令进行处理，用头文件的具体内容代替 #include 命令行，然后再对该程序单元进行整体编译。

程序的第 2 行“using namespace std;”的意思是“使用命名空间 std”。C++标准库中的类和函数是在命名空间 std 中声明的，因此程序中如果需要用到 C++标准库（此时就需要用#include 命令行），就需要用“using namespace std;”进行声明，表示要用到命名空间 std 中的内容。

“//”后面的内容属于注释部分，不属于程序语句，即不编译、不执行。

main 函数内的程序语句的具体含义见注释。

例 1.2 输入两个整数，计算两个整数的和。

程序如下：

```
#include <iostream>                          //引用头文件
using namespace std;                         //使用命名空间
int add(int x,int y)                         //定义 add 函数，函数值为整型，形式参数 x、y 为整型
{
    int z;                                   //变量声明，定义本函数中用到的变量 z 为整型
    z=x+y;                                   //计算两个数的加和
    return z;                                //返回结果
}
int main()                                   //主函数
{
    int a,b,m;                               //变量声明
    cin>>a>>b;                               //输入变量 a 和 b 的值
    m=add(a,b);                              //调用 max 函数，将得到的值赋给 m
    cout<<a<<"+"<<b<<"="<<m<<endl;           //输出和 m 的值
    return 0;                                //如果程序正常结束，向操作系统返回一个零值
}
```

本程序包括两个函数：主函数 main 和被调用的函数 add。

程序运行情况如下：

```
4  5   （回车）（输入 4 和 5 给 a 和 b）
4+5=9  （输出加和）
```

注意输入的两个数间用至少一个空格间隔，不能以逗号或其他符号间隔。

在上面的程序中，add 函数出现在 main 函数之前，因此在 main 函数中调用 add 函数时编译系统能识别 add 是已定义的函数名。如果把两个函数的位置对换一下，即先写 main 函数，后写 add 函数，这时在编译 main 函数遇到 add 时，编译系统无法知道 add 代表什么含义，因而无法编译，按出错处理。为了解决这个问题，在主函数调用该函数前需要对被调用函数进行声明。上面的程序可以改写如下：

```
#include <iostream>                          //引用头文件
using namespace std;                         //使用命名空间
int add(int x,int y);                        //声明 add 函数
int main()                                   //主函数
{
    int a,b,m;                               //变量声明
    cin>>a>>b;                               //输入变量 a 和 b 的值
    m=add(a,b);                              //调用 max 函数，将得到的值赋给 m
    cout<<a<<"+"<<b<<"="<<m<<endl;           //输出和 m 的值
    return 0;                                //如果程序正常结束，向操作系统返回一个零值
}
int add(int x,int y)                         //定义 add 函数，函数值为整型，形式参数 x、y 为整型
{
```

```
    int z;                                  //变量声明，定义本函数中用到的变量 z 为整型
    z=x+y;                                  //计算两个数的加和
    return z;                               //返回结果
}
```

例 1.3　包含类的 C++程序。

程序如下：

```
#include <iostream>
using namespace std;
class Student                               //声明一个类，类名为 Student
{
    private:                                //以下为类中的私有部分
    int num;                                //私有变量 num
    int score;                              //私有变量 score
    public:                                 //以下为类中的公有部分
    void setdata()                          //定义公有函数 setdata
    {
        cin>>num;                           //输入 num 的值
        cin>>score;                         //输入 score 的值
    }
    void display()                          //定义公有函数 display
    {
        cout<<"num="<<num<<endl;            //输出 num 的值
        cout<<"score="<<score<<endl;        //输出 score 的值
    }
};                                          //注意类的声明用分号结束
int main()                                  //主函数首部
{
    Student stud1,stud2;                    //定义 stud1 和 stud2 为 Student 类的变量，称为对象
    stud1.setdata();                        //调用对象 stud1 的 setdata 函数
    stud2.setdata();                        //调用对象 stud2 的 setdata 函数
    stud1.display();                        //调用对象 stud1 的 display 函数
    stud2.display();                        //调用对象 stud2 的 display 函数
    return 0;
}
```

在一个类中包含两种成员：数据和函数，分别称为数据成员和成员函数。在 C++中把一组数据和有权调用这些数据的函数封装在一起，组成一种称为“类（class）”的数据结构。在上面的程序中，数据成员 num、score 和成员函数 setdata、display 组成了一个名为 Student 的“类”类型。成员函数是用来对数据成员进行操作的。也就是说，一个类是由一批数据以及对其操作的函数组成的。

类可以体现数据的封装性和信息隐藏。在上面的程序中，在声明 Student 类时，把类中的数据和函数分为两大类：private（私有的）和 public（公有的）。把全部数据（num、score）指定为私有的，把全部函数（setdata、display）指定为公有的。在大多数情况下，会把所有数据指定为私有的，以实现信息隐藏。

具有“类”类型特征的变量称为“对象”（object）。

程序中 main 部分是主函数。

程序运行情况如下：

```
1001  98.5  （回车）（输入学生 1 的学号和成绩）
1002  76.5  （回车）（输入学生 2 的学号和成绩）
num=1001  （输出学生 1 的学号）
score=98.5  （输出学生 1 的成绩）
```

```
num=1002   （输出学生 2 的学号）
score=76.5   （输出学生 2 的成绩）
```

C++程序的结构和书写格式归纳如下：

（1）一个 C++程序可以由一个程序单位或多个程序单位构成，每一个程序单位作为一个文件。在程序编译时，编译系统分别对各个文件进行编译，因此一个文件是一个编译单元。

（2）在一个程序单位中，可以包括以下几个部分：

- 预处理命令。前面的 4 个程序中都包括#include 命令。
- 全局声明部分（在函数外的声明部分）。在这部分中包括对用户自己定义的数据类型的声明和程序中所用到的变量的定义。
- 函数。函数是实现操作的部分，因此函数是程序中必须有的和最基本的组成部分。每一个程序必须包括一个或多个函数，其中必须有（且只能有）一个主函数（main 函数）。

但是并不要求每一个程序文件都必须具有以上 3 个部分，可以缺少某些部分（包括函数）。

（3）一个函数由两部分组成：函数首部和函数体。

1）函数首部，即函数的第一行，包括函数名、函数类型、函数属性、函数参数（形参）名、参数类型。一个函数名后面必须跟一对圆括号，函数参数可以缺省，如 int main()。

2）函数体，即函数首部下面的大括号内的部分。如果在一个函数中有多个大括号，则最外层的一对{}为函数体的范围。

函数体一般包括以下三个部分：

- 局部声明部分（在函数内的声明部分）：包括对本函数中所用到的类型、函数的声明和变量的定义。
- 对数据的声明：既可以放在函数之外（其作用范围是全局的），也可以放在函数内（其作用范围是局部的，只在本函数内有效）。
- 执行部分：由若干个执行语句组成，用来进行有关的操作，以实现函数的功能。

（4）语句包括两类：一类是声明语句，另一类是执行语句。C++对每一种语句赋予一种特定的功能。语句是实现操作的基本成分，显然没有语句的函数是没有意义的。C++语句必须以分号结束。

（5）一个 C++程序总是从 main 函数开始执行的，而不论 main 函数在整个程序中的位置如何。

（6）类（class）是 C++新增加的重要的数据类型，是 C++对 C 的最重要的发展。有了类，就可以实现面向对象程序设计方法中的封装、信息隐藏、继承、派生、多态等功能。在一个类中可以包括数据成员和成员函数，它们可以被指定为私有的（private）和公有的（public）属性。私有的数据成员和成员函数只能被本类的成员函数所调用。

（7）C++程序书写格式自由，一行内可以写几个语句，一个语句可以分写在多行上。C++程序没有行号。

（8）一个好的、有使用价值的源程序都应当加上必要的注释，以增加程序的可读性。C++还保留了 C 语言的注释形式，可以用“/*……*/”对 C++程序中的任何部分作注释。在“/*”和“*/”之间的全部内容作为注释。

用“//”作注释时，有效范围只有一行，即本行有效，不能跨行。而用“/*……*/”作注释时有效范围为多行，只要在开始处有一个“/*”，在最后一行结束处有一个“*/”即可。因此，一般习惯是：内容较少的简单注释常用“//”，内容较长的注释常用“/*……*/”。

1.2 C++程序的上机步骤

一个程序从编写到最后得到运行结果一般要经历以下步骤：

（1）用 C++语言编写程序。

用高级语言编写的程序称为“源程序”（source program）。C++的源程序是以.cpp 作为后缀的（cpp 是 c plus plus 的缩写）。

（2）对源程序进行编译。

为了使计算机能执行高级语言源程序，必须先用一种称为“编译器（complier）”的软件（也称编译系统）把源程序翻译成二进制形式的“目标程序（object program）”。

编译是以源程序文件为单位分别进行的。目标程序一般以.obj 或.o 作为后缀（object 的缩写）。编译的作用是对源程序进行词法检查和语法检查。编译时对文件中的全部内容进行检查，编译结束后会显示出所有的编译出错信息。一般编译系统给出的出错信息分为两种：一种是错误（error），一种是警告（warning）。

（3）将目标文件连接。

在改正所有的错误并全部通过编译后得到一个或多个目标文件，此时要用系统提供的“连接程序（linker）”将一个程序的所有目标程序和系统的库文件以及系统提供的其他信息连接起来，最终形成一个可执行的二进制文件，它的后缀是.exe，是可以直接执行的。

（4）运行程序。

运行最终形成的可执行的二进制文件（.exe 文件），得到运行结果。

（5）分析运行结果。

如果运行结果不正确，应检查程序或算法是否有问题。

在了解了 C++语言的初步知识后，读者最好尽快在计算机上编译和运行 C++程序，以加深对 C++程序的认识并初步掌握 C++的上机操作。

本书中的程序代码都是在 Visual C++ 6.0 环境下编辑、调试、运行的，下面简要介绍如何使用 Visual C++ 6.0 环境进行代码的编写、编译、执行等操作。

（1）启动 Visual C++ 6.0 环境，操作如图 1-1 和图 1-2 所示。

图 1-1 启动 Visual C++ 6.0

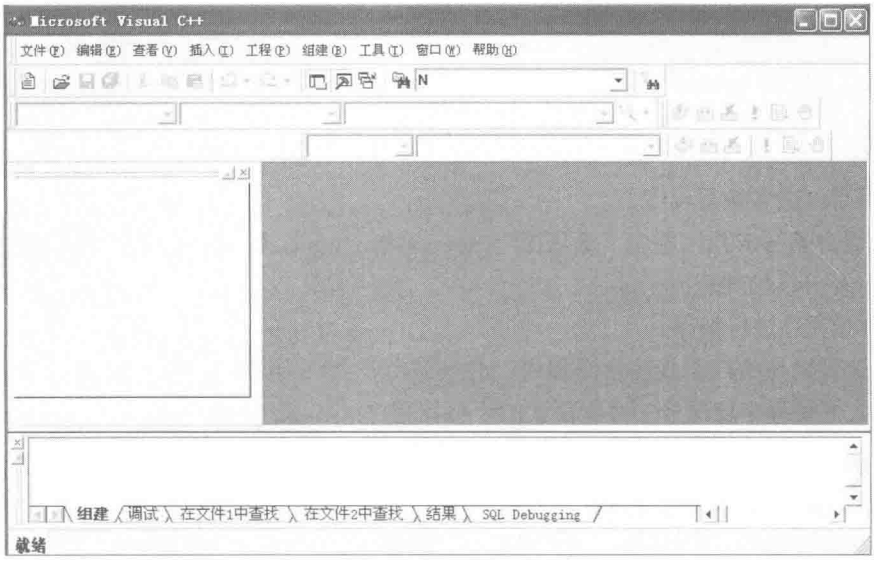

图 1-2　Visual C++ 6.0 界面

（2）单击“文件”菜单中的“新建”，弹出“新建”对话框，选择 Win32 Console Application，输入工程名称，如图 1-3 所示。

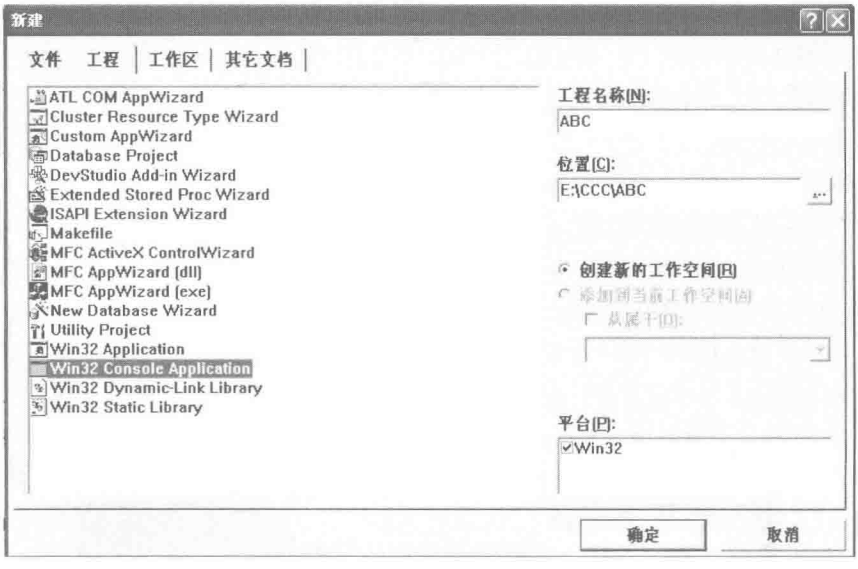

图 1-3　“新建”对话框

单击“确定”按钮，后面的对话框默认确定即可。

（3）再次单击“文件”菜单中的“新建”，弹出“新建”对话框，选择 C++ Source File，输入文件名，如图 1-4 所示。

（4）在“*.cpp”文件中编辑 C++源文件，如图 1-5 所示。

（5）单击“组建”菜单中的编译项进行源程序编译。如果编译成功，则单击“组建”菜单中的执行项进行程序的运行操作。

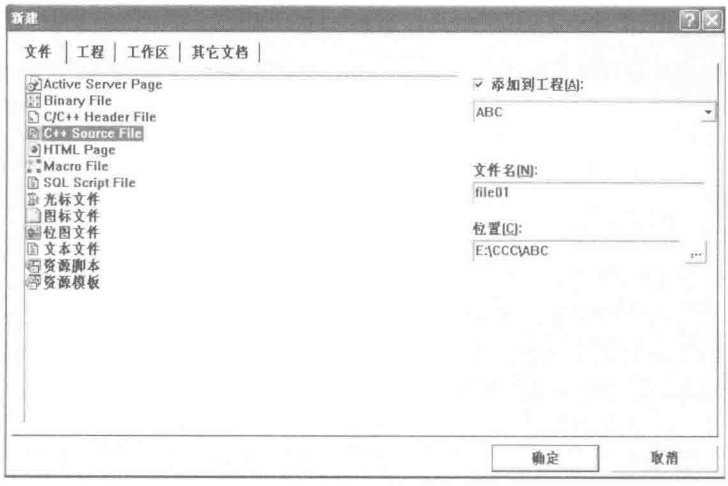

图 1-4 “新建”对话框

图 1-5 编辑源程序

1.3 数据的标准输入输出

在 C++中，输入输出流被定义为类。C++的 I/O 库中的类称为流类（stream class），用流类定义的对象称为流对象。

cout 和 cin 并不是 C++语言中提供的语句，它们是 iostream 类的对象，在未学习类和对象时，在不至于引起误解的前提下，为叙述方便，把它们称为 cout 语句和 cin 语句。

1.3.1 cout 输出流对象

ostream 类定义了一个输出流对象 cout。

cout 是 console output 的缩写，意为在控制台（终端显示器）上输出。

（1）cout 不是 C++预定义的关键字，而是 ostream 流类的对象，在 iostream 中定义。

（2）用“cout<<”输出基本类型的数据时，可以不必考虑数据是什么类型，系统会判断数据

的类型，并根据其类型选择调用与之匹配的运算符重载函数。

（3）cout 流在内存中对应开辟了一个缓冲区，用来存放流中的数据，当向 cout 流插入一个 endl 时，不论缓冲区是否已满都立即输出流中所有的数据，然后插入一个换行符并刷新流（清空缓冲区）。

（4）在 iostream 中只对“<<”和“>>”运算符用于标准类型数据的输入输出进行了重载，未对用户声明的类型数据的输入输出进行重载。

回顾例 1.2 程序中的语句：

```
cout<<a<<"+"<<b<<"="<<m<<endl;    //输出和 m 的值
```

注意一个 cout 可以连续输出多个数据。如果后面带有“<<endl”则具有输出后换行作用，如果输出数据后不希望换行，则 cout 后面没有“<<endl”即可。

1.3.2 cin 输入流对象

cin 是 iostream 类的对象，它从标准输入设备（键盘）获取数据，程序中的变量通过流提取符“>>”从流中提取数据。流提取符“>>”从流中提取数据时通常跳过输入流中的空格、Tab 键、换行符等空白字符。

注意：只有在输入完数据再按回车键后，该行数据才被送入键盘缓冲区，形成输入流，提取运算符“>>”才能从中提取数据。需要注意保证从流中读取数据能正常进行。

例 1.4 通过测试 cin 的真值判断流对象是否处于正常状态。

```
#include <iostream>
using namespace std;
int main()
{
    float grade;
    cout<<"enter grade:";
    while(cin>>grade)                    //能从 cin 流读取数据
    {
        if(grade>=60)
            cout<<grade<<"通过"<<endl;
        if(grade<60)
            cout<<grade<<"没通过"<<endl;
        cout<<"enter grade:";
    }
    cout<<"The end."<<endl;
    return 0;
}
```

运行情况如下：

```
enter grade: 67（回车）
67 通过
enter grade: 56（回车）
56 没通过
enter grade: ^Z（回车）                  //键入文件结束符
The end.
```

1.4 基本数据类型

计算机处理的对象是数据，而数据是以某种特定的形式存在的（例如整数、浮点数、字符等形

式)。不同的数据之间往往还存在某些联系(例如由若干个整数组成一个整数数组)。数据结构指的是数据的组织形式。例如数组就是一种数据结构。不同的计算机语言所允许使用的数据结构是不同的。处理同一类问题,如果数据结构不同,算法也会不同。例如,对 10 个整数排序和对包含 10 个元素的整型数组排序的算法是不同的。

C++可以使用的数据类型如图 1-6 所示。

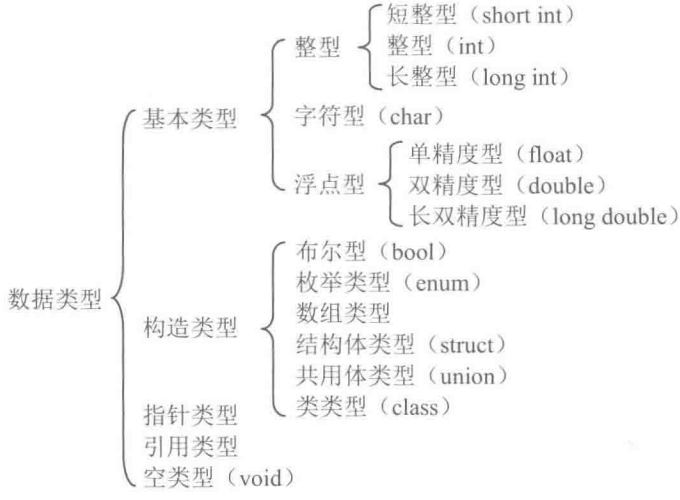

图 1-6 数据类型

C++的数据包括常量与变量,常量与变量都具有类型。由以上这些数据类型还可以构成更复杂的数据结构。例如利用指针和结构体类型可以构成表、树、栈等复杂的数据结构。

说明:

(1)整型数据分为长整型(long int)、整型(int)和短整型(short int)。在 int 前面加 long 和 short 分别表示长整型和短整型。

(2)数据的存储方式为按二进制数形式存储。

(3)在整型符号 int 和字符型符号 char 的前面可以加修饰符 signed(表示“有符号”)或 unsigned(表示“无符号”)。如果指定为 signed,则数值以补码形式存放,存储单元中的最高位(bit)用来表示数值的符号;如果指定为 unsigned,则数值没有符号,全部二进制位都用来表示数值本身。

(4)浮点型(又称实型)数据分为单精度(float)、双精度(double)和长双精度(long double)3 种,在 Visual C++ 6.0 中,对 float 提供 6 位有效数字,对 double 提供 15 位有效数字,并且 float 和 double 的数值范围不同。对 float 分配 4 个字节,对 double 和 long double 分配 8 个字节。

(5)类型关键字中有些是等价的,如 short 和 short int 等价、unsigned int 和 unsigned 等价。

1.5 常量与变量

1.5.1 常量

常量是在程序运行时不会被修改的量。常量分为不同的类型,如 25、0、-8 为整型常量,6.8、

-7.89 为实型常量，'a'、'b'为字符常量。常量一般从其字面形式即可判断，这种常量称为字面常量或直接常量。

1. 整型常量

整型数据可以分为 int、short int、long int、unsigned int、unsigned short、unsigned long 等类别，整型常量也分为以上类别。为什么将数值常量分为不同的类别呢？因为在进行赋值或函数的参数虚实结合时要求数据类型匹配。

那么，一个整型常量怎样从字面上区分为以上的类别呢？

- 一个整数，如果其值在-32768～+32767 范围内，则认为它是 short int 型，它可以赋值给 short int 型、int 型和 long int 型变量。
- 一个整数，如果其值超过了上述范围，而在-2147483648～+2147483647 范围内，则认为它是 long int 型，可以将它赋值给一个 int 型或 long int 型变量。
- 如果某一计算机系统的 C++版本确定 int 与 long int 型数据在内存中占据的长度相同，则它们能够表示的数值的范围相同。因此，一个 int 型常量也同时是一个 long int 型常量，可以赋给 int 型或 long int 型变量。
- 常量无 unsigned 型。但一个非负值的整数可以赋值给 unsigned 整型变量，只要它的范围不超过变量的取值范围即可。

一个整型常量可以用以下三种不同的方式表示：

- 十进制整数。如 1357、-432、0 等。在一个整型常量后面加一个字母 l 或 L，则认为是 long int 型常量，例如 123L、421L、0L 等，这往往用于函数调用中。如果函数的形参为 long int，则要求实参也为 long int 型，此时用 123 作实参不行，而要用 123L 作实参。
- 八进制整数。在常数的开头加一个数字 0，就表示这是以八进制数形式表示的常数。如 020 表示这是八进制数 20，即$(20)_8$，它相当于十进制数 16。
- 十六进制整数。在常数的开头加一个数字 0 和一个英文字母 X（或 x），就表示这是以十六进制数形式表示的常数。如 0X20 表示这是十六进制数 20，即$(20)_{16}$，它相当于十进制数 32。

2. 浮点型常量

一个浮点数可以用以下两种不同的方式来表示：

- 十进制小数形式。如 21.456、-7.98 等，它一般由整数部分和小数部分组成，可以省略其中之一（如 78.或.06、.0），但不能二者皆省略。C++编译系统把用这种形式表示的浮点数一律按双精度常量处理，在内存中占 8 个字节。如果在实数的数字之后加字母 F 或 f，表示此数为单精度浮点数，如 1234F、-43f，占 4 个字节；如果加字母 L 或 l，表示此数为长双精度数（long double），在 GCC 中占 12 个字节，在 Visual C++ 6.0 中占 8 个字节。
- 指数形式（即浮点形式）。一个浮点数可以写成指数形式，如 3.14159 可以表示为 0.314159×10^{1}、3.14159×10^{0}、31.4159×10^{-1}、314.159×10^{-2} 等形式，在程序中应表示为 0.314159e1、3.14159e0、31.4159e-1、314.159e-2，用字母 e 表示其后的数是以 10 为底的幂，如 e12 表示 10^{12}。要求 E（或者 e）前面必须满足数字小于 1，同时 E（或者 e）后面应该是整数。

3. 字符型常量

用单引号括起来的一个字符就是字符型常量。如'a'、'#'、'%'、'D'都是合法的字符常量，在内存