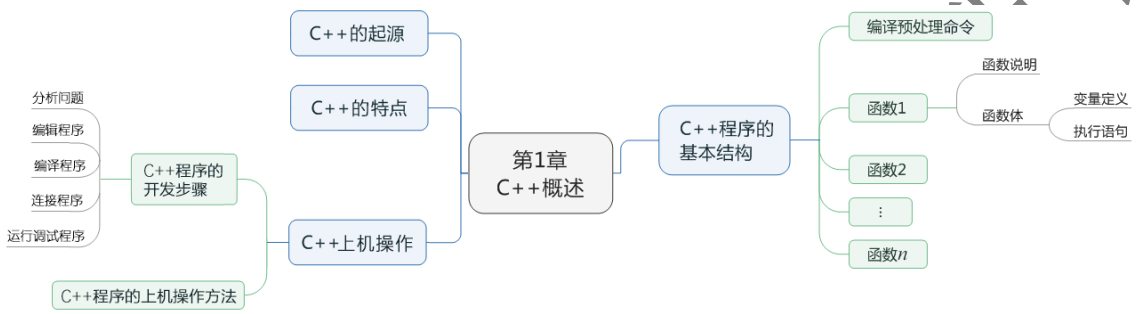
第1章

C++概述



本章知识点导图

通过本章的学习，应了解 C++的起源和特点；掌握 C++程序的基本结构，会编写极简单的 C++程序；了解 C++程序的开发步骤，熟悉 Visual C++集成环境，初步掌握 C++上机操作的方法。

1.1 C++的起源

C++是在 C 语言的基础上逐步发展和完善起来的，因此在介绍 C++之前，先回顾一下 C 语言的发展。

1967 年，Martin Richards 为编写操作系统软件和编译程序开发了 BCPL（Basic Combined Programming Language）；1970 年，Ken Thompson 在继承 BCPL 许多优点的基础上开发了实用的 B 语言；1972 年，贝尔实验室的 Dennis Ritchie 在 B 语言的基础上，进行了进一步的充实和完善，开发出了 C 语言。当时，设计 C 语言是为了编写 UNIX 操作系统，以后，C 语言经过多次改进，逐渐开始流行。目前常用的 C 语言版本基本上都是以 ANSI C 为基础的。

C 语言具有许多优点，如语言简洁灵活、运算符和数据结构丰富、具有结构化控制语句、程序执行效率高，同时具有高级语言和汇编语言的优点等。与其他高级语言相比，C 语言具有可以直接访问物理地址的优点；与汇编语言相比，C 语言具有良好的可读性和可移植性。因此，C 语言得到了极广泛的应用。

随着 C 语言的推广，C 语言存在的一些缺陷或不足也开始暴露。例如，C 语言对数据类型检查的机制比较弱，缺少支持代码重用的结构；随着软件工程规模的扩大，难以适应特大型程序的开发。同时，由于 C 语言是一种面向过程的编程语言，不能满足运用面向对象的方法开发软件的需要。C++

便是在 C 语言的基础上，为克服 C 语言本身存在的缺点，支持面向对象的程序设计而研制的一种通用的程序设计语言，是在 1980 年由贝尔实验室的 Bjarne Stroustrup 创建的。

研制 C++的一个重要目标是使 C++成为一种更好的 C 语言，因此 C++解决了 C 语言存在的问题；另一个重要目标就是面向对象的程序设计，因此在 C++中引入了类的机制。最初的 C++被称为“带类的 C”，1983 年被正式命名为 C++（C Plus Plus）。之后经过不断地完善，形成了目前的 C++。

当前运用较为广泛的 C++有 Microsoft 公司的 Visual C++（简称 VC++）和 Borland 公司的 Borland C++（简称 BC++）。本书以 Microsoft Visual C++ 6.0 集成环境为背景来介绍 C++语言。

1.2 C++的特点

C++的主要特点表现在两方面：一是全面兼容 C 语言，二是支持面向对象的程序设计方法。

（1）C++是一种更好的 C 语言，它保持了 C 语言的优点，对大多数 C 程序代码稍作修改或不修改就可在 C++的集成环境下调试和运行。这对于继承和开发当前已在广泛使用的软件是非常重要的，可以节省大量的人力和物力。

（2）C++是一种面向对象的程序设计语言。它使得程序中各个模块的独立性更强，程序的可读性和可移植性更好，程序代码的结构更合理，程序的扩充性更强。这对于设计、编制和调试一些大型软件尤为重要。

（3）C++集成环境不仅支持 C++程序的编译和调试，还支持 C 程序的编译和调试。通常，C++集成环境约定：当源程序文件的扩展名为.c 时，为 C 程序；当源程序文件的扩展名为.cpp 时，为 C++程序。本书所有例题程序文件的扩展名均为.cpp。

（4）C++的语句非常简练，对语法限制比较宽松，因此 C++语法非常灵活。C++的优点是给用户编程带来了书写上的方便；缺点是由于编译时对语法限制比较宽松，许多逻辑上的错误不容易被发现，所以给用户编程增加了难度。

1.3 C++程序的基本结构

为了说明 C++程序的基本结构，先举三个例题，然后通过这三个例题引出 C++程序的基本结构。

【例 1.1】 文本的原样输出。文件名为 example1_1.cpp。

```
//文本原样输出程序
#include <iostream>
using namespace std;
int main( )
{   cout << "Welcome to C++!\n";
    return 0;
}
```

该程序经编译和连接后，当运行可执行程序时，在显示器上显示：

```
Welcome to C++!
```

在该程序中，main()表示主函数，每个 C++程序必须有且只有一个主函数，C++程序是从主函数开始执行的。main()函数之前的 int 表示 main()函数的类型为整型，即函数返回值的类型为整型，main()函数中的括号内为空表示 main()函数没有形式参数。花括号内的部分是函数体，函数体由语句组成，

每条语句都以分号结束。cout 是 C++程序中的一个输出流，与符号“<<”结合使用可以输出常量、变量的值，以及原样输出双引号中的字符串。“\n”是换行符，即在输出上述信息后换行。return 语句向操作系统返回一个 0 值。如果程序不能正常执行，则自动向操作系统返回一个非零值，一般为-1。

程序中的#include 是 C++编译预处理中的文件包含命令，iostream 是头文件，为了使用输出流 cout 和输入流 cin，程序开头必须用#include 命令将文件 iostream 中的内容包含到本文件中。程序中“using namespace std;”语句的意思是使用命名空间 std。C++标准库中的类和函数，包括输入/输出类，都是在命名空间 std 中进行说明的，因此程序中如果需要用 C++标准库，则需要用“using namespace std;”语句进行说明，表示要用命名空间中的内容。本书中的程序在开头几乎都包含这两行。

程序中以“//”开头的是注释，注释是对程序的说明，是用来提高程序的可读性的。注释对程序的编译和运行不起作用，可以被放在程序的任何位置。

本书是依据 C++标准介绍的，C++标准在一些方面有规定。例如，要求主函数为 int 类型，如果程序正常执行，则返回 0；系统头文件不带后缀.h；当使用系统标准库时，必须使用命名空间 std，这些在本书中都有所体现。

【例 1.2】 求两个整数的和。

```
/*求两个整数的和程序*/
#include <iostream>
using namespace std;
int main( )
{    int a,b,sum;                          //说明变量 a、b、sum 为整型数
     cout<<"Input a,b:";                   //显示提示信息
     cin>>a>>b;                            //通过键盘输入变量 a、b 的值
     sum=a+b;                              //求和
     cout<<"Sum="<<sum<<endl;              //输出结果
     return 0;                             //返回 0 值
}
```

该程序经编译和连接后，当运行可执行程序时，在显示器上显示：

```
Input a,b:3   5
Sum=8
```

该程序中的语句，int a,b,sum; 用来说明变量 a、b、sum 为 int（整型）变量；sum=a+b;是赋值语句，表示将 a 和 b 的值相加，并将结果送给变量 sum。“/*”和“*/”之间的内容也是注释，“endl”是换行符。

【例 1.3】 输入两个整数 a 和 b，用自定义函数 add()求两数的和。

```
#include <iostream>
using namespace std;
int add(int x, int y)
{    int z;
     z=x+y;
     return z;
}
int main( )
{    int a, b, sum;
     cout<< "Input a,b:";
     cin>>a>>b;
     sum=add(a, b);
     cout<<"Sum="<<sum<<endl;
     return 0;
```

```
}
```

该程序经编译和连接后，当运行可执行程序时，在显示器上显示：

```
Input a,b:3  5
Sum=8
```

该程序由两个函数组成：主函数 main()和被调用函数 add()。被调用函数 add()的作用是求 x 和 y 的和，并赋给 z，最后通过 return z;语句返回给主函数 main()。主函数 main()用来输入两个变量 a 和 b 的值，调用 add()函数将变量 a、b 的值传送给形参 x、y，再求两数之和，并将结果返回给 sum 进行输出。

通过例 1.3 可以归纳出 C++程序的基本结构如下。

（1）C++程序由包括主函数 main()在内的一个或多个函数组成，函数是构成 C++程序的基本单位。其中名为 main()的函数称为主函数，它可以被放在程序的任何位置。但是，无论主函数 main()放在程序的什么位置，一个 C++程序总是从主函数开始执行的，并由主函数来调用其他函数。因此，任何一个可运行的 C++程序必须有且只有一个主函数。被调用的其他函数可以是系统提供的库函数，也可以是用户自定义的函数。例如，例 1.3 中的 C++程序就是由主函数 main()和用户自定义函数 add()组成的。

（2）C++函数由函数说明与函数体两部分组成。

① 函数说明。函数说明由函数类型、函数名、函数参数（形式参数）及其类型组成。函数类型为函数返回值的类型。例如：

```
int add(int x,int y)
```

表示自定义了一个名为 add 的函数，函数的类型为 int（整型），该函数有两个形式参数 x、y，类型均为 int（整型）。

主函数 main()是一个特殊的函数，可看作是由操作系统调用的一个函数，其返回值是 int 类型，如果程序正常执行，则返回 0。函数参数可以没有，但函数名后面的括号不能省略。

② 函数体。函数说明后用花括号括起来的部分称为函数体。例如：

```
{    int z;                          //变量定义
      z=x+y;                         //执行语句
      return z;
}
```

如果一个函数内有多对花括号，则最外层的一对花括号为函数体的范围。通常函数体由变量定义和执行语句两部分组成。例如，int z;为变量定义，z=x+y;和 return z;为函数执行语句。在某些情况下可以没有变量定义，甚至可以既无变量定义又无执行语句（空函数）。例如：

```
void dump(void )
{   }
```

（3）在 C++中，每条执行语句和变量定义必须以分号结束。例如：

```
int z;   z=x+y ;
```

（4）C++程序的书写格式。C++程序的书写格式比较自由，一行内可以写多条语句（语句之间用“;”隔开），一条语句也可以分成几行来写。例如：

```
int add(int x,int y)
{   int z;   z=x+y;   return z;}   //将三条语句写在一行内
```

为了便于程序的阅读、修改和相互交流。程序的书写必须符合以下基本规则。

① 同层次语句必须从同一列开始书写，同层次的开花括号必须与对应的闭花括号在同一列上。

② 属于内一层次的语句，必须缩进几个字符，通常缩进两个、四个或八个字符。

③ 任一函数的定义均从第一列开始书写。

（5）C++的输入/输出。C++没有专门的输入/输出语句，输入/输出操作是通过输入/输出流（cin/cout）来实现的。C++默认的标准输入设备是键盘，可形象地将 cin 理解为键盘。因此，cin>>a>>b;表示用键盘输入变量 a 和 b。C++默认的标准输出设备是显示器，可形象地将 cout 理解为显示器。因此，cout<<"Sum="<<sum<<endl;表示将字符串“Sum=”与变量 sum 的值输出到显示器上。

（6）C++严格区分字母的大小写。例如，int a,A;语句表示定义两个不同的变量 a 和 A。

（7）C++注释。在 C++程序的任何位置都可以插入注释信息。注释方法有两种，一种方法是用“/*”和“*/”把注释内容括起来，它可以用在程序中的任何位置。例如：

```
/*求两个整数的和程序*/
```

另一种方法是用“//”字符，它表示从此开始到本行结束为注释内容。例如：

```
//说明变量 a、b、sum 为整型数
```

（8）编译预处理命令。以“#”开头的行称为编译预处理命令。例如，#include <iostream> 表示本程序包含头文件 iostream。

（9）使用命名空间 std 的说明。using namespace std;说明使用命名空间 std。

上述有关函数、输入/输出流等概念会在以后的章节中进行详细介绍。C++程序的基本结构可用图 1.1 表示。

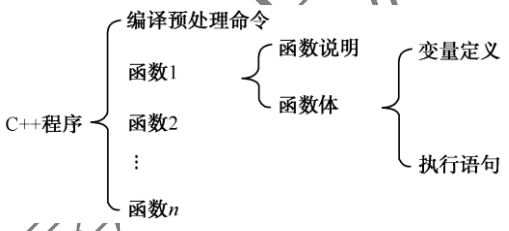


图 1.1　C++程序的基本结构

1.4　C++上机操作

1.4.1　C++程序的开发步骤

C++是一种编译性的语言，在设计好一个C++源程序后，需要经过编译、连接，生成可执行的程序文件，然后执行并调试程序。一个 C++程序的开发可分成如下几步。

（1）分析问题。根据实际问题，分析需求，确定解决方法，并用适当的工具描述它。

（2）编辑程序。编写 C++源程序，并利用一个编辑器将源程序输入计算机的某一个文件中。源程序文件的扩展名为.cpp。

（3）编译程序。编译源程序，生成目标程序。目标程序文件的扩展名为.obj。

（4）连接程序。将一个或多个目标程序与库函数进行连接，生成一个可执行文件。可执行文件的扩展名为.exe。

（5）运行调试程序。运行可执行文件，分析运行结果。若有错误，则进行调试修改。

在编译、连接和运行程序各过程中，都有可能出现错误，此时需要修改源程序，并重复以上步骤，直到得出正确的结果。

1.4.2 C++程序的上机操作方法

本节微课视频

VC++为用户开发 C++程序提供了一个集成环境，这个集成环境包括源程序的输入和编辑、源程序的编译和连接、程序运行时的调试和跟踪、工程的自动管理、为程序的开发提供的各种工具，并具有窗口管理和联机帮助等功能。

使用 VC++集成环境上机调试程序可分成如下几步：进入 VC++集成环境；生成工程；生成和编辑源程序，把一个或多个源程序加入各自的文件中；将源程序文件加入工程中；根据需要改变工程的设置；编译、连接和运行程序。下面以例 1.1 为例来说明 C++程序的上机操作方法，程序的文件名为 example1_1.cpp。

1．启动 VC++

当在桌面上建立 VC++的图标后，可通过双击该图标启动 VC++；若没有建立相应的图标，则可以通过菜单方式启动 VC++，即单击“开始”菜单，选择“程序”命令，选择“Microsoft Visual Studio 6.0”选项启动 VC++。VC++启动成功后，将打开如图 1.2 所示的 VC++集成环境。

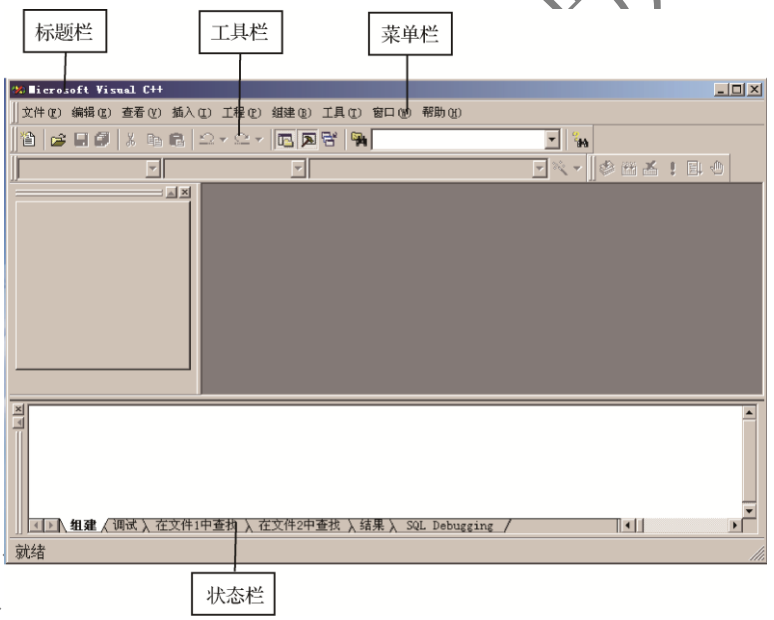


图 1.2 VC++集成环境

VC++集成环境是一个组合窗口。窗口的第一部分为标题栏；第二部分为菜单栏，其中包括“文件”“编辑”“查看”“插入”“工程”“组建”“工具”“窗口”“帮助”菜单；第三部分为工具栏，其中包括常用的工具按钮；第四部分为状态栏，其中包括几个子窗口。

2．生成工程

通常使用工程（项目）的形式来控制和管理 C++程序文件，C++的工程中存放着特定程序的全部信息，包含源程序文件、库文件、建立程序所用的编译器和其他工具的清单。C ++的工程以工程文件的形式存储在磁盘上。

生成工程的操作步骤如下。

（1）选择 VC++集成环境中的“文件”菜单中的“新建”命令，弹出“新建”对话框。

（2）打开“新建”对话框中的“工程”选项卡，如图 1.3 所示，以便生成新的工程。在生成新工程时，系统会自动生成一个工程工作区，并将新生成的工程加入该工程工作区中。

图 1.3 “新建”对话框中的“工程”选项卡

（3）在工程类型清单中，选择“Win32 Console Application”工程，表示要生成一个 Windows 32 位控制台应用程序的工程。

（4）在“位置”文本框中输入存放工程文件的文件夹路径，如 E:\C++。

（5）在“工程名称”文本框中输入工程名，如 example1_1。

（6）“平台”选区中的“Win 32”复选框，表示要开发 32 位的应用程序。

（7）单击“确定”按钮。这时就生成了一个工程文件。系统会自动加上文件扩展名.dsw。例如，系统在文件夹 E:\C++\example1_1 中生成了一个工程文件 example1_1.dsw。

3．生成 C++源程序文件并将其加入工程文件中

操作步骤如下。

（1）选择“文件”菜单中的“新建”命令，弹出“新建”对话框。

（2）打开“新建”对话框中的“文件”选项卡，如图 1.4 所示。

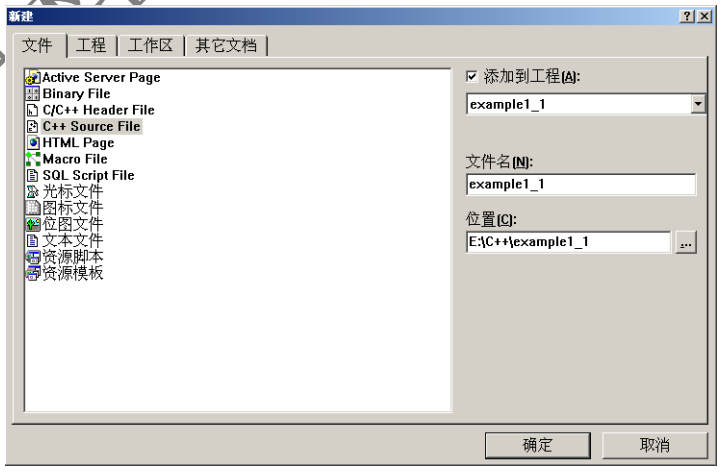


图 1.4 “新建”对话框中的“文件”选项卡

（3）在文件类型清单中，选择“C++ Source File”工程，表示要生成一个 C++源程序。

（4）在“文件名”文本框中输入 C++源程序的文件名。系统会自动添加文件扩展名.cpp，如 example1_1.cpp。

（5）若“添加到工程”复选框没有被选中，则勾选该复选框，表示系统要将指定的源程序文件加入当前的工程文件中。

（6）单击“确定”按钮。这时就生成了一个新的 C++源程序文件，并已被加入当前的工程文件中，打开如图 1.5 所示的窗口。

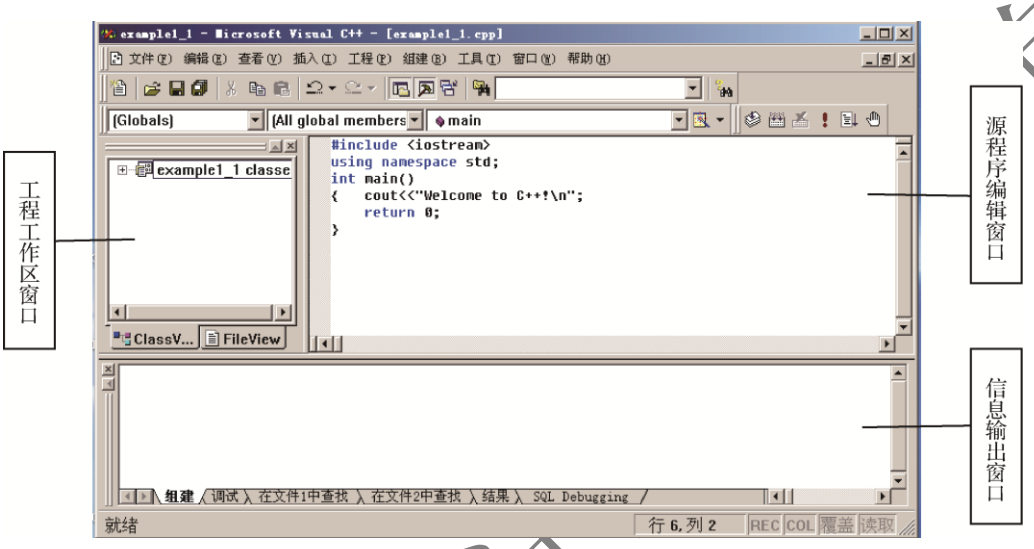


图 1.5 “新建”文件窗口

该窗口有三个子窗口，左边的子窗口为工程工作区窗口；右边的子窗口为源程序编辑窗口，用于输入或编辑源程序；下边的子窗口为信息输出窗口，用来显示出错信息或调试程序的信息。

4．输入和编辑 C++源程序

在源程序编辑窗口中输入例 1.1 中的源程序代码，如图 1.5 所示。

5．保存 C++源程序文件

选择“文件”菜单中的“保存”命令，将源程序保存到相应的文件中。

6．编译和连接源程序文件

选择“组建”菜单中的“编译”或“组建”命令，对源程序进行编译或编译连接，生成可执行文件。系统会自动添加文件扩展名.exe，如 example1_1.exe。

在编译和连接期间，若出现错误，则会在信息输出窗口中显示出错或警告信息。改正错误后，重新编译或编译连接源程序，直到没有错误。

7．运行可执行文件

选择“组建”菜单中的“执行”命令，在 VC++集成环境的控制下运行程序。被启动的程序在控制台窗口运行，这与在 Windows 中运行 DOS 程序的窗口类似。

注意：也可以单击工具栏中的“!”图标或按“Ctrl+F5”组合键，直接编译与运行源程序。

8．打开已存在的工程文件

可采用以下两种方法打开已存在的工程文件。

（1）选择“文件”菜单中的“打开工作空间”命令，然后在弹出的对话框中选择要打开的工程文件。

（2）选择“文件”菜单中的“最近的工作空间”命令，然后选择相应的工程文件。

注意：在调试一个应用程序时，VC++集成环境一次只能打开一个工程文件。当一个程序调试完成，要开始输入另一个程序时，必须先关闭当前的工程文件，然后为新的源程序建立一个新的工程文件。关闭当前的工程文件的方法是：选择“文件”菜单中的“关闭工作空间”命令。

9．退出 VC++集成环境

选择“文件”菜单中的“退出”命令，可以退出 VC++集成环境。

本 章 小 结

1．C++程序的基本结构

（1）C++程序通常由一个或多个函数组成，函数是构成 C++程序的基本单位。C++程序中有且只有一个主函数 main()，一个 C++程序总是从主函数开始执行的。

（2）C++函数由函数说明和函数体两部分组成。函数说明部分包括函数名、函数类型、函数参数（形式参数）及其类型。其中函数类型可以省略，函数参数可以没有，但函数名后面的括号不能省略。

函数体一般包括变量定义和执行语句两部分，并且该部分内容需用花括号括起来。

注意：

(1) C++中的每条语句和数据说明必须以分号结束，分号是 C++语句的必要组成部分。

(2) C++语言没有专门的输入/输出语句，输入/输出操作是通过输入/输出流（cin/cout）来实现的。

(3) 程序的书写必须规范，以便于程序的阅读、修改和相互交流。

(4) 在 C++中，严格区分字母的大小写。

(5) 在 C++程序的任何位置都可以插入注释信息。

2．C++程序的开发步骤

一个 C++程序的开发过程可分成如下几步。

（1）分析问题。

（2）编辑程序，生成扩展名为.cpp 的 C++源程序文件。

（3）编译程序，生成扩展名为.obj 的目标程序文件。
（4）连接程序，生成扩展名为.exe 的可执行文件。
（5）运行调试程序。

习　　题

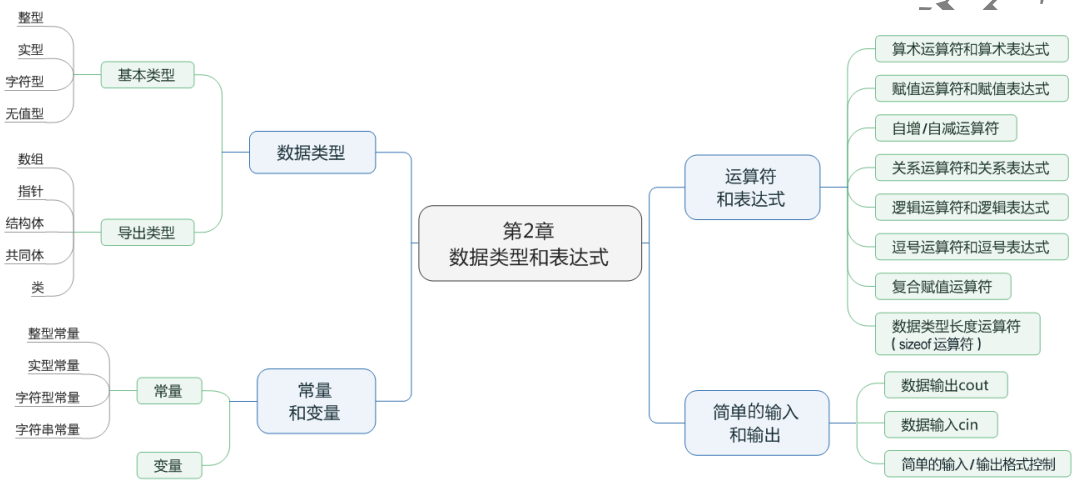
1.1　简述 C++语言程序的结构特点。
1.2　简述 C++程序的开发步骤。
1.3　编写 C++程序要注意哪些问题？
1.4　C++源程序文件的扩展名是什么？
1.5　设计一个 C++程序，输出以下信息：

```
***************
    Hello!
***************
```

1.6　设计一个 C++程序，输入三名学生的成绩，并求其总成绩。
1.7　设计一个 C++程序，输入 a、b 两个整数，并用 sub()函数求两数之差。

第 2 章

数据类型和表达式



本章知识点导图

通过本章的学习，应理解 C++中关键字和标识符的概念，掌握标识符的命名方法。理解 C++中数据类型的种类，掌握基本数据类型的使用方法，理解常量和变量的概念，掌握常量的分类、用法，以及变量的说明、赋初值的方法。理解运算符的优先级和结合性的概念，掌握算术运算符、赋值运算符、自增/自减运算符、关系运算符、逻辑运算符、逗号运算符、复合赋值运算符、数据类型长度运算符的使用，以及由它们构成的表达式的写法和求值方法。掌握使用 cout 和 cin 进行简单的输入和输出的方法。

在 C++中，表达式是由常量、变量、函数等用运算符连接而成的式子。例如，表达式 x*x+2*x－1.5 +sin(x) 是由常量 2 与 1.5、变量 x、函数 sin(x)用运算符“*”“+”“－”连接而成的。要掌握 C++中表达式的使用，必须先了解 C++中与常量、变量、运算符有关的概念。常量分为整数、实数、字符等类型，如整数常量 2，实数常量 1.5 等；同样，变量也分为整数、实数、字符等类型，因此在介绍表达式之前，先介绍 C++中数据类型、常量、变量、运算符的概念，然后介绍表达式的概念。本章内

容包括数据类型、常量、变量、运算符和表达式，以及简单的输入和输出。

2.1 数据类型

由第 1 章可知，C++函数体由变量定义与执行语句两部分组成。例如：

```
#include <iostream>
using namespace std;
int main( )
{     int a,b,sum;
      char c;
      float x, y;
      cout << "Input a,b:";
      cin >> a >> b;
      sum = a + b;
      cout << "Sum=" << sum << endl;
      return 0;
}
```

其中，变量定义 int a,b,sum; 将 a、b、sum 定义为整型变量，char c; 将 c 定义为字符型变量，float x,y; 将 x、y 定义为实型变量；执行语句用于进行数据的输入、计算与输出。C++为何要定义变量，即定义变量的目的是什么。现介绍如下。

1．C++定义变量的目的

（1）为变量分配存储空间。例如，系统为整型变量 a 分配 4B 的存储空间，而为字符型变量 c 分配 1B 的存储空间，为实型变量 x 分配 4B 的存储空间，如图 2.1 所示。

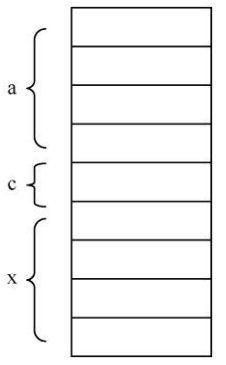


图 2.1　变量内存分配

（2）规定变量可以适用的运算。例如，对整型变量可以进行求余运算，而对实型变量则不能进行求余运算。

只有通过变量定义，给变量分配存储空间，并规定其可以适用的运算，即给变量规定数据类型，变量才能参与各种程序的操作运算。

2．数据类型分类

在 C++中，数据类型分为两大类：基本类型和导出类型，如图 2.2 所示。

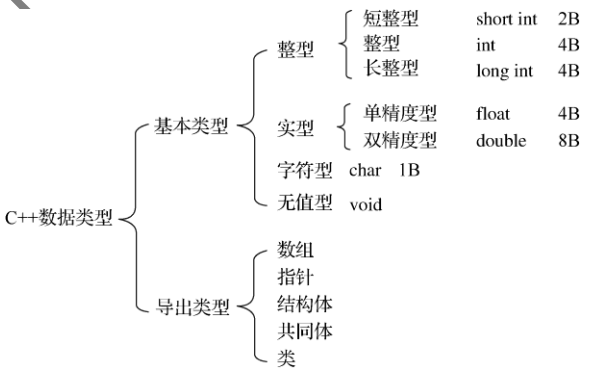


图 2.2　C++的数据类型

基本类型是 C++中预定义的数据类型，所用的定义符号与存储空间占用字节数如图 2.2 所示。导

出类型是用户根据程序设计的需要，按 C++的语法规则由基本类型构造的数据类型。本节只介绍基本类型，导出类型会在后面有关章节中进行介绍。

字符型用来存放一个 ASCII 字符或存放一个 8 位的二进制数；整型用来存放一个整数，其所占字节数随不同型号的计算机而异，可以占用 2B 或 4B（在 32 位的计算机上占用 4B）；实型（分为单精度型和双精度型）用来存放实数。

对于字符型，可分为无符号型和有符号型；对于整型，可分为长整型和短整型、有符号长整型和有符号短整型、无符号长整型和无符号短整型；对于实型、可分为单精度型和双精度型。这些类型可通过在基本类型前面加上以下几个修饰词组合而成：

signed	有符号型
unsigned	无符号型
long	长型
short	短型

例如，unsigned char 为无符号字符型；long int 为长整型；unsigned short int 为无符号短整型；long double 为长双精度型。

当用上述 4 个修饰词来修饰 int 时，关键字 int 可以省略。例如，long 等同于 long int，unsigned 等同于 unsigned int；另外，在 C++中，无修饰词的 int 和 char，编译系统认为它是有符号的，相当于加了修饰词 signed。

这些修饰词与基本类型组合后的数据类型，如表 2.1 所示。

表 2.1　C++中的基本数据类型

数 据 类 型	名　　称	占用字节数	取 值 范 围
char(signed char)	字符型（有符号字符型）	1	−128～127
unsigned char	无符号字符型	1	0～255
short(signed short)	短整型（有符号短整型）	2	−32 768～32 767
unsigned short	无符号短整型	2	0～65 535
int(signed)	整型（有符号整型）	4	-2^{31}～（$2^{31}-1$）
unsigned int	无符号整型	4	0～（$2^{32}-1$）
long int(signed long)	长整型（有符号长整型）	4	-2^{31}～（$2^{31}-1$）
unsigned long	无符号长整型	4	0～（$2^{32}-1$）
float	单精度型	4	-10^{38}～10^{38}
double	双精度型	8	-10^{308}～10^{308}
long double	长双精度型	8	-10^{308}～10^{308}

在 C++中，有符号整数在计算机内是以二进制补码形式存储的，其最高位为符号位，“0”表示正，“1”表示负；无符号整数只能是正数，在计算机内是以绝对值形式存储的。

2.2　常量和变量

2.2.1　常量

在程序执行过程中，其值不能被改变的量称为常量。根据基本数据类型，常量可分为整型常量、实型常量、字符型常量、字符串常量 4 种。

1．整型常量

整型常量，即整数。在C++中，整数可用十进制整数、八进制整数、十六进制整数3种形式表示。

（1）十进制整数。除表示正负号的符号外（“+”号可省略），以1～9开头的整数为十进制整数，如123，−456。

（2）八进制整数。以0开头的整数为八进制整数，如0123，0367。

（3）十六进制整数。以0X或0x开头的整数为十六进制整数，如0X123，0x1ABF，0XABF2。

（4）长整型数与无符号整型数。以L或l结尾的整数为长整型数，如123L，0456l，0X5AL；以U或u结尾的整数为无符号整型数，如23U，0456u，0X3BU；以UL（或ul）或LU（或lu）结尾的整数为无符号长整型数，如24UL，0X95LU。

说明：当没有明确指定某常数为长整型数或无符号长整型数时，编译时由编译系统根据常数的大小自动识别。

2．实型常量

实型常量，即实数，在C++中，它又被称为浮点数。在C++中，实数可用以下两种形式来表示。

（1）十进制小数形式（又称定点数、日常记数法）。它由数字0～9、小数点、正负号组成，如0.123，.123，123.0，123.，0.0，−56.。

注意：必须要有小数点。当整数（或小数）部分为0时，整数（或小数）部分可省略。

（2）指数形式（又称浮点数、科学计数法）。它以10的多少次方表示，由数字、小数点、正负号、E（e）组成。例如，5E6，6.02e−3，−1.0e8，3.0e−4，分别表示5×10^{6}，6.02×10^{-3}，-1.0×10^{8}，3.0×10^{-4}。

注意：字母E（e）的前后必须要有数字，且E（e）后面的指数必须为整数。

3．字符型常量

用单引号引起来的单个字符称为字符型常量，如'a'，'x'，'D'，'?'，'$'，' '，'3'。

注意：C++的字符型常量只能为单个字符，对于字母字符，区分大小写。字符型常量只能用单引号引起来，而不能用双引号。

字符型常量在计算机内是采用该字符的ASCII码值来表示的，其数据类型为char型。

字符型常量有两种表示形式，即普通字符和转义字符。

（1）普通字符，即可显示字符，如'a'，'A'，'#'，' '，'0'。

（2）转义字符，即以反斜杠“\”开头，后面跟一个字符或一个字符的ASCII码值的形式来表示一个字符。

在“\”后面跟一个字符常用来表示一些控制字符。例如，\n可以用来表示换行符。

若“\”后面跟一个字符的ASCII码值，则必须是一个字符的ASCII码值的八进制数形式或十六进制数形式，取值范围必须为0～255，表示形式为\ddd，\xhh。其中，ddd表示三位八进制数、hh表示两位十六进制数。例如，\101，\x41，都可以用来表示字母A。

普通字符用来表示可显示字符；转义字符可用来表示任一字符。控制字符或不可以通过键盘输

入的字符只能用转义字符来表示；但可显示字符直接用单引号引起来会更加直观。

C++中预定义的转义字符及其含义如表 2.2 所示。

表 2.2　C++中预定义的转义字符及其含义

转 义 字 符	名　　称	功能或用途
\a	响铃符	用于输出
\b	退格符（“Backspace”键）	用于退回一个字符
\f	换页符	用于输出
\n	换行符	用于输出
\r	回车符	用于输出
\t	水平制表符（“Tab”键）	用于制表
\v	纵向制表符	用于制表
\\	反斜杠符	用于输出或在文件的路径名中使用
\'	单引号	用于输出单引号
\"	双引号	用于输出双引号

例如，有下列程序段：

```
cout<<"ab\tcd\n";
cout<<"ef\t\bgh\n";
```

第一条语句表示先在第 1 行的第 1 列输出 a，第 2 列输出 b。\t 用于制表，即跳到下一个制表位置，一个制表区占 8 列，即下一个制表位置从第 9 列开始，因此在第 9 列输出 c，第 10 列输出 d。\n，即换行，就是将当前位置移到下一行的开头。第二条语句表示先在第 2 行的第 1 列输出 e，第 2 列输出 f。\t 表示当前位置跳到第 9 列，\b 的作用是退格，即当前位置退到第 8 列，因此在第 8 列输出 g，第 9 列输出 h，最后为\n，进行换行，将当前位置移到下一行的开头准备下一次输出。最终运行结果为：

```
ab      cd
ef      gh
```

注意：对于反斜杠、单引号、双引号字符，尽管它们既可显示又可通过键盘输入，但由于它们在 C++中有特殊的用法（“\”表示转义字符，“'”表示字符常量，“"”表示字符串常量），所以当它们作为字符型常量出现时，也要用转义字符形式来表示。例如，双引号字符表示为'\"'。字符常量的表示方法可归纳为如图 2.3 所示的形式，由图 2.3 可知，字母 A 的表示方法有三种：'A'、'\101'、'\x41'。

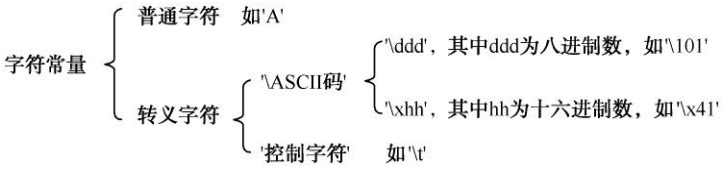


图 2.3　字符常量的表示方法

4．字符串常量

用双引号引起来的若干字符称为字符串常量（简称字符串），如"How do you do!"，"China"，"a"，

"$123.45"。

注意：字符常量和字符串常量的区别如下。

(1) 字符常量为单个字符，字符串常量可以是多个字符。

(2) 分界符不同，字符常量为单引号，字符串常量为双引号。

(3) 字符串常量的结尾有一个字符串结束标志，而字符常量没有。

在 C++中，每个字符串在结尾会自动加一个字符串结束标志，以便系统据此判断字符串是否结束。C++规定以字符'\0'（空操作）作为字符串结束标志。例如，"China"在内存中的存储方式如图 2.4 所示。又如，"a"是一个字符串常量，'a'是一个字符常量；因此两者在内存中的存储方式不同，"a"占用 2B（'a'与'\0'），而'a'只占用 1B（'a'）。

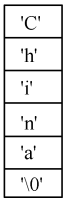


图 2.4 "China"在内存中的存储方式

当双引号作为字符串中的一个字符时，必须采用转义字符表示法；而当单引号作为字符串中的一个字符时，可直接出现在字符串中，也可以采用转义字符表示法。例如：

"\"Beijing\""，"Mark's"，"Mark\'s"

2.2.2 变量

在程序的执行过程中，其值可以改变的量称为变量。变量必须用标识符来命名。变量根据其取值的不同可分为不同的类型：整型变量、实型变量、字符型变量、指针型变量等。对于任何一个变量，编译程序都要为其分配若干字节的内存单元，以存储变量的值。当要改变变量的值时，将新值存放到变量的内存单元中即可；当要用变量的值时，从变量内存单元中取出数据即可。不管什么类型的变量，通常都必须先定义后使用。

1. 变量定义

在 C++中，变量定义也称为变量说明，变量定义的一般格式为：

〔存储类型〕<类型><变量名 1>〔,<变量名 2>,…,<变量名 *n*>〕;

其中，用“〔〕”括起来的是可选择部分，用“< >”括起来的是一个语法单位，“…”表示该部分可以多次重复，以后均采用这种表示方法。存储类型为变量的存储类型（会在第 5 章中介绍）；类型是变量的数据类型，它可以是 C++中预定义的数据类型，也可以是用户自定义的数据类型；变量名是用户给变量起的名字，用标识符作为变量的名字。例如：

```
int   a,b;                //定义了两个整型变量 a、b
float x,y;                //定义了两个实型变量 x、y
char c;                   //定义了一个字符型变量 c
```

在上述定义中，int、float、char 分别是整型、实型、字符型数据类型的关键字；a、b、x、y、c 是变量名的标识符。

2. 关键字

关键字（保留字）是 C++中一批具有特定含义和用途的英文单词。C++中共有 48 个关键字，可以分成说明符、类型说明符、访问说明符、语句、标号、运算符等，如 auto、int、private、if、case 、new 等，详见附录 A。用户不能使用关键字作为标识符。

3．标识符

标识符是用来标识变量名、符号常量名、函数名、类型名、文件名等实体名字的有效字符序列。标识符只能由字母、数字和下画线 3 种字符组成，且第一个字符必须是字母或下画线。例如，下面的字符序列都符合标识符的定义，可以用作标识符：

```
Average     Value     length     Class_1
```

而下面的字符序列都不符合标识符的定义，不可以用作标识符：

```
6book                    //不能以数字开头
#abc                     //不能使用符号#
s4.6                     //不能使用小数点
if                       //if 为关键字，不能用作标识符
```

对于标识符，还需注意以下几点。

（1）标识符中的字母。在 C++中，大写字母和小写字母被认为是两个不同的字符。例如，BOOK 和 book 被认为是两个不同的标识符。

（2）标识符的有效长度。在 C++中，一个有效的标识符的长度为 1～247。当一个标识符的长度超过 247 时，前面的 247 个字符有效，其余字符无效。

（3）标识符的命名方法。通常，为了增强程序的可读性，应使用表示标识符含义的英文单词（或其缩写）或汉语拼音来命名标识符。例如，用 Average 表示平均值。

在 C++中，变量定义是作为变量定义语句来处理的。因此，变量定义语句可以出现在程序中语句可以出现的任何位置。对同一个变量只能进行一次定义性说明。改变一个变量的值，称为对变量进行赋值；取一个变量的值，称为对变量进行使用。只要对变量进行了定义性说明，就可以多次使用该变量。

4．变量赋初值

当首次使用一个变量时，该变量应有一个初值。在 C++中，可用以下两种方法给变量赋初值。

（1）在定义变量时直接赋初值。例如：

```
int   a=3,b=4;          //定义整型变量 a、b，并使它们的初值分别为 3、4
float   x=3.5;          //定义实型变量 x，并使它的初值为 3.5
char   c='a';           //定义字符型变量 c，并使它的初值为'a'
```

（2）使用赋值语句赋初值。例如：

```
int n;
float e;
n=10;                   //使变量 n 的值为 10
e=2.718;                //使变量 e 的值为 2.718
```

2.3　运算符和表达式

在 C++中，表达式是由常量、变量、函数等用运算符连接而成的式子，2.2 节已介绍了常量与变量，

而函数会在第 5 章中讲述。因此，在讲解表达式前先对运算符进行介绍。本节介绍算术运算符，赋值运算符、自增/自减运算符、关系运算符、逻辑运算符、逗号运算符，以及由它们构成的相应的表达式。

2.3.1 算术运算符和算术表达式

1．算术运算符

基本算术运算符有：

+	加法运算符、正值运算符
−	减法运算符、负值运算符
*	乘法运算符
/	除法运算符
%	求余运算符（求模运算符）

加法、减法、乘法、除法、求余运算符都是双目运算符（又称为二元运算符），要求有两个操作对象（操作数）；正值、负值运算符是单目运算符（又称为一元运算符），只需要一个操作对象（操作数）。在 C++中，除法运算比较复杂，大致有以下 3 种情况。

（1）两个整数相除（/）得到整数商，如 3/4=0。

（2）两个整数相余（%）得到余数，如 3%4=3。

（3）整数与实数相除（/）得到双精度小数，如 3/4.0=0.75。

2．算术表达式

（1）算术表达式。用算术运算符和括号将运算对象连接起来的符合 C++语法规则的式子称为 C++的算术表达式。运算对象可以是常量、变量或函数等，如 a*b/c−1.5+'a'+2*sin(x)。

对于上面的算术表达式，先进行什么运算，后进行什么运算，是运算符的优先级与结合性要解决的问题。

（2）算术运算符的优先级。算术运算符的优先级从高到低的次序是“±”→“*”“/”“%”→“+”“−”，即正值与负值运算符优先级最高，然后按先乘除后加减的优先级排序。其中“→”左边运算符的优先级高于右边运算符的优先级。在对表达式求值时，如果表达式中包含多个运算符，则应按运算符的优先级从高到低依次执行。例如：

```
a−b*c                    //先进行“*”运算，再进行“−”运算
```

可使用括号改变运算符的优先级，此时应先计算括号内的表达式，再计算括号外的表达式。

在对表达式求值时，若一个操作对象两侧的运算符的优先级相同，那么先进行什么运算，后进行什么运算，是运算符结合性要解决的问题。

（3）算术运算符的结合性。结合性有以下两种。

① 左结合性：操作对象先与左面的运算符结合，即从左到右进行运算。

② 右结合性：操作对象先与右面的运算符结合，即从右到左进行运算。

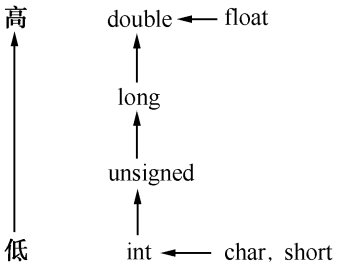
算术运算符的结合性为左结合性（从左到右运算）。例如，有算术表达式 a−b+c，由于操作对象 b 两侧的运算符“−”与“+”的优先级相同，根据算术运算符的左结合性，操作对象 b 先与左面的运算符“−”结合，所以先进行“−”运算，再进行“+”运算。

3．不同类型数据混合运算时的数据类型转换

在 C++中，允许整型、实型、双精度型、字符型等不同类型的数据进行混合运算，如 10+'a'+1.5−8765.1234*'b'。

在进行运算时，不同类型的运算对象要先转换成同一类型，再进行运算。数据类型的转换有两种方式：自动类型转换和强制类型转换。

（1）自动类型转换。转换规则为：



其中，横向向左的箭头表示运算时必须进行的转换；纵向向上的箭头表示当操作对象为不同类型时转换的方向。

注意：纵向箭头的方向只表示数据类型级别的高低，由低级向高级转换，不能理解为 int 型首先转换成 unsigned 型，其次转换成 long 型，最后转换成 double 型。例如，有变量定义：

```
int       i=1;
float     f=1.0;
double    d=4.0;
long      e=2L;
```

则对于表达式 10+'a'+i*f−d/e，按算术运算符的优先级和结合性的次序进行，具体求值过程如下。

① 将'a'→int（ASCII 码值为 97），10+'a'=10+97=107，运算结果为整型数 107，10+'a'+i*f−d/e = 107+i*f−d/e。

② 将 i、f→double，i*f=1.0*1.0=1.0，运算结果为双精度型数 1.0，107+i*f−d/e=107+1.0−d/e。

③ 将 107→double，107.0+1.0=108.0，107+1.0−d/e=108.0−d/e。

④ 将 e→double，d/e=4.0/2.0=2.0，运算结果为双精度型数 2.0，108.0−d/e=108.0−2.0。

⑤ 计算 108.0−2.0，运算结果为双精度型数 106.0。

其中，“→”表示类型转换。

注意：上述类型转换都是由系统自动进行的，不需要人工干预，并且只是在运算时对转换对象的运算值进行转换，而该对象在内存单元中的内容和类型并没有改变。

（2）强制类型转换。强制类型转换的格式为：

（<类型>）<表达式>

或

<类型>（<表达式>）

表示先求出表达式的值，然后由系统强制性地将该值的类型转换为由类型名指定的数据类型。例如：

```
int a=12;
double b;
float x=1.2,y=3.4;
```

```
b=(double)a;                    //将整型变量 a 的值强制转换成双精度型 12.0，并赋给 b
a=int(x+y);                     //将实型变量 x、y 的和 4.6 强制转换成整型数 4，并赋给 a
a=(int)x%(int)y;                //将实型变量 x、y 的值转换成整型后求余（1%3=1），并赋给 a
```

上述强制类型转换也只是在运算时对转换对象的运算值进行转换，而该对象在内存单元中的内容和类型并没有改变。强制类型转换的优先级高于基本算术运算的优先级。

2.3.2 赋值运算符和赋值表达式

1．赋值运算符“=”

赋值运算符“=”的作用是将一个数值或一个表达式的值赋给一个变量。例如：

```
a=10                            //将 10 赋给变量 a
x=a+10                          //首先将变量 a 的值加 10，然后赋给变量 x
```

赋值运算符具有计算和赋值双重功能，即先计算表达式的值，再将该值赋给指定的变量。

2．赋值表达式

用赋值运算符将一个变量和一个表达式连接起来的式子称为赋值表达式，如 a=b+5。

（1）赋值表达式定义格式。赋值表达式定义格式如下：

```
<变量>= <表达式>
```

赋值表达式的求解过程为：先求出赋值运算符右侧表达式的值，然后将该值赋给赋值运算符左侧的变量。左侧变量的值就是整个赋值表达式的值。例如：

```
a=3+5                           //a=8，整个赋值表达式的值是 8
a=a+1                           //a=a+1=8+1=9，整个赋值表达式的值是 9
a=a−1                           //a=a−1=9−1=8，整个赋值表达式的值是 8
```

（2）允许赋值运算符右侧的表达式是另一个赋值表达式。例如，a=(b=5)，则 b 的值为 5，赋值表达式 b=5 的值为 5，a 的值也为 5，即整个赋值表达式的值为 5。

（3）赋值运算符的优先级与结合性。赋值运算符的优先级低于算术运算符的优先级，其结合性为右结合性，即从右到左进行运算。例如：

```
a=b=c=5                         //该表达式等价于 a=(b=(c=5))，得 a、b、c 的值均为 5
                                //整个表达式的值为 5
a=5+(c=6)                       //先进行 c=6 的赋值运算，得 c 的值为 6
                                //再进行 a=5+6 的赋值运算，得 a 的值为 11，整个表达式的值为 11
a=(b=4)+(c=6)                   //b 的值为 4，c 的值为 6，得 a 的值为 10，整个表达式的值为 10
```

赋值表达式 a=a+1 的作用是将变量 a 的值加 1，赋值表达式 a=a−1 的作用是将变量 a 的值减 1。C++为方便用户完成对变量进行加 1 与减 1 的运算，提供了变量自增和自减运算符。

3．赋值运算时数据类型的转换

在进行赋值运算时，若赋值运算符右侧表达式的值的数据类型与其左侧变量数据类型不一致但类型兼容（可进行类型转换），那么系统将会自动进行数据类型转换。类型转换的一般原则为：将赋值运算符右侧表达式的值转换为赋值运算符左侧变量所属的类型，若右侧表达式的值在转换后超出了左侧变量的取值范围，则赋值结果错误。例如：

```
int   i,j,k;
float   f,g,h;
i=1;j=2;f=1.2;g=3.4;
```

```
h=i*j;                          //i*j 的值为 2，转换成实数 2.0 赋给 h，h 的值为 2.0
k=f*g;                          //f*g 的值为 4.08，转换成整数 4 赋给 k，k 的值为 4
```

2.3.3 自增/自减运算符

1．自增运算符“++”

（1）自增运算符“++”的作用是使变量的值加 1。

（2）自增运算符为单目运算符，只需要一个操作对象。

（3）自增运算符有前置和后置两种形式。

① 前置运算为先自加后引用。例如：

```
++i                             //表示先使 i 加 1，后引用 i
```

② 后置运算为先引用后自加。例如：

```
i++                             //表示先引用 i，后使 i 加 1
```

例如，设有如下程序段：

```
int i=3,j=3,m,n;
m=++i;                          //先将变量 i 自加为 4，然后将自加后的值（4）赋给 m
n=j++;                          //先将变量 j 的值（3）赋给变量 n，然后将 j 自加为 4
```

则 m 的值为 4，i 的值为 4，n 的值为 3，j 的值为 4。

2．自减运算符“--”

（1）自减运算符“--”的作用是使变量的值减 1。

（2）自减运算符为单目运算符，只需要一个操作对象。

（3）自减运算符有前置和后置两种形式。

① 前置运算为先自减后引用。例如：

```
--i                             //表示先使 i 减 1，后引用 i
```

② 后置运算为先引用后自减。例如：

```
i--                             //表示先引用 i，后使 i 减 1
```

例如，设有如下程序段：

```
int i=3,j=3,m,n;
m=--i;                          //先将变量 i 自减为 2，然后将自减后的值（2）赋给 m
n=j--;                          //先将变量 j 的值（3）赋给变量 n，然后将 j 自减为 2
```

则 m 的值为 2，i 的值为 2，n 的值为 3，j 的值为 2。

注意：自增运算符（++）和自减运算符（--）的操作对象只能是变量，不能是常量和表达式，且变量的数据类型通常为整型。

3．自增/自减运算符的优先级

自增运算符（++）和自减运算符（--）的优先级高于基本算术运算符的优先级，与正值、负值运算符（+、-）的优先级相同。

2.3.4 关系运算符和关系表达式

1. 关系运算符

（1）关系运算符有 6 种，分别是：>（大于）、>=（大于或等于）、<（小于）、<=（小于或等于）、==（等于）、!=（不等于）。

（2）关系运算符的作用：用于比较两个操作对象的大小，比较结果用逻辑值“真”或“假”来表示。

（3）逻辑值：在 C++中，用数字 1 表示“真”，用数字 0 表示“假”。例如，若 x=9，y=3，则 x>y 为真，运算结果为 1；而 x<y 为假，运算结果为 0。

（4）关系运算符的优先级：关系运算符的优先级低于算术运算符的优先级，高于赋值运算符的优先级。优先级由高到低的次序为：自增/自减运算符→算术运算符→>、>=、<、<=→==、!=→赋值运算符（=）。

（5）关系运算的结果可作为一个整数参与表达式的运算。例如：

```
int a=3,b=5,y;
y=a<b+3;
```

按优先级次序（“+” →“<” →“=”），y=a<b+3 等价于 y=(a<(b+3))，因此 y=1。

（6）关系运算符具有左结合性。C++允许用户进行如 3<x<6 的运算，当 x=5 时，3<x<6 应为多少？

按左结合性，3<x<6 应先进行 3<x 的比较，结果为 1（真），再进行 1<6 的比较，结果为 1（真）。

（7）关系运算符都是双目运算符，要求有两个操作对象。

2. 关系表达式

用关系运算符将两个操作对象连接起来的式子称为关系表达式。例如，5>3，x+5<y−3。

关系表达式的值是一个逻辑值，即真或假，用 1 或 0 来表示，代表关系表达式成立或不成立。

例如，关系表达式 5>3 成立，即真，其值为 1。

当 x=1，y=2 时，由于“+”优先级高于“<”，所以 x+5<y−3 等价于（1+5）<（2−3），即 6<−1，该式不成立，即“假”，其值为 0。

问题：能否用关系表达式 3<=x<=6 来判断 x 是否在闭区间[3,6]内？

假设 x=7，按左结合性，应先进行 3<=x 的比较，结果为 1（真），然后进行 1<=6 的比较，结果为 1（真），即关系表达式 3<=x<=6 的值为 1，这说明 x 在闭区间[3,6]内。但事实上 x=7 并不在闭区间[3,6]内。问题出在第一个关系运算 3<=x 得到结果为 1，再用 1 与 6 比较得出真。由此可见，在判断变量是否在某范围内时，不能用数学中的关系表达式来判断，而要用“3<=x 且 x<=6”这样的逻辑表达式来判断。

2.3.5 逻辑运算符和逻辑表达式

1. 逻辑运算符

（1）逻辑运算符有三个，分别是逻辑与“&&”、逻辑或“||”、逻辑非“!”。

① 逻辑与“&&”是双目运算符，要求有两个操作对象。当两个操作对象的值均为真时，逻辑与运算的结果为真，其值为 1；否则为假，其值为 0。逻辑与的记忆口诀为：全真为真，有假为假。例如：

```
x && y                    //若 x、y 都为真，则 x && y 为真，其值为 1
```

又如，当 x=7 时，3<=x && x<=6。该表达式等价于（3<=x）&&（x<=6），即（3<=7）&&（7<=6），得 1 && 0，结果为 0，表达式为假。这表明 x 不在闭区间[3,6]内。

② 逻辑或“||”也是双目运算符，要求有两个操作对象。当两个操作对象的值均为假时，逻辑或运算的结果为假，其值为 0；否则为真，其值为 1。逻辑或的记忆口诀为：全假为假，有真为真。例如：

```
x || y                          //若 x、y 都为假，则 x || y 为假，其值为“0”
```

③ 逻辑非“!”是单目运算符，只要求有一个操作对象。当操作对象的值为真时，逻辑非运算的结果为假，其值为 0；否则为真，其值为 1。例如：

```
! x                             //若 x 为真，则! x 为假，其值为 0
```

（2）逻辑运算符的优先级：逻辑运算符中逻辑非“!”的优先级最高，逻辑与“&&”次之，逻辑或“||”最低。例如：

```
a && b || !c                    //首先进行！运算，其次进行&&运算，最后进行||运算
```

逻辑运算符中的逻辑与“&&”和逻辑或“||”的优先级低于关系运算符的优先级，但高于赋值运算符的优先级；逻辑非“!”的优先级高于算术运算符的优先级。逻辑运算符与算术、关系、赋值等运算符之间的优先级关系为：( )→单目运算符（！、+、−、类型转换）→算术运算符（*、/、%→+、−）→关系运算符（>、>=、<、<= →= =、!=）→逻辑运算符（&&→||）→赋值运算符（=）

（3）逻辑运算符的结合性：逻辑与“&&”和逻辑或“||”具有左结合性，而逻辑非“!”则具有右结合性。

2．逻辑表达式

（1）逻辑表达式的定义。用逻辑运算符将操作对象连接起来的式子称为逻辑表达式。逻辑表达式的值是一个逻辑值，即真或假，用 1 或 0 来表示。例如：

逻辑表达式 5>3 && 3>1　结果为真，即值为 1；

逻辑表达式 5<3 || 3<1，结果为假，即值为 0。

逻辑运算的结果也可作为一个整数参与表达式的运算。

（2）操作对象逻辑值的表示方法。在 C++中，1 代表真，0 代表假；但在判断一个操作对象逻辑值时，以非 0 作为真，以 0 作为假。例如，在逻辑表达式 3 && 5>3 中，逻辑与“&&”左侧的 3 表示真，右侧关系表达式 5>3 为真，因此逻辑与的结果为真，即值为 1。

（3）“&&”运算符具有左结合性，当“&&”左侧表达式为 0 时，逻辑值也为 0，因此不再对右侧的表达式进行计算。例如：

```
x=3,y=4
z=x>y && (y=5)
```

根据优先级，上述表达式的执行次序是，先计算“&&”左侧的关系表达式 x>y，因为 3>4 不成立，为假，所以表达式的值为 0，逻辑表达式 x>y && (y=5)的逻辑值为 0。此时，不再对右侧的赋值表达式 y=5 进行计算。因此，运算结果是 x=3、y=4、z=0。

（4）“||”运算符具有左结合性，当“||”左侧表达式为 1 时，逻辑值为 1，因此不再对右侧的表达式进行计算。例如：

```
x=4,y=3
z=x>y || (y=5)
```

根据优先级，上述表达式的执行次序是，先计算“||”左侧的关系表达式 x>y，因为 4>3 成立，

为真，所以表达式的值为 1，逻辑表达式 x>y || (y=5)的逻辑值为 1。此时，不再对右侧的赋值表达式 y=5 进行计算。因此，运算结果是 x=4、y=3、z=1。

【例 2.1】　设有如下变量定义：

```
int a=10,b=20,c=5;
float x=10.5,y=20.5;
```

求下列逻辑表达式的值：

```
a<b && x>y || a<b-!c
```

上式的求值顺序为：首先求出 a<b 的值为 1，x>y 的值为 0，1&&0 的值为 0，其次求出!c 的值为 0，再次求出 b-0 的值为 20，然后求出 a<20 的值为 1，最后求出 0||1 的值为 1。因此整个表达式的值为 1。

【例 2.2】　写出判断字符型变量 c 中的字符是否为字母的逻辑表达式。

判断字符型变量 c 中的字符是否为字母，要分解为判断字符型变量 c 中的字符是否为小写字母和判断字符型变量 c 中的字符是否为大写字母两种情况。判断字符型变量 c 中的字符为小写字母的条件是'a'<=c 且 c<='z'，即'a'<=c && c<='z'。判断字符型变量 c 中的字符为大写字母的条件是'A'<=c 且 c<='Z'，即'A'<=c && c<='Z'。两个条件中只要有一个成立即可，因此逻辑表达式为：

```
'a'<=c && c<='z' || 'A'<=c && c<='Z'
```

【例 2.3】　已知年份（year），要求写出判断年份是否为闰年的逻辑表达式。

判断闰年的条件是：year 能被 400 整除或 year 能被 4 整除且不能被 100 整除。因此逻辑表达式为：

```
year % 400==0 || year % 4==0 && year % 100!=0
```

2.3.6　逗号运算符和逗号表达式

1．逗号运算符

在 C++中，逗号“,”可作为运算符，称为逗号运算符。

2．逗号表达式

用逗号运算符将多个表达式连接起来的式子称为逗号表达式。逗号表达式的格式为：

<表达式 1>,<表达式 2>,…,<表达式 *n*>

逗号表达式的求解过程为：按从左到右的顺序依次求出各表达式的值，并把最后一个表达式的值作为整个逗号表达式的值。

3．逗号运算符的优先级

逗号运算符的优先级是最低的。例如，有变量定义：

```
int  b=1,c=2,d=3;
```

则逗号表达式：

```
a=4+4,b=b*b+c,d=d*a+b
```

的求值过程为：先将 8 赋给 a，再将 3（1*1+2=3）赋给 b，最后一个表达式 d=3*8+3=27，则整个逗号表达式的值为 27。

注意：并非所有逗号都是用来构成逗号表达式的。例如：

```
max(a+b,c-d)
```

其中，“a+b,c-d”并不是一个逗号表达式，而是 max()函数的两个参数，此时，逗号用来分隔两个参数。

已介绍过的基本运算符的优先级从高到低依次为：()→单目运算符（！、+、-、类型转换）→算术运算符（*、/、%→+、-）→关系运算符（>、>=、<、<=→==、!=）→逻辑运算符（&&→||）→赋值运算符（=）→逗号运算符（,）。

2.3.7 复合赋值运算符

1．复合赋值运算符及其表达式

在 C++中，所有双目算术运算符均可与赋值运算符组成一个单一的运算符，这种运算符称为复合赋值运算符。它们是：+=（加等），-=（减等），*=（乘等），/=（除等），%=（求余等）。例如：

```
a+=3                    //相当于 a=a+3
x*=y+8                  //相当于 x=x*(y+8)
x%=3                    //相当于 x=x%3
```

使用复合赋值运算符不仅可以简化表达式的书写形式，还可以提高表达式的运算速度。

2．赋值表达式可包含复合赋值运算符

例如，表达式 a+=a-=a*a，若 a 的初值为 12，该赋值表达式的运算过程如下。

（1）先进行 a-=a*a 的运算，它相当于 a=a-a*a，即 a=12-12*12=12-144=-132，得 a 的值为-132，即表达式 a-=a*a 的值为-132。

（2）再进行 a+=-132 的运算，它相当于 a=a+（-132），即 a=-132+（-132），得 a 的值为-264。因此整个表达式的值为-264。

2.3.8 数据类型长度运算符（sizeof 运算符）

sizeof 运算符用于计算某种类型的操作对象在计算机中所占的存储空间的字节数。sizeof 运算符的使用格式为：

```
sizeof（<类型>）
```

或

```
sizeof（<表达式>）
```

运算结果为“类型”所指定的数据类型或“表达式”的结果类型所占的字节数。例如：

```
sizeof(float)           //值为 4
sizeof(4+10)            //值为 4
sizeof(char)            //值为 1
```

注意：在计算过程中，并不对括号内表达式本身求值。

在 C++中，还有一种位运算符，由于篇幅关系，此处不再介绍。

2.4 简单的输入和输出

在程序执行期间，把程序从外部接收数据的操作称为程序的输入，把程序向外部发送数据的操作称为程序的输出。C++中没有专门的输入/输出语句，所有的输入/输出操作都是通过输入/输出流来实现的。这里的输入指的是将通过键盘输入的数据赋给变量，而输出指的是将程序运行的结果发送给显示器并显示。在 C++中，输入操作是通过输入流来实现的，而输出操作是通过输出流来实现的。当使用 C++提供的输入/输出流时，必须在程序的开头增加一行：

```
#include <iostream>
```

即包含输入/输出流的头文件 iostream。

C++提供的输入/输出流有很强的输入/输出功能，极为灵活方便，但使用比较复杂。本节介绍最基本的数据输入/输出方法。有关输入/输出流的概念会在第 12 章中进行介绍。

2.4.1 数据输出 cout

在 C++中，当要输出表达式的值时，可使用 cout 来实现，其一般格式为：

```
cout<<表达式 1〔<<表达式 2<<表达式 3…<<表达式 n〕;
```

其中，“<<”称为插入运算符，它将紧跟其后的表达式的值输出到显示器当前光标的位置。例如，设有如下程序片段：

```
int a=2,b=3;
char c='x';
cout<<"a="<<a<<'\t'<<"b="<<b<<'\n';
cout<<"c="<<c<<'\n';
```

则执行后显示器上显示：

```
a=2          b=3
c=x
```

首先原样输出“a=”，输出变量 a 的值，输出横向制表符，即跳到下一个制表位；其次原样输出“b=”，输出变量 b 的值，输出一个换行符，即换行；然后原样输出“c=”，按字符形式输出字符变量 c 的值；最后输出一个换行符，表示后面的输出从下一行开始。

当用 cout 输出多个数据时，在默认情况下，是按每个数据的实际长度输出的，并且在每个输出的数据之间没有分隔符。例如，设有如下程序片段：

```
int i=10,j=20,k=30;
float x=3.14159,y=100;
cout<<i<<j<<k<<'\n';
cout<<x<<y<<'\n';
```

则执行后的输出结果为：

```
102030
3.14159100
```

显然，根据上述输出结果，无法分清每个变量的输出值。为了区分输出的数据项，在每个输出数据之间要输出分隔符。分隔符可以是空格、制表符或换行符等。例如，上面的输出语句可改为：

```
cout<<i<<'\t'<<j<<'\t'<<k<<'\n';
```

```
cout<<"x="<<x<<'\t'<<"y="<<y<<'\n';
```

则执行后输出结果为：

```
10          20          30
x=3.14159           y=100
```

2.4.2 数据输入 cin

在 C++程序执行期间，当要给变量输入数据时，可使用 cin 来实现，其一般格式为：

```
cin>>变量名 1〔>>变量名 2>>变量名 3 … >>变量名 n〕;
```

其中，“>>”称为提取运算符，表示将暂停程序的执行，等待用户通过键盘输入相应的数据。在提取运算符后只能跟一个变量名，“>>变量名”可以重复多次，即可给一个变量输入数据，也可给多个变量输入数据。例如，有下列程序片段，可通过键盘给变量输入数据：

```
int i,j;
float x,y;
char c;
cin>>i>>j;
cin>>x>>y;
cin>>c;
```

则在程序运行期间，当执行到 cin 时，等待用户通过键盘输入数据。若输入：

```
40   50<CR>
10.5    20.6<CR>
a<CR>
```

即将整数 40 赋给变量 i，将 50 赋给变量 j；将实数 10.5 赋给变量 x，将 20.6 赋给变量 y；将字符'a'赋给变量 c，其中<CR>代表“Enter”键。

需要输入数据的变量的数据类型可以是整型、实型和字符型。在输入数据时，各数据之间要用分隔符隔开，分隔符可以是一个或多个空格，也可以是“Enter”。按“Enter”键的作用是：一方面，告诉 cin 已输入一行数据，cin 开始从输入行中提取输入的数据，并依次将所提取的数据赋给 cin 中所列举的变量；另一方面，作为输入数据之间的分隔符。当 cin 遇到“Enter”键时，若仍有变量没有得到数据，则继续等待用户输入新一行数据。当 cin 遇到“Enter”键时，若输入行中的数据没有被提取完，则可给其他变量赋值。例如，在上例中，数据输入的方法可以是：

```
40   50   10.5   20.6   a<CR>
```

也可以是：

```
40<CR>
50<CR>
10.5<CR>
20.6<CR>
a<CR>
```

注意：通过键盘输入数据的个数、类型、顺序，必须与 cin 列举的变量一一对应。

2.4.3 简单的输入/输出格式控制

当用 cin 和 cout 进行数据的输入和输出时，无论处理的是什么类型的数据，都能够自动按照正确的默认格式对数据进行处理。但在实际应用中这还不够，经常需要设置特殊的输入/输出数据格式。

设置输入/输出数据格式有很多方法，这里只介绍最简单的格式控制方法。

C++中预定义了一些格式控制函数，可以直接嵌入 cin 和 cout 中实现对输入/输出数据格式的控制，如表 2.3 所示。

表 2.3　C++中预定义的格式控制函数

格式控制函数名	功　　能	适用输入/输出流
dec	设置为十进制数	输入/输出流
hex	设置为十六进制数	输入/输出流
oct	设置为八进制数	输入/输出流
ws	提取空白字符	输入流
endl	插入一个换行符（相当于'\n'）	输出流
ends	插入一个空白字符	输出流
setprecision(int)	设置定点数和浮点数的有效小数位数 *n*，或数据总位数（不包括小数点）	输出流
setw(int)	设置域宽	输出流

在使用这些格式控制函数时，必须在程序的开头包含 iomanip 文件，即增加一行：

```
#include <iomanip>
```

例如，有如下程序：

```
#include <iostream>
#include <iomanip>
using namespace std;
int main( )
{    int a=256,b=128;
     double c=12.3456;
     cout<<setw(8)<<a<<"b="<<b<<"c="<<c<<endl;
     cout<<hex<<a<<"b="<<dec<<b<<endl;
     cout<<setw(10)<<setprecision(4)<<c<<endl;
     return 0;
}
```

则程序运行后，输出：

```
     256b=128c=12.3456
100b=128
     12.35
```

说明：

(1) 当指明用一种进制输入/输出数据时，对其后的输入/输出均有效，直到指明以另一种进制输入/输出数据。

(2) 八进制数或十六进制数的输入/输出，只适用于整型数据，不适用于实型数据和字符型数据。

(3) 域宽设置函数 setw(int)仅对其后的一个输出项有效。

(4) 实数的输出位数设置函数 setprecision(int)对其后的所有输出项都有效，直到再一次设置。

本 章 小 结

1．关键字和标识符

关键字是 C++中保留自用的英文单词，不能另作他用。

标识符用于表示常量名、变量名、函数名、类型名等。由字母、下画线和数字组成，且必须以字母或下画线开头。

2．数据类型

数据类型分为基本数据类型和导出数据类型。基本数据类型是 C++中预定义的数据类型，包括整型（int）、实型（float）、字符型（char）和无值型（void）。导出数据类型是用户自定义的类型，包括数组、指针、结构体、共同体和类。

3．常量和变量

（1）常量。在程序执行中，其值不变的量称为常量。常量按其数据类型可以分为整型常量、实型常量、字符型常量和字符串常量 4 种。

（2）变量。在程序执行中，其值可改变的量称为变量。变量必须先定义后使用。变量定义的一般格式为：

〔存储类型〕<类型> <变量名 1> 〔,<变量名 2>,…,<变量名 *n*>〕;

在定义变量的同时，可给它赋初值。

4．运算符和表达式

（1）表达式：用运算符将常量、变量、函数等连接起来的式子称为表达式。

（2）运算符：分为算术运算符、赋值运算符、自增/自减运算符、关系运算符、逻辑运算符、逗号运算符、复合赋值运算符和数据类型长度运算符等。

（3）优先级：运算符的优先级从高到低依次为()→单目运算符（！、+、−、类型转换）→算术运算符（*、/、%→+、−）→关系运算符（>、>=、<、<=→==、!=）→逻辑运算符（&&→||）→赋值运算符（=）→逗号运算符（,）。

（4）结合性：有左结合性（从左到右运算）与右结合性（从右到左运算）两种。已介绍过的双目

运算符、逗号运算符具有左结合性，单目运算符与赋值运算符具有右结合性。

（5）数据类型转换：有自动类型转换与强制类型转换两种。

5．简单输入和输出

（1）数据输出 cout。在 C++程序中，输出表达式的值可用 cout 实现，其一般格式为：

cout<<表达式 1〔<<表达式 2 <<表达式 3 <<… <<表达式 *n*〕；

其中，“<<”称为插入运算符。

（2）数据输入 cin。在 C++程序执行期间，给变量输入数据可用 cin 实现，其一般格式为：

cin>>变量名 1〔>>变量名 2>>变量名 3>>…>>变量名 *n*〕；

其中，“>>”称为提取运算符。

（3）数据输入与输出必须包含输入/输出流的头文件 iostream，即在程序开始处写入文件包含命令：

```
#include <iostream >
```

6．本章重点、难点

重点：C++中的数据类型，各类运算符与表达式。

难点：运算符的优先级、结合性，不同数据类型的相互转换，表达式的综合运算。

习　题

2.1　简述标识符的定义。指出下列用户自己定义的标识符中哪些是合法的？哪些是非法的？如果是非法的，为什么？

xy	Book	3ab	x_2	switch	integer
page−1	_name	MyDesk	#NO	y.5	char

2.2　C++语言中有哪些数据类型？

2.3　什么是常量？什么是变量？

2.4　下列常量的表示在 C++中是否合法？若合法，指出常量的数据类型；若非法，指出原因。

−123	0321	.567	1.25e2.4	32L
'\t'	"Computer"	'x'	"x"	'\85'

2.5　字符常量与字符串常量有什么区别?

2.6　求出下列算术表达式的值。

（1）x+a%3*(int)(x+y)%2/4　　设 x=2.5，y=4.7，a=7

（2）(float)(a+b)/2−(int)x%(int)y　　设 a=2，b=3，x=3.5，y=2.5

（3）'a'+x%3+5/2−'\24'　　设 x=8

2.7　写出以下程序的运行结果。

```
#include <iostream>
using namespace std;
int main( )
{    int i,j,m,n;
      i=8;j=10;
```

```
    m=++i;n=j++;
    cout<<i<<'\t'<<j<<'\n';
    cout<<m<<'\t'<<n<<'\n';
    return 0;
}
```

2.8　将下列数学表达式写成 C++中的算术表达式。

（1）$\frac{a+b}{x-y}$　　（2）$\sqrt{p(p-a)(p-b)(p-c)}$

（3）$\frac{\sin x}{2m}$　　（4）$\frac{a+b}{2}h$

2.9　在 C++语言中，如何表示真和假？系统如何判断一个量的真和假?

2.10　设有变量定义：

```
int  a=3,b=2,c=1;
```

求出下列表达式的值。

（1）a>b　　（2）a<=b　　（3）a!=b　　（4）(a>b)==c

（5）a−b==c

2.11　设有变量定义：

```
int  a=3,b=1,x=2,y=0;
```

求出下列表达式的值。

（1）(a>b) && (x>y)　　（2）a>b && x>y

（3）(y||b) && (y||a)　　（4）y||b && y||a

（5）!a || a>b

2.12　设有变量定义：

```
int  w=3,x=10,z=7;
char  ch='D';
```

求出下列表达式的值。

（1）w++||z++　　（2）!w>z

（3）w && z　　（4）x>10 || z<9

（5）ch>='A' && ch<='Z'

2.13　设有语句：

```
int a=5,b=6,c;
c=!a&&b++;
```

执行以上语句后，求变量 a、b、c 的值。

2.14　设 a、b 的值分别为 6、7，指出分别运算下列表达式后 a、b、c、d 的值。

（1）c=d=a　　（2）b+=b

（3）c=b/=a　　（4）d=(c=a/b+15)

2.15　设 a、b、c 的值分别为 5、8、9，指出分别运算下列表达式后 x、y 的值。

（1）y=(a+b,b+c,c+a)　　（2）x=a,y=x+b

2.16　设计一个程序，通过键盘输入一个圆的半径，并求其周长和面积。

2.17　设计一个程序，通过键盘输入一个小写字母，并将它转换成大写字母。

2.18　通过键盘输入一个三位数 abc，从左到右用 a、b、c 表示各位数字，现要求依次输出从右到左的各位数字，即输出另一个三位数 cba。例如，输入 123，输出 321，试设计程序。（算法提示：a=n/100,b=(n−a*100)/10,c=(n−a*100)%10,m=c*100+b*10+a。）

实　　验

1．实验目的

（1）掌握用 VC++集成开发环境编辑源程序的方法。

（2）掌握在 VC++集成开发环境中编译、调试与运行程序的方法。

（3）理解数据类型、变量、运算符、表达式的概念。

（4）学会使用 cin 进行数据输入。

（5）学会使用算术表达式、关系表达式、赋值表达式完成数据处理工作。

（6）学会使用 cout 进行数据输出。

2．实验内容

（1）设计一个 C++程序，输出以下信息：

```
***************
    Hello!
***************
```

（2）设计一个 C++程序，输入三位职工的工资，并求工资总额。

实验数据：1500，2000，2500。

（3）设计一个程序，通过键盘输入一个矩形的长与宽，并求其周长和面积。

实验数据：50，40。

（4）设计一个程序，输入一个华氏温度值，要求输出其对应的摄氏温度值。温度转换公式为：c=(f−32)*5/9。

实验数据：33。

（5）通过键盘输入一个四位整数 n=abcd，从左到右用 a、b、c、d 表示各位数字，现要求依次输出从右到左的各位数字，即输出另一个四位整数 m=dcba，试设计程序。

实验数据：1234。

3．实验要求

（1）编写实验程序。

（2）在 VC++运行环境中输入源程序。

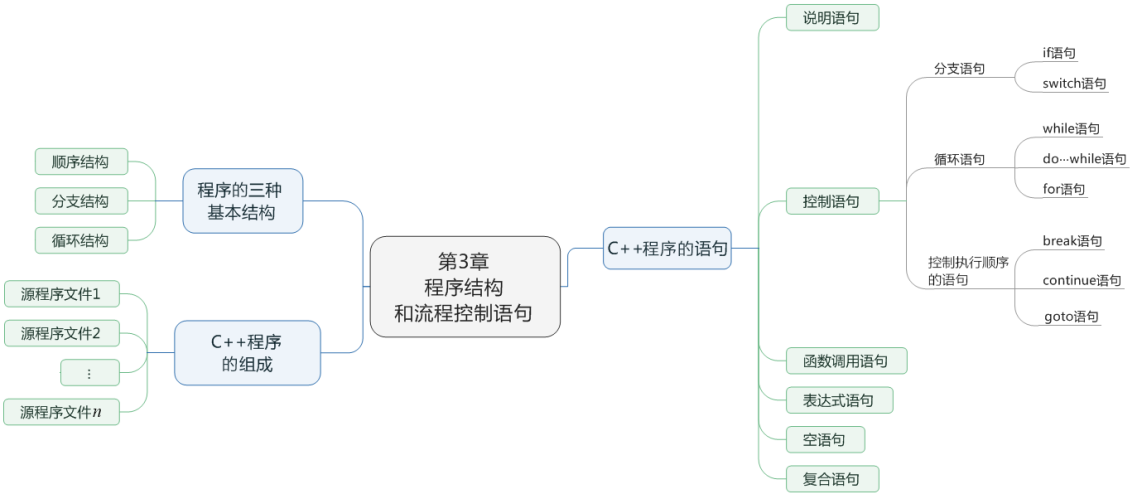
（3）编译运行源程序。

（4）输入测试数据进行程序测试。

（5）写出运行结果。

第 3 章

程序结构和流程控制语句



本章知识点导图

通过本章的学习，应掌握程序的三种基本结构，即顺序结构、分支结构和循环结构；掌握 C++中实现这三种基本结构的控制语句的格式、功能和执行过程；能使用这些控制语句编写具有顺序、分支和循环三种基本结构的程序。

3.1 程序的三种基本结构和语句

3.1.1 程序的三种基本结构

尽管C++是面向对象的程序设计语言，但组成C++程序的函数仍是由若干基本结构组合而成的。每种基本结构可以包含一条或多条语句。程序有三种基本结构，即顺序结构、分支结构和循环结构。

1．顺序结构

按程序中语句的顺序依次执行的结构称为顺序结构，它是最简单的一种基本结构，如图 3.1 所示。在图 3.1 中，按语句的顺序，先执行 S_1 操作，再执行 S_2 操作。图 3.1（a）为顺序结构流程图，图 3.1（b）为其 N-S 流程图。

2．分支结构

在两种可能的操作中按一定条件选取一个执行的结构称为分支结构，如图 3.2 所示。在图 3.2 中，当条件 B 成立（真）时，执行 S_1 操作，否则执行 S_2 操作。图 3.2（a）为分支结构流程图，图 3.2（b）为其 N-S 流程图。

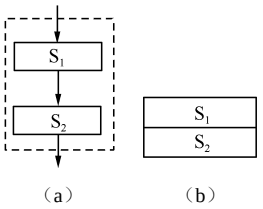


图 3.1　顺序结构

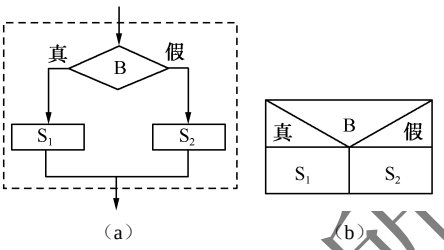


图 3.2　分支结构

由分支结构可以派生一种多分支结构，如图 3.3 所示。在图 3.3 中，依次判断条件 B_i（i=1，2，…，n）是否成立，当 B_i 成立时，就执行相应的 S_i 操作；当所有条件都不成立时，就执行 S_{n+1} 操作。

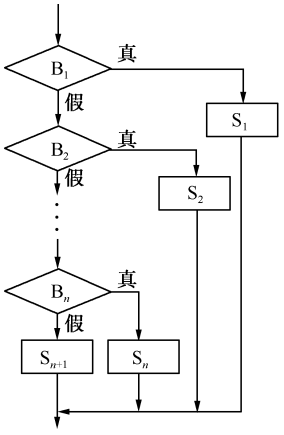


图 3.3　多分支结构

3．循环结构

循环结构有两种形式，即当型循环结构和直到型循环结构。

（1）当型循环结构。当某条件成立时，重复执行一个操作，直到条件不成立的结构称为当型循环结构，如图 3.4 所示。图 3.4（a）为当型循环结构流程图，图 3.4（b）为其 N-S 流程图。在图 3.4 中，当条件 B 成立（真）时，重复执行 S 操作，直到条件 B 不成立（假）时才停止执行 S 操作，转而执行其他操作。

（2）直到型循环结构。重复执行一个操作，直到某条件不成立的结构称为直到型循环结构，如图 3.5 所示。图 3.5（a）为直到型循环结构流程图，图 3.5（b）为其 N-S 流程图。在图 3.5 中，先执行 S 操作，再判断条件 B 是否成立，若条件 B 成立（真），则再次执行 S 操作，如此重复，直到条件 B 不成立（假）时停止执行 S 操作，转而执行其他操作。

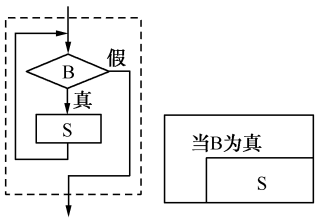


图 3.4　当型循环结构

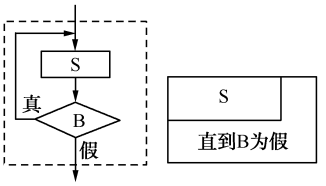


图 3.5　直到型循环结构

说明：在其他语言（如 Pascal）中，也有以条件 B 为真作为退出循环的条件的直到型循环结构。

三种基本结构都具有以下共同特征。

（1）单入口和单出口，即只有一个入口和一个出口。

（2）没有无用的部分，即结构中所有部分都有被执行的机会。

（3）不存在“死循环”（无终止的循环），即执行时间是有限的。

3.1.2　C++程序的组成

C++程序的组成如图 3.6 所示，即一个 C++程序可以由若干源程序文件组成，一个源程序文件可以由若干函数和编译预处理命令组成，一个函数由函数说明和函数体组成，函数体由变量定义和若干执行语句组成。语句是组成程序的基本单元。

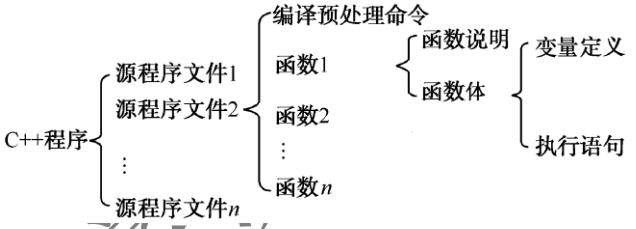


图 3.6　C++程序的组成

3.1.3　C++程序的语句

C++程序的语句可以分成以下六大类。

1．说明语句

在 C++中，把对数据类型的定义和描述、对变量和符号常量的定义性说明统称为说明语句。例如：

```
int   a,b,c;                                    //定义整型变量 a、b、c
```

说明语句在程序的执行过程中，并没有对数据进行操作，而是仅向编译系统提供一些说明性的信息。例如，定义变量语句 int　a,b,c;告诉编译系统为变量 a、b、c 各分配 4B 的存储空间用于存放变量的值。在 C++中，说明语句作为语句来对待，它可以出现在函数中允许出现语句的任何位置，也可以出现在函数定义外。

2．控制语句

完成一定控制功能的语句（有可能改变程序执行顺序的语句）称为控制语句。控制语句包括条件语句、开关语句、循环语句、转向语句、从函数返回语句等。例如：

```
if (x>y) z=x; else z=y;
```

该条件语句表示若 x>y，则 z=x，否则 z=y。根据 x、y 值的大小决定 z 的取值。

3．函数调用语句

在一次函数调用后加一个分号构成的语句称为函数调用语句。例如，例 1.3 中的 add(a,b);，其中，add(a,b)为求 a、b 两个变量的和的函数。

4．表达式语句

在一个表达式的后面加一个分号构成的语句称为表达式语句。例如，由一个赋值表达式加一个分号构成一条赋值表达式语句：

```
y=x*x+2*x-1.5+sin(x);
```

5．空语句

只有一个分号的语句称为空语句，即：

```
;
```

空语句不做任何操作，主要用于指明被转向的控制点或在特殊情况下作为循环语句的循环体。

6．复合语句

用花括号把一条或多条语句括起来后构成的语句称为复合语句（语句块）。C++把复合语句作为一条语句来处理，复合语句可以出现在允许出现一条语句的任何位置。复合语句中的左花括号表明复合语句的开始，右花括号表明复合语句的结束，右花括号后不再需要分号。例如：

```
{t=a;a=b;b=t;}
```

复合语句主要用在控制语句中。

程序的三种基本结构都是通过语句来实现的。由于顺序结构比较简单，已在第 2 章中有所介绍，所以下面只介绍能实现分支结构、循环结构的分支语句和循环语句。

3.2 分支语句

分支语句用于实现分支结构程序设计。分支语句分为两路分支结构和多路分支结构两种，两路分支结构可用 if 语句实现，多路分支结构可用嵌套的 if 语句和 switch 语句实现。

3.2.1 if 语句

if 语句即条件语句，它根据给定的条件决定执行两个分支程序段中的某一个分支程序段。

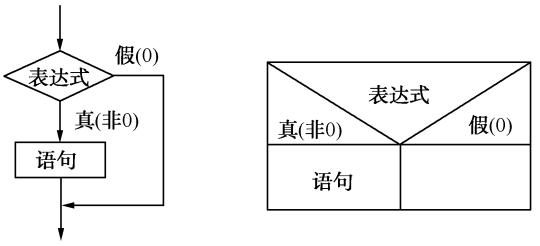
1．if 语句的三种形式

C++中的 if 语句有以下三种形式。

（1）单选 if 语句。单选 if 语句的格式为：

```
if (<表达式>)
    <语句>
```

单选 if 语句的执行流程为：当表达式的值为真（非 0）时，执行语句；否则不执行语句，如图 3.7 所示。图 3.7（a）为单选 if 语句流程图，图 3.7（b）为其 N-S 流程图。



（a）　　（b）

图 3.7　单选 if 语句执行流程图

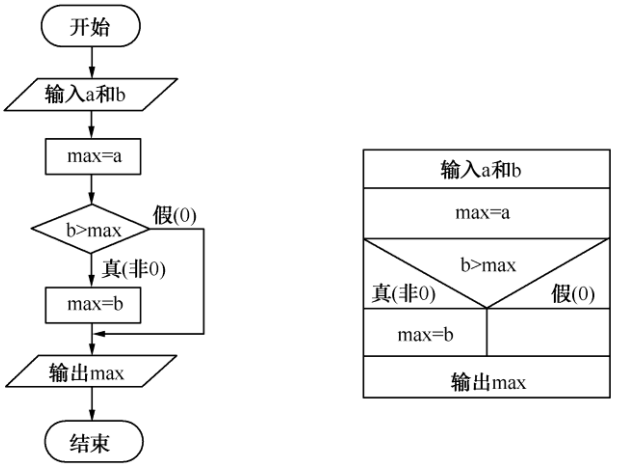
说明：

① 表达式一般为关系表达式或逻辑表达式，但也可为其他表达式，必须用一对圆括号“()”括起来。

② 语句可以是单条语句，也可以是多条语句（此时必须用花括号“{}”将多条语句括起来，构成一条复合语句）。

【例 3.1】　输入两个整数 a 和 b，并输出其中较大的一个数。

求两个数中较大值的流程图如图 3.8 所示。



（a）　　（b）

图 3.8　求两个数中较大值的流程图

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{
    int a,b,max;
    cout<<"Input a,b:";
    cin>>a>>b;
    max=a;
```

```
    if (b>max)
        max=b;
    cout<<"max="<<max<<endl;
    return 0;
}
```

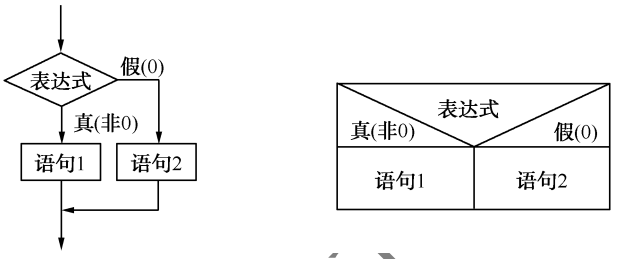
程序执行后显示：

```
Input a,b: 3        8
max=8
```

（2）双选 if 语句。双选 if 语句的格式为：

```
if (<表达式>)
  <语句 1>
else
  <语句 2>
```

双选 if 语句的执行流程为：当表达式的值为真（非 0）时，执行语句 1；否则执行语句 2，如图 3.9 所示。图 3.9（a）为双选 if 语句流程图，图 3.9（b）为其 N-S 流程图。

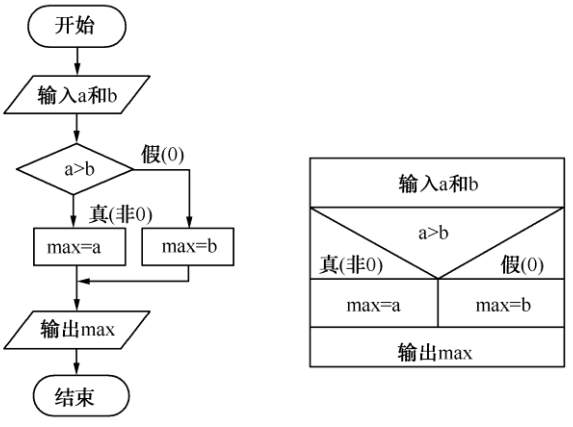


（a）　　（b）

图 3.9　双选条件语句执行流程图

【例 3.2】　输入两个整数 a 和 b，输出其中较大的一个数。

求两个数中较大值的流程图如图 3.10 所示。



（a）　　（b）

图 3.10　求两个数中较大值的流程图

程序如下：

```
#include <iostream>
```

```
using namespace std;
int main( )
{    int a,b,max;
     cout<<"Input a,b:";
     cin>>a>>b;
     if (a>b)
         max=a;
     else
        max=b;
     cout<<"max="<<max<<endl;
     return 0;
}
```

程序执行后显示：

```
Input a,b: 3        8
max=8
```

（3）多选 if 语句。多选 if 语句的格式为：

```
if (<表达式 1>)
  <语句 1>
else   if (<表达式 2>)
  <语句 2>

else   if (<表达式 3>)
  <语句 3>
   …
else   if (<表达式 n−1>)
  <语句 n−1>
else
  <语句 n>
```

【例 3.3】 设有下列分段函数：

$$y=\begin{cases} x+1 & x<0 \\ x^2-5 & 0\leqslant x<10 \\ x^3 & x\geqslant 10 \end{cases}$$


本题微课视频

编一程序，输入 x，并输出 y 的值。

分段函数流程图如图 3.11 所示。

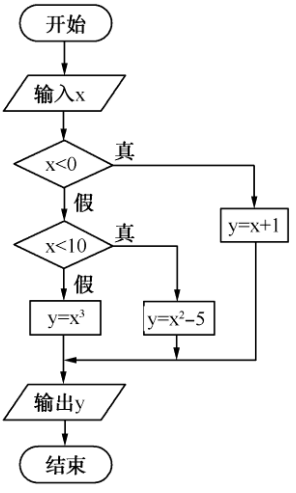


图 3.11　分段函数流程图 1

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    float x,y;
     cout<<"Input x:";
     cin>>x;
     if (x<0)
        y=x+1;
     else   if (x<10)
        y=x*x−5;
     else
        y=x*x*x;
     cout<<"y="<<y<<endl;
     return 0;
}
```

程序执行后显示：

```
Input x: 3
y=4
```

2．if 语句的嵌套

一个 if 语句中包含一个或多个 if 语句的情况称为 if 语句的嵌套，其一般格式为：

```
if (<表达式 1>)
   if (<表达式 2>)
      <语句 1>
   else
      <语句 2>
else
   if (<表达式 3>)
      <语句 3>
   else
      <语句 4>
```

【例 3.4】　设有下列分段函数：

$$y=\begin{cases} x+1 & x<0 \\ x^2-5 & 0\leqslant x<10 \\ x^3 & x\geqslant 10 \end{cases}$$

编一程序，输入 x，并输出 y 的值。

分段函数流程图如图 3.12 所示。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    float x,y;
     cout<<"Input x:";
     cin>>x;
     if (x>=0)
        if (x>=10)
           y=x*x*x;
        else
           y=x*x−5;
     else
        y=x+1;
     cout<<"y="<<y<<endl;
     return 0;
}
```

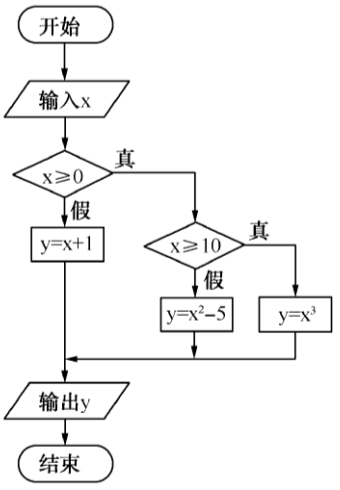


图 3.12　分段函数流程图 2

程序执行后显示：

```
Input x:−3
y=−2
```

在该程序中，内层的 if 语句嵌套在外层的 if 语句的 if 部分。

if 语句在嵌套使用时，应当注意 else 与 if 的配对关系。C++规定：else 总是与其前面最近的还没有配对的 if 进行配对。例如：

```
if (<表达式 1>)
   if (<表达式 2>)
       <语句 1>
   else
       <语句 2>
```

等价于：

```
if (<表达式 1>)
   {if (<表达式 2>)
       <语句 1>
    else
       <语句 2>
   }
```

如果要改变这种约定，则应用花括号构成复合语句。例如：

```
if (<表达式 1>)
   {if (<表达式 2>)
       <语句 1>
   }
else
    <语句 2>
```

此时，else 与第一个 if 配对。

【例 3.5】　求三个整数 a、b、c 中的最大者，a、b、c 由键盘输入。

求三个数中最大数的 N-S 流程图如图 3.13 所示。

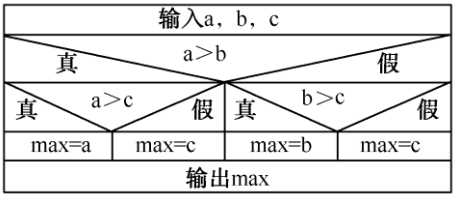


图 3.13　求三个数中最大数的 N-S 流程图

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int a,b,c,max;
     cout<<"Input a,b,c:";
     cin>>a>>b>>c;
     if (a>b)
       if (a>c)
          max=a;
       else
          max=c;
     else
       if (b>c)
          max=b;
       else
          max=c;
     cout<<"max="<<max<<endl;
     return 0;
}
```

程序执行后显示：

```
Input a,b,c:      1      4      5
max=5
```

3.2.2　条件运算符和条件表达式

C++中提供了一个条件运算符“? :”，它是三目运算符（三元运算符），即要求有三个操作对象。由条件运算符构成的条件表达式的一般格式为：

```
<表达式 1>?<表达式 2>:<表达式 3>
```

条件运算符的执行过程为：先计算表达式 1 的值，如果表达式 1 的值为非 0（真），则计算表达式 2 的值，条件表达式的值就是表达式 2 的值；如果表达式 1 的值为 0（假），则计算表达式 3 的值，条件表达式的值就是表达式 3 的值。其中，表达式 1 一般为关系表达式或逻辑表达式，但也可以为其他表达式。

当 if 语句的两个分支中都执行给同一个变量赋值的赋值语句时，可以用条件表达式来代替 if 语句，因此也可以用条件表达式构成分支结构程序。

【例 3.6】　输入两个整数 a 和 b，输出其中较大的一个数。

程序如下：

```
#include <iostream>
using namespace std;
```

```
int main( )
{    int a,b,max;
     cout<<"Input a,b:";
     cin>>a>>b;
     max=(a>b)?a:b;
     cout<< "max="<< max<<endl;
     return 0;
}
```

程序执行后显示：

```
Input a,b:9      4
max=9
```

条件运算符的优先级高于赋值运算符和逗号运算符的优先级，低于算术运算符、关系运算符和逻辑运算符的优先级。因此，赋值表达式 m=(a>b)?a:b 中的括号可以省略，即可写成 m=a>b?a:b。

3.2.3 switch 语句

1．switch 语句的格式及用法

if 语句允许程序在运行时按照表达式的值从两个可能的操作中选择一个来执行，但在程序要求进行多分支选择时，虽然用 if 语句可以实现，但需要多重嵌套，使用不方便，此时可以用 switch 语句来实现。

switch 语句即开关语句，它根据给定的条件决定执行多个分支程序段中的哪个分支程序段。

switch 语句的一般格式为：

```
switch (<表达式>)
{   case <常量表达式 1>:〔<语句 1>〕
    case <常量表达式 2>:〔<语句 2>〕
    …
    case <常量表达式 n-1>:〔<语句 n-1>〕
    〔default:<语句 n>〕
}
```

switch 语句的执行流程为：先计算表达式的值，然后将它与 case 后的常量表达式逐个进行比较，若与某一个常量表达式的值相等，就执行此 case 后面的语句；若都不相等，则执行 default 后面的语句，若没有 default，则不做任何操作就结束。

【例 3.7】　输入 0～6 的整数，将其转换成对应的星期。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int a;
     cout<<"Input an integer(0～6):";
     cin>>a;
     switch (a)
     {   case 0:cout<<"Sunday\n";
         case 1:cout<<"Monday\n";
         case 2:cout<<"Tuesday\n";
         case 3:cout<<"Wednesday\n";
         case 4:cout<<"Thursday\n";
         case 5:cout<<"Friday\n";
```

```
        case 6:cout<<"Saturday\n";
        default:cout<<"Input data error.\n";
    }
    return 0;
}
```

说明：

（1）switch 后面括号内的表达式只能是整型表达式、字符型表达式或枚举型变量。与其对应，case 后面的常量表达式也应为整型常量表达式、字符型常量表达式或枚举型数据。

（2）每个 case 后面的常量表达式的值必须互不相同。

（3）各个 case 和 default 的出现次序不影响执行结果。

（4）一个 case 后面可以包含多条语句，并且这些语句不必用花括号括起来，程序会自动顺序执行该 case 后面的所有语句，一个 case 后面也可以没有任何语句。

2．break 语句在 switch 语句中的作用

在执行 switch 语句的过程中，每当执行完一个 case 后面的语句后，程序就会不加判断地自动执行下一个 case 后面的语句。每个 case 后面的常量表达式只起语句标号的作用，是 switch 语句中执行各语句的入口。例如，在例 3.7 中，若在运行程序时输入 4，则执行结果为：

```
Thursday
Friday
Saturday
Input data error.
```

因此，应该在执行完一个 case 分支后，使程序跳出 switch 语句，即终止执行 switch 语句，这可以用 break 语句来实现。例 3.7 的程序应改写为：

```
#include <iostream>
using namespace std;
int main( )
{   int a;
    cout<<"Input an integer(0～6):";
    cin>>a;
    switch (a)
    {   case 0:cout<<" Sunday\n";break;
        case 1:cout<<" Monday\n";break;
        case 2:cout<<" Tuesday\n";break;
        case 3:cout<<" Wednesday\n";break;
        case 4:cout<<" Thursday\n";break;
        case 5:cout<<" Friday\n";break;
        case 6:cout<<" Saturday\n";break;
        default:cout<<"Input data error.\n";
    }
    return 0;
```

```
}
```

程序执行后显示：

```
Input an integer(0～6): 5
Friday
```

从 switch 语句的执行过程可知，任何 switch 语句均可用 if 语句来实现，并不是任何 switch 语句均可用 switch 语句来实现，这是由于 switch 语句限定了表达式的取值类型，而 if 语句中的条件表达式可取任意类型的值。

【例 3.8】 商店打折售货，购货金额越大，折扣越大，具体标准为（*m*：购货金额/元，*d*：折扣率）：


本题微课视频

m<250	*d*=0%
250≤*m*<500	*d*=5%
500≤*m*<1000	*d*=7.5%
1000≤*m*<2000	*d*=10%
m≥2000	*d*=15%

通过键盘输入购货金额，计算实付的金额。

分析：首先应找出购货金额与折扣率间对应关系的变化规律。从题意可知，当购货金额 *m* 每变化 250 元或 250 元的倍数时，折扣率就会变化。用 *c*=*m*/250 来表示折扣率的分档情况，如表 3.1 所示。

表 3.1　商店打折售货分档情况表

m/元	*c*=*m*/250	*d*
m<250	0	0%
250≤*m*<500	1	5%
500≤*m*<1000	2，3	7.5%
1000≤*m*<2000	4，5，6，7	10%
m≥2000	8	15%

根据购货金额确定好折扣率后，再计算实付金额。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int m,c;
     float d,f;
     cout<<"Input m:";
     cin>>m;
     if (m>=2000)
        c=8;
     else
        c=m/250;
     switch (c)
     {   case 0:d=0;break;
         case 1:d=5;break;
         case 2:
         case 3:d=7.5;break;
         case 4:
         case 5:
         case 6:
```

```
            case 7:d=10;break;
            case 8:d=15;break;
        }
        f=m*(1-d/100.0);
        cout<<"f="<<f<<endl;
        return 0;
    }
```

程序执行后显示：

```
Input m:500
f=462.5
```

3.3 循环语句

在程序设计中，经常会遇到在某一条件成立时，重复执行某些操作。例如，求：

$$S=1+2+3+4+\cdots+n$$

显然，在程序中不可能依次列出 1～n 个数，要完成以上的累加求和运算，可设两个整型变量 sum 和 i，sum 存放累加和，i 从 1 变化到 n，并按下列步骤进行操作。

（1）给 sum 赋值 0，i 赋值 1。

（2）令 sum=sum+i，i=i+1。

（3）若 i≤n，则重复执行步骤（2）。

（4）输出 sum 的值。

在以上步骤中，步骤（2）和步骤（3）是需要重复执行的操作。这种重复执行的操作可由程序中的循环结构来完成。

所谓循环结构，就是在给定条件成立的情况下，重复执行一个程序段；当给定条件不成立时，退出循环，再执行循环下面的程序。实现循环结构的语句称为循环语句。在 C++中，循环语句有 while 语句、do…while 语句和 for 语句。

3.3.1 while 语句

1．while 语句的格式

while 语句用来实现当型循环结构，其一般格式为：

```
while   (<表达式>)
  <语句>
```

说明：

（1）表达式称为循环条件表达式，一般为关系表达式或逻辑表达式，也可为其他表达式，必须用一对圆括号“()”括起来。

（2）语句称为循环体，可以是单条语句，也可以是多条语句（此时必须用花括号“{ }”将多条语句括起来，构成一个复合语句）。

2．while 语句的执行过程

while 语句的执行过程为：先计算表达式的值，当表达式的值为真（非 0）时，重复执行指定的语句；当表达式的值为假（0）时，结束循环，如图 3.14 所示。图 3.14（a）为 while 语句的流程图，图 3.14（b）为其 N-S 流程图。

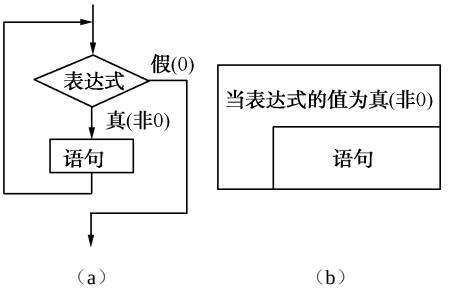


图 3.14　while 语句执行流程图

【例 3.9】　用 while 语句计算 $S=\sum_{i=1}^{n} i$，即求累加和 S=1+2+3+4+…+n。

用 while 语句求累加和的流程图如图 3.15 所示，程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int i,n,sum;
     cout<<"Input an integer:";
     cin>>n;
     sum=0;
     i=1;
     while (i<=n)
     {   sum=sum+i;
         i++;
     }
     cout<<"sum="<<sum<<endl;
     return 0;
}
```

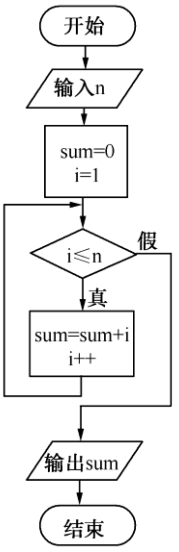


图 3.15　用 while 语句求累加和的流程图

程序执行后显示：

```
Input an integer:5
sum=15
```

说明：

（1）while 语句是先判断表达式 i<=n 是否成立，若条件成立，则将 sum 加 i 后赋给 sum，且 i 增加 1；若条件不成立，则不执行相应语句，退出循环。

（2）若表达式的值一开始就不成立，则语句一次也不执行。例如，当输入 n 为 0 时，i<=n 不成立，语句 sum=sum+i;和 i++;一次也不执行。

（3）在循环体中应有不断修改循环条件并最终使循环结束的语句，否则会形成死循环。例如，i++;语句，使 i 不断加 1，直到大于 n。

【例 3.10】　用 while 语句计算 $T=n!$，即求连乘积 $T=1\times2\times3\times4\times\cdots\times n$。

求连乘积流程图如图 3.16 所示，程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int i,n;
     float t;
     cout<<"Input an integer:";
     cin>>n;
     t=1.0;
     i=1;
     while (i<=n)
     {   t=t*i;
         i++;
     }
     cout<<"t="<<t<<endl;
     return 0;
}
```

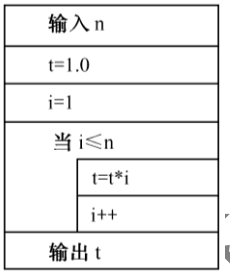


图 3.16　求连乘积流程图

程序执行后显示：

```
Input an integer:5
t=120
```

3.3.2 do…while 语句

1．do…while 语句的格式

do…while 语句用来实现直到型循环结构，其一般格式为：

```
do
   <语句>
while   (<表达式>);
```

说明：

(1) 表达式称为循环条件表达式，一般为关系表达式或逻辑表达式，也可为其他表达式，必须用一对圆括号“()”括起来。

(2) 语句称为循环体，可以是单条语句，也可以是多条语句（此时必须用花括号“{ }”将多条语句括起来，构成一个复合语句)。

(3) do...while 语句以分号结束。

2．do…while 语句的执行过程

do…while 语句的执行过程为：先执行语句，然后计算表达式的值，当表达式的值为真（非 0）时，重复执行指定的语句；当表达式的值为假（0）时，结束循环。do…while 执行流程图如图 3.17 所示，其中图 3.17（a）为 do…while 语句的流程图，图 3.17（b）为其 N-S 流程图。

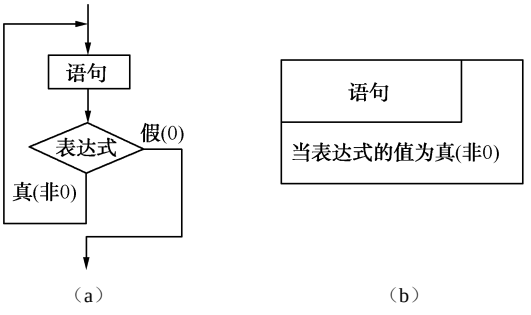


图 3.17　do…while 执行流程图

【例 3.11】　用 do…while 语句计算 $S=\sum_{i=1}^{n} i$，即求累加和 S=1+2+3+4+…+n。

用 do…while 语句求累加和的流程图如图 3.18 所示，程序如下：

```
#include <iostream>
using namespace std;
int main( )
```

```
{    int i,n,sum;
     cout<<"Input an integer:";
     cin>>n;
     sum=0;
     i=1;
     do
     {   sum=sum+i;
         i++;
     }
     while (i<=n);
     cout<<"sum="<<sum<<endl;
     return 0;
}
```

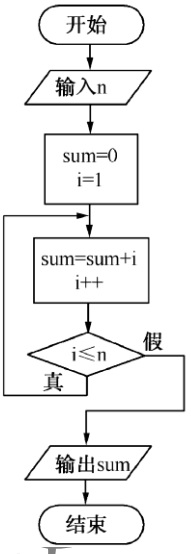


图 3.18　用 do…while 语句求累加和的流程图

说明：

（1）do...while 语句是先执行 sum=sum+i;和 i++;语句，后判断表达式 i<=n 是否成立。若条件成立，则继续执行循环体；若条件不成立，则不执行相应语句，退出循环。

（2）即使表达式的值一开始就不成立，语句仍要执行一次。例如，当输入 n 为 0 时，i<=n 不成立，语句 sum=sum+i;和 i++;也要执行一次。

（3）在循环体中，应有不断修改循环条件并最终使循环结束的语句，否则会形成死循环。

【例 3.12】　用 do…while 语句计算 $T=n!$，即求连乘积 $T=1\times2\times3\times4\times\cdots\times n$。

用 do…while 语句求连乘积的流程图如图 3.19 所示，程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int i,n;
     float t;
     cout<<"Input an integer:";
     cin>>n;
     t=1.0;
     i=1;
     do
     {   t=t*i;
         i++;
     }
     while (i<=n);
     cout<<"t="<<t<<endl;
     return 0;
}
```

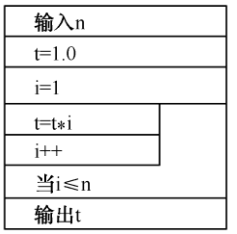


图 3.19　用 do…while 语句求连乘积的流程图

3.3.3　for 语句

1．for 语句的一般格式

for 语句是一种功能较强的循环语句，其一般格式为：

```
for   (<表达式 1>;<表达式 2>;<表达式 3>)
      <语句>
```

说明：

(1) 圆括号内的三个表达式之间用分号“;”隔开。

(2) 表达式 1 称为循环初始化表达式，通常为赋值表达式，在简单情况下为循环变量赋初值。

(3) 表达式 2 称为循环条件表达式，通常为关系表达式或逻辑表达式，在简单情况下为循环结束条件。

(4) 表达式 3 称为循环增量表达式，通常为赋值表达式，在简单情况下为循环变量增量。

(5) 语句部分为循环体，它可以是单条语句，也可以是多条语句（此时必须用花括号“{ }”将多条语句括起来，构成一个复合语句）。

2．for 语句的执行过程

for 语句的执行过程如下。

（1）计算表达式 1 的值。

（2）计算表达式 2 的值，若表达式 2 的值为真（非 0），则转到步骤（3）；若表达式 2 的值为假（0），则结束循环。

（3）执行循环体语句。

（4）计算表达式 3 的值，返回步骤（2）继续执行。

for 语句执行流程图及其执行过程如图 3.20 和图 3.21 所示。

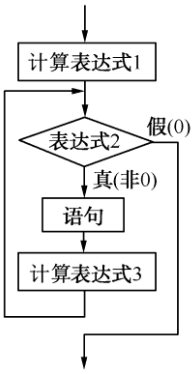


图 3.20　for 语句执行流程图

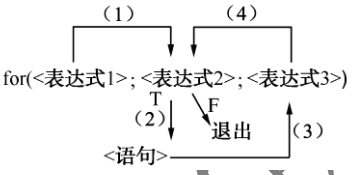


图 3.21　for 语句执行过程

for 语句可以等效于下列 while 语句：

```
<表达式 1>;
while   (<表达式 2>)
   {<语句>
    <表达式 3>;
   }
```

【例 3.13】　用 for 语句计算 $S=\sum_{i=1}^{n} i$，即求累加和 S=1+2+3+4+…+n。

用 for 语句求累加和流程图如图 3.22 所示，程序如下：

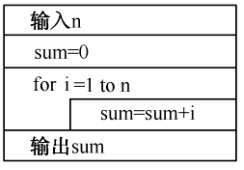


图 3.22　用 for 语句求累加和流程图

```
#include <iostream>
using namespace std;
int main( )
{     int i,n,sum;
      cout<<"Input an integer:";
      cin>>n;
      sum=0;
      for (i=1;i<=n;i++)
         sum=sum+i;
      cout<<"sum="<<sum<<endl;
      return 0;
}
```

在例 3.13 中，表达式 1，即 i=1，完成对循环变量 i 的初始化赋值工作，使 i 的初值为 1。表达式 2，即 i<=n，判断循环变量 i 的值是否小于或等于 n，若不成立则结束循环；若成立则执行 sum=sum+i; 语句，再执行表达式 3。表达式 3，即 i++，使循环变量 i 加 1，然后转表达式 2 继续判断 i<=n 是否成立。

说明：

(1) for 语句中的 3 个表达式都可以省略，但其中的两个分号不可以省略。

(2) 若省略表达式 1，则应在 for 语句之前给循环变量赋初值。例如：

```
i=1;
```

```
for (;i<=n;i++)
  sum=sum+i;
```

（3）若省略表达式 2，则不判断循环条件，循环无终止地进行下去，形成死循环，即认为表达式 2 始终为真，因此表达式 2 通常不能省略。

（4）若省略表达式 3，则在循环体中应有不断修改循环条件的语句。例如：

```
for    (i=1;i<=n;)
{    sum=sum+i;
     i++;
}
```

（5）若省略表达式 1 和表达式 3，则只有表达式 2，即只给出循环条件。例如：

```
i=1;
for    (;i<=n;)
   {    sum=sum+i;
        i++;
   }
```

此时，for 语句和 while 语句完全相同，上述语句相当于：

```
i=1;
while    (i<=n)
{    sum=sum+i;
     i++;
}
```

（6）表达式 1 和表达式 3 可以是一个简单的表达式，也可以是其他表达式，当然可以是逗号表达式，即用逗号“,”隔开的多个简单表达式，它们的运算顺序是按从左到右的顺序进行的。例如：

```
for (i=0,j=0;i+j<40;i++,j+=10)
   cout<<i<<"\t"<<j<<endl;
```

【例 3.14】 用 for 语句计算 $T=n!$，即求连乘积 $T=1\times2\times3\times4\times\cdots\times n$。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{     int i,n;
      float t;
      cout<<"Input an integer:";
      cin>>n;
      t=1.0;
      for (i=1;i<=n;i++)
         t=t*i;
```

```
    cout<<"t="<<t<<endl;
    return 0;
}
```

【例 3.15】　计算 $S=\sum_{k=1}^{20}\frac{1}{k(k+1)}$，即求 $S=\frac{1}{1\times2}+\frac{1}{2\times3}+\mathrm{L}+\frac{1}{19\times20}+\frac{1}{20\times21}$。

分析：求解该题仍采用求累加和的思想，即用循环语句将级数中各项值：t=1.0/(i*(i+1))依次加入累加和 sum 中。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{   int i;
    float t,sum;
    sum=0;
    for (i=1;i<=20;i++)
    {   t=1.0/(i*(i+1));
        sum=sum+t;
    }
    cout<<"S="<<sum<<endl;
    return 0;
}
```

程序执行后输出：

```
S=0.952381
```

3.3.4　三种循环语句的比较

对于任何一个循环结构，在一般情况下，均可用三种循环语句中的任何一种来实现。但对不同的循环结构，使用不同的循环语句，不仅可以优化程序的结构，还可以精简程序、提高程序执行效率。

（1）while 语句和 for 语句都是先判断循环条件表达式的值，后执行循环体语句；而 do…while 语句则是先执行循环体语句，后判断循环条件表达式的值。

（2）while 语句、do…while 语句和 for 语句都是在循环条件表达式为真时，重复执行循环体语句，在循环条件表达式为假时，结束循环。

（3）当第一次执行 while 语句或 for 语句时，若循环条件表达式为假，则一次也不执行循环体语句；而当第一次执行 do…while 语句时，即使循环条件表达式为假，也要执行一次循环体语句。也就是说，do…while 语句至少执行一次循环体，而 while 语句和 for 语句有可能一次也不执行循环体。

（4）while 语句和 for 语句用来构成当型循环结构；do…while 语句用来构成直到型循环结构。

（5）在循环体语句至少执行一次的情况下，三种循环语句构成的循环结构可以相互转换。

实际上，用得最多的是 for 语句，其次是 while 语句，do…while 语句相对于前两种语句用得较少。

3.3.5 循环语句的嵌套

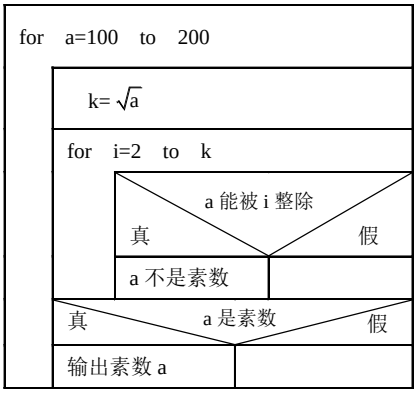


图 3.23 求 100～200 中的所有素数的流程

在程序设计中，如果一个循环语句的循环体包含循环语句，则称其为循环语句的嵌套。循环语句的嵌套又称多重循环。当一个循环语句的循环体中只包含一层循环语句时，称双重循环；若第二层循环语句的循环体中还包含一层循环语句，则称三重循环，依次类推。三种循环语句可以互相嵌套。

【例 3.16】 求 100～200 中的所有素数，输出时一行打印 5 个素数。

分析：判断一个数 a 是否为素数，只需将它除以 2～$\sqrt{a}$（取整）即可，如果都不能整除，那么 a 就是素数。求 100～200 中的所有素数的流程图如图 3.23 所示，程序如下：


本题微课视频

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
int main( )
{   int a,k,i,n;
    n=0;
    for (a=100;a<=200;a++)                    //a 从 100 循环到 200
    {  k=sqrt(a);
       for (i=2;i<=k;i++)                     //判断 a 是否为素数
          if (a%i==0)                         //a 不是素数
             break;
       if (i>k)                               //a 是素数
       {  cout<<setw(12)<<a;                  //输出素数
          n=n+1;                              //统计素数的个数
          if (n%5==0)                         //控制每行输出 5 个素数
             cout<<endl;
       }
    }
    cout<<endl;
    return 0;
}
```

程序执行后输出：

```
101         103         107         109         113
127         131         137         139         149
151         157         163         167         173
179         181         191         193         197
199
```

在程序中，调用求解平方根的 sqrt( )库函数需要包含数学函数的头文件 cmath，因此，在程序中使用文件包含命令# include <cmath>。

在例 3.16 中，用内循环 for (i=2;i<=k;i++)判断 a 是否为素数。若 a 不能被 i（i=2～k）整除，则说明 a 是素数，内循环正常结束，此时 i=k+1，输出素数 a；若 a 能被某一个 i（i=2～k）整除，则说明 a 不是素数，使用 break 语句终止内循环的执行，此时 i<=k，不输出 a。用外循环 for (a=100;a<=200;a++)

{…}来判断 100～200 中哪些数是素数。因此程序使用了循环语句的嵌套。

3.4 控制执行顺序的语句

前面已介绍的 C++语句都是根据其在程序中的先后次序，从主函数开始依次执行各语句的。这里要介绍另一类语句，当执行该类语句时，会改变程序的执行顺序，即不依次执行紧跟其后的语句，而是跳到另一条语句处接着执行。从表面上看，循环语句或条件语句也改变了程序的执行顺序，但由于整个循环只是一个语句（条件语句也一样），所以它们仍然是顺序执行的。

3.4.1 break 语句

前面已经介绍过使用 break 语句终止 switch 语句的执行，实际上，break 语句也可以用于 while、do…while、for 循环语句中，使它们终止执行，即用于从循环体内跳出，从而提前结束循环。

break 语句的一般格式为：

```
break;
```

若 break 语句出现在嵌套的循环语句中，那么哪一层循环有 break 语句，就跳出该层的循环。break 语句不能用于循环语句和 switch 语句之外的语句中。

3.4.2 continue 语句

continue 语句能用于 while、do…while、for 循环语句中，当执行 continue 语句时，将控制转移到 while 和 do…while 语句的循环判断条件处，或 for 语句的增量表达式处，即结束本次循环，重新开始下一次循环。

continue 语句的一般格式为：

```
continue;
```

【例 3.17】 输入 10 个整数，统计其中正数的个数及正数的和。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int a,i,k=0,s=0;                    //k 存放正数的个数，s 存放正数的和
     cout<<"Input 10 integers:";
     for (i=1;i<=10;i++)
     {   cin>>a;                         //输入整数到变量 a 中
         if (a<=0)
            continue;                    //若 a 为负数或零，则转到 for 语句中的表达式 3 处执行
         k++;                            //若 a 为正数则 k 加 1，并将 a 累加到 s 中
         s+=a;
     }
     cout<<"k="<<k<<'\t'<<"s="<<s<<'\n';
     return 0;
}
```

程序执行后显示：

```
Input 10 integers:1 2 3 4 5 6 7 8 9 0
k=9      s=45
```

程序在执行时，用循环语句依次先输入 10 个整数，每次输入后均判断该数是否为负数或零，若为负数或零，则执行 continue 语句，转到 for 语句中的表达式 3 处执行 i++，开始新一轮循环。若为正数，则执行 k++与 s+=a，将 k 加 1，将 a 累加到 s 中。最后输出正数的个数及正数的和。

若 continue 语句出现在嵌套的循环语句中，则 continue 语句只对当前循环起作用。continue 语句不能用于循环语句之外的语句中。

continue 语句和 break 语句的区别是：continue 语句只结束本次循环，而不结束整个循环的执行；而 break 语句则是结束整个循环，不管循环条件是否成立。例如，设有如下程序：

```
#include <iostream>
using namespace std;
int main( )
{     int i;
      for (i=100;i<=200;i++)
      {   if   (i%3==0)
                continue;
          cout<<i<<'\t';
      }
      return 0;
}
```

运行该程序后，输出为：

```
100   101   103   104 …   199   200
```

该程序输出 100～200 中不能被 3 整除的数。

若把程序中的 continue 语句改为 break 语句，即将程序改为：

```
#include <iostream>
using namespace std;
int main( )
{     int i;
      for (i=100;i<=200;i++)
      {   if   (i%3==0)
                break;
          cout<<i<<"   ";
      }
      return 0;
}
```

则运行该程序后，先输出不能被 3 整除的 100、101，当循环变量 i=102 时，由于 i 能被 3 整除，所以执行 break 语句终止循环。因此，程序执行后输出：

```
100    101
```

3.4.3 语句标号和 goto 语句

1．语句标号

前面介绍的语句都是无标号语句，即在语句前面没有标号的语句。C++还允许使用带标号的语句，即在语句前带一个标号。例如：

```
label_10 : a=3
```

C++中带标号语句的一般格式为：

```
<语句标号>:<语句>
```

语句标号指示语句在程序中的位置，常常作为转移语句（goto 语句）的转移目标。

语句标号用标识符来表示，它的命名规则与标识符的命名规则相同，即由字母、数字和下画线组成，且第一个字符必须为字母或下画线，不能用整数作为标号。在 C++中，语句标号可以直接使用，不必先定义后使用，这一点与变量不同。

2．goto 语句

goto 语句将改变程序的流程，无条件地转移到指定语句标号的语句处执行。

goto 语句的一般格式为：

```
goto <语句标号>;
```

其中语句标号是用户指定的，出现在本函数的一条语句的前面。

goto 语句的使用仅局限于函数内，不允许在一个函数中使用 goto 语句转移到其他函数中。goto 语句可以从条件语句或循环语句里面转移到条件语句或循环语句外面，但不允许从条件语句或循环语句外面转移到条件语句或循环语句里面。从结构化程序设计的角度出发，在进行程序设计时，应尽量避免使用 goto 语句。goto 语句可与 if 语句一起构成循环结构。

3.4.4 exit()函数和 abort()函数

exit()函数和 abort()函数都是 C++的库函数，其功能都是终止程序的执行，并将控制返回给操作系统。通常，exit()函数用于正常终止程序的执行，而 abort()函数用于异常终止程序的执行。显然，当执行两个函数中的任意一个时，都将改变程序的执行顺序。当使用这两个函数中的任意一个时，都应包含头文件 cstdlib。

1．exit()函数

exit()函数的格式为：

```
exit(<表达式>);
```

其中，表达式的值只能是整型数。通常把表达式的值作为终止程序执行的原因。在执行该函数时，将无条件地终止程序的执行而不管该函数处于程序的什么位置，并将控制返回给操作系统。通常表达式的取值是一个常数：用 0 表示正常退出，用其他整数表示异常退出。

当执行 exit()函数时，系统要做终止程序执行前的收尾工作，如关闭该程序打开的文件、释放变量占用的存储空间（不包括动态分配的存储空间）等。

2．abort()函数

abort()函数的格式为：

```
abort();
```

在调用该函数时，括号内不能有任何参数。在执行该函数时，系统不做结束程序前的收尾工作，直接终止程序的执行。因此通常使用 exit()函数来终止程序的执行。

除了上面介绍的语句和库函数可以改变程序的执行顺序，return 语句也可以改变程序的执行顺序。return 语句的执行过程会在第 5 章进行介绍。

3.5 程序设计举例

前面介绍了程序设计的三种基本结构，即顺序结构、分支结构和循环结构。其中分支结构程序主要是用分支语句（if、switch）实现的，而循环结构程序则主要是用循环语句（while、do…while、for）语句实现的。本节对分支语句与循环语句的应用进行了举例。

3.5.1 分支语句应用举例

分支语句用于实现分支结构程序设计。能够用分支结构程序解决的常见问题有：比较数的大小、找出若干数中的最大值与最小值、分段函数、解一元二次方程、判断闰年、多分支问题。分支语句有 if 语句和 switch 语句。

1．if 语句

if 语句又有单选 if 语句、双选 if 语句与多选 if 语句，格式分别为：

```
单选 if 语句
if (<表达式>)
    <语句>
```

```
双选 if 语句
if (<表达式>)
    <语句 1>
else
    <语句 2>
```

```
多选 if 语句
if (<表达式 1>)
    <语句 1>
else if <表达式 2>
    <语句 2>
    …
else
    <语句 n>
```

if 语句可以嵌套使用，但应当注意 else 与 if 的配对关系。C++规定：else 总是与其前面最近的还没有配对的 if 进行配对。

【例 3.18】 编写程序，求一元二次方程 $ax^2+bx+c=0$ 的解。

分析：一元二次方程的解有以下几种情况。

（1）当 $a=0$ 时，若 $b=0$，则方程无解；若 $b\neq0$，则方程有单根。

（2）当 $a\neq0$ 时，若 $d=b^2-4ac=0$，则方程有两个相等的实根；若 $b^2-4ac>0$，则方程有两个不等的实根；若 $b^2-4ac<0$，则方程有两个共轭复根。

方程根 x_1、x_2 的求解公式如图 3.24 所示。

$$
a\begin{cases}
=0,\ b\begin{cases}=0,\ \text{方程无解}\\ \neq0,\ x_1=-c/b\end{cases}\\
\neq0,\ d=b^2-4ac\begin{cases}=0,\ x_1=x_2=t_1\\ >0,\ x_1=t_1+t_2,\ x_2=t_1-t_2,\\ <0,\ x_1=t_1+t_2\mathrm{i},\ x_2=t_1-t_2\mathrm{i},\end{cases}\\
t_1=-b/(2a)\\
t_2=\sqrt{|d|}/(2a)
\end{cases}
$$

图 3.24 方程根 x_1、x_2 的求解公式

根据求解公式，可将程序的结构框架按如下方式构建：判断 a 是否为 0，可以用双选 if 语句；判断 b 是否为 0，可以用嵌套双选 if 语句；判断 d，可用嵌套三选 if 语句。因此，程序的结构框架如下：

```
if (a==0)                          //双选 if 语句
    if (b==0)                      //嵌套双选 if 语句
      输出“方程无解”
```

```
    else
      x1=-c/b;
else
    if(d==0)                                    //嵌套三选 if 语句
     x1=x2=t1;
    else if(d>0)
      x1=t1+t2;x2=t1-t2;
    else
      x1=t1+t2i;x2=t1-t2i;
```

由上述程序结构框架可知，程序中应定义实型变量 a、b、c、d、t1、t2、x1、x2，用 cin 输入方程系数 a、b、c，计算并输出方程根 x1、x2。按程序的结构框架很容易写出下列程序：

```
#include <iostream>
#include <cmath>
using namespace std;
int main( )
{   float a,b,c,d,t1,t2,x1,x2;
    cout<<"Input a,b,c:";
    cin>>a>>b>>c;
    if (a==0.0)                                 //a=0，方程退化为一元一次方程
      if(b==0.0)                                //b=0，方程无解
        cout<<"Input data error!"<<endl;
      else                                      //b!=0，方程有单根
      {  x1=-c/b;
        cout<<"Single root:"<<x1<<endl;
      }
    else                                        //a!=0，方程为一元二次方程
    {     d=b*b-4*a*c;                          //计算 d=b2-4ac
          t1=-b/(2*a);
          t2=sqrt(fabs(d))/(2*a);
          if (d==0.0)                           //d=0，方程有两个相等的实根
          {  x1=t1;
             cout<<"Two equal real roots:"<<x1<<endl;
          }
          else if (d>0.0)                       //d>0，方程有两个不等的实根
          {  x1=t1+t2;
             x2=t1-t2;
             cout<<"Two distinct real roots:"<<x1<<','<<x2<<endl;
          }
          else                                   //d<0，方程有两个共轭复根
          {  cout<<"Complex roots:";
                cout<<t1<<'+'<<t2<<'i'<<','<<t1<<'－'<<t2<<'i'<<endl;
          }
    }
    return 0;
}
```

其中，sqrt()为求实数平方根的函数，fabs()为求实数绝对值的函数。当使用它们时，需要在程序中使用文件包含命令 # include <cmath>。

程序运行后显示：

```
Input a,b,c:2   -5   3
```

输出：

```
Two distinct real roots:1.5,1
```

由例 3.18 可知，C++程序设计的一般步骤如下。

（1）进行数学分析，建立求解数学模型。

（2）根据数学模型确定程序框架及所用语句。

（3）根据数学模型确定需要定义的变量及其数据类型。

（4）给变量输入数据。

（5）编写计算程序，执行程序得到运算结果。

（6）输出运算结果。

2．switch 语句

switch 语句的格式为：

```
switch (<表达式>)
  {case <常量表达式 1>:〔<语句 1>〕
   …
   case <常量表达式 n-1>:〔<语句 n-1>〕
   〔default:<语句 n>〕
  }
```

注意：在执行 switch 语句的过程中，一般用 break 语句结束 switch 语句的执行。

【例 3.19】 输入某一年的年份和月份，计算该月的天数。

分析：

（1）一年中的大月（1 月、3 月、5 月、7 月、8 月、10 月、12 月），每月的天数为 31 天。

（2）一年中的小月（4 月、6 月、9 月、11 月），每月的天数为 30 天。

（3）对于 2 月，则要判断该年是平年还是闰年，平年的 2 月为 28 天，闰年的 2 月为 29 天。

某年符合下面两个条件之一就是闰年：①年份能被 400 整除；②年份能被 4 整除，但不能被 100 整除。

已知年（year）、月（month）求天数（day）的求解公式如图 3.25 所示。

$$month=\begin{cases}1、3、5、7、8、10、12, & day=31\\ 4、6、9、11, & day=30\\ 2, & year=\begin{cases}\text{闰年}, day=29\\ \text{平年}, day=28\end{cases}\end{cases}$$

图 3.25　已知年、月求天数的求解公式

根据求解公式，可将程序的结构框架按如下方式构建：用 switch 语句对月份 month 进行多选判断，当 month=2 时，用双选 if 语句判断是否为闰年。程序的结构框架如下：

```
switch (month)
{  case  1、3、5、7、8、10、12 ：  day=31
   case  4、6、9、11 ：    day=30
   case  2 ：
          if (year==闰年)
              day=29
          else
              day=28
```

```
}
```

由上述程序的结构框架可知，程序中应定义整型变量 year、month、day，用 cin 输入 year、month，然后计算并输出 day。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int year,month,day;
     cout<<"Input year and month:";
     cin>>year>>month;
     switch (month)
     {   case 1:
         case 3:
         case 5:
         case 7:
         case 8:
         case 10:
         case 12:day=31;break;
         case 4:
         case 6:
         case 9:
         case 11:day=30;break;
         case 2:if (year%400==0 || year%4==0 && year%100!=0)
                    day=29;
                else
                    day=28;
                break;
         default:cout<<"Input data error!"<<endl;day=0;break;
     }
     if (day!=0)
     cout<<"The day of "<<year<<','<<month<<" is "<<day<<endl;
     return 0;
}
```

程序运行后显示：

```
Input year and month:2002,1
```

输出：

```
The day of 2002,1 is 31
```

3.5.2 循环语句应用举例

循环语句用于实现循环结构程序设计。循环语句有 while 语句、do…while 语句和 for 语句。三种语句的格式如下：

while 语句	do…while 语句	for 语句
while(<表达式>) <语句>;	do <语句> while (<表达式>);	for (<表达式 1>,<表达式 2>,<表达式 3>) <语句>;

应注意三种循环语句的特点、区别及由它们构成的嵌套循环结构。

能够用循环结构程序解决的常见问题有：累加和、连乘积、求一批数的和及最大值与最小值、求

数列的前 n 项、判断素数、求两个整数的最大公约数和最小公倍数、用迭代法求平方根、用穷举法求不定方程组的整数解、打印图形等。

【例 3.20】 用公式 $e=1+\sum_{n=1}^{10}\frac{1}{n!}$，即 $e=1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\cdots+\frac{1}{n!}$，求自然对数底 e 的近似值（$n$=5）。

分别用 while 语句、do…while 语句和 for 语句三种语句实现。

本例应用求累加和的方法求 e 的近似值，而求累加和的方法可归纳如下：

定义变量	算法	初值
t：存放第 i 项分母值	t=t*i;	t=1
p：存放第 i 项值	p=1/t;	p=1
s：存放累加和	s=s+p;	s=1
i：循环变量	i=i+1;	i=1

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{int i=1;
 float s=1.0,t=1,p=1;
 while (i<=5)
      {t=t*i;
       p=1/t;
       s=s+p;
       i++;
      }
cout<<"e="<<s<<endl;
return 0;
}
```

```
#include <iostream>
using namespace std;
int main( )
{int i=1;
 float s=1.0,t=1, p=1;
 do
      {t=t*i;
       p=1/t;
       s=s+p;
       i++;
      } while (i<=5);
cout<<"e="<<s<<endl;
return 0;
}
```

```
#include <iostream>
using namespace std;
int main( )
{int i;
 float s=1.0,t=1,p=1;
 for (i=1; i<=5;i++)
      {t=t*i;
       p=1/t;
       s=s+p;
      }
cout<<"e="<<s<<endl;
return 0;
}
```

程序执行输出：

```
e=2.71667
```

在例 3.20 中，程序运行输出结果 e=2.71667 与自然对数的底 e=2.71828 误差较大。为了使误差控制在规定的范围 δ 内，应要求数列中最后一项的值小于 δ，即 $\frac{1}{n!}<\delta$。

若取 δ=0.00001，则循环的结束条件为 p<δ=0.00001，而循环条件为 p>=0.00001。将上述程序改为：

```
#include <iostream>
using namespace std;
int main( )
{int i=1;
 float s=1.0,t=1,p=1;
 while (p>=0.00001)
      {t=t*i;
       p=1/t;
       s=s+p;
       i++;
      }
cout<<"e="<<s<<endl;
return 0;
}
```

```
#include <iostream>
using namespace std;
int main( )
{int i=1;
 float s=1.0,t=1, p=1;
 do
  {t=t*i;
   p=1/t;
   s=s+p;
   i++;
  }while (p>0.00001);
cout<<"e="<<s<<endl;
return 0;
}
```

```
#include <iostream>
using namespace std;
int main( )
{int i;
 float s=1.0, t=1, p=1;
 for (i=1; p>=0.00001;i++)
      {t=t*i;
       p=1/t;
       s=s+p;
      }
cout<<"e="<<s<<endl;
return 0;
}
```

程序执行输出：

```
e=2.71828
```

从例 3.20 可以看出，累加和问题用三种语句都可以实现。当读者学会这三种语句后，用哪种语句都可以编写循环结构程序。

【例 3.21】 $s=\frac{1}{0!}-\frac{1}{1!}+\frac{1}{2!}-\frac{1}{3!}+\frac{1}{4!}+\cdots+\frac{(-1)^n}{n!}$，用累加和的方法求 s 的值，直到最后一项 $\left|\frac{(-1)^n}{n!}\right|<0.00001$。

分析：本例与例 3.20 的唯一区别是，第 i 项值由 p=$\frac{1}{\mathrm{i}!}$ 改成 p=$\frac{(-1)^{\mathrm{i}}}{\mathrm{i}!}$。因此，需要定义一个能表示分子值的变量 t1，其初值为 1，循环算法为 t1=(−1)* t1；第 i 项值 p= t1/t。

循环结束条件是 $\left|\frac{(-1)^n}{n!}\right|=\frac{1}{n!}<0.00001$，即循环条件是 1/t>=0.00001。用 do…while 语句实现的程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int i, t1=1;
     float s=1.0, t=1,p;
     i=1;
     do
     {    t=t*i;
          t1=(−1)*t1;
          p=t1/t;
          s=s+p;
          i++;
     } while (1/t>0.00001);
     cout<<"s="<<s<<endl;
     return 0;
}
```

程序执行后输出：

```
s=0.367879
```

【例 3.22】 裴波那契数列的前几个数为 1，1，2，3，5，8，…其规律为：


本题微课视频

$f_1=1$　　（n=1）
$f_2=1$　　（n=2）
$f_n=f_{n-1}+f_{n-2}$　　（$n\geqslant 3$）

编写程序求此数列的前 40 个数。

分析：可设两个变量 f1 和 f2，它们的初值为 f1=1，即数列的第 1 项；f2=1，即数列的第 2 项，用一个循环结构来求数列的前 40 项，每次处理两项，共循环 20 次。进入循环后，先输出 f1、f2，然后令 f1=f1+f2，即可求得第 3 项，再令 f2=f2+f1，注意此时的 f1 已经是第 3 项了，即可求得第 4 项；当进入下一次循环时，先输出第 3 项和 4 项，然后按上述方法求得第 5 项和 6 项，依次类推，即可求得前 40 项。

程序如下：

```
#include <iostream>
#include <iomanip>
using namespace std;
int main( )
{    long int f1,f2;
     int i;
     f1=1;f2=1;
     for (i=1;i<=20;i++)
     {   cout<<setw(12)<<f1<<setw(12)<<f2;
         if (i%2==0)
            cout<<endl;
         f1=f1+f2;
         f2=f2+f1;
     }
     return 0;
}
```

在例 3.22 中，if 语句的作用是在一行中输出 4 个数。

程序运行后输出：

1	1	2	3
5	8	13	21
34	55	89	144
233	377	610	987
1597	2584	4181	6765
10946	17711	28657	46368
75025	121393	196418	317811
514229	832040	1346269	2178309
3524578	5702887	9227465	14930352
24157817	39088169	63245986	102334155

【例 3.23】 100 名学生种 100 棵树，其中高中生每人种 3 棵树，初中生每人种 2 棵树，小学生每 3 人种 1 棵树，问高中生、初中生、小学生各有多少人？

分析：设有 i 名高中生、j 名初中生、k 名小学生，根据题意可以列出如下方程组：

$$\begin{cases} i+j+k=100 \\ 3i+2j+k/3=100 \end{cases}$$

三个变量只列出两个方程式，为不定方程组，此时可以用双重循环来求解，外循环的循环变量为 i，因为高中生每人种 3 棵树，所以 i 的最大值为 33；内循环的循环变量为 j，因为初中生每人种 2 棵树，所以 j 的最大值为 50。这种方法称为枚举法或穷举法。

种树问题流程图如图 3.26 所示。

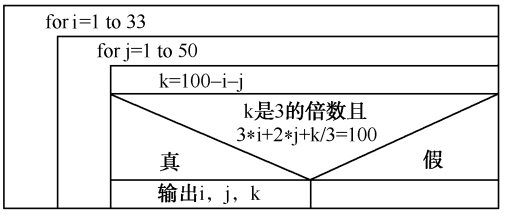


图 3.26　种树问题流程图

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int i,j,k;
     for (i=1;i<=33;i++)                                  //高中生人数 i，从 1 循环到 33
       for (j=1;j<=50;j++)                                //初中生人数 j，从 1 循环到 50
       {   k=100-i-j;                                     //计算小学生人数 k=100-i-j
           if ((k%3==0) && (3*i+2*j+k/3==100))            //小学生人数必须是 3 的倍数
                                                          //并且总共种树 100 棵
           cout<<i<<'\t'<<j<<'\t'<<k<<endl;               //输出一个解
       }
     return 0;
}
```

程序运行后输出：

```
 5        32        63
 10       24        66
 15       16        69
 20       8         72
```

【例 3.24】　编写程序，按下列格式打印九九乘法表。

```
*   1   2   3   4   5   6   7   8   9
1   1
2   2   4
3   3   6   9
4   4   8   12  16
5   5   10  15  20  25
6   6   12  18  24  30  36
7   7   14  21  28  35  42  49
8   8   16  24  32  40  48  56  64
9   9   18  27  36  45  54  63  72  81
```

分析：用双循环完成，外层用 for 语句构成循环，用于控制输出的行数（共 9 行）；内层也用 for 语句构成循环，用于控制每行的输出。

打印九九乘法表流程图如图 3.27 所示，程序如下：

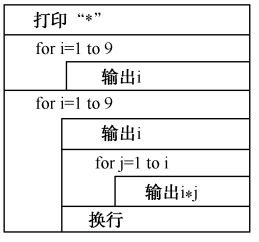


图 3.27　打印九九乘法表流程图

```
#include <iostream>
using namespace std;
int main( )
{    int i,j;
     cout<<"*"<<'\t';
     for (i=1;i<=9;i++)                          //输出标题行“*  1  2  3  4 … 9”
       cout<<i<<'\t';
     cout<<endl;
     for (i=1;i<=9;i++)                          //外循环控制第一个乘数 i 从 1 变到 9
     {   cout<<i<<'\t';
         for (j=1;j<=i;j++)                      //内循环控制第二个乘数 j 从 1 变到 i
            cout<<i*j<<'\t';                     //计算并输出每项 i*j
         cout<<endl;                             //输出完一行后换行
     }
     return 0;
}
```

本 章 小 结

一个 C++程序由若干源程序文件组成，一个源程序文件可以由编译预处理命令和若干函数组成，一个函数由函数说明和函数体组成，函数体由变量定义和若干执行语句组成。语句是组成程序的基本单元。

1．程序的三种基本结构

组成 C++程序的函数是由若干基本结构组成的。C++有三种基本结构，即顺序结构、分支结构和循环结构，各种控制结构是通过语句来实现的。

2．C++中的语句

C++中的语句可以分成六大类，即说明语句、控制语句、函数调用语句、表达式语句、空语句和复合语句，其中控制语句主要有分支语句与循环语句。

3．分支语句

分支语句用于实现分支结构程序设计。分支语句有 if 语句和 switch 语句。

（1）if 语句。

if 语句可以嵌套使用，但应当注意 else 与 if 的配对关系。C++规定：else 总是与其前面最近的还没有配对的 if 进行配对。

（2）条件运算符和条件表达式。

由条件运算符（?:）可以构成条件表达式。

（3）switch 语句。

在执行 switch 语句的过程中，每当执行完一个 case 后面的语句后，程序就会不加判断地自动执行下一个 case 后面的语句。如果要结束 switch 语句的执行，可用 break 语句来实现。

能够用分支结构程序解决的常见问题有：比较数的大小、找出若干数中的最大值与最小值、分段函数、解一元二次方程、判断闰年、多分支问题。

4．循环语句

循环语句用于实现循环结构程序设计。循环语句有 while 语句、do…while 语句和 for 语句。

（1）while 语句。

while 语句用来实现当型循环结构。

（2）do…while 语句。

do…while 语句用来实现直到型循环结构。

（3）for 语句。

for 语句是功能较强的一种循环语句。

应注意三种循环语句的特点、区别及由它们构成的嵌套循环结构。

能够用循环结构程序解决的常见问题有：累加和、连乘积、求一批数的和及最大值与最小值、求数列的前 *n* 项、判断素数、求两个整数的最大公约数和最小公倍数、用迭代法求平方根、用穷举法求不定方程组的整数解、打印图形等。

5．控制执行顺序的语句

（1）break 语句。

break 语句只能用在循环语句和 switch 语句中，其功能是终止循环语句和 switch 语句的执行。

（2）continue 语句。

continue 语句只能用在循环语句中，其功能是结束本次循环，重新开始下一次循环。

（3）goto 语句。

C++允许语句前带有一个标号，该标号称为语句标号。

goto 语句的功能是改变程序的执行流程，无条件地转移到指定语句标号的语句处执行。从结构化程序设计的角度出发，在进行程序设计时，应尽量避免使用 goto 语句。

6．本章重点、难点

重点：程序的三种基本结构，各种控制语句的格式、功能、执行过程及使用方法。

难点：使用 if 语句、while 语句、for 语句编写分支程序与循环程序。

习　　题

3.1　程序的三种基本结构是什么？

3.2　C++中的语句分为哪几类？

3.3　怎样区分表达式和语句？

3.4　程序的多路分支可通过哪两种语句来实现？说出用这两种语句实现多路分支的区别。

3.5　在使用 switch 语句时应注意哪些问题？

3.6　用于实现循环结构的循环语句有哪三种？分别用于实现哪两种循环结构？这三种循环语句在使用上有何区别？

3.7　分支程序与循环程序常用于解决哪些实际问题？

3.8　当 continue 语句与 break 语句用于循环结构时，在使用上有何区别？

3.9　程序的正常终止与异常终止有何区别，分别用什么函数来实现？在使用这些函数时应包含什么头文件？

3.10　设有程序段：

```
x=-1;
if  (a!=0)  {if  (a>0)  x=1;}  else    x=0;
```

写出该程序段表示的数学函数关系。

3.11　写出下列程序的运行结果。

```
#include <iostream>
using namespace std;
int main( )
{    int a=2,b=-1,c=2;
     if (a<b)
     if (b<0) c=0;
     else c=c+1;
     cout<<c<<endl;
     return 0;
}
```

3.12　写出下列程序的运行结果。

```
#include <iostream>
using namespace std;
int main( )
{    int i=10;
     switch (i)
        {   case    9:i=i+1;
            case   10:i=i+1;
            case   11:i=i+1;
            default :i=i+1;
        }
     cout<<i<<endl;
     return 0;
}
```

3.13　设计一个程序，判断通过键盘输入的整数的正负性和奇偶性。

3.14　有下列函数：

$$y=\begin{cases} -x+2.5 & (x<2) \\ 2-1.5(x-3)^2 & (2\leqslant x<4) \\ \dfrac{x}{2}-1.5 & (x\geqslant 4) \end{cases}$$

设计一个程序，通过键盘输入 x 的值，并输出 y 的值。

3.15　设计一个程序，通过键盘输入 a、b、c 三个整数，将它们按照从大到小的顺序输出。

3.16　输入平面直角坐标系中一点的坐标值（x，y），判断该点是在哪一个象限中或哪一条坐标轴上。

3.17　简单计算器。设计一个程序计算表达式 data1 op data2 的值，其中 data1、data2 为两个实数，op 为运算符（+、−、*、/），并且都由键盘输入。

3.18　奖金税率如下（a 代表奖金，r 代表税率）：

$a<500$	$r=0$
$500\leqslant a<1000$	$r=3\%$
$1000\leqslant a<2000$	$r=5\%$
$2000\leqslant a<5000$	$r=8\%$
$a\geqslant 5000$	$r=12\%$

输入一个奖金数，求税率、应交税款及实得奖金数。

3.19　写出下列程序的运行结果。

```
#include <iostream>
using namespace std;
int main( )
{    int n=4;
     while (--n)
        cout<<n<<'\t';
     cout<<endl;
      return 0;
}
```

3.20　写出下列程序的运行结果。

```
#include <iostream>
using namespace std;
int main( )
{    int x=3;
     do
       {cout<<x<<'\t';
       }
     while (!(x--));
     cout<<endl;
      return 0;
}
```

3.21　写出下列程序的运行结果。

```
#include <iostream>
using namespace std;
int main( )
{    int i=0,j=0,k=0,m;
     for (m=0;m<4;m++)
     switch (m)
     {   case 0:i=m++;
         case 1:j=m++;
         case 2:k=m++;
         case 3:m++;
     }
     cout<<i<<'\t'<<j<<'\t'<<k<<'\t'<<m<<endl;
      return 0;
}
```

3.22　写出下列程序的运行结果。

```
#include <iostream>
using namespace std;
int main( )
{     int n=0,m=0;
      for (int i=0;i<3;i++)
          for (int j=0;j<3;j++)
              if (j>=i) n++;m++;
     cout<<n<<'\n'<<m<<'\n';
     return 0;
}
```

3.23　求 $\sum_{n=1}^{100}\frac{1}{n}$ 的值，即求 $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{100}$ 的值。

3.24　编程计算 $y=1+\frac{1}{x}+\frac{1}{x^2}+\frac{1}{x^3}+\cdots$ 的值（$x>1$），直到最后一项小于 10^{-4}。

3.25　输入两个正整数 m 和 n，求其最大公约数和最小公倍数。

3.26　某月 10 天内的气温（℃）为：−5、3、4、0、2、7、0、5、−1、2，编程统计气温在 0℃以上、0℃和 0℃以下的天气各多少天？并计算这 10 天的平均气温值。

3.27　求 $\sum_{n=1}^{10}n!$，即求 1!+2!+3!+4!+…+10!。

3.28　设用 100 元买 100 支笔，其中钢笔每支 3 元，圆珠笔每支 2 元，铅笔每支 0.5 元，问钢笔、

圆珠笔和铅笔可以各买多少支（每种笔至少买 1 支）？

3.29　编程显示如图 3.28 所示的菱形。

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

图 3.28　编程输出菱形

实　验　A

1．实验目的

（1）初步学会 VC++开发环境中单步执行程序的方法。

（2）掌握 if 语句的格式与使用方法，学会两路分支程序的设计方法。

（3）掌握嵌套 if 语句的格式与使用方法，学会多路分支程序的设计方法。

（4）掌握 switch 语句的格式与使用方法，学会多路分支程序的设计方法。

2．实验内容

（1）演示单步执行程序的方法。

（2）设计一个程序，判断通过键盘输入的整数的正负性和奇偶性。

实验数据：−3 与 5。

（3）有下列函数：

$$y=\begin{cases}-x+3.5 & (x<5)\\ 20-3.5(x+3)^2 & (5\leqslant x<10)\\ \dfrac{x}{2}-3.5+\sin x & (x\geqslant 10)\end{cases}$$

设计一个程序，通过键盘输入 x 的值，并输出 y 的值。

实验数据：6 与 11。

（4）编写程序，将百分制成绩转换成等级制成绩。

转换方法为：

90～100	A
80～89	B
70～79	C
60～69	D
0～59	E

通过键盘输入百分制成绩，输出成绩等级。

实验数据：75

3．实验要求

（1）编写实验程序。

（2）在 VC++运行环境中输入源程序。

（3）单步执行程序。

（4）编译运行源程序。

（5）输入测试数据进行程序测试。

（6）写出运行结果。

实　验　B

1．实验目的

（1）掌握 while 语句的格式与使用方法，学会当型循环程序的设计方法。

（2）掌握 for 语句的格式与使用方法，学会当型循环程序的设计方法。

（3）掌握 do…while 语句的格式与使用方法，学会直到型循环程序的设计方法。

（4）学会求常用级数的编程方法。

2．实验内容

（1）输入一行字符，分别统计其中英文字母、空格、数字字符和其他字符的个数。

提示：用 cin.get(c)函数通过键盘输入一个字符给变量 c，直到输入换行字符'\n'。

实验数据：

```
I am Student 1234
```

（2）设有一个数列，前四项为 0、0、2、5，以后每项分别是其前四项之和，编程求此数列的前 20 项。

（3）求π的近似值的公式为：

$$\frac{\pi}{2}=\frac{2}{1}\times\frac{2}{3}\times\frac{4}{3}\times\frac{4}{5}\times\cdots\times\frac{2n}{2n-1}\times\frac{2n}{2n+1}\times\cdots$$

其中，n=1，2，3，…设计一个程序，求出当 n=1000 时π的近似值。

（4）求出 1～599 中能被 3 整除且至少有一位数字为 5 的所有整数。例如，15、51、513 均是满足条件的整数。

提示：将 1～599 中三位整数 i 分解成个位、十位、百位，分别存放在变量 a、b、c 中。然后判断 a、b、c 中是否有 5。将三位整数 i（设 i=513）分解成个位、十位、百位的方法是：

```
c=i%10;                 //c= i%10=513%10=3
a=i/10;                 //a= i/10=51
b=a%10;                 //b=a%10=51%10=1
a=a/10;                 //a=a%10=51/10=5
```

3．实验要求

（1）编写实验程序。

（2）在 VC++运行环境中输入源程序。

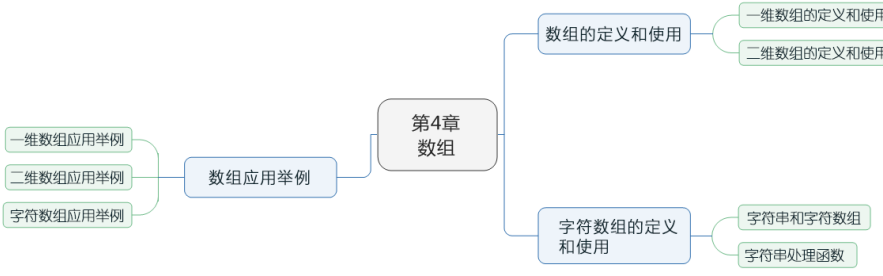
（3）编译运行源程序。

（4）输入测试数据进行程序测试。

（5）写出运行结果。

第 4 章

数组



本章知识点导图

通过本章的学习，应掌握一维数组和二维数组的定义、初始化赋值与使用方法，字符数组的定义、初始化赋值和使用方法，字符串处理函数的使用方法，利用数组编写程序的基本方法。

C++除了提供基本数据类型，还提供了导出数据类型，它们有数组、指针、结构体、共同体和类，它们都是将基本数据类型数据按照一定规律构造而成的。

所谓数组，就是把一系列有序的相同类型的数据组合起来的数据集合。每个数组都有一个名字，即数组名。数组中的每个数据称为数组元素，数组元素在数组中的位置由下标确定。根据数组元素下标的个数，数组分为一维数组和多维数组。当数组元素的数据类型为字符型时，该数组称为字符数组。

4.1 数组的定义和使用

4.1.1 一维数组的定义和使用

1. 一维数组的定义与初始化赋值

（1）一维数组的定义。在使用数组前，必须对数组进行定义。一维数组的定义包含对数组名、数组元素的数据类型和个数的说明。一维数组的定义格式为：

〔存储类型〕<类型> <数组名>[<常量表达式>];

说明：

① 存储类型是数组元素的存储类型，是可选的，有关存储类型的概念会在第 5 章中介绍。

② 类型说明了数组中数组元素的数据类型，可以是 C++的基本数据类型，也可以是导出数据类型。

③ 数组名的确定应符合标识符的命名规则。

④ 常量表达式的值是一个正整数，它规定了数组中数组元素的个数，即数组长度。常量表达式中可以包含常量和符号常量，不包含变量。通常情况下为整型常量或由整型常量构成的表达式。

⑤ 数组元素的下标从 0 开始且不能超出范围。例如：

```
int  a[10];
```

表示定义了一个名为 a 的整型数组，它有 10 个元素，分别为 a[0]、a[1]、a[2]、a[3]、a[4]、a[5]、a[6]、a[7]、a[8]、a[9]。

（2）一维数组的初始化赋值。与一般变量一样，在数组定义的同时就可以对数组元素赋初值。方法是，从数组的第一个元素开始依次给出初值，形成一个初值表，表中各初值之间用逗号分开，初值表用一对花括号括起来。一维数组初始化赋值的方法有以下几种。

① 给数组的所有元素赋初值。给数组的所有元素赋初值的方法为：将每个元素的初值放在一个用花括号括起来的、各初值间用逗号分开的初值表中。例如：

```
int  a[10]={1,2,3,4,5,6,7,8,9,10};
```

表示数组 a 中的所有数组元素 a[0]、a[1]、a[2]、a[3]、a[4]、a[5]、a[6]、a[7]、a[8]、a[9]分别获得初值 1、2、3、4、5、6、7、8、9、10。

如果一个数组在定义时对它的所有元素赋了初值，那么可以不指定数组的长度，系统会自动计算数组的长度。例如：

```
int  b[ ]={1,2,3,4,5};
```

系统自动计算数组 b 的长度为 5，即数组 b 有 5 个数组元素 b[0]、b[1]、b[2]、b[3]、b[4]，并分别获得初值 1、2、3、4、5。

② 给数组的部分元素赋初值。给数组的部分元素赋初值的方法与给数组的所有元素赋初值的方法类似。例如：

```
int  a[10]={1,2,3,4,5};
```

表示数组 a 中的前 5 个元素 a[0]、a[1]、a[2]、a[3]、a[4]分别获得初值 1、2、3、4、5，其他元素的值为 0。

③ 当把数组定义为全局变量或静态变量时，所有数组元素的初值均为 0；当把数组定义为其他存储类型的局部变量时，数组元素没有确定的值，即其值是随机的。关于局部变量、全局变量、静态变量的概念会在后面章节中介绍。

2．一维数组在内存中的存储方式

当定义了一个数组后，系统就会为数组分配一串连续的内存单元，来依次存放各个数组元素。

例如，在定义数组 int　a[10]={1,2,3,4,5,6,7,8,9,10}；后，系统将为数组 a 分配 10 个元素的存储空间，每个元素占 4B，其存储空间的分配情况如图 4.1 所示。

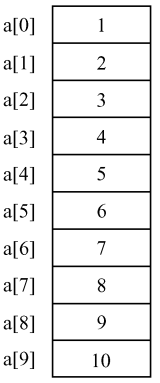


图 4.1　一维数组的存储方式

3．一维数组元素的访问

数组必须先定义后使用。C++规定只能对数组中的元素进行访问，不能把整个数组作为一个整体使用。一维数组元素的访问形式为：

<数组名>[<下标表达式>]

其中，下标表达式的值就是被访问的数组元素的下标，其数据类型必须为整型。

【例 4.1】　通过键盘将 10 个整数依次输入一个数组中，然后按倒序输出。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int a[10],i;
     cout<<"Input ten integers:";
     for(i=0;i<=9;i++)
        cin>>a[i];
     for(i=9;i>=0;i--)
        cout<<a[i]<<'\t';
     cout<<endl;
     return 0;
}
```

程序执行后显示：

```
Input ten integers:      0    1    2    3    4    5    6    7    8    9
                         9    8    7    6    5    4    3    2    1    0
```

4．一维数组应用举例

【例 4.2】　某小组有 10 名学生进行了数学考试，求他们数学成绩的平均分、最高分和最低分。

分析：求 N 个数的平均值的方法是，首先求 N 个数的累加和，并保存在变量 sum 中，然后将累加和 sum 除以数据个数 N，即可求得 N 个数的平均值。求 N 个数中的最大值的方法是，首先设变量 max，用于存放第 1 个数，然后将余下的数按次序分别与 max 进行比较，若某数大于 max，则将其值赋给 max，若某数小于 max，则 max 的值不变，当余下的数都比较完后，max 中存放的就是 N 个数中的最大值。求 N 个数中的最小值的方法与求 N 个数中的最大值的方法类似。求一批数的平均值、最大值、最小值流程图如图 4.2 所示，程序如下：

```
#include <iostream>
#define N 10
using namespace std;
int main( )
{    float a[N],sum,ave,max,min;
     int i;
     cout<<"Input integers:";
     for (i=0;i<N;i++)
        cin>>a[i];
     sum=0;
     max=a[0];
     min=a[0];
     for (i=0;i<N;i++)
        {   sum=sum+a[i];
            if (a[i]>max)
               max=a[i];
            if (a[i]<min)
               min=a[i];
        }
     ave=sum/N;
     cout<<"ave="<<ave<<','<<"max="<<max<<','<<"min="<<min;
     cout<<endl;
     return 0;
}
```

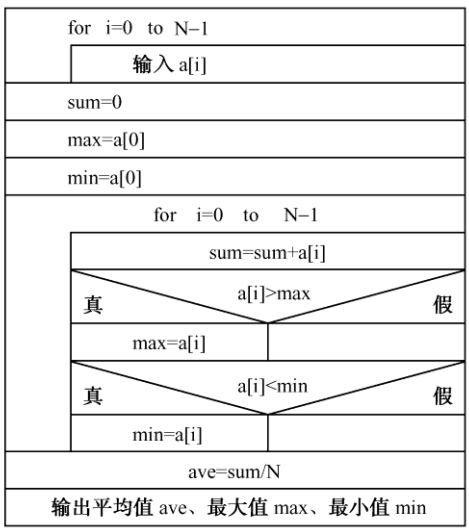


图 4.2　求一批数的平均值、最大值、最小值流程图

程序运行后显示：

```
Input integers:90   87   68   92   56   78   85   90   98   64
ave=80.8,max=98,min=56
```

【例 4.3】　将一个数组的内容按颠倒的次序重新存放。例如，数组中数组元素原来的值依次为：8、3、5、1、9、7、2，要求改为：2、7、9、1、5、3、8。

分析：设数组 a 有 n 个元素，将数组 a 中的内容按颠倒的次序重新存放，只需将元素 a[0]的内容与元素 a[n−1]的内容交换，将元素 a[1]的内容与元素 a[n−2]的内容交换，…，将元素 a[i]的内容与元

素 a[n−i−1]的内容交换即可；i 的取值范围为 0 到 n/2（取整）−1。例如，将数组内容 8、3、5、1、9、7、2 改为 2、7、9、1、5、3、8 的方法如图 4.3 所示。

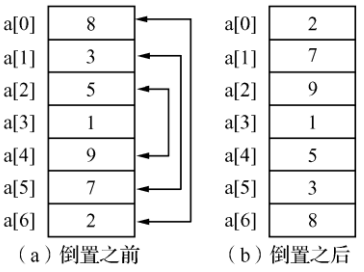


图 4.3　数组内容倒置

程序如下：

```
#include <iostream>
#include <iomanip>
#define N 7
using namespace std;
int main( )
{    int a[N],i,temp;
     cout<<"Input 7 integers:";
     for (i=0;i<N;i++)
        cin>>a[i];
     for (i=0;i<N;i++)
        cout<<setw(4)<<a[i];
     cout<<endl;
     for (i=0;i<=N/2−1;i++)
        {   temp=a[i];
            a[i]=a[N−i−1];
            a[N−i−1]=temp;
        }
     for (i=0;i<N;i++)
        cout<<setw(4)<<a[i];
     cout<<endl;
     return 0;
}
```

程序执行后显示：

```
Input 7 integers:    8    3    5    1    9    7    2
                     8    3    5    1    9    7    2
                     2    7    9    1    5    3    8
```

【例 4.4】　某班有 10 名学生，进行了数学考试，现要求将数学成绩按由低到高的顺序排序。


本题微课视频

分析：排序是指将一组无序的数据按从小到大（升序）或从大到小（降序）的次序重新排列。常用的排序方法有三种：冒泡法、选择法和擂台法，现介绍冒泡法和选择法。

（1）冒泡法。先举一实例来说明冒泡法排序的原理：要求将 5 个数 7、5、6、3、2 按从小到大的次序排列。冒泡排序法如图 4.4 所示。

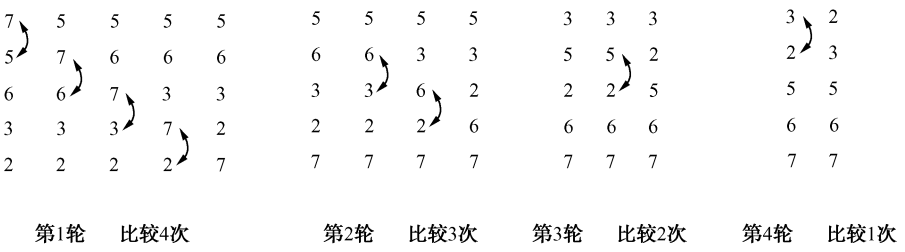


图 4.4 冒泡排序法

第 1 轮第 1 次将第 1 个数 7 和第 2 个数 5 进行比较，因为 7 大于 5，所以将它们交换；第 2 次将第 2 个数 7 和第 3 个数 6 进行比较，因为 7 大于 6，所以将它们交换；按同样方法进行第 3 次、第 4 次比较与交换。当第 1 轮结束时，最大数 7 已“沉底”，而小的数“上升”，第 1 轮共进行了 4 次比较，得到 5、6、3、2、7 的数据序列。第 2 轮第 1 次将第 1 个数 5 和第 2 个数 6 进行比较，因为 5 小于 6，所以不变。按同样方法进行第 2 次、第 3 次比较与交换，第 2 轮共进行了 3 次比较，得到 5、3、2、6、7 的数据序列。依次类推，第 3 轮进行 2 次比较，得到 3、2、5、6、7 的数据序列。第 4 轮进行 1 次比较，得到 2、3、5、6、7 的数据序列。至此完成了排序工作。

由以上分析可知，对 5 个数进行排序，要进行 4 轮比较，第 1 轮要进行 4 次相邻两个数间的比较，第 2 轮要进行 3 次相邻两个数间的比较，第 3 轮要进行 2 次相邻两个数间的比较，第 4 轮只进行 1 次相邻两个数间的比较。在比较时，若前一个数大于后一个数，即上大下小则交换。这样才能使 5 个数按从小到大的次序排列。这种排序方法被形象地称为冒泡法，在排序的过程中，小的数就像气泡一样逐层上冒，而大的数则逐个下沉。

在进行程序设计时，可以将这 5 个数存入一个数组 a 中。

用 N=5 表示数组 a 中元素的个数，用双重循环实现其排序的操作步骤如下。

① 外循环用变量 i 控制轮次，i 的值从 0 到 3（3=N−2），共进行 N−1 轮。

用循环语句表示为：for(i=0;i<=N−2;i++)或 for(i=0;i<N−1;i++)。

② 内循环用变量 j 控制第 i 轮比较的次数，j 从 0 到 3−i（3−i=N−2−i），共 N−1−i 次。

用循环语句表示为：for(j=0;j<=N−2−i;j++)或 for(j=0;j<N−1−i;j++)。

③ 每次相邻两数两两相比（a[j]与 a[j+1]相比），上大下小则交换（见图 4.5）的语句描述为：

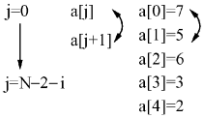


图 4.5 第 i 轮相邻两数两两相比，上大下小则交换

```
if (a[j]>a[j+1]){ temp=a[j]; a[j] =a[j+1]; a[j+1]=temp; }
```

程序如下：

```
#include <iostream>
#include <iomanip>
#define N 10
using namespace std;
int main( )
{    float a[N],temp;
     int i,j;
     cout<<"Input score:";
     for(i=0;i<N;i++)
        cin>>a[i];
     for (i=0;i<N−1;i++)
        for (j=0;j<N−i−1;j++)
```

```
            if (a[j]>a[j+1])
               {   temp=a[j]; a[j] =a[j+1]; a[j+1]=temp; }
        for (i=0;i<N;i++)
           cout<<setw(7)<<a[i];
        cout<<endl;
        return 0;
    }
```

程序运行后显示：

```
Input score:90   78   68   96   88   75   67   85   92   84
67   68   75   78   84   85   88   90   92   96
```

对于降序排序的冒泡法，只要将上述步骤③改为相邻两数两两相比，上小下大则交换。语句描述为：

```
if (a[j]<a[j+1]){ temp=a[j]; a[j] =a[j+1]; a[j+1]=temp; }
```

为便于读者记忆，将升序排序冒泡法归结为口诀：相邻两数两两相比，上大下小则交换，共进行 N−1 轮，每轮进行 N−1−i 次。降序排序冒泡法口诀由读者自己总结。

（2）选择法。同样，先举一实例来说明选择法排序的原理：要求将 5 个数 7、5、6、3、2 按从小到大的次序排列。选择排序法如图 4.6 所示。

选择排序法的基本思想是首先用求最小值的算法找出 5 个数中的最小数 2，将它放在第 1 个数的位置；其次在余下的 4 个数中找出最小数 3，将它放在第 2 个数的位置；然后在余下的 3 个数中找出最小数 5，将它放在第 3 个数的位置，依次类推，就能将 5 个数按从小到大的次序排列。具体实现方法如下。

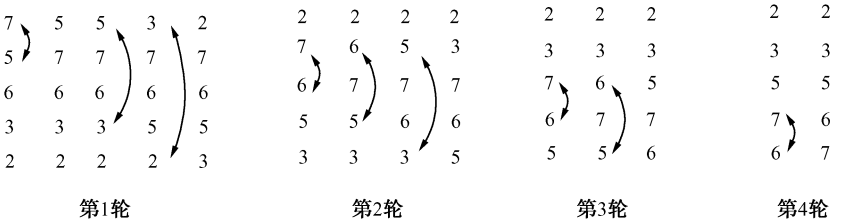


图 4.6　选择排序法

第 1 轮第 1 次将第 1 个数 7 与第 2 个数 5 进行比较，因为 7 大于 5，所以将它们交换，使第 1 个数成为 5；第 2 次将第 1 个数 5 与第 3 个数 6 进行比较，因为 5 小于 6，所以保持原状不变。按同样方法进行第 3 次、第 4 次比较与交换。当第一轮结束时，最小数 2 已处在第 1 个数的位置，得到 2、7、6、5、3 的数据序列，即第 1 轮取出第 1 个数，让它与其下一个数（第 2 个数）到最后一个数（第 5 个数）逐一进行比较，若取出的第 1 个数比后面的一个数大，则交换它们的值，否则不交换。

第 2 轮第 1 次，因第 1 个数 2 已是 5 个数中的最小数，所以不必参与比较，只要将第 2 个数与其下一个数（第 3 个数）到最后一个数（第 5 个数）逐一进行比较，若第 2 个数比后面的一个数大，则交换它们的值，否则不交换。当第 2 轮比较结束时，得到 2、3、7、6、5 的数据序列。依次类推，共进行 4 轮比较就可以将 5 个数按从小到大的次序排列，得到 2、3、5、6、7 的数据序列。由此得出结论，将 N 个数按从小到大的次序排序要进行 N−1 轮比较，第 i 轮比较是将第 i 个数（首数）与后面的数（第 i+1 个数到最后一个数）逐一进行比较，若首数大于后面的数则交换它们值。

现将 N 个数存入数组 a 中。用双重循环来实现其排序操作的步骤如下。

① 外循环用变量 i 控制轮次，i 的值从 0 到 3（3=N−2），共进行 N−1 轮。

用循环语句表示为：for(i=0;i<=N−2;i++)或 for(i=0;i<N−1;i++)。

② 内循环用变量 j 控制第 i 轮比较的次数，j 从 i+1 到 N−1，共 N−1−i 次。

用循环语句表示为：for(j=i+1;j<=N−1;j++)或 for(j=i+1;j<N;j++)。

③ 每次首数与后面的数两两相比（a[i]与 a[j]相比，j=i+1 到 N−1），首大后小则交换（见图 4.7）的语句描述为：

```
if (a[i]>a[j]){ temp=a[i]; a[i] =a[j]; a[j]=temp; }
```

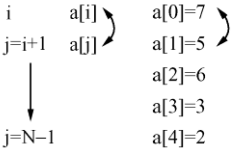


图 4.7　第 i 轮首数与后面的数两两相比，首大后小则交换

程序如下：

```
#include <iostream>
#include <iomanip>
#define N 10
using namespace std;
int main( )
{    float a[N],temp;
     int i,j;
     cout<<"Input score:";
     for (i=0;i<N;i++)
        cin>>a[i];
     for (i=0;i<N−1;i++)
        for (j=i+1;j<N;j++)
           if (a[i]>a[j])
              {   temp=a[i]; a[i]=a[j]; a[j]=temp;}
     for (i=0;i<N;i++)
        cout<<setw(5)<<a[i];
     cout<<endl;
     return 0;
}
```

对于降序排序的选择法，只要将上述步骤③改为首数与后面的数两两相比，首小后大则交换。语句描述为：

```
if(a[i]<a[j]){ temp=a[i]; a[i] =a[j]; a[j]=temp; }
```

为便于读者记忆，将升序排序选择法归结为口诀：首数与后数两两相比，首大后小则交换，共进行 N−1 轮，每轮从 i+1 开始到 N−1 结束。降序排序选择法口诀由读者自己总结。

4.1.2　二维数组的定义和使用

1．二维数组的定义与初始化赋值

（1）二维数组的定义。二维数组的定义格式为：

```
〔存储类型〕<类型> <数组名>[<常量表达式 1>][<常量表达式 2>];
```

说明：

常量表达式 1 指明了二维数组的行数，常量表达式 2 指明了二维数组中每行的元素个数，即二维数组的列数。其他与一维数组相同。例如：

```
int   a[3][4];
```

表示定义了一个二维数组 a，其元素的数据类型为整型，它有 3 行 4 列共 12 个元素，分别为：

```
a[0][0]  a[0][1]  a[0][2]  a[0][3]
a[1][0]  a[1][1]  a[1][2]  a[1][3]
a[2][0]  a[2][1]  a[2][2]  a[2][3]
```

可以把二维数组看作一种特殊的一维数组，即它的每个元素又是一个一维数组。

（2）二维数组的初始化赋值。与一维数组一样，二维数组也可以进行初始化赋值。二维数组初始化赋值的方法有以下几种。

① 给数组的所有元素赋初值。给数组的所有元素赋初值的方法有以下两种。

方法一：按行的顺序将每行元素的初值放在一个用花括号括起来的、各初值间用逗号分开的序列中，用花括号括起来的序列间用逗号分隔，全部初值再用一个花括号括起来。例如：

```
int  a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
```

方法二：将所有初值放在一个花括号括起来的序列中，按数组排列的顺序给各元素赋初值。例如：

```
int  a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};
```

如果给二维数组的所有元素赋初值，则在定义二维数组时，第一维的长度（数组的行数）可以不指定，而第二维的长度（数组的列数）必须指定。例如：

```
int  a[ ][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
```

或

```
int  a[ ][4]={1,2,3,4,5,6,7,8,9,10,11,12};
```

② 给数组的部分元素赋初值。给数组的部分元素赋初值的方法与给数组的所有元素赋初值的方法类似。例如：

```
int  a[3][4]={{1,2},{5},{9,10,11}};
```

表示给二维数组 a 的元素 a[0][0]、a[0][1]、a[1][0]、a[2][0]、a[2][1]、a[2][2]赋了初值，其余元素的初值为 0。

2．二维数组在内存中的存储方式

虽然在逻辑上可以把二维数组看成一张表格或一个矩阵，但是在计算机中存储二维数组时，需要在内存中开辟一串连续的内存单元，依次存放各个数组元素。

C++中是按行顺序存放二维数组的各个数组元素的，即先存放第一行的元素，再存放第二行的元素，依次把各行的元素存入一串连续的内存单元中。

例如，把前面定义的数组 a 存储在内存中时各数组元素的排列顺序如图 4.8 所示。

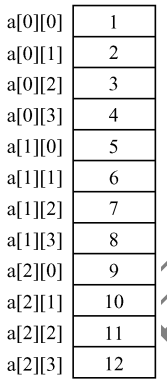


图 4.8　二维数组的存储方式

3．二维数组元素的访问

二维数组元素的访问形式为：

<数组名>[<下标表达式 1>][<下标表达式 2>]

其中，下标表达式 1 和下标表达式 2 的值就是被访问的数组元素的两个下标，其数据类型必须为整型。

【例 4.5】　通过键盘给一个 3 行 4 列的二维数组输入整型数值，并按表格形式输出此数组的所有元素。

程序如下：

```
#include <iostream>
#include <iomanip>
using namespace std;
int main( )
{    int a[3][4],i,j;
     cout<<"Input twelve integers:";
     for (i=0;i<3;i++)
        for (j=0;j<4;j++)
           cin>>a[i][j];
     for (i=0;i<3;i++)
     {   for (j=0;j<4;j++)
              cout<<setw(4)<<a[i][j];
         cout<<endl;
     }
     return 0;
}
```

程序执行后显示：

```
Input twelve integers: 1    2    3    4    5    6    7    8    9    10   11   12
1    2    3    4
5    6    7    8
9    10   11   12
```

多维数组的定义和使用方法与二维数组的定义和使用方法类似。

4．二维数组应用举例

【例 4.6】　某小组有 5 名学生，考了 3 门课程，他们的学号及成绩，如表 4.1 所示，试编程求每名学生的平均成绩，并按表格形式输出每名学生的学号、3 门课程的成绩和平均成绩。

本题微课视频

表 4.1　学生成绩情况表一

学　　号	数　　学	语　　文	外　　语	平 均 成 绩
1001	90	80	85	
1002	70	75	80	

1003	65	70	75	
1004	85	50	60	
1005	80	90	70	

分析：可定义一个具有 5 行 5 列的二维数组来存放 5 名学生的学号、数学成绩、语文成绩、外语成绩、平均成绩。当统计某学生的平均成绩时，只需将二维数组某一行的数学成绩、语文成绩、外语成绩相加除以 3，然后将其存入平均成绩对应元素即可。像这类数据处理问题，采用程序让计算机来处理的一般过程为：输入数据→处理数据→输出数据。

程序如下：

```
#include <iostream>
#include <iomanip>
#define M 5
#define N 5
using namespace std;
int main( )
{   int s[M][N];
    float sum;
    int i,j;
    cout<<"Input data:\n";                                  //输入数据
    for (i=0;i<M;i++)                                       //输入 5 名学生的学号与 3 门课程的成绩
      for (j=0;j<N-1;j++)
          cin>>s[i][j];
    for (i=0;i<M;i++)                                       //处理数据
    {   sum=0.0;
        for (j=1;j<N-1;j++)                                 //计算每名学生的总成绩
            sum=sum+s[i][j];
        s[i][j]=sum/(N-2);                                  //计算每名学生的平均成绩
    }
    cout<<setw(5)<<" Num."<<" Math. Chin. Engl.Ave."<<endl;  //输出数据
    cout<<"-----------------------------\n";
    for (i=0;i<M;i++)
    {   for (j=0;j<N;j++)                                   //输出每名学生的学号与成绩
            cout<<setw(6)<<s[i][j];
        cout<<endl;
    }
    cout<<"-----------------------------\n";
    return 0;
}
```

程序运行后，提示输入学号与成绩：

```
Input data:
1001    90   80   85
1002    70   75   80
1003    65   70   75
1004    85   50   60
1005    80   90   70
Num.   Math.   Chin.   Engl.   Ave.
-----------------------------------------------
1001     90      80      85      85
1002     70      75      80      75
1003     65      70      75      70
1004     85      50      60      65
1005     80      90      70      80
```

【例 4.7】 将一个二维数组的行和列元素互换，形成另一个二维数组，即数组的转置运算。例如：

数组 a

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

数组 b

$$\begin{pmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{pmatrix}$$

分析：将数组 a 转置成数组 b，只要将其每个数组元素的两个下标交换即可，即 b[i][j]=a[j][i]，程序如下：

```
#include <iostream>
#include <iomanip>
using namespace std;
int main( )
{    int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
     int b[4][3];
     int i,j;
     cout<<"Array a:\n";
     for(i=0;i<3;i++)                                    //输出数组 a
     {   for(j=0;j<4;j++)
              cout<<setw(5)<<a[i][j];
         cout<<endl;
     }
     for (i=0;i<4;i++)                                   //将数组 a 转置成数组 b
       for (j=0;j<3;j++)
         b[i][j]=a[j][i];
     cout<<"Array b:\n";
     for (i=0;i<4;i++)                                   //输出数组 b
     {   for (j=0;j<3;j++)
              cout<<setw(5)<<b[i][j];
         cout<<endl;
     }
     return 0;
}
```

程序运行后输出：

```
Array a:
    1    2    3    4
    5    6    7    8
    9   10   11   12
Array b:
    1    5    9
    2    6   10
    3    7   11
    4    8   12
```

4.2 字符数组的定义和使用

4.2.1 字符串和字符数组

1．字符串及其结束标志

字符串就是用一对双引号引起来的字符序列，如"C++ program"、"A"、"It is an integer.\n"等。字符串中的字符可以是能显示的字符，也可以是转义字符（如换行符"\n"）。

在 C++中，为了判断字符串是否结束，系统会自动在字符串的末尾加一个字符'\0'，作为字符串的结束标志。例如，字符串 "C++ program"共 11 个字符，但实际上它在内存中占 12 个内存单元，最后一个内存单元存放'\0'，作为字符串的结束标志，因此它有 12 个字符。

需要指出的是，'\0'代表 ASCII 码为 0 的字符，它不是一个可显示的字符，而是一个“空操作”字符，即它什么也不做。用它作为字符串的结束标志不会产生任何附加操作或增加有效字符，只是作为一个供识别的标志。

注意：字符串和字符是有区别的，字符串是用双引号引起来的字符序列，而字符则是用单引号引起来的单个字符，它们所占的存储空间也不同。例如，"A"是字符串，而'A'则是字符；在内存中字符串"A"占两个内存单元，而字符'A'仅占一个内存单元。

2．字符数组

字符数组就是存放字符数据的数组。在字符数组中，一个元素存放一个字符。字符数组必须先定义后使用。

（1）字符数组的定义。字符数组的定义与一般数组的定义完全相同，只是在字符数组中，每个元素的数据类型为字符型。定义字符数组的一般格式为：

〔存储类型〕 char <数组名>[<表达式>];

例如：

```
char s[10];
```

在 C++中，将字符串作为字符数组来处理，也就是说，用字符数组来存放字符串。例如，为了存放字符串"C++ program"，应定义一个字符数组：

```
char  str[12];
```

可以用前面介绍的方法，对定义的字符数组的元素一个一个地进行处理；但通常都是将字符数组作为一个整体（字符串）来处理的。

（2）字符数组的初始化赋值。对字符数组进行初始化赋值的方法有以下两种。

① 给字符数组的各个元素逐个赋初值。例如：

```
char  str[12]={'C','+','+',' ','p','r','o','g','r','a','m','\0'};
```

在用这种方法对字符数组进行初始化赋值时，可以不指定字符数组的长度。例如：

```
char  str[ ]={'C','+','+',' ','p','r','o','g','r','a','m','\0'};
```

② 给字符数组指定一个字符串初值。例如：

```
char str[]={"C++ program"};
```

其中花括号“{ }”可以省略，即可以写成：

```
char str[]="C++ program";
```

在用这种方法对字符数组进行初始化赋值时，系统会自动在最后一个字符后面加一个字符'\0'，表示字符串的结束。字符数组的长度为12，而不是11。字符数组的存储方式如图4.9所示。

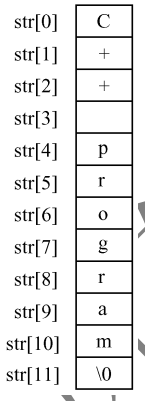


图4.9 字符数组的存储方式

（3）字符数组的输入与输出。

字符数组和数值数组一样，可对其中的元素进行逐个输入/输出。但常把字符数组作为字符串进行整体的输入/输出。

在将字符数组作为字符串进行输入/输出时，在cin/cout中仅给出字符数组名即可。

【例4.8】 将两个字符串分别输入两个字符数组中，并输出这两个数组中的字符串。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{     char s1[40],s2[40];
      cout<<"Input two strings:";
      cin>>s1>>s2;
      cout<<"s1="<<s1<<'\t'<<"s2="<<s2<<endl;
      return 0;
}
```

说明：

① 在输入字符串时，如果遇到空格字符或换行字符（“Enter”键），则认为一个字符串结束，接着的非空格字符作为一个新的字符串的开始，并且系统会自动在每个字符串后加一个'\0'。例如，在例4.8中，如果输入：

```
very good
```

则s1的内容为字符串"very"，s2的内容为字符串"good"。

当要把输入的一行字符（包括空格字符）作为一个字符串送到字符数组中时，要使用函数cin.getline(str,n)，该函数的第一个参数str为字符数组名，第二个参数n为允许输入的最大字符个数。例如：

```
char s[12];
cin.getline(s,12);
```

如果输入：

```
C++ program
```

则字符数组 s 的内容为"C++ program"。

② 当一个字符数组作为一个字符串输出时，必须保证在数组中包含字符串结束标志'\0'，当遇到'\0'时，输出自动结束，且'\0'不输出。因此，在例 4.8 中，如果输入 very good，则输出为：

```
s1=very    s2=good
```

4.2.2 字符串处理函数

在 C++的库函数中，提供了字符串处理函数，它们包含在头文件 string 中。这里介绍几个常用的字符串处理函数及其使用方法。

1．求字符串长度函数 strlen()

格式：strlen(<字符串>)。

功能：求字符串的长度。

例如，设有如下程序：

```
char str[]="C++ program";
cout<<strlen(str)<<endl;
```

则输出字符串的长度为 11。

说明：

(1) 字符串可以是字符数组名，也可以是字符串常量。

(2) 函数值为字符串的实际长度，不包括最后的'\0'。

2．字符串复制函数 strcpy()

格式：strcpy(<字符数组 1>,<字符串 2>)。

功能：将字符串 2 复制到字符数组 1 中。

例如，设有如下程序：

```
char str1[12],str2[]="C++ program";
strcpy(str1,str2);
```

则字符数组 str1 的内容为"C++ program"。

说明：

(1) 字符数组 1 必须是字符数组名，字符串 2 可以是字符数组名，也可以是字符串常量。

(2) 字符数组 1 必须足够大，以便容纳被复制的字符串 2。

(3) 字符串 2 后的'\0'也一起被复制到字符数组 1 中。

(4) 在赋值运算符“=”没有重载之前，不能用赋值语句将一个字符串常量或字符数组赋给另一个字符数组，只能用字符串复制函数来处理。例如，下面的操作是非法的：

```
str2="C++ program";
str1=str2;
```

有关运算符重载的概念会在第 11 章进行介绍。

3．字符串连接函数 strcat()

格式：strcat(<字符数组 1>,<字符串 2>)。

功能：将字符串 2 连接到字符数组 1 中的字符串后，其结果存放在字符数组 1 中。

例如，设有如下程序：

```
char str1[30]="I am a ";
char str2[]="student.";
strcat(str1,str2);
```

则字符数组 str1 的内容为"I am a student."。

说明：

(1) 字符数组 1 必须是字符数组名，字符串 2 可以是字符数组名，也可以是字符串常量。

(2) 字符数组 1 必须足够大，以便容纳连接后的新字符串。

(3) 在连接时，字符串 1 之后的'\0'取消，只在新字符串最后保留一个'\0'。

4．字符串比较函数 strcmp()

格式：strcmp(<字符串 1>,<字符串 2>)。

功能：将两个字符串从左到右进行逐个字符的比较（按 ASCII 码的大小比较），直到出现不同的字符或遇到'\0'。如果所有字符都相同，则认为两个字符串相等；如果出现不相同的字符，则以第一个不相同的字符的比较结果作为两个字符串的比较结果。比较结果由函数值返回。

（1）如果字符串 1=字符串 2，则函数值为 0。

（2）如果字符串 1>字符串 2，则函数值为一个正整数。

（3）如果字符串 1<字符串 2，则函数值为一个负整数。

说明：

(1) 字符串 1 和字符串 2 都可以是字符数组名，也都可以是字符串常量。

(2) 在比较运算符（如==）没有重载之前，两个字符串不能用关系运算符进行比较，而只能用字符串比较函数来处理。例如，下面的操作是非法的：

```
if(str1==str2)
   cout<<"True";
else
   cout<<"False";
```

5．字符串中大写字母变换为小写字母函数 strlwr()

格式：strlwr(<字符数组>)。

功能：将字符数组中的所有大写字母变换为小写字母。

例如，设有如下程序：

```
char str[]="CHINA";
strlwr(str);
```

则 str 中的内容变换为"china"。

6．字符串中小写字母变换为大写字母函数 strupr()

格式：strupr(<字符数组>)。

功能：将字符数组中的所有小写字母变换为大写字母。

例如，设有如下程序：

```
char str[]="china";
strupr(str);
```

则 str 中的内容变换为"CHINA"。

【例 4.9】　通过键盘输入两个字符串，将它们连接成一个字符串。


本题微课视频

方法一：使用字符串连接函数 strcat()连接两个字符串。

程序如下：

```
#include <iostream>
#include <string>
using namespace std;
int main( )
{    char str1[40],str2[20];                    //定义字符数组 str1、str2
     cout<<"Input two strings:\n";
     cin.getline(str1,19);                      //输入字符串 1 到 str1 中
     cin.getline(str2,19);                      //输入字符串 2 到 str2 中
     strcat(str1,str2);                         //将 str1、str2 连接后存入 str1 中
     cout<<str1<<endl;                          //输出 str1
     return 0;
}
```

程序运行后显示：

```
Input two strings:
very
good
```

输出：

```
very good
```

方法二：不使用字符串连接函数 strcat()连接两个字符串。

定义两个变量 i、j，首先使 i 指向字符数组 str1 的第一个元素，移动 i 使其指向 str1 的末尾，即指向 str1 的最后一个元素'\0'；其次使 j 指向字符数组 str2 的第一个元素，将 str2 中的元素分别赋给 str1 中相应位置的元素，直到 str2 结束，即 j 指向 str2 的最后一个元素'\0'；最后在 str1 的末尾添加一个结束标志'\0'，如图 4.10 所示。

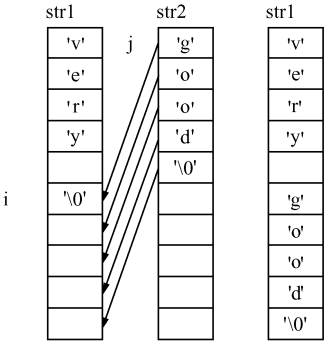


图 4.10　字符串的连接

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    char str1[40],str2[20];                    //定义字符数组 str1、str2
     int i,j;
     cout<<"Input two strings:\n";
     cin.getline(str1,19);                      //输入字符串 1 到 str1 中
     cin.getline(str2,19);                      //输入字符串 2 到 str2 中
     i=0;                                       //使 i 指向 str1 的第一个元素
     while (str1[i]!='\0')                      //判断 str1 是否结束
         i++;                                   //str1 没有结束，使 i 指向下一个元素
```

```
    j=0;                                    //使 j 指向 str2 的第一个元素
    while (str2[j]!='\0')                   //判断 str2 是否结束
      {  str1[i]=str2[j];                   //str2 没有结束，将 str2[j]赋给 str1[i]
         i++;                               //使 i 指向 str1 的下一个元素
         j++;                               //使 j 指向 str2  的下一个元素
      }
    str1[i]='\0';                           //在 str1 的末尾添加结束标志'\0'
    cout<<str1<<endl;                       //输出 str1
    return 0;
}
```

4.3 数组应用举例

前面主要介绍了一维数组、二维数组和字符数组的定义与引用，现就这三类数组的定义、引用、数据输入/输出、主要应用总结如下。

4.3.1 一维数组应用举例

1．一维数组内容小结

（1）定义格式：

〔存储类型〕<类型><数组名>[长度]={初值};

例如，N 个元素组成的一维整型数组 a[N]的定义格式为：

```
# define N 10
int a[N]={1,2,3,4,5,6,7,8,9,10};
```

（2）元素引用：

<数组名>[下标表达式]

（3）输入/输出：一维数组元素只允许进行单个元素的输入/输出。例如：

```
int i;
```

输入语句段：for(i=0;i<N;i++) cin>>a[i];

输出语句段：for(i=0;i<N;i++) cout<<a[i];

注意：不允许用数组名 a 进行输入或输出。例如，cin>>a;或 cout<<a;都是错误的。

2．一维数组具体应用举例

（1）求数组中的最大值与最小值。对数组 a[N]求最大值 max 与最小值 min 的程序段为：

```
int i;
float max=min=a[0];
for(i=0;i<N;i++)
{  if(a[i]>max) max=a[i];              //a[i]与  max 相比，大的留下，小的不要
   if(a[i]<min) min=a[i];              //a[i]与  min 相比，小的留下，大的不要
}
```

（2）求数组的和与平均值。对数组 a[N]求和与平均值的程序段为：

```
int i,sum=0;
```

```
for(i=0;i<N;i++)
   sum+=a[i];                          //将 a[i]依次累加到 sum 中
   ave=sum/N;                          //sum 除以 N 得到平均值
```

（3）对数组进行排序。对一维数组进行排序共有三种方法，即冒泡法、选择法与擂台法。

① 冒泡法。升序排序冒泡法的口诀是：相邻两数两两相比，上大下小则交换，共进行 N−1 轮，每轮进行 N−1−i 次，其核心程序段为：

```
for (i=0;i<N−1;i++)
{   for (j=0;j<N−i−1;j++)
       if (a[j]>a[j+1])
       { temp=a[j]; a[j] =a[j+1]; a[j+1]=temp; }
}
```

② 选择法。升序排序选择法的口诀是：首数与后数两两相比，首大后小则交换，共进行 N−1 轮，每轮从 i+1 开始到 N−1 结束，其核心程序段为：

```
for (i=0;i<N−1;i++)
{   for (j=i+1;j<N;j++)
        if (a[i]>a[j])
        { temp=a[i]; a[i] =a[j]; a[j]=temp; }
}
```

③ 擂台法。擂台法源于选择法。在选择法的每次比较后，若首大后小则交换，当数组元素较多时，频繁的交换将延长程序的执行时间，降低程序的执行效率。解决此问题的方法是改每次交换为每轮交换，具体做法如下。

每轮（第 i 轮）设一个擂主 k，初值为 k=i；使擂主值 a[k]与后面的数 a[j]相比，主大后小则换主，即 if (a[k]>a[j]) k=j;。在本轮结束时，判断擂主是否改变，若改变则将擂主值与首数值交换，即 if(k>i) { temp=a[i]; a[i]=a[k]; a[k]=temp; }，共进行 N−1 轮，每轮 j 从 i+1 开始到 N−1 结束，其核心程序段为：

```
for (i=0;i<N−1;i++)
{    int k=i;                           //每轮设置擂主 k，初值 k=i
     for (j=i+1;j<N;j++)
         if (a[k]>a[j]) k=j;            //主大后小则换主
     if (k>i)                           //当擂主改变时，将擂主值与首数值交换
     { temp=a[i]; a[i]=a[k]; a[k]=temp; }
}
```

由于擂台法采用每轮交换的方法，所以其排序效率大大高于冒泡法与选择法的排序效率。

若要进行降序排序，则只需将程序中两数相比的关系运算符由“>”改为“<”即可。

【例 4.10】 已有一按从小到大次序排列的数组，现输入一数，要求按原来排序的规律将它插入数组中。

分析： 先定位，找到插入元素 a[i]，然后向右移，将下标从 i 到 N−2 的元素向后移一个元素的位置，最后插入新元素值。

程序如下：

```
#include <iostream>
#include <iomanip>
#define N 10
using namespace std;
int main()
{   float a[N];
    int i,b,j;
```

```
    cout<<"Input sort array a[9]:"<<endl;
    for(i=0;i<N-1;i++)
        cin>>a[i];
    cout<<"Input number b:";
    cin>>b;
    i=0;
    while (i<N&&a[i]<b) i++;
    for(j=N-1;j>i;j--)   a[j]=a[j-1];
    a[i]=b;
    for(i=0;i<N;i++)
      cout<<setw(6)<<a[i];
    cout<<endl;
    return 0;
}
```

输入：1 2 3 4 5 6 7 8 10

插入：9

输出：1 2 3 4 5 6 7 8 9 10

4.3.2 二维数组应用举例

（1）定义格式：

〔存储类型〕<类型><数组名>[行长][列长]={初值};

（2）元素引用：

<数组名>[行下标表达式] [列下标表达式]

（3）输入/输出：二维数组元素只允许进行单个元素的输入/输出，而不允许用数组名进行输入/输出。

【例 4.11】 某小组有 5 名学生，考了 3 门课程，他们的学号及成绩如表 4.2 所示，试编程求每名学生的总成绩及每门课程的平均成绩，并按表格形式输出每名学生的学号、3 门课程的成绩、总成绩及各门课程的平均成绩。要求用一个 6 行 5 列的数组完成上述操作。将学生成绩按总成绩降序排序后输出。

表 4.2 学生成绩情况表二

学　号	数　学	语　文	外　语	总 成 绩
1001	90	80	85	
1002	70	75	80	
1003	65	70	75	
1004	85	50	60	
1005	80	90	70	
平均成绩				

程序如下：

```
#include <iostream>
#include <iomanip>
#define M 6
#define N 5
using namespace std;
int main()
{    float s[M][N],sum,temp;
```

```
    int i,j,k;
    cout<<"Input data:\n";                                  //输入数据
    for (i=0;i<M-1;i++)                                     //输入 5 名学生的学号与 3 门课程的成绩
    {   for (j=0;j<N-1;j++)
            cin>>s[i][j];
    }
    for (i=0;i<M-1;i++)                                     //处理数据
    {   sum=0.0;
        for (j=1;j<N-1;j++)                                 //计算每名学生的总成绩
            sum=sum+s[i][j];
        s[i][N-1]=sum;                                      //存放每名学生的总成绩
    }
    for (j=1;j<N;j++)                                       //处理数据
    {   sum=0.0;
        for (i=0;i<M-1;i++)                                 //计算每门课程的总成绩
            sum=sum+s[i][j];
        s[M-1][j]=sum/(M-1);                                //计算每门课程的平均成绩
    }
    for (i=0;i<M-2;i++)
    {   k=i;                                                //按总成绩降序排序
        for(j=i+1;j<M-1;j++)
          if (s[k][N-1]<s[j][N-1]) k=j;
        if (k!=i)
          for (j=0;j<N;j++)
          {   temp=s[i][j];s[i][j]=s[k][j];s[k][j]=temp;}
    }
    cout<<setw(5)<<"Num. "<<"Math.Chin.Engl.Sum."<<endl;    //输出数据
    cout<<"-----------------------------\n";
    for (i=0;i<M;i++)
    {   for (j=0;j<N;j++)                                   //输出学号、3 门课程的平均成绩与总成绩
          if(i==M-1 && j==0) cout<<setw(6)<<"平均成绩";
          else cout<<setw(6)<<s[i][j];
        cout<<endl;
    }
    cout<<"-----------------------------\n";
    return 0;
}
```

【例 4.12】　设计一个程序，按下列格式打印杨辉三角形。

```
1
1       1
1       2       1
1       3       3       1
1       4       6       4       1
1       5       10      10      5       1
1       6       15      20      15      6       1
```

方法一：

杨辉三角形第 n 行第 m 个元素值为多项式$(1+x)^n$展开式的系数 $C_n^m=\dfrac{n!}{m!(n-m)!}$。

例如，第 4 行各元素值为：

$C_4^0=\dfrac{4!}{0!(4-0)!}=1$，$C_4^1=\dfrac{4!}{1!(4-1)!}=4$，$C_4^2=\dfrac{4!}{2!(4-2)!}=6$，$C_4^3=\dfrac{4!}{3!(4-3)!}=4$，

$C_4^4=\frac{4!}{4!(4-4)!}=1$。

因此，只需计算组合数 $C_n^m=\frac{n!}{m!(n-m)!}$ 即可得到第 n 行第 m 个元素值。

用二重循环即可打印杨辉三角形，程序如下：

```
#include <iostream>
#include <iomanip>
#define N 7
using namespace std;
int main()
{   int c[N][N],m,n,k,m1,n1,n_m;
    for(n=0; n<N;n++)
      for(m=0; m<=n;m++)
      {   for(m1=1,k=1;k<=m;k++)    m1*=k;
          for(n1=1,k=1;k<=n;k++)    n1*=k;
          for(n_m=1,k=1;k<=n-m;k++)    n_m*=k;
          c[n][m]=n1/(m1*n_m);
      }
     for(n=0;n<N;n++)
     {   for(m=0; m<=n;m++)
           cout<<setw(6)<<c[n][m];
       cout<<endl;
     }
     return 0;
}
```

方法二：

从杨辉三角形各元素值可以看出如下规律。

（1）第一列和对角线元素值都为 1。

（2）其他元素值为 c[n][m]=c[n-1][m-1]+c[n-1][m]。

程序如下：

```
#include <iostream>
#include <iomanip>
#define N 7
using namespace std;
int main()
{   int c[N][N],m,n;
    for(n=0;n<N;n++)
    {   c[n][n]=1;
        c[n][0]=1;
    }
    for(n=2; n<N;n++)
      for(m=1; m<=n-1;m++)
        c[n][m]=c[n-1][m-1]+c[n-1][m];
    for(n=0; n<N;n++)
    {   for(m=0;m<=n;m++)
          cout<<setw(6)<<c[n][m];
      cout<<endl;
    }
     return 0;
}
```

4.3.3 字符数组应用举例

（1）定义格式：

〔存储类型〕 char <数组名>[长度]=“字符串”;

（2）数组元素引用：

<数组名>[下标];

（3）输入/输出：可以整体赋初值、整体输入/输出。例如：

```
# define N 80
char str[N]="String!";
cin.getline(str,60);
cout<<str;
```

上述语句是正确可行的。

（4）字符串处理函数：C++提供的字符串处理函数包含在头文件 string 中。常用的字符串处理函数有：

```
求字符串长度函数：strlen(str);          //返回字符串 str 的长度
字符串复制函数：strcpy(s1,s2);           //s1←s2
字符串连接函数：strcat(s1,s2);           //s1←s1+s2
字符串比较函数：strcmp(s1,s2);           //当 s1>s2 时函数值>0，当 s1=s2 时函数值=0
                                         //当 s1<s2 时函数值<0
```

【例 4.13】 通过键盘输入 3 个字符串，找出其中的最大者。

分析：设一个有 3 行 20 列的二维字符数组 str[3][20]，C++规定二维数组 str[3][20]中的一行可用 str[i]来表示，因此二维数组 str[3][20]的 3 行分别为 str[0]、str[1]和 str[2]，而 str[0]、str[1]和 str[2]又分别是 3 个一维字符数组，它们各有 20 个元素。对于 str[0]、str[1]和 str[2]，可以像一维字符数组一样进行处理，用 cin.getline()函数输入 3 个字符串，找出其中的最大者，并存入一维字符数组 string 中。

程序如下：

```
#include <iostream>
#include <string>
using namespace std;
int main()
{   char str[3][20],string[20];          //定义字符数组 str[3][20]和 string[20]
    int i;
    cout<<"Input three strings:\n";
    for (i=0;i<3;i++)
      cin.getline(str[i],19);            //输入 3 个字符串并存放在 str[3][20]中
    if (strcmp(str[0],str[1])>0)         //找出 str[0]和 str[1]中的较大者，并将其存入 string 中
       strcpy(string,str[0]);
    else
       strcpy(string,str[1]);
    if (strcmp(str[2],string)>0)         //若 str[2]比 string 大，则将 str[2]存入 string 中
       strcpy(string,str[2]);
    cout<<string<<endl;                  //输出 string
    return 0;
}
```

程序运行后显示：

```
Input three strings:
```

we
you
they

输出：

you

本 章 小 结

数组是一系列相同类型数据的有序集合。

1．一维数组

一维数组就是一组有序排列的数据。在定义一维数组的同时，可以为数组元素赋初值。当定义了一个一维数组后，系统就会在内存中开辟一串连续的内存单元，依次存放各个数组元素。

2．二维数组

二维数组就是按表格或矩阵形式排列的数据。在定义二维数组的同时，也可以为数组元素赋初值。当定义了一个二维数组后，系统也会在内存中开辟一串连续的内存单元，并按行顺序依次存放二维数组的各元素。

注意：无论是一维数组还是二维数组，都必须先定义后使用，并且只能对其中的元素进行访问；数组元素的下标从 0 开始，且不能超出范围。

3．字符串和字符数组

（1）字符串。

字符串就是用一对双引号引起来的字符序列。在字符串的末尾有一个结束标志'\0'。

（2）字符数组。

当数组元素为字符型数据时，就形成了字符数组。

在 C++中，将字符串作为字符数组来处理，也就是说，用字符数组来存放字符串。因此，对字符数组来说，可以进行整体赋初值、整体输入/输出。

（3）字符串处理函数。

在 C++的库函数中，提供了字符串处理函数，它们包含在头文件 string 中。常用的字符串处理函数有：求字符串长度函数 strlen()、字符串复制函数 strcpy()、字符串连接函数 strcat()、字符串比较函数 strcmp()、字符串中大写字母变换为小写字母函数 strlur()、字符串中小写字母变换为大写字母函数 struprr()。

注意：在字符串运算符未重载之前，不能用赋值语句将一个字符串或字符数组赋给另一个字符数组，只能用字符串复制函数来处理；两个字符串也不能用关系运算符进行比较，而只能用字符串比较函数来处理。

4．本章重点、难点

重点：一维数组和二维数组的定义、初始化赋值与使用，字符数组的定义、初始化赋值与使用，字符串处理函数。

难点：数组的应用编程，如三种排序方法。

习　　题

4.1　什么是数组？

4.2　一维数组与二维数组在内存中是如何存储的？

4.3　有如下数组定义：

```
int  a[20];
```

指出该数组的数组名、数组元素类型、数组元素个数、第一个数组元素的下标值和最后一个数组元素的下标值。

4.4　写出下列程序的运行结果。

```
#include <iostream>
using namespace std;
int main( )
{   int i,a[10],p[3];
    for (i=0;i<10;i++)
       a[i]=i+1;
    for (i=0;i<3;i++)
       p[i]=a[i*(i+1)];
    for (i=0;i<3;i++)
       cout<<p[i]<<'\t';
    cout<<endl;
    return 0;
}
```

4.5　写出下列程序的运行结果。

```
#include <iostream>
using namespace std;
int main( )
{    static int a[7]={1},i,j;
     for (i=1;i<=6;i++)
         for (j=i;j>0;j--)
            a[j]+=a[j-1];
     for (j=0;j<7;j++)
        cout<<a[j]<<endl;
     return 0;
}
```

4.6　写出下列程序的运行结果。

```
#include <iostream>
using namespace std;
int main( )
{   int a[6][6],i,j;
    for (i=1;i<6;i++)
       for (j=1;j<6;j++)
          a[i][j]=(i/j)*(j/i);
    for (i=1;i<6;i++)
    {   for (j=1;j<6;j++)
           cout<<a[i][j]<<'\t';
        cout<<endl;
    }
    return 0;
}
```

4.7　写出下列程序的运行结果。

```
#include <iostream>
#include <string>
```

```
using namespace std;
int main( )
{    char str[80];
     int i,j,k;
     cout<<"Input string:";
     cin>>str;
     for(i=0,j=strlen(str)-1;i<j;i++,j--)
     {   k=str[i];
         str[i]=str[j];
         str[j]=k;
     }
     cout<<str<<endl;
     return 0;
}
```

运行时输入：

abcdef（“Enter”键）

4.8　某班有 30 名学生，进行了数学考试，编写程序将考试成绩输入一维数组，并求数学的平均成绩和不及格学生的人数。

4.9　设有一个数列，它的前四项为 0、0、2、5，以后每项分别是其前四项之和，编程求此数列的前 20 项。用一维数组完成此操作。

4.10　某班有 30 名学生，进行了数学考试，编写程序将考试成绩输入一维数组，并将数学成绩用冒泡法、选择法与擂台法三种排序算法按由高到低的顺序排序后输出。

4.11　已有一按从大到小次序排列的数组，现输入一数，要求按原来排序的规律将它插入数组中。

4.12　定义一个二维数组，用编程的方法形成如下矩阵（数据不能通过键盘输入），并按下列格式输出矩阵。

$$A=\begin{pmatrix}1&2&3&4&5\\1&1&2&3&4\\1&1&1&2&3\\1&1&1&1&2\\1&1&1&1&1\end{pmatrix}$$

4.13　设计一个程序，打印如下格式的杨辉三角形。

```
                    1
                1       1
            1       2       1
        1       3       3       1
    1       4       6       4       1
1       5       10      10      5       1
    1       6       15      20      15      6       1
```

4.14　输入一个字符串，求出字符串的长度（不能用 strlen()函数），并输出字符串及其长度。

4.15　输入一行字符，分别统计其中英文字母、空格、数字字符和其他字符的个数。

4.16　编写一程序，对通过键盘输入的两个字符串进行比较，然后输出两个字符串中第一个不同字符的 ASCII 码之差。例如，输入的两个字符串分别为 abcdef 和 abceef，则第一个不同字符为 d 和 e，输出为-1。

4.17　通过键盘输入三个字符串，将其合并成一个字符串，并求合并后字符串的长度。

4.18　已知某运动会上男子百米赛跑决赛成绩。要求编写程序，按成绩排序，并按名次输出排序结果，输出包括名次、运动员号码和成绩三项内容。

4.19　某小组有 5 名学生，考了 3 门课程，他们的学号及成绩如表 4.3 所示，试编程求每名学生的总成绩及每门课程的最高分，并按表格形式输出每名学生的学号、3 门课程的成绩、总成绩及各门课程的最高分。要求用一个 6 行 5 列的数组完成上述操作。

表 4.3　学生成绩情况表三

学　号	数　学	语　文	外　语	总 成 绩
1001	90	80	85	
1002	70	75	80	
1003	65	70	75	
1004	85	50	60	
1005	80	90	70	
最高分				

4.20　将习题 4.19 中的学生成绩用擂台法按总成绩升序排序后输出，并输出每门课程不及格学生的学号、课程名称及成绩。

4.21　设 ***A*** 为 *m* 行 *n* 列的矩阵，***B*** 为 *n* 行 *k* 列的矩阵，***C*** 为 *m* 行 *k* 列的矩阵。设计矩阵乘法程序，要求完成 ***C*=*A*×*B*** 的操作。*m*、*n* 与 *k* 用 define 定义为常量，其值由用户自定义。

实　验　A

1．实验目的

（1）掌握一维数组的定义、初始化赋值、数组元素的引用方法。

（2）掌握二维数组的定义、初始化赋值、数组元素的引用方法。

（3）学会求数组元素中的最大值、最小值、平均值的方法。

（4）学会对数组元素进行排序的两种编程方法。

2．实验内容

（1）某班第 1 组有 10 名学生，进行了 C++考试，编写程序将考试成绩输入一维数组，并求出 C++的平均成绩及优（90～100）、良（80～89）、中（70～79）、及格（60～69）与不及格（0～59）学生的人数。

实验数据：90、85、80、75、70、65、60、55、50、45。

（2）某班第 1 组有 10 名学生，进行了 C++考试，编写程序将考试成绩输入一维数组，并将 C++成绩用冒泡法、选择法两种排序算法按由低到高的顺序排序后输出。

实验数据：90、85、80、75、70、65、60、55、50、45。

（3）输入一个 5 行 5 列的二维数组，编程实现：

① 求出其中的最大值和最小值及其对应的行列位置；

② 求出主副对角线上各元素之和。

实验数据：

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \\ 5 & 6 & 7 & 8 & 9 \end{pmatrix}$$

（4）设 ***A***、***B***、***C*** 均为 *m* 行 *n* 列的矩阵。设计矩阵加法程序，要求完成 ***C***=***A***+***B*** 的操作。并输出 ***C*** 的元素值。*m* 与 *n* 用 define 定义为常量，取值 3、3。***A***、***B*** 矩阵的元素值如下。

$$\boldsymbol{A}=\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad \boldsymbol{B}=\begin{pmatrix} 3 & 2 & 1 \\ 6 & 5 & 4 \\ 9 & 8 & 7 \end{pmatrix}$$

3．实验要求

（1）编写实验程序。

（2）在 VC++运行环境中输入源程序。

（3）编译运行源程序。

（4）输入测试数据进行程序测试。

（5）写出运行结果。

实　验　B

1．实验目的

（1）初步掌握有序数组的查找、增加、删除的编程方法。

（2）初步掌握字符数组的定义、赋初值与字符串处理函数的使用方法。

（3）初步掌握字符串复制、连接、测长等程序的编写方法。

（4）学会打印杨辉三角形的编程方法。

（5）学会对二维数据表进行排序的编程方法。

2．实验内容

（1）已有一按从小到大次序排列的数组，现输入一数，要求用折半查找法（学生自己查阅资料学习）找出该数在数组中的位置。

实验数据：10、12、14、16、18、20、22、24、26、28。

输入数：16。

（2）编写程序，实现 str=str1+str2 的操作，此处运算符“+”表示将 str1、str2 两个字符串连接成一个字符串 str。用键盘将两个字符串输入字符数组 str1 与 str2 中，连接后的字符串存放在字符数组 str 中，并输出连接后的字符串 str。

① 用 C++提供的字符串处理函数完成上述要求。

② 不用 C++提供的字符串处理函数完成上述要求。

实验数据：a　b　c　d　e

f g h i j

（3）设计一个程序，按习题 4.13 的要求打印杨辉三角形。

（4）学生成绩如表 4.4 所示，编程求每名学生的最高分和每门课程的平均分，并按表格形式输出。

表 4.4 学生成绩情况表四

学 号	数 学	语 文	外 语	最 高 分
1001	90	80	85	
1002	70	75	80	
1003	65	70	75	
1004	85	50	60	
1005	80	90	70	
平均分				

3．实验要求

（1）编写实验程序。

（2）在 VC++运行环境中输入源程序。

（3）编译运行源程序。

（4）输入测试数据进行程序测试。

（5）写出运行结果。

说明：在后面章节的实验中，因为实验要求基本一样，所以实验要求不再一一列出。