

# C++语言

## 程 序 设 计

主编 © 赵 宇 蒋郑红 王 崇

天津出版传媒集团

 天津科学技术出版社

# C++ 语言程序设计

主 编 赵 宇 蒋 郑 红 王 崇

天津出版传媒集团

 天津科学技术出版社

编 著 人 编 出  
电 话 022-2333330  
022-2333330 (230) 233330  
网 址: www.tjpub.com.cn  
邮 政 代 理 处

开 本: 787×1095 1/16 印 张: 18  
2018年8月第1版第1次印刷  
定 价: 35.00元

图书在版编目 (CIP) 数据

C++ 语言程序设计 / 赵宇, 蒋郑红, 王崇主编. —

天津: 天津科学技术出版社, 2018. 6

ISBN 978-7-5576-3123-9

I. ①C… II. ①赵… ②蒋… ③王… III. ①C 语言—  
程序设计 IV. ①TP312.8

中国版本图书馆 CIP 数据核字 (2017) 第 146274 号

责任编辑: 石 崑

责任印制: 兰 毅

天津出版传媒集团



天津科学技术出版社出版

出 版 人: 蔡 颢

天津市西康路35号 邮编 300051

电话(022)23332369 (编辑室)

网址: [www.tjkjcbbs.com.cn](http://www.tjkjcbbs.com.cn)

新华书店经销

天津印艺通制版印刷有限责任公司印刷

开本: 787×1092 1/16 印张 15 字数 370000

2018年6月第1版第1次印刷

定价: 32.00元

# 前 言

目前世界上90%以上的程序员用C++或Java的编译器作为工具，这足以说明C++在当今计算机科学研究和软件开发中的地位。C++语言既是目前应用较广的一种优秀的高级程序设计语言，也是一个可编写高质量的用户自定义类型库的工具。C++保留了对传统的结构化程序设计方法的支持，同时又增加了对面向对象程序设计方法的完全支持，它具有丰富的数据类型和各种运算功能，还有庞大的函数库和类库，既支持面向过程的设计，又支持面向对象的程序设计。

本书对C++语言的基本概念、原理和方法的叙述由浅入深，条理分明、循序渐进，帮助读者快速理解与程序设计相关的基本概念，掌握程序设计语言的基本知识，建立程序设计的基本思想，并通过大量的示例引导读者逐步熟悉程序设计。

本书由多位从事高职院校C++语言课程教学的一线老师，根据多年的教学和研究经验总结编写而成。本书的主要特点可归纳如下：

- 教材内容循序渐进、由浅入深，达到逐步提高的目的。
- 每章首均以精炼的文字描述本章的学习目标以及重难点。
- 每一章均编入了实例，使读者将本章的概念、实例通过应用举例的方式加深理解，达到理论与实践紧密结合的目的。
- 精选实用的例题讲解程序设计及调试方法，所有程序代码都在 Visual C++6.0 的环境下调试完成。
- 课后精选大量的“思考训练”及答案，达到读者自我提高的目的。

本书内容先进，体系合理，概念清晰，讲解详尽，降低台阶，分散难点，例题丰富，深入浅出，文字流畅，通俗易懂，是初学者学习的理想读物，可作为高等学校各专业的正式教材，也是一本自学的好教材。

由于时间仓促，书内可能会有疏漏之处，敬请各位读者提出宝贵意见，以便我们及时修改。

编者  
2018年4月

# 目 录

|                        |    |
|------------------------|----|
| 第一章 C++概述 .....        | 1  |
| 第一节 C++语言的产生与发展 .....  | 1  |
| 第二节 C++程序的组成 .....     | 1  |
| 第三节 C++语言的特点 .....     | 2  |
| 第四节 C++程序开发过程 .....    | 3  |
| 第五节 第一个C++程序 .....     | 4  |
| 第二章 基本数据类型 .....       | 9  |
| 第一节 C++语言的基本数据类型 ..... | 9  |
| 第二节 常量 .....           | 10 |
| 第三节 变量 .....           | 15 |
| 第四节 void 类型 .....      | 17 |
| 第五节 变量初始化 .....        | 17 |
| 第三章 控制结构 .....         | 21 |
| 第一节 顺序结构 .....         | 21 |
| 第二节 选择结构 .....         | 22 |
| 第三节 循环结构 .....         | 27 |
| 第四节 跳转语句 .....         | 29 |
| 第四章 运算符和表达式 .....      | 39 |
| 第一节 算术运算符及其表达式 .....   | 39 |
| 第二节 关系运算符及其表达式 .....   | 40 |
| 第三节 逻辑运算符及其表达式 .....   | 42 |
| 第四节 赋值运算符及赋值表达式 .....  | 43 |
| 第五节 条件运算符及表达式 .....    | 44 |
| 第六节 逗号运算符及其表达式 .....   | 45 |
| 第七节 自增、自减运算符 .....     | 45 |
| 第八节 其他运算符 .....        | 46 |



|                          |     |
|--------------------------|-----|
| 第五章 结构体、共用体与枚举类型 .....   | 54  |
| 第一节 结构体 .....            | 54  |
| 第二节 共用体 .....            | 66  |
| 第三节 枚举类型 .....           | 69  |
| 第四节 用 typedef 声明类型 ..... | 71  |
| 第六章 类和对象 .....           | 77  |
| 第一节 面向对象概述 .....         | 77  |
| 第二节 类的定义 .....           | 78  |
| 第三节 构造函数 .....           | 79  |
| 第四节 析构函数 .....           | 80  |
| 第五节 类的成员函数 .....         | 82  |
| 第六节 对象成员的引用 .....        | 86  |
| 第七节 类的封装性 .....          | 88  |
| 第七章 继承与派生 .....          | 94  |
| 第一节 继承与派生的概念 .....       | 94  |
| 第二节 派生类的继承方式 .....       | 96  |
| 第三节 派生类的构造函数与析构函数 .....  | 104 |
| 第四节 多重继承的相关问题 .....      | 109 |
| 第五节 虚基类 .....            | 111 |
| 第八章 多态性 .....            | 118 |
| 第一节 多态性和编联 .....         | 118 |
| 第二节 虚函数 .....            | 119 |
| 第三节 抽象类 .....            | 121 |
| 第四节 模板 .....             | 123 |
| 第五节 运算符重载 .....          | 128 |
| 第九章 数 组 .....            | 134 |
| 第一节 数组 .....             | 134 |
| 第二节 一维数组的初始化 .....       | 136 |
| 第三节 字符数组 .....           | 137 |
| 第四节 多维数组 .....           | 139 |
| 第五节 数组参数 .....           | 140 |
| 第十章 函 数 .....            | 151 |
| 第一节 函数定义 .....           | 151 |
| 第二节 函数的调用与声明 .....       | 153 |

|      |                             |     |
|------|-----------------------------|-----|
| 第三节  | 函数参数的传递方式 .....             | 156 |
| 第四节  | 变量作用域 .....                 | 160 |
| 第五节  | 递归函数 .....                  | 167 |
| 第六节  | 内联函数 .....                  | 168 |
| 第七节  | 函数重载 .....                  | 169 |
| 第八节  | 预处理指令 .....                 | 169 |
| <br> |                             |     |
| 第十一章 | 指 针 .....                   | 180 |
| 第一节  | 指针的定义及引用 .....              | 180 |
| 第二节  | 指针与函数 .....                 | 183 |
| 第三节  | 指针与数组 .....                 | 186 |
| 第四节  | 指针与字符串 .....                | 193 |
| 第五节  | 指针数组和指向指针的指针 .....          | 195 |
| 第六节  | 变量的引用 .....                 | 197 |
| <br> |                             |     |
| 第十二章 | Visual C++6.0 集成环境 .....    | 207 |
| 第一节  | Visual C++6.0 概述 .....      | 207 |
| 第二节  | Visual C++ 6.0 的安装与启动 ..... | 207 |
| 第三节  | 菜单栏 .....                   | 208 |
| 第四节  | 工具栏 .....                   | 213 |
| 第五节  | 项目和项目工作区 .....              | 214 |
| 第六节  | Visual C++6.0 的使用 .....     | 216 |
| <br> |                             |     |
| 附 录  | .....                       | 226 |
| 附录 1 | 常用字符与 ASCII 码对照表 .....      | 226 |
| 附录 2 | 运算符和结合性 .....               | 228 |
| 附录 3 | C 语言中的关键字 .....             | 230 |
| <br> |                             |     |
| 参考文献 | .....                       | 232 |

# 第一章 C++概述

## 本章要点

- 了解 C++ 语言的发展和特点。
- 了解 C++ 程序的结构组成。
- 了解 C++ 语言程序的编写和运行过程。
- 了解 C++ 程序的构成和书写规则。
- 了解 C++ 程序的编写和实现。

## 第一节 C++ 语言的产生与发展

美国 AT&T 贝尔实验室的本贾尼·斯特劳斯特卢普 (Bjarne Stroustrup) 博士在 20 世纪 80 年代初期发明并实现了 C++ (最初这种语言被称作 “C with Classes”)。C++ 语言发展大概可以分为三个阶段:

第一阶段从 80 年代到 1995 年。这一阶段 C++ 语言基本上传统类型上的面向对象语言, 并且凭借着接近 C 语言的效率, 在工业界使用的开发语言中占据了相当大份额。

第二阶段从 1995 年到 2000 年, 这一阶段由于标准模板库 (STL) 和后来的 Boost 等程序库的出现, 泛型程序设计在 C++ 中占据了越来越多的比重性。当然, 同时由于 Java、C# 等语言的出现和硬件价格的大规模下降, C++ 受到了一定的冲击。

第三阶段从 2000 年至今, 由于以 Loki、MPL 等程序库为代表的产生式编程和模板元编程的出现, C++ 出现了发展历史上又一个新的高峰, 这些新技术的出现以及和原有技术的融合, 使 C++ 已经成为当今主流程序设计语言中最复杂的一员。

## 第二节 C++ 程序的组成

### 一、C++ 程序的基本结构

一般一个 C++ 程序有如下结构特征:

- 1、一个 C++ 程序由若干个函数或若干个类组成。
- 2、每一个 C++ 程序都有且仅有一个 main 函数。程序总是从 main () 函数开始执行，执行完 main () 函数中的所有语句后，程序就结束了。
- 3、每个语句后面都必须要有个分号，它是语句之间的分隔符。
- 4、C++ 程序通过流 cin 和 cout 来实现标准输入和输出。
- 5、C++ 程序严格区分子母的大小写。
- 6、C++ 程序的编译预处理命令以 “#” 开头，编译预处理命令不需要以分号结束。

## 二、C++ 程序的书写规则

为了便于程序的阅读、理解和维护，在书写程序时，应遵循以下规则：

- 1、C++ 程序的标识符习惯上使用小写英文字母，一般只有符号常量或有其他特殊用途的符号才大写。
- 2、一条语句占一行或多行，避免一行写多条语句，除非这些语句非常相关。
- 3、C++ 程序注释包含段注释和行注释两种，段注释单独一行或几行，主要说明下面一段程序实现的功能；行注释用于解释某条语句的功能。在程序编写中建议多使用注释信息，以增加程序的可读性。例如：
  - (1) /\* 注释内容 \*/：是多行注释；
  - (2) // 注释内容：是单行注释。
- 4、用 {} 括起来的部分，通常表示了程序的某一层结构。{} 一般与该结构语句的第一个字母对齐，并单独占一行。例如：

```
{
    cout<<"Hello C++ program.";
}
```

- 5、低一层次的语句或说明可比高一层次的语句或说明缩进若干格后书写，以便看起来更加清晰，增加程序的可读性。

## 第三节 C++ 语言的特点

C++ 是一种使用非常广泛的计算机编程语言，是一种静态数据类型检查的、支持多重编程范式的通用程序设计语言；

C++ 语言具有以下特点：

- 1、C++ 设计成静态类型、和 C 同样高效且可移植的多用途程序设计语言；
- 2、C++ 设计成直接的和广泛的支持多种程序设计风格（程序化程序设计、资料抽象化、面向对象程序设计、泛型程序设计）；
- 3、C++ 设计成给程序设计者更多的选择；
- 4、C++ 设计成尽可能与 C 兼容，以提供一个从 C 到 C++ 的平滑过渡；
- 5、C++ 避免平台限定或没有普遍用途的特性；

6、C++不会带来额外开销的特性。

出于对保证语言的简洁和运行高效等方面的考虑，C++的很多特性都是以库（如STL）或其他的形式提供的，没有直接添加到语言本身里。

## 第四节 C++程序开发过程

C++程序从编写到最后得到运行结果的过程称为程序开发过程，C++程序的开发过程如图 1-1 所示：

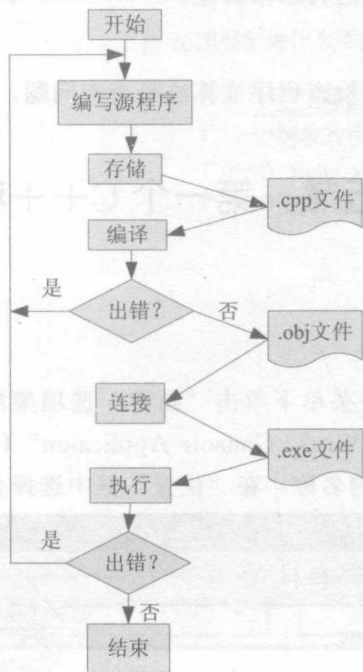


图 1-1 C++程序的开发过程

### 1. 编写源程序

用高级语言编写的程序称为“源程序 (source program)”，将源程序编写后，以文本形式保存到磁盘中。在 Visual C++ 6.0 系统下，经过编辑后得到的源程序文件以“.cpp”做为源程序文件扩展名。

### 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，必须先调用一种被称为“编译器 (compiler)”的软件（也称编译程序或编译系统），将源程序编译成二进制形式的“目标程序 (object program)”。系统对源程序进行编译的同时，还对源程序的语法进行检查。当发现错误时，会在屏幕上列出错误的位置和种类。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)。此时要再次使用编辑工具对源程序进行排错修正。如果

源程序没有语法错误，编译后将产生一个与源程序同名，以“.obj”为扩展名的目标程序。

### 3. 连接目标文件

编译后产生的目标程序还不能直接运行，因为每一个模块往往都是单独编译的，需要把各模块编译后得到的目标程序与系统提供的标准库函数等连接后才能运行。连接过程是使用系统提供的“连接程序”进行的，连接后产生以“.exe”为扩展名的可执行程序。

### 4. 运行程序

可执行目标程序生成后，就可以在操作系统的支持下运行。若执行结果达到预期的目的，则开发工作到此完成。否则，要进一步检查修改源程序，再经过“编辑→编译→连接”的过程，直到取得正确的运行结果为止。

### 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。

## 第五节 第一个 C++ 程序

### 一、建立一个工程

打开 VC++6.0，在文件菜单下单击“新建”选项菜单，出现如图 1-2 所示对话框，一般选择“工程”菜单下的“Win32 Console Application”（win32 控制台应用程序），并在“工程名称”框中输入此工程的名称，在“位置”框中选择合适的存储位置。

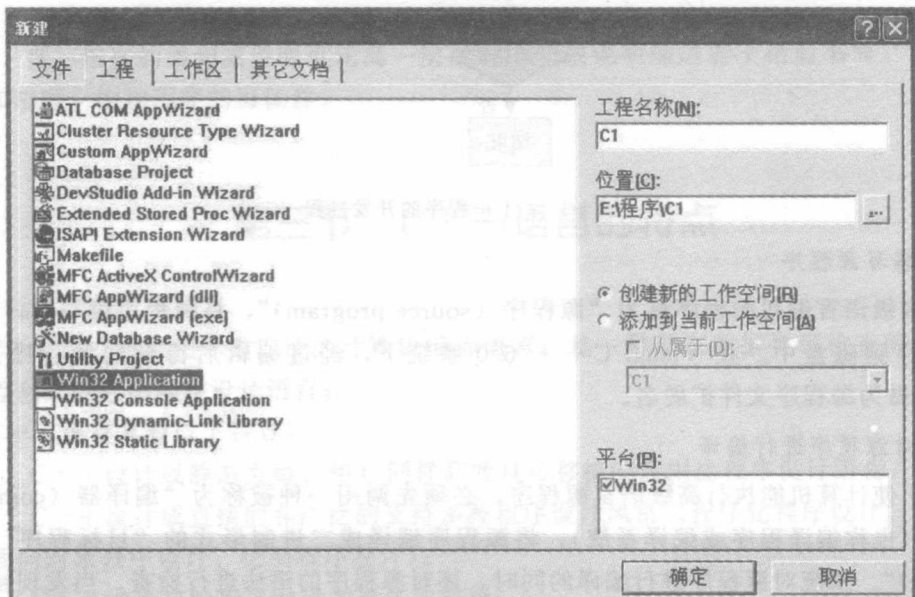


图 1-2 新建空白工程

点击“确定”按钮后，出现如图 1-3 所示对话框，提示选择控制台程序的类型，包括四种类型：

- (1) 空程序。选择此类型，系统不生成任何程序，需要自己添加。
- (2) 简单的程序。系统会自带一个头文件和一个 .cpp 文件。
- (3) “Hello World”程序。VC++6.0 自带的测试程序，运行之后就会在屏幕上显示“Hello World”。
- (4) 支持 MFC 的程序。只能使用 MFC 中与窗口无关的类，同时程序中会含有 MFC 的东西。

选择合适的类型的控制台程序后，点击“完成”按钮，则建立了一个工程。

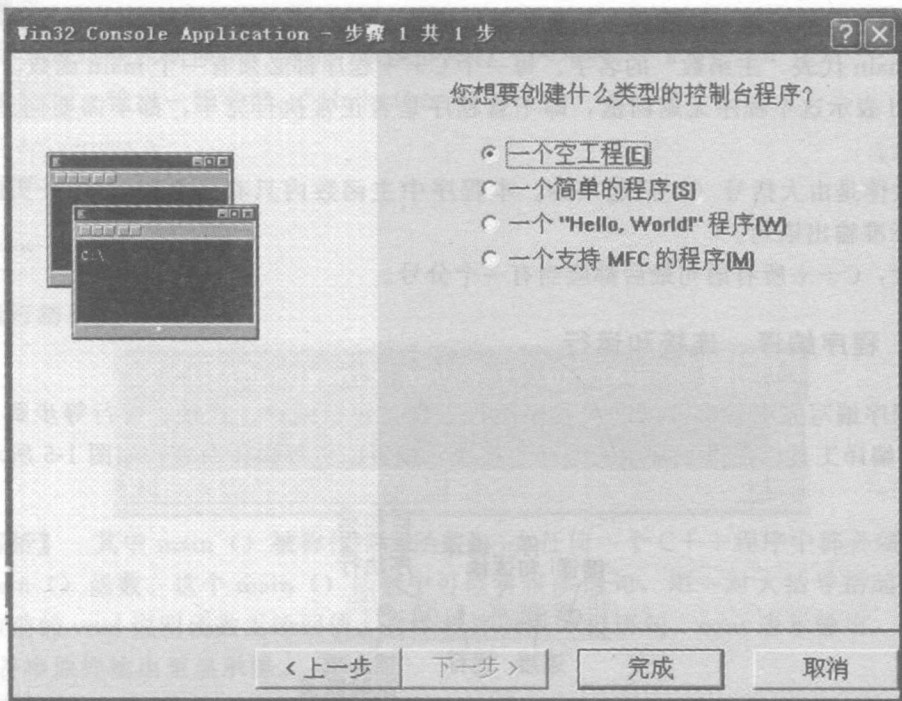


图 1-3 控制台程序的不同类型

## 二、编写源程序

**例 1.1** 在屏幕上输出一行字符：“同学们，大家好！”。

程序如下：

```
#include<iostream.h> //包含头文件 iostream
void main () //主函数
{
    cout<< " 同学们，大家好!";
}
```

在运行时会在屏幕上输出如图 1-4 所示的一行信息：

程序的第 1 行“#include <iostream>”，这是 C++ 的一个预处理命令，不是 C++ 的语句，它以“#”开头与 C++ 语句相区别，行的末尾没有分号。#include <iostream

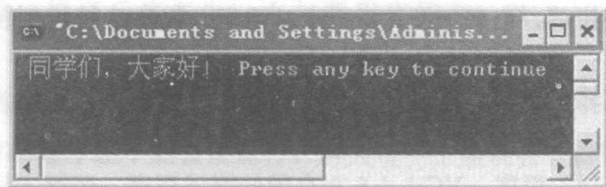


图 1-4 控制台程序的不同类型

>是一个“包含命令”，它的作用是将文件 `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。在程序进行编译时，先对所有的预处理命令进行处理，将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。

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。`main` 前面的 `void` 表示这个程序无返回值，即不管程序是否正常执行完毕，都不需要向系统返回任何信息。

函数体是由大括号 `{}` 括起来的。本程序中主函数内只有一个以 `cout` 开头的语句，`cout` 为标准输出语句。

注意：C++ 所有语句最后都应当有一个分号。

### 三、程序编译、连接和运行

源程序编写完毕后就可以进入程序运行过程，包括编译、连接、运行等步骤。VC++ 6.0 的编译工具栏提供了常用的编译、连接、设置断点等操作命令，如图 1-5 所示。

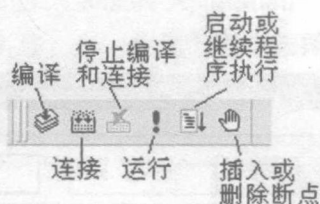


图 1-5 编译工具栏

#### 1. 编译

选择主菜单“组建 (Build)”中的“编译 (Compile)”命令，或单击工具条上的图标 ，系统只编译当前文件而不调用链接器或其他工具。输出 (Output) 窗口将显示编译过程中检查出的错误或警告信息，在错误信息处单击鼠标右键或双击鼠标左键，可以使输入焦点跳转到引起错误的源代码处大致位置以进行修改。

#### 2. 连接

选择主菜单“组建 (Build)”中的“组建 (Build)”命令，或单击工具条上的图标 ，对最后修改过的源文件进行编译和连接。

选择主菜单“组建 (Build)”中的“全部重建 (Rebuild All)”命令，允许用户编译所

有的源文件，而不管它们何时曾经被修改过。

选择主菜单“组建 (Build)”中的“批组建 (Batch Build)”命令，能单步重新建立多个工程文件，并允许用户指定要建立的项目类型。

程序组建完成后生成的目标文件 (.obj)、可执行文件 (.exe) 存放在当前工程项目所在文件夹的“Debug”子文件夹中。

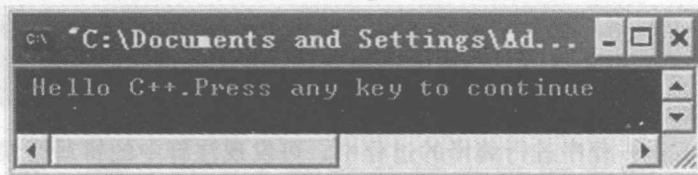
### 3. 运行

选择主菜单“组建 (Build)”中的“执行 (Build Execute)”命令，或单击工具条上的图标，执行程序，将会出现一个新的用户窗口显示运行的结果。对于比较简单的程序，可以直接选择该项命令，编译、连接和运行一次完成。

#### 例 1.2 输出一行字符：“Hello C++。”

```
#include<iostream.h>
void main ()
{
    cout<<" Hello C++.";
}
```

【运行结果】:

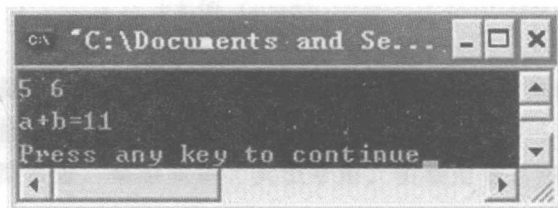


【解析】 其中 main () 被称作“主函数”，在任何一个 C++ 程序中都必须有且只有一个 main () 函数。这个 main () 函数中可以有很多语句，用一对大括号括起来。main () 函数前的 void 说明函数无返回值。这个程序只有一行语句，cout 表示输出，将双引号中的字符串原样输出至显示屏。

#### 例 1.3 输出下列程序的运行结果。

```
#include <iostream.h>
void main ()
{
    int a, b, sum;
    cin>>a>>b;
    sum=a+b;
    cout<<" a+b=" <<sum<<endl;
}
```

【运行结果】:



**【解析】** 这个程序的运行结果是求出从键盘输入的任意两个整数的和。第四行是变量定义，说明了三个整型变量 a、b 和 sum。“cin>>a>>b;”语句的作用是从键盘输入两个数；“sum=a+b;”语句的作用是求出 a+b 并将 a+b 的值赋给 sum；cout 表示输出，作用是向屏幕上输出信息，它将双引号之间的内容完整的输出到屏幕。“\n”表示换行符，它的作用是将光标移到下一行的开始。语句后面有一个分号，表示该语句结束，这个分号是必不可少的。

## 思考训练

1. 一个 C++ 程序的执行是从( )。
  - A. 本程序的 main 函数开始，到 main 函数结束
  - B. 本程序文件的第一个函数开始，到本程序文件的最后一个函数结束
  - C. 本程序文件的第一个函数开始，到本程序 main 函数结束
  - D. 本程序的 main 函数开始，到本程序文件的最后一个函数结束
2. 以下叙述正确的是( )。
  - A. C++ 程序书写格式严格，每行只能写一个语句
  - B. C++ 程序书写格式严格，每行必须有行号
  - C. C++ 程序书写格式自由，每行可以写多条语句，但之间必须用逗号隔开
  - D. C++ 程序书写格式自由，一个语句可以分写在多行上
3. 以下叙述正确的是( )。
  - A. 在对一个 C++ 程序进行编译的过程中，可发现注释中的拼写错误
  - B. 在 C++ 程序中，main 函数必须位于程序的最前面
  - C. C++ 语言本身没有输入输出语句
  - D. C++ 程序的每行中只能写一条语句
4. C++ 与 C 的区别有哪些？
5. C++ 程序开发的一般步骤是什么？

## 第二章 基本数据类型

### ① 本章要点

- 掌握 C++ 语言的基本数据类型。
- 掌握常量与变量、整型数据与实型数据的概念。
- 掌握各种类型的变量定义及其初始化。
- 掌握整型数据的定义方法，根据使用整数的范围选用定义整型变量的类型。
- 掌握实型数据的定义方法，单精度数和双精度数的表示范围。
- 掌握字符型数据与字符串型数据的概念以及二者的区别。
- 掌握各种类型的变量定义及其初始化。

### 第一节 C++ 语言的基本数据类型

C++ 语言的数据类型包括基本数据和构造数据类型两类。构造数据类型又称复合数据类型，它是一种更高级的抽象。当变量被定义为某种类型时会受到系统对该类型的特别保护，确保其值不受非法操作。本章主要介绍基本数据类型，C++ 语言提供的基本数据类型如图 2-1 所示。

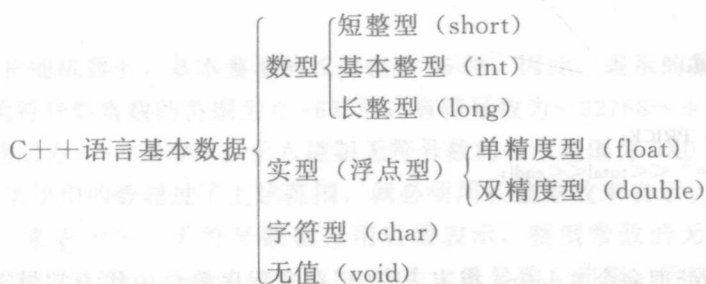


图 2-1 C++ 语言的基本数据类型

对于基本数据类型量，按其取值是否可改变又分为常量和变量两种。在程序执行过程中，其值不发生改变的量称为常量，取值可变的量称为变量。它们可与数据类型结合起来分类。例如，可分为整型常量、整型变量、实型常量、实型变量、字符常量、字符变量、

枚举常量、枚举变量。在程序中，常量是可以不经说明而直接引用的，而变量则必须先说明后使用。

## 第二节 常量

常量是指在程序运行的过程中，其值不能被改变的量。常量在程序中不需要进行任何说明就可以直接使用，因为常量本身就隐含了它的类型。常量可以是不同类型的数据，如：123、-6、9 为整型常量；5.14、0.635 为实型常量；'a'、'B' 为字符常量。常量可以分为普通常量和符号常量。

### 一、符号常量

符号常量是指用一个标识符代表一个常量，这样的标识符称为符号常量。符号常量在使用之前必须先定义，其一般形式为：

```
#define 标识符 常量表达式<或字符串>
```

其中 #define 是一条预处理命令，也称为宏定义命令，其功能是把该标识符定义为其后的常量值。一经定义，以后在程序中所有出现该标识符的地方均代之以该常量值。习惯上符号常量的标识符用大写字母，变量标识符用小写字母以示区别。

关于符号常量有几点说明：

- (1) 一个 #define 语句只能定义一个符号常量，并且只占据一个书写行；
- (2) 由于符号常量定义式，不是 C++ 语句，所以行尾没有分号；
- (3) 符号常量不是变量，它所代表的值在整个作用域内不能再改变。也就是说，在程序中，不能再赋值语句对它重新赋值。

#### 例 2.1 符号常量的使用。

```
#include<iostream.h>
#define PRICE 30 //注意这不是语句，末尾不要加分号
int main ()
{
    int num, total;
    num=10;
    total=num * PRICE;
    cout<<" total=" <<total<<endl;
    return 0;
}
```

程序中用预处理命令 #define 指定 PRICE 在本程序单位中代表常量 30，此后凡在本程序单位中出现的 PRICE 都代表 30，可以和常量一样进行运算，程序运行结果为

```
total=300
```

请注意符号常量虽然有名字，但它不是变量。它的值在其作用域（在本例中为主函数）内是不能改变的，也不能被赋值，如用赋值语句“PRICE=40;”给 PRICE 赋值是错