



C语言

程序设计

主编 田丰春 杨种学
副主编 曹晨 杨鑫
王小正 李朔



南京大学出版社

L 语言

程序设计

主编 田丰春 杨种学
副主编 曹晨 杨鑫
王小正 李朔

 南京大学出版社

图书在版编目(CIP)数据

C 语言程序设计 / 田丰春, 杨种学主编. —南京:
南京大学出版社, 2016.1

ISBN 978 - 7 - 305 - 16333 - 3

I. ①C… II. ①田… ②杨… III. ①C 语言 - 程序设计 IV. ①TP312

中国版本图书馆 CIP 数据核字(2015)第 309024 号

出版发行 南京大学出版社
社 址 南京市汉口路 22 号 邮 编 210093
出 版 人 金鑫荣

书 名 C 语言程序设计
主 编 田丰春 杨种学
责任编辑 单 宁 吴宜锴 编辑热线 025 - 83596923

照 排 南京理工大学资产经营有限公司
印 刷 南通印刷总厂有限公司
开 本 787 × 1092 1/16 印张 22 字数 536 千
版 次 2016 年 1 月第 1 版 2016 年 1 月第 1 次印刷
ISBN 978 - 7 - 305 - 16333 - 3
定 价 45.00 元

网 址: <http://www.njupco.com>
官方微博: <http://weibo.com/njupco>
官方微信: njupress
销售咨询热线: (025)83594756

* 版权所有, 侵权必究
* 凡购买南大版图书, 如有印装质量问题, 请与所购
图书销售部门联系调换

前 言

C 语言程序设计是高等院校计算机专业及相关专业重要的专业基础课,其目的是培养学生的程序设计理念、掌握程序设计的基本方法,为后续课程打下坚实的基础。

我们在多年教授 C 语言的过程中,深切感受到学习 C 语言程序设计不仅要掌握 C 语言的语法要点和编程规范,更重要的是要领会结构化程序设计的思想,实现思维方式的转换,使学生真正拥有解决实际问题的能力。本教材从计算思维培养的角度出发,面向实际应用,以“学生成绩管理系统”案例贯穿始终,以讲授程序设计为主,借助任务驱动的模式将知识点串接起来,循序渐进地引入各章节知识点的讲解,并通过实际案例的思考分析,形成逻辑清晰的脉络和主线,从而加深对 C 语言的理解和驾驭能力,提升分析问题和解决问题的能力。

全书共分 10 个章节,内容包括:绪论、顺序结构程序设计、选择结构程序设计、循环结构程序设计、数组、函数、指针、结构体和共用体、文件、C 语言应用程序设计实例。每个章节都以“学生成绩管理系统”为例引入本章知识点,以循序渐进的任务驱动方式,引导读者逐步编写出有一定规模的、贯穿全书的综合应用程序。每章结尾以表格形式给出本章知识点和常见错误小结,帮助读者理清思路。

本书由田丰春、杨种学策划主编并整理。第 1~4 章由田丰春编写,第 5~6 章由曹晨编写,第 7 章由杨鑫编写,第 8 章由王小正编写,第 9 章由李朔编写,第 10 章由田丰春编写。

由于时间仓促,作者的水平有限,书中难免有不足之处,恳请读者批评指正。

编 者
2015 年 12 月

目 录

01	第1章 绪论	1
02	1.1 程序与程序设计语言	1
03	1.2 为什么学习 C 语言	2
04	1.3 C 语言程序的结构	4
05	1.4 如何运行 C 程序	7
06	1.5 算法	8
07	1.6 软件开发	12
08	小结	12
09	习题	13
10	第2章 顺序结构程序设计	14
11	2.1 引例	14
12	2.2 数据的表现形式	15
13	2.2.1 标识符	15
14	2.2.2 数据类型	16
15	2.2.3 不同类型的常量	19
16	2.2.4 不同类型的变量	22
17	2.2.5 符号常量和常变量	26
18	2.3 基本运算	29
19	2.3.1 算术运算符和表达式	29
20	2.3.2 赋值运算	31
21	2.3.3 自动类型转换与强制类型转换	33
22	2.3.4 C 语言的其他运算	34
23	2.4 常用数学库函数	35
24	2.5 数据的输入输出	36
25	2.5.1 格式化输出函数 printf()	36
26	2.5.2 格式化输入函数 scanf()	41
27	2.5.3 字符数据的输入输出函数	44
28	小结	46
29	习题	47

第3章 选择结构程序设计	49
3.1 引例.....	49
3.2 关系运算符和关系表达式.....	50
3.3 逻辑运算符和逻辑表达式.....	51
3.4 用 if 语句实现选择结构	53
3.5 条件运算符和条件表达式.....	58
3.6 if 语句的嵌套	59
3.7 实现多分支选择的 switch 语句.....	63
小结	66
习题	67
第4章 循环结构程序设计	68
4.1 引例.....	68
4.2 循环控制结构与循环语句.....	69
4.3 几种循环的比较.....	75
4.4 循环嵌套.....	81
4.5 流程控制语句.....	84
4.5.1 用 break 语句提前终止循环	84
4.5.2 用 continue 语句提前结束本次循环	85
4.6 循环程序举例	87
小结	96
习题	98
第5章 数组	99
5.1 引言.....	99
5.2 一维数组	100
5.2.1 一维数组的定义	100
5.2.2 一维数组元素的引用	101
5.2.3 一维数组元素的初始化	102
5.2.4 一维数组的应用(1)	103
5.2.5 一维数组的应用(2)	107
5.3 二维数组	112
5.3.1 二维数组的定义	112
5.3.2 二维数组元素的引用	113
5.3.3 二维数组的初始化	113
5.3.4 二维数组的应用	114
5.4 字符数组	119
5.4.1 字符数组的定义与初始化	119

081 5.4.2 字符串的输入和输出	120
081 5.4.3 字符串的处理函数	122
081 5.4.4 字符串的应用	125
081 小结	128
081 习题	129
第6章 函数	131
081 6.1 引言	131
081 6.1.1 函数的作用	131
081 6.1.2 模块化的程序设计思想	133
081 6.2 函数定义	135
081 6.3 函数的调用和参数传递	138
081 6.3.1 函数的调用	138
081 6.3.2 函数的参数传递	140
081 6.3.3 函数的返回值	143
081 6.4 函数的声明和原型	144
081 6.5 函数的嵌套与递归调用	146
081 6.5.1 函数的嵌套调用	146
081 6.5.2 函数的递归调用	148
081 6.6 数组作为函数参数	153
081 6.7 变量的作用域和存储类型	156
081 6.7.1 变量的作用域	156
081 6.7.2 变量的存储类型	161
081 6.8 内部函数和外部函数	166
081 小结	166
081 习题	167
第7章 指针	169
081 7.1 什么是指针	169
081 7.2 指针的定义及使用	170
081 7.2.1 指针变量的定义及赋值	170
081 7.2.2 指针变量的引用	172
081 7.2.3 指针相关的运算	174
081 7.2.4 指向指针的指针	175
081 7.3 指针与数组	175
081 7.3.1 一维数组与指针	175
081 7.3.2 指针与二维数组	177
081 7.3.3 数组指针	178
081 7.3.4 指针与字符串	179

051	7.3.5 指针数组	180
051	7.4 指针和函数	182
051	7.4.1 指针作为函数的参数	182
051	7.4.2 指向函数的指针变量	187
051	7.4.3 返回指针值的函数	190
151	小结	192
151	习题	193
151	第8章 结构体与共用体	194
051	8.1 概述	194
051	8.1.1 结构体的引入	194
051	8.1.2 结构体类型的定义	195
051	8.2 结构体变量定义	195
051	8.2.1 结构体变量的定义与初始化	195
051	8.2.2 结构体变量的引用	198
051	8.3 结构体数组	199
051	8.3.1 结构体数组的定义与初始化	200
051	8.3.2 结构体数组应用举例	201
051	8.4 指向结构体类型数据的指针	203
051	8.4.1 指向结构体变量的指针	203
051	8.4.2 指向结构体数组的指针	204
051	8.4.3 用结构体变量和指向结构体的指针作函数参数	206
051	8.5 用指针处理链表	209
051	8.5.1 链表概述	209
051	8.5.2 简单链表	210
051	8.5.3 处理动态链表所需的函数	211
051	8.5.4 建立动态链表	213
051	8.5.5 输出链表	216
051	8.5.6 对链表的删除操作	217
051	8.5.7 对链表的插入操作	220
051	8.5.8 对链表的综合操作	222
051	8.6 共用体	224
051	8.6.1 共用体类型定义	224
051	8.6.2 共用体变量定义与引用	224
051	8.7 枚举类型	227
051	8.8 用 <code>typedef</code> 定义类型	228
051	小结	230
051	习题	231

第 9 章 文件	232
9.1 文件概述	232
9.2 常用文件操作函数	235
9.2.1 文件打开/关闭	235
9.2.2 文件读/写	238
9.3 文件操作示例	242
小结	251
习题	251
第 10 章 C 语言应用程序设计实例	252
10.1 背景知识	252
10.2 核心知识点	252
10.3 系统开发环境	252
10.4 系统实施	252
10.5 小结	276
附 录	277
附录 A 在 Visual C ++ 6.0 环境下运行 C 程序的方法	277
附录 B 常用字符与 ASCII 代码对照表	280
附录 C C 语言中的关键字	281
附录 D 运算符的优先级及结合方式	281
附录 E 常用标准库函数	282
参考文献	287

第1章 終論

1.1 程序与程序设计语言

我们从小到大进行过无数次的考试，老师们常常对我们的成绩要进行求平均分、求总分、排序、查询等各种统计分析工作，如果都用手工来进行，那将是一件非常繁琐的工作。所幸如今已是信息时代，老师们常借助计算机来完成这些成绩管理的工作。但是，我们看到的计算机都是一些物理硬件，它是怎么样工作的？它为什么能对成绩进行排序等操作呢？

其实，计算机都是按照程序来工作的，有人按照老师的需求编写了用于学生成绩管理的程序，老师在计算机上运行这个程序就可以进行学生成绩的管理了。那么什么是程序呢？

程序是告诉计算机做什么和如何做的一组指令（语句），是人们事先编写好的一组计算机能识别和执行的指令，每一条指令使计算机执行特定的操作。计算机按照程序来工作，离开程序，计算机什么也做不了，只有懂得程序设计，才能真正了解计算机是怎样工作的，才能更深入地使用计算机。

人与人交流的语言称为自然语言，人与计算机交流的语言，我们就称为计算机语言。不同领域问题的描述是有差异的，为了能确切而又简单地描述这些问题，人们设计了数千种专用的和通用的计算机语言，有的适用于科学计算，有的适用于编写系统软件，有的适用于自动控制。

计算机语言的发展，经历了从机器语言、汇编语言到高级语言的历程。

1. 机器语言

计算机是一种电子机器，其硬件使用的是二进制，只能识别和接受由“0”和“1”构成的一串二进制代码，如：

1011001000100000001

这种计算机能直接识别和接受的二进制代码称为机器指令。每种计算机都有自己的一套机器指令，机器指令的集合就是该计算机的机器语言。机器语言直接用二进制表示，是计算机硬件系统真正理解和执行的唯一语言。用机器语言编写的程序可以被计算机直接执行，因此它的效率最高，执行速度最快。但由于不同类型计算机的指令系统不同，因而在某一种类型计算机上编写的机器语言程序，在另一种不同类型的计算机上也可能不能运行，即移植性差。而且它与人们所习惯的自然语言差别太大，人们不易记忆和理解，也难于修改和维护，所以现在人们已不再用机器语言编制程序了。

2. 汇编语言

汇编语言用助记符来代替机器指令的操作数和操作码，如用ADD表示加法、SUB表示减法等，这样就构成了计算机符号语言，如：

ADD AX,BX

用汇编语言编写的源程序计算机并不能直接识别和执行，必须经过“翻译”变成机器语

言后,才能被计算机识别和执行,所以其执行速度要慢于机器语言编写的程序。这种将汇编语言源程序翻译成机器语言的软件我们称为汇编程序,“翻译”的过程称为“汇编”。

汇编语言在一定程度上克服了机器语言难以辨认和记忆的缺点,但仍然不够直观简便、可移植性差、难以普及,只在专业人员中使用。

机器语言和汇编语言是完全依赖于具体机器特性的,是面向机器的语言,我们统称为计算机低级语言。

3. 高级语言

为了克服低级语言的缺点,提高编写程序和维护程序的效率,一种接近人们自然语言的程序设计语言应运而生了,这就是高级语言。

高级语言功能很强,它的表示方法更接近人类的自然语言(英语)和数学语言,具有学习容易、使用方便等特点,如:

$$B = A + 5$$

显然,这与使用数学语言对计算过程的描述是一致的,而且这样的描述适用于任何配置了这种高级语言处理系统的计算机,具有通用性,在一定程度上与计算机无关,即不依赖于具体机器,与具体机器距离较远。

用高级语言编写程序,虽然方便,但计算机不能直接识别和执行,必须经过“翻译”,用一种称为编译程序的软件把用高级语言编写的程序(称为源程序)转换为机器指令的程序(称为目标程序),然后才能由计算机来执行。

早期应用比较广泛的几种高级语言有 FORTRAN、BASIC、PASCAL 和 C 等,在此之后,又诞生了上百种高级程序设计语言,并根据应用领域的不同和语言本身侧重点差异,分成了许多类别。但高级语言的本质都是相通的,在学会了一门经典语言之后,就能很容易地掌握其它高级语言。

1.2 为什么学习 C 语言

早期的系统软件主要用汇编语言编写,因而程序与计算机硬件的关系十分密切,使程序编写难度大、可读性差、难于一致。

1969 年,美国贝尔实验室的 Ken Thompson 和 Dennis Ritchie 为 DEC PDP-7 计算机设计了一个操作系统软件,即最早的 UNIX 系统。

在 1972~1973 年间,贝尔实验室的 Dennis Ritchie 在 B 语言的基础上设计出了 C 语言。1973 年,Ken Thompson 和 Dennis Ritchie 两人合作把 UNIX 操作系统的 90% 以上程序用 C 语言改写,增加了多道程序设计能力,同时大大提高了 UNIX 操作系统的可移植性和可读性。

后来,C 语言又做了多次改进,渐渐形成了不依赖于具体机器的 C 语言编译程序,于是 C 移植到其它机器时所需做的工作大大简化了,成为如今广泛应用的计算机语言之一。

随着 C 语言使用得越来越广泛,C 语言的编译程序也有不同的版本。一般来说,1978 年 B. W. Kernighan 和 Dennis Ritchie 合著的 The C Programming Language 是各种 C 语言版本的基础,称之为旧标准 C 语言或“K&R C”。1989 年,美国国家标准局(ANSI)颁布了第一个官方的 C 语言标准,称为“ANSI C”或“C89”。1990 年,这个标准被国际标准化组织(ISO)

采纳,将其命名为“ISO C”或“C90”。目前使用的如 Microsoft C、Turbo C 等版本都把 ANSI C 作为一个子集,并在此基础上做了合乎它们各自特点的扩充。

总的来说,C 语言是一种用途广泛、功能强大、使用灵活的过程性编程语言,既能用于编写系统软件,又能用于编写应用软件。因此,C 语言问世以后得到迅速推广,学习和使用 C 语言的人越来越多。C 语言成为学习和使用人数最多的一种计算机语言。掌握 C 语言成为计算机开发人员的一项基本功。从图 1-1 所示的 Tiobe 在 2015 年 11 月公布的编程语言受欢迎程度的排行榜(最新的统计图请访问图 1-1 中的网址),可以看到,C 语言依然位于前列。

Nov 2015	Nov 2014	Change	Programming Language	Ratings	Change
1	2	▲	Java	20.403%	+6.01%
2	1	▼	C	17.145%	-0.32%
3	4	▲	C++	6.198%	+0.10%
4	5	▲	C#	4.318%	-0.67%
5	7	▲	Python	3.771%	+1.18%
6	6	▲	PHP	3.248%	+0.20%
7	8	▲	JavaScript	2.473%	+0.38%
8	10	▲	Visual Basic .NET	2.223%	+0.16%
9	14	▲	Ruby	2.038%	+0.83%
10	9	▼	Perl	2.032%	-0.04%
11	29	▲	Assembly language	1.883%	+1.28%
12	15	▲	Delphi/Object Pascal	1.682%	+0.73%
13	11	▼	Visual Basic	1.681%	+0.02%
14	3	▼	Objective-C	1.426%	-7.64%
15	18	▲	Swift	1.236%	+0.40%

图 1-1 2015 年 11 月统计的编程语言热门程度排行榜 TOP 15 榜单
 (资料来源:<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>)

目前大多数高校都将 C 语言作为新生的第一门语言课,因此学过 C 语言的人最多,熟悉 C 语言风格语法的人也最多,所以 C 语言成为程序设计思想交流的首选媒介语言。在涉及编程能力考查的笔试、面试时,C 语言通常都是必考的。现在大多数的主流编程语言如 Java、C++ 等语言也都是与 C 语言一脉相承的。从 C 语言入门后,再学习其他语言,也就容易多了。

1.3 C 语言程序的结构

下面通过两个实例,来说明 C 语言程序的基本结构。

【例 1.1】 输入一个学生的两门课程成绩,计算成绩之和并输出(注:成绩不含小数)。

【程序代码】

```
/* 下面程序用于求成绩之和 */
#include <stdio.h> // 包含文件 stdio.h 在本程序中
int main(void) // 定义主函数
{
    int a,b,sum; // 函数开始的标志
    // 定义三个整型变量 a,b,sum
    printf("Please input a and b:\n"); // 在屏幕上输出提示信息
    scanf("%d%d", &a, &b); // 从键盘输入两个整数并保存在变量 a 和 b 中
    sum = a + b; // 计算 a + b,结果保存在变量 sum 中
    printf("The sum is %d\n",sum); // 在屏幕上输出 sum 的值
    return 0; // 结束主函数的执行,返回 0 值到系统
} // 函数结束的标志
```

【运行结果】

```
Please input a and b:
89 95
The sum is 184
Press any key to continue
```

以上运行结果是在 Visual C++ 6.0 环境下运行的结果,屏幕首先显示“Please input a and b:”,此时用户从键盘输入 89 和 95,屏幕接着显示“The sum is 184”。屏幕最后一行是 Visual C++ 6.0 系统在输出完运行结果后自动输出的一行信息,告诉用户:“如果想继续进行下一步,请按任意键”。当用户按任意键后,屏幕上不再显示运行结果,而返回程序窗口,以便进行下一步工作(如修改程序)。

【程序分析】程序第 1 行是注释,以“/*”开始,以“*/”结束,中间的文字用来对程序有关部分进行必要的说明。这种注释称为块式注释,可以包含多行内容,也可以单独占一行,编译系统在发现一个“/*”后,会开始找注释结束符“*/”,把二者间的内容作为注释。C 语言还允许以“//”开始的单行注释,它可以单独占一行,也可以出现在一行中其他内容

的右侧,注释的范围从“//”开始,以换行符结束。注释是给人看的,不是让计算机执行的,所以在编译程序时注释部分是不产生目标代码的。

程序第3行的main是函数的名字,表示“主函数”,括号中的void表示main函数没有参数,void也可以省略。main前面的int表示此函数的类型是int类型(整型),在执行主函数后会得到一个值(即函数值),其值为整型。程序第10行“return 0;”的作用是:当main()函数执行结束时将整数0作为函数值,返回到调用函数处。每一个C语言程序都必须有一个main函数()。函数体由花括号括起来,函数体中的每个语句最后都有一个分号,表示语句的结束。

程序第5行是声明部分,定义a,b和sum为整型(int)变量。

程序第6行是一个输出语句,作用是在屏幕上显示提示信息,提示用户输入a和b的值.printf函数中的双撇号内的字符串“Please input a and b:”按原样输出,“\n”是换行符,即在输出“Please input a and b:”后,显示屏上的光标位置移到下一行的开头。

程序第7行是一个输入句,作用是输入变量a和b的值。scanf后面圆括号中包含两部分的内容:一是双撇号中的内容,它指定输入的数据按什么格式输入,“%d”的含义是十进制整数形式。二是输入的数据准备放在哪里,即赋给哪个变量。现在,scanf()函数中指定的是a和b,在a和b的前面各有一个“&”,在C语言中“&”是地址符,“&a”的含义是“变量a的地址”,“&b”是“变量b的地址”。执行scanf()函数,从键盘读入两个整数,送到变量a和b的地址处,然后把这两个整数分别赋给变量a和b。

程序第8行是一个赋值语句,作用是先将变量a与变量b的值相加,然后赋值给sum。

程序第9行也是一个输出语句,作用是输出变量sum的值。在执行printf()函数时,对双撇号括起来的“The sum is%d\n”是这样处理的,将“The sum is”原样输出,“%d”由变量sum的值取代之,“\n”执行换行。

程序第2行是一个编译预处理指令,作用将stdio.h文件包含到本程序中来。scanf()和printf()都是C编译系统提供的函数库中的标准输入输出函数,在使用这些函数时,编译系统要求程序提供有关此函数的信息(例如对这些输入输出函数的声明和宏的定义、全局量的定义等),“#include <stdio.h>”就是用来提供这些信息的。“stdio.h”是系统提供的一个文件名,stdio是“standard input & output”的缩写,文件后缀.h的意思是头文件,因为这些文件都是放在程序各文件模块的开头的。输入输出函数的相关信息已事先放在stdio.h文件中。现在,用include指令把这些信息调入程序供使用。

【例1.2】编写一个既可以求排列,又能求组合的应用程序。

排列公式: $A(n,m) = n! / (n - m)!$

组合公式: $C(n,m) = n! / (m! * (n - m)!)$

【程序代码】

文件Exam1-2.c代码如下:

```
//Exam1-2.c:实现求排列组合的源文件
#include <stdio.h>           /* 包含文件stdio.h在本程序中 */
#include "Exam1-2.h"           /* 包含文件Exam1-2.h在本程序中 */
```

```

int main(void) /* 主函数 */
{
    int n,m,p,z; /* 定义 n、m、p、z 为整型的变量 */
    printf("input n,m:");
    scanf("%d,%d",&n,&m);
    p=pailie(n,m); /* 调用函数 pailie 求排列,保存到变量 p 中 */
    z=zuhe(n,m); /* 调用函数 zuhe 求组合,保存到变量 z 中 */
    printf("A(%d,%d)=%d\n",n,m,p); /* 输出排列 */
    printf("C(%d,%d)=%d\n",n,m,z); /* 输出组合 */
    return 0; /* 结束主函数的执行,返回 0 值到系统 */
}

int pailie(int n,int m) /* 求排列的函数 */
{
    return(fac(n)/fac(n-m));
}

int zuhe(int n,int m) /* 求组合的函数 */
{
    return(fac(n)/(fac(m)*fac(n-m)));
}

int fac(int n) /* 求阶乘的函数 */
{
    int i,t;
    for(i=1,t=1;i<=n;i++)
        t=t*i;
    return (t);
}

```

文件 Exam1 - 2.h 代码如下：

```

// Exam1 - 2.h: 实现求排列组合的头文件
int pailie(int n,int m); /* 求排列的函数声明 */
int zuhe(int n,int m); /* 求组合的函数声明 */
int fac(int n); /* 求阶乘的函数声明 */

```

【运行结果】

```

Input n,m:6,3
A(6,3) = 120
C(6,3) = 20
Press any key to continue

```

【程序分析】

本程序由 2 个文件组成,“Exam1 - 2.c”文件是用于求排列组合的源程序,“Exam1 - 2.h”文件是用于声明函数的头文件。

“Exam1 - 2.c”源文件包含 4 个函数:主函数 main();求排列的函数 pailie();求组合的函数 zuhe();求阶乘的函数 fac()。

每一个 C 语言程序都必须有且仅有一个 main() 函数,它是程序的入口,程序的运行从 main() 函数开始,当 main() 函数执行结束时,整个程序也就结束了。main() 前面的 int 表示此函数的类型是 int 类型(整型)。在执行主函数后会得到一个值(即函数值),其值为整型。函数体用花括号括起来。

程序从 main() 函数开始执行函数体内的代码,当执行到“`p = pailie(n, m);`”时,程序调用 pailie() 函数,流程跳转到 pailie() 函数中执行 pailie() 函数体内的代码,在 pailie() 函数体内又两次调用 fac() 函数求阶乘,得到排列的值后返回到主函数 main(),继续执行后面的“`z = zuhe(n, m);`”,此时程序又调用 zuhe() 函数,流程跳转到 zuhe() 函数中执行组合函数体内的代码,在 zuhe() 函数体内又三次调用 fac() 函数求阶乘,得到组合的值后返回到主函数 main(),继续执行后面的代码,直到 main() 函数体内代码执行完毕,整个程序也就结束了。

“Exam1 - 2.h”头文件对 pailie() 函数、zuhe() 函数、fac() 函数进行了声明。因为 main() 函数中要调用 pailie() 和 zuhe() 两个函数,而这两个函数又要调用 fac() 函数。程序的编译是自上而下进行的,如果被调用函数的定义在主调函数之后,主调函数是不认识这些函数的,所以要先进行声明。声明语句可以放在主调函数的开头,也可以集中放在头文件中。再通过 #include “Exam1 - 2.h” 将头文件包含进来。

本例所涉及的细节读者现在可能不大理解,在学习了相关章节后自然就能理解了。在本章节介绍此例只是想让读者对 C 程序的组成和形式有一个初步的了解。

通过两个实例,我们可以看到 C 语言程序的组成及书写规则为:

- (1) 一个程序由一个或多个文件组成。
- (2) 函数是 C 程序的主要组成部分。C 程序是由一个或多个函数组成的,其中必须要有一个且只能有一个 main() 函数。无论这个函数的位置在哪里,程序总是从它开始执行。main() 函数可以调用其他函数,但是其他函数不能调用 main() 函数。
- (3) 在一个函数内,语句的执行顺序是从上到下的。
- (4) C 语言程序书写形式自由,一行可以写多条语句,每条语句以分号结束(为了程序格式的清晰,最好一行只写一条语句)。程序中的所有标点符号都是英文符号。
- (5) C 语言严格区分大小写,即大写字母“`A`”和小写字母“`a`”被认为是不同的符号。
- (6) 程序应当包含注释。给程序添加必要的注释可以提高程序的可读性,编程者应养成给程序注释的好习惯。

1.4 如何运行 C 程序

计算机不能直接识别和执行用高级语言编写的指令。用 C 语言编写的源程序必须经

过编译、连接生成可执行文件后方可运行。具体步骤如下：

(1) 上机输入和编辑 C 源程序。输入的源程序文件用.c 作为后缀名(如 Exam1 - 1. c)。

(2) 对源程序进行编译,生成二进制目标文件(如 Exam1 - 1. obj)。

(3) 进行连接处理。经过编译后的目标文件还不能直接执行,必须将所有编译后得到的目标模块连接装配起来,再与函数库相连接成一个整体,生成一个可执行程序(如 Exam1 - 1. exe)。

(4) 运行可执行程序。生成可执行文件后,程序就可以在操作系统控制下运行。

一个程序从编写到运行成功,并不是一次成功的,往往需要经过多次反复的测试与修改。图 1-2 给出了运行 C 程序的基本流程,可以看到这其实是一个不断修正、完善的过程。

为了编译、连接和运行 C 程序,必须要有相应的编译系统。目前使用的 C 语言编译系统很多都是集成在开发环境(IDE)的,如基于 DOS 环境的 Turbo C、基于 Windows 环境的 Visual C++ 都是常用的集成开发环境。本书的程序都是用 Visual C++ 6.0 集成环境编译的。本书后面的附录 A 中详细介绍了 Visual C++ 6.0 集成开发环境。

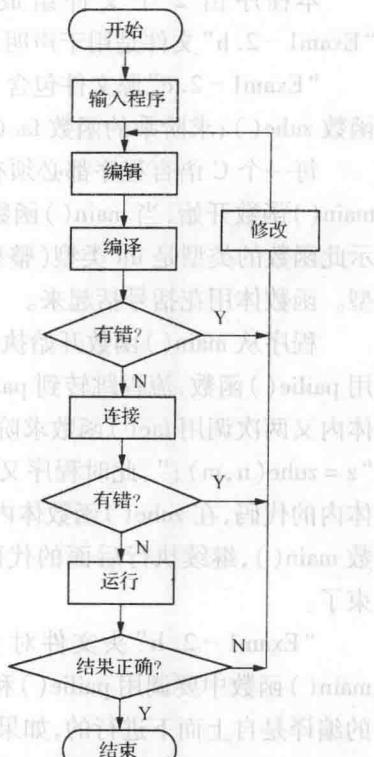


图 1-2 运行 C 程序的基本流程

1.5 算 法

人们常说:“软件的主体是程序,程序的核心是算法。”这是因为要使计算机解决某个问题,首先必须进行问题分析,确定该问题的解决方法与步骤,即算法,然后再据此编写程序并交给计算机执行。下面通过一个求解问题的实例来说明算法。

【例 1.3】 输入两个整数 a 和 b,如果 a > b,就将 a 和 b 的值交换,然后输出 a 和 b 的值,否则,直接输出 a 和 b 的值。

要实现 a 和 b 的值的互换,必须借助第 3 个变量。我们可以这样考虑:有一个装着水的杯子 a 和一个装着牛奶的杯子 b,要将两个杯子的东西互换,只靠这两个杯子倒来倒去是无法实现的,必须借助第 3 个杯子 t。先将 a 杯中的水倒在 t 杯中,再将 b 杯中的牛奶倒在 a 杯中,最后再把 t 杯的水倒在 b 杯中,这就实现了水和牛奶的互换。

基于上述解决问题的思路,就可以明确解决问题的步骤,即确定解决问题的算法。

算法是一组明确解决问题的步骤,它产生结果并可在有限的时间内终止。

算法必须满足下列基本要求:

(1) 确定性。算法中的每一步操作必须清楚明确,无二义性。