编程语言基础——C++

詹金权 ⊙主编

北京理工大学出版社
G INSTITUTE OF TECHNOLOGY PRESS

编程语言基础

——C++

主　编　廖金权
副主编　封宏观

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

内容提要

本书以面向对象的程序设计方法贯穿始终，每一章都首先阐述面向对象的程序设计思想和方法，然后引出必要的语法知识，在讲解语法时着重从程序设计方法学的角度讲述基本意义和用途，力求使读者在掌握 C++ 语言的同时，能够对现实世界中较简单的问题及其解决方法用计算机语言进行描述。针对初学者和自学者的特点，书中以结合实例讲解基本概念和方法为主，力求将复杂的概念用简洁浅显的语言来描述，做到深入浅出。

全书共 11 模块，可作为高等院校程序设计专业的入门教材。

图书在版编目（CIP）数据

编程语言基础：C++ / 廖金权主编 . —北京：北京理工大学出版社，2016.3

ISBN 978-7-5682-1947-1

Ⅰ . ①编…　Ⅱ . ①廖…　Ⅲ . ① C 语言 - 程序设计 - 高等学校 - 教材　Ⅳ . ① TP312

中国版本图书馆 CIP 数据核字（2016）第 043695 号

出版发行 / 北京理工大学出版社有限责任公司

社　　址 / 北京市海淀区中关村南大街 5 号

邮　　编 /100081

电　　话 /（010）68914775（总编室）

82562903（教材售后服务热线）

68948351（其他图书服务热线）

网　　址 / http：//www.bitpress.com.cn

经　　销 / 全国各地新华书店

印　　刷 / 北京通县华龙印刷厂

开　　本 / 787 毫米 ×1092 毫米　1/16

印　　张 / 13

字　　数 / 337 千字

版　　次 / 2016 年 3 月第 1 版第 1 次印刷

定　　价 / 43.00 元

责任校对 / 陈玉梅

责任印制 / 边心超

PREFACE 前言

C++ 是一门高效实用的程序设计语言，它既可进行过程化程序设计，也可进行面向对象程序设计。这种面向对象程序设计语言已经成为了程序员最广泛使用的工具。本课程是一门计算机应用及相关专业的重要的专业基础课，通过学习使学生掌握有关 C++ 语言的基本概念、基本语法和编程方法，理解 C++ 语言面向对象的重要特征，促使学生理论联系实际，能够灵活应用所学的理论知识进行程序开发，增强学生的实践动手技能，并能够提高学生独立分析问题和解决问题的能力。

本书以面向对象的程序设计方法贯穿始终，每一章都首先阐述面向对象的程序设计思想和方法，然后引出必要的语法知识，在讲解语法时着重从程序设计方法学的角度讲述基本意义和用途，力求使读者在掌握 C++ 语言的同时，能够对现实世界中较简单的问题及其解决方法用计算机语言进行描述。针对初学者和自学者的特点，书中以结合实例讲解基本概念和方法为主，力求将复杂的概念用简洁浅显的语言来描述，做到深入浅出。

全书共 11 模块：

模块 1 为 C++ 概述，主要介绍 C++ 的简单概念、开发过程、结构与组成及程序风格。

模块 2 为数据类型，介绍 C++ 的基本数据类型。

模块 3 为运算符和表达式，介绍基本运算符和基本表达式。

模块 4 为语句，主要介绍了 C++ 语言中的基本语句。

模块 5 为函数，主要介绍了 C++ 语言中常用函数。

模块 6 为指针，主要讲述了指针的定义及使用。

模块 7 为类与对象，主要介绍了类与对象的定义和使用，包括了构造函数和析构函数等函数的相关介绍。

模块 8 为继承与派生，主要讲述了继承与派生的基础知识。

模块 9 为多态与虚函数，讨论了类之间派生中的动态继承问题。

模块 10 为运算符重载，讨论了运算符的重载问题。

模块 11 为 C++ 流与文件，简述了文件的相关知识。

本书语言表达严谨、流畅、实例丰富，同时配有大量习题，适合高等院校程序设计课程的入门教材。

本书由廖金权任主编，封宏观任副主编。廖金权负责模块 1~6 的编写，封宏观负责模块 7~11 的编写，全书由廖金权统稿。

由于编者水平所限，难免有疏漏之处，希望各位读者和专业人士批评指正。

编　者

目 录

模块1　C++概述

学习目标

①了解C++程序设计语言的产生和发展历程。

②了解C++程序相关的基本概念。

③掌握C++程序的开发过程。

④掌握C++程序的结构与组成。

⑤了解C++程序的风格。

本章导读

C++语言是当今最流行的一种计算机程序设计语言，为了更好地适应现代信息社会的发展，我们很有必要对C++程序设计语言进行深入的了解和学习。

任务1　C++的产生和发展

C++程序设计语言由C语言发展而来。C语言最早是由贝尔实验室的Dennis Ritchie在B语言的基础上开发出来的，并于1972年在一台DEC PDP－11计算机上首次实现。C语言产生以后，最早在UNIX操作系统上使用，并且迅速被人们接受而得到了广泛的应用。到了20世纪80年代，C语言已经风靡全球，成为一种应用最为广泛的程序设计语言。但C语言在盛行的时候也显现出了自己的局限性，突出表现在以下几个方面。

(1)C语言类型检查机制较弱，这使得程序中的一些错误不能在编译时被发现。

(2)C语言本身几乎没有支持代码的重用机制，这使得各个程序的代码很难为其他程序所用。

(3)C语言不适合开发大型的程序，当程序达到一定的规模时，程序员很难控制程序的复杂性。

为了弥补C语言的不足之处，1980年贝尔实验室的Bjarne Stroustrup博士开始对C语言

进行改编。一开始 C++ 是作为 C 语言的增强版出现的,从给 C 语言增加类开始,不断增加新特性。虚函数(virtual function)、运算符重载(operator overloading)、多重继承(multiple inheritance)、模板(template)、异常(exception)、RTTI、名字空间(namespace)逐渐被加入标准,1983 年正式命名为 C++。1998 年国际标准组织(ISO)颁布了 C++ 程序设计语言的国际标准 ISO/IEC 14882 - 1998。C++ 是具有国际标准的编程语言,通常称作 ANSI/ISO C++。1998 年是 C++ 标准委员会成立的第一年,以后每 5 年视实际需要更新一次标准。C++ 不仅继承了 C 语言的原有精髓,并且增加了对开发大型软件非常有效的面向对象的机制,弥补了 C 语言不支持代码重用的不足,成为一种既可表现过程模型,又可表现对象模型的优秀的程序设计语言之一。目前 C++ 仍在不断的发展当中。

另外,就目前学习 C++ 而言,可以认为它是一门独立的语言;它并不依赖 C 语言,我们可以完全不学 C 语言,而直接学习 C++。根据《C++ 编程思想》(Thinking in C++)一书所评述的,C++ 与 C 的效率往往相差 ±5%。所以有人认为在大多数场合 C++ 完全可以取代 C 语言(然而在单片机等需要谨慎,利用空间、直接操作硬件的地方还是要使用 C 语言)。

任务 2　C++ 程序设计语言的特点

C++ 继承了 C 语言的所有特点,包括语言简洁、紧凑;使用方便、灵活;拥有丰富的运算符;生成的目标代码质量高,程序执行效率高;可移植性好等。

同时,C++ 对 C 语言进行了一定的改进。如引入 const 常量和内联函数,取代宏定义;引入 refrence(引用)概念等;支持面向过程和面向对象的方法。在 C++ 环境下既可以进行面向对象的程序设计,也可以进行面向过程的程序设计。

C++ 现在得到了越来越广泛的应用,其特点主要有以下几个方面。

(1)兼容 C 语言。这主要表现在大部分 C 程序不需修改即可在 C++ 的编译环境下运行,用 C 语言编写的许多库函数和应用软件都可用于 C++ 中。

(2)用 C++ 编写的程序可读性更好,代码结构更合理,可直接在程序中映射问题空间的结构。

(3)生成的代码质量高,运行效率仅比汇编语言代码段慢 10% ~ 20%。

(4)从开发时间、费用到形成的软件的可重用性、可扩充性、可维护性和可靠性等方面有了很大的提高,使得大中型的程序开发项目变得容易得多。

(5)C++ 是面向对象的程序设计语言,可方便地构造出模拟现实问题的实体和操作。

任务3　C++语言的基本概念

1.3.1　程序

为了对C++程序结构有一个初步了解，下面通过一个简单例子来了解C++程序。

【例1-1】一个简单的C++程序。

```
#include <iostream.h>
void main(void)
{
/*输出 This is my first C++ program. */
cout<<"This is my first C++ program.\n";
}
```

这个程序运行后在屏幕上输出字符串"This is my first C++ program."。我们可以发现以下几方面的内容。

(1)C++源程序文件的扩展名为.cpp。

(2)C++注释不但可以使用符号"/*"和"*/"，表示符号"/*"和"*/"之间的内容都是注释；而且还可以使用一个双斜线"//"，表示"//"之后的本行内容是注释，注释在按回车键后自动结束。

(3)C++程序一般包含的是标准输入、输出流的头文件iostream.h，输入、输出可以通过使用输入、输出流对象(如cin、cout)来完成。

一个结构化的C++程序可以由多个函数构成，函数与函数之间是相对独立的，它们的定义是并行的，一个函数可以被其他函数调用。每个程序都从主函数main()开始执行，从主函数返回时结束执行。同其他高级编程语言一样，C++程序可以由多个源文件构成，它们分别被编译成目标代码，然后连接成一个可执行程序。

1.3.2　对象和类

从面向对象的角度来说，类是对某一类对象的抽象，而对象是类的具体实例；从程序设计语言的角度来说，类是一种复杂的自定义数据类型，对象是属于这种数据类型的变量。C++引入了类这种抽象数据类型，实现了对对象的抽象和封装。

C++类的结构比较复杂，可以将其看做是一种既包含数据又包含函数的数据类型。显然，描述不同编程对象的类应有不同的内部结构，在使用类来声明对象前应先定义其结构。

C++定义类的基本形式如下：

```
class <类名>
```

```
};
private:
<私有数据成员和私有成员函数的声明列表>;
public:
<公有数据成员和公有成员函数的声明列表>;
protected:
<保护数据成员和保护成员函数的声明列表>;
};
```

其中,class 是定义类的关键字,<类名>是用户自定义的标识符,花括号括起来的部分称为类体,它包括所有数据成员和成员函数的声明。数据成员又称成员变量(member variable),成员函数又称为方法(method)。关键字 private、public 和 protected 称为访问权限控制符,用来设置数据成员和成员函数的访问权限属性。

private 属性表示数据成员或成员函数是类的私有成员,这类成员只允许被本类的成员函数访问或调用;public 属性表示数据成员或成员函数是公有成员,这类成员允许被本类或其他类的成员函数(通过对象)访问或调用;protected 属性表示数据成员或成员函数是保护成员,这类成员允许被本类的成员函数和派生类的成员函数访问或调用。

在类的外部通过对象只能访问所属类的公有成员,而私有成员只能在类的成员函数中被访问。一般而言,数据成员定义为 private 属性;成员函数定义为 public 属性,作为类的外部接口。在声明成员时如果省略了访问权限控制符,则其属性默认为 private。

【例 1 -2】定义类 Time(时间)。

```
class Time{
private:
int hour;
int minute;
int second;
public:
void setTime(int,int,int);
void showTime();
};
```

类 Time 有 hour、minute 和 second 三个私有数据成员,它们只能在类的成员函数中被访问或赋值;类 Time 有两个公有成员函数,可在类的外部被调用,它们可视为访问类 Time 的外部接口,但必须通过一个对象实现其调用。

利用 C++类进行面向对象的程序设计,以上声明类的成员只是完成了任务的第一步,最重要的任务是实现定义的类。类的实现实质上是类的成员函数的实现,即定义类的成员函数。成员函数的定义形式与一般函数的定义基本相同,但如果在类的外部定义成员函数,必须在成员函数名前加上类名和作用域限定符“::”。

【例1-3】类Time的实现。

```
void Time :: setTime(int h,int m,int s)
{
hour = (h >= 0&&h < 24)? h:0;          //设置时间
minute = (m >= 0&&m < 60)? m:0;
second = (s >= 0&&s < 60)? s:0;
}
void Time :: showTime()
{
cout≪hour?':'≪minute?':'≪second≪endl;          //输出时间
}
```

由于不允许在类的外部访问或修改类Time的私有数据成员hour、minute和second，所以类Time增加两个公有成员函数setTime()和showTime()，以供在类的外部设置或显示类Time的私有数据成员hour、minute和second。

一般将类的定义放在头文件(*.h)中，类的实现放在源文件(*.cpp)中，而main主函数可以放在另一个源文件中。在源文件中用#include编译预处理指令包含头文件。

对象是类的一个实例，定义并实现了类后，就可以利用类来声明对象，其形式与普通变量的声明类似。例如，以下用类Time声明了对象t1、today和对象的指针ptl：

```
Time t1,today;              //声明对象 t1,today
Time * ptl = &t1;           //声明指向对象 t1 的指针 ptl
```

声明对象后，就可以像引用结构变量一样，利用成员运算符“.”或指向运算符“->”引用对象的公有成员，但注意不能引用对象的非公有成员。

1.3.3　常量和变量

在程序中使用的数据有常量和变量两种类型，常量的值是始终不变的，而变量的值是可以被改变的。常量和变量的主要区别在于：常量不占内存空间，不能为常量赋值；而变量需要占内存空间，可以给变量赋不同的值。不管常量还是变量，程序中使用的每一个数据都属于一种特定的数据类型。

常量的书写方式规定了该常量的数据类型，而变量在使用之前必须先进行变量的声明(declaration)，以便编译程序为变量分配合适的内存空间，并可以给变量赋一个初值(即初始化)。变量声明语句的一般形式如下：

<数据类型><变量名1>[=<初始值1>],<变量名2>[=<初始值2>],…;

例如：

```
float x,y,z;                //声明 3 个浮点型变量 x、y、z
int total,num = 2008;       //声明整型变量 total、num,并对 num 初始化
```

变量声明语句定义了变量的名称和数据类型，在程序中通过变量名存取其中的数据，数

据类型说明了变量所占用空间的大小和可以进行的运算。

1.3.4 函数

函数是 C++ 程序的构成基础。C++ 程序都是由一个个函数所组成的。一个 C++ 程序无论多么复杂,规模有多么大,程序的设计最终都落实到每个函数的设计与编写上。

在 C++ 中,函数是结构化设计的“自顶向下、逐步求精”思想的具体体现。函数是程序模块划分的基本单位,程序员可将一个复杂的程序分解为若干个相对独立且功能单一的子程序即函数进行设计。

C++ 函数有三种:主函数(即 main( )函数)、C++ 提供的库函数和自定义函数。C++ 系统函数库提供了几百个完成不同功能的函数,编程时可以直接拿来使用。合理地编写用户自定义函数,可以简化程序模块的结构,便于阅读和调试,是结构化程序设计方法的主要思想之一。

1.3.5 输入和输出

程序从外部设备获得数据称为输入(input),反之,将程序中的数据送到外部设备,如屏幕、打印机等,称为程序的输出(output)。

大多数 C++ 程序都要有数据的输入/输出。在此,先简单地介绍 C++ 程序中最常用的标准输入/输出语句。

1. 标准输出语句

```
cout≪"let\'s learn to write a C++ Program. ";
cout≪endl;
cout≪" chicken hen cock"≪endl;
cout≪" "≪chicken≪" "≪hen≪" "≪cock≪endl;
```

上面是四条标准输出语句。cout 是标准输出文件(默认为屏幕)的名字,“≪”称为插入运算符。它们完成向显示器屏幕输出指定的内容。

第一条语句输出一个字符串常量,即在屏幕上显示包括空格和标点符号在内的一个字符串。

第二条语句中 endl 有特定的含义,它表示有“回车换行”,即下一次输出数据从下一行左端开始。

第三条语句则连续向屏幕输出两项内容,先是字符串常量“chicken hen cock”,然后换行。它等价于下面两条语句:

```
cout≪" chicken hen cock";
cout≪endl;
```

第四条语句等价于七条输出语句:

```
cout≪" ";
```

```
cout≪chicken;
cout≪" ";
cout≪hen;
cout≪" ";
cout≪cock;
cout≪endl;
```

其中,chicken、hen、cock 不是字符串(它们没有引号),而是变量,输出变量当前的值。

2. 标准输入语句

下面的程序使用了标准输入语句 cin≫myage;

```
#include < iostream. h >
void main( void)
{
int myage;
cout≪" My age is" ;
cin≫myage;                    //输入年龄(一个整数)
cout≪myage≪endl;
}
```

这时 C++ 程序允许从键盘输入一个整数,例如 21,结果在屏幕上显示:

My age is 21

cin 与 cout 类似,是标准输入文件(指键盘)的名字。“≫”称为提取运算符。myage 是一个程序员定义的整型变量名,变量不同于常量,在程序运行中其值可以改变。该语句的作用是把从键盘输入的值赋给变量 myage。

1.3.6　预处理命令#include

编译预处理是指在对源程序进行编译之前,编译预处理程序(precompiler)根据源程序中的编译预处理指令对源程序预处理。编译预处理程序是编译器的一部分,由编译器自动启动。与一般的 C++ 语句不同的是,编译预处理指令以符号“#”开头,结尾不使用分号“;”。编译预处理指令改善了编程环境,提高了编程效率。

C++ 提供的预处理有宏定义命令、文件包含命令和条件编译命令 3 种。我们着重介绍#include 文件包含指令。

#include 文件包含指令是指将一个源文件嵌入到当前源文件中该指令处。#include 指令有以下两种形式,其格式如下:

```
#include < 文件名 >
#include" 文件名"
```

例如:

```
#include < stdlib. h >              //stdlib. h:声明公共的系统标准函数
```

```
#include" MyPrg. h"          //MyPrg. h:声明用户自定义的常量、变量及函数
```

第一种形式中,所要嵌入源文件用尖括号括起来。这种形式的#include 指令告诉编译预处理程序在编译器自带的外部库的头文件中搜索要嵌入的文件,它们一般是系统提供的公共头文件,存放在系统目录中的 Include 子目录下。

第二种形式中,所要嵌入的源文件用双引号括起来。这种形式的#include 指令告诉编译预处理程序先在当前子目录搜索要嵌入的文件(一般是用户自定义的头文件或源文件),如果没有找到文件,则再去搜索编译器自带的或外部库的头文件。

按照 C++ 函数使用要求,如果函数调用在前、函数定义在后,或者调用其他文件中(如系统库)定义的函数时,必须先进行函数声明。系统函数按其功能被分成几个库,对应每个库都有一个头文件,头文件的内容是某一类函数的原型声明。显然,只需在程序中使用#include 指令包含相应的头文件,而不必在程序中直接给出函数的声明。

以多文件方式组织的程序常常需要在各文件之间共享一些常量声明、变量声明、结构声明、函数声明和宏定义,可以将这些语句放在一个 C++ 头文件中(以. h 作为扩展名),然后利用#include 指令将该头文件包含到需要它的源文件中。

1.3.7 头文件

每个 C++ 程序通常分为两个文件,一个文件用于保存程序的声明(declaration),称为头文件;另一个文件用于保存程序的实现(implementation),称为定义(definition)文件。C++ 程序的头文件以“. h”为后缀。头文件由以下三部分内容组成。

(1)头文件开头处的版权和版本声明。

(2)预处理块。

(3)函数和类结构声明等。

头文件的作用有以下几方面。

(1)通过头文件来调用库功能。在很多场合,源代码不便(或不准)向用户公布,只要向用户提供头文件和二进制的库即可。用户只需要按照头文件中的接口声明来调用库功能,而不必关心接口是怎么实现的。编译器会从库中提取相应的代码。

(2)头文件能加强类型安全检查。如果某个接口被实现或被使用时,其方式与头文件中的声明不一致,编译器就会指出错误,这一简单的规则能大大减轻程序员调试、改错的负担。

1.3.8 消息

面向对象的方法提供了对象之间的通信机制。程序由一些相互作用的对象(类)构成,就像人们彼此之间互通信息一样,对象之间的交互通过发送消息来实现。程序通过执行对象的各种行为方法,来改变对象的状态(属性数据),从而使该对象发生某些事件。当对象发生某些事件时,通常需向其他相关对象发送消息,请求它们做出一些处理。

消息是向某对象请求服务的一种表达方法。对象内有方法和数据,外部的用户或对象

对该对象提出了服务请求，可以称为向该对象发送了消息。

1.3.9　继承

客观世界的事物有很多相似性，其一个基本原因是事物之间具有某种“继承”关系。例如：人与古猿有许多相似之处，因为人是由古猿进化而来。面向对象方法正是利用了客观世界的继承特性来构造对象，很好地解决了软件的可重用性问题。

继承(inheritance)是面向对象方法的一个特征，指一个新类可以从已有的类派生而来。新类继承了原有类的特性(即属性和行为)。并且，为了满足具体的需要，新类可以对原有类的行为进行修改，还可以增加新的属性和行为。例如：所有的 Windows 应用程序都有一个窗口，它们可以看做是从一个窗口类派生出来的，但有的用于文字处理，有的程序用于表格操作，有的用于画图，这是根据具体需要，通过继承派生出了不同的类。类与类之间可以组成继承层次，一个类(子类)可以定义在另一个已定义类(父类)的基础上。子类不仅可以继承父类中的属性和操作，而且可以定义自己的属性和操作。

1.3.10　封装

封装(encapsulation)是面向对象的另一个特征，是对象和类的主要特性。它有两个含义：第一个含义是把对象的全部属性和全部服务结合在一起，形成一个不可分割的独立单位(即对象)；第二个含义也称为“信息隐蔽”，即尽可能隐蔽对象的内部细节，对外形成一个边界(或者说形成一道屏障)，只保留有限的对外接口使之与外部发生联系。这主要是指对象的外部不能直接地存取对象的属性，只能通过几个允许外部使用的服务与对象发生联系。封装保证了模块具有较好的独立性，使得程序维护修改较为容易。由于对应用程序的修改仅限于类的内部，因而可以将应用程序修改带来的影响减小到最低限度。

1.3.11　多态性

多态性(Polymorphism)是面向对象的另一重要特征。在通过继承而派生出的一系列类中，可能存在一些名称相同，但实现过程和功能不同的方法(Method)。

例如，我们可以定义一个描述“打印”的类，然后再派生出“在屏幕上打印”的类和“在打印机上打印”的类，所有这些类都需要“打印”的功能。但是，将信息打印到屏幕与将信息打印到打印机的实现方法显然是不同的。

多态性是指当程序中的其他部分发出同样的消息时，按照接收消息对象的不同能够自动执行类中相应的方法。其好处是，用户不必知道某个对象所属的类就可以执行多态行为，从而为程序设计带来更大方便。