

HOPE

Microsoft C6.0 Quick C2.5版

程序设计教程

沈炳范 罗 成 译

QuickC集成开发环境教程：编辑、编译、调试、运行和选择项的设置

程序设计教程： 简单类型及运算 预处理程序的使用

用 C 进行输入输出 流程控制

函数的编写 原型的使用

变量及其作用域和存储类型

指针、数组、字符串的使用

文件和内存的操作 操作符的使用

库函数的使用 C中的图形



北京希望电脑公司

73.87221
1109

Microsoft C 6.0 QuickC 2.5

程序设计教程

沈炳范 罗成 译



QuickC集成开发环境教程：编辑、编译、调试、运行和选择项的设置

程序设计教程： 简单类型及运算 预处理程序的用法

用 C 进行输入输出 流程控制

函数的编写 原型的使用

变量及其作用域和存贮类型

指针、数组、字符串的使用

文件和内存的操作 操作符的使用

库函数的使用 C中的图形

北京希望电脑公司

一九九一年十月

前 言

本书是 **Microsoft C** 和 **QuickC** 的程序设计的教程。循序渐进地让读者学习编辑、编译、调试、运行和选择项的设置的集成开发环境的用法。然后通过介绍简单类型的运算、预处理程序的用法、用 **C** 进行输入输出、流程控制、函数的编写、原型的使用、变量及其作用域和存贮类型、指针数组字符串的使用、文件和内存的操作、操作符的使用、库函数的使用和 **C** 中的图形，让读者进行程序设计。对 **C** 稍有一点基础的人都可以阅读此书。

C 语言正作为通用语言来普及。**C** 正在代替 **BASIC** 语言。

编者

目 录

前言

第一章 C 语言简介	1
§ 1.1 C 的特征	1
§ 1.1.1 性能特征	1
§ 1.1.2 可移植性	2
§ 1.1.3 模块化	2
§ 1.2 C 编程的一个例子	2
§ 1.3 C 的特点	3
§ 1.3.1 一个小规模的语言	3
§ 1.3.2 C 的能力	4
§ 1.3.3 标准版本	4
§ 1.4 使用 C	4
§ 1.4.1 编译和链接	5
§ 1.4.2 库文件	6
§ 1.5 Why QuickC?	7
第二章 快速浏览 QuickC	8
§ 2.1 掌握 QuickC	8
§ 2.1.1 一个快速介绍	8
§ 2.1.2 修改程序	10
§ 2.1.3 分隔程序元素	13
§ 2.1.4 复习(Review)	17
§ 2.2 讨论程序的执行	17
§ 2.2.1 调试准备	19
§ 2.2.2 一个更深的例子	21
§ 2.3 QuickC 的命令	27
§ 2.3.1 在线提示	27
§ 2.3.2 建立和编辑文件	27
§ 2.3.2.1 编辑方式	27
§ 2.3.2.2 移动你的文件	28
§ 2.3.2.3 标记和移动文本	30
§ 2.3.2.4 更多的 QuickC 的特色	35
§ 2.3.2.5 使用文件菜单的文件操作	35
§ 2.3.2.6 Search 菜单	36
§ 2.3.2.7 Find 选项	37
§ 2.3.2.8 Change 选项	39
§ 2.3.3 Run 菜单	40

§ 2.3.3.1	开始和继续执行程序	40
§ 2.3.3.2	调试你的程序时编译和运行	40
§ 2.3.4	指定编译程序选项	41
§ 2.3.4.1	选择内存模式	42
§ 2.3.4.2	说明警告级别	42
§ 2.3.4.3	确定一个语言版本	43
§ 2.3.4.4	调试标志	43
§ 2.3.4.5	解除标志	43
§ 2.3.4.6	其它方面的设置	44
§ 2.4	总结	44
第三章	简单类型和操作	45
§ 3.1	C 程序结构(C Program Structure)	45
§ 3.1.1	注释说明	46
§ 3.1.2	C 的 main() 规则	46
§ 3.1.3	printf() 函数	47
§ 3.1.4	Escape Codes(转换代码)	49
§ 3.2	语句	49
§ 3.2.1	简单语句	49
§ 3.2.2	复合语句	50
§ 3.3	标识符	50
§ 3.3.1	C 中命名规则	51
§ 3.3.2	不要用关键字来做标识符	52
§ 3.4	定义变量: 开辟空间	52
§ 3.4.1	赋值语句	53
§ 3.4.2	在变量定义时初始化	54
§ 3.5	简单数据类型	54
§ 3.5.1	char 类型	54
§ 3.5.2	int 类型	55
§ 3.5.3	float 类型	55
§ 3.5.4	double 类型	57
§ 3.5.5	整型和浮点型类型	59
§ 3.5.6	讨论简单类型	59
§ 3.6	简单类型的变体	63
§ 3.6.1	常量	65
§ 3.6.2	整数常量	66
§ 3.6.3	字符常量	67
§ 3.6.4	字符串常量	68
§ 3.6.5	浮点和双精度常量	68
§ 3.7	算术运算符	69
§ 3.7.1	使用浮点类型的运算符	69

§ 3.7.2	使用整型的运算符	69
§ 3.7.3	二元和一元运算符	70
§ 3.7.4	运算符优先权	70
§ 3.7.5	混合类型的操作	71
§ 3.8	printf()小结	73
§ 3.9	总结	74
第四章	预处理和程序	75
§ 4.1	程序结构	75
§ 4.2	C 预处理程序	78
§ 4.2.1	普通的预处理规则	79
§ 4.3	预处理命令	80
§ 4.3.1	#include 命令	80
§ 4.3.2	#define 命令	81
§ 4.3.3	显示常量	81
§ 4.3.4	带参数的宏	83
§ 4.3.5	预处理宏的决策产生	86
§ 4.3.6	#undef 命令	88
§ 4.3.7	#ifdef 和 #else 命令	88
§ 4.3.8	#if 和 #endif 命令	89
§ 4.3.9	其它预处理命令	89
§ 4.4	库函数	89
§ 4.5	类函数宏	94
§ 4.6	头文件和其他包含文件	95
§ 4.7	总结	96
第五章	C 中的读写	97
§ 5.1	读字符	97
§ 5.1.1	getchar()和 fegetch()	99
§ 5.1.2	getc()、ungetc()和 fgetc()	100
§ 5.1.3	putchar()和 fputc(), putc()和 fputc()	100
§ 5.1.4	putch(), getch()和 getche()命令	101
§ 5.2	更多的关于 printf()	102
§ 5.2.1	整数类型	102
§ 5.2.2	浮点类型	102
§ 5.2.3	域宽和精度	103
§ 5.2.4	printf() 的标志	104
§ 5.2.5	fprintf()函数	105
§ 5.3	值和变量: 地址运算符	105
§ 5.3.1	scanf()	107
§ 5.3.2	删掉值	110
§ 5.4	小结	111

第六章 流程控制	112
§ 6.1 关系和相等运算符	112
§ 6.1.1 关系运算符	112
§ 6.1.2 相等运算符	113
§ 6.2 逻辑运算符	113
§ 6.2.1 二元逻辑运算符	113
§ 6.2.2 一元取负运算符	114
§ 6.3 if-else 结构	114
§ 6.3.1 注意 else 语句的使用	117
§ 6.4 Switch 语句	119
§ 6.5 其它的运算符	123
§ 6.5.1 复合赋值运算符	123
§ 6.5.2 递增和递减运算符	124
§ 6.6 C 中的循环	126
§ 6.7 while 循环的组成部分	127
§ 6.7.1 关于 while 循环的其它问题	129
§ 6.8 do while 循环	130
§ 6.8.1 for 循环	131
§ 6.8.2 在 for 循环中省略表达式	136
§ 6.8.3 逗号运算符	139
§ 6.9 break 和 continue 语句	140
§ 6.10 小结	142
第七章 C 中的函数	143
§ 7.1 C 函数	143
§ 7.2 控制流程	144
§ 7.3 函数返回	149
§ 7.4 形参和实参	152
§ 7.5 总结	158
第八章 函数原型	159
§ 8.1 模型对形参语法的影响	162
§ 8.2 语法：新旧之别	166
§ 8.3 函数：各种观点	167
§ 8.4 小结	167
第九章 域、作用域和存贮类型	168
§ 9.1 域和可见性	168
§ 9.1.1 参数和域	171
§ 9.1.2 在块中定义变量	173
§ 9.2 变量的持续时间	175
§ 9.3 存贮类型	176
§ 9.3.1 静态存贮类型	176

§ 9.3.2 自动存贮类型	179
§ 9.3.3 寄存器存贮类型	180
§ 9.3.4 外部存贮类型	181
§ 9.4 省缺的存贮类型和存贮期限	182
§ 9.4.1 typedef 存贮类型	183
§ 9.5 小结	185
第十章 指 针	186
§ 10.1 目标, 定位值, 值	187
§ 10.2 指针	188
§ 10.2.1 指针的语法	190
§ 10.2.2 指针和地址	193
§ 10.2.3 为什么用指针	194
§ 10.3 做为函数参数的指针	194
§ 10.4 指针的算术运算	199
§ 10.5 各种观点	203
§ 10.6 指针易出的错误	204
§ 10.6.1 指针别名	204
§ 10.6.2 被错误引用的指针	205
§ 10.7 总结	205
第十一章 数组和字符串	206
§ 11.1 数组的基本知识	206
§ 11.1.1 数组: 存贮类型和初使化	207
§ 11.2 数组和指针	209
§ 11.2.1 数组元素的访问	210
§ 11.2.2 数组和指针之间的差别	212
§ 11.3 数组参数	212
§ 11.4 字符串	218
§ 11.4.1 做为指向 char 类型的指针的字符串	221
§ 11.4.2 字符串库函数	223
§ 11.4.3 各种字符串函数	228
§ 11.5 多维数组	228
§ 11.5.1 多维数组和指针	233
§ 11.6 指针数组	237
§ 11.6.1 字符串数组	238
§ 11.7 总结	241
第十二章 函数的更多细节和更多的函数	242
§ 12.1 返回指针函数	242
§ 12.1.1 返回指针注意事项	244
§ 12.1.2 字符串注意事项	245
§ 12.2 错误处理和标准错误流	249

§ 12.3 往集成环境中增加函数	254
§ 12.3.1 已定义函数	254
§ 12.3.2 字符串函数	255
§ 12.3.3 面向金融的函数	262
§ 12.3.4 用调试来检查中间结果	269
§ 12.4 递归	269
§ 12.4.1 递归的代价	274
§ 12.5 总结	276
第十三章 命令行参数	277
§ 13.1 main()作为程序和作为函数	277
§ 13.2 什么是命令行参数	277
§ 13.3 DOS 的命令行参数	278
§ 13.3.1 echo 命令	278
§ 13.3.2 dir 命令	279
§ 13.4 main()的参数	281
§ 13.5 使用 main()中的参数	283
§ 13.6 试验命令行参数的程序	286
§ 13.7 总结	296
第十四章 文件和内存	297
§ 14.1 文件	297
§ 14.1.1 对文件的操作	298
§ 14.1.2 一个简单的记事本程序	299
§ 14.1.3 一个简单的读文件程序	300
§ 14.1.4 一个删除文件中窄的程序	302
§ 14.1.5 一个用空格替换制表符的程序	305
§ 14.1.6 文件处理概述	307
§ 14.2 动态存储分配	307
§ 14.2.1 C 映射操作符	308
§ 14.2.2 动态存储分配方法	309
§ 14.2.3 使用 malloc()	311
§ 14.2.4 calloc()	317
§ 14.2.5 申请更多的内存	317
§ 14.2.6 释放内存	319
§ 14.2.7 确定可用空间	319
§ 14.2.8 动态存储分配注意事项	321
§ 14.2.9 动态存储分配的用途	321
§ 14.3 总结	321
第十五章 位操作符和其它操作符	322
§ 15.1 按位操作符	322
§ 15.1.1 位模式	322

§ 15.1.2	按位求反操作符	324
§ 15.1.3	按位与操作符	327
§ 15.1.4	按位或操作符	331
§ 15.1.5	按位异或操作符	334
§ 15.1.6	移位操作符	338
§ 15.1.7	移位操作和 2 的幂	348
§ 15.2	条件操作符	353
§ 15.3	总结	356
第十六章 结构、联合和其它类型		357
§ 16.1	结构	357
§ 16.1.1	访问结构成员	359
§ 16.1.2	有关结构的规则	361
§ 16.1.3	结构指针	371
§ 16.1.4	结构数组和结构指针数组	374
§ 16.1.5	自引用结构	380
§ 16.2	位域	395
§ 16.2.1	位域限制	398
§ 16.3	联合	399
§ 16.4	结构的联合	401
§ 16.5	枚举类型	401
§ 16.6	总结	404
第十七章 Quick C 库函数		405
§ 17.1	stdio.h	405
§ 17.1.1	熟悉的函数	405
§ 17.1.2	新函数	407
§ 17.1.3	一个练习 I/O 函数的程序	409
§ 17.2	stdlib.h	413
§ 17.2.1	熟悉的函数	413
§ 17.2.2	新函数	414
§ 17.2.3	练习 stdlib.h 库文件中的函数	417
§ 17.3	string.h	423
§ 17.3.1	熟悉的函数	423
§ 17.3.2	新函数	423
§ 17.3.3	练习函数	426
§ 17.4	ctype.h	436
§ 17.4.1	转换子程序	436
§ 17.4.2	练习字符过程	437
§ 17.5	dos.h	441
§ 17.5.1	背景	441
§ 17.5.2	使用 DOS 中断的注意事项	448

§ 17.6	io.h	448
§ 17.6.1	熟悉的函数	448
§ 17.6.2	新函数	448
§ 17.6.3	练习一些 io.h 函数	449
§ 17.7	math.h	451
§ 17.7.1	熟悉的函数	451
§ 17.7.2	新函数	452
§ 17.8	其它文件	455
§ 17.9	总结	455
第十八章	QuickC 中的图形	456
§ 18.1	这一章其它部分的要求	456
§ 18.2	设置图形模式	456
§ 18.3	屏幕, 窗口和视口	460
§ 18.3.1	物理屏幕	460
§ 18.3.2	视口	460
§ 18.3.3	窗口	463
§ 18.3.4	在轴上设置标记	465
§ 18.4	图形模拟结果	467
§ 18.4.1	扔硬币	467
§ 18.5	一起使用窗口和视口	470
§ 18.5.1	各种问题和建议	474
§ 18.6	画几何图形	474
§ 18.6.1	使用多窗口时的比例因子	476
§ 18.6.2	各种问题和建议	477
§ 18.7	一个探索程序	477
§ 18.8	总结	494
附录 A	ASCII 码	495
附录 B	QuickC 命令概述	499
§ B.1	调用 QuickC 的命令	499
§ B.2	QuickC 环境中的有用命令	500
§ B.3	编辑命令	500
§ B.3.1	移动命令	500
§ B.3.2	选择命令	501
§ B.3.3	插入命令	501
§ B.3.4	删除命令	501
§ B.3.5	其它命令	501
§ B.4	菜单命令	502
§ B.4.1	文件菜单(ALT-F)	502
§ B.4.1.1	文件处理命令	502
§ B.4.1.2	各种文件命令	502

§ B.4.2	编辑菜单(ALT-E).....	502
§ B.4.3	视口(VIEW)菜单.....	502
§ B.4.4	查找菜单(ALT-S).....	503
§ B.4.5	Make 菜单(ALT-M).....	504
§ B.4.6	运行菜单(ALT-R).....	504
§ B.4.7	调试菜单(ALT-D).....	505
§ B.4.8	实用菜单(ALT-U).....	506
§ B.4.9	选择菜单(ALT-O).....	506
§ B.5	在 QuickC 环境外部编译和连接程序.....	508
§ B.5.1	内存模式等.....	509
§ B.5.2	控制预处理器.....	509
§ B.6	控制连接.....	509
§ B.6.1	建立独立库.....	510
§ B.6.2	使用独立库.....	515

第一章 C 语言简介

C 是一种优雅、简练的语言，它对程序员也是一个挑战。有时你会喜爱 C，有时你又会憎恨它。

C 有许多可爱之处：它规模小、功能强，若使用得当时，它快速、有效并且可移植，这意味着它可以方便地从一种计算机或操作系统移植到另一种上去。

C 可让你做神奇的事情，象建立编译程序、操作系统和编辑程序。C 甚至可以改写难以改写的编译程序、操作系统和编辑程序，或者其它可能存在你系统的内存单元里的任何东西。

C 给你象 Pascal 或 Modula-2 这样的高级语言的能力和潜在的可移植性，还有象汇编语言那样的低级语言的灵活性和潜在的破坏性。C 让你建立其它高级语言所具有的复杂数据结构，并以通常只有汇编语言才可能有的方式对数据结构的独立位进行操作。

然而 C 自身不会保护你的程序，如果你不注意它就会毁坏你的数据。另一方面，你可以用 C 做所有种类的事情，只要你没有太多的幻想或失望，就不会比你使用其他程序语言更麻烦。

任何一种语言——不论英语或 C——都有一套定义语言的规则。例如，在英语中一个句号结束一个句子。这样一套规则被称为是该语言的语法。学习语法是 C 的简单部分，学习怎样最充分地使用语言是你的真正任务。你所得到的好处包括编程能力和对计算机的更深入了解，它们与你付出的努力的确是相配的。

Microsoft 公司的 QuickC 通过提供一个完整的、便于使用的语言实现使 C 趋于商品化。QuickC 包括一个增强的编译程序(它把程序转换成目标代码)，一个链接程序(它产生运行文件)，一个 SHELL(它作为一个命令解释程序服务)以及一个丰富的编程环境。有了这本书的帮助，你就能充分利用 QuickC 特有的特征，使 C 为你工作。

§ 1.1 C 的特征

种种特征使 C 成为对大小项目的一个非常有吸引力的程序语言。这些特征包括象速度和可执行程序的效率的性能以及与程序的转换能力和模块化有关的特征。模块化使你能把大的程序分成小的易管理的部分，使得能在其他程序中重复使用之。当然，一个特定的程序利用 C 的这些特点的程度还取决于程序编写的好坏程度。本书中的例子为你开发好 C 程序提供了一个坚实的基础。

§ 1.1.1 性能特征

C 比 Pascal、Modula-2 甚至 BASIC 规模小，只有大约 30 个关键字或保留字。因为 C 的许多运算接近于汇编水平，使用 C 可以提高你程序的运行速度并且易于建立简明的优化表达式。

C 还给你一整套工具，充分发挥它的潜力是你的工作。因为 C 使你非常接近计算机中的实际存贮单元和寄存器，你几乎可以在一个 C 程序中做任何事情。这种能力使 C 功能强大——这于对复杂项目或需要充分利用运行系统的程序是非常理想的。

§ 1.1.2 可移植性

C 非常便于移植,原因之一是有预处理程序,这就可能在运行或编译一个程序时替代一种特定机器的特定信息。就象你在后面的章节中会看到的那样,你可以用预处理程序帮助你的程序运行得更快。

§ 1.1.3 模块化

促使 C 可移植的一个特征是它有用已编译过的小规模函数和库模块来建立程序的能力。库模块可能包括针对一种特定机器的 C 函数的说明。因为你能够建立和编译独立的函数库, C 允许你用模块化方式编写程序。你可以把大程序分成小的函数单元,其中包括在别的应用中可以一起使用的函数。因为独立模块是分别地建立和修改的,你可以用一个策略使移植程序到别的机器上变得更容易。

§ 1.2 C 编程的一个例子

下述列表展示了一种从其他文件合并信息到一个程序的方法,也列举了一种提高程序可移植性的方法:

```
/* Program showing the use of an include file with
   program specific message. Each program can have its own
   system or program specific include file.
*/

/* a file containing various definitions. */
#include "proginfo.h"

main() /* the main C program */
{
    /* write value of PROG_MESSAGE to screen */
    printf( "%s\n", PROG_MESSAGE);
}
```

C 的主程序即函数 main() 只在屏幕上显示出某些名为 PROG_MESSAGE 的值的消息。

写屏是由 QuickC 以及其它 C 实现为你预定义的函数 printf() 做的。函数 printf() 能够写出所有你让它写的信息。在这里, 程序告诉它去写 PROG_MESSAGE 的值。

当它遇到\n 时, printf() 写一个回车和一个把光标置于下一行的最左列的换行符。

% 开始一个在 printf() 写内容时其值被替代的特定信息类型的占位符, s 表明将写一个字符串。在本例中, %s 是包含一个字符串的 PROG_MESSAGE 的占位符。在 C 中, 字符串是我们以后将讨论的以一个特定字符结束的字符序列。

ESC 代码\n 是用特定的方式移动光标的一种简单方法。后面我们会更详细地学习关于 printf() 的语法, 包括怎样写其它类型的数据(象整数和实数), 并且会有更多的象\n 这样的 ESC 代码。

main() 函数的结构表明了 C 中函数的格式。一个函数开始于一个名字(这里是 main),

后面跟着括号(其中可能有某些内容), 函数的主体在{和}中。

在函数体内部, 独立的语句以分号结束。注释可以在程序的任何地方出现, 并且包含在/*和*/之中。

注意在这个程序中的任何地方都没有定义 `PROG_MESSAGE`, 相反, 这个值在编译程序处理包括主程序的文件时读入的头文件 `proginfo.h` 中被定义。这样的头文件在 C 中是很普遍的, 每一个系统或程序都可以有自己的头文件, 这些文件包括程序或操作系统特有的值和定义。例如, 在 QuickC 下运行, 程序的信息为 "QuickC Version 2.5 "; 在 Microsoft C 版本 6.0 下运行时, 信息是 "Microsoft C Version 6.0", 这些信息存贮在象 `proginfo.h` 这样的文件中。QuickC 样本文件 `proginfo.h` 的内容将在后面的章节中展示。

多数 C 编译程序提供一个叫做 `stdio.h` 的文件, 它包含系统说明和信息说明的数据, 这些数据把值(如 "QuickC Version 2.5")和名(如 `PROG_MESSAGE`)联系起来。查看你的 QuickC 的目录中有 `.h` 扩展名的文件, 那些就是你的编译程序提供的头文件。

§ 1.3 C 的特点

C 被描述成一个中级语言, 具有高级和低级语言的特点和性能。这就是它具有明显矛盾的性能的一个原因。C 允许你象其它高级语言做的那样建立复杂的数据结构, 但不同于这些语言, C 在你使用这些变量时只做最少的范围和类型核对。另一方面, C 让你看到内存中特定地址的存贮单元中的内容; C 甚至让你用地址进行算术运算。这种能力只在汇编语言中能够达到, 而在高级语言中很少有。

C 还有许多杰出特点的原因, 是它被设计成一种灵活的、随意的、快速的和可移植的语言。Dennis Ritchie 开发了 C, Brian Kernighan 用 C 做了许多前期工作, 旨在用 C 编写系统(如操作系统)以及应用程序(如编辑程序)。因此, 创建者给了 C 汇编语言速度和灵活性, 以及高级语言的数据结构和控制结构(象 `if-then, for, while`)。

表 1-1 列出了一些高级和低级语言的特点, 并且指出了就这些特点而论, C 处于什么位置。注意 C 几乎具有所列出的每个特点。

高级语言	C		低级语言
结构化	有	有	快速性
易于使用	有	有	紧密性
安全性	有/无	有	高效性
复杂的数据结构	有	有	地址和位运算

表1-1 用高级和低级语言的特点来衡量C

§ 1.3.1 一个小规模的语言

C 的词汇非常少——只有大约 30 个关键字。虽然词汇很少, 但可以灵活组织它们。小规模词汇并不意味着弱的语言, 某些语言学家断言用英语有效地交流需要的词汇不超过 900 个。

另一方面, C 有许多运算符和用于说明运算的符号。实际上, C 有比关键字更多的运算符! 这种少量词汇和多种运算符的结合让你象用一个构造集合一样使用 C。可以容易地

建立独立的语句或函数去做适当的工作。正如你将要看到的那样，C 程序通常由小的能完成独立任务的函数组成，这些函数被收集在模块中并且在需要时使用。

§ 1.3.2 C 的能力

尽管 C 规模小，它在编程中正被大规模地使用着，功能强大的 UNIX、DOS 和 OS/2 操作系统的绝大部分都是用 C 编写的。同样，C 已被用来写编译程序(包括 C 编译程序)，编辑程序，排版程序(象 troff 和 TeX)以及 Dbase III 和 Microsoft Windows 这样的程序。

在这一系列的另一端，你会发现许多完成特殊任务的 C 程序。象显示任一年或月的日历，计算一个文件中单词的数目，列出一个目录中的内容，在特定的文件中找出一个特定的单词或表达式，在一个借贷中计算在不同的条件下的利息偿付等等。许多这些程序的源代码可从用户组或便宜实用的软件包中的电子公告报上得到。你可能想从中看或用某些程序、去学习和运用这些代码，后面的章节中我们会看到一些这样的程序。

§ 1.3.3 标准版本

不管你相信与否，C 至今没有正式的标准。虽然 C 被广泛地使用，但还未被正式地定义，不过也不要把事情想得那样糟糕。直到几年前才有这种语言的一个事实上的标准版本，它被概括在第一次出版的 <<The C Programming Language>> (Prentice-Hall, 1978) (作者 Brain Kernighan 和 Dennis Ritchie)。从那时起，这个 ANSI 标准草案规定的语言版本已被作为语言的日常定义。多数目前已发行的编译程序——包括 QuickC 和 Microsoft C 都遵从这个标准草案。实质上，C 的正式名字是 ANSI 标准 C (ANSI 表示美国国家标准协会)。

这个委员会的任务是消除分歧并且使移植 C 程序更容易(在不损失 C 的任何能力的前提下)。委员会还增加了一些高级语言特点帮助程序减少错误。从这个委员会反复审议中形成的 C 的版本，其功能将比以前更加强大。

这本书中，QuickC 自身以及大多数新的 C 编译程序将遵从这个 ANSI 标准草案。

§ 1.4 使用 C

从设想到完成任务，C 程序要经过几步，每一步都会修改或扩充程序。下一章你将学习怎样编写 C 程序并使之在 QuickC 下运行。这节概括了一个 C 程序的主要组成部分和从文本文件到运行程序所涉及的主要任务。我们将看到 QuickC 提供的工具在一个 SHELL 程序下完成各种任务提供的环境。这使得从程序建立过程的一个状态到另一个状态变得容易。

当建立一个 C 程序时，将建立和使用各种类型的文件：包括用文本编辑程序建立的源文件和头文件，与编译过的主程序相链接了的库函数和其他目标文件，以及运行程序的可执行文件(在这章的后面讨论)。

第一步是以文本形式建立 C 程序——这就是源文件。下面举出了一个 C 程序的源文件的例子：

```
/* sample C source file.
```

```
    This program add two numbers and writes sum to the screen. */
```

```
/* file containing program specific information */
```

```
# include "proginfo.h"
```

```

main( )      /* main program function */
{
    /* definition of a integer variable, named result */
    int result;
    Printf("%s\n", PROG_MESSAGE);
    result=7+9; /* add 7+9, assign sum to result */
    /* write result on screen */
    printf("sum= %d\n", result);
}

```

这个程序的输出是:

QuickC Version 2.5

Sum = 16

文件包括调用了 printf() 的主程序 main()。就象我们第一个例子所做的, 这里也有一个指令(# include "proinfo.h")读入头文件 proinfo.h 中的内容。

还有一些新的特点, 注意到这个程序还写了表示数量的信息即变量 i1 的值, %d(代表十进制)占位符指出了这个事实。在第二次对 printf() 的调用中代入的是 i1 的值, 它在做完加法之后是 16。

第二个参数即传给 printf() 的数据, 用来代替在函数中第一个参数的占位符。这个简单的替代策略, 扩展了处理另外参数的能力, 这点不久你就会看到。这个程序还列举了在 C 中怎样定义一个变量, 换句话说, 就是怎样去把一个数据类型和变量名联系起来(如果你不习惯用数据类型来工作并且对它们还不清楚, 那也不必担心, 数据类型将在第三章中详述)。变量 i1 被定义成 int 类型, 表示整型。

最后, 语句

i1=7+9

把 7+9 的结果赋给 i1。在第三章中将学习变量定义和赋值。

§ 1.4.1 编译和链接

包含列表的源文件被 QuickC 编译程序处理, 它通过把程序翻译成另一种形式来建立另一种版本的文件。在编译过程中, QuickC 读入头文件 proinfo.h 并完成所要求的替代和定义。

源文件包括程序的函数所需要的定义, 并且可能包括被许多函数使用的变量的定义, 也可能有读入另外的头文件或其它文件的指令。程序将有一个主源文件和许多附加的包含文件。

头文件是用户编写的或编译程序提供的普通的文本文件。这些文件通常包括用特定的计算机或用特定的编译程序来完成所需的信息。与这些信息相关的是助记符, 以简化难以理解的数值的记忆。当使用这些名字时, 编译程序用在头文件中找到的每一个名字的值来替代它们, 例如, 头文件 proinfo.h 在 QuickC 中包括下列定义:

```

/* Partial contents of file, proinfo.h for Quick C programs */
#define PROG_MESSAGE "Quick C Version 2.5" /* C Version name */
#define MAX_INT 32767 /* largest integer value in system */

```