



普通高等教育“十一五”国家级规划教材
普通高等学校计算机教育“十二五”规划教材

C 及 C++ 程序设计

(第4版)

C&C++ PROGRAMMING
(4th edition)

张富 王晓军 ◆ 编著



人民邮电出版社
POSTS & TELECOM PRESS



普通高等教育“十一五”国家级规划教材

普通高等学校计算机教育“十二五”规划教材

C 及 C++ 程序设计

(第 4 版)

C&C++ PROGRAMMING
(4th edition)

张富 王晓军 ◆ 编著



人民邮电出版社

北京

图书在版编目 (CIP) 数据

C及C++程序设计 / 张富, 王晓军编著. -- 4版. --
北京: 人民邮电出版社, 2013. 8
普通高等学校计算机教育“十二五”规划教材
ISBN 978-7-115-32039-1

I. ①C… II. ①张… ②王… III. ①
C语言—程序设计—高等学校—教材 IV. ①TP312

中国版本图书馆CIP数据核字(2013)第140533号

内 容 提 要

本书以 Turbo C++为依据, 以 C 语言为起点, 全面地介绍 C++语言的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。

本书可作为高等学校“程序设计语言”课程的教材或参考书, 也可供初学者自学参考。

-
- ◆ 编 著 张 富 王晓军
责任编辑 滑 玉
责任印制 彭志环 杨林杰
 - ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子邮件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京中新伟业印刷有限公司印刷
 - ◆ 开本: 787×1092 1/16
印张: 22.25 2013 年 8 月第 4 版
字数: 580 千字 2013 年 8 月北京第 1 次印刷
-

定价: 49.80 元

读者服务热线: (010)67170985 印装质量热线: (010)67129223

反盗版热线: (010)67171154

再版前言

本书自问世至今，经过多年的教学实践，表明其编写原则和对教材结构的设计，都取得了很好的效果。

本次修订仍是将重点放在了解决教学中的重点和难点问题上。为了有助于学习和理解，特别选择了一些典型的例题由浅入深、由简及繁，引导学习者一步一步地积累经验、启发兴趣，直到能够解决较难、较复杂的问题。这种方式在指针的概念和在数组中的使用、结构类型的应用、文件操作等重点、难点章节中都有体现。

在本次修订中，除增加、调整了一些例题外，还注意使用示意图、流程图，更清晰地描述相关概念和方法，使读者更容易理解和掌握相关内容。

作者利用这次重版的机会，对教材作了一次全面的修订，更正了一些文字、符号等的错误。

尽管做了一些工作，书中的缺点和错误仍然难以完全避免，希望读者和使用本教材的老师继续关注，提出意见和建议，以便不断改进和完善。

编 者

2013年6月

目 录

第一篇 C 语言基础

第 1 章 对 C 语言的初步认识.....2

1.1 程序与程序设计语言.....2
1.1.1 程序、程序设计和程序设计语言2
1.1.2 结构化程序设计方法3
1.2 C 语言及其源程序的基本结构.....4
1.2.1 C 语言.....4
1.2.2 C 语言源程序的基本结构.....5
1.2.3 C 语言的基本语句.....6
1.3 C 语言的基本词法.....7
1.3.1 C 语言的字符集.....7
1.3.2 标识符8
1.3.3 保留字8
1.3.4 C 语言的词类.....8
1.4 源程序的编译和 C 语言的集成开发环境 8
1.4.1 C 程序的开发过程.....8
1.4.2 C 语言的集成开发环境.....9
小结11
习题11

第 2 章 基本数据类型、操作符和表达式.....12

2.1 数据类型.....12
2.2 整型数据.....13
2.2.1 整型常量13
2.2.2 整型变量13
2.3 实型数据.....14
2.3.1 实型常量14
2.3.2 实型变量15
2.4 字符型数据与字符串.....15
2.4.1 字符型常量15
2.4.2 字符型变量16

2.4.3 字符串常量.....16
2.5 变量说明与初始化.....17
2.5.1 变量说明（定义）17
2.5.2 变量的初始化.....17
2.6 运算符和表达式.....17
2.6.1 运算符.....17
2.6.2 表达式.....18
2.6.3 表达式中数据类型的转换.....20
小结21
习题22

第 3 章 顺序结构程序设计.....23

3.1 算术运算符和赋值运算符.....23
3.1.1 算术运算符与算术表达式.....23
3.1.2 赋值运算符与赋值表达式.....24
3.1.3 自反赋值运算符.....25
3.2 赋值语句和注释语句.....26
3.2.1 赋值语句.....26
3.2.2 注释语句.....26
3.3 输入输出语句.....27
3.3.1 字符输出函数 putchar().....27
3.3.2 字符输入函数 getchar().....28
3.3.3 格式输出函数 printf().....28
3.3.4 格式输入函数 scanf().....30
3.4 顺序结构程序设计.....32
小结35
习题35

第 4 章 选择结构程序设计.....38

4.1 关系运算符及关系运算表达式.....38
4.2 逻辑运算符及逻辑表达式.....40
4.3 选择语句.....41
4.3.1 单分支 if 选择语句.....42

4.3.2 双分支 if_else 选择语句	43	7.2.3 二维数组程序设计	91
4.3.3 多分支结构	44	7.3 字符数组与字符串	94
4.3.4 多分支开关语句 switch	47	7.3.1 字符数组的定义与初始化	94
4.4 选择结构程序设计	49	7.3.2 字符串与字符数组	95
4.5 条件运算符	53	7.4 常用的字符串系统库函数	96
小结	55	7.4.1 字符串输入函数 gets()	97
习题	55	7.4.2 字符串输出函数 puts()	97
第 5 章 循环结构程序设计	57	7.4.3 字符串复制函数 strcpy()	97
5.1 逗号运算符和逗号表达式	57	7.4.4 字符串连接函数 strcat()	98
5.2 goto 语句	58	7.4.5 字符串比较函数 strcmp()	98
5.3 循环语句	59	7.4.6 测试字符串长度函数 strlen()	99
5.3.1 for 循环语句	59	7.5 字符数组程序设计实例	99
5.3.2 while 循环语句	63	小结	101
5.3.3 do_while 循环语句	64	习题	101
5.4 多重循环——循环的嵌套	65	第 8 章 指针	103
5.5 break 语句和 continue 语句	67	8.1 指针的概念	103
5.5.1 break 语句	67	8.1.1 指针和指针变量	103
5.5.2 continue 语句	68	8.1.2 指针变量的定义	105
5.6 循环程序设计	69	8.2 指针运算符和指针变量的初始化	105
小结	74	8.2.1 指针运算符	105
习题	74	8.2.2 指针变量的初始化	107
第 6 章 位运算	76	8.2.3 指针运算与指针表达式	107
6.1 位运算符及位运算表达式	76	8.2.4 用指针处理简单变量	109
6.2 位逻辑运算	77	8.3 数组的指针	111
6.3 移位运算	78	8.3.1 指向一维数组的指针	111
6.4 位自反赋值运算	81	8.3.2 指向二维数组的指针	113
小结	81	8.4 用指针处理字符串	118
习题	82	8.5 指针数组	119
第 7 章 数组与字符串	83	8.6 多级指针	121
7.1 一维数组	83	小结	124
7.1.1 一维数组的定义	83	习题	124
7.1.2 数组元素的引用	84	第 9 章 函数	126
7.1.3 一维数组的初始化	85	9.1 函数概述	126
7.1.4 一维数组程序设计	86	9.2 函数的定义、调用和返回	127
7.2 多维数组	89	9.2.1 函数的定义	127
7.2.1 多维数组的定义和引用	90	9.2.2 函数的返回	128
7.2.2 二维数组的初始化	91	9.2.3 函数的调用	129
		9.2.4 函数原型的使用	130

9.2.5 指针类型函数	132	11.2 结构型数组	167
9.3 函数参数的传递方式	133	11.2.1 结构型数组的定义和初始化	167
9.3.1 值传递方式	133	11.2.2 结构型数组元素成员的引用	168
9.3.2 地址传递方式	134	11.3 指向结构型数据的指针	169
9.3.3 数组作为函数参数	134	11.3.1 指向结构型变量指针的定义、 初始化和引用	169
9.4 函数指针	135	11.3.2 结构型变量指针的应用举例	170
9.5 函数的嵌套调用和递归调用	137	11.3.3 指向结构型数组的指针	172
9.5.1 函数的嵌套调用	137	11.4 位域型 (Bit_Fields)	174
9.5.2 函数的递归调用	139	11.4.1 位域型的定义	174
9.6 主函数 main() 的参数	140	11.4.2 位域型变量的说明和初始化	174
9.6.1 主函数 main() 的参数	140	11.4.3 位域型变量的引用	175
9.6.2 函数 main() 的返回值	143	11.5 联合型 (Unions)	176
小结	143	11.5.1 联合型的定义	176
习题	144	11.5.2 联合型变量的说明	176
第 10 章 数据的存储类型	146	11.5.3 联合型变量的引用	177
10.1 变量在内存中的存储	146	11.6 枚举型 (Enumerations)	180
10.2 局部变量和全局变量	146	11.6.1 枚举型的定义	180
10.2.1 局部变量	147	11.6.2 枚举型变量的定义	181
10.2.2 全局变量	147	11.6.3 枚举型变量的引用	181
10.3 变量的存储类型	149	11.7 用户自定义数据类型名称	183
10.3.1 局部变量的存储定义	150	小结	185
10.3.2 全局变量的存储定义	152	习题	185
10.3.3 变量存储类型小结	154	第 12 章 C 语言的预处理器	187
10.4 内部函数和外部函数	154	12.1 宏定义和宏替换	187
10.4.1 内部函数与外部函数	154	12.1.1 不带参数的宏定义和引用	187
10.4.2 在 Turbo C++ 集成环境下编译 多文件程序	155	12.1.2 带参数的宏定义和引用	190
10.5 动态存储单元	156	12.1.3 取消宏定义	192
10.6 修饰符 const	157	12.2 文件包含	192
小结	158	12.3 条件编译	193
习题	158	12.3.1 #if #endif 类型的条件编译命令	194
第 11 章 用户定义数据类型	160	12.3.2 #ifdef 和 #ifndef 类型的条件编译 命令	195
11.1 结构型 (Structure)	160	小结	196
11.1.1 结构型的定义	160	习题	196
11.1.2 结构型变量的定义	161	第 13 章 磁盘文件操作 (I/O 系统)	197
11.1.3 结构型变量的初始化	162	13.1 文件概述	197
11.1.4 结构型变量成员的引用	163		
11.1.5 结构型变量作为函数的参数	165		

13.1.1 C 语言文件的概念	197	13.3.2 文件尾测试函数、错误测试函数和 文件头定位函数	204
13.1.2 二进制文件和文本文件	197	13.3.3 字符串文件读写函数	206
13.1.3 顺序文件和随机文件	198	13.3.4 数据块文件读写函数	207
13.1.4 缓冲文件系统和非缓冲文件 系统	198	13.3.5 格式化读写文件函数	212
13.1.5 文件型指针	198	13.4 文件的定位与文件的随机存取	214
13.2 打开文件和关闭文件	199	13.4.1 文件随机定位函数	214
13.2.1 打开文件函数	199	13.4.2 随机读写文件举例	214
13.2.2 关闭文件函数	201	13.4.3 当前位置函数 ftell()	217
13.2.3 标准设备文件	201	小结	217
13.3 文件的读和写	202	习题	218
13.3.1 字符文件读写函数	202		

第二篇 C++ 面向对象程序设计

第 14 章 C++ 概述	220	15.1.1 类的定义	233
14.1 面向对象的程序设计	220	15.1.2 类成员函数的定义	234
14.1.1 传统的程序设计方法	220	15.1.3 类的对象的定义与访问	235
14.1.2 面向对象的程序设计	221	15.2 类中的内联函数	238
14.2 面向对象方法的基本特征	221	15.2.1 用修饰符 inline 说明成员函数	238
14.2.1 对象	221	15.2.2 隐式内联函数	239
14.2.2 类	222	15.3 类的友元成员	239
14.2.3 继承 (inheritance)	222	15.3.1 定义友元函数	240
14.2.4 多态性 (polymorphism)	223	15.3.2 定义友元成员函数	240
14.3 C++ 对 C 语法的扩充	223	15.3.3 定义友元类	242
14.3.1 变量的定义	223	15.4 类的静态成员	243
14.3.2 C++ 的函数原型	224	15.4.1 静态数据成员	243
14.3.3 常数说明	224	15.4.2 静态成员函数	245
14.3.4 C++ 的注释语句	225	15.5 对象作为函数的参数	246
14.3.5 C++ 的标准 I/O 操作	225	15.5.1 值传递	246
14.3.6 作用域区分符	227	15.5.2 引用传递	247
14.3.7 函数参数的缺省	227	15.6 类的指针	248
14.3.8 引用型变量	228	15.6.1 对象指针	248
14.3.9 内联函数	229	15.6.2 this 指针	249
14.3.10 动态内存的分配	230	小结	251
小结	232	习题	251
习题	232		
第 15 章 类	233	第 16 章 类的工具	253
15.1 类的结构	233	16.1 构造函数和析构函数	253
		16.1.1 不带参数的构造函数	253
		16.1.2 析构函数	254

16.1.3 带参数的构造函数	256	18.2 纯虚函数和抽象基类	299
16.1.4 构造函数参数的缺省值	257	18.2.1 纯虚函数	299
16.2 函数重载	258	18.2.2 抽象基类	301
16.2.1 一般函数的重载	258	18.3 编译连接与执行连接	301
16.2.2 构造函数重载	259	小结	302
16.2.3 重载类成员函数	260	习题	302
16.2.4 构造函数的动态初始化	262		
16.3 运算符重载	263	第 19 章 C++ 的 I/O 系统	304
16.3.1 用成员函数重载运算符	263	19.1 C++ 的 I/O 系统概述	304
16.3.2 用友元函数重载运算符	266	19.1.1 C++ 的 I/O 流的基本概念	304
16.4 对象的动态存储管理	268	19.1.2 输入/输出操作符的使用	305
小结	270	19.2 用户自定义插入操作符和	
习题	271	提取操作符	306
第 17 章 类的继承	273	19.2.1 创建插入操作符 “<<”	306
17.1 继承	273	19.2.2 重载提取操作符 “>>”	309
17.1.1 继承与派生类	273	19.3 格式化 I/O	310
17.1.2 公有派生	274	19.3.1 用 ios 类的成员函数实现	
17.1.3 私有派生	276	格式化 I/O	310
17.2 继承机制中的初始化	277	19.3.2 使用控制器函数实现	
17.2.1 不带参数的基类构造函数	277	格式化 I/O	314
17.2.2 带参数的基类构造函数	279	19.3.3 建立自己的控制器函数	315
17.3 多重继承	282	19.4 文件的 I/O	317
17.3.1 多重继承的继承机制	282	19.4.1 打开和关闭文件	317
17.3.2 指向派生类的指针	284	19.4.2 文件的读和写	319
17.4 虚基类	286	19.4.3 二进制文件的读和写	324
17.4.1 多重继承中的二义性	286	19.4.4 文件的随机访问	325
17.4.2 虚基类	288	小结	327
小结	290	习题	328
习题	290	附录 1 实验指导书	330
第 18 章 虚函数与多态性	293	附录 2 常用 Turbo C 库函数	342
18.1 虚函数	293	附录 3 常用字符的 ASC II	344
18.1.1 虚函数的概念	293		
18.1.2 虚函数的应用	295		

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++语言的基础,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,我们首先来学习C语言的程序设计。然后,在此基础上再学习C++的面向对象的程序设计方法。

- 第1章 对C语言的初步认识
- 第2章 基本数据类型、操作符和表达式
- 第3章 顺序结构程序设计
- 第4章 选择结构程序设计
- 第5章 循环结构程序设计
- 第6章 位运算
- 第7章 数组与字符串
- 第8章 指针
- 第9章 函数
- 第10章 数据的存储类型
- 第11章 用户定义数据类型
- 第12章 C语言的预处理器
- 第13章 磁盘文件操作(I/O系统)

第 1 章

对 C 语言的初步认识

本章介绍程序设计及程序设计语言的有关概念，了解结构化程序设计的基本思想；介绍 C 语言的概况，C 语言程序的基本结构和 C 语言程序的开发过程以及在 Turbo C++ 的集成开发环境下编译、连接和运行 C 语言源程序的操作步骤。本章还将介绍 C 语言的基本词法。

通过本章的学习，读者可以建立起关于计算机程序、程序设计语言和程序设计等基本概念，并对 C 语言有一个初步了解，为进一步学习 C 和 C++ 语言程序设计打下基础。

1.1 程序与程序设计语言

1.1.1 程序、程序设计和程序设计语言

一般来说，程序是对解决或处理一个问题的方法步骤的描述。而计算机程序，则是用某种计算机能识别的语言工具描述的解决问题的方法步骤。例如，有两个数据 a 和 b ，它们的值分别为 1 和 2，求这两个量的和 c 。此问题程序（方法步骤）可描述为：

```
a=1;
b=2;
c=a+b;
```

可以看到，这里是通过 3 个步骤，也叫做 3 条语句来完成的。它们的意义是：

- (1) 将数值 1 赋给 a ;
- (2) 将数值 2 赋给 b ;
- (3) 计算 $a+b$ 的和并将结果赋给 c 。

其中 a 、 b 和 c 在计算机的程序设计语言中，通常称做变量。

编制并记录解决问题的方法步骤的过程就是程序设计。在计算机技术中，将解决一个问题的方法和步骤叫做算法。进行程序设计时要使用计算机能识别的描述算法的工具，这个工具就是计算机程序设计语言。

人们最早使用的程序设计语言是二进制机器语言。二进制机器语言是由计算机硬件系统所决定的，每种计算机都有自己的一套用二进制数码表示的指令的集合，称为该计算机的指令系统，也称为计算机机器语言。显然，不同的计算机系统，其机器语言是不同的。

用机器语言设计程序是相当烦琐、费时和费力的工作。为了减轻程序设计人员的工作量和难度，提高程序设计的效率和质量，很快就出现了用简单的英文字母组合来代表烦琐的二进制指令

的语言——符号语言。用符号编写的程序，计算机的硬件不能直接识别，需要把它翻译成机器语言，用来翻译的软件叫做汇编程序或汇编器，因此，符号语言又称为汇编语言。用汇编语言编写的程序叫做汇编语言源程序，简称源程序。源程序经过汇编后产生计算机能直接执行（运行）的机器语言程序。汇编语言与机器语言一样，也是依赖于机器的语言，不同的计算机系统，其汇编语言是不同的。因此，我们说二进制机器语言和汇编语言都是面向机器的程序设计语言。

计算机程序设计语言的进一步发展，出现了与具体的计算机硬件系统无关的，着重于描述解决问题的算法过程的语言。这种语言接近人类自然语言和数学公式的表达方式，且与机器的具体型号无关，称其为高级程序设计语言，简称高级语言。高级语言属于面向问题的语言，如 ALGOL，FORTRAN，PASCAL 等。相对于高级语言，人们称二进制机器语言和汇编语言为低级计算机程序设计语言。

用高级语言编写的程序也叫做源程序。源程序需要有专门的翻译程序将其翻译成机器语言，计算机才能执行这个程序。高级语言的翻译过程有两种不同的方式：一种称为解释执行方式，其特点是翻译一句执行一句，完成这种翻译工作的程序叫做解释程序；另一种方式是将源程序全部翻译成机器语言程序后，才可以执行的编译方式。编译方式中的翻译软件叫做编译系统，或编译器。

1.1.2 结构化程序设计方法

计算机程序设计语言经历了由机器语言、汇编语言到高级语言的发展过程。在高级语言发展的初期，人们广泛使用的语言有 FORTRAN，ALGOL，BASIC 等，这些语言的特点是以简单的语句序列构成程序。以语句序列为基础组织起来的程序，虽然简单，但对于大型软件来说，不便于管理和维护。为了解决这个问题，在 20 世纪 60 年代末开始提出结构化程序设计概念，也就是将程序由语句序列结构转变为模块集合。在 20 世纪 70 年代这种结构化的程序设计语言（如 PASCAL，C 等）得到了飞速地发展和广泛应用。

结构化程序设计方法的基本思想是，将任何复杂问题分解为若干较为简单的功能模块，每个模块中的任何逻辑问题再用少数几种基本结构（如顺序结构、选择结构、循环结构）加以描述。支持这种结构化程序设计方法的语言称为结构化的程序设计语言。结构化的程序设计方法，主要是实现两个方面的问题：程序的模块化设计和结构化编码。

模块化设计就是将一个复杂问题，自上往下逐步细化，形成多层次的多个相对比较简单功能模块，这样就使问题的解决变得层次清晰简单容易了。所谓结构化编码就是对每个小模块中的每个逻辑功能用几种简单的基本结构进行描述。

结构化程序设计中采用的 3 种基本结构如图 1-1 所示，所有的程序代码都实现在这 3 种结构中。

图 1-1 中虚线框表示一种基本结构。矩形框表示一个简单操作或也可以是一个基本结构。菱形框表示对给定条件进行判断操作，它有两个可能的结果：条件成立（是）或条件不成立（否），根据判断的结果执行不同的操作。图中的带箭头的实线表示流程的走向。

用图 1-1 这种框图方式表示流程的方法，是描述算法的一个很重要和常用的工具，今后还会在描述解决具体问题的算法时使用。

结构化的程序具有层次分明，易编、易读和易修改，从而有利于提高编程的效率和质量。这种方法在软件开发中至今仍占有重要的地位。

进入 20 世纪 80 年代后，为了适应庞大而复杂程序的开发，出现了面向对象的程序设计方法和语言。然而它也吸收和继承了结构化程序设计的方法。

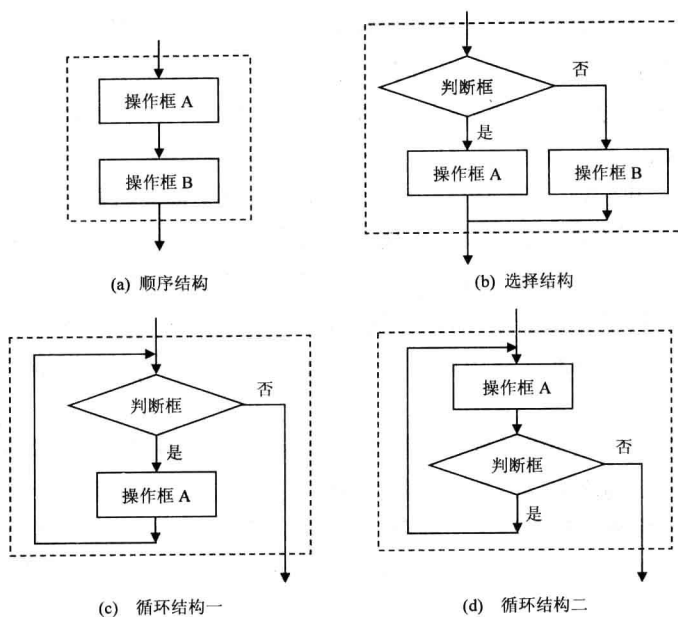


图 1-1 结构化程序设计的 3 种基本结构

1.2 C 语言及其源程序的基本结构

1.2.1 C 语言

C 语言是一种编译方式的结构化高级程序设计语言。1963 年剑桥大学在 ALGOL 语言基础上开发出具有处理硬件能力的语言 CPL (Combined Programming Language)。1967 年该语言经简化演变语言为 BCPL (Basic Combined Programming Language)。1970 年美国贝尔实验室对 BCPL 进一步简化并加强对硬件处理的能力, 推出了 B 语言 (取自 BCPL 中的字母 B)。1972 年贝尔实验室在 B 语言的基础上, 进一步完善和扩充, 开发出 C 语言 (取自 BCPL 中的字母 C)。C 语言最初用于实现 UNIX 操作系统。1977 年 Brain Kernighan 和 Dennis Ritchie 编写了《The C Programming Language》一书, 对 C 语言进行规范化, 成为当时的标准。随着计算机的发展和普及, 出现了一些不同的 C 语言版本。为统一标准, 美国标准化协会 (ANSI) 制定了 C 语言标准, 称为 ANSI C。现在 C 语言已经成为广受欢迎的一种结构化程序设计语言。

目前, 在微型计算机上使用的 C 编译系统有多种版本。本书将以 “Turbo C++ 1.0” 的编译系统为基础介绍 C 语言的程序设计。

C 语言主要有下列一些特点:

- (1) C 语言是一种结构化的程序设计语言, 它具有完整的程序控制语句。
- (2) C 语言是一种模块化程序设计语言, 函数是组成程序的基本程序单位。
- (3) C 语言有丰富的数据类型和运算操作, 使程序设计更为简单和方便。

(4) C 语言提供了类似于汇编语言的低级语言功能, 如直接访问内存的物理地址, 对二进制数进行位操作等。这种语言具有某些低级语言的特性及优点, 从而使这种语言更能接近

硬件。

(5) 语法结构简单, 语句数目不多, 但功能很强。所以, C语言简单易学且应用广泛。

1.2.2 C语言源程序的基本结构

C语言源程序, 简称C程序, 是建立在模块的基础上的, 而基本的模块就是函数。因此, 一个C程序是由一个或多个函数组成的。每个函数是完成一定功能的一段C语言程序。编写一个程序, 首先是建立一个或若干个函数, 然后把它们组织在一起, 构成一个完整的C语言程序。在这些函数中程序必须有且只能有一个函数, 这就是 `main()` 函数, 称为主函数。这就是说, 一个C程序, 至少要由一个函数组成, 这个函数就是主函数 `main()`。一个程序无论包含多少个函数, 程序的运行总是从主函数开始, 在主函数结束。这是C语言程序运行的基本方式。

在C语言中, 除了 `main()` 函数外, 其他函数的函数名是由用户选定的, 称为自定义函数。每个函数都是包含一个或若干个语句并完成一定任务的独立程序单元。每个函数有一个写在函数名后面圆括号内的参数。自定义函数不能像 `main()` 函数那样能独立运行, 它们只能由主函数或其他函数激活 (它也能激活其他函数) 后开始运行。在计算机术语中, 把这种激活叫做调用。所谓调用, 简单地说, 就是函数暂时中断本函数的执行, 转去执行所调用的函数。前者称为主调用函数, 后者称为被调用函数。被调用函数执行完自己的任务后, 必须返回原来的主调用函数, 并从中断了的地方继续执行。这里发生了两个过程: 调用和返回。可见, 函数之间是互相调用和返回的关系。自定义函数可以互相调用, 但不能调用主函数。

在最简单的情况下, C函数有如下的格式:

```
函数名()
{
    函数体
}
```

函数名是用户为函数起的名字; 函数名后跟圆括号, 其内可以含有参数, 也可以没有参数; 写在花括号内的是实现函数功能的程序语句, 称为函数体。

【例 1-1】 一个最简单的C语言源程序。

程序由如下一个主函数组成:

```
main()
{
}
```

因为函数体不包含任何语句, 所以该程序不执行任何功能, 称它是空操作。

【例 1-2】 给例 1-1 的程序加入一个功能: 在显示器上输出: `hello!`。

程序如下:

```
main()
{
    printf("hello!");
}
```

主函数的函数体由一个语句组成。C语言规定每个语句必须以分号结束。即分号表示一个语句的结束。语句

```
printf("hello!");
```

是一个输出语句。在这里它的功能是在显示器的屏幕上显示 (输出) 如下的字符串:

```
hello!
```


【例 1-3】 由两个函数组成的 C 程序。其功能仍然是在显示器上输出：hello!。

程序由主函数 main() 和函数 hello() 组成。主函数的功能是调用函数 hello(), hello() 函数的功能是在显示器的屏幕上输出字符串："hello!"。

下面是源程序清单：

```
main()
{
    hello();
}
hello()
{
    printf("hello!");
}
```

程序运行后，在显示器的屏幕上显示如下字符串：

hello!

主函数的函数体是 C 语言中的一个语句：

```
hello();
```

它的功能是调用函数 hello()。执行调用的结果，使程序从主函数转去执行 hello() 函数。函数 hello() 的函数体是由一个 C 语句组成：

```
printf("hello!");
```

我们已经知道这个语句的功能是在显示器的屏幕上显示 hello!。该语句执行完毕后，返回主函数。由于主函数的所有语句已经执行完毕，于是程序结束运行。程序的执行从主函数开始，在主函数结束。图 1-2 表示出程序的执行过程。

从以上的讨论，可以总结出下列几点。

(1) C 程序是由一个或多个函数组成的，其中必须有一个也只能有一个规定名为 main() 的主函数。

(2) 程序的执行总是从主函数开始，在主函数结束。其他函数是通过调用来执行的。

(3) 主函数可以调用任何非主函数，非主函数之间可以互相调用，但不能调用主函数。

(4) 函数体是完成函数功能的一组 C 语言的语句，每个语句完成一个小功能，并以分号作为一个语句的结束标志。

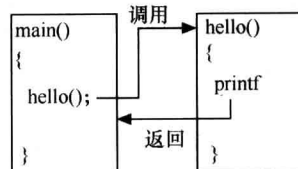


图 1-2 例 1-3 程序运行过程示意图

1.2.3 C 语言的基本语句

C 语言源程序是由函数组成的，其中包括主函数和自定义函数。函数的功能是由 C 语句组成的函数体来实现的。因此，语句是 C 语言中最基本的成分，没有语句也就没有了程序。在例 1-2 和例 1-3 的简单程序中，出现了两个语句：调用函数语句和输出语句。学习 C 语言实现各种功能的语句，是本课程的重要内容之一，是设计程序的基本工具和基础。

C 语言的基本语句主要有以下几种：

- (1) 数据定义语句；
- (2) 赋值语句；
- (3) 函数调用语句和返回语句；
- (4) 输入语句和输出语句；
- (5) 流程控制语句。

1.3 C 语言的基本词法

构成 C 语言的最小单位是字符，由字符构成词类，再由词构成各种语句。有关 C 语言的字符和词类的规定，就形成了 C 语言的词法。本节介绍词法方面的有关规定。

1.3.1 C 语言的字符集

在 C 语言程序中允许使用的所有基本字符的集合，称为 C 语言的字符集。C 语言的字符集采用的是 ASCII (American Standard Code for Information Interchange) 字符集。其中包括：

- (1) 52 个大写和小写的英文字母；
- (2) 10 个数字；
- (3) 如表 1-1 所示的 33 个键盘符号；

表 1-1 键盘符号

符 号	说 明	符 号	说 明	符 号	说 明
	空格符号	!	惊叹号	"	双引号
#	井号	\$	美元号	%	百分号
&	与符号	'	单引号	(	左圆括号
)	右圆括号	*	星号	+	加号
,	逗号	-	减号	.	小数点
/	正斜杠	:	冒号	;	分号
<	小于号	=	等号	>	大于号
?	问号	@	at 圈号	[	左方括号
\	反斜杠	]	右方括号	^	异或号
_	下划线号	`	重音号	{	左花括号
	或符号	}	右花括号	~	波浪号

(4) 如表 1-2 所示的转义字符集。转义字符总是由反斜杠开始，后跟一个或几个字符，用来表示控制代码或特殊符号。转义字符的具体应用，将在以后的章节中介绍。

附录 3 给出 ASCII 基本字符集的编码表。

表 1-2 转义字符集

符 号	说 明	符 号	说 明	符 号	说 明
\n	回车换行符号	\r	回车符号	\'	单引号
\t	tab 符号	\f	换页符号	\\	反斜杠
\v	垂直制表符号	\a	响铃符号	\ddd	1~3 八位进制数 ddd 对应的符号
\b	左退一格符号	\"	双引号	\xhh	1~2 位十六进制数 hh 对应的符号

1.3.2 标识符

用 C 字符集中的字符组合, 可以为程序中的各种对象起名字。这些名字统称为标识符。例如常量名、变量名、函数名等都是标识符。例 1-3 中为函数起的名字 “hello” 就是一个标识符。C 语言对如何构成标识符做了如下的规定。

(1) 标识符是由字母和下划线开头的数字、字母、下划线组成的一串符号。例如:

`abc2, a2bc, a_2, _xq`

都是合法的标识符, 而

`2a, a.2`

则是不合法的, 因为标识符不能以数字开头, 标识符中不能含有小数点。

(2) 一个字母的大写和小写看作是不同的字符。例如:

`abc, aBc`

是两个不同的标识符。

(3) 一个标识符可以由一个或多个字符组成, 但 ANSI C 规定, 标识符的长度不得超过 32 个字符。

1.3.3 保留字

保留字又称关键字, 是 C 语言编译系统使用的、具有特定语法意义的一些标识符。这些标识符用户不能作为自己的标识符使用, 所以把这些标识符称为保留字。C 语言中的保留字有: `auto`, `break`, `case`, `char`, `continue`, `const`, `default`, `do`, `double`, `else`, `enum`, `extern`, `float`, `for`, `goto`, `int`, `if`, `long`, `register`, `return`, `short`, `signed`, `sizeof`, `static`, `struct`, `switch`, `typedef`, `union`, `unsigned`, `void`, `volatile` 和 `while` 等。

1.3.4 C 语言的词类

C 语言的词类主要有以下几种。

- (1) 常量: 在程序运行中值不发生改变的数据。
- (2) 变量: 存放值可变化的数据。
- (3) 运算符: 加工数据的运算符号。
- (4) 表达式: 由常量、变量、函数及运算符组成的式子。
- (5) 保留字: 编译系统使用的、具有特定语法意义的标识符。

上述词类的具体使用将在后续的章节详细讲述。

1.4 源程序的编译和 C 语言的集成开发环境

1.4.1 C 程序的开发过程

开发一个 C 语言程序, 要经过以下 4 个阶段:

- (1) 编辑源程序文件;
- (2) 编译源程序;