清华大学出版社
TSINGHUA UNIVERSITY PRESS

中国高等院校计算机基础教育课程体系规划教材

谭浩强 编著

C++程序设计

总 目 录

第1篇 基 本 知 识

第1章 C++的初步知识

第2章 数据类型与表达式

第2篇 面向过程的程序设计

第3章 程序设计初步

第4章 函数与预处理

第5章 数组

第6章 指针

第7章 自定义数据类型

第3篇 基于对象的程序设计

第8章 类和对象

第9章 关于类和对象的进一步讨论

第10章 运算符重载

第4篇 面向对象的程序设计

第11章 继承与派生

第12章 多态性与虚函数

第13章 输入输出流

第14章 C++工具

第1篇

基 本 知 识

第1章 C++的初步知识
第2章 数据类型与表达式

第1章 C++的初步知识

*1.1 从C到C++
*1.2 最简单的C++程序
1.3 C++程序的构成和书写形式
1.4 C++程序的编写和实现
1.5 关于C++上机实践

*1.1 从C到C++

计算机诞生初期，人们要使用计算机必须用机器语言或汇编语言编写程序。世界上第一种计算机高级语言诞生于**1954**年，它是**FORTRAN**语言。先后出现了多种计算机高级语言。其中使用最广泛、影响最大的当推**BASIC**语言和**C**语言。

BASIC语言是**1964**年在**FORTRAN**语言的基础上简化而成的，它是为初学者设计的小型高级语言。

C语言是**1972**年由美国贝尔实验室的**D.M.Ritchie**研制成功的。它不是为初学者设计的，而是为计算机专业人员设计的。大多数系统软件和许多应用软件都是用**C**语言编写的。

但是随着软件规模的增大，用C语言编写程序渐渐显得有些吃力了。

C++是由AT&T Bell(贝尔)实验室的Bjarne Stroustrup博士及其同事于20世纪80年代初在C语言的基础上开发成功的。C++保留了C语言原有的所有优点，增加了面向对象的机制。

C++是由C发展而来的，与C兼容。用C语言写的程序基本上可以不加修改地用于C++。从C++的名字可以看出它是C的超集。C++既可用于面向过程的结构化程序设计，又可用于面向对象的程序设计，是一种功能强大的混合型的程序设计语言。

C++对C的“增强”，表现在两个方面：

(1) 在原来面向过程的机制基础上，对C语言的功能做了不少扩充。

(2) 增加了面向对象的机制。

面向对象程序设计，是针对开发较大规模的程序而提出来的，目的是提高软件开发的效率。

不要把面向对象和面向过程对立起来，面向对象和面向过程不是矛盾的，而是各有用途、互为补充的。

学习C++，既要会利用C++进行面向过程的结构化程序设计，也要会利用C++进行面向对象的程序设计。本书既介绍C++在面向过程程序设计中的应用，也介绍C++在面向对象程序设计中的应用。

*1.2 最简单的C++程序

例1.1 输出一行字符：“This is a C++ program.”。

程序如下：

```
#include <iostream>                //包含头文件iostream
using namespace std;               //使用命名空间std
int main( )
{
cout<<"This is a C++ program.";
return 0;
}
```

在运行时会在屏幕上输出以下一行信息：

This is a C++ program.

用**main**代表“主函数”的名字。每一个**C++**程序都必须有一个 **main** 函数。**main**前面的**int**的作用是声明函数的类型为整型。程序第**6**行的作用是向操作系统返回一个零值。如果程序不能正常执行，则会自动向操作系统返回一个非零值，一般为**-1**。

函数体是由大括号**{ }**括起来的。本例中主函数内只有一个以**cout**开头的语句。注意**C++**所有语句最后都应当有一个分号。

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

#include <iostream>是一个“包含命令”，它的作用是将文件**iostream**的内容包含到该命令所在的程序文件中，代替该命令行。文件**iostream**的作用是向程序提供输入或输出时所需要的一些信息。

iostream是**i-o-stream 3**个词的组合，从它的形式就可以知道它代表“输入输出流”的意思，由于这类文件都放在程序单元的开头，所以称为“头文件”**(head file)**。在程序进行编译时，先对所有的预处理命令进行处理，将头文件的具体内容代替**#include**命令行，然后再对该程序单元进行整体编译。

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

在初学**C++**时，对本程序中的第**1,2**行可以不必深究，只需知道： 如果程序有输入或输出时，必须使用**“#include <iostream>”**命令以提供必要的信息，同时要用**“using namespace std;”** ，使程序能够使用这些信息，否则程序编译时将出错。

例**1.2** 求**a**和**b**两个数之和。

可以写出以下程序：

```
// 求两数之和                    (本行是注释行)
#include <iostream>              //预处理命令
using namespace std;             //使用命名空间std
int main( )                      //主函数首部
{                                //函数体开始
    int a,b,sum;                 //定义变量
    cin>>a>>b;                   //输入语句
    sum=a+b;                     //赋值语句
cout<<"a+b="<<sum<<endl;         //输出语句
return 0;                        //如程序正常结束，向操作系统返回一个零值
}                                //函数结束
```

本程序的作用是求两个整数**a**和**b**之和**sum**。第**1**行“//求两数之和”是一个注释行，**C++**规定在一行中如果出现“//”，则从它开始到本行末尾之间的全部内容都作为注释。

如果在运行时从键盘输入

123 456↙

则输出为

a+b=579

例1.3 给两个数x和y，求两数中的大者。

在本例中包含两个函数。

```
#include <iostream>                    //预处理命令
using namespace std;
int max(int x,int y)        //定义max函数，函数值为整型，形式参数x，y为整型
{                                  //max函数体开始
    int z;                          //变量声明，定义本函数中用到的变量z为整型
    if(x>y) z=x;                    //if语句，如果x>y，则将x的值赋给z
else z=y;                          //否则，将y的值赋给z
   return(z);                       //将z的值返回，通过max带回调用处
}                                  //max函数结束

int main( )                        //主函数
```

```
{                                   //主函数体开始
  int a,b,m;                        //变量声明
     cin>>a>>b;                     //输入变量a和b的值
     m=max(a,b);                    //调用max函数，将得到的值赋给m
cout<<"max="<<m<<'\\n';             //输出大数m的值
return 0;                           //如程序正常结束，向操作系统返回一个零值
}                                   //主函数结束
```

本程序包括两个函数:主函数**main**和被调用的函数**max**。

程序运行情况如下:

18 25 ↙ **(输入18和25给a和b)**

max=25 **(输出m的值)**

注意输入的两个数据间用一个或多个空格间隔，不能以逗号或其他符号间隔。

在上面的程序中，**max**函数出现在**main**函数之前，因此在**main**函数中调用**max**函数时，编译系统能识别**max**是已定义的函数名。如果把两个函数的位置对换一下，即先写**main**函数，后写**max**函数，这时在编译**main**函数遇到**max**时，编译系统无法知道**max**代表什么含义，因而无法编译，按出错处理。

为了解决这个问题，在主函数中需要对被调用函数作声明。上面的程序可以改写如下：

```
#include <iostream>
using namespace std;
int main( )
{ int max(int x,int y);        //对max函数作声明
  int a,b,c;
  cin>>a>>b;
  c=max(a,b);                  //调用max函数
```

```
  cout<<"max="<<c<<endl;
  return 0;
}

int max(int x,int y)          //定义max函数
{ int z;
  if(x>y) z=x;
  else z=y;
  return(z);
}
```

只要在被调用函数的首部的末尾加一个分号，就成为对该函数的函数声明。函数声明的位置应当在函数调用之前。

下面举一个包含类(class)和对象(object)的C++程序，目的是使读者初步了解C++是怎样体现面向对象程序设计方法的。

例1.4 包含类的C++程序。

```
#include <iostream>                    //预处理命令
using namespace std;
class Student                          //声明一个类，类名为Student
{private:                              //以下为类中的私有部分
int num;                               //私有变量num
int score;                             //私有变量score
public:                                //以下为类中的公用部分
void setdata( )                        //定义公用函数setdata
{cin>>num;                             //输入num的值
cin>>score;                            //输入score的值
}
```

```
        void display( )                  // 定义公用函数display
       {cout<<"num="<<num<<endl;       // 输出num的值
        cout<<"score="<<score<<endl;   //输出score的值
 };
};                                  //类的声明结束
Student stud1,stud2;          //定义stud1和stud2为Student类的变量，称为对象
int main( )                         //主函数首部
{stud1.setdata( );                  // 调用对象stud1的setdata函数
    stud2.setdata( );               // 调用对象stud2的setdata函数
    stud1.display( );               // 调用对象stud1的display函数
stud2.display( );                   // 调用对象stud2的display函数
return 0;
}
```

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

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

具有“类”类型特征的变量称为“对象”(object)。

程序中第18~24行是主函数。

程序运行情况如下：

1001 98.5 ↙ (输入学生1的学号和成绩)

1002 76.5 ↙ (输入学生2的学号和成绩)

num=1001 (输出学生1的学号)

score=98.5 (输出学生1的成绩)

num=1002 (输出学生2的学号)

score=76.5 (输出学生2的成绩)

1.3 C++程序的构成和书写形式

C++程序的结构和书写格式归纳如下：

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

(2) 在一个程序单位中，可以包括以下几个部分：

① 预处理命令。上节4个程序中都包括#include命令。

② 全局声明部分(在函数外的声明部分)。在这部分中包括对用户自己定义的数据类型的声明和程序中所用到的变量的定义。

③ 函数。函数是实现操作的部分，因此函数是程序中必须有的和最基本的组成部分。每一个程序必须包括一个或多个函数，其中必须有一个(而且只能有一个)主函数(**main**函数)。

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

(3) 一个函数由两部分组成**:**

① 函数首部，即函数的第一行。包括函数名、函数类型、函数属性、函数参数(形参)名、参数类型。

一个函数名后面必须跟一对圆括号， 函数参数可以缺省，如**int main( )**。

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

函数体一般包括:

•局部声明部分 (在函数内的声明部分)。包括对本函数中所用到的类型、函数的声明和变量的定义。

对数据的声明既可以放在函数之外(其作用范围是全局的)，也可以放在函数内(其作用范围是局部的，只在本函数内有效)。

•执行部分。由若干个执行语句组成，用来进行有关的操作，以实现函数的功能。

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

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

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

(7)C++程序书写格式自由，一行内可以写几个语句，一个语句可以分写在多行上。**C++**程序没有行号，也不像**FORTRAN**或**COBOL**那样严格规定书写格式(语句必须从某一列开始书写)。

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

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

1.4 C++程序的编写和实现

一个程序从编写到最后得到运行结果要经历以下一些步骤。

1. 用C++语言编写程序

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

2. 对源程序进行编译

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

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

3. 将目标文件连接

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

4. 运行程序

运行最终形成的可执行的二进制文件(.exe文件)，得到运行结果。

5. 分析运行结果

如果运行结果不正确，应检查程序或算法是否有问题。

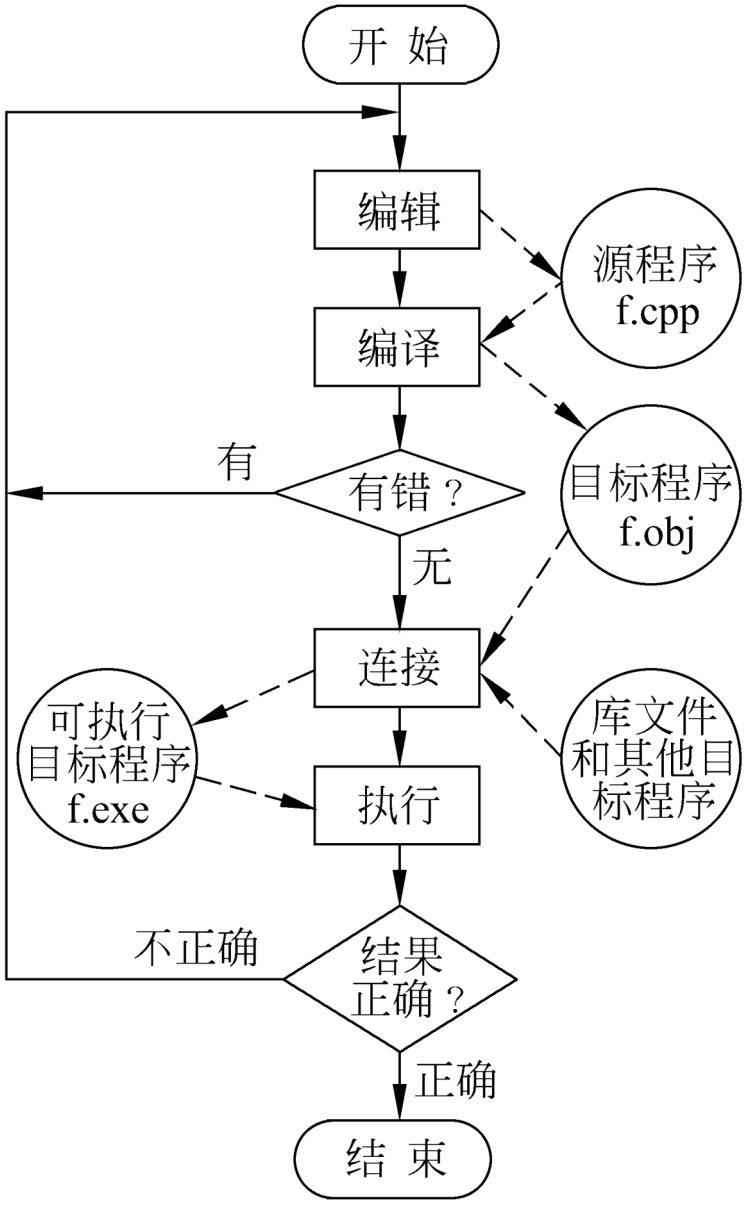
开 始
编辑
源程序
f.cpp
编译
有
有错？
目标程序
f.obj
无
连接
可执行
目标程序
f.exe
库文件
和其他目
标程序
执行
不正确
结果
正确？
正确
结 束

图1.1

1.5 关于C++上机实践

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

读者可以使用不同的C++编译系统，在不同的环境下编译和运行一个C++程序。但是需要强调的是，我们学习的是C++程序设计，应当掌握的是标准C++，而不应该只了解某一种“方言化”的C++。不应当只会使用一种C++编译系统，只能在一种环境下工作，而应当能在不同的C++环境下运行自己的程序，并且了解不同的C++编译系统的特点和使用方法，在需要时能将自己的程序方便地移植到不同的平台上。

在本书的参考书《**C++**程序设计题解与上机指导》一书中简单介绍了在**Visual C++ 6.0**和**GCC**两种典型的环境下运行**C++**程序的方法。

请读者选择一种(如能做到两种更好) **C++**编译系统，在该环境下输入和运行习题中的程序，掌握上机的方法和步骤。