



高职高专计算机精品课程系列规划教材

C++语言程序设计

詹发荣 编著



中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

高职高专计算机精品课程系列规划教材

C++语言程序设计

詹发荣 编著

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

内 容 简 介

本书是一本非常适合初学者学习 C++ 编程的入门指导书, 对学习面向对象程序设计有一定的帮助作用。全书共分 9 章, 第 1 章介绍了 C++ 程序的基本结构和运行环境; 第 2 章介绍了 C++ 语言中的基本数据类型和表达式以及表达式的副作用; 第 3 章介绍了程序设计中的 3 种基本控制结构与语句; 第 4 章介绍了数组类型及其定义和使用方法, 字符数组和字符串之间的关系; 第 5 章介绍了函数的概念, 重载函数和函数模板, 函数的作用域; 第 6 章介绍了指针的概念以及与指针相关的动态内存管理的概念; 第 7 章介绍了用户自定义数据类型, 即结构类型和联合类型的定义, 以及结构的重要应用即链表的建立与遍历; 第 8 章介绍了面向对象中的两个重要概念, 即类与对象, 初步引入了面向对象的思想; 第 9 章介绍了用流类实现数据的输入/输出, 以及各流类之间的继承关系。

本书适合作为高职高专院校“C++ 语言程序设计”课程的教材, 也可作为面向对象程序设计的入门参考书籍。

图书在版编目 (CIP) 数据

C++ 语言程序设计/詹发荣编著. —北京: 中国铁道出版社, 2009. 4

(高职高专计算机精品课程系列规划教材)

ISBN 978-7-113-09867-4

I. C… II. 詹… 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 (2009) 第 052336 号

书 名: C++ 语言程序设计

作 者: 詹发荣 编著

策划编辑: 秦绪好 辛 杰

责任编辑: 黄园园

封面设计: 付 巍

责任印制: 李 佳

编辑部电话: (010) 63583215

封面制作: 白 雪

出版发行: 中国铁道出版社 (北京市宣武区右安门西街 8 号 邮政编码: 100054)

印 刷: 北京新魏印刷厂

版 次: 2009 年 5 月第 1 版 2009 年 5 月第 1 次印刷

开 本: 787mm×1092mm 1/16 印张: 20.25 字数: 476 千

印 数: 4 000 册

书 号: ISBN 978-7-113-09867-4/TP·3203

定 价: 29.00 元

版权所有 侵权必究

本书封面贴有中国铁道出版社激光防伪标签, 无标签者不得销售

凡购买铁道版的图书, 如有缺页、倒页、脱页者, 请与本社计算机图书批销部调换。

C++语言保留了 C 语言的精华部分并提供了全面的面向对象的编程支持,使得开发的程序结构更加清晰、更容易维护和扩充,同时又不失其高效性。为了让从未学习过 C 语言的读者尽快掌握 C++语言编程技术,本书以基本概念结合具体实例由浅入深地逐步介绍,特别是对于理解困难的知识点,在介绍时将复杂问题分解成最小的概念问题,力求为读者提供一个思路清晰、举例恰当准确、知识系统完整的编程语言体系。

全书共分 9 章,第 1 章,首先通过一个完整的简单编程实例,引出了 C++语言程序的基本结构和语法规则,使读者对 C++程序有一个感性认识;为了帮助读者对后续内容进行编程并上机检验其正确性,本教材以 Microsoft Visual C++ 6.0 (简称 VC++ 6.0) 为程序开发环境,介绍了 C++程序的上机操作步骤;第 2 章,设计程序的主要目的是处理数据对象,但现实生活中的数据有各种各样的表现形式和不同的类型,为此本章首先介绍了 C++语言中的所有基本数据类型,并以由数据对象构成表达式,由表达式在其末尾添加分号构成语句,由语句构成函数,由函数构成程序文件,由程序文件构成程序这样一条主线帮助读者理顺和建立构成一个完整 C++程序的概念;第 3 章,针对顺序、分支和循环问题的处理,介绍了 C++中提供的 3 种流程控制结构及其实现方法,这是学习程序设计的读者必须掌握的内容;第 4 章,针对大量同类型的数据处理问题,介绍了数组的概念,包括一维数组和二维数组的定义及使用;第 5 章,介绍了编制解决复杂问题的程序时 C++提供的函数的概念,函数是构成程序的基本组成单位;第 6 章,介绍 C++中的精髓部分,即指针的概念,灵活运用指针和动态内存空间管理是解决特殊问题的利器;第 7 章,为了将同一实体的不同属性有机地结合在一起,介绍了 C++提供的结构类型的定义格式和应用,同时为了保证不同属性共用同一内存空间,介绍了联合类型的定义格式和应用;第 8 章,介绍了面向对象程序设计的思想,并说明与传统的面向过程程序设计方法的不同之处及优点,对象的继承和模板等内容;第 9 章,介绍了外部设备与内存进行数据交换的处理机制,即使用 C++系统提供的类库解决输入/输出问题。

本书的特点是概念性和实例性相结合,对于不容易理解的概念通过简单的实例加以解释和说明,书中部分章节以大量图例帮助读者理解知识点。

由于编者水平有限,疏漏和不足之处在所难免。如果有任何问题或建议,请发电子邮件到 nxzhanfr@126.com,期待读者的宝贵意见。

詹发荣
2009 年 3 月

目 录

CONTENTS

第 1 章 C++程序的基本结构和运行环境	1
1.1 C++语言简介	1
1.1.1 C++语言的特点	1
1.1.2 C++和 C 的关系和区别	2
1.2 C++程序的基本结构和语法规则	2
1.2.1 一个简单的 C++程序	3
1.2.2 C++程序结构及语法规则	3
1.3 程序的编辑、编译、连接和运行	8
1.3.1 开发 C++应用程序的步骤	9
1.3.2 VC++ 6.0 集成开发环境简介	9
第 2 章 筑牢 C++语言编程基础	20
2.1 C++数据类型	20
2.1.1 C++数据类型的分类	20
2.1.2 C++程序中数据的表现形式	21
2.2 整数类型数据	21
2.2.1 各种整型数据的基本情况	21
2.2.2 整数常量的表示	22
2.2.3 整型变量的定义和初始化	23
2.3 字符型数据	24
2.3.1 字符型数据的基本情况	24
2.3.2 字符型常量的表示	24
2.3.3 字符型和整型的关系	26
2.4 枚举型数据	26
2.5 实型数据	28
2.5.1 各种实型数据的基本情况	28
2.5.2 实型常量的表示和实型变量的定义和初始化	29
2.6 符号常量与常值变量	29
2.6.1 用符号代替常量的两种定义方法	30
2.6.2 符号常量应用举例	30
2.6.3 使用符号常量的优点及注意事项	31
2.7 数值表达式	31
2.7.1 算术表达式	31

2.7.2	位运算表达式	33
2.7.3	增 1/减 1 运算符.....	34
2.7.4	赋值表达式	35
2.7.5	数学表达式和 C++表达式	36
2.7.6	类型的自动转换和强制转换	37
2.8	逻辑型数据与逻辑表达式	38
2.8.1	逻辑型数据	39
2.8.2	逻辑表达式	39
2.8.3	有关条件的表达	40
2.8.4	逻辑型数据与其他类型数据的关系	41
2.8.5	条件表达式	42
2.9	自定义类型修饰符	42
2.9.1	自定义类型修饰符的定义格式	43
2.9.2	自定义类型修饰符的作用	43
2.10	表达式的副作用与表达式语句	43
2.10.1	表达式的副作用	43
2.10.2	表达式副作用的应用	44
第 3 章	灵活控制 C++程序流程	46
3.1	流程控制与程序结构	46
3.2	条件分支结构	47
3.2.1	if 语句	47
3.2.2	switch 多分支结构语句	53
3.3	循环结构	54
3.3.1	C++中循环流程控制	54
3.3.2	while 循环	55
3.3.3	do...while 循环	56
3.3.4	for 循环	57
3.3.5	循环嵌套	58
3.3.6	break 和 continue 语句的使用	59
3.4	其他流程控制	60
3.4.1	goto 语句	60
3.4.2	return 语句	61
第 4 章	解决复杂问题的工具——数组	63
4.1	一维数组	63
4.1.1	一维数组的引入	63
4.1.2	一维数组	63
4.2	多维数组	68
4.2.1	二维数组	68

4.2.2	三维数组	70
4.2.3	二维数组元素的访问	71
4.3	字符数组与字符串	72
4.3.1	字符串的基本概念	72
4.3.2	字符串存储空间的分配	73
4.3.3	常用的字符串操作函数	75
4.3.4	字符串处理应用举例	78
第 5 章	解决应用问题的手段——函数	80
5.1	函数的定义	80
5.1.1	C++中函数的定义	80
5.1.2	函数的分类	81
5.1.3	函数返回值的默认类型	83
5.2	函数的调用	84
5.2.1	函数调用格式及调用方式	84
5.2.2	函数的递归调用	87
5.3	函数原型与头文件	89
5.3.1	函数原型	89
5.3.2	头文件	92
5.4	函数调用中的参数传递	93
5.4.1	值传递和地址传递	93
5.4.2	数组参数	94
5.4.3	可选参数	97
5.5	内联函数	98
5.6	函数重载与名字混成	99
5.6.1	函数重载	99
5.6.2	名字混成	101
5.7	函数和变量的作用域	102
5.7.1	函数的作用域	102
5.7.2	变量的作用域和生存期	105
5.7.3	符号常量的作用域和生存期	112
5.8	函数模板	113
5.8.1	函数模板的引入	113
5.8.2	函数模板的定义	114
5.8.3	函数模板的实例化	114
第 6 章	指针、引用和动态空间管理	120
6.1	指针的概念和指针变量的定义	120
6.1.1	指针变量的定义和初始化	121
6.1.2	常值指针	124

6.2	指针的基本操作.....	126
6.2.1	指针赋值.....	126
6.2.2	取变量的地址.....	127
6.2.3	间接访问.....	127
6.2.4	判断指针是否是空指针.....	128
6.2.5	计算两地址间数据单元的个数.....	129
6.2.6	指针移动.....	129
6.2.7	指针表达式的副作用.....	133
6.2.8	指针类型的强制转换.....	134
6.2.9	指针操作符的综合应用.....	134
6.3	指针与数组.....	136
6.3.1	一维数组元素的访问方式.....	136
6.3.2	二维数组元素的访问方式.....	138
6.3.3	关于指向数组的指针.....	143
6.3.4	字符指针与字符串.....	145
6.3.5	指针数组与命令行参数.....	147
6.3.6	数组参数实际上是指针.....	153
6.4	指针与函数.....	156
6.4.1	指针参数.....	156
6.4.2	指针函数.....	157
6.4.3	函数指针.....	159
6.5	引用.....	161
6.5.1	引用的概念.....	161
6.5.2	指针和引用的比较.....	164
6.5.3	引用使用举例.....	166
6.6	动态空间管理.....	169
6.6.1	动态空间的引入.....	169
6.6.2	动态空间的管理.....	170
第7章	结构类型与联合类型.....	175
7.1	结构的定义.....	175
7.1.1	结构的基本概念.....	175
7.1.2	结构类型的定义格式.....	176
7.2	结构变量的定义和初始化.....	179
7.3	结构成员的访问操作.....	181
7.3.1	赋值运算.....	181
7.3.2	直接成员运算与间接成员运算.....	184
7.3.3	结构应用举例.....	185
7.4	结构与函数.....	189

7.4.1	结构作为函数的参数类型.....	189
7.4.2	结构作为函数的返回值类型.....	191
7.5	结构与链表.....	193
7.5.1	链表的基本概念.....	193
7.5.2	结构类型的应用——链表.....	194
7.6	结构与操作符重载.....	198
7.7	联合.....	204
7.7.1	联合的基本概念.....	204
7.7.2	联合类型使用举例.....	207
第 8 章	学会应用面向对象编程.....	210
8.1	类.....	210
8.1.1	类和对象的概念.....	210
8.1.2	类的定义.....	210
8.1.3	类的使用说明.....	216
8.2	构造函数.....	224
8.2.1	构造函数的定义.....	224
8.2.2	无参构造函数和有参构造函数.....	225
8.2.3	复制构造函数.....	230
8.2.4	赋值操作符的重载.....	235
8.2.5	构造函数中的初始化表.....	237
8.3	析构函数.....	238
8.3.1	析构函数的定义.....	239
8.3.2	默认析构函数.....	239
8.3.3	析构函数研究.....	239
8.4	友元函数和友元类.....	243
8.4.1	友元函数.....	243
8.4.2	友元类.....	250
8.5	类的继承.....	254
8.5.1	类继承的概念.....	254
8.5.2	单继承的实现方法.....	255
8.5.3	多继承的实现方法.....	270
8.6	类的虚函数与多态性.....	276
8.6.1	虚函数和多态性.....	277
8.6.2	虚函数和多态性应用举例.....	278
8.7	类的静态成员.....	281
8.7.1	静态成员的基本概念.....	281
8.7.2	静态成员的应用举例.....	282
8.8	模板类.....	285

第 1 章 C++程序的基本结构和运行环境

C++语言是在 C 语言基础上发展起来的支持面向对象思想的高级语言，其程序的基本结构与 C 语言程序相同，语法也极为相近，但运行环境要使用能支持面向对象功能的程序开发工具，如 Microsoft Visual C++ 6.0 和 C++ Builder 6.0。

1.1 C++语言简介

C++语言是在 C 程序语言基础上进化而来的，C 语言功能强大、风格简洁、适用于大多数的系统程序设计等优点被 C++保留下来，同时 C++完全支持面向对象的程序设计。

C++语言还与其他高级语言有密切的联系，如 C++中重要的组成部分——类和继承是从 Simula 语言中引进的，运算符重载的机制是借用 Algol 60 的。

C++语言是 20 世纪 80 年代由 AT&T 公司贝尔实验室的 Bjarne Stroustrup 博士对 C 语言进行改进和扩充而开发的程序设计语言，最初称为“带类的 C”，后取名为 C++，表示一种高效率的 C。

C++语言先后经历了 3 次修改，1994 年制定了 ANSI C++标准草案，并经过不断完善成为目前的 C++。在此期间，著名的软件公司 Borland 于 1986 年开发了 Turbo C++程序语言，而后又推出了 Borland C++。

1.1.1 C++语言的特点

C++语言具有面向对象程序设计语言的特点，主要表现在以下几个方面。

1. C++语言支持数据封装

在 C++语言中，类是支持数据封装的工具，对象则是数据封装的实现。

在面向对象的程序设计中，将数据和对数据进行合法操作的函数封装在一起，作为一个类来定义。另外，封装还提供一种对数据访问严格控制的机制。数据将被隐藏在封装体中，该封装体通过操作接口与外界交换信息。

C++语言中的类是数据和操作函数的封装体，即

类 = 数据 + 操作函数

2. C++类中包含有私有成员、公有成员和保护成员

- 私有成员：只有在类中说明的函数才能访问该类的成员。
- 公有成员：类外函数可以访问的类内成员。公有成员成为该类与外部的接口。

- 保护成员：这类成员只能被该类的成员函数和友元函数以及由该类的派生类的成员函数和友元函数所访问。

3. C++通过函数调用来处理对象

C++是通过面向对象发送消息来处理对象的，每个对象根据所接收到的消息的性质来决定需要采取的行动，以响应这个消息。送到一个对象的所有可能的消息在对象的类的描述中都需要定义，即对每个可能的消息给出一个相应的方法。方法是在类定义中使用函数来定义，使用一种类似于函数调用的机制，把消息发送到一个对象上。

4. C++中允许友元破坏封装性

类中的私有成员一般是不允许类外任何函数访问的，但友元可以访问类的私有成员。友元可以是在类外定义的函数（友元函数），也可以是在类外定义整个类（友元类）。友元打破了类的封装性，它是 C++面向对象的又一个重要特征。

5. C++支持继承性

C++中允许单继承和多继承。一个类可以根据需要生成派生类。派生类继承了基类的所有方法，另外派生类自身还可以定义所需要的不包含在父类中的新方法。一个子类的每个对象包含从父类那里继承来的数据成员以及自己所特有的数据成员。

6. C++支持多态性

C++支持多态性，即以同样的表示方式可以处理多种不同类型的数据。例如，命名相同的函数，其实现的功能却不同，C++会依据所给的函数形式参数来选择合适功能的函数。

C++的多态性还有静态和动态之分。

- 静态多态性：在编译阶段由编译程序根据操作数从多个同名函数中选择相应的调用。
- 动态多态性：在执行阶段才处理目标的实际差别，而调用不同函数对不同类执行不同的处理方式。

C++的上述特点，有待于随着教与学的深入，加深理解。

1.1.2 C++和 C 的关系和区别

C 语言最早是由 UNIX 操作系统发展起来的。

C 语言的特点是：语言简洁、功能强大、运行速度快，因此，成为计算机程序设计的重要语言之一。

1980 年美国贝尔实验室的专家 Bjarne Stroustrup 博士为了解决面向对象的程序工程而将 C 语言发展成 C++语言。

C++包含了 C 语言的特性和功能，并扩展了 C 语言的功能，特别是它具有面向对象的功能。

C 语言是一个低层的函数型语言，它缺乏高层次的数据抽象机制，而 C++则弥补了这一不足。

C++和 C 语言的本质区别就是，C 语言是面向过程的，而 C++是面向对象的。

1.2 C++程序的基本结构和语法规则

一个完整的 C++程序是由一个或多个程序文件构成，而程序文件是由一个或多个函数组成，下面通过一个完整例子详细说明 C++程序的基本结构和语法规则。

1.2.1 一个简单的 C++ 程序

【例 1.1】试编写一个完整的 C++ 程序，实现由键盘输入的三角形 3 边 a、b 和 c，判断 a、b、c 能否构成三角形，若能构成，计算其三角形的面积。

```
//程序文件名为: Triangle_Area.cpp
#include <iostream.h>
#include <math.h>
float AREA(float x, float y, float z)
{
    float s, S;
    s=(x+y+z)/2;
    S=(float)(sqrt(s*(s-x)*(s-y)*(s-z)));
    return S;
}
void main( )
{
    float a, b, c, A;
    cout<<"请输入三角形的三边";
    cin>>a>>b>>c;
    if(a+b>c && b+c>a && a+c>b)
    {
        A=AREA(a,b,c);
        cout<<"三角形的面积是: "<<A<<endl;
    }
}
```

1.2.2 C++ 程序结构及语法规则

下面通过例 1-1 介绍 C++ 程序的基本结构和语法规则。

1. 一个完整的 C++ 程序由一个或多个程序文件构成

下面通过两个示例进行解释说明。

(1) 由一个程序文件构成

```
//程序文件名为 Triangle_Area.cpp
#include <iostream.h>
#include <math.h>
float AREA(float x, float y, float z)
{
    float s,S;
    s=(x+y+z)/2;
    S=float(sqrt(s*(s-x)*(s-y)*(s-z)));
    return S;
}
void main( )
{
    float a,b,c,A;
    cout<<"请输入三角形的三边";
    cin>>a>>b>>c;
    if(a+b>c && b+c>a && a+c>b)
    {
        A=AREA(a,b,c);
```

```

        cout<<"三角形的面积是: "<<A<<endl;
    }
}

```

(2) 由多个程序文件构成

//头件名为: Area.h

```
float AREA(float x, float y, float z);
```

//程序文件名为 Triangle_Area.cpp

```
#include "Area.h"
```

```
#include <math.h>
```

```
float AREA(float x, float y, float z)
```

```
{
    float s, S;
    s=(x+y+z)/2;
    S=float(sqrt(s*(s-x)*(s-y)*(s-z)));
    return S;
}
```

//主程序文件名为 Area.cpp

```
#include "Area.h"
```

```
#include <iostream.h>
```

```
void main( )
```

```
{
    float a,b,c,A;
    cout<<"请输入三角形的三边";
    cin>>a>>b>>c;
    if(a+b>c && b+c>a && a+c>b)
    {
        A=AREA(a,b,c);
        cout<<"三角形的面积是: "<<A<<endl;
    }
}
```

2. 一个程序文件由一个或多个函数组成

下面通过两个示例进行解释说明。

(1) 由一个函数组成

//程序文件名为: Triangle_Area.cpp

```
#include <iostream.h>
```

```
#include <math.h>
```

```
void main( )
```

```
{
    float a,b,c, s, A;
    cout<<"请输入三角形的三边";
    cin>>a>>b>>c;
    if(a+b>c && b+c>a && a+c>b)
    {
        s=(x+y+z)/2;
        A=float(sqrt(s*(s-x)*(s-y)*(s-z)));
        cout<<"三角形的面积是: "<<A<<endl;
    }
}
```

(2) 由多个函数组成

```
//程序文件名为: Triangle_Area.cpp
#include <iostream.h>
#include <math.h>
float AREA(float x,float y,float z)
{
    float s,S;
    s=(x+y+z)/2;
    S=float(sqrt(s*(s-x)*(s-y)*(s-z)));
    return S;
}
void main( )
{
    float a,b,c,A;
    cout<<"请输入三角形的三边";
    cin>>a>>b>>c;
    if(a+b>c && b+c>a && a+c>b)
    {
        A=AREA(a,b,c);
        cout<<"三角形的面积是: "<<A<<endl;
    }
}
```

3. 函数的定义格式

格式:

类型修饰符 函数名(形式参数表)

函数体

例如,上面程序中的 AREA 函数,对照语法格式定义如下:

```
float AREA(float x,float y,float z)
{
    float s,S;
    s=(x+y+z)/2;
    S=float(sqrt(s*(s-x)*(s-y)*(s-z)));
    return S;
}
```

函数定义格式说明:

- 类型修饰符: 是用于说明函数的返回值类型。
- 函数名: 是由用户自己给函数起的名字, 便于其他函数来调用此函数。
- 形式参数表: 用于说明函数所需的接口信息, 即函数的自变量; 形式参数可以是一个或多个, 若有多个, 各参数之间用逗号隔开, 也可以没有, 但两边的括号不得省略(因为它是函数的标识)。
- 函数体: 由一对花括号{ }括起来的一系列语句组成, 用于实现函数所要完成的功能。

4. 有关 C++程序的几点说明

- ① 每一个 C++程序的文件扩展名约定为.cpp。
- ② 每一个 C++程序都必须有 1 个主函数, 其函数名为 main, 它是整个 C++程序的入口点。
- ③ C++程序是区分字母的大小写的, 不得写错。如 main 不能写成 MAIN 或 Main。
- ④ C++语言的语句有两种: 基本语句和复合语句。

- 基本语句：是以分号“;”作为结束标记。
- 复合语句：是用一对花括号{ }括起来的若干条基本语句组成。由此得出，函数中的函数体实际上就是一个复合语句。
- ⑤ C++中包括字符常量和字符串常量。
- 字符常量：是由单引号括起来的单个字符，如'a'、'@'。
- 字符串常量：是由双引号括起来的若干个（可以没有）字符，如"abc"、"a"、""。

注意：

在 C++语言中，只有字符串常量中双引号内的字符以及注释文字可以是汉字，其他字符都必须是西文字符（即半角字符）。

⑥ 程序的书写风格

- 一行中可以有多个语句，例如：

```
int a,b;      a=2;      b=-5;
```

- 一个语句也可以分写在连续的若干行中，例如：

```
cout<<"a+b="
<<a+b<<endl;
```

以上两语句等价于

```
cout<<"a+b="<<a+b<<endl;
```

⑦ 数据的输入和输出

- cin 和键盘输入：cin 是一个连接键盘的输入流对象，与之相联系的输入操作符是>>。

语句格式：

```
cin>>变量[>>变量];
```

- cout 和屏幕输出：cout 是一个连接显示器的输出流对象，与之相联系的输出操作符是<<。

语句格式：

```
cout<<表达式[<<表达式];
```

⑧ 换行操作符 endl。

功能举例：

```
cout<<"a="<<35<<endl<<"b="<<-55<<endl;
```

屏幕显示：

```
a=35
```

```
b=-55
```

区别此语句：

```
cout<<"a="<<35<<"b="<<-55<<endl;
```

屏幕显示：

```
a=35b=-55
```

⑨ 预处理命令和头文件。

- 预处理命令：以#号打头的命令称为预处理命令（注意命令后不得有分号“;”）。例如：

```
#include <iostream.h>
```

```
#define PI 3.14159
```

- 文件包含预处理命令：#include。

功能是把一个文本文件的内容包含（即插入）到该命令处。

- 头文件：一般是由系统提供的文本文件，也可以由用户自己定义的一个文本文件，其扩展名约定为.h。凡是程序中使用键盘输入和屏幕输出，都必须在程序的开始位置包含系统头文件：iostream.h，格式如下：

```
#include <iostream.h>
```

- 宏定义预处理命令：#define。

命令格式：

```
#define <符号常量> 常量
```

试比较下面两个程序：

【例 1.2】从键盘输入一个半径值，计算其圆的周长、面积和球的体积。

方法 1：

```
//文件名: sample_1.cpp
#include <iostream.h>
void main( )
{
    float r,L,S,V;
    cout<<"请输入一个圆的半径: ";
    cin>>r;
    L=2*r*3.14;
    S=r*r*3.14;
    V=4.0/3*3.14*r*r*r;
    cout<<L<<" "<<S<<" "<<V<<endl;
}
```

方法 2：

```
//文件名: sample_2.cpp
#include <iostream.h>
#define PI 3.14
void main( )
{
    float r,L,S,V;
    cout<<"请输入一个圆的半径: ";
    cin>>r;
    L=2*r*PI;
    S=r*r*PI;
    V=4.0/3*PI*r*r*r;
    cout<<L<<" "<<S<<" "<<V<<endl;
}
```

通过比较以上两种方法可知，灵活使用#define 命令，可使程序结构清晰，保证常量在程序中的一致性，便于修改常量。

⑩ 函数调用。调用格式如下：

函数名 (实在参数表)

- 主函数调用：主函数是由操作系统自动调用执行的，不得由其他函数来调用。
- 其他函数调用：其他函数是通过 C++程序中的函数调用执行的。在调用时，实在参数表中的参数必须在数量和类型上与相应函数定义中的形式参数表相吻合。

⑪ C++程序的注释有两种：单行注释和多行注释。

- 单行注释：以//打头，其后是注释文字，可一直延续到该行行尾。

- 多行注释：以/*打头，以*/结尾，其间是注释文字，可以跨越多行。

试比较以下语句中两种注释的用法：

```
L=2*r*3.14;           //用来计算圆的周长
S=r*r*3.14;           //用来计算圆的面积
L=2*r*3.14;           /*用来计算圆的周长，其中
S=r*r*3.14;           r是由键盘输入的圆的半径*/
```

以上斜体部分为注释文字。

⑫ 标识符是由程序设计者为函数、变量等命名时所用的符号，其命名规则如下：

- 第一个字符必须是字母或下划线，即可取 A~Z、a~z 或 _ 中的字符一。
- 第二个字符以后可以是 0~9、A~Z、a~z 或 _ 中的字符。
- 标识符的长度不超过 32 个字符。
- 标识符不得取汉字。

⑬ 常量、变量和变量的数据类型。

- 常量：是其值在程序运行过程中不变化的量，如：23, 3.14, 1.23e+5, 'A', "中国"
- 变量：是其值在程序运行过程中变化的量，如：i1=45; i1=-55; 语句中的 i1 和 i2 即为变量。

变量的定义格式如下：

类型修饰符 变量名表；

例如：

```
int a1, a2, a3;
```

变量的数据类型可以是 char、int、long、float、double 等。

⑭ 赋值表达式和赋值语句

赋值表达式的格式：

<变量名>=表达式

赋值语句的格式：

<变量名>=表达式；

例如：

```
int a1, a2, a3;
```

```
a1=2;           //常量 2 也是表达式的一种
```

```
a2=a1=2;        //a1=2 是赋值表达式
```

```
a3=a2=a1=2;     //a2=a1=2 是赋值表达式
```

在赋值表达后添加一个分号即成为赋值语句。

1.3 程序的编辑、编译、连接和运行

开发一个应用系统需要经过编辑、编译、连接和测试运行程序一系列过程，初学者使用集成化编译系统最为适宜，最常见的 C++ 集成化编译系统有 Microsoft 公司的 Visual C++ 6.0（简称 VC++ 6.0）和 Inprise 公司的 C++ Builder 6.0，下面只对 VC++ 6.0 开发环境进行简单介绍，目的是让读者掌握编辑、编译、连接和运行一个 C++ 控制台应用程序（console application program）的简要过程。