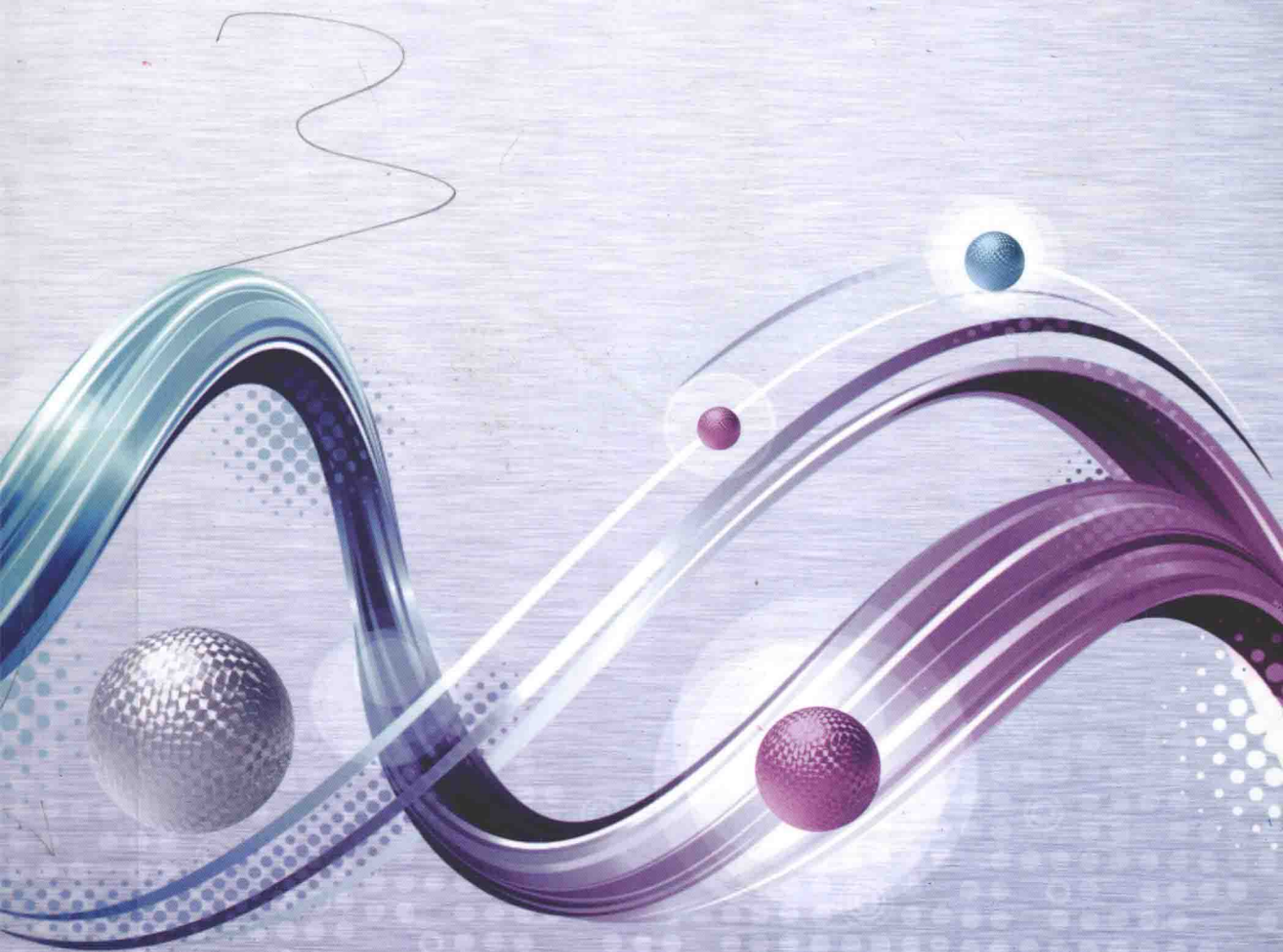




高等学校电子信息类“十三五”规划教材
应用型网络与信息安全工程技术人才培养系列教材

C语言程序设计

蔺冰 王力洪 等编著



高等学校电子信息类“十三五”规划教材

应用型网络与信息安全工程技术人才培养系列教材

C 语言程序设计

蔺冰 王力洪 等编著

西安电子科技大学出版社

内 容 简 介

本书是一本实用性 C 语言程序设计教程, 所讲内容既充分考虑了 C 语言重要语法的全面性, 又突出对学生程序开发的实践能力和工程能力的训练。本书共分为 13 章, 内容包括 C 语言概述, 面向过程的算法设计, 数据类型及格式输出, 运算符、格式输入、顺序结构程序设计, 选择结构程序设计, 循环结构程序设计, 函数框架及语法, 数组使用, 结构体和共用体, 指针, 文件操作, 链表, 位运算和预处理命令。本书通过大量实例介绍 C 语言, 引导读者运用调试手段完善程序设计, 让读者逐步加深对程序设计方法的理解, 掌握程序的设计与调试。本书语言通俗易懂、示例丰富, 并提供了习题和参考答案。

本书可作为高等学校计算机及相关专业的教材, 也可供计算机应用开发者自学使用。

图书在版编目(CIP)数据

C 语言程序设计/蔺冰等编著. —西安: 西安电子科技大学出版社, 2016.6

高等学校电子信息类“十三五”规划教材

ISBN 978-7-5606-4117-1

I. ① C… II. ① 蔺… 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 124067 号

策 划 李惠萍

责任编辑 李惠萍 杨 璠

出版发行 西安电子科技大学出版社(西安市太白南路 2 号)

电 话 (029)88242885 88201467 邮 编 710071

网 址 www.xduph.com 电子邮箱 xdupfb001@163.com

经 销 新华书店

印刷单位 陕西华沐印刷科技有限责任公司

版 次 2016 年 6 月第 1 版 2016 年 6 月第 1 次印刷

开 本 787 毫米×1092 毫米 1/16 印 张 20.5

字 数 481 千字

印 数 1~3000 册

定 价 38.00 元

ISBN 978-7-5606-4117-1/TP

XDUP 4409001-1

如有印装问题可调换

目 录

第 1 章 C 语言概述.....	1	3.3.4 整型数据编码及溢出.....	29
1.1 计算机语言.....	1	3.4 浮点型数据.....	31
1.2 C 语言出现的历史背景.....	3	3.4.1 浮点型常量.....	31
1.3 C 语言的特点.....	3	3.4.2 浮点型变量.....	31
1.4 运行 C 程序的步骤.....	4	3.4.3 浮点格式输出.....	31
1.4.1 使用 VC++ 6.0 运行 C 程序的步骤.....	4	3.4.4 浮点型数据编码及舍入误差.....	32
1.4.2 使用 GCC 运行 C 程序的步骤.....	9	3.5 字符型数据.....	33
1.5 简单的 C 语言程序介绍.....	10	3.5.1 字符常量.....	33
习题.....	13	3.5.2 字符串常量.....	34
第 2 章 面向过程的算法设计.....	14	3.5.3 字符变量及其格式输出.....	34
2.1 算法的概念.....	14	3.5.4 字符数据在内存中的存储形式.....	35
2.2 面向过程算法采用的结构及 传统流程图.....	14	习题.....	36
2.2.1 顺序结构.....	14	第 4 章 运算符、格式输入、 顺序结构程序设计.....	38
2.2.2 选择结构.....	15	4.1 运算符.....	38
2.2.3 循环结构.....	15	4.1.1 运算符.....	38
2.2.4 传统流程图.....	16	4.1.2 赋值运算符.....	38
2.3 简单算法的举例.....	17	4.1.3 算术运算符.....	39
2.4 算法的特性和要求.....	21	4.1.4 复合赋值运算符.....	39
习题.....	21	4.1.5 自增、自减运算符.....	40
第 3 章 数据类型及格式输出.....	22	4.1.6 变量赋初值.....	40
3.1 C 语言的数据类型.....	22	4.1.7 各类数值型数据间的混合运算.....	40
3.2 常量、变量.....	22	4.1.8 逗号运算符和逗号表达式.....	41
3.2.1 常量.....	22	4.1.9 C 语句.....	41
3.2.2 标识符.....	23	4.2 格式输入.....	42
3.2.3 变量的定义.....	23	4.2.1 数据输入/输出的概念.....	42
3.2.4 内存内容和内存地址.....	23	4.2.2 格式输出.....	42
3.2.5 printf 函数格式输出.....	26	4.2.3 格式输入.....	43
3.3 整型数据.....	27	4.3 字符数据的输入/输出.....	45
3.3.1 整型常量.....	27	4.3.1 putchar 函数.....	45
3.3.2 整型变量.....	27	4.3.2 getchar 函数.....	46
3.3.3 整型格式输出.....	27	4.4 输入缓冲区.....	47

4.5 顺序程序设计	48	7.4.1 局部变量	97
习题	50	7.4.2 全局变量	98
第 5 章 选择结构程序设计	52	7.5 变量的存储类别	99
5.1 关系运算符和关系表达式	52	7.5.1 动态存储方式与静态存储方式	99
5.1.1 关系运算符及其优先级	52	7.5.2 auto 变量	99
5.1.2 关系表达式	52	7.5.3 用 static 声明的变量	99
5.2 逻辑运算符和逻辑表达式	53	7.5.4 用 extern 声明的外部变量	100
5.2.1 逻辑运算符及其优先级	53	7.5.5 关于变量的声明和定义	103
5.2.2 逻辑表达式	53	7.5.6 内存区域划分简介	105
5.3 if 语句	54	习题	105
5.3.1 if 语句的三种形式	54	第 8 章 数组使用	108
5.3.2 if 语句的嵌套	60	8.1 一维数组的定义和使用	108
5.3.3 条件运算符	63	8.1.1 一维数组的定义	108
5.4 switch 语句	64	8.1.2 一维数组元素的引用	108
5.5 选择结构程序设计	66	8.1.3 一维数组的初始化	109
习题	68	8.1.4 一维数组程序示例	109
第 6 章 循环结构程序设计	69	8.2 二维数组的定义和使用	116
6.1 while 语句	69	8.2.1 二维数组的定义	117
6.2 do-while 语句	70	8.2.2 二维数组的引用	117
6.3 for 语句	71	8.2.3 二维数组的初始化	117
6.4 循环嵌套	73	8.2.4 二维数组程序示例	118
6.5 break 语句和 continue 语句	77	8.3 字符数组	122
6.5.1 break 语句	77	8.3.1 字符数组的定义	122
6.5.2 continue 语句	79	8.3.2 字符串和字符串的结束标志	122
6.6* goto 语句	81	8.3.3 字符数组的初始化	123
6.7 循环程序举例	83	8.3.4 字符数组的引用	123
习题	86	8.3.5 字符数组的输入/输出	124
第 7 章 函数框架及语法	87	8.3.6 字符串处理函数	126
7.1 函数相关术语及执行流程	87	8.4 数组作为函数参数	131
7.2 函数的分类	89	8.4.1 数组元素作函数实参	131
7.2.1 库函数	89	8.4.2 数组名作函数参数	132
7.2.2 自定义函数	89	习题	134
7.2.3 无返回值函数	90	第 9 章 结构体和共用体	137
7.2.4 无参函数	91	9.1 定义结构体类型变量的方法	137
7.3 函数的调用	91	9.2 结构体变量的引用	139
7.3.1 对被调用函数的声明	91	9.3 结构体数组	140
7.3.2 函数调用及调用格式	92	9.3.1 定义结构体数组	140
7.3.3 函数的递归调用	95	9.3.2 结构体数组的初始化	141
7.4 局部变量和全局变量	97	9.4 共用体	149

9.5 枚举类型.....	156	11.3 文件的打开与关闭.....	196
9.6 用 typedef 定义类型.....	158	11.3.1 文件打开函数.....	196
习题.....	158	11.3.2 文件关闭函数.....	197
第 10 章 指针	160	11.4 文件的读写.....	197
10.1 地址和指针的概念.....	160	11.4.1 文本文件读写函数.....	198
10.2 变量的指针和指向变量的指针变量.....	160	11.4.2 二进制文件读写函数.....	200
10.2.1 定义一个指针变量.....	160	11.5 文件随机读写.....	202
10.2.2 指针变量的引用.....	161	习题.....	205
10.2.3 指针变量作为函数参数.....	163	第 12 章 链表	206
10.3 数组与指针.....	166	12.1 链表概述.....	206
10.3.1 指向数组元素的指针.....	166	12.2 简单链表.....	206
10.3.2 通过指针引用数组元素.....	166	12.3 动态链表.....	208
10.3.3 用数组名作函数参数.....	168	12.3.1 创建动态链表.....	209
10.3.4 多维数组与指针.....	170	12.3.2 链表的查找.....	213
10.4 字符串与指针.....	172	12.3.3 对链表的删除操作.....	214
10.4.1 字符串的操作方式.....	172	12.3.4 对链表的插入操作.....	216
10.4.2 字符指针作函数参数.....	173	习题.....	222
10.4.3 const 类型限定符.....	174	第 13 章 位运算和预处理命令	224
10.5 指向结构体类型的指针.....	176	13.1 位运算符和位运算.....	224
10.5.1 指向结构体数据类型的指针.....	177	13.2 位段.....	227
10.5.2 指向结构体数组的指针.....	178	13.3 预处理命令.....	228
10.5.3 用结构体变量和指向结构体的 指针作函数参数.....	179	13.3.1 宏定义.....	228
10.6 返回指针值的函数.....	181	13.3.2 文件包含.....	229
10.7 指针数组和指向指针的指针.....	182	13.3.3 条件编译.....	229
10.7.1 指针数组的概念.....	182	习题.....	230
10.7.2 指向指针的指针.....	185	附录 A ASCII 码表	231
10.7.3 指针数组作 main 函数的形参.....	187	附录 B C 语言关键字	233
10.8 指向函数的指针.....	189	附录 C 运算符及结合性	234
10.8.1 用函数指针变量调用函数.....	189	附录 D 常用 C 语言库函数	235
10.8.2 用指向函数的指针作函数参数.....	190	附录 E 二、八、十、十六进制换算	242
习题.....	192	附录 F 整数的补码	244
第 11 章 文件操作	194	附录 G 文件路径	245
11.1 C 语言文件概述.....	194	附录 H 习题参考答案	247
11.2 文件处理流程.....	195	参考文献	316

第 1 章 C 语言概述

自然语言是人与人交流的工具，人类的思维可以通过语言来表达。程序设计语言经处理后成为计算机可以识别的机器语言，是人与计算机交流的工具。人们把需要计算机完成的工作告诉计算机，就要用程序设计语言来编写程序，然后让计算机执行程序完成相应的工作。

1.1 计算机语言

计算机语言分为机器语言、汇编语言和高级程序语言。

机器语言是计算机能够直接识别的二进制代码(为方便阅读采用十六进制描述)，如：

```
55  
8B EC  
83 EC 44  
53  
56  
57  
8D 7D BC  
B9 11 00 00 00  
B8 CC CC CC CC  
F3 AB  
C7 45 FC 02 00 00 00  
8B 45 FC  
83 C0 03  
89 45 FC  
8B 4D FC  
51  
68 08 70 42 00  
E8 FA 06 FF FF  
83 C4 08
```

```
33 C0
5F
5E
5B
83 C4 44
3B EC
E8 18 08 FF FF
8B E5
5D
C3
```

编写机器语言时需要查阅相关指令表，将需要执行的指令翻译成十六进制后输入计算机并运行。机器语言难记、难写、难修改、难检查，使用非常困难。

为了较容易地编写程序，可使用简单的助记符号来表达机器指令，这样就产生了汇编语言，如：

```
push    ebp
mov     ebp,esp
sub     esp,44h
push    ebx
push    esi
push    edi
lea    edi,[ebp-44h]
mov     ecx,11h
mov     eax,0CCCCCCCCh
rep stos dword ptr [edi]
mov     dword ptr [ebp-4],2
mov     eax,dword ptr [ebp-4]
add     eax,3
mov     dword ptr [ebp-4],eax
mov     ecx,dword ptr [ebp-4]
push    ecx
push    offset string "a=%d, b=%d" (00427008)
call    printf (004010b0)
add     esp,8
xor     eax,eax
pop     edi
pop     esi
pop     ebx
```



```
add     esp,44h
cmp     ebp,esp
call    __chkesp (004011e0)
mov     esp,ebp
pop     ebp
ret
```

汇编语言源程序并不能直接在计算机中运行，需要经过专门的汇编程序将其转换为机器指令后才能运行。

机器语言和汇编语言与硬件密切相关，不能移植。在编写汇编语言时最好对计算机整体工作模式有一定的了解。

高级语言使用自然语言和数学语言来描述，容易编写和维护修改，如：

```
#include <stdio.h>
int main(void)
{
    int c;
    c = 2;
    c = c + 3;
    printf("%d\n", c);
    return 0;
}
```

高级语言程序也不能被计算机直接运行，需要通过编译程序把高级语言源程序转换成机器指令后才能执行。高级语言的一条语句往往对应多条机器指令。

1.2 C 语言出现的历史背景

1972 年，贝尔实验室的 Dennis M. Ritchie 在 B 语言的基础上设计出了 C 语言。C 语言既保持了精练、接近硬件的优点，又克服了过于简单、数据无类型等缺点。

1973 年，K. Thompson 和 Dennis M. Ritchie 合作把 UNIX 90% 以上的代码用 C 语言改写。

1977 年出现了不依赖于具体 C 语言的编译文本——“可移植 C 语言编译程序”，使 C 语言移植到其他机器时所需做的工作大大简化，这也推动了 UNIX 操作系统迅速地在各种机器上应用。随着 UNIX 的广泛使用，C 语言也迅速得到推广。

1.3 C 语言的特点

C 语言之所以能存在和发展，并具有较强的生命力，成为程序员的首选语言之一，是

因为它具有不同于其他语言的特点。

(1) C 语言简洁、紧凑，使用方便、灵活。C 语言只有 32 个关键字、9 种控制语句。

(2) C 语言具有丰富的运算符。C 语言的运算符包含的范围很广泛，共有 34 种运算符。C 语言把括号、赋值、强制类型转换、逗号等都作为运算符处理，从而使 C 语言的运算类型极其丰富，表达式类型多样。

(3) C 语言的数据类型丰富，能实现各种复杂的运算。C 语言的数据类型有整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型等，能用来实现各种复杂的数据结构(如链表、栈、树等)的运算。尤其是指针类型数据，可使编程更加灵活、多样。

(4) C 语言是完全模块化和结构化的语言。C 语言具有结构化的控制语句(如 if-else 语句、while 语句、do-while 语句、switch 语句、for 语句)，并且可用函数作为程序的模块单位，便于实现程序的模块化。C 语言是良好的结构化语言，符合现代编程风格的要求。

(5) C 语言兼有高级语言和低级语言的特点。C 语言程序也和其他高级语言一样需要通过编译、链接才能得到可执行的目标程序。另外 C 语言可以直接访问物理内存，进行位(bit)操作，以及对硬件进行操作，能实现低级汇编语言的大部分功能。

(6) C 语言编写的程序可移植性好。C 语言编写的程序基本上不做修改就能用于各种型号的计算机和各种操作系统。

(7) C 语言生成的目标代码质量高，程序运行效率高。C 语言生成的目标代码效率只比汇编程序低 10%~20%。

1.4 运行 C 程序的步骤

1.4.1 使用 VC++ 6.0 运行 C 程序的步骤

下面通过例 1-1 源程序，介绍 C 语言程序运行的步骤。

例 1-1 程序功能：在屏幕上输出“Hello World!”。

程序源代码如下：

```
#include <stdio.h>           //预处理命令，包含头文件 stdio.h
/*
main 函数也叫主函数、C 语言的入口函数
C 语言程序的开始和结束唯一函数
*/
int main(void)
{                               //函数开始标识

    printf("Hello World!\n");   //调用库函数，在屏幕输出一串字符串
    return 0;
}                               //函数结束标识
```

运行 Microsoft Visual C++ 6.0 后的界面如图 1-1 所示。

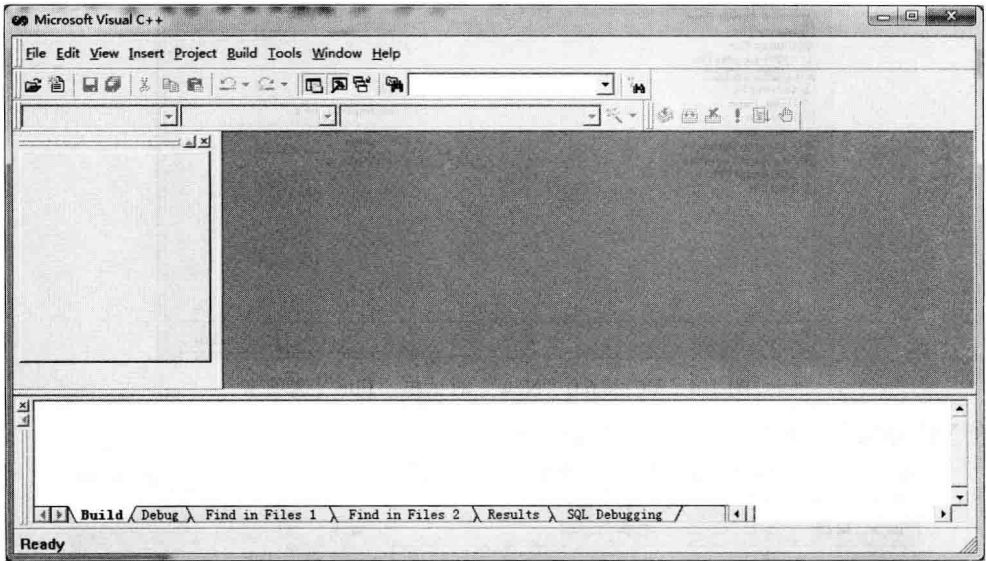


图 1-1 VC++ 6.0 运行界面

打开如图 1-2 所示菜单栏上的“File”菜单，显示子菜单项如图 1-3 所示。

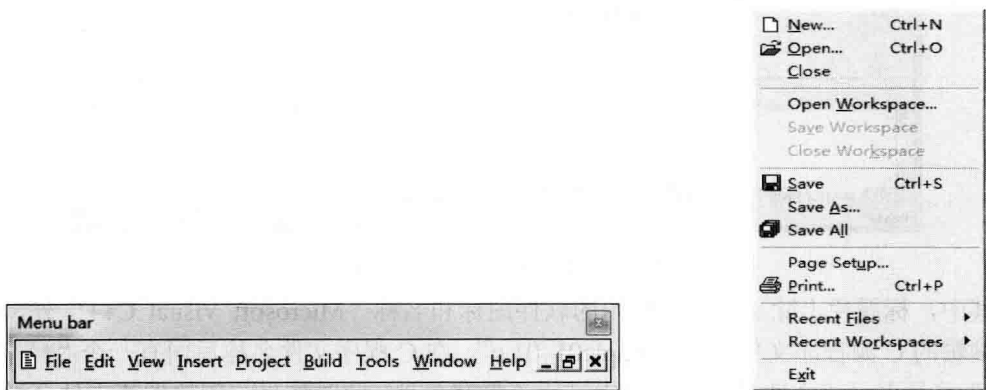


图 1-2 VC++ 6.0 菜单栏

图 1-3 VC++ 6.0 “File”的子菜单项

点击“New...”命令，打开如图 1-4 所示的 New 对话框，选择“Files”选项卡，选择“C++ Source File”选项，在右边的“File”文本框中键入源程序文件名 example01_01.c。其中，example01_01 是文件主名，由编程者自己命名，.c 是扩展名，标识该程序是 C 源程序(需要在文件名后手动添加.c 扩展名，如果不加.c 扩展名，则建立的是面向对象的源程序.cpp)。点击“Location:”文本框后面的“...”按钮，选择该文件的存盘路径，点击“OK”按钮后，就可以编辑 C 源程序了。

由于 VC++ 6.0 默认是支持面向对象程序 C++ 的，必须手动指定建立的是面向过程的 C 源程序。C 源程序可以用任何文本编辑软件进行编辑，也可以在不同的操作系统中运行。在 Windows 操作系统中文件名大小写均可以，Linux 系统会区分大小写，即 a.c 和 a.C 在 Windows 系统下是同名文件，而在 Linux 系统下是不同名文件。

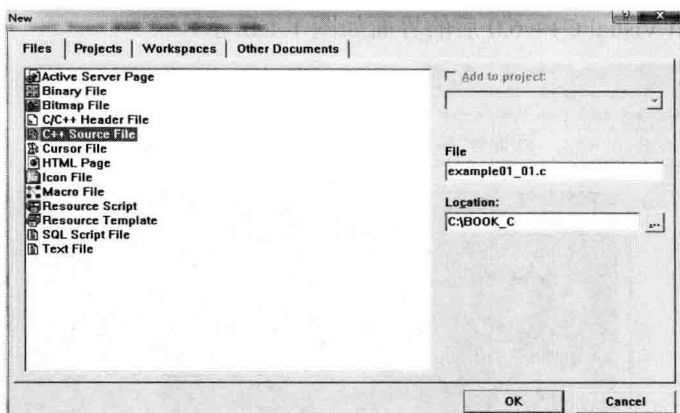


图 1-4 VC++ 6.0 “New”对话框“File”选项卡

在 VC 界面的源程序编辑区中输入代码，如图 1-5 所示。



图 1-5 在 VC++ 6.0 中编辑 C 源程序

其中，标题栏上除了有正在使用的软件图标和名称“Microsoft Visual C++”外，还有正在编辑的 C 源程序文件名“Example01_01.c”。在 C 程序文件名称后面有一个“*”符号，有“*”符号表示该文件被修改或编辑过后还没有存盘，此时如果关闭软件或文件会弹出提示框，如图 1-6 所示。如果修改或编辑过后存过盘，则不会出现“*”符号，并且在关闭软件或文件时不会弹出图 1-6 所示的文本框。

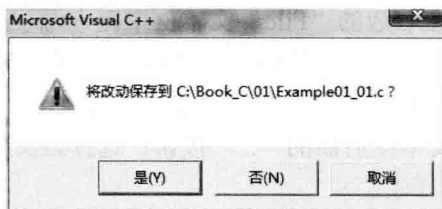



图 1-6 提示保存对话框

单击菜单栏上的“Build”菜单，弹出如图 1-7 所示子菜单项。点击 Compile Example01_01.c 命令，会对 Example01_01.c 源程序进行编译，也可点击如图 1-8 所示“Build MiniBar”工具栏上的按钮进行编译。

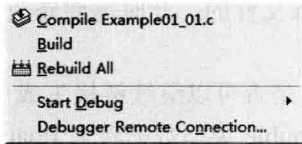


图 1-7 VC++ 6.0 “Build” 菜单



图 1-8 VC++ 6.0 “Build MiniBar” 工具栏

编译新的 C 源程序会弹出如图 1-9 所示的提示框，询问编程人员是否需要创建一个项目工作区。由于程序必须工作在项目里，所以此处必须点击“是”按钮，否则程序不进行编译操作。

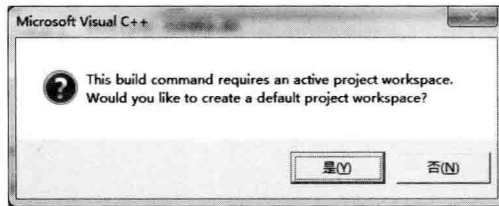


图 1-9 提示创建项目工作区对话框

对于任何一个完整的程序，VC 都需要用项目的方式进行组织。一个项目可以使用多个文件、多个资源等。

编译后，软件下方区域会显示编译的结果，如图 1-10 所示的“0 error(s), 0 warning(s)”，表示编译正确并生成了扩展名为 .obj 的文件(Example01_01.obj)。

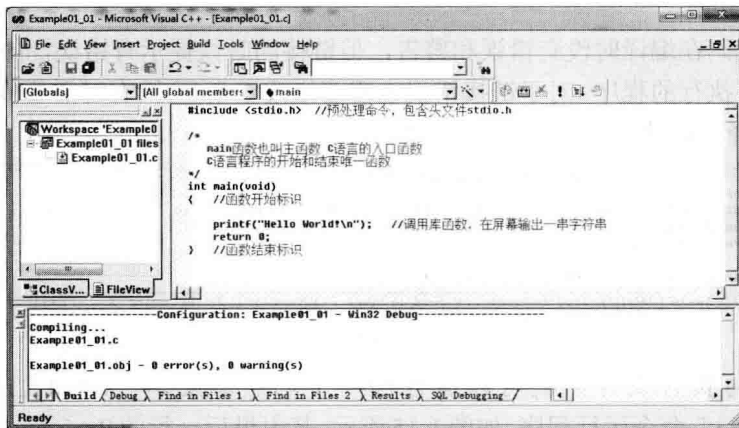


图 1-10 编译成功界面

如果编译结果是如图 1-11 所示的界面，则表示编辑的源代码中有一个错误，一个警告，即：在源程序的第 6 行有一个错误——在 return 前缺少“;”，在第 6 行有一个警告——“main 必须返回一个值”。

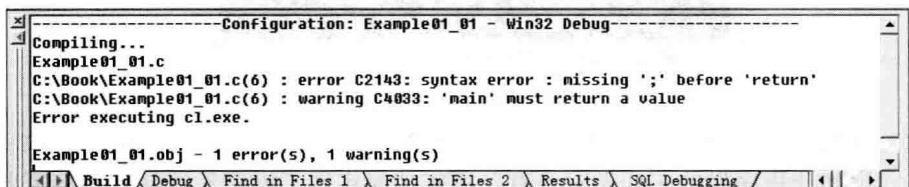



图 1-11 编译警告错误提示界面

编译后的程序如果有错误是不可能链接生成可执行文件的，此时需要修改 C 源程序后重新进行编译。编译后应该没有错误或警告。

对于警告，可以通过警告描述来判断是否有问题，是否可以继续链接生成可执行文件。有少部分的警告可以不管，如定义的变量没有引用，double 类型数据转为 float 可能会丢失精度等。但初学时最好不要出现任何警告。

创建项目工作区后，“Build”菜单如图 1-12 所示。

编译成功后就可以链接生成可执行文件了。点击菜单“Build”下的命令“Build Example01_01.exe”命令(或按钮)，如果没有链接错误就可以生成 Example01_01.exe 可执行文件，如图 1-13 所示。

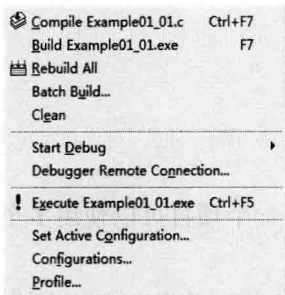


图 1-12 有项目工作区的

“Build”菜单

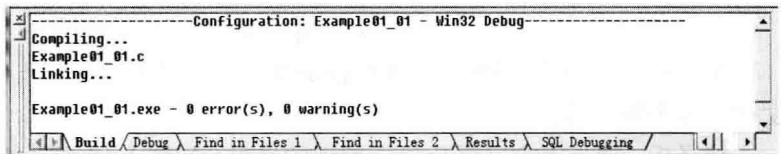


图 1-13 链接成功提示界面

若程序源代码在编译时没有错误和警告，但链接的时候提示有错误，如图 1-14 所示，则也无法生成可执行的程序。对于链接错误也需要修改 C 源程序(有时需要修改项目工作区环境)后重新编译再链接。

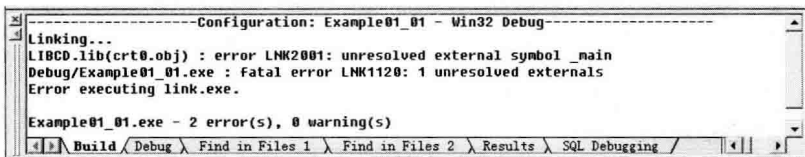


图 1-14 链接错误提示界面

链接生成可执行文件后就可以运行程序了。点击菜单“Build”下的“Execute Example01_01.exe”命令运行程序，如图 1-15 所示。其中最后一行的 Press any key to continue 是 VC++ 6.0 软件添加上的内容，不是程序运行的结果，该行之前是程序的运行结果。

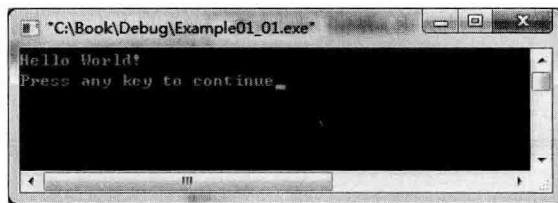


图 1-15 VC++ 6.0 运行程序结果

至此，开发一个 C 语言程序的方法就基本讲解完了，但实际上要完成一个项目还要做很多工作。比如让程序做的东西做对没有？所需功能实现了没有？这些工作都需要针对程序要求和结果进行程序测试。

比如给定程序要求：从键盘输入一个数 n ，计算这个数的阶乘 $n!$ 并输出。

如果程序运行的结果是 $3! = 6$ ，仅仅说明这个数据运算是正确的，并不表示程序一定完全正确。如果输入值是一个负数，程序会得到什么结果？如果输入比较大的数 20，程序是不是运行正确？

对于输入的测试数据，如果程序运行错误，则必须修改源代码，然后再编译、链接、运行并查看结果，直到测试数据都运行正确。

由于数据类型匹配和精度等问题，并不是用户输入的任意数据都能很好地处理，所以先需要保证用户输入的是某些范围的数据时能正确处理(如整数、实数、字符)，随后，不断扩大用户可输入数据范围，直到用户输入任意数据(任意符号)都可以正确处理。

开发一个完整的 C 语言程序的流程如图 1-16 所示。

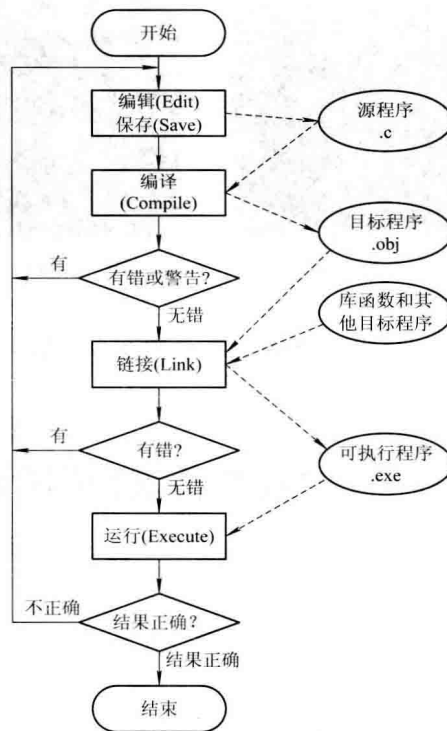


图 1-16 开发一个 C 语言程序的流程

开发一个 C 语言程序，先需要针对问题进行源程序的编辑并存盘，保存成扩展名为 `.c` 的文件(`.c` 源程序文件是一个文本文件，可以使用任意文本编辑器进行编辑)。然后对 `.c` 源程序文件进行编译，编译正确后编译器会生成扩展名为 `.obj` 的二进制目标文件(一个 `.c` 源程序会对应生成一个 `.obj` 目标文件)。之后，链接将 `.obj` 文件和其他库函数及其他目标文件生成一个扩展名为 `.exe` 的可执行文件(多源程序文件会有多个 `.obj` 文件)。程序链接成功后，还要运行测试，如果测试结果不正确(或者没有达到目的)，还需要修改源程序，重新进行编译和链接。

1.4.2 使用 GCC 运行 C 程序的步骤

使用 `vi` 命令编辑器编写 C 源程序并存盘，如图 1-17 所示。

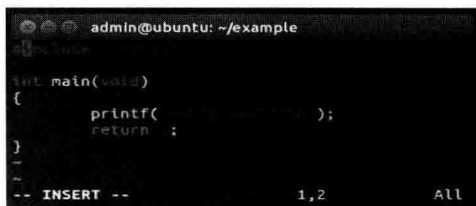


图 1-17 用 vi 编辑器编辑 C 源程序界面

源程序编辑完成并存盘后，在终端使用命令：

`gcc 源程序文件名-o 目标文件名`

对 C 源程序进行编译和链接，如果程序正确可以得到可执行目标文件名，如图 1-18 所示。

在终端使用命令运行程序，查看程序的运行结果，如图 1-19 所示。

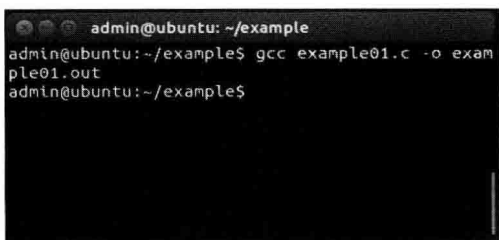


图 1-18 Linux 下对 C 源程序进行编译和链接的终端界面

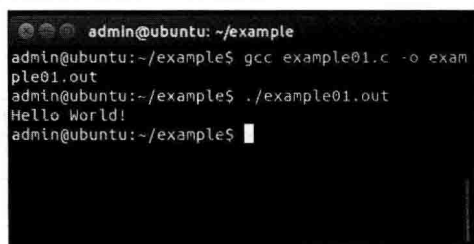


图 1-19 Linux 下运行 C 程序的终端界面

使用 GCC 调试程序时，调试跟踪等都需要使用命令行带参数的形式，在这里就不一一介绍了。

1.5 简单的 C 语言程序介绍

例 1-2 从键盘输入两个整数，输出这两个整数的和。

```
#include <stdio.h>

int main(void)
{
    int number1, number2, sum;
    printf("Input two number:");
    scanf("%d%d", &number1, &number2);
    sum = number1 + number2;
    printf("sum = %d\n", sum);
    return 0;
}
```

程序运行后，等待用户输入，用户输入两个数后，程序会输出这两个数的和。运行结果如图 1-20 所示。

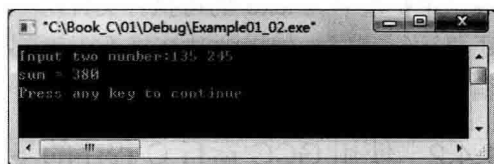


图 1-20 例 1-2 程序运行结果

例 1-3 绘制图形。

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

int main(void)
{
    initgraph(400, 400);           //初始化为图形模式，窗口大小为 400 * 400
    circle(200, 200, 75);         //以(200,200)为圆心，绘制半径为 75 的圆
    circle(200, 225, 25);
    circle(190, 225, 5);
    circle(210, 225, 5);
    circle(175, 175, 15);
    circle(225, 175, 15);
    circle(140, 135, 30);
    circle(260, 135, 30);
    getch();
    closegraph();                 //关闭图形模式
    return 0;
}
```

VC 在 C++ 中专门有一套用于绘制图形的方法库。在这里，笔者选用 EasyX 图形工具包来绘制图形。由于该工具包需要使用 C++ 面向对象的一些特性，文件命名时后缀需要是 .CPP。

在编译程序前，需要对环境进行配置。打开“Tools”菜单，如图 1-21 所示。执行“Options...”命令后，会打开“Options”对话框，选择“Directories”选项卡，如图 1-22 所示。

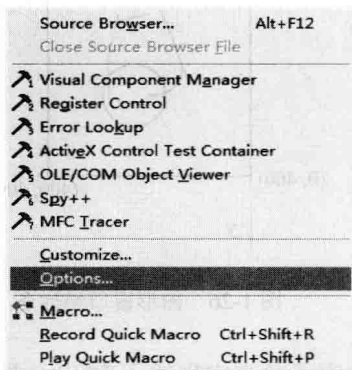


图 1-21 VC++ 6.0 “Tools”菜单

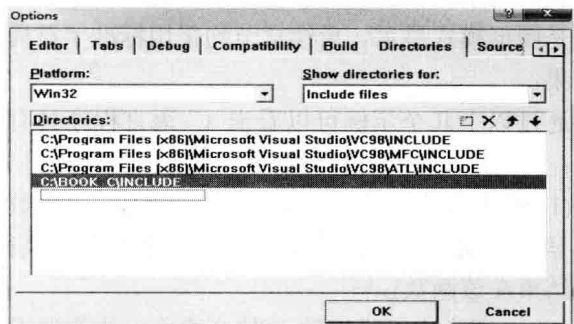


图 1-22 “Options”对话框“Directories”选项卡的
“Include files”表项

在“Show directories for:”下拉列表中选择“Include files”表项，如图 1-23 所示。此时点击“Directories:”列表框最后一行的虚框，选中变蓝后再单击，就可以添加 EasyX