

邮电高等函授教材

C 及 C++程序设计

张 富 编

人 民 邮 电 出 版 社

内 容 简 介

本书以 Turbo C++为依据,以 C 语言为起点,全面地介绍 C 语言程序设计基础和面向对象的程序设计方法。全书分为两大部分,第一部分介绍 C 语言基础;第二部分介绍面向对象程序设计的概念和方法。

本书可作为大学的程序设计语言教材或参考书,也可供从事计算机工作的技术人员参考。

邮电高等函授教材 C 及 C++程序设计

◆ 编 张 富
责任编辑 滑 玉

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
北京汉魂图文设计有限公司制作
印刷厂印刷
新华书店总店北京发行所经销

◆ 开本: 787×1092 1/16
印张: 23.75
字数: 581 千字 2000 年 12 月第 1 版
印数: 1-0 000 册 2000 年 12 月北京第 1 次印刷

ISBN 7-115-08476-9/TP·1592

定价: 36.00 元

编 者 的 话

进入 80 年代，出现了一种新的程序设计方法——面向对象的程序设计（Object Oriented Programming），简称 OOP。面向对象的程序设计方法，包括了结构化程序设计的特性，为程序设计人员提供了分析和解决问题的一种全新的方法。在本课程中，你将学习如何用 C++ 语言实现面向对象的程序设计。

C++ 是 C 语言的一个面向对象程序设计的超集。C 语言是 70 年代出现的一种非常卓越的程序设计语言，是由贝尔实验室的 Dennis Richie 设计的一个通用的程序设计语言。C 语言具有丰富的数据类型和结构化程序设计所需要的语句。C 语言的特点是效率高、功能强、简洁灵活。C 语言所具有的地址操作和位操作，使它在一定程度上具有了低级语言的特性，因而可以在相当大的程度上，代替设计难度较大的汇编语言。C++ 语言继承和保持了 C 语言的优点，扩充了 C 语言的功能，又增加了支持 OOP 的能力。

1980 年，贝尔实验室的 Bjarne Stroustrup 博士及其同事，开始对 C 语言进行改进和扩充。最初它被称为“带类的 C”，1983 年取名为 C++，以后又经过不断地完善和发展，成为目前的 C++。

C++ 保持与 C 兼容，许多用 C 语言编写的库函数和实用软件，可以用于 C++ 中。

在 C++ 中，数据和处理数据的函数被结合在一起，称为类。C++ 已被应用于程序设计的众多领域，它特别适用中、大型的程序开发项目。

C++ 语言已经有几个著名的版本。本书基本上是根据 Turbo C++ 编写的。

本教材分为两个部分。第一部分介绍 C 语言基础；第二部分介绍 C++ 的面向对象的程序设计。

在本书中，C 语言部分（第一篇）既是一个相对独立的部分，又可作为 C++ 的组成部分和基础，为进一步学习面向对象作准备。

学习程序设计的最有效的方法，是通过阅读程序例子来掌握编程的思路和技巧，边学习、边设计、边上机练习。书中提供了大量的例题和习题供读者参考。

编 者

2000 年 1 月

目 录

第一篇 C 语言基础

第一章 对 C 语言的初步认识	2
1.1 C 语言源程序	2
1.1.1 什么是源程序	2
1.1.2 C 语言源程序的基本结构	2
1.2 源程序的编译和 C++语言的集成开发环境	5
1.2.1 C 语言源程序的编译	5
1.2.2 C 程序的开发过程	6
1.2.3 C++语言的集成开发环境	6
1.3 函数的初步概念	7
1.3.1 C 函数	7
1.3.2 函数的参数	9
1.3.3 函数的返回值	10
1.3.4 函数的一般格式	10
1.4 几个基本程序语句的小结	11
1.4.1 注释语句	11
1.4.2 赋值语句	11
1.4.3 变量说明语句	11
1.5 简单的数据输入与输出	12
1.5.1 printf()函数	12
1.5.2 scanf()函数	13
1.5.3 putchar()函数	14
1.5.4 getchar()函数	15
小结	15
习题	16
第二章 数据、操作符和表达式	19
2.1 数据类型	19
2.1.1 基本数据类型	19
2.1.2 标识符(Identifier)	19
2.2 整型数据	20
2.2.1 整型变量	20
2.2.2 整型常量	21
2.3 实型数据	21
2.3.1 实型变量	21

2.3.2 实型常量	21
2.4 字符型数据	22
2.4.1 字符型变量	22
2.4.2 字符型常量	23
2.5 变量说明与初始化	24
2.5.1 变量说明	24
2.5.2 全局变量和局部变量	24
2.5.3 变量的初始化	25
2.6 操作符和表达式	26
2.6.1 算术操作符	26
2.6.2 关系操作符	27
2.6.3 逻辑操作符	28
2.6.4 赋值操作符	30
2.7 表达式	31
2.7.1 表达式中数据类型的转换	31
2.7.2 数据类型的强制转换	32
小结	32
习题	33
第三章 程序控制语句——分支结构	35
3.1 空语句与语句块	35
3.1.1 空语句	35
3.1.2 语句块	35
3.2 if 语句	36
3.2.1 简单的 if 语句	36
3.2.2 if_else 语句	39
3.2.3 if_else_if 语句	40
3.2.4 if 中的表达式	42
3.2.5 if 语句的嵌套	43
3.3 switch 语句	46
3.3.1 switch 语句的一般格式和功能	46
3.3.2 default 和 break 语句	49
3.3.3 switch 嵌套	51
小结	53
习题	54
第四章 程序控制语句——循环结构	55
4.1 goto 语句	55
4.2 for 循环语句	57
4.2.1 for 循环语句的格式和基本应用	57

4.2.2 对 for 语句的进一步讨论.....	60
4.3 while 循环.....	65
4.4 do_while 循环.....	67
4.5 多重循环——循环的嵌套.....	69
4.6 continue 语句.....	72
小结.....	74
习题.....	74
第五章 位操作及其它操作	76
5.1 按位操作	76
5.1.1 按位操作符	76
5.1.2 位操作的优先级.....	79
5.2 条件运算操作符“?”.....	80
5.3 逗号(,)、方括号[]和圆括号()操作符.....	82
5.3.1 逗号结合操作	82
5.3.2 方括号和圆括号.....	82
5.4 复合赋值运算符.....	82
小结.....	83
习题.....	84
第六章 数组与字符串.....	85
6.1 一维数组	85
6.1.1 一维数组的定义.....	85
6.1.2 下标变量的引用.....	86
6.1.3 一维数组的初始化.....	89
6.1.4 数组边界检查问题.....	89
6.2 字符串	90
6.2.1 字符串数组	90
6.2.2 字符型数组的初始化.....	90
6.2.3 常用的字符串库函数.....	92
6.2.4 字符串结束标识.....	95
6.3 二维数组与多维数组	95
6.3.1 二维数组与多维数组的定义和引用	95
6.3.2 二维字符型数组.....	97
6.3.3 二维数组的初始化.....	98
小结.....	100
习题.....	100
第七章 指针.....	102
7.1 指针的概念.....	102

7.1.1 指针和指针变量.....	102
7.1.2 指针变量说明	103
7.2 指针操作符和指针表达式.....	103
7.2.1 指针操作符	103
7.2.2 指针变量的初始化.....	104
7.2.3 指针运算与指针表达式	105
7.3 指针和数组.....	107
7.3.1 指向数组的指针.....	107
7.3.2 指针的下标	108
7.4 字符指针与字符数组	110
7.5 指针数组	111
7.6 指向指针的指针	113
小结.....	115
习题.....	115
第八章 函数.....	117
8.1 函数概述	117
8.2 函数的定义和调用	117
8.2.1 函数的定义	117
8.2.2 return 语句.....	119
8.2.3 函数的调用	120
8.3 返回非整型值的函数	121
8.3.1 函数原型的使用.....	122
8.3.2 void 类型函数	123
8.3.3 指针类型函数	123
8.4 函数的值调用和引用调用.....	124
8.4.1 值调用	125
8.4.2 引用调用	125
8.4.3 用数组作为参数调用函数	126
8.5 函数指针	127
8.6 函数的嵌套调用和递归调用	129
8.6.1 函数的嵌套调用.....	129
8.6.2 函数的递归调用.....	131
8.7 主函数 main()	132
8.7.1 主函数 main()的参数	132
8.7.2 函数 main()的返回值	134
小结.....	135
习题.....	135

第九章 数据的存储方式	137
9.1 局部变量和全局变量	137
9.1.1 局部变量	137
9.1.2 全局变量	138
9.2 变量的存储方式	141
9.2.1 局部变量的存储定义	142
9.2.2 全局变量的存储定义	144
9.2.3 变量存储类型小结	147
9.3 内部函数和外部函数	147
9.3.1 内部函数	147
9.3.2 外部函数	148
9.3.3 多文件程序的编译—工程文件	149
9.4 动态存储单元	149
9.5 变量修饰符 const	150
小结	151
习题	152
第十章 用户定义数据类型	153
10.1 结构(Structure)	153
10.1.1 结构的定义、结构变量说明和结构变量成员的访问	153
10.1.2 结构变量的初始化	156
10.1.3 结构数组	158
10.1.4 把结构传递给函数	161
10.1.5 结构指针	162
10.2 位域(Bit_Fields)	167
10.2.1 位域结构的定义	167
10.2.2 位域结构变量的说明	167
10.2.3 位域变量的引用	168
10.3 联合(Unions)	169
10.3.1 联合类型的定义	169
10.3.2 联合类型变量的说明	169
10.3.3 联合类型变量的引用	170
10.4 枚举(Enumerations)	172
10.4.1 枚举类型的定义	172
10.4.2 枚举类型变量的定义	172
10.4.3 枚举类型变量的引用	173
小结	173
习题	174

第十一章 C 语言的预处理器	176
11.1 宏定义和宏替换	176
11.1.1 不带参数的宏定义	176
11.1.2 带参数的宏定义	179
11.1.3 取消宏定义	180
11.2 文件包含	180
11.3 条件编译	181
11.3.1 #if_#endif 类型的条件编译命令	181
11.3.2 #ifdef 和#ifndef 类型的条件编译命令	182
小结	184
习题	184
第十二章 屏幕控制	185
12.1 文本模式	185
12.1.1 文本窗口	185
12.1.2 建立窗口	185
12.1.3 建立窗口函数 window()	187
12.1.4 窗口 I/O 函数	188
12.1.5 文本的几种不同模式	189
12.2 图形模式	192
12.2.1 初始化视频适配器	192
12.2.2 退出图形模式	194
12.3 常用绘图函数	194
12.3.1 图形窗口—视区	194
12.3.2 基本图形函数	195
12.4 图形方式下输出文本	197
12.4.1 文本函数	198
12.4.2 图形文本的样式	198
小结	199
习题	199
第十三章 磁盘文件操作(I/O 系统)	200
13.1 文件概述	200
13.1.1 C 文件的概念	200
13.1.2 流的概念	200
13.1.3 缓冲文件系统和非缓冲文件系统	201
13.2 控制台 I/O	201
13.2.1 简单的控制台 I/O 函数	202
13.2.2 格式化的控制台 I/O	204

13.2.3 标准设备文件及 I/O 改向	204
13.3 ANSI 文件 I/O 系统	205
13.3.1 文件结构类型和文件指针	205
13.3.2 打开文件和关闭文件	206
13.3.3 文件的读和写	208
13.3.4 文件的定位与文件的随机存取	218
13.3.5 删除文件	221
小结	221
习题	221

第二篇 C++面向对象程序设计

第十四章 C++ 概述	224
14.1 面向对象的程序设计	224
14.1.1 传统的程序设计方法	224
14.1.2 面向对象的程序设计	225
14.2 面向对象方法的基本特征	225
14.2.1 对象	225
14.2.2 类	226
14.2.3 继承 (inheritance)	226
14.2.4 多态性 (polymorphism)	226
14.3 C++对 C 的扩充	226
14.3.1 变量的定义	227
14.3.2 C++的函数原型	227
14.3.3 常数说明	227
14.3.4 C++的注释语句	228
14.3.5 C++的标准 I/O 操作	228
14.3.6 作用域区分符	229
14.3.7 函数参数的缺省	230
14.3.8 引用型变量	231
14.3.9 内联函数	233
14.3.10 动态内存分配	234
小结	236
习题	236
第十五章 类	237
15.1 类的结构	237
15.1.1 类的定义	237
15.1.2 类成员函数的定义	238
15.1.3 类的对象的定义与访问	239

15.2	内联函数	242
15.2.1	用修饰符 inline 说明成员函数	243
15.2.2	隐式内联函数	243
15.3	友元函数	244
15.4	类的静态成员	249
15.5	传递对象给函数	251
15.5.1	值调用	251
15.5.2	引用调用	252
15.6	类和指针	254
15.6.1	对象指针	254
15.6.2	this 指针	256
15.7	类与结构类型和联合类型的关系	258
15.7.1	类与结构类型	258
15.7.2	类与联合类型	260
	小结	261
	习题	261
第十六章	类的工具	262
16.1	构造函数和析构函数	262
16.1.1	不带参数的构造函数	262
16.1.2	析构函数	263
16.1.3	带参数的构造函数	265
16.1.4	构造函数参数的缺省值	266
16.2	函数重载	268
16.2.1	一般函数的重载	268
16.2.2	构造函数重载	269
16.2.3	重载类成员函数	271
16.2.4	构造函数的动态初始化	272
16.3	运算符重载	273
16.3.1	成员运算符函数的重载	274
16.3.2	友元运算符函数	278
16.4	对象的动态存储管理	282
	小结	283
	习题	283
第十七章	类的继承	285
17.1	继承	285
17.1.1	继承与派生类	285
17.1.2	公有派生	286
17.1.3	私有派生	289

17.2 继承机制中的初始化	291
17.2.1 不带参数的基类构造函数	291
17.2.2 带参数的基类构造函数	293
17.3 多重继承	295
17.3.1 多重继承的继承机制	295
17.3.2 指向派生类的指针	297
17.4 虚基类	299
17.4.1 多重继承中的二义性	299
17.4.2 虚基类	302
小结	304
习题	304
第十八章 虚函数	307
18.1 虚函数	307
18.1.1 虚函数的概念	307
18.1.2 为什么要使用虚函数	309
18.2 纯虚函数和抽象基类	312
18.2.1 纯虚函数	312
18.2.2 抽象基类	313
18.3 编译连接与执行连接	316
小结	316
习题	316
第十九章 C++的 I/O 系统	317
19.1 C++的 I/O 系统概述	317
19.1.1 流与文件	317
19.1.2 C++的流	318
19.2 定义自己的插入操作符和提取操作符	319
19.2.1 创建插入符	319
19.2.2 重载提取符	322
19.3 格式化 I/O	324
19.3.1 用 ios 类的成员函数实现格式化输出	324
19.3.2 使用控制器函数实现格式化 I/O	327
19.4 文本文件的 I/O	328
19.4.1 打开和关闭文件	328
19.4.2 文本文件的读和写	330
19.4.3 检测文件尾	332
19.5 二进制文件的 I/O	333
19.5.1 字节方式的读写	333
19.5.2 块方式的读写	335

19.6 文件的随机访问	336
小结	339
习题	339
附录一 实验指导书	341
附录二 常用 Turbo C 库函数	355
附录三 常用字符的 ASCII 码	357
附录四 教学大纲	358
附录五 邮电高等函授《 C 及 C++ 程序设计》教学进程表	361
参考资料	363

第一篇

C 语言基础

C++是 C 语言的超集 ,或者说 ,C 语言是 C++的一个子集。作为 C++的基础 ,首先掌握好 C 语言 ,无疑会减少直接学习 C++程序设计中的许多困难。同时 ,C 语言还可以作为程序设计语言独立使用。因此 ,要首先学习 C 语言的程序设计。然后 ,在此基础上再学 C++的面向对象的方法。

第一章 对C语言的初步认识

内 容 提 要

本章介绍 C 语言程序的结构，建立函数的初步概念，本章学习的目的，是使读者首先建立起对 C 语言程序的初步概念，并能开始着手编写(设计)简单的 C 语言程序。

1.1 C语言源程序

1.1.1 什么是源程序

源程序就是用某种程序设计语言编写的程序，它表达了程序设计人员解决某问题的具体方法。源程序以文件的形式存储在计算机的软盘或硬盘中。通常它是一种文本文件。所谓文本文件，就是以 ASCII(American Standard Code for Information Interchange)码存储的文件，它可以用任何编辑软件编写，用 C 语言编写的程序称为 C 语言源程序。

文件要有文件名和扩展名。用 C 语言编写的源程序，其文件扩展名通常为 c，按 C++编写的源程序，其文件扩展名通常为 cpp。

1.1.2 C语言源程序的基本结构

首先考察几个简单的 C 语言源程序(简称 C 源程序)，以便有一个初步的认识。

最简单 C 语言程序有如下的形式：

```
main()
{
}
```

这个程序不做任何事，我们称它是空操作。这个程序没有什么实际意义，我们写出这个程序的目的，只是为了观察 C 语言程序的基本结构。

这里 main()称为主函数。main()函数后面的一对花括号(大括号)，分别表示主函数 main()的开始和结束。在中间，可以写该函数所要做的事情。一个 C 程序可能由一个或若干个函数组成，但必须有一个而且只能有一个 main()函数。所有的 C 或 C++程序，都从主函数 main()开始执行。

例 1.1.1 现在为上面的程序(主函数)添加上具体的功能。

```
/* This is a C program */
#include <stdio.h>
main()
{
    int a,b,c;
    a=1;
    b=2 ;
    c=a+b;
```

```
}
```

程序的第一行是：

```
#include <stdio.h>
```

它是一个系统预处理命令，其作用是，告诉编译器，在编译时需要文件 `stdio.h`。这个文件含有程序为进行正确 I/O(输入输出)操作所需要的库函数，称为包含文件。C++ 提供大量的这类文件，它们需要写在程序的头部，因此又称为头文件(header file)，它们都是以 `h` 为扩展名的文件。一个源程序，根据它的内容，可能需要多个头文件。

下面就是我们已经熟悉的函数 `main(){ }`。花括号内是函数的具体操作内容，称为函数体。只要有点代数的基础，就是没有学过程序设计的人也能看出，这个程序的功能是计算两个整数 `a` 与 `b` 的和，其结果为 `c`。

程序中的 `a`、`b` 和 `c` 称为变量。程序中以分号“`;`”结尾的每一行都是一条语句。

函数体的第一行：

```
int a,b,c;
```

是变量说明语句，它说明或告诉编译器，在这个程序中，将会用到三个整型数值(`int` 类型)的变量 `a`、`b` 和 `c`。在 C 语言中，所有要使用的变量，在使用前都必须先进行说明(定义)，说明中包括变量名和它的类型。语句中用 `int` 定义整型变量，`int` 叫做关键字。用户定义变量名时必须注意，变量名中字母的大小写是有区别的。例如，`A` 和 `a` 是不同的变量。

紧接下面的三个语句称为赋值语句。它们的功能分别是：

将整数 1 赋给变量 `a`;

将整数 2 赋给变量 `b`;

计算 `a+b` 并将计算的结果赋给变量 `c`。

程序运行时，将按语句书写的顺序逐条语句执行，执行完程序的最后一个语句后，运行结束。

例 1.1.2 使程序运行后，在显示器的屏幕上输出(显示)如下一串字符：

```
I'm 20 years old.
```

程序如下：

```
#include <stdio.h>
/* My first program */
main()
{
    int age;

    age = 20;

    printf("I'm %d years old. \n", age);
}
```

看一下这个源程序有什么新的内容。

第二行是：

```
/* My first program */
```

它的作用是为程序加注释，称为注释语句。在 C 语言中，注释的内容以“`/*`”开始，以

“ */ ” 结束。在它们之间的任何内容(可以写若干行),编译器都不予理会,注释是程序员给程序写的一些说明。

下面就是我们已经熟悉的函数 `main()` { }。花括号内是函数的具体操作内容,称为函数体。

函数体的第一行是:

```
int age;
```

是变量说明语句。必须注意变量名中字母的大小写是有区别的。`Age`、`AGE`、`age` 和 `agE` 是一些不同的变量。

下一行:

```
age = 20;
```

是我们已经熟悉的赋值语句。

再下一行是:

```
printf("I'm %d years old. \n", age);
```

`printf()` 是系统提供的用于输出的库函数。它的作用是将程序的输出信息显示到屏幕上,其括号内有两个用逗号分开的参数:

```
"I'm %d years old. \n"
```

```
age
```

这两个参数是调用 `printf()` 函数时传送给它的值。用双引号扩起来的第一个参数叫做控制字符串(control string),它包含有要直接显示的字符串和(或)由“%”开头的格式码。在本例中,要求显示的字符串是 `I'm` 和 `years old.`。格式码 `%d` 的意义是输出十进制整数。输出的数据是第二参数 `age` 的值,它按控制字符串中的格式码 `%d` 进行输出。符号“\n”表示输出数据后再输出一个回车-换行(carriage - linefeed),在 C 中也称为新行(newline)。

例 1.1.3 具有输入数据功能的程序。将例 1.1.1 程序中变量 `a` 和 `b` 的值由程序员从键盘输入。

```
/* This is a C program */
```

```
#include <stdio.h>
```

```
main()
```

```
{
```

```
int a,b,c;
```

```
scanf("%d%d",&a,&b);
```

```
c=a+b;
```

```
}
```

程序中增加了语句:

```
scanf("%d%d",&a,&b);
```

`scanf()` 为输入函数。其中控制字符串中的格式控制码“`%d%d`”说明读入两个整型数,并把这个数值赋给它后面的变量 `a` 和 `b`。变量前的符号 `&`,是为使函数 `scanf()` 正确工作所必需有的,其具体含义将在以后介绍。

程序运行中,当执行这个函数时,将等待程序员从键盘输入两个整型数据,数据输入后程序继续执行下面的语句。

例 1.1.4 编写一程序,其具体任务是将你输入的英尺数转换为公尺数。