C语言设计界面大全

李论　姚远　等编

机械工业出版社

C语言设计界面大全

李论　姚远　等编

机械工业出版社

(京)新登字 054 号

本书向读者提供一个最新用户屏幕界面设计方法。由于本书是用 Turbo C 和 Quick C 编写程序，因此本书具有良好的通用性。本书共分九章，包括以下内容：Turbo C 和 Quick C 简介，用户界面设计，存取显示适配器，窗口和屏幕函数，菜单设计，数据输入屏幕，列表选择，目录函数，求助屏幕。

本书适用于高等院校师生和相关领域的工程技术人员。

本书附全部应用程序软盘一张，定价 40 元，欲购者请与本书责任编辑联系。

C 语言设计界面大全

李论　姚远　等编

*

责任编辑：蒋　克　版式设计：霍永明

封面设计：肖　晴　责任校对：肖新民

*

机械工业出版社出版（北京阜成门外百万庄南街一号）

邮政编码：100037

（北京市书刊出版业营业许可证出字第 117 号）

北京通县财联印刷厂印刷

新华书店北京发行所发行・新华书店经售

*

开本 787×1092 1/16・印张 20 1/2・字数 505 千字

1993 年 10 月北京第 1 版・1993 年 10 月北京第 1 次印刷

印数 0 001—5 400 ・定价：22.50 元

*

ISBN 7-111-03824-X/TP・189

前　言

C 语言作为一种具有高级和低级语言的共同特点并以它鲜明的特性和独特的设计而受到程序设计者的普遍重视。用 **Turbo C** 和 **Quick C** 设计屏幕界面更具特色，不仅新颖别致而且更加适用。

众所周知：一个良好的用户界面开发系统对一个程序的认可以及其成功起关键作用。本书可帮助程序员实现功能卓越、直观易学的屏幕接口。

本书选材广泛、内容丰富，给 C 语言设计人员提供了最新的界面设计方法。对于缺乏 C 语言知识的人，本书特给出了 **Turbo C** 和 **Quick C** 简介。对已经掌握 C 语言的读者，本书是一本极好的参考书。

本书主要由李论、姚远编写，张鸥、冯春燕协助，并特请中科院青年专家吴倩、杨仕润审阅全书。

本书是作者在中国科学院工作期间编写的，得到了领导的大力协助，在此深表谢意！

由于编者水平有限，不当之处在所难免，恳请广大读者和专家予以指正。

编　者

1992 年 6 月

目　录

第 1 章　Turbo C 和 Quick C 简介

1.1　Turbo C 简介

本节的主要目的是帮助用户尽可能快地熟悉 Turbo C，运行该编译器，并忽略该软件包中各组成部分所包含的复杂性。当用户安装了 Turbo C，并且编译及运行程序后，会发觉该系统提示丰富，极易使用，尤其是集成开发环境。

Turbo C 有一些复杂的软件包，实际上它包含两个不同的开发系统：从 MS－DOS 命令行执行的传统编译器和连接程序(tcc. exe 及 tlink. exe)；以及类似于 Turbo　Pascal 的集成开发环境(tc. exe)。在使用手册中这两个系统总是同时说明的。本章每次说明一个系统，但将会帮助用户区别这两个系统。

Turbo C 复杂的另一个原因是大量可用的选项。对于原来使用简单开发环境，比如：BASIC 或 Turbo Pascal 的程序员，文件数量、配置参数，以及文档页数可能开始时显得很吓人。但很快就会发现系统的优点并喜欢其灵活性。而且，当水平有了更高的长进后，会更适应于这些特点。对于那些使用过其它 C 编译器的有经验的 C 程序员，Turbo C 是非常好的一个工具，本节的内容将帮助用户以基本的、通常使用的选项集来启动系统。

本节中的过程假定用户已经能够编写 C 程序。也许用户有一个其它编译器下工作的程序要移植到 Turbo C 中；也许用户是个初学者，准备作一些练习。在任一情况下，即便最简单的程序也要求一些规定的步骤，这些步骤对于不同的编译器有很大的不同。因此，本章首先总结这些步骤。

本节第二部分主要描述该编译器的专门特性：与传统语言定义的区别以及与其它常用 C 语言运行的区别。该部分对于已经熟悉标准 C 语言定义，希望快速浏览 Turbo C 专门特性的有经验的 C 程序员特别有用。大多数的这些特性在相当低的机器级下操作。

1.1.1　建立 Turbo C 环境

以下说明安装和运行 Turbo C 的两个步骤：首先，在计算机上安装 Turbo C 系统；第二，使用基本步骤工作，包括编译和运行一个程序。为了能尽快地运行 Turbo C，重点在于说明建立一个基本的、通用的配置。系统开始工作以后可以加入另外的工具和选项，这在本书后面都有相应的说明。

1.1.1.1　安装 Turbo C

将 Turbo C 一号盘插入驱动器 A，转到 A：提示符下，并键入命令：

Install

以后按照屏幕上的提示换插盘，即可安装 Turbo C。

1.1.1.2　开发一个程序

系统安装好以后，就可学习使用 Turbo C 的基本技术。本节介绍编译和运行一个 C 程序的例子，来说明该编译器的基本特征及使用过程。虽然 Turbo C 编译包含许多选项和命令，此处总结的是一些最重要，并且是学习本系统的良好起点。程序先使用命令行方式编译，然后使用

集成化环境。因为大多数C程序包含多于一个的源文件(模块化编译是C的重要特点),例子程序包含两个文件:图1-1中所示的main.c和图1-2中所示的utility.c。main.c文件代表一些应用的基本文件,它调用文件utility.c中的函数RepeatStr。

```
#include <stdio.h>

char *RepeatStr(int Number,char Ch);

void main ()
    {
    printf("This is the main application,\n");
    printf("Sign your name on the dotted line ");
    printf("RepeatStr (40,'.'));

    } /* end main */
```

图1-1 一个C程序的例子

```
char *RepeatStr (int Number,Char Ch)
/*
    This function return a pointer to a null - terminated static string
    consisting of 'Number' copies of character 'Ch'. The string is changed with
    subsequent calls to 'RepeatStr'. Maximum value of 'Number' is 255.
*/
    {
    static char Buf[256];
    register int I=0;

    while (Number -- && I < 255)
    Buf [I++]=Ch;
    Buf [I]='\0';
    return Buf;

    } /* end RepeatStr */
```

图1-2 一个C模块例子

1. 命令行系统

(1)建立源文件:使用最熟悉的正文编译器输入并保存两个源文件main.c和utility.c。对于开发大型的C应用程序,最好使用多窗口、灵活的编译器,比如用Brief,这样的编译器能很方便地在文件之间删除和复制代码。对于双软驱系统,应转到B提示符下,因为此处包含用户

文件，对于硬盘系统，可以转入任一用户希望保存 C 文件的目录下，假定符合前面所述的路径和配置文件要求。

(2)编译并连接程序：使用 tc.exe 命令行编译程序编译和连接是很简单的。该程序不仅省去了单独的编译和连接步骤，而且自动地结合了适当的起始代码(c0s.obj)和库例程(emu.lib，maths.lib 和 cs.lib)。如果使用双软驱系统，前面创建的运行盘应在驱动器 A 中，用户应转入驱动器 B。对于硬盘系统，用户应继续转入存放这两个源文件的目录中。对任一种设置，在 DOS 提示符下键入下面的命令：

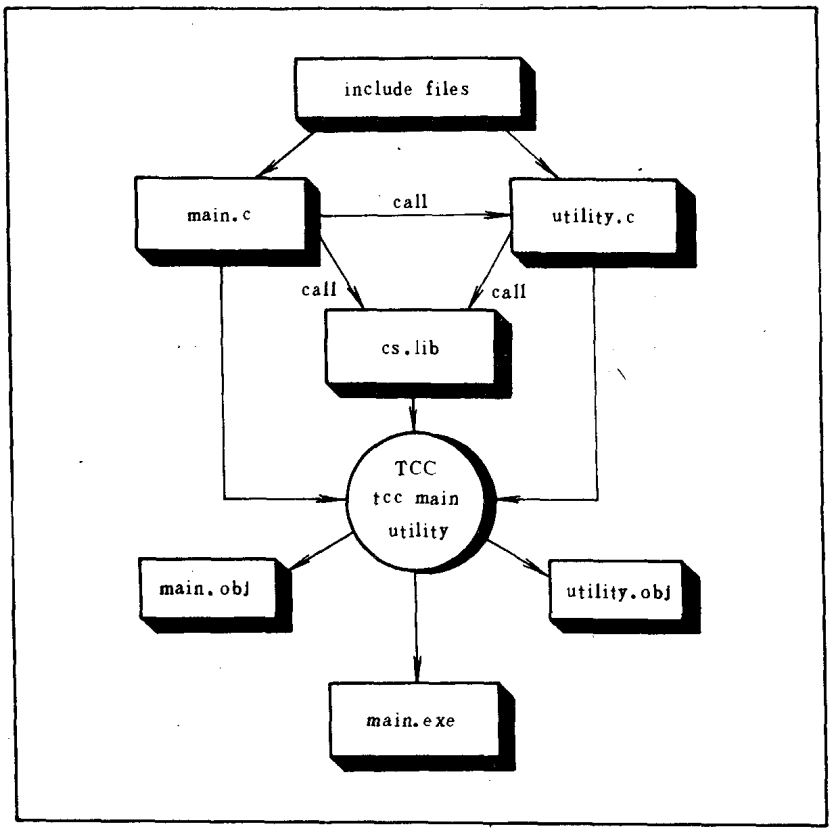

图 1-3　使用命令行系统开发实例程序

```
tcc  main.c  utility.c
```

编译程序将编译这两个 C 程序，然后自动地启动连接程序，传递以正确的命令行。结果文件是 main.obj，utility.obj 和 main.exe。现在键入：

```
main
```

注意每次运行 tcc 程序时，它都自动地重编译两个 C 文件，即使只修改了其中之一。使用命令行系统开发实例程序的过程总结于图 1-3 中。

2. 集成开发环境

本节介绍 Turbo C 集成环境，给出开发实例程序的基本步骤，然后是学习和使用该系统的一些基本要点。

要开发一个实例程序，有以下步骤：

* 启动集成环境
* 在编辑器中建立源文件
* 产生.exe 文件
* 运行程序

(1)启动集成环境

键入 tc 启动 Turbo C 的集成开发环境。

(2)在编辑器中建立源文件

启动 Turbo C 后，按 Alt－E 进入编辑器。该编辑器所用的命令几乎等同于 Turbo　Pascal

和 Sidekick，类似于 Wordstsr。若用户未用过这些编辑器，见附录 A 中 Turbo C 的参数指南，那里包含了所有的指令，首先键入图 1－1 中的源程序。输入完成后，按 Alt—F 进入文件菜单，然后按 S 保存该文件到盘上。系统询问是否替换默认名 noname.c，此时键入 main.c 并按回车键。

存入文件后，仍处于文件菜单。此时按 N 清除编辑器，准备输入一个新的文件。用同样的方法键入图 1－2 中所列的文件，以文件名 utility.c 保存，然后清除编辑器。

在程序编译以前还有一个文件需要建立。如果此时调回 main.c 执行编译和连接，Turbo C 无法知道存在另一个文件与该程序相关，报告以下信息：

Undefined symbol' - RepeatStr' in module main.c

因此，用户必须建立一个工程(project)文件来通知 Turbo C 组成该程序的模块工程文件(类似于 make 文件)。建立该文件很简单，只由两行组成：

main.c

utility.c

以文件名 main.prj 保存该文件。按 Alt—P，再按 P，然后键入 main 使 Turbo C 知道该工程文件名，注意如果整个程序仅由一个文件组成时不需要建立工程文件。

(3)产生.exe 文件

建立了工程文件后，可在任何时候编译和连接整个系统，通过简单地按 Alt—C 调出编译菜单，按 M(或按 F9)产生.exe 文件。此时生成可执行文件 main.exe。

注意使用了工程文件后，哪个文件当前处于编辑器中是无关紧要的，当要求 Turbo C 根据一个工程文件生成一个程序时，它自动地存取所有需要的文件，并输出.obj 和.exe 文件到磁盘上(不同于 Turbo Pascal，Turbo C，它不编译到内存，而总是写.obj 和.exe 文件到磁盘上)。

Turbo C 的工程机制不仅自动地生成多文件的应用程序，而且产生最新的.exe 文件。类似于 make 工具，它检查文件修改的日期，不重编译未修改的源代码文件。本书为所有的例子程序提供工程文件，并对该工具的另外一些特征作了说明。

(4)运行程序

要运行程序，在系统中按 Alt—R。注意 Alt—R 命令还自动完成将可执行文件变成最新版的编译和连接过程。因此不耐烦的用户可直接按 Alt—R，而不用前述命令生成.exe 文件。

3.学习集成环境的要点

经过了前面一节的基本步骤以后，学习集成环境其它特征的最好方法是经常使用帮助功能(按 F1 启动)来熟悉窗口和菜单。通常，在线帮助提供的内容与手册上的内容相同，更方便之处是通过选择相应的关键词来查询所需内容。要记住的一个重要命令是 Alt—F1，它使你获得当前屏幕的上一屏幕。使用 F1 和 Alt—F1 用户可以查看各页的内容。

Turbo C 集成化环境的另一个方便特征是可通过热键执行命令。这些键或是功能键，或是 Alt 键组合。与当前上下文相关的功能热键表位于屏幕的底行，若按住 Alt 几秒钟，则会出现相关的 Alt 键组合，并不是所有的有用键都出现于快速参考行中。

集成系统中其它的可用选项在本书高级技术部分讨论。图 1－4 中总结了在集成环境中产生可执行文件的步骤。

4.同时使用两种环境

虽然命令行编译器和集成环境是独立的系统，但有时在开发一个程序的不同阶段使用相应的一个是很有必要的。

例如，在多窗口的编辑器中建立一个大程序应用是很方便的，允许在文件之间删除和复制，同时浏览几个文件。文件编辑完成并存盘后，可用命令行系统编译(使用 Turbo C 提供的 make 程序)。如果存在许多编译错误或需要调整一些运行特征，可以用集成环境方便地消除错误及优化运行特征，此时集成环境的优点是快速修改/编译/运行，以及在源代码文件中自动定位编译错误(用户可能已注意到当键入源程序出错时系统的报错功能)。

1.1.2 Turbo C 的专门特性

1978 年 Kerighan 和 Ritchie 编写的《C 语言程序设计》说明了 C 语言的最低标准。几乎所有的 C 编译程序，包括 Turbo C 完全支持该分类标准。但 Turbo C 跟大多数最新的 C 编译程序一样，给该基本的语言定义增加了许多新的功能。大多数的这些功能与 ANSI 开发的联合标准一致。另外还增加一些功能不是 ANSI 标准的功能，或为 Turbo C 独有，或出现于另外的最新编译程序中。

这些特性包括新的数据类型，符号元素，预处理命令，以及一组标准库函数。多数的新编译程序，例如 Microsoft C4.0 和 5.0，实现了一些 ANSI 特性。Turbo C 则包含了其中的大部分功能。

本节不想说明所有的 ANSI 标准特性，因为太多，无法在此列出，而且其中许多是相当普通且一般书中都有述及。这里只集中讨论在 Turbo C 中出现较不普遍的 ANSI 特性以及非 ANSI 标准的特性。其目的是为已经使用过其它编译器的 C 程序员快速浏览这些特性，或帮助未曾注意到这些内容的初学者(非 ANSI 标准的特性将被指出)。

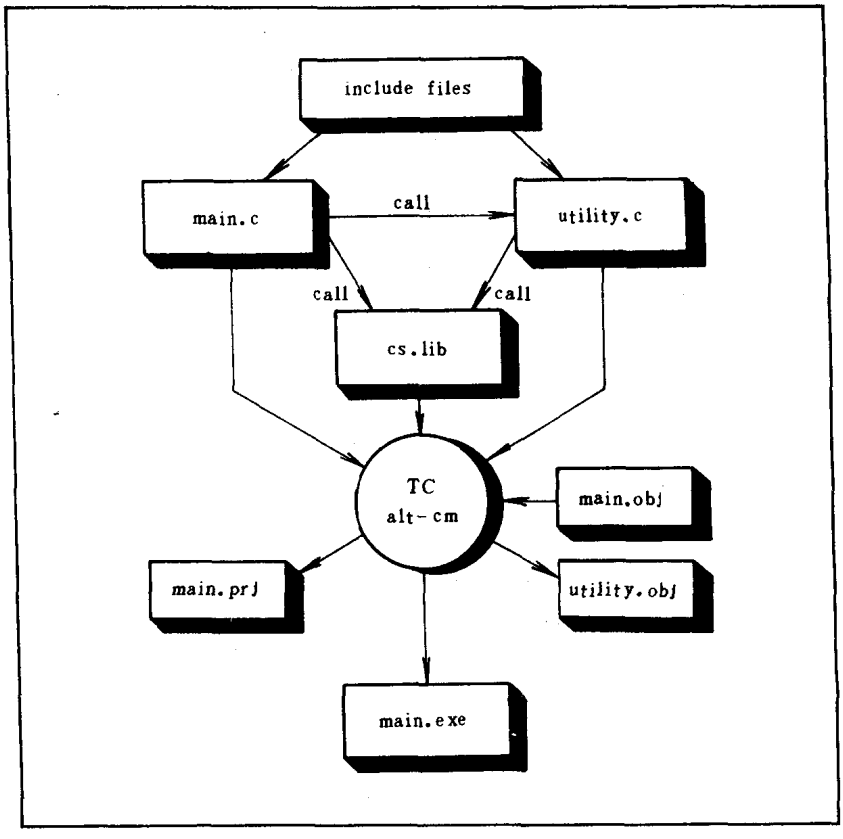

图 1-1 使用集成环境开发实例程序

1.1.2.1 函数原型

Turbo C 完全以 ANSI 标准实现函数原型。注意在 Turbo C 中使用函数原型是可选的，用户也可以将原型方法与传统方法结合使用。但要充分利用这种新格式，应使用下面两条规则来说明和定义函数：

1. 所有的函数先说明后再调用，不但要说明返回类型，而且要说明每个参数的类型，例如：

int prtposition(char * String,int Row,int Col)；标识符 String，Row 和 Col 仅仅是位置变量，其使用是可选的，但使用这些标识符有几个优点。首先，它们说明了每个参数的意义，

并在调用时可以避免参数位置出错。第二，使得复杂类型说明简单，因为它们按照标准的说明格式，并避免如 void(*)(void)之类古怪的类型说明，而用更易理解的方式来说明：

```
void  ( *  FuncPtr)(void)
```

第三，使用这些标识符允许函数说明正好符合函数定义的第一 行(假定按照下一条规则)。

注意可通过省略号(…)来指出 0 个或多个参数。例如，函数 int prinft (char * format,…)至少有一个类型为 char * 的参数，后跟 0 个或多个类型未定的参数。

2. 定义函数时，将参数和它们的类型一起放于参数表中，紧跟在函数名后。使用同一例子，函数定义具有以下格式：

```
int  prtposition(char  *  String,int  Row,int  Col)
{
      / * 函数体 * /
}
```

这种格式使得函数说明可直接从定义复制，防止不匹配(当它们位于同一个文件之中时，编译器将检查这两个参数列表是否匹配)。

通常将一些函数的原型说明放于一个头文件中，然后在使用这些函数的所有文件中包含该头文件。该方法有两个附加的重要优点。首先，若存在原型，编译器将为所有的函数调用检查参数的数量和类型。其次，编译器会根据原型提供的信息正确地将每个实际参数转为适当的数据类型，就象完成赋值(因此，参数的数据类型必须根据赋值语句的规则与说明的类型匹配)。例如，若一个函数说明为：

```
void  Func(double  Parm)
```

则在下面的函数调用中编译器正确地将 long 参数换为类型 double：

```
long  john =123;
Func  (john);
```

如果未提供原型，编译器会错误地传送一个 long 值的参数。当 far 类型地址与 near 类型地址必须相互转换时，这种参数类型之间的正确转换在混合内存模式程序中尤其重要，见《Turbo C 用户指南》中关于内存模式的讨论。要保证在程序中所有的函数调用都有相应的原型，使用—wpro 编译选项(或集成环境中适当的出错警告选项)来对所有无原型的函数调用发出警告。

1.1.2.2 新的类型修饰符

“类型修饰符”是关键字，它改变数据和函数说明的含义。Turbo C 提供了几个新的类型修饰符(不是按原始 Kernighan 和 Ritchie 标准)：

```
signed
const
volatile
pascal
cdecl
near
far
huge
_cs
```

_ds

_ss

_es

interrupt

1. Signed 修饰符

在 Turbo C 中，字符的默认类型是 signed。这特别意味着如果一个 char 类型数据项转换为一个整数，它将进行符号扩展；因此如果一个字符变量等于"\xff"，若将它赋给整型变量，该整数值等于-1，而不是 255。但有一个编译程序选项(对 tcc 是-K，对 tc 通过选项菜单)，使得所有字符被当作是 unsigned 值。当该编译器选项生效时，要使一个整数作为有符号值，则新的关键字 signed 可用作一个类型修饰符。例如：

```
signed  char  Ch;
```

2. const 修饰符

如果一个标准类型前有修饰符 const，则说明的变量不能被修改。例如给定以下说明：

```
const  NumCols=80;
```

语句：

```
++NumCols;
```

是非法的，若一个指针说明为 const，则指向的值可被修改，但包含在实际指针变量中的地址不能被改变。

3. volatile 修饰符

使用 volatile 修饰符限制编译器对变量作任何优化，假定变量保持其当前值，或使用该变量作临时存储。该修饰符适用于说明其值可能在不可预知的时候被外部过程修改的数据项(例如当发生中断时能修改变量的硬件中断处理程序)。下面是一个例子：

```
volatile  int  KeyReady;
KeyReady=0;
while(! KeyReady)
```

4. Pascal 修饰符

Pascal 修饰符不是 ANSI 标准的一部分，用于与其它语言兼容，或用于提高效率。该修饰符以三种方式影响一个标识符的说明。首先，在目标文件中外部标识符转换成大写，去除 C 语言中通常的大小写敏感性。其次，在目标文件中外部标识符开始不加下划线。最后，函数调用使用 Pascal 的调用习惯，即参数以函数调用中所列的同样顺序压入堆栈，且从函数调用返回时不恢复栈指针(栈指针由函数本身恢复)。下面的说明是一个例子：

```
Pascal  PritLine(char  *  String,int  Row,  int  Col);
```

5. Cdecl 修饰符

Cdecl 也不包含于 ANSI 标准中，它说明通常的 C 命令和调用格式可用于强制编译器对整个文件默认为 Pascal 约定格式(对 tc 通过—P 开关，对 tcc 通过选项菜单)。关键词 cdecl 对指定的标识符进行强制。例如，若—P 选项在起作用，下面的说明强制一个库函数为通常的 C 约定(不是使用 Pascal 原型定义并编译，因此要求标准的 C 约定)：

```
int  cdecl  printf(char  *  Format,…);
```

6. near，far 和 huge 修饰符

这些修饰符不是 ANSI 标准的一部分。它们同时影响数据指针和函数的说明，主要用于改变编译一个程序的当前内存模式。它们将在第 2 章中内存模式处理部分讨论。

7. __cs，__ds，__ss 和__es 修饰符

这四个修饰符是 Turbo C 专有，与给定指针结合说明使用哪个段寄存器。用这些修饰符说明的指针总是 near(见第 2 章)。例如：

```
int __ss  *  Stkptr;
```

指出该指针包含相对于栈段(ss)寄存器值的内存偏移量。若 stkptr 是间接的，则它存取当前栈段内的值。

8. interrupt 修饰符

interrupt 修饰符也不是 ANSI 标准的一部分，只用于函数说明，且指出该函数要保存所有的寄存器，设置 ds 寄存器为模块的当前数据段，并返回一个中断返回指令。见第 2 章和第 8 章中有关 Turbo C 中断函数的更多信息。下面是该修饰符格式的一个例子：

```
vold  interrupt  int08(vold);
```

1.1.2.3 结构/联合对齐

默认情况下，Turbo C 将结构或联合的所有域放于第一个可用地址处，而不管该地址是奇数还是偶数。但是编译选项__a(对 tcc 而言；对于 tc 则用相应的选项菜单项)将使得 Turbo C 做下面的工作：

1. 所有的结构或联合开始于偶数地址。
2. 将所有非字符域放于偶数地址。
3. 保证结构具有偶数个字节。

通常在必要时项 2 和项 3 加入额外字节起作用。

如果用户使用了__a 选项，要确保编译所有存取公共结构数据的源程序文件。而且，结构对齐对于从另外字对齐编译器移植到 Turbo C 的应用程序很重要(例如，Microsoft C4.0)。用户可能需要在 Turbo C 中强制字对齐，或重新安排根据结构编写的数据文件格式。

1.1.2.4 预处理命令

大多数的 Turbo C 预处理命令符合 ANSI 标准。但有几个是 Turbo C 特有的。

1. ANSI 预处理功能

Turbo C 支持一些 ANSI 标准的预处理命令：

- ##(Token 连接)
- #(转化成串)
- #error 语句
- #pragma 语句
- 预定义宏

(1)Token 连接

符号##将使出现于一个宏定义中的两个 Token 连接。例如，宏定义：

```
#define  ID  (x,y)  (x##y)
```

将使 ID(Var,2)扩展为 Var 2. 注意#之后的空格被去除，且连接在完成了所有的宏替换后进行(因此 Var 可以在与 2 连接以前被扩展为另外的内容)。

符号#使得一个宏参数“串化”，即转变为一个串。例如，给出宏定义：

```
#define  Hello  (x)  printf  ("Hello"  #x"\n");
```

HELLO(world)扩展为：

```
printf("Hello"  "world"  "\n")
```

变为：

```
printf("Hello  world  \n")
```

通过本章稍后所述的串连结过程实现。

(2)串中的宏参数

按照建议，Turbo C 不扩展出现于串中的宏参数或宏定义中的字符常数。

(3)#error 语句

#error 语句在前面条件语句为真时使编译程序不编译并打印特定的错误信息。例如，预处理命令系列：

```
#define  MAX500
#if  MAX>100
#error"MAX  is  too  large"
#endif
```

将导致编译程序停止并打印信息：

```
Error  noname.c  4:Error  directive:"Max  is  too  large"
```

该功能对标志某些程序错误很有用。

(4)#pragma 语句

#Pragma 语句由 ANSI 定义，允许程序员向编译器传送信息。可被传送的信息完全依赖于特定的编译器，如果编译器不认识#pragma 后的一条指令，它就忽略该语句。Turbo C 定义了下面的 pragma 语句：

- #pragma inline
- #pragma warn

inline 告诉编译器文件中存在汇编语句。见本节后面关于 asm 关键词的说明，第 2 章中有关于嵌入汇编代码的更多信息。

warn 允许用户控制编译器提高警告信息。例如，考虑下面的三行：

```
#pragma  warn+pro
#pragma  warn-eff
#pragma  warn.dup
```

第一行设置了没有原型函数的警告，第二行关闭无效代码的警告，第三行控制一个标识符不等价重定义的警告，当编译程序开始处理时它恢复有效的警告状态(on 或 off)。注意由 pragma 语句指定的警告状态从遇到 pragma 之处开始，直到由另一个 pragma 语句改变为止。而且，这些指令同时取消默认的编译器状态以及由命令行开关—W 指定的选项(对于 tcc，或对 tc 是一个选项菜单项)。

(5)ANSI 预定义宏

ANSI 标准说明了以下的预定义宏：

__LINE__	当前源文件中的行号，表达为一个十进制整数常量。
__FILE__	当前源文件名，表达为一个串常量。

__DATA__　源文件开始预处理的日期(一个串常量)。

__TIME__　源文件开始预处理的时间(一个串常量)。

__STDC__　该常量的定义与 ANSI 标准一致(一个非零值)。Turbo C 中仅当设置了与 ANSI 兼容进行编译时定义(=1),即—A 选项,指出只有 ANSI 关键词被识别,且忽略象 far 这样的 Turbo 关键词。

2. 非 ANSI 预处理功能

Turbo C 中可以在预处理表达式中使用 sizeof 操作符。

另外,下面的预定义宏为 Turbo C 特有:

__TURBOC__ 当前的 Turbo C 版本,是带有主版本号(例如 2.0 中的 2)在高位字节,次版本号(例如 2.0 中的 0)在低位字节。

__PASCAL__ 仅当 Pascal 约定被激活(通过__p 标志)时有定义(=1)。

__MSDOS__ 对该版本的编译器总是为 1。

__CDECL__ 仅当 Pascal 约定不被激活时有定义(=1)。

在 Turbo C 中还有六个宏,对一个文件只有其中的一个有定义(=1),依赖于对当前的编译指定了哪种内存模式:__TINY__, __SMALL__, __MEDIUM__, __COMPACT__, __LARGE, __HUGE__。

1.1.3 Turbo C 编译程序的其它特性

Turbo C 编译程序的其它特性如下:

- 注释嵌套
- 双字符常量
- 标识符
- 一元运算+
- 串文字
- 关键词 asm
- 伪常量
- 库函数

1. 注释嵌套　虽然不是标准,或不是 ANSI 定义的一部分,在 Turbo C 默认情况下不允许,但编译选项__C(对于 tcc,对于 tc 是通过适当的选项菜单项)允许在源文件中进行注释嵌套,因此注释可以用来分块包含注释的代码段。

2. 双字符常数　Turbo C 允许字符常数包含两个字符,例如'AB'或'r\n'。这些值以 16 位整数保存,第一个字符为低字节,第二个字符为高字节(由于在内存中整数反序存储,这种顺序有意义,字符在内存中应与书写的顺序同样出现,就象一个包含两个字符的数组)。该特性不在 ANSI 标准中定义,且不可移植。

3. 标识符　C 标识符中除了允许出现标准的字母数字和下划线(__)外,Turbo C 还允许使用¥,并且该字符不能是标识符的第一个字符。该特性不是 ANSI 标准的一部分。

4. 一元运算+　Turbo C 实现了 ANSI 建议的一元运算符+,用来阻止编译器使用相关的加和乘,以重新安排一个表达式。例如,考虑下面所有变量为 float 类型的表达式:

```
Result=+(x+y)+z;
```

前面的+是一元运算符,使得 X 与 Y 先进行加,其结果加入到 Z。这对控制操作数的关

系，在某些情况下防止舍入或溢出错误是很重要的。该特性可避免必须使用几个对齐语句和中间临时变量。

5. 串文字　按照 ANSI 标准，Turbo C 提供了自动的串文字合并功能。在末尾加一个 Null 字符来构成一个串。例如：

```
static char message[]="it is no longer necessary to use back"
                      "slashes and mess up the program"
                      "listing to enter a long string constant.";
```

6. 关键词　asm Turbo C 允许通过 asm 关键词在文件中直接插入汇编语句。该特性在第 2 章中讨论。

7. 伪变量　通过使用伪变量，用户可以在 C 程序中直接存取 8086/8088 寄存器。这是系统级程序设计的一个有用工具，在第 2 章中有详细讨论。本书中许多程序设计实例都使用该特性。

8. 库函数　除了 ANSI 标准指出的库函数，Turbo C 运行库还包含了许多的函数和宏，并且还包括了大多数 C 编译器中包含的函数。许多这些函数用于直接存取低级硬件以及 MS—DOS 机器中的操作系统资源。可以通过快速浏览一下在《Turbo C 参考指南》中列出的运行库函数来获得该重要资源的总体认识。本书中给出的高级 Turbo C 库函数和扩充的函数集很大程度上是针对如何最优地使用 MS—DOS 环境。

1.2　Quick C 简介

你大概很希望能立刻在系统中装入 Quick C 编译。但应首先花几分钟检查一下你的系统是否能满足 Quick C 编译器的最低要求，并确认包装盒内的东西是否齐全。

1.2.1　打开 Quick C 的包装

Qukck　C 要求如下最低配置：

* 一台可运行 DOS2.1 或更高版本的 IBM　PC 或其兼容机。
* 一个硬盘驱动器和一个软盘驱动器。
* 448KB 的可用内存（为了大，中型项目的使用，建议有 512KB 内存）。

注：Microsoft 的资料中使用的“DOS”一词意指 Microsoft 和 IBM 磁盘操作系统（MS　DOS 和 PC　DOS）。

检查你的 Quick C 包装盒看看一切是否齐全。在包装盒内，你应看到如下几样东西：

* 注册卡：做为一个注册的 Quick C 2.5 软件的拥有人，你将得到许多好处。其中包括得到软件修改（改版）通知及容易得到主顾的帮助等。请您现在就花几分钟把注册卡填好，并将其寄出。
* 磁盘：8 片 5.25 英寸（1 英寸＝25.4mm，下同）软盘或 4 片 3.5 英寸软盘，带“Setup”标签的原盘中包含一个名为 PACKING.LST 的文件，该文件中列出了 Microsoft　Quick C 中所有文件的位置和描述。
* 《安装和运行》：它告诉你如何安装和运行 Quick C。
* 《C 语言参考手册》：这本书是为至少知道一种语言（如 basic 或 Pascal），但不了解 C 语言的程序员编写的，该书第一部分“学习 C”是 C 语言编程指南。其中包含许多例子，第二部分“C 的使用”进一步描述了输入输出库函数、图形产生函数和一些新功能，如

实坐标图形、表示图形、字型库及直插汇编等。该书的附录对 C 语言和 Quick C 库函数做了概述。

* 《Microsoft Quick C 成套工具》：该书介绍了配合 Quick C 使用的专用工具和实用程序，初学者在学习 C 语言的基础时不必参考该书。高级 C 语言程序员在需要有关编译、连接、建库多模块程序维护等的详细信息时，应该读这本书。

1.2.2 Quick C 的安装

本节告诉你如何在系统中安装 Quick C 编译器。Setup 原盘中的 SETUP. EXE 程序将完成该安装工作。SETUP. EXE 要做两件事。第一，它将原盘上的一部分程序（编译程序、连接程序、库管理程序、帮助系统及其它有关程序）拷贝到硬盘上。第二，它将产生一个或多个组合库，没有库你将无法用 C 语言编写程序。

本节还阐述了为什么需要建立组合库及组合库的各个单元如何组合在一起。如果你按照下面的说明去做，在读完本节时，你将在系统上拥有一个 Quick C 并可以开始用 C 语言编写程序。

1.2.2.1 Quick C 概要

安装 Quick C 需如下四个步骤：

1. 对所有的原盘做一个备份。

2. 阅读 README. DOC 文件中与安装 Quick C 有关的第一节。本书的任何修改都将列在 README. DOC 的开始。

3. 运行 SETUP. EXE 文件，这是一个交互式程序，你所回答的问题将决定被安装的 Quick C 的环境，缺省回答都列在括号内，每一屏的底部是对每个问题的简要说明。

4. 调整系统及环境变量。SETUP. EXE 将产生两个文件：NEW __ VARS. BAT 和 NEW __ CONF. SYS。将 NEW __ VARS. BAT 中的内容加入 AUTOEXE. BAT 中。如果你不愿将这种改变永久化，你可将 NEW __ VARS. BAT 作为批处理文件运行。如果变量 files 和 Duffers 的当前值小于其在 NEW __ CONF. SYS 中相应的值，则应修改 CONFIG. SYS 文件。文件修改之后应重新启动。

整个安装过程并不困难并且每屏都提供帮助指导信息，如果你具有足够的 DOS 和编程经验，不需要进一步帮助就可完成以上四个步骤，我们鼓励你这样做（如果你遇到困难，请回过头来再阅读本章）。你可以跳过本节而阅读“Quick C 的使用”。

注：如果你在系统安装过程中出现差错，只需重新运行 SETUP. EXE，SETUP. EXE 绝不会删除原盘中的文件。

1.2.2.2 运行 SETUP

在运行 SETUP 之前，用 DOS 的 COPY 命令或 DISKCOPY 程序对原盘做个备份，然后将 README. DOC 装入字处理器或使用 TYPE 命令阅读 README. DOC 的第一部分：

```
TYPE  README. DOC/MORE
```

当你安装 Quick C 时，先将 Setup 盘插入驱动器 A 并转换到该驱动器（键入 A:）。在 DOS 命令行键入：

```
SETUP
```

注：下面的指导说明假设你安装 Quick C 的系统至少具有一个硬盘驱动器和一个软盘驱动器。