C 语言基础教程(一)

Turbo C 语言概述

1.1 C 语言的产生与发展

C 语言是 1972 年由美国的 Dennis Ritchie 设计发明的，并首次在 UNIX 操作系统的 DEC PDP-11 计算机上使用。 它由早期的编程语言 BCPL(Basic Combind Programming Language) 发展演变而来。在 1970 年， AT&T 贝尔实验室的 Ken Thompson 根据 BCPL 语言设计出较先进的并取名为 B 的语言，最后导了 C 语言的问世。随着微型计算机的日益普及，出现了许多 C 语言版本。由于没有统一的标准，使得这些 C 语言之间出现了一些不一致的地方。为了改变这种情况，美国国家标准研究所(ANSI)为 C 语言制定了一套 ANSI 标准，成为现行的 C 语言标准

C 语言的特点

C 语言发展如此迅速，而且成为最受欢迎的语言之一，主要因为它具有强大的功能。许多著名的系统软件，如 DBASE Ⅲ PLUS、DBASE Ⅳ 都是由 C 语言编写的。用 C 语言加上一些汇编语言子程序，就更能显示 C 语言的优势了，象 PC- DOS 、WORDSTAR 等就是用这种方法编写的。归纳起来 C 语言具有下列特点:

1. C 是中级语言

它把高级语言的基本结构和语句与低级语言的实用性结合起来。C 语言可以象汇编语言一样对位、字节和地址进行操作，而这三者是计算机最基本的工作单元。

2. C 是结构式语言

结构式语言的显著特点是代码及数据的分隔化，即程序的各个部分除了必要的信息交流外彼此独立。这种结构化方式可使程序层次清晰，便于使用、维护以及调试。C 语言是以函数形式提供给用户的，这些函数可方便的调用，并具有多种循环、条件语句控制程序流向，从而使程序完全结构化。

3. C 语言功能齐全

C 语言具有各种各样的数据类型，并引入了指针概念，可使程序效率更高。另外 C 语言也具有强大的图形功能，支持多种显示器和驱动器。而且计算功能、逻辑判断功能也比较强大，可以实现决策目的。

4. C 语言适用范围大

C 语言还有一个突出的优点就是适合于多种操作系统，如 DOS、UNIX,也适用于多种机型。

Turbo C 2.0 的安装和启动

Turbo C 2.0 的安装非常简单，只要将 1#盘插入 A 驱动器中，在 DOS 的"A>" 下键入:

A>INSTALL

即可，此时屏幕上显示三种选择:

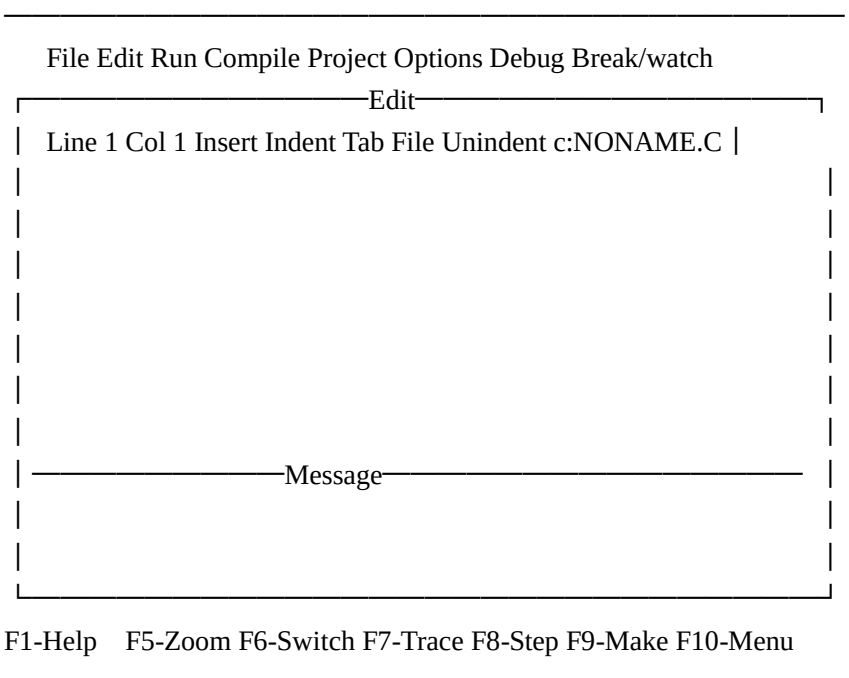
1. 在硬盘上创造一个新目录来安装整个 Turbo C 2.0 系统。

2. 对 Turbo C 1.5 更新版本。
这样的安装将保留原来对选择项、颜色和编辑功能键的设置。
3. 为只有两个软盘而无硬盘的系统安装 Turbo C 2.0。
这里假定按第一种选择进行安装，只要在安装过程中按对盘号的提示，顺序插入各个软盘，就可以顺利地进行安装，安装完毕将在 C 盘根目录下建立一个 TC 子目录，TC 下还建立了两个了目录 LIB 和 INCLUDE，LIB 子目录中存放库文件，INCLUDE 子目录中存放所有头文件。
运行 Turbo C 2.0 时，只要在 TC 子目录下键入 TC 并回车即可进入 Turbo C 2. 0 集成开发环境。

C 语言基础教程(二)

Turbo C 2.0 集成开发环境的使用

进入 Turbo C 2.0 集成开发环境中后，屏幕上显示:



其中顶上一行为 Turbo C 2.0 主菜单，中间窗口为编辑区，接下来是信息窗口，最底下一行为参考行。这四个窗口构成了 Turbo C 2.0 的主屏幕，以后的编程、编译、调试以及运行都将在这个主屏幕中进行。下面详细介绍主菜单的内容。

1.5.1 主菜单

主菜单 在 Turbo C 2.0 主屏幕顶上一行，显示下列内容:

File Edit Run Compile Project Options Debug Break/watch

除 Edit 外，其它各项均有子菜单，只要用 Alt 加上某项中第一个字母(即大写字母)，就可进入该项的子菜单中。

一、File(文件)菜单

按 Alt+F 可进入 File 菜单，该菜单包括以下内容:

.Load(加载)

装入一个文件，可用类似 DOS 的通配符(如*.C)来进行列表选择。也可装入其它扩展名的文件，只要给出文件名(或只给路径)即可。该项的热键为 F3，即只要在主菜单中按 F3 即可进入该项，而不需要先进入 File 菜单再选此项。

.Pick(选择)

将最近装入编辑窗口的 8 个文件列成一个表让用户选择， 选择后将该程序装入编辑区，并将光标置在上次修改过的地方。其热健为 Alt-F3。

.New(新文件)

说明文件是新的，缺省文件名为 NONAME.C，存盘时可改名。

.Save(存盘)

将编辑区中的文件存盘，若文件名是 NONAME.C 时，将询问是否更改文件名，其热键为 F2。

.Write to(存盘)

可由用户给出文件名将编辑区中的文件存盘，若该文件已存在，则询问要不要覆盖。

.Directory(目录)

显示目录及目录中的文件，并可由用户选择。

.Change dir(改变目录)

显示当前目录，用户可以改变显示的目录。

.Os shell(暂时退出)

暂时退出 Turbo C 2.0 到 DOS 提示符下， 此时可以运行 DOS 命令， 若想回到 Turbo C 2.0 中，只要在 DOS 状态下键入 EXIT 即可。

.Quit(退出)

退出 Turbo C 2.0，返回到 DOS 操作系统中，其热键为 Alt+X。

说明:

以上各项可用光标键移动色棒进行选择，回车则执行。也可用每一项的第一个大写字母直接选择。若要退到主菜单或从它的下一级菜单列表框退回均可用 Esc 键，Turbo C 2.0 所有菜单均采用这种方法进行操作，以下不再说明。

C 语言基础教程(三)

二、Edit(编辑)菜单

按 Alt+E 可进入编辑菜单，若再回车，则光标出现在编辑窗口， 此时用户可以进行文本编辑。

编辑方法基本与 wordstar 相同，可用 F1 键获得有关编辑方法的帮助信息。

与编辑有关的功能键如下:

F1	获得 Turbo C 2.0 编辑命令的帮助信息
F5	扩大编辑窗口到整个屏幕
F6	在编辑窗口与信息窗口之间进行切换
F10	从编辑窗口转到主菜单

编辑命令简介:

PageUp	向前翻页
PageDn	向后翻页
Home	将光标移到所在行的开始
End	将光标移到所在行的结尾
Ctrl+Y	删除光标所在的一行
Ctrl+T	删除光标所在处的一个词
Ctrl+KB	设置块开始
Ctrl+KK	设置块结尾
Ctrl+KV	块移动
Ctrl+KC	块拷贝
Ctrl+KY	块删除
Ctrl+KR	读文件
Ctrl+KW	存文件
Ctrl+KP	块文件打印
Ctrl+F1	如果光标所在处为 Turbo C 2.0 库函数, 则获得有关该函数的帮助信息
Ctrl+Q[	查找 Turbo C 2.0 双界符的后匹配符
Ctrl+Q]	查找 Turbo C 2.0 双界符的前匹配符

说明:

1. Turbo C 2.0 的双界符包括以下几种符号:

花括符	{和}
尖括符	<和>
圆括符	(和)
方括符	[和]
注释符	/*和*/
双引号	"
单引号	'

2. Turbo C 2.0 在编辑文件时还有一种功能, 就是能够自动缩进, 即光标定位和上一个非空字符对齐。在编辑窗口中, Ctrl+OL 为自动缩进开关的控制键。

C 语言基础教程(四)

三、Run(运行)菜单

按 Alt+R 可进入 Run 菜单, 该菜单有以下各项:

.Run(运行程序)

运行由 Project/Project name 项指定的文件名或当前编辑区的文件。如果对上次编译后的源代码未做过修改, 则直接运行到下一个断点(没有断点则运行到结束)。否则先进行编译、连接后才运行, 其热键为 Ctrl+F9。

.Program reset(程序重启)

中止当前的调试，释放分给程序的空间，其热键为 Ctrl+F2。

.Go to cursor(运行到光标处)

调试程序时使用，选择该项可使程序运行到光标所在行。光标所在行必须为一条可执行语句，否则提示错误。其热键为 F4。

.Trace into(跟踪进入)

在执行一条调用其它用户定义的子函数时，若用 Trace into 项，则执行长条将跟踪到该子函数内部去执行，其热键为 F7。

.Step over(单步执行)

执行当前函数的下一条语句，即使用户函数调用，执行长条也不会跟踪进函数内部，其热键为 F8。

.User screen(用户屏幕)

显示程序运行时在屏幕上显示的结果。其热键为 Alt+F5。

C 语言基础教程(五)

四、Compile(编译)菜单

按 Alt+C 可进入 Compile 菜单，该菜单有以下几个内容:

.Compile to OBJ(编译生成目标码)

将一个 C 源文件编译生成.OBJ 目标文件， 同时显示生成的文件名。其热键为 Alt+F9。

.Make EXE file(生成执行文件)

此命令生成一个.EXE 的文件，并显示生成的.EXE 文件名。其中.EXE 文件名是下面几项之一。

1. 由 Project/Project name 说明的项目文件名。
2. 若没有项目文件名，则由 Primary C file 说明的源文件。
3. 若以上两项都没有文件名，则为当前窗口的文件名。

.Link EXE file(连接生成执行文件)

把当前.OBJ 文件及库文件连接在一起生成.EXE 文件。

.Build all(建立所有文件)

重新编译项目里的所有文件，并进行装配生成.EXE 文件。该命令不作过时检查(上面的几条命令要作过时检查， 即如果目前项目里源文件的日期和时间与目标文件相同或更早，则拒绝对源文件进行编译)。

.Primary C file(主 C 文件)

当在该项中指定了主文件后，在以后的编译中，如没有项目文件名则编译此项中规定的主 C 文件，如果编译中有错误，则将此文件调入编辑窗口， 不管目前窗口中是不是主 C 文件。

.Get info(获得有关当前路径、源文件名、源文件字节大小、编译中的错误数目、可用空间等信息。

C 语言基础教程(六)

五、Project(项目)菜单

按 Alt+P 可进入 Project 菜单，该菜单包括以下内容:

.Project name(项目名)

项目名具有.PRJ 的扩展名，其中包括将要编译、连接的文件名。例如有一个程序由 file1.c, file2.c, file3.c 组成，要将这 3 个文件编译装配成一个 file.exe 的执行文件，可以先建立一个 file.prj 的项目文件，其内容如下:

file1.c
file2.c
file3.c

此时将 file.prj 放入 Project name 项中，以后进行编译时将自动对项目文件中规定的三个源文件分别进行编译。然后连接成 file.exe 文件。

如果其中有些文件已经编译成.OBJ 文件，而又没有修改过，可直接写上.OBJ 扩展名。此时将不再编译而只进行连接。

例如: file1.obj
file2.c
file3.c

将不对 file1.c 进行编译，而直接连接。

说明:

当项目文件中的每个文件无扩展名时，均按源文件对待，另外，其中的文件也可以是库文件，但必须写上扩展名.LIB。

.Break make on(中止编译)

由用户选择是否在有 Warining(警告)、Errors(错误)、Fatal Errors(致命错误)时或 Link(连接)之前退出 Make 编译。

.Auto dependencies(自动依赖)

当开关置为 on，编译时将检查源文件与对应的.OBJ 文件日期和时间，否则不进行检查。

.Clear project(清除项目文件)

清除 Project/Project name 中的项目文件名。

.Remove messages(删除信息)

把错误信息从信息窗口中清除掉。

C 语言基础教程(七)

六、Options(选择菜单)

按 Alt+O 可进入 Options 菜单，该菜单对初学者来说要谨慎使用。

.Compiler(编译器)

本项选择又有许多子菜单，可以让用户选择硬件配置、存储模型、调试技术、代码优化、对话信息控制和宏定义。这些子菜单如下:

Model

共有 Tiny, small, medium, compact, large, huge 六种不同模式可由同户选择。

Define

打开一个宏定义框，同户可输入宏定义。多重定义可同分号，赋值可用等号。

Code generation

它又有许多任选项，这些任选项告诉编译器产生什么样的目标代码。

Calling convention 可选择 C 或 Pascal 方式传递参数。

Instruction set 可选择 8088/8086 或 80186/80286 指令系列。

Floating point 可选择仿真浮点、数学协处理器浮点或无浮点运算。

Default char type 规定 char 的类型。

Alignonent 规定地址对准原则。

Merge duplicate strings 作优化用，将重复的字符串合并在一起。

Standard stack frame 产生一个标准的栈结构。

Test stack overflow 产生一段程序运行时检测堆栈溢出的代码。

Line number 在.OBJ 文件中放进行号以供调试时用。

OBJ debug information 在.OBJ 文件中产生调试信息。

Optimization

- Optimize for 选择是对程序小型化还是对程序速度进行优化处理。
- Use register variable 用来选择是否允许使用寄存器变量。
- Register optimization 尽可能使用寄存器变量以减少过多的取数操作。
- Jump optimization 通过去除多余的跳转和调整循环与开关语句的办法，压缩代码。

Source

- Indentifier length 说明标识符有效字符的个数，默认为 32 个。
- Nested comments 是否允许嵌套注释。
- ANSI keywords only 是只允许 ANSI 关键字还是也允许 Turbo C 2.0 关键字

Error

- Error stop after 多少个错误时停止编译，默认为 25 个。
- Warning stop after 多少个警告错误时停止编译，默认为 100 个。

Display warning

- Portability warning 移植性警告错误。
- ANSI Violations 侵犯了 ANSI 关键字的警告错误。
- Common error 常见的警告错误。
- Less common error 少见的警告错误。

Names 用于改变段(segment)、 组(group) 和类(class)的名字，默认值为 CODE,DATA,BSS。

.Linker(连接器)

本菜单设置有关连接的选择项，它有以下内容:

Map file menu 选择是否产生.MAP 文件。

Initialize segments 是否在连接时初始化没有初始化的段。

Devault libraries 是否在连接其它编译程序产生的目标文件时去寻找其缺省库。

Graphics library 是否连接 graphics 库中的函数。

Warn duplicate symbols　当有重复符号时产生警告信息。
Stack warinig　是否让连接程序产生 No stack 的警告信息。
Case-sensitive link　是否区分大、小写字。

.Environment(环境)
本菜单规定是否对某些文件自动存盘及制表键和屏幕大小的设置
Message tracking
Current file　跟踪在编辑窗口中的文件错误。
All files　跟踪所有文件错误。
Off　不跟踪。
Keep message　编译前是否清除 Message 窗口中的信息。
Config auto save　选 on 时，在 Run, Shell 或退出集成开发环境之前，如果 Turbo C 2.0 的配置被改过，则所做的改动将存入配置文件中。选 off 时不存。
Edit auto save　是否在 Run 或 Shell 之前，自动存储编辑的源文件。
Backup file　是否在源文件存盘时产生后备文件(.BAK 文件)。
Tab size　设置制表键大小，默认为 8。
Zoomed windows　将现行活动窗口放大到整个屏幕，其热键为 F5。
Screen size　设置屏幕文本大小。

.Directories(路径)
规定编译、连接所需文件的路径，有下列各项:
Include directories　包含文件的路径，多个子目录用";"分开。
Library directories　库文件路径，多个子目录用";"分开。
Output directoried　输出文件(.OBJ, .EXE, .MAP 文件)的目录。
Turbo C directoried　Turbo C 所在的目录。
Pick file name　定义加载的 pick 文件名，如不定义则从 current pick file 中取。

.Arguments(命令行参数)
允许用户使用命令行参数。

.Save options(存储配置)
保存所有选择的编译、连接、调试和项目到配置文件中，缺省的配置文件为 TCCONFIG.TC。

.Retrive options
装入一个配置文件到 TC 中, TC 将使用该文件的选择项。

七、Debug(调试)菜单

按 Alt+D 可选择 Debug 菜单，该菜单主要用于查错，它包括以下内容:

Evaluate	
Expression	要计算结果的表达式。
Result	显示表达式的计算结果。
New value	赋给新值。
Call stack	该项不可接触。而在 Turbo C debuger 时用于检查堆栈情况。
Find function	在运行 Turbo C debugger 时用于显示规定的函数。
Refresh display	如果编辑窗口偶然被用户窗口重写了可用此恢复编辑窗口的内容。

八、Break/watch(断点及监视表达式)

按 Alt+B 可进入 Break/watch 菜单，该菜单有以下内容:

Add watch	向监视窗口插入一监视表达式。
Delete watch	从监视窗口中删除当前的监视表达式。
Edit watch	在监视窗口中编辑一个监视表达式。
Remove all watches	从监视窗口中删除所有的监视表达式。
Toggle breakpoint	对光标所在的行设置或清除断点。
Clear all breakpoints	清除所有断点。
View next breakpoint	将光标移动到下一个断点处。

1.5.2 Turbo C 2.0 的配置文件

所谓配置文件是包含 Turbo C 2.0 有关信息的文件，其中存有编译、连接的选择和路径等信息。

可以用下述方法建立 Turbo C 2.0 的配置:

1. 建立用户自命名的配置文件

可以从 Options 菜单中选择 Options/Save options 命令， 将当前集成开发环境的所有配置存入一个由用户命名的配置文件中。下次启动 TC 时只要在 DOS 下键入:

tc/c<用户命名的配置文件名>

就会按这个配置文件中的内容作为 Turbo C 2.0 的选择。

2. 若设置 Options/Environment/Config auto save 为 on，则退出集成开发环境时，当前的设置会自动存放到 Turbo C 2.0 配置文件 TCCONFIG.TC 中。Turbo C 在启动时会自动寻找这个配置文件。

3. 用 TCINST 设置 Turbo C 的有关配置，并将结果存入 TC.EXE 中。Turbo C 在启动时，若没有找到配置文件，则取 TC.EXE 中的缺省值。

数据类型、变量和运算符

本节首先介绍 Turbo C 程序的基本组成部分；然后介绍 Turbo C 的数据类型、变量类型、变量的初始化和赋值；最后介绍 Turbo C 的有关操作。 通过本节的学习，可以对 Turbo C 语言有一个初步认识。

1. Turbo C 程序的一般组成部分

Turbo C 2.0 象其它语言一样按其规定的格式和提供的语句由用户编写应用程序。请看下面一段 Turbo C 源程序。

例 1:

```
/*Example program of Turbo C*/
#include <stdio.h>                          /*包含文件说明*/
void lgc(void);                             /*子函数说明*/
char answer;                                 /*定义全程变量*/
int main()                                   /*主函数定义*/
{
        char a;                              /*定义局部变量*/
        clrscr();
        gotoxy(12,3);
        puts("Welcome to use Turbo C2.0!");
        gotoxy(15, 13);
        printf("<Esc>--Exit");
        gotoxy(15, 15);
        printf("<CR>--Continue");
        while(1)
        {
          a=getch();
          if(a==27)
              break;
          if(a==13)
          {
              lgc();
              if(answer=='y'||answer=='Y')
              {
                gotoxy(23,14);
                puts("Please Write to the Company");
                getch();
                break;
              }
          }
        }
```

```
        return(0);
}
void lgc(void)
{
    clrscr();
    gotoxy(12,8);
    printf("The Excellent Selection!");
    gotoxy(21,12);
    printf("Do you have any question?(Y/N)");
    answer=getche();
}
```

由例子程序可以看出, Turbo C 源程序主要有以下几个特点:

1. 程序一般用小写字母书写;

2. 大多数语句结尾必须要用";"作为终止符, 否则 Turbo C 不认为该语句结束;

3. 每个程序必须有一个而且只能有一个称作主函数的 main()函数;

4. 每个程序体 (主函数和每个子函数, 如上例中的 main()函数和 sub()函数)必须用一对花括号"{"和"}"括起来;

5. 一个较完整的程序大致包括:包含文件(一组#include<*.h>语句)、用户函数说明部分、全程变量定义、主函数和若干子函数组成。在主函数和子函数中又包括局部变量定义、若干个 Turbo C 库函数、控制流程语句、 用户函数的调用语句等;

6. 注释部分包含在"/*"和"*/"之间, 在编译时它被 Turbo C 编译器忽略。

说明:

1. 象其它一些语言一样, Turbo C 的变量在使用之前必须先定义其数据类型, 未经定义的变量不能使用。定义变量类型应在可执行语句前面, 如上例 main()函数中的第一条语句就是变量定义语句, 它必须放在第一各执行语句 clrscr()前面。

2. 在 Turbo C 中, 大、小写字母是有区别的, 相同字母的大、小写代表不同的变量。

3. Turbo C 程序的书写格式非常灵活, 没有严格限制。

例 1 的主函数可写成:

```
main(){char c; clrscr(); gotoxy(12,3);
        puts("Welcome to use Turbo C2.0!"); gotoxy(15,13);
        printf("<CR>--Continue"); gotoxy(15,15);...}
```

这样写语法上没有错误, 但阅读起来不方便, 同时也使得程序层次不明确。作者建议用 Turbo C 编程时, 一行一条语句, 遇到嵌套语句向后缩进, 必要时对程序加上注释行。这样可以便程序结构清楚、易于阅读、维护和修改。

通过以上介绍, 可以得出 Turbo C 源程序的一般形式为:

```
包含文件
子函数类型说明
全程变量定义
main()
```

```
{
  局部变量定义
  <程序体>
}
sub1()
{
  局部变量定义
  <程序体>
}
sub2()
{
  局部变量定义
  <程序体>
}
    .
    .
    .
subN()
{
  局部变量定义
  <程序体>
}
```

其中 sub1(), ..., subN()代表用户定义的子函数，程序体指 Turbo C 2.0 提供的任何库函数调用语句、控制流程语句或其它用子函数调用语句等。

C 语言基础教程(十)

数据类型

在 Turbo C 语言中，每个变量在使用之前必须定义其数据类型。Turbo C 有以下几种类型：整型(int)、浮点型(float)、字符型(char)、指针型(*)、无值型(void)以及结构(struct)和联合(union)。其中前五种是 Turbo C 的基本数据类型、后两种数据类型(结构和联合)将在第五章介绍。

2.1 整型(int)

一、整型数说明

加上不同的修饰符，整型数有以下几种类型;

signed short int　有符号短整型数说明。简写为 short 或 int，字长为 2 字节共 16 位二进制数，数的范围是-32768~32767。

signed long int　有符号长整型数说明。简写为 long，字长为 4 字节共 32 位二进制数，数的范围是-2147483648~2147483647。

unsigned short int　无符号短整型数说明。简写为 unsigned int，　字长

为 2 字节共 16 位二进制数，数的范围是 0~65535。

unsigned long int　　无符号长整型数说明。简写为 unsigned long，字长为 4 字节共 32 位二进制数，数的范围是 0~4294967295。

二、整型变量定义

可以用下列语句定义整型变量

```
int a, b;                /*a、b 被定义为有符号短整型变量*/
unsigned long c;         /*c 被定义为无符号长整型变量*/
```

三、整型常数表示

按不同的进制区分，整型常数有三种表示方法:

十进制数:　以非 0 开始的数

如:220, -560, 45900

八进制数:　以 0 开始的数

如:06; 0106, 05788

十六进制数:以 0X 或 0x 开始的数

如:0X0D, 0XFF, 0x4e

另外，可在整型常数后添加一个"L"或"l"字母表示该数为长整型数，如 22L, 0773L, 0Xae4l。

2.2 浮点型(float)

一、浮点数说明

Turbo C 中有以下两种类型的浮点数:

float　　单浮点数。字长为 4 个字节共 32 位二进制数，　数的范围是 3.4x10-38E~3.4x10+38E。

double　　双浮点数。字长为 8 个字节共 64 位二进制数，　数的范围是 1.7x10-308E~1.7x10+308E。

说明:

浮点数均为有符号浮点数，没有无符号浮点数。

二、浮点型变量定义

可以用下列语句定义浮点型变量:

```
float a, f;        /*a, f 被定义为单浮点型变量*/
double b;          /*b 被定义为双浮点型变量*/
```

三、浮点常数表示

例如:　+29.56, -56.33, -6.8e-18, 6.365

说明:

1. 浮点常数只有一种进制(十进制)。

2. 所有浮点常数都被默认为 double。

3. 绝对值小于 1 的浮点数，其小数点前面的零可以省略。如:0.22 可写为.22, -0.0015E-3 可写为-.0015E-3。

4. Turbo C 默认格式输出浮点数时，最多只保留小数点后六位。

2.3 字符型(char)

加上不同的修饰符，可以定义有符号和无符号两种类型的字符型变量，例如:

```
char a:                  /*a 被定义为有符号字符变量*/
unsigned char l;         /*l 被定义为无符号字符变量*/
```

字符在计算机中以其 ASCII 码方式表示，其长度为 1 个字节，有符号字符型数取值范围为-128~127，无符号字符型数到值范围是 0~255。因此在 Turbo C 语言中，字符型数据在操作时将按整型数处理，如果某个变量定义成 char，则表明该变量是有符号的，即它将转换成有符号的整型数。

Turbo C 中规定对 ASCII 码值大于 0x80 的字符将被认为是负数。例如 ASCII 值为 0x8c 的字符，定义成 char 时，被转换成十六进制的整数 0xff8c 。这是因当 ASCII 码值大于 0x80 时，该字节的最高位为 1，计算机会认为该数为负数，对于 0x8c 表示的数实际上是-74(8c 的各位取反再加 1)，而-74 转换成两字节整型数并在计算机中表示时就是 0xff8c(对 0074 各位取反再加 1) 。因此只有定义为 unsigned char 0x8c 转换成整型数时才是 8c。这一点在处理大于 0x80 的 ASCII 码字符时(例如汉字码)要特别注意。一般汉字均定义为 unsigned char(在以后的程序中会经常碰到)。

另外，也可以定义一个字符型数组(关于数组后面再作详细介绍)，此时该数组表示一个字符串。

例如:

```
char str[10];
```

计算机在编译时，将留出连续 10 个字符的空间，即 str[0]到 str[9]共 10 个变量，但只有前 9 个供用户使用。第 10 个 str[9]用来存放字符串终止符 NULL 即"\0"，但终止符是编编译程序自动加上的，这一点应特别注意。

二、字符常数表示

能用符号表示的字符可直接用单引号括起来表示，如'a', '9', 'Z', 也可用该字符的 ASCII 码值表示，例如十进制数 85 表示大写字母'U'，十六进制数 0x5d 表示']'，八进制数 0102 表示大写字母'B'。

一些不能用符号表示的控制符，只能用 ASCII 码值来表示，如十进制数 10 表示换行，下六进制数 0x0d 表示回车，八进制数 033 表示 Esc。Turbo C2.0 中也有另外一种表示表示方法，如'\033'表示 Esc， 这里'\ 0' 符号后面的数字表示十六进制的 ASCII 值当然这种表示方法也适用于可睦接用符号表示的字符。

另外, Turbo C2.0 中有些常用的字符用以下特殊规定来表示:

规定符	等价于	含义
'\f'	'\X0C'	换页
'\r'	'\X0D'	回车
'\t'	'\X09'	制表键
'\n'	'\X0A'	换行
'\\'	'\X5C'	\符
'\''	'\X27'	'符
'\"'	'\X22'	"符

对于字符串常量，一般用双引号括起来表示，如"Hello Turbo C2.0"。

2.4 指针型(*)

指针是一种特殊的数据类型，在其它语言中一般没有。指针是指向变量的地址，实质上指针就是存贮单元的地址。 根据所指的变量类型不同， 可以是整型指针(int *)、浮点型指针(float *)、字符型指针(char *)、结构指针(struct *)和联合指针(union *)(结构指针和联合指针将在第 4 节中介绍)。

2.5 无值型(void)

无值型字节长度为 0，主要有两个用途： 一是明确地表示一个函数不返回任何值；一是产生一个同一类型指针(可根据需要动态分配给其内存)。

例如:

```
void *buffer;       /*buffer 被定义为无值型指针*/
```

C 语言基础教程(十一)

变量

变量说明

Turbo C2.0 规定所有变量在使用前都必须中以说明。一条变量说明语句由数据类型和其后的一个或多个变量名组成。变量说明的形式如下:

类型 <变量表>;

这里类型是指 Turbo C2.0 的有效数据类型。变量表是一个或多个标识符名，每个标识符之间用","分隔。

例如:

```
int i, j, k;      unsigned char c, str[5], *p;
```

4.2 变量种类

变量可以在程序中三个地方说明: 函数内部、函数的参数定义中或所有的函数外部。根据所定义位置的不同，变量可分为局部变量、形式参数和全程变量。

一、局部变量

局部变量是指在函数内部说明的变量(有时也称为自动变量)。用关键字 auto 进行说明，当 auto 省略时，所有的非全程变量都被认为是局部变量，所以 auto 实际上从来不用。

局部变量在函数调用时自动产生，但不会自动初始化，随函数调用的结束，这个变量也就自动消失了，下次调用此函数时再自动产生，还要再赋值，退出时又自动消失。

二、形式参数

形式参数是指在函数名后面的小括号里定义的变量，用于接受来自调用函数的参数。形式参数在函数内部可以象其它局部变量那样来作用。

例如:

```
puthz(int x, int y, int color, char *p)
{
        int i, j, k;     /*定义局部变量*/
        <程序体>
}
```

其中 x, y, color, *p 为函数的形式参数，　不需要再进行说明就可在该函数内直使用。

三、全程变量

全程变量是指在所有函数之外说明的变量，它在整个程序内部者是"可见的"，可以被任何一个函数使用，并且在整个程序的运行中都保留其值。全程变量只要满足在使用它以前和函数以外这两个条件，可在程序的任何位置进行说明，习惯上通常在程序的主函数前说明。

例如:

```
#include<stdio.h>
int test;                       /*定义全程变量*/
void f1(int x, float y); /*子函数说明*/
void f2(void);                  /*子函数说明*/
main()
{
        test=5;                   /*给全程变量赋值*/
        f1(20, 5.5);              /*调用有形式参数的子函数 f1()*/
                                    /*test 的值变成 115*/
        f2();                     /*调用 f2(), test 的值变为 1150*/
}
void f1(int x, float y)
{
        float z;                  /*z 定义为局部变量*/
        z=x*y;                      /*计算*/
        test=test+z;
}
void f2(void)
{
        int count=10;             /*定义局部变量并初始化*/
        test=test*count;
}
```

由于全程变量可被整个程序内的任何一个函数使用，所以可作为函数之间传递

参数的手段，但全程变量太多时，内存开销变大。

C 语言基础教程(十二)

变量

4.3 变量存储类型

Turbo C2.0 支持四种变量存储类型。说明符如下:
auto static extern register

下面分别来介绍。

一、auto

auto 称为自动变量，已在前面作了介绍，这里不再重复。

二、static

static 称为静态变量。根据变量的类型可以分为静态局部变量和静态全程变量。

1. 静态局部变量

它与局部变量的区别在于：在函数退出时，这个变量始终存在，但不能被其它函数使用，当再次进入该函数时，将保存上次的结果。其它与局部变量一样。

2. 静态全程变量

Turbo C2.0 允许将大型程序分成若干独立模块文件分别编译，然后将所有模块的目标文件连接在一起，从而提高编译速度，同时也便于软件的管理和维护。静态全程变量就是指只在定义它的源文件中可见而在其它源文件中不可见的变量。它与全程变量的区别是：全程变量可以再说明为外部变量(extern)，被其它源文件使用,而静态全程变量却不能再被说明为外部的，即只能被所在的源文件使用。

三、extern

extern 称为外部变量。为了使变量除了在定义它的源文件中可以使用外，还要被其它文件使用。因此，必须将全程变量通知每一个程序模块文件，此时可用 extern 来说明。

例如:

文件 1 为 file1.c 文件 2 为 file2.c
int i, j;/*定义全程变量*/ extern int i, j;/*说明将 i, j 从

```
   文件 1 中复制过来*/
char c;    extern char c; /*将 c 复制过来*/
void func1(int k);      func2()    /*用户定义函数*/
{
main()      static float k;/*定义静态变量*/
{    i=j*5/100;
func1(20);/*调用函数*/    k=i/1.5;
func2();      .
. .
. .
. }
   }
   func1(int k) /*用户定义函数*/
   {
j=k*100;
   }
```

对于以上两个文件 file1.c 和 file2.c，用 Turbo C2.0 的集成开发环境进行编译连接时，首先应建立一个.prj 的文件。例如 file.prj，该文件内容如下:

```
   file1.c
   file2.c
```

然后将 file.prj 的文件名写入主菜单 Project 中的 Project Name 项中。 再用 F9 编译连接，就可产生一个文件名为 fioe.exe 的可执行文件。

外部变量和 FORTRAN 语言中的 COMMOM 定义的公共变量一样。

四、register

register 称为寄存器变量。它只能用于整型和字符型变量。定义符 register 说明的变量被 Turbo C2.0 存储在 CPU 的寄存器中， 而不是象普通的变量那样存储在内存中，这样可以提高运算速度。但是 Turbo C2.0 只允许同时定义两个寄存器变量,一旦超过两个，编译程序会自动地将超过限制数目的寄存器变量当作非寄存器变量来处理。因此，寄存器变量常用在同一变量名频繁出现的地方。

另外，寄存器变量只适用于局部变量和函数的形式参数，它属于 auto 型变量,因此，不能用作全程变量。定义一个整型寄存器变量可写成:

```
    register int a;
```

对于以上所介绍的变量类型和变量存储类型将会在以后的学习中，通过例行程序中的定义、使用来逐渐加深理解。

C 语言基础教程(十三)

变量

4.4　数组变量

所谓数组就是指具有相同数据类型的变量集，并拥有共同的名字。数组中的每个特定元素都使用下标来访问。数组由一段连续的存贮地址构成，最低的地址对应于第一个数组元素，最高的地址对应最后一个数组元素。数组可以是一维的、也可以是多维的。Turbo C2.0 象它高级语方一样也使用数组变量。

一、一维数组

一维数组的说明格式是:

　　类型　变量名[长度];

类型是指数据类型，即每一个数组元素的数据类型，包括整数型、浮点型、字符型、指针型以及结构和联合。

例如:

```
int a[10];
unsigned long a[20];
char *s[5];
char *f[];
```

说明:

1. 数组都是以 0 作为第一个元素的下标，因此，当说明一个 int a[16] 的整型数组时，表明该数组有 16 个元素, a[0]~a[15]，一个元素为一个整型变量。

2. 大多数字符串用一维数组表示。数组元素的多少表示字符串长度，　数组名表示字符串中第一个字符的地址，　例如在语句 char　str[8] 说明的数组中存入"hello"字符串后, str 表示第一个字母"h"所在的内存单元地址。str[0] 存放的是字母"h"的 ASCII 码值，以此类推, str[4]存入的是字母"o"的 ASCII 码值, str[5]则应存放字符串终止符'\0'。

3. Turbo C2.0 对数组不作边界检查。例如用下面语句说明两个数组

```
char str1[5], str2[6];
```

当赋给 str1 一个字符串"ABCDEFG"时，只有"ABCDE"被赋给, "E" 将会自动的赋给 str2，这点应特别注意。

三、多维数组

多维数组的一般说明格式是:

类型　数组名[第 n 维长度][第 n-1 维长度].....[第 1 维长度];

这种说明方式与 BASIC、FORTRAN 等语言中多维数组的说明不一样。

例如:

```
int m[3][2]; /*定义一个整数型的二维数组*/
char c[2][2][3];　/*定义一个字符型的三维数组*/
```

数组 m[3][2]共有 3*2=6 个元素，顺序为:

```
m[0][0], m[0][1], m[1][0], m[1][1], m[2][0], m[2][1];
```

数组 c[2][2][3]共有 2*2*3=12 个元素，顺序为:

```
c[0][0][0], c[0][0][1], c[0][0][2],
c[0][1][0], c[0][1][1], c[0][1][2],
c[1][0][0], c[1][0][1], c[1][0][2],
```

c[1][1][0], c[1][1][1], c[1][1][2],
数组占用的内存空间(即字节数)的计算式为:
字节数=第 1 维长度*第 2 维长度*...*第 n 维长度*该数组数据类型占用的字节数

C 语言基础教程(十四)

变量

二、变量的赋值
变量赋值是给已说明的变量赋给一个特定值。
1. 单个变量的赋值
(1) 整型变量和浮点变量
这两种变量采用下列格式赋值
变量名=表达式;
例如:

```
main()
{
int a, m;     /*定义局部整型变量 a, m*/
float n; /*定义局部浮点变量 f*/
a=100, m=20;     /*给变量赋值*/
n=a*m*0.1;
.
.
.
}
```

说明:
Turbo C2.0 中允许给多个变量赋同一值时可用连等的方式。
例如:

```
main()
{
int a, b, c;
a=b=c=0; /*同时给 a,b,c 赋值*/
.
.
.
}
```

(2) 字符型变量
字符型变量可以用三种方法赋值。
例如:

```
main()
{
```

```
char a0, a1, a2; /*定义局部字符型变量 a0, a1, a2*/
a0='b'; /*将字母 b 赋给 a0*/
a1=50;   /*将数字 2(十进制 ASCII 值为 50 赋给 a1*/
a2='\x0d'; /*将回车符赋给 a2*/
.
.
.
   }
```

(3) 指针型变量

例如:

```
   main()
   {
int *i;
char *str;
*i=100;
str="Good";
.
.
.
   }
```

*i 表示 i 是一个指向整型数的指针，即*i 是一个整型变量, i 是一个指向该整型变量的地址。

*str 表示 str 是一个字符型指针，即保留某个字符地址。在初始化时, str 没有什么特殊的值，而在执行 str="Good"时，编译器先在目标文件的某处保留一个空间存放"Good\0"的字符串，然后把这个字符串的第一个字母"G"的地址赋给 str，其中字符串结尾符"\0"是编译程序自动加上的。

对于指针变量的使用要特别注意。上例中两个指针在说明前没有初始化，因此这两指针为随机地址，在小存储模式下使用将会有破坏机器的危险。正确的使用办法如下:

例如:

```
   main()
   {
int *i;
char *str;
i=(int*)malloc(sizeof(int));
i=420;
str=(char*)malloc(20);
str="Good, Answer!";
.
.
.
   }
```

上例中，函数(int*)malloc(sizeof(int))表示分配连续的 sizeof(int)=2 个字

节的整型数存储空间并返回其首地址。同样(char*)malloc(20)表示分配连续 20 个字节的字符存储空间并返回首地址(有关该函数以后再详述)。由动态内存分配函数 malloc()分配了内存空间后, 这部分内存将专供指针变量使用。
如果要使 i 指向三个整型数, 则用下述方法。
例如:

```
   #include<alloc.h>
   main()
   {
int *a;
a=(int*)malloc(3*sizeof(int));
*a=1234;
*(a+1)=4567;
*(a+2)=234;
.
.
.
   }
```

i=1234 表示把 1234 存放到 i 指向的地址中去, 但对于(i+1)=4567, 如果认为将 4567 存放到 i 指向的下一个字节中就错了。Turbo C2.0 中只要说明 i 为整型指针, 则 (i+1) 等价于 i+1*sizeof(int)
同样 (i+2) 等价于 i+2*sizeof(int)

C 语言基础教程(十五)

变量

2. 数组变量的赋值
(1) 整型数组和浮点数组的赋值
例如:

```
   main()
   {
int m[2][2];
float n[3];
m[0][0]=0, m[0][1]=17, m[1][0]=21;/*数组元素赋值*/
n[0]=109.5, n[1]=-8.29, n[2]=0.7;
.
.
.
   }
```

(2)字符串数组的赋值
例如:

```
    main()
```

```
   {
char s[30];
strcpy(s, "Good News!");   /*给数组赋字符串*/
.
.
.
   }
```

上面程序在编译时，遇到 char s[30]这条语句时，编译程序会在内存的某处留出连续 30 个字节的区域，并将第一个字节的地址赋给 s。当遇到 strcpy(strcpy 为 Turbo C2.0 的函数)时，首先在目标文件的某处建立一个"Good News!\0" 的字符串。其中\0 表示字符串终止，终止符是编译时自动加上的，然后一个字符一个字符地复制到 s 所指的内存区域。因此定义字符串数组时， 其元素个数至少应该比字符串的长度多 1。

注意:

1. 字符串数组不能用"="直接赋值，即 s="Good News!"是不合法的。所以应分清字符串数组和字符串指针的不同赋值方法。

2. 对于长字符串, Turbo C2.0 允许使用下述方法:

例如:

```
   main()
   {
char s[100];
strcpy(s, "The writer would like to thank you for"
   "your interest in his book. He hopes you"
   "can get some helps from the book.");
.
.
.
   }
```

(3)指针数组赋值

例如:

```
   main()
   {
char *f[2];
int *a[2];
f[0]="thank you";     /*给字符型数组指针变量赋值*/
f[1]="Good Morning";
*a[0]=1, *a[1]=-11; /*给整型数数组指针变量赋值*/
.
.
.
   }
```

变量

三、数组与指针

数组与指针有密切的联系。数组名本身就是该数组的指针，反过来，也可以把指针看成一个数组，数组名和指针实质上都是地址，但是指针是变量，可以作运算。而数组名是常量，不能进行运算。

例如:

```
   main()
   {
char s[30], *p; /*定义字符型数组和指针变量*/
p=s;     /*指针 p 指向数组 s 的第一个元素 s[0]的地址*/
.
.
.
*(p+8);     /*指针 p 指向数组 s 的第 9 个元素 s[8]的地址*/
.
.
.
   }
```

由上例可以看出数组和指针有如下关系:

(p+i)=&(s[i])

*(p+i)=s[i]

因此，利用上述表达式可以对数组和指针进行互换。两者的区别仅在于：数组 s 是程序自动为它分配了所需的存储空间；而指针 p 则是利用动态分想函数为它分配存储空间或赋给它一个已分配的空间地址。

C 语言基础教程(十七)

运算符(1)

Turbo C 的运算符非常丰富，主要分为三大类：算术运算符， 关系运算符与逻辑运算符，按位运算符。除此之外，还有一些用于完成特殊任务的运算符。下面分别进行介绍。

5.1 算术运算符

Turbo C 的算术运算符如下:

操作符	作用
+	加，一目取正

-	减，一目取负
*	乘
/	除
%	取模
--	减 1
++	加 1

一、一目和二目操作

一目操作是指对一个操作数进行操作。例如: -a 是对 a 进行一目负操作。

二目操作(或多目操作)是指两个操作数(或多个操作数)进行操作。

在 Turbo C 中加、减、乘、除、取模的运算与其它高级语言相同。需要注意的是除法和取模运算。

例如:

15/2　　是 15 除以 2 商的整数部分 7

15%2　　是 15 除以 2 的余数部分 1

对于取模运算符"%"，不能用于浮点数。

另外，由于 Turbo C 中字符型数会自动地转换成整型数，因此字符型数也可以参加二目运算。

例如:

```
main()
{
    char m, n;        /*定义字符型变量*/
    m='c';            /*给 m 赋小写字母'c'*/
    n=m+'A'-'a';      /*将 c 中的小写字母变成大写字母'B'后赋给 n*/
    ...
}
```

上例中 m='c'即 m=98，由于字母 A 和 a 的 ASCII 码值分别为 65 和 97。这样可以将小写字母变成大写字母，反之，如果要将大写字母变成小写字母，则用 c+ 'a' -'A'进行计算。

C 语言基础教程(十八)

运算符(2)

二、增量运算

在 Turbo C 中有两个很有用的运算符，在其它高级语言中通常没有。这两个运算符就是增 1 和减 1 运算符"++"和"--"，运算符"++"是操作数加 1，而"--" 则是操作数减 1。

例如:

x=x+1　　可写成 x++，或++x

x=x-1　　可写成 x--，或--x

x++(x--)与++x(--x)在上例中没有什么区别，但 x=m++和 x=++m 却有很大差别。

x=m++　　表示将 m 的值赋给 x 后, m 加 1。
x=++m　　表示 m 先加 1 后，再将新值赋给 x。

三、赋值语句中的数据类型转换

类型转换是指不同类型的变量混用时的类型改变。

在赋值语句中，类型转换规则是:

等号右边的值转换为等号左边变量所属的类型。

例如:

```
main()
{
        int i, j;          /*定义整型变量*/
        float f, g=2.58;       /*定义浮点型变量*/
        f=i*j;              /*i 与 j 的乘积是整型数，被转换成为浮点数赋给 f*/
        i=g;                /*g 中的浮点型数转换成为整型数赋给 i*/
        ...
}
```

由于 Turbo C 按上述数据类型转换规则，因此在作除法运算时应特别注意。

例如:

```
main()
{
        float f;
        int i=15;
        f=i/2;
}
```

上面程序经运行后, f=7 并不等于准确值 7.5。正确的程序应该是:

```
main()
{
        float f;
        int i=15;
        f=i/2.0;
}
```

也可直接将 i 定义为浮点数。

C 语言基础教程(十九)

运算符(3)

5.2 关系运算符和逻辑运算符

一、逻辑运算符

逻辑运算符是指用形式逻辑原则来建立数值间关系的符号。

Turbo C 的逻辑运算符如下:

操作符	作用

&&	逻辑与
||	逻辑或
!	逻辑非

二、关系运算符

关系运算符是比较两个操作数大小的符号。

Turbo C 的关系运算符如下:

操作符	作用
>	大于
>=	大于等于
<	小于
<=	小于等于
==	等于
!=	不等于

关系运算符和逻辑运算符的关键是真(true)和假(false)的概念。Turbo C 中 true 可以是不为 0 的任何值，而 false 则为 0。 使用关系运算符和逻辑运算符表达式时，若表达式为真(即 true)则返回 1，否则，表达式为假(即 false)， 则返回 0。

例如:

100>99 返回 1

10>(2+10) 返回 0

!1&&0 返加 0

对上例中表达式!1&&0，先求!1 和先求 1&&0 将会等于出不同的结果， 那么何者优先呢？这在 Turbo C 中是有规定的。 有关运算符的优先级本节后面将会讲到。

[上一页] [下一页]

C 语言基础教程(二十)

运算符(4)

5.3 按位运算符

Turbo C 和其它高级语言不同的是它完全支持按位运算符。这与汇编语言的位操作有些相似。

Turbo C 中按位运算符有:

操作符	作用
&	位逻辑与

|	位逻辑或
^	位逻辑异或
-	位逻辑反
>>	右移
<<	左移

按位运算是对字节或字中的实际位进行检测、设置或移位，它只适用于字符型和整数型变量以及它们的变体，对其它数据类型不适用。

关系运算和逻辑运算表达式的结果只能是 1 或 0。 而按位运算的结果可以取 0 或 1 以外的值。

要注意区别按位运算符和逻辑运算符的不同，例如，若 x=7，则 x&&8 的值为真(两个非零值相与仍为非零)，而 x&8 的值为 0。

移位运算符">>"和"<<"是指将变量中的每一位向右或向左移动，其通常形式为:

右移: 变量名>>移位的位数

左移: 变量名<<移位的位数

经过移位后，一端的位被"挤掉"， 而另一端空出的位以 0 填补， 所以，Turbo C 中的移位不是循环移动的。

[上一页] [下一页]

C 语言基础教程(二十一)

运算符(5)

5.4 Turbo C 的特殊运算符

一、"?"运算符

"?"运算符是一个三目运算符，其一般形式是:

<表达式 1>?<表达式 2>:<表达式 3>;

"?"运算符的含义是: 先求表达式 1 的值，如果为真，则求表达式 2 的值并把它作为整个表达式的值; 如果表达式 1 的值为假，则求表达式 3 的值并把它作为整个表达式的值。

例如:

```
main()
{
    int x, y;
    x=50;
    y=x>70?100:0;
}
```

本例中, y 将被赋值 0。如果 x=80, y 将被赋值 100。

因此, "?"运算符可以代替某些 if-then-else 形式的语句。

二、"&"和"*"运算符

"&"运算符是一个返回操作数地址的单目操作符。

"*"运算符是对"&"运算符的一个补充，它返回位于这个地址内的变量值，也是单目操作符。

例如:

```
main()
{
        int i, j, *m;
        i=10;
        m=&i;                /*将变量 i 的地址赋给 m*/
        j=*m;                /*地址 m 所指的单元的值赋给 j*/
}
```

上面程序运行后, i=10, m 为其对应的内存地址, j 的值也为 10。

三、","运算符

","运算符用于将多个表达式串在一起, ","运算符的左边总不返回，右边表达式的值才是整个表达式的值。

例如:

```
main()
{
        int x, y;
        x=50;
        y=(x=x-5, x/5);
}
```

上面程序执行后 y 值为 9，因为 x 的初始值为 50，减 5 后变为 45, 45 除 5 为 9 赋给 y。

C 语言基础教程(二十二)

运算符(6)

四、sizeof 运算符

sizeof 运算符是一个单目运算符，它返回变量或类型的字节长度。

例如:

```
sizeof(double)  为 8
sizeof(int)     为 2
```

也可以求已定义的变量，例如:

```
float f;
int i;
i=sizeof(f);
```

则 i 的值将为 4。

五、联合操作

Turbo C 中有一特殊的简写方式，它用来简化一种赋值语句， 适用于所有的

双目运算符。其一般形式为:

<变量>=<变量><操作数><表达式>

相当于

<变量><操作数>=<表达式>

例如:

a=a+b　　可写成　　a+=b

a=a&b　　可写成　　a&=b

a=a/(b-c)　　可写成　　a/=b-c

5.5 Turbo C 运算符的优先级

Turbo C 规定了运算符的优先次序即优先级。 当一个表达式中有多个运算符参加运算时, 将按下表所规定的优先级进行运算。表中优先级从上往下逐渐降低, 同一行优先级相同。

例如:

表达式 10>4&&!(100<99)||3<=5 的值为 1

表达式 10>4&&!(100<99)&&3<=5 的值为 0

Turbo C 运算符的优先次序

表达式	优先级
()(小括号) [](数组下标) .(结构成员) ->(指针型结构成员)	最高
!(逻辑非) .(位取反) -(负号) ++(加 1) --(减 1) &(变量地址)	↑
*(指针所指内容) type(函数说明) sizeof(长度计算)	|
*(乘) /(除) %(取模)	|
+(加) -(减)	|
<<(位左移) >>(位右移)	|
<(小于) <=(小于等于) >(大于) >=(大于等于)	|
==(等于) !=(不等于)	|
&(位与)	|
^(位异或)	|
|(位或)	|
&&(逻辑与)	|

||(逻辑或)	低
?:(?表达式)	于
= += -=(联合操作)	优
,(逗号运算符)	先
	级

[上一页] [下一页]

语言基础教程(二十四)

函数(2)

1. 函数的说明与定义

Turbo C2.0 中所有函数与变量一样在使用之前必须说明。所谓说明是指说明函数是什么类型的函数，一般库函数的说明都包含在相应的头文件<*.h>中， 例如标准输入输出函数包含在 stdio.h 中，非标准输入输出函数包含在 io.h 中， 以后在使用库函数时必须先知道该函数包含在什么样的头文件中，在程序的开头用#include <*.h>或#include"*.h"说明。只有这样程序在编译，连接时 Turbo C 才知道它是提供的库函数，否则，将认为是用户自己编写的函数而不能装配。

1.1 函数说明

1. 经典方式

其形式为: 函数类型 函数名();

2. ANSI 规定方式

其形式为: 函数类型 函数名(数据类型 形式参数， 数据类型 形式参数, ...);

其中: 函数类型是该函数返回值的数据类型，可以是以前介绍的整型(int)，长整型(long)，字符型(char)，单浮点型(float)，双浮点型(double)以及无值型(void)，也可以是指针，包括结构指针。无值型表示函数没有返回值。

函数名为 Turbo C2.0 的标识符，小括号中的内容为该函数的形式参数说明。可以只有数据类型而没有形式参数，也可以两者都有。对于经典的函数说明没有参数信息。如:

```
int putlll(int x,int y,int z,int color,char *p)/*说明一个整型函数*/
char *name(void);                    /*说明一个字符串指什函数*/
void student(int n, char *str); /*说明一个不返回值的函数*/
float calculate();                   /*说明一个浮点型函数*/
```

注意: 如果一个函数没有说明就被调用，编译程序并不认为出错，而将此函数默认为整型(int)函数。因此当一个函数返回其它类型，又没有事先说明， 编译时将会出错。

1.2 函数定义

函数定义就是确定该函数完成什么功能以及怎么运行，相当于其它语言的一个子程序。Turbo C2.0 对函数的定义采用 ANSI 规定的方式。即:

```
函数类型  函数名(数据类型形式参数; 数据类型 形式参数...)
{
    函数体;
}
```

其中函数类型和形式参数的数据类型为 Turbo C2.0 的基本数据类型。函数体为 Turbo C2.0 提供的库函数和语句以及其它用户自定义函数调用语句的组合，并包括在一对花括号"{"和"}"中。

需要指出的是一个程序必须有一个主函数，其它用户定义的子函数可以是任意多个，这些函数的位置也没有什么限制，可以在 main()函数前， 也可以在其后。Turbo C2.0 将所有函数都被认为是全局性的。而且是外部的，即可以被另一个文件中的任何一个函数调用。

C 语言基础教程(二十五)

函数(3)

2 函数的调用

2.1 函数的简单调用

Turbo C2.0 调用函数时直接使用函数名和实参的方法，也就是将要赋给被调用函数的参量，按该函数说明的参数形式传递过去，然后进入子函数运行，运行结束后再按子函数规定的数据类型返回一个值给调用函数。使用 Turbo C2.0 的库函数就是函数简单调用的方法。举例说明如下:

例 1:

```
#include<stdio.h>
int maxmum(int x, int y, int z);     /*说明一个用户自定义函数*/
int main()
{
        int i, j, k;
        printf("i, j, k=?\n");
        scanf("%4d%4d%4d", &i, &j, &k);
        maxmum(i, j, k);
        getch();
        return 0;
}

maxmum(int x, int y, int z)
{
```

```
    int max;
    max=x>y?x:y;
    max=max>z?max:z;
    printf("The maxmum value of the 3 data is %d\n", max);
}
```

C 语言基础教程(二十六)

函数(4)

2 函数的调用

2.2 函数参数传递

一、调用函数向被调用函数以形式参数传递

用户编写的函数一般在对其说明和定义时就规定了形式参数类型，因此调用这些函数时参量必须与子函数中形式参数的数据类型、顺序和数量完全相同，否则在调用中将会出错，得到意想不到的结果。

注意:

当数组作为形式参数向被调用函数传递时，只传递数组的地址，而不是将整个数组元素都复制到函数中去，即用数组名作为实参调用子函数，调用时指向该数组第一个元素的指针就被传递给子函数。因为在 Turbo C2.0 中，没有下标的数组名就是一个指向该数组第一个元素的指针。当然数组变量的类型在两个函数中必须相同。

用下述方法传递数组形参。

例 2:

```
#include<stdio.h>
void disp(int *n);
int main()
{
    int m[10], i;
    for(i=0; i<10; i++)
       m[i]=i;
    disp(m);          /*按指针方式传递数组*/
    getch();
    return 0;
}
void disp(int *n)
{
    int j;
    for(j=0; j<10; j++)
       printf("%3d", *(n++));
    printf("\n");
```

```
}
```

另外，当传递数组的某个元素时，数组元素作为实参，此时按使用其它简单变量的方法使用数组元素。例 2 按传递数组元素的方法传递时变为:

```
#include<stdio.h>
void disp(int n);
int main()
{
        int m[10], i;
        for(i=0; i<10; i++){
           m[i]=i;
           disp(m[i]);   /*逐个传递数组元素*/
        }
        getch();
        return 0;
}
void disp(int n)
{
        printf("%3d\t");
}
```

这时一次只传递了数组的一个元素。

语言基础教程(二十七)

函数(5)

2 函数的调用

2.2 函数参数传递

二、被调用函数向调用函数返回值

一般使用 return 语句由被调用函数向调用函数返回值，该语句有下列用途:

1. 它能立即从所在的函数中退出，返回到调用它的程序中去。
2. 返回一个值给调用它的函数。

有两种方法可以终止子函数运行并返回到调用它的函数中: 一是执行到函数的最后一条语句后返回; 一是执行到语句 return 时返回。前者当子函数执行完后仅返回给调用函数一个 0。若要返回一个值，就必须用 return 语句。只需在 return 语句中指定返回的值即可。例 1 返回最大值时变为:

例 3:

```
#include<stdio.h>
int maxmum(int x, int y, int z);     /*说明一个用户自定义函数*/
int main()
```

```
{
        int i, j, k, max;
        printf("i, j, k=?\n");
        scanf("%4d%4d%4d", &i, &j, &k);
        max=maxmum(i, j, k);        /*调用子函数，并将返回值赋给 max*/
        printf("The maxmum value is %d\n", max);
        getch();
        return 0;
}

maxmum(int x, int y, int z)
{
        int max;
        max=x>y?x:y;              /*求最大值*/
        max=max>z?max:z;
        return(max);              /*返回最大值*/
}
```

return 语句可以向调用函数返回值，但这种方法只能返回一个参数，在许多情况下要返回多个参数，这是用 return 语句就不能满足要求。Turob C2.0 提供了另一种参数传递的方法，就是调用函数向被调用函数传递的形式参数不是传递变量本身，而是传递变量的地址，当子函数中向相应的地址写入不同的数值之后，也就改变了调用函数中相应变量的值，从而达到了返回多个变量的目的。

C 语言基础教程(二十八)

函数(6)

2 函数的调用

2.2 函数参数传递

二、被调用函数向调用函数返回值

下面以两个例子来解释一下。

例 4:

```
#include<stdio.h>
void subfun(int *m, int *n);   /*说明子函数*/
int main()
{
        int i, j;
        printf("i, j=?\n");
        scanf("%d, %d", &i, &j); /*从键盘输入 2 个整数*/
```

```
        printf("In main before calling\n"/*输出此 2 数及其乘积*/
                "i=%-4d j=%-4d i*j=%-4d\n", i, j, i*j);
        subfun(&i, &j);                 /*以传送地址的方式调用子函数*/
        printf("In main after calling\n"/*调用子函数后输出变量值*/
                "i=%-4d j=%-4d i*j=%-4d\n", i, j, i*j);
        getch();
        return 0;
}
void subfun(int *m, int *n)
{
        *m=*m+2;
        *j=*i-*j;
        printf("In subfun after calling\n" /*子函数中输出变量值*/
                "i=%-4d j=%-4d i*j=%-4d\n", *i, *j, *i**j);
 }
```

上例中, *i**j 表示指针 i 和 j 所指的两个整型数*i 和*j 之乘积。

另外, return 语句也可以返回一个指针, 举例如下。

下例中先等待输入一字符串, 再等待输入要查找的字符, 然后调用 match() 函数在字符串中查找该字符。若有相同字符, 则返回一个指向该字符串中这一位置的指针, 如果没有找到, 则返回一个空(NULL)指针。

例 5:

```
#include<stdio.h>
char *match(char c, char *s);
int main()
{
        char s[40], c, *str;
        str=malloc(40);                 /*为字符串指什分配内存空间*/
        printf("Please input character string:");
        gets(s);                        /*键盘输入字符串*/
        printf("Please input one character:");
        c=getche();                     /*键盘输入字符*/
        str=match(c, s);                /*调用子函数*/
        putchar('\n');
        puts(str);                      /*输出子函数返回的指针所指的字符串*/
        getch();
        return 0;
}
char *match(char c, char *s)
{
        int i=0;
        while(c!=s[i]&&s[i]!='\n')/*找字符串中指定的字符*/
                i++;
```

```
        return(&s[i]);                  /*返回所找字符的地址*/
    }
```

C 语言基础教程(二十九)

函数(7)

2 函数的调用

2.2 函数参数传递

三、用全程变量实现参数互传

以上两种办法可以在调用函数和被调用函数间传递参数，但使用不太方便。如果将所要传递的参数定义为全程变量，可使变量在整个程序中对所有函数都可见。这样相当于在调用函数和被调用函数之间实现了参数的传递和返回。这也是实际中经常使用的方法，但定义全程变量势必长久地占用了内存。因此，全程变量的数目受到限制，特别对于较大的数组更是如此。当然对于绝大多数程序内存都是够用的。

例 6:

```
#incluide<stdio.h>
void disp(void);
int m[10];                  /*定义全程变量*/
int main()
{
        int i;
        printf("In main before calling\n");
        for(i=0; i<10; i++){
           m[i]=i;
           printf("%3d", m[i]);     /*输出调用子函数前数组的值*/
        }
        disp();                          /*调用子函数*/
        printf("\nIn main after calling\n");
        for(i=0; i<10; i++)
           printf("%3d", m[i]);     /*输出调用子函数后数组的值*/
        getch();
        return 0;
}
void disp(void)
{
        int j;
        printf("In subfunc after calling\n");/*子函数中输出数组的值*/
        for (j=0; i<10; j++){
              m[j]=m[j]*10;
```

```
            printf("%3d", m[i]);
        }
    }
```

C 语言基础教程(三十)

函数(8)

2 函数的调用

2.3 函数的递归调用

Turbo C2.0 允许函数自己调用自己，即函数的递归调用，递归调用可以使程序简洁、代码紧凑，但要牺牲内存空间作处理时的堆栈。

如要求一个 n!(n 的阶乘)的值可用下面递归调用:

例 8:

```
#include<stdio.h>
unsigned ling mul(int n);
int main()
{
    int m;
    puts("Calculate n! n=?\n");
    scanf("%d", &m);              /*键盘输入数据*/
    printf("%d!=%ld\n", m, mul(m));/*调用子程序计算并输出*/
    getch();
    retun 0;
}
unsigned long mul(int n)
{
    unsigned long p;
    if(n>1)
        p=n*mul(n-1);          /*递归调用计算 n!*/
    else
        p=1L;
    return(p);                 /*返回结果*/
}
```

运行结果:

```
    calculate n!    n=?
```

输入 5 时结果为:

```
    5!=120
```

C 语言基础教程(三十一)

函数(9)

3. 函数作用范围

Turbo C2.0 中每个函数都是独立的代码块，函数代码归该函数所有，除了对函数的调用以外，其它任何函数中的任何语句都不能访问它。例如使用跳转语句 goto 就不能从一个函数跳进其它函数内部。除非使用全程变量，否则一个函数内部定义的程序代码和数据，不会与另一个函数内的程序代码和数据相互影响。

Turbo C2.0 中所有函数的作用域都处于同一嵌套程度，即不能在一个函数内再说明或定义另一个函数。

Turbo C2.0 中一个函数对其它子函数的调用是全程的，即是函数在不同的文件中，也不必附加任何说明语句而被另一函数调用，也就是说一个函数对于整个程序都是可见的。

4. 函数的变量作用域

在 Turbo C2.0 中，变是可以在各个层次的子程序中加以说明，也就是说，在任何函数中，变量说明有只允许在一个函数体的开头处说明，而且允许变量的说明(包括初始化)跟在一个复合语句的左花括号的后面，直到配对的右花括号为止。它的作用域仅在这对花括号内，当程序执行到出花括号时，它将不复存在。当然，内层中的变量即使与外层中的变量名字相同，它们之间也是没有关系的。

例 9.

```
#include<stdio.h>
int i=10;
int main()
{
        int i=1;
        printf("%d\t, i);
        {
                int i=2;
                pritnf("%d\t", i);
                {
                        extern i;
                        i+=1;
                        printf("%d\t", i);
                }
                printf("%d\t", ++i);
        }
        printf("%d\n", ++i);
        return 0;
}
```

运行结果为

```
1       2       11      3       2
```

从程序运行的结果不难看出程序中各变量之间的关系，以及各个变量的作用域。

C 语言基础教程(三十二)

函数篇(a)

函数名: abort

功 能: 异常终止一个进程
用 法: void abort(void);
程序例:

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
printf("Calling abort()\n");
abort();
return 0; /* This is never reached */
}
```

函数名: abs
功 能: 求整数的绝对值
用 法: int abs(int i);
程序例:

```
#include <stdio.h>
#include <math.h>

int main(void)
{
int number = -1234;

printf("number: %d absolute value: %d\n", number, abs(number));
return 0;
}
```

函数名: absread, abswirte
功 能: 绝对磁盘扇区读、写数据
用 法: int absread(int drive, int nsects, int sectno, void *buffer);
int abswrite(int drive, int nsects, in tsectno, void *buffer);
程序例:

```
/* absread example */

#include <stdio.h>
```

```
#include <conio.h>
#include <process.h>
#include <dos.h>

int main(void)
{
int i, strt, ch_out, sector;
char buf[512];

printf("Insert a diskette into drive A and press any key\n");
getch();
sector = 0;
if (absread(0, 1, sector, &buf) != 0)
{
perror("Disk problem");
exit(1);
}
printf("Read OK\n");
strt = 3;
for (i=0; i<80; i++)
{
ch_out = buf[strt+i];
putchar(ch_out);
}
printf("\n");
return(0);
}
```

函数名: access
功 能: 确定文件的访问权限
用 法: int access(const char *filename, int amode);
程序例:

```
#include <stdio.h>
#include <io.h>

int file_exists(char *filename);

int main(void)
{
printf("Does NOTEXIST.FIL exist: %s\n",
file_exists("NOTEXISTS.FIL") ? "YES" : "NO");
return 0;
}
```

```
int file_exists(char *filename)
{
return (access(filename, 0) == 0);
}
```

函数名: acos
功 能: 反余弦函数
用 法: double acos(double x);
程序例:

```
#include <stdio.h>
#include <math.h>

int main(void)
{
double result;
double x = 0.5;

result = acos(x);
printf("The arc cosine of %lf is %lf\n", x, result);
return 0;
}
```

函数名: allocmem
功 能: 分配 DOS 存储段
用 法: int allocmem(unsigned size, unsigned *seg);
程序例:

```
#include <dos.h>
#include <alloc.h>
#include <stdio.h>

int main(void)
{
unsigned int size, segp;
int stat;

size = 64; /* (64 x 16) = 1024 bytes */
stat = allocmem(size, &segp);
if (stat == -1)
printf("Allocated memory at segment: %x\n", segp);
else
printf("Failed: maximum number of paragraphs available is %u\n",
stat);

return 0;
```

}

函数名: arc
功 能: 画一弧线
用 法: void far arc(int x, int y, int stangle, int endangle, int radius);
程序例:

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

int main(void)
{
/* request auto detection */
int gdriver = DETECT, gmode, errorcode;
int midx, midy;
int stangle = 45, endangle = 135;
int radius = 100;

/* initialize graphics and local variables */
initgraph(&gdriver, &gmode, "");

/* read result of initialization */
errorcode = graphresult(); /* an error occurred */
if (errorcode != grOk)
{
printf("Graphics error: %s\n", grapherrormsg(errorcode));
printf("Press any key to halt:");
getch();

exit(1); /* terminate with an error code */
}

midx = getmaxx() / 2;
midy = getmaxy() / 2;
setcolor(getmaxcolor());

/* draw arc */
arc(midx, midy, stangle, endangle, radius);

/* clean up */
getch();
closegraph();
return 0;
```

```
}
```

函数名: asctime
功 能: 转换日期和时间为 ASCII 码
用 法: char *asctime(const struct tm *tblock);
程序例:

```
#include <stdio.h>
#include <string.h>
#include <time.h>

int main(void)
{
struct tm t;
char str[80];

/* sample loading of tm structure */

t.tm_sec = 1; /* Seconds */
t.tm_min = 30; /* Minutes */
t.tm_hour = 9; /* Hour */
t.tm_mday = 22; /* Day of the Month */
t.tm_mon = 11; /* Month */
t.tm_year = 56; /* Year - does not include century */
t.tm_wday = 4; /* Day of the week */
t.tm_yday = 0; /* Does not show in asctime */
t.tm_isdst = 0; /* Is Daylight SavTime; does not show in asctime */

/* converts structure to null terminated
string */

strcpy(str, asctime(&t));
printf("%s\n", str);

return 0;
}
```

函数名: asin
功 能: 反正弦函数
用 法: double asin(double x);
程序例:

```
#include <stdio.h>
#include <math.h>

int main(void)
```

```
{
double result;
double x = 0.5;

result = asin(x);
printf("The arc sin of %lf is %lf\n", x, result);
return(0);
}
```

函数名: assert
功 能: 测试一个条件并可能使程序终止
用 法: void assert(int test);
程序例:

```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>

struct ITEM {
int key;
int value;
};

/* add item to list, make sure list is not null */
void additem(struct ITEM *itemptr) {
assert(itemptr != NULL);
/* add item to list */
}

int main(void)
{
additem(NULL);
return 0;
}
```

函数名: atan
功 能: 反正切函数
用 法: double atan(double x);
程序例:

```
#include <stdio.h>
#include <math.h>

int main(void)
{
double result;
```

```
double x = 0.5;

result = atan(x);
printf("The arc tangent of %lf is %lf\n", x, result);
return(0);
}
```

函数名: atan2
功 能: 计算 Y/X 的反正切值
用 法: double atan2(double y, double x);
程序例:

```
#include <stdio.h>
#include <math.h>

int main(void)
{
double result;
double x = 90.0, y = 45.0;

result = atan2(y, x);
printf("The arc tangent ratio of %lf is %lf\n", (y / x), result);
return 0;
}
```

函数名: atexit
功 能: 注册终止函数
用 法: int atexit(atexit_t func);
程序例:

```
#include <stdio.h>
#include <stdlib.h>

void exit_fn1(void)
{
printf("Exit function #1 called\n");
}

void exit_fn2(void)
{
printf("Exit function #2 called\n");
}

int main(void)
{
/* post exit function #1 */
```

```
atexit(exit_fn1);
/* post exit function #2 */
atexit(exit_fn2);
return 0;
}
```

函数名: atof
功 能: 把字符串转换成浮点数
用 法: double atof(const char *nptr);
程序例:

```
#include <stdlib.h>
#include <stdio.h>

int main(void)
{
float f;
char *str = "12345.67";

f = atof(str);
printf("string = %s float = %f\n", str, f);
return 0;
}
```

函数名: atoi
功 能: 把字符串转换成长整型数
用 法: int atoi(const char *nptr);
程序例:

```
#include <stdlib.h>
#include <stdio.h>

int main(void)
{
int n;
char *str = "12345.67";

n = atoi(str);
printf("string = %s integer = %d\n", str, n);
return 0;
}
```

函数名: atol
功 能: 把字符串转换成长整型数
用 法: long atol(const char *nptr);
程序例:

```
#include <stdlib.h>
#include <stdio.h>

int main(void)
{
long l;
char *str = "98765432";

l = atol(lstr);
printf("string = %s integer = %ld\n", str, l);
return(0);
}
```

C 语言基础教程(三十三)

函数篇(b)

函数名: bar
功 能: 画一个二维条形图
用 法: void far bar(int left, int top, int right, int bottom);
程序例:

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

int main(void)
{
/* request auto detection */
int gdriver = DETECT, gmode, errorcode;
int midx, midy, i;

/* initialize graphics and local variables */
initgraph(&gdriver, &gmode, "");

/* read result of initialization */
errorcode = graphresult();
if (errorcode != grOk) /* an error occurred */
{
printf("Graphics error: %s\n", grapherrormsg(errorcode));
printf("Press any key to halt:");
```

```
getch();
exit(1); /* terminate with an error code */
}

midx = getmaxx() / 2;
midy = getmaxy() / 2;

/* loop through the fill patterns */
for (i=SOLID_FILL; i<USER_FILL; i++)
{
/* set the fill style */
setfillstyle(i, getmaxcolor());

/* draw the bar */
bar(midx-50, midy-50, midx+50,
midy+50);

getch();
}

/* clean up */
closegraph();
return 0;
}
```

函数名: bar3d
功 能: 画一个三维条形图
用 法: void far bar3d(int left, int top, int right, int bottom,
int depth, int topflag);
程序例:

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

int main(void)
{
/* request auto detection */
int gdriver = DETECT, gmode, errorcode;
int midx, midy, i;

/* initialize graphics, local variables */
initgraph(&gdriver, &gmode, "");
```

```
/* read result of initialization */
errorcode = graphresult();
if (errorcode != grOk) /* an error occurred */
{
printf("Graphics error: %s\n", grapherrormsg(errorcode));
printf("Press any key to halt:");
getch();
exit(1); /* terminate with error code */
}

midx = getmaxx() / 2;
midy = getmaxy() / 2;

/* loop through the fill patterns */
for (i=EMPTY_FILL; i<USER_FILL; i++)
{
/* set the fill style */
setfillstyle(i, getmaxcolor());

/* draw the 3-d bar */
bar3d(midx-50, midy-50, midx+50, midy+50, 10, 1);

getch();
}

/* clean up */
closegraph();
return 0;
}
```

函数名: bdos
功 能: DOS 系统调用
用 法: int bdos(int dosfun, unsigned dosdx, unsigned dosal);
程序例:

```
#include <stdio.h>
#include <dos.h>

/* Get current drive as 'A', 'B', ... */
char current_drive(void)
{
char curdrive;
```

```
/* Get current disk as 0, 1, ... */
curdrive = bdos(0x19, 0, 0);
return('A' + curdrive);
}

int main(void)
{
printf("The current drive is %c:\n", current_drive());
return 0;
}
```

函数名: bdosptr
功 能: DOS 系统调用
用 法: int bdosptr(int dosfun, void *argument, unsigned dosal);
程序例:

```
#include <string.h>
#include <stdio.h>
#include <dir.h>
#include <dos.h>
#include <errno.h>
#include <stdlib.h>

#define BUFLEN 80

int main(void)
{
char buffer[BUFLEN];
int test;

printf("Enter full pathname of a directory\n");
gets(buffer);

test = bdosptr(0x3B,buffer,0);
if(test)
{
printf("DOS error message: %d\n", errno);
/* See errno.h for error listings */
exit (1);
}

getcwd(buffer, BUFLEN);
printf("The current directory is: %s\n", buffer);
```

```
return 0;
}
```

函数名: bioscom
功 能: 串行 I/O 通信
用 法: int bioscom(int cmd, char abyte, int port);
程序例:

```
#include <bios.h>
#include <conio.h>

#define COM1 0
#define DATA_READY 0x100
#define TRUE 1
#define FALSE 0

#define SETTINGS ( 0x80 | 0x02 | 0x00 | 0x00)

int main(void)
{
int in, out, status, DONE = FALSE;

bioscom(0, SETTINGS, COM1);
cprintf("... BIOSCOM [ESC] to exit ...\n");
while (!DONE)
{
status = bioscom(3, 0, COM1);
if (status & DATA_READY)
if ((out = bioscom(2, 0, COM1) & 0x7F) != 0)
putch(out);
if (kbhit())
{
if ((in = getch()) == '\x1B')
DONE = TRUE;
bioscom(1, in, COM1);
}
}
return 0;
}
```

函数名: biosdisk
功 能: 软硬盘 I/O
用 法: int biosdisk(int cmd, int drive, int head, int track, int sector
int nsects, void *buffer);

程序例:

```
#include <bios.h>
#include <stdio.h>

int main(void)
{
int result;
char buffer[512];

printf("Testing to see if drive a: is ready\n");
result = biosdisk(4,0,0,0,0,1,buffer);
result &= 0x02;
(result) ? (printf("Drive A: Ready\n")) :
(printf("Drive A: Not Ready\n"));

return 0;
}
```

函数名: biosequip
功 能: 检查设备
用 法: int biosequip(void);
程序例:

```
#include <bios.h>
#include <stdio.h>

int main(void)
{
int result;
char buffer[512];

printf("Testing to see if drive a: is ready\n");
result = biosdisk(4,0,0,0,0,1,buffer);
result &= 0x02;
(result) ? (printf("Drive A: Ready\n")) :
(printf("Drive A: Not Ready\n"));

return 0;
}
```

函数名: bioskey
功 能: 直接使用 BIOS 服务的键盘接口
用 法: int bioskey(int cmd);

程序例:

```
#include <stdio.h>
#include <bios.h>
#include <ctype.h>

#define RIGHT 0x01
#define LEFT 0x02
#define CTRL 0x04
#define ALT 0x08

int main(void)
{
int key, modifiers;

/* function 1 returns 0 until a key is pressed */
while (bioskey(1) == 0);

/* function 0 returns the key that is waiting */
key = bioskey(0);

/* use function 2 to determine if shift keys were used */
modifiers = bioskey(2);
if (modifiers)
{
printf("[");
if (modifiers & RIGHT) printf("RIGHT");
if (modifiers & LEFT) printf("LEFT");
if (modifiers & CTRL) printf("CTRL");
if (modifiers & ALT) printf("ALT");
printf("]");
}
/* print out the character read */
if (isalnum(key & 0xFF))
printf("'%c'\n", key);
else
printf("%#02x\n", key);
return 0;
}
```

函数名: biosmemory
功 能: 返回存储块大小
用 法:int biosmemory(void);
程序例:

```
#include <stdio.h>
#include <bios.h>

int main(void)
{
int memory_size;

memory_size = biosmemory(); /* returns value up to 640K */
printf("RAM size = %dK\n",memory_size);
return 0;
}
```

函数名: biosprint
功 能: 直接使用 BIOS 服务的打印机 I/O
用 法: int biosprint(int cmd, int byte, int port);
程序例:

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

int main(void)
{
#define STATUS 2 /* printer status command */
#define PORTNUM 0 /* port number for LPT1 */

int status, abyte=0;

printf("Please turn off your printer. Press any key to continue\n");
getch();
status = biosprint(STATUS, abyte, PORTNUM);
if (status & 0x01)
printf("Device time out.\n");
if (status & 0x08)
printf("I/O error.\n");

if (status & 0x10)
printf("Selected.\n");
if (status & 0x20)
printf("Out of paper.\n");

if (status & 0x40)
printf("Acknowledge.\n");
```

```
if (status & 0x80)
printf("Not busy.\n");

return 0;
}
```

函数名: biostime
功 能: 读取或设置 BIOS 时间
用 法: long biostime(int cmd, long newtime);
程序例:

```
#include <stdio.h>
#include <bios.h>
#include <time.h>
#include <conio.h>

int main(void)
{
long bios_time;

clrscr();
cprintf("The number of clock ticks since midnight is:\r\n");
cprintf("The number of seconds since midnight is:\r\n");
cprintf("The number of minutes since midnight is:\r\n");
cprintf("The number of hours since midnight is:\r\n");
cprintf("\r\nPress any key to quit:");
while(!kbhit())
{
bios_time = biostime(0, 0L);

gotoxy(50, 1);
cprintf("%lu", bios_time);

gotoxy(50, 2);
cprintf("%.4f", bios_time / CLK_TCK);

gotoxy(50, 3);
cprintf("%.4f", bios_time / CLK_TCK / 60);

gotoxy(50, 4);
cprintf("%.4f", bios_time / CLK_TCK / 3600);
}
return 0;
}
```

函数名: brk
功 能: 改变数据段空间分配
用 法: int brk(void *endds);
程序例:

```
#include <stdio.h>
#include <alloc.h>

int main(void)
{
char *ptr;

printf("Changing allocation with brk()\n");
ptr = malloc(1);
printf("Before brk() call: %lu bytes free\n", coreleft());
brk(ptr+1000);
printf(" After brk() call: %lu bytes free\n", coreleft());
return 0;
}
```

函数名: bsearch
功 能: 二分法搜索
用 法: void *bsearch(const void *key, const void *base, size_t *nelem,
size_t width, int(*fcmp)(const void *, const *));
程序例:

```
#include <stdlib.h>
#include <stdio.h>

#define NELEMS(arr) (sizeof(arr) / sizeof(arr[0]))

int numarray[] = {123, 145, 512, 627, 800, 933};

int numeric (const int *p1, const int *p2)
{
return(*p1 - *p2);
}

int lookup(int key)
{
int *itemptr;

/* The cast of (int(*)(const void *,const void*))
```

```
is needed to avoid a type mismatch error at
compile time */
itemptr = bsearch (&key, numarray, NELEMS(numarray),
sizeof(int), (int(*)(const void *,const void *))numeric);
return (itemptr != NULL);
}

int main(void)
{
if (lookup(512))
printf("512 is in the table.\n");
else
printf("512 isn't in the table.\n");

return 0;
}
```

C 语言基础教程(三十四)

函数篇(c)

函数名: cabs
功 能: 计算复数的绝对值
用 法: double cabs(struct complex z);
程序例:

```
#include <stdio.h>
#include <math.h>

int main(void)
{
struct complex z;
double val;

z.x = 2.0;
z.y = 1.0;
val = cabs(z);

printf("The absolute value of %.2lfi %.2lfj is %.2lf", z.x, z.y, val);
return 0;
}
```

函数名: calloc

功 能: 分配主存储器
用 法: void *calloc(size_t nelem, size_t elsize);
程序例:

```
#include <stdio.h>
#include <alloc.h>

int main(void)
{
char *str = NULL;

/* allocate memory for string */
str = calloc(10, sizeof(char));

/* copy "Hello" into string */
strcpy(str, "Hello");

/* display string */
printf("String is %s\n", str);

/* free memory */
free(str);

return 0;
}
```

函数名: ceil
功 能: 向上舍入
用 法: double ceil(double x);
程序例:

```
#include <math.h>
#include <stdio.h>

int main(void)
{
double number = 123.54;
double down, up;

down = floor(number);
up = ceil(number);

printf("original number %5.2lf\n", number);
printf("number rounded down %5.2lf\n", down);
```

```
printf("number rounded up %5.2lf\n", up);

return 0;
}
```

函数名: cgets
功 能: 从控制台读字符串
用 法: char *cgets(char *str);
程序例:

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

int main(void)
{
char buffer[83];
char *p;

/* There's space for 80 characters plus the NULL terminator */
buffer[0] = 81;

printf("Input some chars:");
p = cgets(buffer);
printf("\ncgets read %d characters: \"%s\"\n", buffer[1], p);
printf("The returned pointer is %p, buffer[0] is at %p\n", p, &buffer);

/* Leave room for 5 characters plus the NULL terminator */
buffer[0] = 6;

printf("Input some chars:");
p = cgets(buffer);
printf("\ncgets read %d characters: \"%s\"\n", buffer[1], p);
printf("The returned pointer is %p, buffer[0] is at %p\n", p, &buffer);
return 0;
}
```

函数名: chdir
功 能: 改变工作目录
用 法: int chdir(const char *path);
程序例:

```
#include <stdio.h>
#include <stdlib.h>
#include <dir.h>
```

```
char old_dir[MAXDIR];
char new_dir[MAXDIR];

int main(void)
{
if (getcurdir(0, old_dir))
{
perror("getcurdir()");
exit(1);
}
printf("Current directory is: \\%s\n", old_dir);

if (chdir("\\"))
{
perror("chdir()");
exit(1);
}

if (getcurdir(0, new_dir))
{
perror("getcurdir()");
exit(1);
}
printf("Current directory is now: \\%s\n", new_dir);

printf("\nChanging back to orignal directory: \\%s\n", old_dir);
if (chdir(old_dir))
{
perror("chdir()");
exit(1);
}

return 0;
}
```

函数名: _chmod, chmod
功 能: 改变文件的访问方式
用 法: int chmod(const char *filename, int permiss);
程序例:

```
#include <sys\stat.h>
#include <stdio.h>
#include <io.h>
```

```
void make_read_only(char *filename);

int main(void)
{
make_read_only("NOTEXIST.FIL");
make_read_only("MYFILE.FIL");
return 0;
}

void make_read_only(char *filename)
{
int stat;

stat = chmod(filename, S_IREAD);
if (stat)
printf("Couldn't make %s read-only\n", filename);
else
printf("Made %s read-only\n", filename);
}
```

函数名: chsize
功 能: 改变文件大小
用 法: int chsize(int handle, long size);
程序例:

```
#include <string.h>
#include <fcntl.h>
#include <io.h>

int main(void)
{
int handle;
char buf[11] = "0123456789";

/* create text file containing 10 bytes */
handle = open("DUMMY.FIL", O_CREAT);
write(handle, buf, strlen(buf));

/* truncate the file to 5 bytes in size */
chsize(handle, 5);

/* close the file */
close(handle);
```

```
return 0;
}
```

函数名: circle
功 能: 在给定半径以(x, y)为圆心画圆
用 法: void far circle(int x, int y, int radius);
程序例:

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

int main(void)
{
/* request auto detection */
int gdriver = DETECT, gmode, errorcode;
int midx, midy;
int radius = 100;

/* initialize graphics and local variables */
initgraph(&gdriver, &gmode, "");

/* read result of initialization */
errorcode = graphresult();
if (errorcode != grOk) /* an error occurred */
{
printf("Graphics error: %s\n", grapherrormsg(errorcode));
printf("Press any key to halt:");
getch();
exit(1); /* terminate with an error code */
}

midx = getmaxx() / 2;
midy = getmaxy() / 2;
setcolor(getmaxcolor());

/* draw the circle */
circle(midx, midy, radius);

/* clean up */
getch();
closegraph();
return 0;
```

```
}
```

C 语言基础教程(三十五)

函数篇(c)

函数名: cleardevice
功 能: 清除图形屏幕
用 法: void far cleardevice(void);
程序例:

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

int main(void)
{
/* request auto detection */
int gdriver = DETECT, gmode, errorcode;
int midx, midy;

/* initialize graphics and local variables */
initgraph(&gdriver, &gmode, "");

/* read result of initialization */
errorcode = graphresult();
if (errorcode != grOk) /* an error occurred */
{
printf("Graphics error: %s\n", grapherrormsg(errorcode));
printf("Press any key to halt:");
getch();
exit(1); /* terminate with an error code */
}

midx = getmaxx() / 2;
midy = getmaxy() / 2;
setcolor(getmaxcolor());

/* for centering screen messages */
settextjustify(CENTER_TEXT, CENTER_TEXT);

/* output a message to the screen */
```

```
outtextxy(midx, midy, "press any key to clear the screen:");

/* wait for a key */
getch();

/* clear the screen */
cleardevice();

/* output another message */
outtextxy(midx, midy, "press any key to quit:");

/* clean up */
getch();
closegraph();
return 0;
}
```

函数名: clearerr
功 能: 复位错误标志
用 法:void clearerr(FILE *stream);
程序例:

```
#include <stdio.h>

int main(void)
{
FILE *fp;
char ch;

/* open a file for writing */
fp = fopen("DUMMY.FIL", "w");

/* force an error condition by attempting to read */
ch = fgetc(fp);
printf("%c\n",ch);

if (ferror(fp))
{
/* display an error message */
printf("Error reading from DUMMY.FIL\n");

/* reset the error and EOF indicators */
clearerr(fp);
```

```
}

fclose(fp);
return 0;
}
```

函数名: clearviewport
功 能: 清除图形视区
用 法: void far clearviewport(void);
程序例:

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

#define CLIP_ON 1 /* activates clipping in viewport */

int main(void)
{
/* request auto detection */
int gdriver = DETECT, gmode, errorcode;
int ht;

/* initialize graphics and local variables */
initgraph(&gdriver, &gmode, "");

/* read result of initialization */
errorcode = graphresult();
if (errorcode != grOk) /* an error occurred */
{
printf("Graphics error: %s\n", grapherrormsg(errorcode));
printf("Press any key to halt:");
getch();
exit(1); /* terminate with an error code */
}

setcolor(getmaxcolor());
ht = textheight("W");

/* message in default full-screen viewport */
outtextxy(0, 0, "* <-- (0, 0) in default viewport");
```

```
/* create a smaller viewport */
setviewport(50, 50, getmaxx()-50, getmaxy()-50, CLIP_ON);

/* display some messages */
outtextxy(0, 0, "* <-- (0, 0) in smaller viewport");
outtextxy(0, 2*ht, "Press any key to clear viewport:");

/* wait for a key */
getch();

/* clear the viewport */
clearviewport();

/* output another message */
outtextxy(0, 0, "Press any key to quit:");

/* clean up */
getch();
closegraph();
return 0;
}
```

函数名: _close, close
功 能: 关闭文件句柄
用 法: int close(int handle);
程序例:

```
#include <string.h>
#include <stdio.h>
#include <fcntl.h>
#include <io.h>

main()
{
int handle;
char buf[11] = "0123456789";

/* create a file containing 10 bytes */
handle = open("NEW.FIL", O_CREAT);
if (handle > -1)
{
write(handle, buf, strlen(buf));
```

```
/* close the file */
close(handle);
}
else
{
printf("Error opening file\n");
}
return 0;
}
```

函数名: clock
功 能: 确定处理器时间
用 法: clock_t clock(void);
程序例:

```
#include <time.h>
#include <stdio.h>
#include <dos.h>

int main(void)
{
clock_t start, end;
start = clock();

delay(2000);

end = clock();
printf("The time was: %f\n", (end - start) / CLK_TCK);

return 0;
}
```

C 语言基础教程(三十六)

函数篇(c)

函数名: closegraph
功 能: 关闭图形系统
用 法: void far closegraph(void);
程序例:

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

int main(void)
{
/* request auto detection */
int gdriver = DETECT, gmode, errorcode;
int x, y;

/* initialize graphics mode */
initgraph(&gdriver, &gmode, "");

/* read result of initialization */
errorcode = graphresult();

if (errorcode != grOk) /* an error
occurred */
{
printf("Graphics error: %s\n", grapherrormsg(errorcode));
printf("Press any key to halt:");
getch();
exit(1); /* terminate with an error code */
}

x = getmaxx() / 2;
y = getmaxy() / 2;

/* output a message */
settextjustify(CENTER_TEXT, CENTER_TEXT);
outtextxy(x, y, "Press a key to close the graphics system:");

/* wait for a key */
getch();

/* closes down the graphics system */
closegraph();

printf("We're now back in text mode.\n");
printf("Press any key to halt:");
getch();
return 0;
}
```

函数名: clreol
功 能: 在文本窗口中清除字符到行末
用 法: void clreol(void);
程序例:

```
#include <conio.h>

int main(void)

{
clrscr();
cprintf("The function CLREOL clears all characters from the\r\n");
cprintf("cursor position to the end of the line within the\r\n");
cprintf("current text window, without moving the cursor.\r\n");
cprintf("Press any key to continue . . .");
gotoxy(14, 4);
getch();

clreol();
getch();

return 0;
}
```

函数名: clrscr
功 能: 清除文本模式窗口
用 法: void clrscr(void);
程序例:

```
#include <conio.h>

int main(void)
{
int i;

clrscr();
for (i = 0; i < 20; i++)
cprintf("%d\r\n", i);
cprintf("\r\nPress any key to clear screen");
getch();
```

```
clrscr();
cprintf("The screen has been cleared!");
getch();

return 0;
}
```

函数名: coreleft
功 能: 返回未使用内存的大小
用 法: unsigned coreleft(void);
程序例:

```
#include <stdio.h>
#include <alloc.h>

int main(void)
{
printf("The difference between the highest allocated block and\n");
printf("the top of the heap is: %lu bytes\n", (unsigned long) coreleft());

return 0;
}
```

函数名: cos
功 能: 余弦函数
用 法: double cos(double x);
程序例:

```
#include <stdio.h>
#include <math.h>

int main(void)
{
double result;
double x = 0.5;

result = cos(x);
printf("The cosine of %lf is %lf\n", x, result);
return 0;
}
```

函数名: cosh
功 能: 双曲余弦函数
用 法: dluble cosh(double x);
程序例:

```
#include <stdio.h>
#include <math.h>

int main(void)
{
double result;
double x = 0.5;

result = cosh(x);
printf("The hyperboic cosine of %lf is %lf\n", x, result);
return 0;
}
```

函数名: country
功 能: 返回与国家有关的信息
用 法: struct COUNTRY *country(int countrycode, struct country *country);
程序例:

```
#include <dos.h>
#include <stdio.h>

#define USA 0

int main(void)
{
struct COUNTRY country_info;

country(USA, &country_info);
printf("The currency symbol for the USA is: %s\n",
country_info.co_curr);
return 0;
}
```

C 语言基础教程(三十七)

函数篇(c)

函数名: cprintf
功 能: 送格式化输出至屏幕
用 法: int cprintf(const char *format[, argument, ...]);
程序例:

```
#include <conio.h>

int main(void)
{
/* clear the screen */
clrscr();

/* create a text window */
window(10, 10, 80, 25);

/* output some text in the window */
cprintf("Hello world\r\n");

/* wait for a key */
getch();
return 0;
}
```

函数名: cputs
功 能: 写字符到屏幕
用 法: void cputs(const char *string);
程序例:

```
#include <conio.h>

int main(void)
{
/* clear the screen */
clrscr();

/* create a text window */
window(10, 10, 80, 25);

/* output some text in the window */
cputs("This is within the window\r\n");

/* wait for a key */
getch();
```

```
return 0;
}
```

函数名: _creat creat
功 能: 创建一个新文件或重写一个已存在的文件
用 法: int creat (const char *filename, int permiss);
程序例:

```
#include <sys\stat.h>
#include <string.h>
#include <fcntl.h>
#include <io.h>

int main(void)
{
int handle;
char buf[11] = "0123456789";

/* change the default file mode from text to binary */
_fmode = O_BINARY;

/* create a binary file for reading and writing */
handle = creat("DUMMY.FIL", S_IREAD | S_IWRITE);

/* write 10 bytes to the file */
write(handle, buf, strlen(buf));

/* close the file */
close(handle);
return 0;
}
```

函数名: creatnew
功 能: 创建一个新文件
用 法: int creatnew(const char *filename, int attrib);
程序例:

```
#include <string.h>
#include <stdio.h>
#include <errno.h>
#include <dos.h>
#include <io.h>
```

```
int main(void)
{
int handle;
char buf[11] = "0123456789";

/* attempt to create a file that doesn't already exist */
handle = creatnew("DUMMY.FIL", 0);

if (handle == -1)
printf("DUMMY.FIL already exists.\n");
else
{
printf("DUMMY.FIL successfully created.\n");
write(handle, buf, strlen(buf));
close(handle);
}
return 0;
}
```

函数名: creattemp
功 能: 创建一个新文件或重写一个已存在的文件
用 法: int creattemp(const char *filename, int attrib);
程序例:

```
#include <string.h>
#include <stdio.h>
#include <io.h>

int main(void)
{
int handle;
char pathname[128];

strcpy(pathname, "\\");

/* create a unique file in the root directory */
handle = creattemp(pathname, 0);

printf("%s was the unique file created.\n", pathname);
close(handle);
return 0;
}
```

函数名: cscanf
功 能: 从控制台执行格式化输入
用 法: int cscanf(char *format[,argument, ...]);
程序例:

```
#include <conio.h>

int main(void)
{
char string[80];

/* clear the screen */
clrscr();

/* Prompt the user for input */
cprintf("Enter a string with no spaces:");

/* read the input */
cscanf("%s", string);

/* display what was read */
cprintf("\r\nThe string entered is: %s", string);
return 0;
}
```

函数名: ctime
功 能: 把日期和时间转换为字符串
用 法: char *ctime(const time_t *time);
程序例:

```
#include <stdio.h>
#include <time.h>

int main(void)
{
time_t t;

time(&t);
printf("Today's date and time: %s\n", ctime(&t));
return 0;
}
```

函数名: ctrlbrk
功 能: 设置 Ctrl-Break 处理程序
用 法: void ctrlbrk(*fptr)(void);
程序例:

```
#include <stdio.h>
#include <dos.h>

#define ABORT 0

int c_break(void)
{
printf("Control-Break pressed. Program aborting ...\n");
return (ABORT);
}

int main(void)
{
ctrlbrk(c_break);
for(;;)
{
printf("Looping... Press <Ctrl-Break> to quit:\n");
}
return 0;
}
```

C 语言基础教程(三十八)
作者：网络收集 ［责编:爆破手］

函数篇(d)

函数名: delay
功 能: 将程序的执行暂停一段时间(毫秒)
用 法: void delay(unsigned milliseconds);
程序例:

```
/* Emits a 440-Hz tone for 500 milliseconds */
#include <dos.h>

int main(void)
{
sound(440);
```

```
delay(500);
nosound();

return 0;
}
```

函数名: delline
功 能: 在文本窗口中删去一行
用 法: void delline(void);
程序例:

```
#include <conio.h>

int main(void)
{
clrscr();
cprintf("The function DELLINE deletes \
the line containing the\r\n");
cprintf("cursor and moves all lines \
below it one line up.\r\n");
cprintf("DELLINE operates within the \
currently active text\r\n");
cprintf("window. Press any key to \
continue . . .");
gotoxy(1,2); /* Move the cursor to the
second line and first column */
getch();

delline();
getch();

return 0;
}
```

函数名: detectgraph
功 能: 通过检测硬件确定图形驱动程序和模式
用 法: void far detectgraph(int far *graphdriver, int far *graphmode);
程序例:

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

```
/* names of the various cards supported */
char *dname[] = { "requests detection",
"a CGA",
"an MCGA",
"an EGA",
"a 64K EGA",
"a monochrome EGA",
"an IBM 8514",
"a Hercules monochrome",
"an AT&T 6300 PC",
"a VGA",
"an IBM 3270 PC"
};

int main(void)
{
/* returns detected hardware info. */
int gdriver, gmode, errorcode;

/* detect graphics hardware available */
detectgraph(&gdriver, &gmode);

/* read result of detectgraph call */
errorcode = graphresult();
if (errorcode != grOk) /* an error
occurred */
{
printf("Graphics error: %s\n", \
grapherrormsg(errorcode));
printf("Press any key to halt:");
getch();
exit(1); /* terminate with an error
code */
}

/* display the information detected */
clrscr();
printf("You have %s video display \
card.\n", dname[gdriver]);
printf("Press any key to halt:");
getch();
return 0;
}
```

函数名: difftime
功 能: 计算两个时刻之间的时间差
用 法: double difftime(time_t time2, time_t time1);
程序例:

```
#include <time.h>
#include <stdio.h>
#include <dos.h>
#include <conio.h>

int main(void)
{
time_t first, second;

clrscr();
first = time(NULL); /* Gets system
time */
delay(2000); /* Waits 2 secs */
second = time(NULL); /* Gets system time
again */

printf("The difference is: %f \
seconds\n",difftime(second,first));
getch();

return 0;
}
```

C 语言基础教程(三十九)

作者：网络收集 ［责编:爆破手］

函数篇(d)

函数名: disable
功 能: 屏蔽中断
用 法: void disable(void);
程序例:

```
/***NOTE: This is an interrupt service
routine. You cannot compile this program
with Test Stack Overflow turned on and
get an executable file that operates
correctly. */
```

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

#define INTR 0X1C /* The clock tick
interrupt */

void interrupt ( *oldhandler)(void);

int count=0;

void interrupt handler(void)
{
/* disable interrupts during the handling of
the interrupt */
disable();
/* increase the global counter */
count++;
/* reenable interrupts at the end of the
handler */
enable();
/* call the old routine */
oldhandler();
}

int main(void)
{
/* save the old interrupt vector */
oldhandler = getvect(INTR);

/* install the new interrupt handler */
setvect(INTR, handler);

/* loop until the counter exceeds 20 */
while (count < 20)
printf("count is %d\n",count);

/* reset the old interrupt handler */
setvect(INTR, oldhandler);

return 0;
}
```

函数名: div

功 能: 将两个整数相除, 返回商和余数
用 法: div_t (int number, int denom);
程序例:

```
#include <stdlib.h>
#include <stdio.h>

div_t x;

int main(void)
{
x = div(10,3);
printf("10 div 3 = %d remainder %d\n", x.quot, x.rem);

return 0;
}
```

函数名: dosexterr
功 能: 获取扩展 DOS 错误信息
用 法: int dosexterr(struct DOSERR *dblkp);
程序例:

```
#include <stdio.h>
#include <dos.h>

int main(void)
{
FILE *fp;
struct DOSERROR info;

fp = fopen("perror.dat","r");
if (!fp) perror("Unable to open file for
reading");
dosexterr(&info);

printf("Extended DOS error \
information:\n");
printf(" Extended error: \
%d\n",info.exterror);
printf(" Class: \
%x\n",info.class);
printf(" Action: \
%x\n",info.action);
printf(" Error Locus: \
```

```
%x\n",info.locus);

return 0;
}
```

函数名: dostounix
功 能: 转换日期和时间为 UNIX 时间格式
用 法: long dostounix(struct date *dateptr, struct time *timeptr);
程序例:

```
#include <time.h>
#include <stddef.h>
#include <dos.h>
#include <stdio.h>

int main(void)
{
time_t t;
struct time d_time;
struct date d_date;
struct tm *local;

getdate(&d_date);
gettime(&d_time);

t = dostounix(&d_date, &d_time);
local = localtime(&t);
printf("Time and Date: %s\n", \
asctime(local));

return 0;
}
```

函数名: drawpoly
功 能: 画多边形
用 法: void far drawpoly(int numpoints, int far *polypoints);
程序例:

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

int main(void)
```

```
{
/* request auto detection */
int gdriver = DETECT, gmode, errorcode;
int maxx, maxy;

/* our polygon array */
int poly[10];

/* initialize graphics and local
variables */
initgraph(&gdriver, &gmode, "");

/* read result of initialization */
errorcode = graphresult();
if (errorcode != grOk)
/* an error occurred */
{
printf("Graphics error: %s\n", \
grapherrormsg(errorcode));
printf("Press any key to halt:");
getch();
/* terminate with an error code */
exit(1);
}

maxx = getmaxx();
maxy = getmaxy();

poly[0] = 20; /* 1st vertext */
poly[1] = maxy / 2;

poly[2] = maxx - 20; /* 2nd */
poly[3] = 20;

poly[4] = maxx - 50; /* 3rd */
poly[5] = maxy - 20;

poly[6] = maxx / 2; /* 4th */
poly[7] = maxy / 2;
/*
drawpoly doesn't automatically close
the polygon, so we close it.
*/
poly[8] = poly[0];
```

```
poly[9] = poly[1];

/* draw the polygon */
drawpoly(5, poly);

/* clean up */
getch();
closegraph();
return 0;
}
```

C 语言基础教程(四十)

作者：网络收集 ［责编:爆破手］

函数篇(d)

函数名: dup
功 能: 复制一个文件句柄
用 法: int dup(int handle);
程序例:

```
#include <string.h>
#include <stdio.h>
#include <conio.h>
#include <io.h>

void flush(FILE *stream);

int main(void)
{
FILE *fp;
char msg[] = "This is a test";

/* create a file */
fp = fopen("DUMMY.FIL", "w");

/* write some data to the file */
fwrite(msg, strlen(msg), 1, fp);

clrscr();
printf("Press any key to flush \
DUMMY.FIL:");
getch();
```

```
/* flush the data to DUMMY.FIL without
closing it */
flush(fp);

printf("\nFile was flushed, Press any \
key to quit:");
getch();
return 0;
}

void flush(FILE *stream)
{
int duphandle;

/* flush TC's internal buffer */
fflush(stream);

/* make a duplicate file handle */
duphandle = dup(fileno(stream));

/* close the duplicate handle to flush the
DOS buffer */
close(duphandle);
}
```

函数名: dup2
功 能: 复制文件句柄
用 法: int dup2(int oldhandle, int newhandle);
程序例:

```
#include <sys\stat.h>
#include <string.h>
#include <fcntl.h>
#include <io.h>

int main(void)
{
#define STDOUT 1

int nul, oldstdout;
char msg[] = "This is a test";

/* create a file */
```

```
nul = open("DUMMY.FIL", O_CREAT | O_RDWR,
S_IREAD | S_IWRITE);

/* create a duplicate handle for standard
output */
oldstdout = dup(STDOUT);
/*
redirect standard output to DUMMY.FIL
by duplicating the file handle onto the
file handle for standard output.
*/
dup2(nul, STDOUT);

/* close the handle for DUMMY.FIL */
close(nul);

/* will be redirected into DUMMY.FIL */
write(STDOUT, msg, strlen(msg));

/* restore original standard output
handle */
dup2(oldstdout, STDOUT);

/* close duplicate handle for STDOUT */
close(oldstdout);

return 0;
}
```

C 语言基础教程(四十一)

作者：网络收集 ［责编:爆破手］

函数篇(e)

函数名: ecvt
功 能: 把一个浮点数转换为字符串
用 法: char ecvt(double value, int ndigit, int *decpt, int *sign);
程序例:

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

int main(void)
```

```
{
char *string;
double value;
int dec, sign;
int ndig = 10;

clrscr();
value = 9.876;
string = ecvt(value, ndig, &dec, &sign);
printf("string = %s dec = %d \
sign = %d\n", string, dec, sign);

value = -123.45;
ndig= 15;
string = ecvt(value,ndig,&dec,&sign);
printf("string = %s dec = %d sign = %d\n",
string, dec, sign);

value = 0.6789e5; /* scientific
notation */
ndig = 5;
string = ecvt(value,ndig,&dec,&sign);
printf("string = %s dec = %d\
sign = %d\n", string, dec, sign);

return 0;
}
```

函数名: ellipse
功 能: 画一椭圆
用 法: void far ellipse(int x, int y, int stangle, int endangle,
int xradius, int yradius);
程序例:

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

int main(void)
{
/* request auto detection */
int gdriver = DETECT, gmode, errorcode;
int midx, midy;
```

```
int stangle = 0, endangle = 360;
int xradius = 100, yradius = 50;

/* initialize graphics, local variables */
initgraph(&gdriver, &gmode, "");

/* read result of initialization */
errorcode = graphresult();
if (errorcode != grOk)
/* an error occurred */
{
printf("Graphics error: %s\n",
grapherrormsg(errorcode));
printf("Press any key to halt:");
getch();
exit(1);
/* terminate with an error code */
}

midx = getmaxx() / 2;
midy = getmaxy() / 2;
setcolor(getmaxcolor());

/* draw ellipse */
ellipse(midx, midy, stangle, endangle,
xradius, yradius);

/* clean up */
getch();
closegraph();
return 0;
}
```

函数名: enable
功 能: 开放硬件中断
用 法: void enable(void);
程序例:

```
/* ** NOTE:
This is an interrupt service routine. You can NOT compile this program
with Test Stack Overflow turned on and get an executable file which will
operate correctly.
*/
```

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

/* The clock tick interrupt */
#define INTR 0X1C

void interrupt ( *oldhandler)(void);

int count=0;

void interrupt handler(void)
{
/*
disable interrupts during the handling of the interrupt
*/
disable();
/* increase the global counter */
count++;
/*
re enable interrupts at the end of the handler
*/
enable();
/* call the old routine */
oldhandler();
}

int main(void)
{
/* save the old interrupt vector */
oldhandler = getvect(INTR);

/* install the new interrupt handler */
setvect(INTR, handler);

/* loop until the counter exceeds 20 */
while (count < 20)
printf("count is %d\n",count);

/* reset the old interrupt handler */
setvect(INTR, oldhandler);

return 0;
}
```