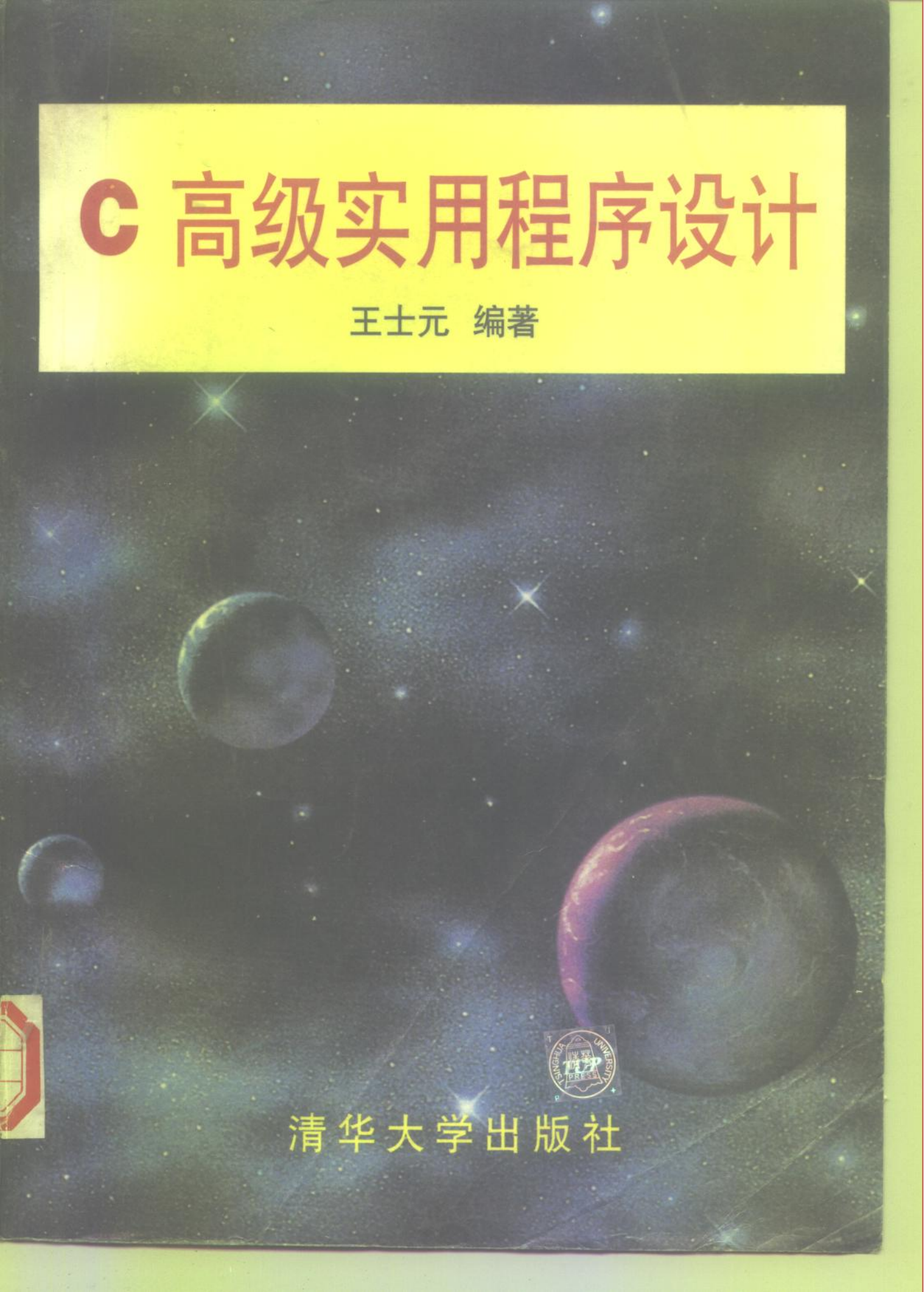
C 高级实用程序设计

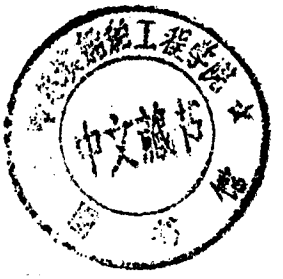
王士元 编著



清华大学出版社

C 高级实用程序设计

王士元 编著



清华大学出版社

(京)新登字 158 号

内 容 简 介

本书针对目前应用程序设计的热点，如中、英文菜单设计，画图，动画，中断程序，程序的驻留，屏幕图形的存取、打印，C 程序汉字显示技术，C 语言与汇编语言的混合编程，C 语言与 FoxBASE 的混合编程等进行设计示范，附有大量示例程序和注释。本书也用了部分篇幅对高级程序设计涉及的硬件及 C 中的文件、指针、内存分配、图形适配器等内容进行分析，并简单介绍了实用程序编程方法。本书适用于理工科本科生、研究生和广大计算机应用人员。

图书在版编目(CIP)数据

C 高级实用程序设计/王士元编著. —北京：清华大学出版社，1996.3
ISBN 7-302-02063-9

Ⅰ.C… Ⅱ.王… Ⅲ.C 语言-程序设计 Ⅳ.TP312C

中国版本图书馆 CIP 数据核字(95)第 23619 号

C 高级实用程序设计

出版者：清华大学出版社(北京清华大学校内，邮编 100084)
印刷者：北京大中印刷厂
发行者：新华书店总店北京科技发行所
开　本：787×1092　1/16　**印张**：32　**字数**：793 千字
版　次：1996 年 6 月第 1 版　1997 年 2 月第 2 次印刷
书　号：ISBN 7-302-02063-9/TP・959
印　数：6001—12000
定　价：32.00 元

前　言

C语言使用愈来愈普及，愈来愈深层次，许多读者，由于对知识的渴望或实际工作的需要，希望能更深入的了解一些C语言与PC机硬件的联系，希望能对C语言编程时遇到的一些技术难点给予帮助，这些难点诸如对PC机内存的分配与管理，80x86 CPU内部寄存器的使用，BIOS和DOS的调用，指针的分类与使用，堆和栈的结构及使用，文件的操作，对PC机输入输出接口的编程，中断处理程序的编写，驻留程序的设计，各种图形的绘制技术，中英文菜单程序设计，屏幕图形的存取与打印技术，C程序中的汉字使用及显示技术，C语言与汇编语言的混合编程，C语言与FoxBASE(dBASE)的接口技术等，这些问题也是目前C程序应用设计中的一些热点，它们涉及到PC机的硬件结构，中英文DOS系统和编程时积累的实践经验。

由于目前较多地看到的是一些有关C语言的教科书和参考书，对于上面提到的程序设计中的热点问题(它们应用于科研、教学、工程项目和软件产品中)的书较少见，这也是目前广大读者急需的。由于这方面的程序设计往往和PC机的系统结构联系在一起，属于C程序深层次的应用，故而称为高级实用程序设计。

笔者作为一个教师，一个科研工作者，一个较早搞计算机设计与应用且目前又直接讲授语言的人，了解一些学生及搞计算机应用开发的人们急需的要求，为此将自己的一些实践、体会、心得及参考到的新内容写到这本书中，奉献给读者，以期望能对进行C程序设计及软硬件开发人员能有所帮助。

由于Turbo C 2.0和Microsoft 5.0～7.0是目前大家较熟悉且又比较流行的C语言编译系统，而Turbo C有较良好的集成开发环境，所以本书以Turbo C 2.0为开发环境，书中的程序均在此环境中进行了调试，源程序集中在一张高密盘上，这些程序也可以方便地移植到别的编译系统上。

由于作者水平所限，加之时间仓促，书中错误及不妥之处在所难免，希望广大读者及专家批评指正。

作者在此对引入我出书的引路者——王家骅教授表示感谢，乔晓原、乔圆圆、王津涛、刘振实各位老师给予了不少帮助，李乐天、朱恩愈教授审阅了本书，提出了宝贵意见，在此深表谢意。

作　者

1994年12月于南开大学

目 录

第 1 章

概　述

C 语言是当前最流行的程序设计语言，它像其它高级语言一样，面向用户，面向解题的过程，编程者不必熟悉具体的计算机内部结构和指令；C 语言又像汇编语言一样，可以对机器硬件进行操作，如进行端口 I/O 操作、位操作、地址操作，并可内嵌汇编指令，将汇编指令当作它的语句一样。我们知道，汇编语言将涉及计算机硬件，所以 C 语言又像低级语言一样，可以对计算机硬件进行控制，因此人们把它称为介于高级语言与低级语言之间一种中级语言。

由于 C 语言的这种特点，因而它不但用于编一般应用程序，而且许多大的操作系统、编译系统也是由 C 语言来写的，甚至可以说 C 原来就是写系统软件的，因为它是和 UNIX 操作系统同时发展起来的，它最初用来写 UNIX 操作系统，由于 UNIX 的不断移植和推广，C 语言也就不断得到发展和普及，像后来的 PC-DOS，WORDSTAR，DBASE Ⅲ，PLUS 等都是由 C 语言和汇编语言相结合写成的。

1.1　C 语言的发展状况

1970 年美国贝尔实验室的 Ken Thompson 和 Dennis Ritchie 完成了 UNIX 的初版，与此同时，他们还改写了由 Martin Richards 开发的 BCPC 语言，形成了一种称为 B 的语言，此后 B 语言又进一步被进行了改进和完善，形成了称之为 C 的语言。

C 语言大约形成于 1972 年，1973 年 Dennis Ritchie 把 UNIX 系统中的 90％又用 C 语言进行了改写，并在 PDP-11 小型机上完成了用 C 语言及汇编语言编写的 UNIX 操作系统的调试，并将其投入运行，因而随着 UNIX 的移植、推广，C 语言也得到移植和推广，DOS 支持下的，甚至 Windows 支持下的 C 语言都相继出现，如我们目前广泛使用的在 PC 微机上的 MS-DOS 支持下的 Turbo C 和 Micrsoft C 就是典型的 C 语言版本。

由于众多的适用于不同机种、不同字长的 C 编译系统的出现（达几十种），给 C 语言的统一性和兼容性带来了困难，虽然它们基本遵循在 UNIX V 操作系统上配备的 C 语言标准，但仍有许多各自的特点，使得相互移植困难，不利于推广，为此美国国家标准局（ANSI）语言标准化委员会公布了一个 C 语言标准草案（83 ANSI C），该委员会又进一步修改完善该草案，终于在 1987 年公布了 C 语言标准，这就是目前投入商业运营的各种 C 编译系统所遵循的 87 ANSI C 语言标准，Turbo C、Microsoft C 均遵守这个标准。由于 PC 机的快速发展，原来的 C 标准已不能满足对 C 的功能要求，因而在不同的 C 语言版本中又增加了许多新的功能，它们可看作是对 C 标准的扩充和增强，如最明显的是 C 中增加了对屏幕分辨率的定义和一些图形功能的实现。

C 语言是一种结构程序设计的好语言，但随着软件处理对象从简单的数字和字符串发展到目前的图、文、声、象，信息量愈来愈大，愈来愈复杂，因而对程序的设计方法提出了更高的要求，随之在 80 年代出现了一种崭新的程序设计方法——面向对象的程序设计方法，结构程序设计的基本单位是模块，而面向对象的程序设计基本单位则是对象，因此 C 又进一步发展和充实，又出现了支持面向对象的程序设计语言C++，它实际上是 C 的超级集，其基本核心仍是 C。

例如美国 BorLand 公司 1989 年推出了 Turbo C 2.0，它又在继承和发扬 Turbo C 2.0 的集成开发环境的基础上，推出了面向对象的程序设计语言 Turbo C++，它实际上是 Turbo C 的一个超集，Turbo C++包含了所有 Turbo C 的内容，因而学好、用好 C，仍是面向对象程序设计的基础和前提，它甚至也可体现出面向对象设计的一些方法。

1.2 C 语言的编程格式

C 语言程序一般用小写字母，而仅在一些宏定义中，将常量名用大写字母表示，或对一些有特殊含义的变量，偶而也用大写表示，C 语言中对大小写字母是视作两个不同的量。

在 C 语言程序中没有程序行的概念，即在一行中可以任意书写多个语句，只要每个语句用分号作为结尾即可，多个语句还可用大括号{}括起来，形成一个如同单独语句一样的复合语句。一般情况下，为了层次清楚，一行只写一个语句，对复合语句也是按组成的语句依次写在不同的行中。

1.2.1 C 语言程序的结构特点

1. 一个 C 语言程序，通常由带有＃号的编译预处理语句开始，如＃include ＜文件名＞它表示，在编译源程序前，用指明的文件取代该预处理语句，通常文件名是指带有后缀为 .h 的磁盘文件，程序编译时，它将从磁盘中读出并插入到原来的预处理语句处，即预处理语句被所指明的包含文件(头文件)代替。

头文件通常是在程序中被调用函数的说明语句和该函数用到的一些符号常量的宏定义等，如在程序中要调用一些标准库函数时，系统提供了相应的头文件，它们其中的一些内容是对该函数的说明及该函数用到的符号常量的宏定义等，如对 fgets( )的说明放在 stdio.h 头文件中，在该文件中，包含了对 fgets( )函数的说明：

```
char *fgets(char *,int,file *)
```

对符号常量 NULL 的宏定义：

```
#define NULL 0
```

当然还包含对其它一些标准 I/O 函数的说明和宏定义等。

用户也可根据需要建立自己的头文件，如将程序中用到的符号常量组成一个单独的头文件，在程序开头用＃include 进行包含预处理，使得程序简捷方便。如在进行按键判别时，常要对键值(字符键值对应于 ASCII 码，功能键则为扩充的 ASCII 码)进行分析判别，以确定何键按下，键值为两字节的整型数，若高字节为 0，则表示低字节是代表相应的普通键的 ASCII 码；若低字节为 0，则高字节代表相应的功能键的扩充 ASCII 码。如在菜单程序设计中常用到一些功能键，为此可将它们的宏定义集合成一个文本文件，起名为 menu.h 头文

件：

```
#define     INSERT      0x5200
#define     ESC         0x0016
#define     TAB         0x0f09
#define     RETURN      0x000d
#define     RIGHT       0x4d00
#define     LEFT        0x4b00
#define     UP          0x4800
#define     DOWN        0x5000
#define     B           0x0e08
#define     HOME        0x4700
#define     END         0x4f00
#define     PGUP        0x4900
#define     PGDN        0x5100
#define     DEL         0x5300
#define     F1          0x3b00
#define     F2          0x3c00
#define     F3          0x3d00
#define     F4          0x3e00
#define     F5          0x3f00
#define     F6          0x4000
#define     F7          0x4100
#define     F8          0x4200
#define     F9          0x4300
#define     F10         0x4400
```

在程序开头用#include "menu.h"进行包含预处理后，在程序中就可用这些大写的键名，以代替难懂的键值，这样程序便显得明快易懂。不易因键值写错而导致程序有错。

当在模块化程序设计中，一个大的程序由多个文件组成，若这多个文件共同使用一些函数或常量时，可将这些共用的函数说明和符号常量的定义写在一个头文件中，然后在每个文件的开头用#include 进行包含说明，这样使得程序简捷、可靠，可维护性能提高。一般包含预处理有两种形式，即：

#include <文件名>

和 #include "文件名"

它们的区别在于对包含文件的查找路径不同，前者表示在安装编译系统时，在系统设定的标准目录中去查找，由于由系统提供的头文件一般放在 include 子目录下，所以常使用这种形式。

第二种形式表示在当前目录下去查找包含文件，若找不到，再到系统设定的目录中去查找。这种形式常用于用户自己写的包含文件，因为它们通常放在和用户程序同一个目录下。当然在这种形式中，文件名也可用文件路径名代替，这时，将在指定的文件路径中去查找文件。

文件包含也可用在程序中，如：

```
main ( )
  {
    ⋮
  #include "con.c"
    ⋮
  }
```

其中 con.c 是一个文本文件，编译时，它将从磁盘中调出，插入到该位置处。

一些在程序中用到的符号常量也常用编译预处理命令 #define 来进行定义，如在进行真假判断时，常用符号常量 TURE 和 FALSE 表示真假，这时可以定义：

```
#define   TURE 1
#define   FALSE 0
```

写在程序开头，这样在编译时在程序中出现 TURE 的地方，就表示为 1，即用 1 代替，出现 FALSE 的地方则用 0 代替。

当程序中经常要用到同一形式的表达式时，也可将该表达式定义为一个带参数的宏，使用时，如同调用函数一样，宏中写上参数即可。例如

求 x 和 y 中较大的一个：

```
#define max(x,y) (x>y ? x : y)
```

程序中经常要用到输出语句 printf("%d\n",x)，可定义为：

```
#define PRINTX printf("%d\n",x)
```

这样，只要在程序中需要求两个变量中的大者(如 a 和 b)，或需要上述格式的输出语句的地方，写上 max(a,b)或 PRINTX 即可。

2. 一个完整的 C 程序，总是由 main()函数开始，它像一个大型乐曲的引子，由此引出许多乐章——执行不同功能的函数；main()函数又像一个大型建筑的框架，它显示了要完成这个建筑的轮廓，这些轮廓就是由一个个的函数调用勾画出来的。因此可以说一个 C 程序是由一个个模块堆砌起来的，这个模块的最小单位是一个函数。当然模块也可以是一个源程序，它又由许多函数组成。所以 C 程序的设计，是一种模块化的设计，是许多函数的堆砌。因此在应用程序设计中，应将一个个的功能用一个个的函数来实现。

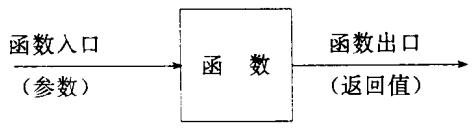
3. 函数的使用

Turbo C 提供了 400 多个标准库函数，每个函数都完成一定的功能，当程序需要执行这些功能时，只要调用这些函数即可，用户不用自己再去定义。这些函数包括输入输出函数、数学函数、字符串处理函数、内存函数、与 BIOS 和 DOS 调用有关的函数、屏幕和图形功能函数、过程控制函数、目录函数等。

当标准库函数中没有用户要用的功能函数时，就必须自己设计，设计函数的好方法是：

① 一个函数不宜处理太多的功能，要保持函数的小型化，功能单一化。

② 一个函数要保持自己的独立性，如同一个黑匣子一样，单进单出，即如下图如示：



这样一个函数编译后，其内部定义的程序代码与数据和另一函数的程序代码和数据不会相互影响，为此在函数中要使用局部变量，即它的生存期只维持在调用该函数并执行时，也就是说函数被激活时。要尽量少用或不用全局变量，它将破坏函数的独立性。函数的这种设计方法类似于面向对象设计方法中的封装性。

③ 可以在函数中多使用复合语句，使得函数也具有结构化，且可提高运行效率和节省存储空间。如假设一个函数可以划分成若干个逻辑上独立的程序段，将每个独立的程序段用复合语句表示之，然后用开关语句来控制它们的执行，即：

```
{
  ⋮
  switch (x) {
        case A : {…;break;}
        case B : {…;break;}
                 ⋮
        default : {…;}
```

使程序显得清晰明快。

又如一个函数要对三个整型数组 a[300]、b[400]、c[500]分别进行同一处理后进行存盘。为了节省空间，可用复合语句来设计函数结构：

```
array() {
    {int a[300];…;}        /*分程序 1*/
    {int b[400];…;}        /*分程序 2*/
    {int c[500];…;}        /*分程序 3*/
  }
```

这样相当于三个分程序，当执行完第一个分程序后，原分配的存储空间 300×2 字节将被收回，依次类推。实际上该函数仅用到了 500×2 个字节的空间，它比用形如 array(){int a[300],b[400],c[500],…}定义的结构节省了更多的空间。

在复合语句中定义一个局部变量的方法也可推广应用在其它程序的设计中。由于这种局部变量的生存期仅存在于复合语句的生命期中，复合语句执行完，它即失去作用，其占用的存储空间被系统收回，因而可节约存储空间，使程序层次清楚。

④ 在主函数前，要罗列出所有使用的自定义函数的原型说明，这有利于在大程序设计中追踪要调用的函数设置是否正确，如参数传递对否？返回值一致吗？在函数定义时，采用现代定义风格。例如：一函数说明为

```
int average (int x,int y)
main()
  {
  int x
  char c;
  register result
  x=20;
  c='0'
```

```
  result=average (x,c);
  printf ("\The average is %d",result);
}
int average (int x,int y)
  {
    return ((x+y)/2);
  }
```

这样编译时，将会告诉你函数参数类型不匹配，并将参数 c(字符型)转换为 2 字节的整型数放入堆栈中。若没有程序前的函数说明，它将按一个字节的字符放入堆栈中。所以使用函数说明，可保证参数类型的正确性。如在主函数中用如下调用：

```
ch=average (x,r,p);
```

其中 ch 定义为字符，r 为一实数，检查该函数时马上会发现返回值和传递的参数个数和函数原说明不一致，立即会发现错误，编译时，也会立即报错。

当编制大型程序，不同的源程序模块共用一些函数时，可将这些函数的说明放在一个头文件中，每个源程序用 #include 进行包含说明，这是程序设计中常用的方法。

⑤ 在程序中适当的地方用/*…*/加入注释，这便于程序的阅读和调试。一个较大程序，经过数月后，再读它，可能许多关键的地方已记不起了，加注释可帮助你恢复记忆，调试时，也易于找错。

⑥ 采用层次的书写程序格式，按程序的不同功能分层，对 for、while、do_while、if_else、switch_case 等一类控制语句或它们的多重嵌套，采用缩格方式，可用 TAB 键进行控制，如设 TAB 键为每按一次为后移 4 个格(缺省方式为 8 个格，当要改变此值时，可在 Turbo C 集成环境下选择 Options 项下的环境子菜单 Environment，然后用光标再移到该子菜单的 Tab Size 项下，选择缩格的格数)。例如：

```
# include <…>
# define M 450
void fun1(int a,int b)
int fun2(char *p,int c)
void main()
  {int x,y;
    ⋮
   for (i=0;i<M;i++)
    {
      while (…)
       { ⋮
        if (x>100)
          {
            ⋮
          }
         else
          {
            ⋮
```

```
            }
        }
    }
    ⋮
  printf(…);
}
```

1.2.2 模块化的程序设计

C 语言作为结构化的程序设计语言，易采用自顶向下的设计方法，即开始暂不涉及问题的实质和具体解决的步骤，而只是从问题的全局出发，给出一个概括性的抽象的描述。例如编写一个信号处理程序，它要求对信号数据经过数字处理后进行图形显示并存盘，因而程序大轮廓应是：

① 信号数据的输入
② 信号预处理
③ 信号进行数字处理
④ 进行显示
⑤ 进行存盘。

接着对各功能进行细分，例如对于信号的输入，又可分为：

① 通过 COM1 口由 RS-232 接口输入
② 由磁盘数据文件输入

对信号预处理又可分为：

① 对信号输入进行反序排列
② 用窗函数预处理

对数字处理又可分为：

① 求快速付立叶变换
② 求功率谱

对用窗函数预处理，又可分为：

① 海明窗处理
② 汉宁窗处理
③ 布拉格曼窗处理

其它功能类推。

在此细化的基础上再进行细化，以至于成为一个个单独的功能，因而可用一个个函数来实现。

设计一个个具体函数，就进入实质性阶段。要定义变量，要选取标准函数，要确定算法，这就是构造程序的基本单元。当一个个函数都设计完后，便可将这些函数按照调用的先后次序在主函数中堆砌起来，并用主函数作为总控程序，完成对它们的参数传递，控制选择对这些函数的调用，形成一个完整的实用的信号处理程序。

如要求数组 a[]的各元素和，奇数下标的元素的和，各元素中的最大值，各元素的平均值。设计程序时，可将要求实现的各项功能用函数实现，每个函数单独完成一个功能，数组传