新世纪

计算机基础教育丛书

丛书主编
谭浩强

C程序设计题解与上机指导（第二版）

（电子版）

谭浩强　主编

清华大学出版社
http://www.tup.tsinghua.edu.cn

第一部分 《C程序设计》(第二版)习题与参考解答

第1章 C语言概述

1.1 请根据自己的认识,写出C语言的主要特点。

解: 略。

1.2 C语言主要用途是什么? 它和其他高级语言有什么异同?

解: 略。

1.3 写出一个C程序的构成。

解: 略。

1.4 C语言以函数为程序的基本单位,有什么好处?

解: 略。

1.5 请参照本章例题,编写一个C程序,输出以下信息:

```
**************************
    Very  good!
**************************
```

解:

```
main ( )
{printf("************************** \n");
 printf ("\n");
 printf("      Very good! \n");
 printf(" \n");
 printf("************************** \n");
}
```

运行结果:

```
**************************
    Very  good!
**************************
```

1.6 编写一个程序,输入a、b、c三个值,输出其中最大者。

解：

```
main ( )
{int a,b,c,max;
 printf ("请输入三个数 a,b,c: \n");
 scanf ("%d,%d,%d",&a,&b,&c);
 max=a;
 if (max<b)
   max=b;
 if (max<c)
   max=c;
 printf (" 最大数为:%d",max);
}
```

运行结果：

```
请输入三个数 a,b,c:
6,5,1↙
最大数为:6
```

1.7 上机运行本章 3 个例题，熟悉所用系统的上机方法与步骤。

解： 略。

1.8 上机运行本章习题 1.5 和 1.6。

解： 略。

说明：为了便于读者阅读和理解程序，在本章的一些程序中字符串的内容为汉字信息，以便用汉字输出有关的信息。在后面几章的程序中有的加了汉字注释。这些汉字是在汉字操作系统(例如 UCDOS)的支持下和程序一起输入的。如果离开汉字操作系统，这些汉字是显示不出来的(显示出来的是一些乱码)。如果读者认为输入汉字不方便，也可以改用英文字符串和英文注释。

第2章　程序的灵魂——算法

2.1　什么是算法？试从日常生活中找3个例子，描述它们的算法。

解：略。

2.2　什么叫结构化的算法？为什么要提倡结构化的算法？

解：略。

2.3　试述三种基本结构的特点，你能否自己另外设计两种基本结构（要符合基本结构的特点）。

解：见图2.1和图2.2。

2.4　用传统流程图表示求解以下问题的算法。

(1) 有两个瓶子A和B，分别盛放醋和酱油，要求将它们互换（即A瓶原来盛醋，现改盛酱油，B瓶则相反）。

解：流程图见图2.3。

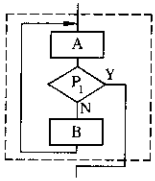


图　2.1

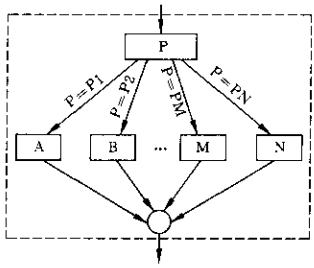


图　2.2

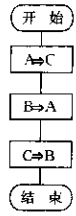


图　2.3

图中的符号“⇒”表示“倒给”，例如“$A⇒C$”表示“将A瓶中的液体倒给C瓶”。

(2) 依次将10个数输入，要求将其中最大的数打印出来。

解：流程图见图2.4。

(3) 有3个数a、b、c，要求按大小顺序把它们打印出来。

解：流程图见图2.5。

(4) 求$1+2+3+\cdots+100$。

解：流程图见图2.6。

(5) 判断一个数n能否同时被3和5整除。

解：流程图见图2.7(a)或图2.7(b)。

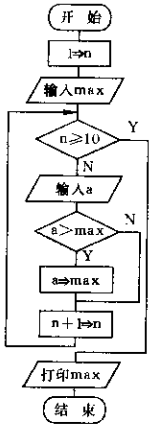


图 2.4

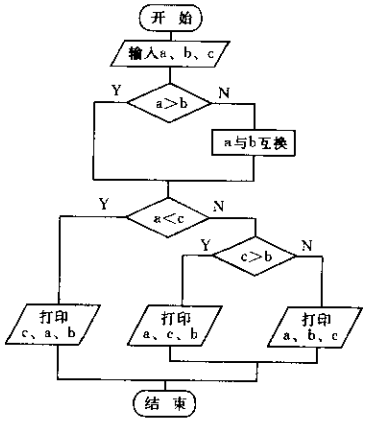


图 2.5

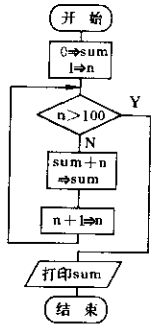
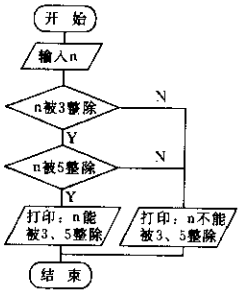
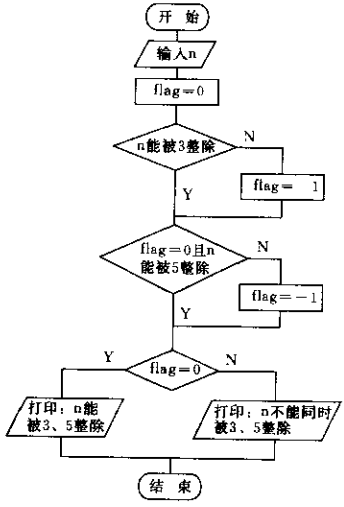


图 2.6



(a)



(b)

图 2.7

(6) 将 100～200 之间的素数打印出来。

解：流程图见图 2.8。

(7) 求两个数 m 和 n 的最大公约数。

解：流程图见图 2.9。

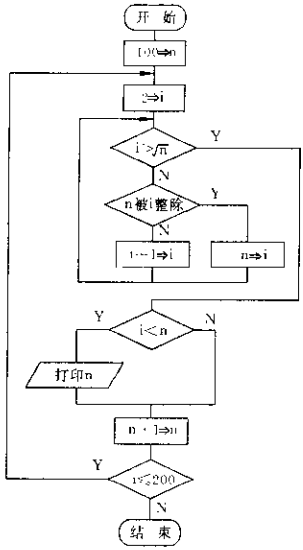


图 2.8

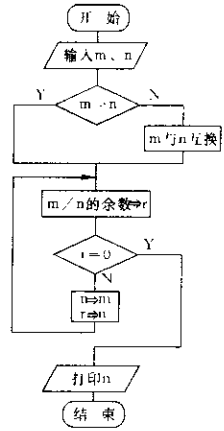


图 2.9

(8) 求方程式 $ax^2+bx+c=0$ 的根。分别考虑:① 有两个不等的实根;② 有两个相等的实根。

解: 流程图见图 2.10。

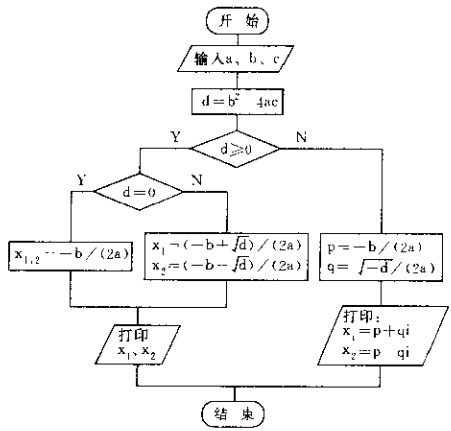


图 2.10

2.5 用 N-S 图表示 2.4 题中各小题的算法。

(1) 将 a 和 b 对换。

解：N-S 图见图 2.11。

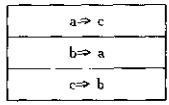


图 2.11

(2) 输出 10 个数中的最大数。

解：N-S 图见图 2.12。

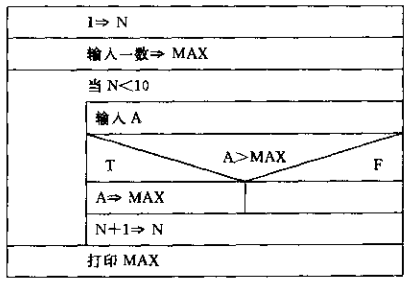


图 2.12

(3) 有 3 个数 a、b、c，要求按大小顺序把它们打印出来。

解：N-S 图见图 2.13。

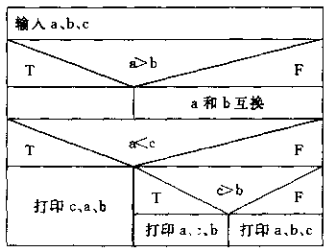


图 2.13

(4) 求 1+2+3+…+100。

解：N-S 图见图 2.14。

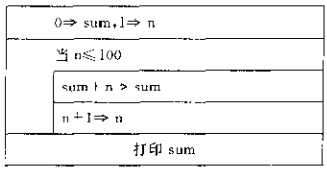


图 2.14

(5) 判断一个数 n 能否同时被 3 和 5 整除。

解： N S 图见图 2.15。

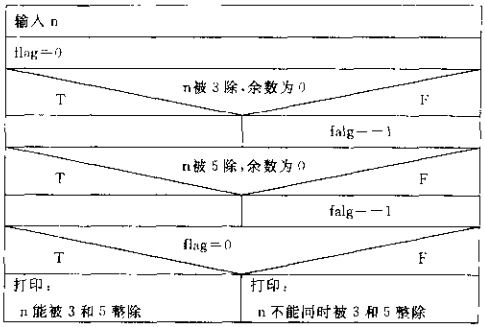


图 2.15

(6) 将 100～200 之间的素数打印出来。

解： N-S 图见图 2.16。

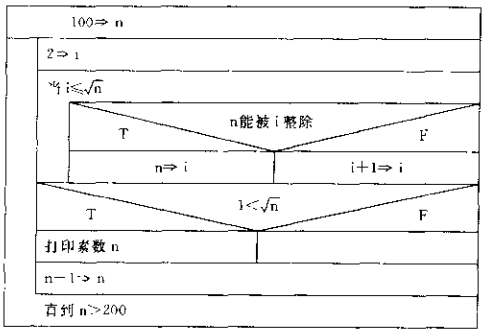


图 2.16

(7) 求两个数 m 和 n 的最大公约数。

解： N-S 图见图 2.17。

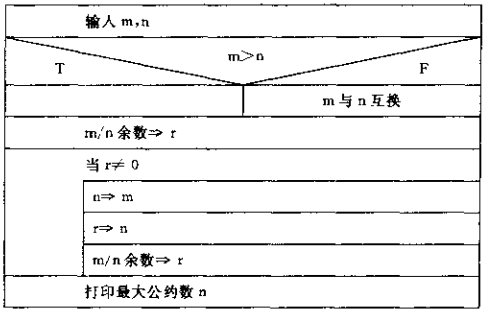


图 2.17

(8) 求方程式 $ax^2+bx+c=0$ 的根。分别考虑:① 有两个不等的实根;② 有两个相等的实根。

解: N-S 图见图 2.18。

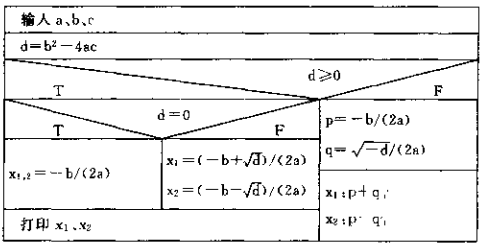


图 2.18

2.6 用伪代码表示 2.4 题中各题的算法。

(1) 将 a 和 b 对换。

解:

```
c=a
a=b
b=c
```

(2) 输出 10 个数中的最大数。

解:

```
n=1
input max
while n<10 do
  input a
  if a>max then max=a
```

```
    n=n+1
  end do
  print max
```

(3) 有 3 个数 a、b、c，要求按大小顺序把它们打印出来。

解：

```
  input a, b, c
  if a<b then swap a, b        (swap 表示互换)
  if a<c then
     print c, a, b
  else
    if c>b then
      print a, c, b
    else
      print a, b, c
    end if
  end if
```

(4) 求 1+2+3+…+100。

解：

```
  sum=0
  n=1
  while n≤100 do
    sum=sum+n
  n=n+1
  end do
  print sum
```

(5) 判断一个数 n 能否同时被 3 和 5 整除。

解：

```
  input n
  flag=0
  if mod (n, 3) ≠0 then flag=-1
  if mod (n, 5) ≠0 then flag=-1
  if flag=0 then
    print n"能被 3 和 5 整除"
  else
    print n "不能被 3 和 5 整除"
  end if
```

上面的 mod 代表求余，mod (n, 3) 表示 n 被 3 除得到的余数。

(6) 将 100～200 之间的素数打印出来。

解：

```
n=100
while n>200 do
  i=2
  while i≤√n do
    if mod (n, I) then
      i=n
    else
      i=i+1
    end if
  end do
  if i<√n then print n
  n=n+1
end do
```

(7) 求两个数 m 和 n 的最大公约数。

解：

```
input m, n
if m<n then swap m, n
r=mod (m, n)
while r≠0 do
  m=n
  n=r
  r=mod (m, n)
end do
print n
```

(8) 求方程式 $ax^2+bx+c=0$ 的根。分别考虑：① 有两个不等的实根；② 有两个相等的实根。

解：

```
input a, b, c
disc=b²-4ac
if disc≥0 then
  if disc=0 then
    x1, x2=-b/(2a)
  else
    x1=(-b+√disc) /(2a)
    x2=(-b-√disc) /(2a)
```

```
    end if
    print x1, x2
  else
    p=-b/(2a)
    q=√disc /(2a)
    print p-q, "+", p-q, "i"
  end if
```

2.7 什么叫结构化程序设计？它的主要内容是什么？

解：略。

2.8 用自顶向下、逐步细化的方法进行以下算法的设计：

(1) 打印出 1900—2000 年中是闰年的年份，闰年的条件是：① 能被 4 整除但不能被 100 整除；或② 能被 100 整除且能被 400 整除。

解：先画出图 2.19(a)，对它细化得到图 2.19(b)；再对图 2.19(b)中的 S1.1 细化，得图 2.19(c)。

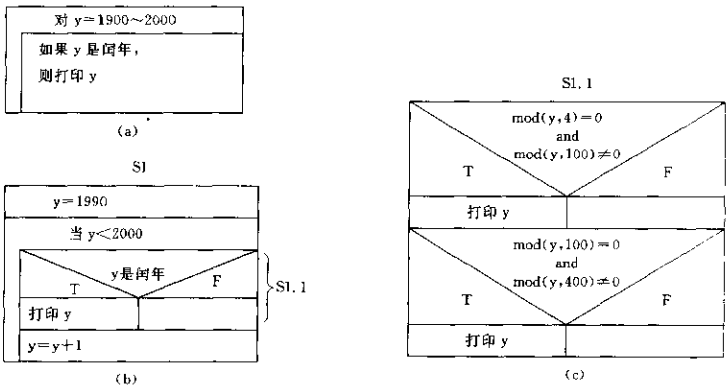


图 2.19

(2) 求 $ax^2+bx+c=0$ 的根。分别考虑 $d=b^2-4ac$ 大于 0、等于 0 和小于 0 三种情况。

解：先画出图 2.20(a)，对其中的 S3 细化为图 2.20(b)；再对图 2.20(b)中的 S3.1 细化为图 2.20(c)；对图 2.20(c)中的 S3.1.1 细化为图 2.20(d)；对图 2.20(c)中的 S3.1.2 细化为图 2.20(e)；最后对图 2.20(b)中的 S3.2 细化为图 2.20(f)。

请读者将它们合成一个总的 N-S 图。

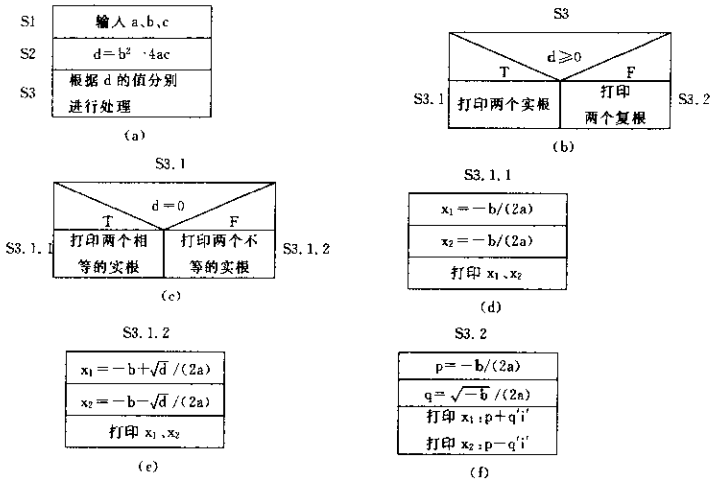


图 2.20

(3) 输入 10 个数，找出最大的一个数，并打印出来。

解：先初步画出图 2.21(a)。考虑到还没有引入数组的概念，因而不能做到将 100 个数全部输入到数组中，然后再从中找最大者。由于不采用数组这种数据结构，算法也应与采用数组时有所不同。现在只用普通变量，逐个读入数据，并把当前各数中的最大者保留下来，以便再与后面读入的数比较。将图 2.21(a)细化为图 2.21(b)，再细化为图 2.21(c)。

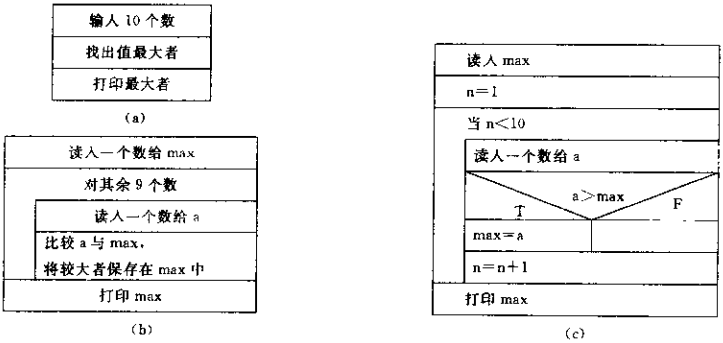


图 2.21

第3章　数据类型、运算符与表达式

3.1　请将C语言的数据类型和其他高级语言的数据类型做比较，C有哪些特点？

解：略。

3.2　C语言为什么要规定对所有用到的变量要"先定义，后使用"。这样做有什么好处？

解：略。

3.3　请将下面各数用八进制和十六进制数(补码)表示：

(1) 10　　(2) 32　　(3) 75　　(4) −617

(5) --111　　(6) 2483　　(7) --28654　　(8) 21003

解：

(1) $(10)_{10}=(12)_8=(a)_{16}$

(2) $(32)_{10}=(40)_8=(20)_{16}$

(3) $(75)_{10}=(113)_8=(4b)16$

(4) $(-617)_{10}=(176627)_8=(fd97)_{16}$

(5) $(-111)_{10}=(177621)_8=(ff91)_{16}$

(6) $(2483)_{10}=(4663)_8=(963)_{16}$

(7) $(-28654)_{10}=(110022)_8=(9012)_{16}$

(8) $(21003)_{10}=(51013)_8=(520b)_{16}$

3.4　将以下3个整数分别赋给不同类型的变量，请画出赋值后数据在内存中的存储形式。

变量的类型	25	-2	32769
int 型(16 位)			
long 型(32 位)			
sort 型(16 位)			
signed char(8 位)			
unsigned int 型			
unsigned long 型			
unsigned short 型			
unsigned char 型			

注：如果没有学过二进制和补码，此题可以不做。

解：各数据在内存中的存储形式如下表所示：

变量的类型	25	−2	32769
int 型(16 位)	8 / 00… 000011001	15 / 1111111111111110	14 / 100 … 001(溢出)
long 型(32 位)	24 / 00… 000011001	31 / 11 … 1110	16 / 00… 0100 / 14 / … 001
short 型(16 位)	8 / 100… 000011001	15 / 1111111111111110	14 / 100 … 001(溢出)
signed char(8 位)	100011001	11111110	00000001(溢出)
unsigned int 型	8 / 00… 000011001	15 / 111 … 110	14 / 100 … 001
unsigned long 型	24 / 00… 000011001	31 / 111 … 110	16 / 00 … 0 / 14 / 100 … 001
unsigned short 型	8 / 00… 000011001	15 / 111 … 110	5 / 100 … 001
unsigned char 型	00011001	11111110	00000001

3.5 字符常量和字符串常量有什么区别?

解: 字符常量是一个字符,用单引号括起来。字符串变量是由 0 个或若干个字符组合而成,用双引号把它们括起来,存储时自动在字符串最后加一个结束符号'\ 0'。

3.6 写出以下程序运行的结果:

```
main()
{ char c1='a', c2='b', c3='c', c4='\101', c5='\116';
  printf ("a%c b%c\t c%c\t abc\n", c1, c2, c3);
  printf ("\t\b%c %c", c4, c5);
}
```

解: 程序的运行结果为:

```
aa␣bb␣␣␣ cc␣␣␣␣␣␣abc
      A␣N
```

3.7 要将"China"译成密码,密码规律是:用原来的字母后面第 4 个字母代替原来的字母。例如,字母"A"后面第 4 个字母是"E",用"E"代替"A"。因此,"China"应译为"Glmre"。请编一程序,用赋初值的方法使 c1、c2、c3、c4、c5 5 个变量的值分别为'C'、'h'、'i'、'n'、'a',经过运算,使 c1、c2、c3、c4、c5 的值分别变为'G'、'l'、'm'、'r'、'e',并输出。

解:

```
main()
{char c1='C', c2='h', c3='i', c4='n', c5='a';
 c1+=4;
 c2+=4;
 c3+=4;
 c4+=4;
```

```
c5 = 4;
printf("密码是%c%c%c%c%c\n", c1, c2, c3, c4, c5);
}
```

运行结果：

```
密码是 Glmre
```

3.8 例 3.6 能否改成如下：

```
main()
{ int c1, c2;            (原为 char c1, c2)
  c1 = 97;
  c2 = 98;
  printf ("%c %c\n", c1, c2);
  printf ("%d %d\n", c1, c2);
}
```

解： 可以。因为在可输出的字符范围内，用整型和用字符型作用相同。

3.9 求下面算术表达式的值。

(1) x+a%3 * (int)(x+y)%2/4

设 x=2.5，a=7，y=4.7

(2) (float)(a+b)/2+(int)x%(int)y

设 a=2，b=3，x=3.5，y=2.5

解：

(1) 2.5

(2) 3.5

3.10 写出程序运行的结果。

```
main()
{ int i, j, m, n;
  i = 8;
  j = 10;
  m = ++i;
  n = j++;
  printf ("%d, %d, %d, %d", i, j, m, n);
}
```

解： 运行结果为：

```
9,11,9,10
```

3.11 写出下面赋值的结果。格中写了数值的是要将它赋给其他类型的变量，将所有空格填上赋值后的数值。

int	99					42	
char		'd'					×
unsigned int			76				65535
float				53.65			
long int					68		

解：

int	99	100	76	53	68	42	−1
char	'c'	'd'	'L'	'5'	'D'	'*'	×
unsigned int	99	100	76	53	68	42	65535
float	99.000000	100.000000	76.000000	53.65	68.000000	42.000000	65535.000000
long int	99	100	76	53	68	42	65535

3.12 写出下面表达式运算后 a 的值，设原来 a=12。设 a 和 n 都已定义为整型变量。

(1) a+=a　　(2) a−=2

(3) a*=2+3　　(4) a/=a+a

(5) a%=(n%=2)，n 的值等于 5　　(6) a+=a−=a*=a

解：

(1) 24　　(2) 10　　(3) 60　　(4) 0

(5) 0　　(6) 0

第4章　最简单的C程序设计
——顺序程序设计

4.1　C语言中的语句有哪几类？C语言与其他语言中的语句有哪些异同？

解： 略。

4.2　怎样区分表达式和表达式语句？C语言为什么要设表达式语句？什么时候用表达式，什么时候用表达式语句？

解： 略。

4.3　C语言为什么要把输入输出的功能作为函数，而不作为语言的基本部分？

解： 略。

4.4　若a=3，b=4，c=5，x=1.2，y=2.4，z=-3.6，u=51274，n=128765，c1='a'，c2='b'，想得到以下的输出格式和结果，请写出程序（包括定义变量类型和设计输出）

要求输出的结果如下：

```
a=␣3␣␣b=␣4␣␣c=␣5
x=1.200000,y=2.400000,z=-3.600000
x+y=␣3.60␣␣y+z=-1.20␣␣z+x=-2.40
u=␣51274␣␣n=␣␣␣128765
c1='a'␣or␣97(ASCII)
c2='b'␣or␣98(ASCII)
```

解：

```
main()
{
 int a,b,c;
 long int u,n;
 float x,y,z;
 char c1,c2;
 a=3;b=4;c=5;
 x=1.2;y=2.4;z=-3.6;
 u=51274;n=128765;
 c1='a';c2='b';
 printf("\n");
 printf("a=%2db=%2dc=%2d\n",a,b,c);
 printf("x=%8.6f,y=%8.6f,z=%9.6f\n",x,y,z);
 printf("x+y=%5.2fy+z=%5.2fz+x=%5.2f\n",x+y,y+z,z+x);
```

```
    printf("u=%6ldn=%9ld\n",u,n);
    printf("c1='%c' or %d(ASCII)\n",c1,c1);
    printf("c2='%c' or %d(ASCII)\n",c2,c2);
}
```

4.5 请写出下面程序的输出结果：

```
main()
{ int a=5,b=7;
  float x=67.8564,y=-789.124;
  char c='A';
  long n=1234567;
  unsigned u=65535;
  printf("%d%d\n",a,b);
  printf("%3d%3d\n",a,b);
  printf("%f,%f\n",x,y);
  printf("%-10f,%-10f\n",x,y);
  printf("%8.2f,%8.2f,%4f,%4f,%3f,%3f\n",x,y,x,y,x,y);
  printf("%e,%10.2e\n",x,y);
  printf("%c,%d,%o,%x\n",c,c,c,c);
  printf("%ld,%lo,%x\n",n,n,n);
  printf("%u,%o,%x,%d\n",u,u,u,u);
  printf("%s,%5.3s\n","COMPUTER","COMPUTER");
}
```

运行结果：

```
57
 5  7
67.856400,-789.124023
67.856400 ,-789.124023
 67.86, -789.12,67.856400,-789.124023,67.856400,-789.124023
6.785640e+01,-7.9e+02
A,65,101,41
1234567,4553207,d687
65535,177777,ffff,-1
COMPUTER, COM
```

4.6 用下面的 scanf 函数输入数据，使 a=3，b=7，x=8.5，y=71.82，c1='A'，c2='a'。问在键盘上如何输入？

```
main()
{ int a,b;
  float x,y;
  char c1,c2;
```

```
    scanf("a=%d␣b=%d",&a,&b);
    scanf("%f␣%e",&x,&y);
    scanf("␣%c␣%c",&c1,&c2);
    printf("a=%d,b=%d,x=%f,y=%f,c1=%c,c2=%c\n",a,b,x,y,c1,c2);
}
```

解：可按如下方式在键盘上输入：

```
a=3 b=7↙
8.5 71.82↙
A a↙
```

输出为：

```
a=3,b=7,x=8.500000,y=71.820000,c1=A,c2=a
```

请注意：在第三个 scanf 函数双引号中第一个字符为空格字符。如果没有这个空格字符，而写成：

```
scanf("%c%c",&c1,&c2);
```

按以上的输入，输出就会变成以下两行：

```
a=3,b=7,x=8.500000,y=71.820000,c1=
,c2=A
```

这是因为在输入完第二行数据后按的回车键被作为一个字符送到内存输入缓冲区中，因此第三个 scanf 函数中的第一个变量 c1 读入了回车符(实际上是回车符的 ASCII 码)。第三行输入的第一个字符 A 被 c2 读取，所以在执行 printf 函数输出 c1 时，就输出一个回车符，输出 c2 时就输出字符 A。我们在程序第三个 scanf 函数双引号中第一个字符处放了一个空格字符，这样第二行末尾输入的回车符就不会输入给 c1，而是与该空格字符对应，第三行输入的第一个字符 A 就被 c1 读取。也可以不在 scanf 函数中加空格，而在第三个函数前加一个 getchar 函数：getchar()；(注意要相应地在程序开头加：#include <stdio.h>)用它将前面的回车符"吃掉"。

在一个函数中如果有几个 scanf 函数，在输入数据时往往会出现一些想象不到的情况(例如前面碰到的情况)，其中一个重要的原因就是由回车符引起的。C 语言很灵活，书上不可能把一切细节都讲到，读者在遇到类似情况时，上机多试验一下就可以找出规律来。

4.7 用下面的 scanf 函数输入数据，使 a=10，h=20，c1='A'，c2='a'，x=1.5，y=-3.75，z=67.8，请问在键盘上如何输入数据？

```
scanf("%5d%5d%c%c%f%f%*f,%f",&a,&b,&c1,&c2,&x,&y,&z);
```

解：

```
main()
{
  int a,b;float x,y,z;
  char c1,c2;
  scanf("%5d%5d%c%c%f%f%*f,%f",&a,&b,&c1,&c2,&x,&y,&z);
```

```
    printf("a=%d, b=%d, c1=%c, c2=%c, x=%6.2f, y=6.2f, z=6.2f\n",a,b,c1,c2,
    x,y,z);
}
```

运行情况如下：

```
␣␣␣10␣␣␣20Aa1.5␣-3.75␣1.5,67.8↙          (此行为输入的数据)
a=10, b=20, c1=A, c2=a, x= 1.50, y= -3.75, z= 67.80  (此行为输出)
```

说明：按%5d格式的要求输入a与b时，要先键入三个空格，然后再键入10与20。%*f是用来禁止赋值的。在输入时，对应于%*f的地方，随意打入了一个数1.5，该值不会赋给任何变量。

4.8 设圆半径 $r=1.5$，圆柱高 $h=3$，求圆周长、圆面积、圆球表面积、圆球体积、圆柱体积。用scanf输入数据，输出计算结果，输出时要求有文字说明，取小数点后2位数字。请编程序。

解：

```
main()
{
 float pi,h,r,l,s,sq,vq,vz;
 pi=3.141526;
 printf("请输入圆半径 r,圆柱高 h: \n");
 scanf("%f,%f",&r,&h);
 l=2*pi*r;
 s=r*r*pi;
 sq=4*pi*r*r;
 vq=3.0/4.0*pi*r*r*r;
 vz=pi*r*r*h;
 printf("圆周长为:      l=%6.2f\n",l);
 printf("圆面积为:      s=%6.2f\n",s);
 printf("圆球表面积为:  sq=%6.2f\n",sq);
 printf("圆球体积为:    sv=%6.2f\n",vq);
 printf("圆柱体积为:    sz=%6.2f\n",vz);
}
```

运行结果：

```
请输入圆半径 r,圆柱高 h:
1.5, 3↙
圆周长为:      l= 9.42
圆面积为:      s= 7.07
圆球表面积为:  sq= 28.27
圆球体积为:    sv= 7.95
圆柱体积为:    sz= 21.21
```

4.9 输入一个华氏温度，要求输出摄氏温度。公式为

$$C=\frac{5}{9}(F-32)$$

输出要有文字说明，取 2 位小数。

解：

```
main()
{ float c,f;
  printf("请输入一个华氏温度:\n");
  scanf("%f",&f);
  c=(5.0/9.0)*(f-32);      /*注意 5 和 9 要用实型表示，否则 5/9 的值为 0*/
  printf("摄氏温度为:%5.2f\n",c);
}
```

运行结果：

```
请输入一个华氏温度:
78↙
摄氏温度为:25.56
```

4.10 编程序，用 getchar 函数读入两个字符给 c1、c2，然后分别用 putchar 函数和 printf 函数输出这两个字符。并思考以下问题：(1) 变量 c1、c2 应定义为字符型或整型？抑或二者皆可？(2) 要求输出 c1 和 c2 值的 ASCII 码，应如何处理？用 putchar 函数还是 printf 函数？(3) 整型变量与字符变量是否在任何情况下都可以互相代替？如 char "c1,c2;"与 "int c1,c2;"是否无条件地等价？

解：

```
#include <stdio.h>
main()
{char c1,c2;
 printf("请输入两个字符 c1,c2:\n");
 c1=getchar();
 c2=getchar();
 printf("用 putchar 语句输出结果为:\n");
 putchar(c1);
 putchar(c2);
 printf("\n");
 printf("用 printf 语句输出结果为:\n");
 printf("%c,%c\n",c1,c2);
}
```

运行结果：

```
请输入两个字符 c1,c2:
```

ab↙

用 putchar 语句输出结果为：

ab

用 printf 语句输出结果为：

a,b

请注意连续用两个 getchar 函数时怎样输入字符。不应当用以下方法输入：

a↙

b↙

因为第一行将 a 和回车符输入到内存的输入缓冲区，因此 c1 得到 a，c2 得到一个回车符。在输出 c2 时就会产生一个回车换行，而不会输出任何可显示的字符。在实际操作时，只要输入了"a↙"后，屏幕显示马上从用户屏切换到 TC 窗口，程序接着执行下去。因为系统认为用户已输入了两个字符，所以我们连续输入 ab 两个字符然后再按回车键，就保证了 c1 和 c2 分别得到字符 a 和 b。

回答思考问题：

(1) c1 和 c2 可以定义为字符型或整型，二者皆可。

(2) 在 printf 函数中用%d 格式符输出，即：

printf("%d,%d\n",c1,c2);

(3) 字符变量在计算机内占一个字节，而整型变量占两个字节，因此整型变量在可输出字符的范围内(ASCII 码为 0～255 之间的字符)是可以与字符数据互相转换的。如果整数在此范围外，则不能代替。请分析以下三个程序。

程序 1：

```
main()
{ int c1,c2;        /*  定义为整型  */
  printf("请输入两个整数 c1,c2:\n");
  scanf("%d,%d",&c1,&c2);
  printf("按字符输出结果为:\n");
  printf("%c,%c\n",c1,c2);
  printf("按 ASCII 码输出结果为:\n");
  printf("%d,%d\n",c1,c2);
}
```

运行结果：

```
请输入两个整数 c1,c2:
97,98↙
按字符输出结果为:
a,b
按 ASCII 码输出结果为:
97,98
```

程序 2：

```
main()
{
 char c1,c2;                    /*  定义为字符型  */
 int i1,i2;                     /*  定义为整型  */
 printf("请输入两个字符 c1,c2:\n");
 scanf("%c,%c",&c1,&c2);
 i1=c1;                         /*  将字符型赋值给整型变量 */
 i2=c2;
 printf("按字符输出结果为:\n");
 printf("%c,%c\n",i1,i2);   /*  将整型变量按字符输出  */
 printf("按整数输出结果为:\n");
 printf("%d,%d\n",c1,c2);   /*  将字符变量按整型输出  */
}
```

运行结果：

```
请输入两个字符 c1,c2:
a,b↙
按字符输出结果为:
a,b
按整数输出结果为:
97,98
```

程序 3：

```
main()
{ char c1,c2;                  /*  定义为字符型  */
  char i1,i2;                  /*  定义为整型  */
  printf("请输入两个整数 i1,i2:\n");
  scanf("%c,%c",&i1,&i2);
  c1=i1;                       /*  将整数赋给字符变量  */
  c2=i2;
  printf("按字符输出结果为:\n");
  printf("%c,%c\n",c1,c2);
  printf("按整数输出结果为:\n");
  printf("%d,%d\n",c1,c2);}
```

运行结果：

```
请输入两个整数 i1,i2:
289,330↙
按字符输出结果为:
```

!,J

按整数输出结果为：

33,74

请注意 c1 和 c2 是字符变量，只占一个字节，只能存放 0～255 范围内的整数，而现在输入给 i1 和 i2 的值已超过 0～255 的范围，所以只将 i1 和 i2 在内存中两个字节的低 8 位赋给 c1 和 c2。可以看到：289－255＝33，330－255＝74；而与 ASCII 码 33 和 74 相应的字符为“！”和“J”。请读者注意分析。

第5章　选择结构程序设计

5.1　什么是算术运算？什么是关系运算？什么是逻辑运算？

解：略。

5.2　C语言中如何表示“真”和“假”？系统如何判断一个量的“真”和“假”？

解：设有一个逻辑表达式，若其结果为“真”，则以1表示；若其结果为“假”，则以0表示。但是判断一个逻辑量的值时，以0代表“真”，以非0代表“假”。例如3 && 5的值为“真”，系统给出3 && 5的值为1。

5.3　写出下面各逻辑表达式的值。设$a=3,b=4,c=5$。

(1) a+b>c && b==c

(2) a||b+c && b-c

(3) !(a>b) && !c||1

(4) !(x=a) && (y=b) && 0

(5) !(a+b)+c-1 && b+c/2

解：

(1) 0

(2) 1

(3) 1

(4) 0

(5) 1

5.4　有3个整数a、b、c，由键盘输入，输出其中最大的数。

解：方法一：N-S图见图5.1。

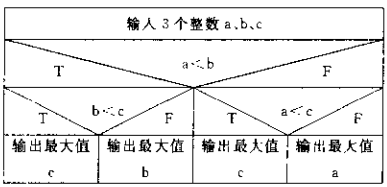


图　5.1

程序如下：

```
main()
{ int a,b,c;
  printf("请输入3个整数:");
```

```
    scanf("%d,%d,%d",&a,&b,&c);
    if (a<b)
      if (b<c)
        printf("max=%d\n",c);
      else
        printf("max=%d\n",b);
    else if (a<c)
          printf("max=%d\n",c);
        else
          printf("max=%d\n",a);
}
```

运行结果：

```
请输入 3 个整数：12，34，9↙
max=34
```

方法二：使用条件表达式，可以使程序更加简明、清晰。

```
main()
{ int a,b,c,temp,max ;
  printf("请输入 3 个整数:");
  scanf("%d,%d,%d",&a,&b,&c);
  temp=(a>b)? a:b;          /*  将 a 和 b 中的大者存入 temp 中   */
  max=(temp>c)? temp:c;     /*  将 a 和 b 中的大者与 c 比较,取最大者   */
  printf("3 个整数的最大数是 %d\n",max);
}
```

运行结果：

```
请输入 3 个整数：2，34，9↙
3 个整数的最大数是 34。
```

5.5 有一函数：

$$y=\begin{cases} x & (x<1) \\ 2x-1 & (1\leqslant x<10) \\ 3x-11 & (x\geqslant 10) \end{cases}$$

写一程序，输入 x 值，输出 y 值。

解：

```
main()
{ int x,y;
  printf("输入 x:");
  scanf("%d",&x);
  if (x<1)                    /* x<1 */
  { y=x;
    printf("x=%3d,      y=x=%d\n",x,y);
  }
```

```
    else if (x<10)              /* 1≤x<10 */
          { y=2*x-1;
            printf("x=%3d,      y=2*x-1=%d\n",x,y);
          }
       else                     /* x≥10 */
          {y=3*x-11;
           printf("x=%3d,      y=3*x-11=%d\n",x,y);
          }
    }
```

运行结果：

① 输入 x:4↙

x= 4, y=2*x-1=7

② 输入 x:-1↙

x= -1, y=x=-1

③ 输入 x:20↙

x= 20, y=3*x-11=49

5.6 给一百分制成绩，要求输出成绩等级'A'、'B'、'C'、'D'、'E'。90 分以上为'A'，80～89 分为'B'，70～79 分为'C'，60～69 分为'D'，60 分以下为'E'。

解： N-S 图见图 5.2。

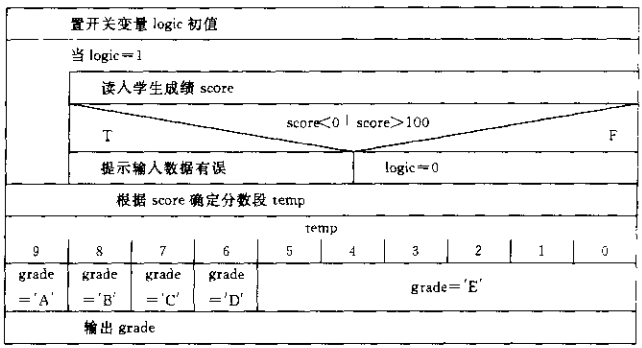


图 5.2

```
main()
{ float score;
  char grade;
  printf("请输入学生成绩:");
  scanf("%f",&score);
  while (score>100||score<0)
     { printf("\n 输入有误,请重输.");
       scanf("%f",&score);
```

```
    }
  switch((int)(score/10))
       {case 10:
        case 9:grade='A';break;
        case 8:grade='B';break;
        case 7:grade='C';break;
        case 6:grade='D';break;
        case 5:
        case 4:
        case 3:
        case 2:
        case 1:
        case 0:grade='E';
       }
    printf("成绩是 %5.1f,相应的等级是 %c。\n",score,grade);
  }
```

运行结果：

① 请输入学生成绩：90.5↙
成绩是 90.5,相应的等级是 A。
② 请输入学生成绩：59↙
成绩是 59.0,相应的等级是 E。

说明：对输入的数据进行检查，如小于 0 或大于 100，要求重新输入。(int)(score/10) 的作用是将(score/10) 的值进行强制类型转换，得到一个整型值。

5.7 给定一个不多于 5 位的正整数，要求：① 求它是几位数；② 分别打印出每一位数字；③ 按逆序打印出各位数字。例如原数为 321，应输出 123。

解：

```
main()
{
 long int num;
 int indiv,ten,hundred,thousand,ten_thousand,place;
                    /*分别代表个位、十位、百位、千位、万位和位数*/
 printf("请输入一个整数(0~99999):");
 scanf("%ld",&num);
 if (num>9999)
     place=5;
 else if (num>999)
     place=4;
 else if (num>99)
     place=3;
 else if (num>9)
     place=2;
```

```
    else place=1;
    printf("place=%d\n",place);
    printf("每位数字为:");
    ten_thousand=num/10000;
    thousand=(int)(num-ten_thousand*10000)/1000;
    hundred=(int)(num-ten_thousand*10000-thousand*1000)/100;
    ten=(int)(num-ten  thousand*10000-thousand*1000-hundred*100)/10;
    indiv=(int)(num-ten_thousand*10000-thousand*1000-hundred*100-ten*10);
    switch(place)
      {case 5:printf("%d,%d,%d,%d,%d",ten_thousand,thousand,hundred,ten,indiv);
        printf("\n反序数字为:");
        printf("%d%d%d%d%d\n",indiv,ten,hundred,thousand,ten_thousand);
        break;
       case 4:printf("%d,%d,%d,%d",thousand,hundred,ten,indiv);
         printf("\n反序数字为:");
         printf("%d%d%d%d\n",indiv,ten,hundred,thousand);
         break;
       case 3:printf("%d,%d,%d",hundred,ten,indiv);
         printf("\n反序数字为:");
         printf("%d%d%d\n",indiv,ten,hundred);
         break;
       case 2:printf("%d,%d",ten,indiv);
         printf("\n反序数字为:");
         printf("%d%d\n",indiv,ten);
         break;
       case 1:printf("%d",indiv);
         printf("\n反序数字为:");
         printf("%d\n",indiv);
         break;
      }
    }
```

运行结果:

```
请输入一个整数(0~99999):98765↙
位数:5
每位数字为:9,8,7,6,5
反序数字为:56789
```

5.8 企业发放的奖金根据利润提成。利润 I 低于或等于 10 万元时,奖金可提 10%;利润高于 10 万元,低于 20 万元($100000<I\leqslant200000$)时,其中 10 万元按 10%提成,高于 10 万元的部分,可提成 7.5%;$200000<I\leqslant400000$ 时,其中 20 万元仍按上述办法提成(下同),高于 20 万元的部分按 5%提成;$400000<I\leqslant600000$ 时,高于 40 万元的部分按 3%提成;$600000<I\leqslant1000000$ 时,高于 60 万的部分按 1.5%提成;$I>1000000$ 时,超过 100 万的部分按 1%提成。从键盘输入当月利润 I,求应发放

奖金总数。

要求：(1) 用 if 语句编程序；(2) 用 switch 语句编程序。

解：计算利润时，要特别注意不同利润的不同提成比例。例如，利润为 15 万元，其中有 10 万元按 10%的比例提成，另外 5 万元则按 7.5%提成。

(1) 用 if 语句编程序，N-S 图见图 5.3。

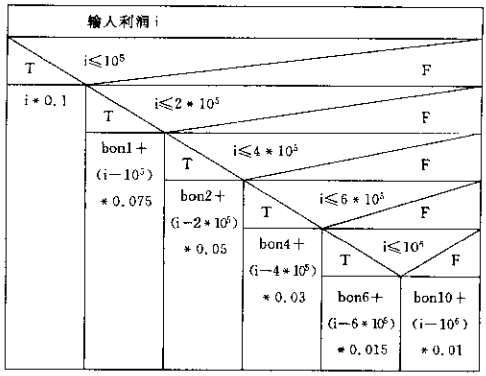


图 5.3

```
main()
{ long i;
  float bonus,bon1,bon2,bon4,bon6,bon10;
  bon1=100000*0.1;                    /*利润为 10 万元时的奖金*/
  bon2=bon1+100000*0.075;             /*利润为 20 万元时的奖金*/
  bon4=bon2+200000*0.05;              /*利润为 40 万元时的奖金*/
  bon6=bon4+200000*0.03;              /*利润为 60 万元时的奖金*/
  bon10=bon6+400000*0.015;            /*利润为 100 万元时的奖金*/
  printf("请输入利润 i:");
  scanf('%ld",&i);
  if (i<=100000)
    bonus=i*0.1;                      /*利润在 10 万元以内按 0.1 提成奖金*/
  else if (i<=200000)
    bonus=bon1+(i-100000)*0.075;      /*利润在 10 万元至 20 万元时的奖金*/
  else if (i<=400000)
    bonus=bon2+(i-200000)*0.05;       /*利润在 20 万元至 40 万元时的奖金*/
  else if (i<=600000)
    bonus=bon4+(i-400000)*0.03;       /*利润在 40 万元至 60 万元时的奖金*/
  else if (i<=1000000)
    bonus=bon6+(i-600000)*0.015;      /*利润在 60 万元至 100 万元时的奖金*/
  else
    bonus=bon10+(i-1000000)*0.01;     /*利润在 100 万元以上时的奖金*/
```

```
    printf("奖金是 %10.2f",bonus);
}
```

运行结果：

```
请输入利润 i:234000↙
奖金是:19200.00
```

此题的关键在于正确写出每一区间的奖金计算公式。例如利润在 10 万元至 20 万时，奖金应由两部分组成：① 利润为 10 万元时应得的奖金，即 100000×0.1；② 10 万元以上部分应得的奖金，即(num－100000)×0.075。同理，20 万～40 万这个区间的奖金也应由两部分组成：① 利润为 20 万元时应得的奖金，即 100000×0.1＋10 万×0.075；② 20 万元以上部分应得的奖金，即(num－200000)×0.05。

程序中先把 10 万、20 万、40 万、60 万、100 万各关键点的奖金计算出来，即 bon1、bon2、bon4、bon6、bon10；然后再加上各区间附加部分的奖金。

(2) 用 switch 语句编程序，N-S 图如图 5.4 所示。

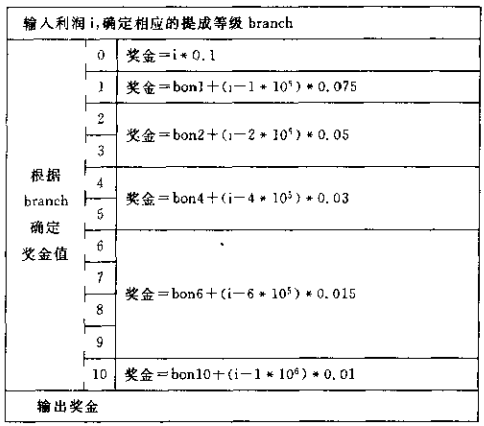


图 5.4

```
main()
{ long i;
  float bonus,bon1,bon2,bon4,bon6,bon10;
  int c;
  bon1=100000*0.1;
  bon2=bon1+100000*0.075;
  bon4=bon2+200000*0.05;
  bon6=bon4+200000*0.03;
  bon10=bon6+400000*0.015;
  printf("请输入利润 i :");
```

```
    scanf("%ld",&i);
    c=i/100000;
    if (c>10) then c=10;
    switch(c)
      {case 0: bonus=I*0.1; break;
       case 1: bonus=bon1+(i-100000)*0.075; hreak;
       case 2:
       case 3: bonus=bon2+(i-200000)*0.05;break;
       case 4:
       case 5: bonus=bon4+(i-400000)*0.03;break;
       case 6:
       case 7:
       case 8:
       case 9: bonus=bon6+(i-600000)*0.015; break;
       case 10: bonus=bon10+(i-1000000)*0.01;
       }
    printf("奖金是%10.2f",bonus);
}
```

运行结果：

```
请输入利润：234000↙
奖金是 19200.00
```

5.9 输入 4 个整数，要求按由大到小的顺序输出。

解：此题采用依次比较的方法排出其大小顺序。在学习了循环和数组以后，可以有更多的排序方法。

```
main()
{ int t,a,b,c,d;
  printf("请输入 4 个整数:");
  scanf("%d,%d,%d,%d",&a,&b,&c,&d);
  printf("\n a=%d, b=%d, c=%d, d=%d \n",a,b,c,d);
  if (a>b)
    {t=a; a=b; b=t;}
  if (a>c)
    {t=a; a=c; c=t;}
  if (a>d)
    {t=a; a=d; d=t;}
  if (b>c)
    {t=b; b=c; c=t;}
  if (b>d)
    {t=b; b=d; d=t;}
```

```
    if (c>d)
      {t=c; c=d; d=t;}
    printf("排序结果如下: \n");
    printf("%d %d %d %d\n",a,b,c,d);
}
```

运行情况:

```
请输入4个整数: 6,8,1,4 ↙
a=6,b=8,c=1,d=4
排序结果如下:
1 4 6 8
```

5.10 有4个圆塔,圆心分别为(2,2)、(-2,2)、(2,-2)、(-2,-2),圆半径为1。见图5.5。这4个塔的高度分别为10m。塔以外无建筑物。今输入任一点的坐标,求该点的建筑高度(塔外的高度为零)。

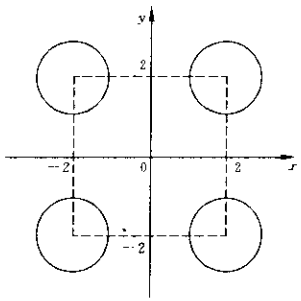


图 5.5

解: N-S图见图5.6。

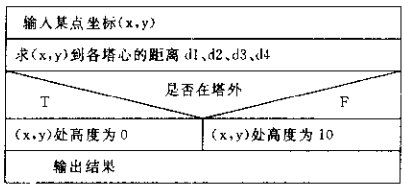


图 5.6

程序如下:

```
main()
{ int h=10;
  float x1=2,y1=2,x2=-2,y2=2,x3=-2,y3=-2,x4=2,y4=-2,x,y,d1,d2,d3,d;
```

```
    printf("请输入一个点(x,y):");
    scanf("%f,%f",&x,&y);
    d1=(x-x1)*(x-x1)+(y-y1)*(y-y1);          /*求该点到各中心点距离*/
    d2=(x-x2)*(x-x2)+(y+y2)*(y+y2);
    d3=(x+x3)*(x+x3)+(y-y3)*(y-y3);
    d4=(x+x4)*(x+x4)+(y+y4)*(y+y4);
    if(d1>1 && d2>1 && d3>1 && d4>1) h=0;  /*判断该点是否在塔外*/
    printf("该点高度为%d",h);
}
```

运行情况：

① 请输入一个点(x,y)：0.5,0.7↙
该点高度为 0

② 请输入一个点(x,y)：2.1,2.3↙
该点高度为 10

第6章 循环控制

6.1 输入两个正整数 m 和 n，求其最大公约数和最小公倍数。

解：

```
main()
{int p,r,n,m,temp;
 printf("请输入两个正整数 n,m:");
 scanf("%d,%d",&n,&m);
 if (n<m)
  { temp=n;
    n=m;
    m=temp;      /*把大数放在 n 中，小数放在 m 中*/
  }
 p=n*m;          /*先将 n 和 m 的乘积保存在 p 中，以便求最小公倍数时用*
 while (m!=0)    /*求 n 和 m 的最大公约数*/
  { r=n%m;
    n=m;
    m=r;
  }
 printf("它们的最大公约数为:%d\n",n);
 printf("它们的最小公倍数为:%d\n",p/n);    /*p 是原来两个整数的乘积*
}
```

运行情况：

```
请输入两个正整数：12,8↙
它们的最大公约数为:4
它们的最小公倍数为:24
```

6.2 输入一行字符，分别统计出其中英文字母、空格、数字和其他字符的个数。

解：

```
#include <stdio.h>
main()
{char c;
 int letter=0,space=0,digit=0,other=0;
 printf("请输入一行字符:\n");
 while((c=getchar())!='\n')
   {if (c>='a'&&c<='z'||c>='A'&&c<='Z')
      letter++;
```

```
        else if (c==' ')
            space++;
        else if (c>='0' && c<='9')
            digit++;
        else
            other++;
    }
    printf("字母数=%d,空格数=%d,数字数=%d,其他字符数=%d\n",letter,space,digit,other);
}
```

运行情况：

```
请输入一行字符：
My teacher's address is "#123 Beijing Road,Shanghai".
字母数:38,空格数:6,数字数:3,其他字符数:6
```

6.3 求 $S_n=a+aa+aaa+\cdots+\overbrace{aa\cdots a}^{n个}$之值，其中 a 是一个数字。例如：2+22+222+2222+22222(此时 $n=5$)，n 由键盘输入。

解：

```
main()
{int a,n,i=1,sn=0,tn=0;
 printf("a,n=:");
 scanf("%d,%d",&a,&n);
 while (i<=n)
   { tn=tn+a;          /*赋值后的 tn 为 i 个 a 组成数的值*/
     sn=sn+tn;         /*赋值后的 sn 为多项式前 i 项之和*/
     a=a*10;
     ++i;
   }
 printf("a+aa+aaa+…=%d\n",sn);
}
```

运行情况：

```
a,n: 2,5↙
a+aa+aaa+…=24690
```

6.4 求 $\sum_{n=1}^{20} n!$(即求 1+2! +…+20!)。

解：

```
main()
{float s=0,t=1;
 int n;
 for (n=1;n<=20;n++)
```

```
    }
    t=t*n;          /* 求n! */
    s=s+t;          /* 将各项累加 */
  }
  printf("1!+2!+…+20!=%e\n",s);
}
```

运行结果：

```
1!+2!+…+20!=2.56133e+18
```

请注意：s不能定义为int型，因为int型数据的范围为－32 768～32 767；也不能定义为long型，因为long型数据的范围为－21亿～21亿，无法容纳求得的结果。

6.5 求 $\sum_{k=1}^{100} k+\sum_{k=1}^{50} k^2+\sum_{k=1}^{10}\frac{1}{k}$。

解：

```
main()
{int n1=100,n2=50,n3=10;
 float k;
 float s1=0,s2=0,s3=0;
 for (k=1;k<=n1;k++) /*计算1到100的和*/
 s1=s1+k;
 for (k=1;k<=n2;k++) /*计算1到50各数的平方和*/
 s2=s2+k*k;
 for (k=1;k<=n3;k++) /*计算1到10各数的倒数和*/
 s3=s3+1/k;
 printf("总和=%8.2f\n",s1+s2+s3);
}
```

运行结果：

```
总和=47977.93
```

6.6 打印出所有的“水仙花数”。所谓“水仙花数”是指一个3位数，其各位数字的立方和等于该数本身。例如，153是一个“水仙花数”，因为 $153=1^3+5^3+3^3$。

解：

```
main()
{int i,j,k,n;
 printf("“水仙花”数是:");
 for (n=100;n<1000;n++)
   {i=n/100;
    j=n/10-i*10;
    k=n%10;
```

```
    if (n == i*i*i + j*j*j + k*k*k)
      printf("%4d",n);
   }
  printf("\n");
}
```

运行结果：

"水仙花"数是：153 370 371 407

6.7 一个数如果恰好等于它的因子之和，这个数就称为"完数"。例如，6 的因子为 1、2、3，而 6=1+2+3，因此 6 是"完数"。编程序找出 1000 以内的所有"完数"，并按下面格式输出其因子：

6 Its factors are 1,2,3

解：方法一：

```
# define M 1000                    /* 定义寻找范围 */
main()
{int k1,k2,k3,k4,k5,k6,k7,k8,k9,k10;
 int i,a,n,s;
 for (a=2;a<=M;a++)               /* a是2~1000之间的整数,检查它是否完数 */
 {n=0;                            /* n用来累计a的因子的个数 */
  s=a;                            /* s用来存放尚未求出的因子之和,开始时等于a */
  for (i=1;i<a;i++)               /* 检查i是否a的因子 */
   if (a%i==0)                    /* 如果i是a的因子 */
    {n++;                         /* n加1,表示新找到一个因子 */
     s=s-i;                       /* s减去已找到的因子,s的新值是尚未求出的因子之和 */
     switch(n)                    /* 将找到的因子赋给k1,…,k10 */
      {case 1:
           k1=i; break;           /* 找出的第1个因子赋给k1 */
       case 2:
           k2=i; break;           /* 找出的第2个因子赋给k2 */
       case 3:
           k3=i; break;           /* 找出的第3个因子赋给k3 */
       case 4:
           k4=i; break;           /* 找出的第4个因子赋给k4 */
       case 5:
           k5=i; break;           /* 找出的第5个因子赋给k5 */
       case 6:
           k6=i; break;           /* 找出的第6个因子赋给k6 */
       case 7:
           k7=i; break;           /* 找出的第7个因子赋给k7 */
       case 8:
           k8=i; break;           /* 找出的第8个因子赋给k8 */
```

```
            case 9:
                k9=i; break;        /* 找出的第 9 个因子赋给 k9 */
            case 10:
                k10=i; break;       /* 找出的第 10 个因子赋给 k10 */
          }
      }
    if (s==0)                       /* s=0 表示全部因子都已找到 */
      {printf("%d Its factors are",a);
       if (n>1)printf("%d,%d",k1,k2);    /* n>1 表示 a 至少有 2 个因子 */
       if (n>2)printf(",%d",k3);    /* n>2 表示至少有 3 个因子,故应再输出一个因子 */
       if (n>3)printf(",%d",k4);    /* 以下类似 */
       if (n>4)printf(",%d",k5);
       if (n>5)printf(",%d",k6);
       if (n>6)printf(",%d",k7);
       if (n>7)printf(",%d",k8);
       if (n>8)printf(",%d",k9);
       if (n>9)printf(",%d",k10);
       printf("\n");
      }
  }
}
```

运行结果：

```
6 Its factors are 1,2,3
28 Its factors are 1,2,4,7,14
496 Its factors are 1,2,4,8,16,31,62,124,248
```

方法二：

```
main()
{int m,s,i;
 for (m=2;m<1000;m++)
  {s=0;
   for (i=1;i<m;i++)
     if ((m%i)==0) s=s+i;
   if(s==m)
     {printf("%d 是一个"完数",它的因子是",m);
      for (i=1;i<m;i++)
        if (m%i==0) printf("%d ",i);
          printf("\n");
     }
  }
```

```
}
```

方法三：此题用数组方法更为简单。

```
main()
{ int k[11];
  int i,a,n,s;
  for (a=2;a<=1000;a++)
 {n=0;
  s=a;
  for (i=1;i<a;i++)
    if ((a%i)==0)
      {n++;
       s=s-i;
       k[n]=i;        /* 将找到的因子赋给 k[1],…,k[10] */
      }
    if (s==0)
      {printf("\n%d 是一个"完数",它的因子是:",a);
       for (i=1;i<n;i++)
          printf("%d,",k[i]);
       printf("%d\n",k[n]);
      }
 }
}
```

运行结果：

```
6 是一个"完数",它的因子是:1,2,3
28 是一个"完数",它的因子是:1,2,4,7,14
496 是一个"完数",它的因子是:1,2,4,8,16,31,62,124,248
```

6.8 有一分数序列：

$$\frac{2}{1},\frac{3}{2},\frac{5}{3},\frac{8}{5},\frac{13}{8},\frac{21}{13},\cdots$$

求出这个数列的前 20 项之和。

解：

```
main()
{int i,t,n=20;
 float a=2,b=1,s=0;
 for (i=1;i<=n;i++)
    {s=s+a/b;
     t=a;
     a=a+b;        /* 将前一项分子与分母之和作为下一项的分子 */
     b=t;          /* 将前一项的分子作为下一项的分母 */
```

```
    }
    printf("sum=%9.6f\n",s);
}
```

运行结果：

```
sum=32.660259
```

6.9 一球从100m高度自由落下，每次落地后反跳回原高度的一半，再落下。求它在第10次落地时，共经过多少m？第10次反弹多高？

解：

```
main()
{float sn=100.0,hn=sn/2;
 int n;
 for(n=2;n<=10;n++)
  {sn=sn+2*hn;          /*第n次落地时共经过的米数*/
   hn=hn/2;             /*第n次反跳的高度*/
  }
 printf("第10次落地时共经过%f m.\n",sn);
 printf("第10次反弹%f m.\n",hn);
}
```

运行结果：

```
第10次落地时共经过299.609375m.
第10次反弹0.097656m.
```

6.10 猴子吃桃问题。猴子第一天摘下若干个桃子，当即吃了一半，还不过瘾，又多吃了一个。第二天早上又将剩下的桃子吃掉一半，又多吃了一个。以后每天早上都吃了前一天剩下的一半零一个。到第10天早上想再吃时，见只剩一个桃子了。求第一天共摘多少桃子。

解：

```
main()
{int day,x1,x2;
 day=9;
 x2=1;
 while(day>0)
  {x1=(x2+1)*2;          /*第1天的桃子数是第2天桃子数加1后的2倍*/
   x2=x1;
   day--;
  }
 printf("total=%d\n",x1);
}
```

运行结果：

```
total=1534
```

6.11 用迭代法求 $x=\sqrt{a}$. 求平方根的迭代公式为

$$x_{n+1}=\frac{1}{2}(x_n)+\frac{a}{x_n}$$

要求前后两次求出的 x 的差的绝对值小于 10^{-5}。

解：用迭代法求平方根的算法如下：

(1) 设定一个 x 的初值 x_0；

(2) 用上述公式求出 x 的下一个值 x_1；

(3) 再将 x_1 代入上述公式，求出 x 的下一个值 x_2；

(4) 如此继续下去，直到前后两次求出的 x 值(x_{n+1}和 x_n)满足以下关系：

$$|x_{n+1}-x_n|<10^{-5}$$

为了便于程序处理，今只用变量 x_0 和 x_1，先令 x 的初值 $x_0=a/2$(也可以是另外的值)，求出 x_1；如果此时 $|x_1-x_0|\geqslant 10^{-5}$，则使 $x_1\Rightarrow x_0$，然后用这个新的 x_0 求出下一个 x_1；如此反复，直到 $|x_1-x_0|<10^{-5}$ 为止。

程序如下：

```
#include <math.h>
main()
{float a,x0,x1;
 printf("Enter a positive number:");
 scanf("%f",&a);                    /* 输入 a 的值 */
 x0=a/2;
 x1=(x0+a/x0)/2;
 do
   {x0=x1;
    x1=(x0+a/x0)/2;
   }
   while(fabs(x0-x1)>=1e-5);
 printf("The square root of %5.2f is %8.5f\n",a,x1);
}
```

运行结果：

```
Enter a positive number: 2↵
The square root of 2.00 is 1.41421
```

6.12 用牛顿迭代法求下面方程在 1.5 附近的根。

$$2x^3-4x^2+3x-6=0$$

解：牛顿迭代法又称牛顿切线法。它采用以下的方法求根：先任意设定一个与真实的根接近的值 x_0 作为第一次近似根，由 x_0 求出 $f(x_0)$，过$(x_0,f(x_0))$点做$f(x)$的切线，交 x 轴于 x_1，把它作为第二次近似根；再由 x_1 求出 $f(x_1)$，过$(x_1,f(x_1))$

点做 $f(x)$ 的切线，交 x 轴于 x_2，求出 $f(x_2)$；再作切线……如此继续下去，直到足够接近真正的根 x^* 为止，见图 6.1。

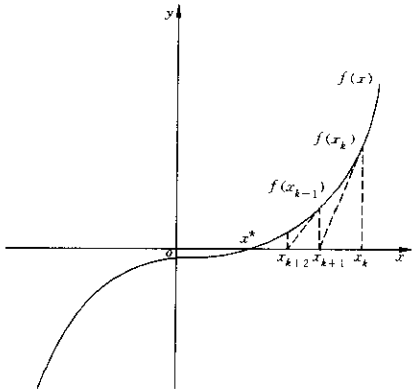


图 6.1

从图 6.1 可以看出：

$$f'(x_0) = f(x_0)/(x_1 - x_0)$$

因此：

$$x_1 = x_0 - f(x_0)/f'(x_0)$$

这就是牛顿迭代公式。可以利用它由 x_0 求出 x_1，然后再由 x_1 求出 x_2……

设 $f(x) = 2x^3 - 4x^2 + 3x - 6$

可以写成以下形式：

$$f(x) = [(2x-4)x+3]x-6$$

同样，$f'(x)$ 可写成：

$$f'(x) = 6x^2 - 8x + 3 = (6x-8)x+3$$

用这种方法表示的表达式，在运算时可节省时间。例如求 $f(x)$ 只需要进行 3 次乘法和 3 次加法，而原来的表达式要经过多次指数运算、对数运算和乘法、加法运算，花费时间较多。现在由于计算机的运算速度愈来愈快，这点时间开销是微不足道的，这是以前计算机的运算速度较慢时所提出的问题。由于过去编写的程序往往采用这种形式，所以我们在此也顺便介绍一下，以便在阅读别人所写的程序时知道其所以然。

程序如下：

```
#include <math.h>
main()
{float x,x0,f,f1;
 x=1.5;
 do
   {x0=x;
```

```
    f=((2*x0-4)*x0+3)*x0-6;
    f1=(6*x0-8)*x0+3;
    x=x0-f/f1;
    }
  while(fabs(x-x0)>=1e-5);
  printf("The root of equation is %5.2f\n");
}
```

运行结果：

```
The root of equation is 2.00
```

为了便于循环处理，程序中只设了 x0 和 x，x0 代表前一次的近似根，x 代表后一次的近似根。求出一个 x 后，把它的值赋给 x0，然后用它求下一个 x。由于第一次执行循环体时，需要对 x0 赋值，故在开始时应先对 x 赋一个初值（今为1.5，也可以是接近真实根的其他值）。

6.13 用二分法求下面方程在(−10,10)之间的根。

$$2x^3-4x^2+3x-6=0$$

解：

二分法的思路如下：先指定一个区间$[x_1,x_2]$，如果函数 $f(x)$在此区间是单调变化的，则可以根据 $f(x_1)$和 $f(x_2)$是否同号来确定方程 $f(x)=0$ 在区间$[x_1,x_2]$内是否有一个实根。若 $f(x_1)$和 $f(x_2)$不同号，则 $f(x)=0$ 在区间$[x_1,x_2]$内必有一个(且只有一个) 实根；如果 $f(x_1)$和 $f(x_2)$同号，则 $f(x)$在区间$[x_1,x_2]$内无实根，要重新改变 x_1 和 x_2 的值。当确定 $f(x)$在$[x_1,x_2]$内有一个实根后，可采取二分法将$[x_1,x_2]$一分为二，再判断在哪一个小区间中有实根。如此不断进行下去，直到小区间足够小为止，见图 6.2。

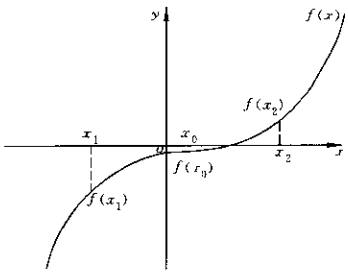


图 6.2

具体算法如下：

(1) 输入 x_1 和 x_2 的值。

(2) 求 $f(x_1)$和 $f(x_2)$。

(3) 如果 $f(x_1)$和 $f(x_2)$同号说明在$[x_1,x_2]$内无实根，返回步骤(1)，重新输入 x_1 和 x_2 的值；若 $f(x_1)$和 $f(x_2)$不同号，则在$[x_1,x_2]$必有一个 实根，执行步骤(4)。

(4) 求 x_1 和 x_0 的中点：$x_0=(x_1+x_2)/2$。

(5) 求 $f(x_0)$。

(6) 判断 $f(x_0)$与 $f(x_1)$是否同号。

① 如果同号，则应在$[x_0,x_2]$中寻找根，此时 x_1 已不起作用，用 x_0 代替 x_1，用 $f(x_0)$代替 $f(x_1)$。

② 如果 $f(x_0)$与 $f(x_1)$不同号，则说明应在$[x_1,x_0]$中寻找根，此时 x_2 已不起作用，用 x_0 代替 x_2，用 $f(x_0)$代替 $f(x_2)$。

(7) 判断 $f(x_0)$的绝对值是否小于某一个指定的值(例如 10^{-5})。若不小于 10^{-5}，则返回步骤(4) 重复执行步骤(4)、(5)、(6)；否则执行步骤(8)。

(8) 输出 x_0 的值，它就是所求出的近似根。

N S图见图 6.3。

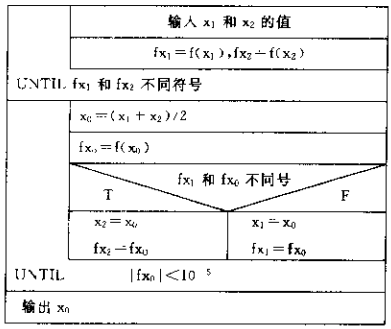


图 6.3

程序如下：

```
#include <math.h>
main()
{float x0,x1,x2,fx0,fx1,fx2;
 do
   {printf("Enter x1 & x2:");
    scanf("%f,%f",&x1,&x2);
    fx1=x1*((2*x1-4)*x1+3)-6;
    fx2=x2*((2*x2-4)*x2+3)-6;
   }
    while(fx1*fx2>0);
 do
   {x0=(x1-x2)/2;
    fx0=x0*((2*x0-4)*x0+3)-6;
    if ((fx0*fx1)<0)
      {x2=x0;
       fx2=fx0;
      }
    else
```

```
            {x1=x0;
             fx1=fx0;
            }
        }
         while(fabs (fx0)>=1e-5);
        printf("x=%6.2f\n",x0);
    }
```

运行结果：

```
Enter x1 & x2: -10,10↙
x= 2.00
```

6.14 打印出以下图案。

```
    *
   * * *
  * * * * *
 * * * * * * *
  * * * * *
   * * *
    *
```

解：

```
main()
{int i,j,k;
 for (i=0;i<=3;i++)                    /* 输出上面4行*号 */
   {for (j=0;j<=2-i;j++)
      printf(" ");                     /* 输出*号前面的空格 */
    for (k=0;k<=2*i;k++)
      printf("*");                     /* 输出*号 */
    printf("\n");                      /* 输出完一行*号后换行 */
   }
 for (i=0;i<=2;i++)                    /* 输出下面3行*号 */
   {for (j=0;j<=i;j++)
      printf(" ");                     /* 输出*号前面的空格 */
    for (k=0;k<=4-2*i;k++)
      printf("*");                     /* 输出*号 */
    printf("\n");                      /* 输出完一行*号后换行 */
   }
}
```

运行结果：

```
    *
   * * *
  * * * * *
 * * * * * * *
  * * * * *
   * * *
    *
```

6.15 两个乒乓球队进行比赛，各出三人。甲队为 A、B、C 三人，乙队为 X、Y、Z 三人。

已抽签决定比赛名单。有人向队员打听比赛的名单，A 说他不和 X 比，C 说他不和 X、Z 比。请编程序找出三对赛手的名单。

解：

先分析题目。按题意，画出图 6.4 的示意图。

图中带"×"符号的虚线表示不允许的组合。从图中可以看到：① X 既不与 A 比赛，又不与 C 比赛，必然与 B 比赛；② C 既不与 X 比赛，又不与 Z 比赛，必然与 Y 比赛；③ A 只能与 Z 比赛，见图 6.5。

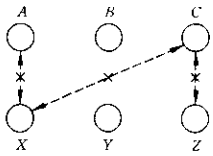


图 6.4

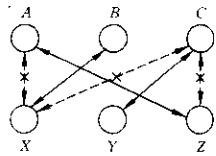


图 6.5

以上是经过逻辑推理得到的结论。用计算机程序处理此问题时，不可能立即就得出此结论，而必须对所有组合一一检验，看它们是否符合条件。

开始设计程序时，并不知道 A、B、C 与 X、Y、Z 中的哪一个进行比赛，可以假设：A 与 i 比赛，B 与 j 比赛，C 与 k 比赛；即：

$$A-i$$

$$B-j$$

$$C-k$$

i、j、k 分别是 X、Y、Z 之一，且 i、j、k 互不相等（一个队员不能与对方的两个队员比赛）。N-S 图见图 6.6。

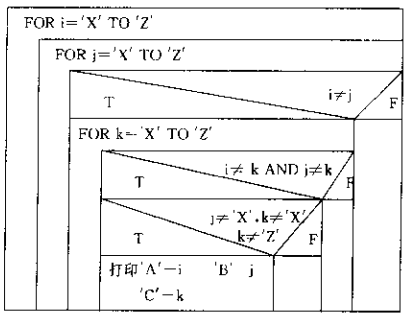


图 6.6

其中，外循环使 i 由'X'变到'Z'，中循环使 j 由'X'变到'Z'（但 i 不应与 j 相等）。然后，对每一组 i、j 的值，找符合条件的 k 值。k 同样也可能是'X'、'Y'、'Z'之一，但

k 也不应与 i 或 j 相等。在 i≠j≠k 的条件下，把 i≠'X' 和 k≠'X' 以及 k'Z'的 i、j、k 的值输出即可。

```
main()
{ char i,j,k;        /* i是a的对手;j是b的对手;k是c的对手*/
  for (i='X';i<='Z';i++)
    for (j='X';j<='Z';j++)
      if (i! =j)
        for (k='X';k<='Z';k++)
          if (i! =k && j! =k)
            if (i! ='X' && k! ='X' && k! ='Z')
              printf("A--%c\tB--%c\tC--%c\n",i,j,k);
}
```

运行结果：

```
A--Z    B--X    C--Y
```

说明：

(1) 整个执行部分只有一个语句，所以只在语句的最后有一个分号。请读者弄清楚循环和选择结构的嵌套关系。

(2) 分析最下面一个 if 语句中的条件：i≠'X'，k≠'X'，k≠'Z'，因为我们已事先假定 $A-i, B-j, C-k$，由于题目规定 A 不与 X 对抗，因此 i 不能等于'x'；同理，C 不与 X、Z 对抗，因此 k 不应等于'X'和'Z'。

(3) 题目给的是 A、B、C、X、Y、Z，而程序中用了加撇号的字符常量'X'、'Y'、'Z'，这是为什么？这是为了在运行时能直接打印出字符'A'、'B'、'C'、'X'、'Y'、'Z'，以表示三组对抗的情况。

第7章 数 组

7.1 用筛法求 100 之内的素数。

解：用筛法求素数的算法已在教材第 2 章例 2.22 中介绍了。根据该算法可以画出 N-S 图，见图 7.1。

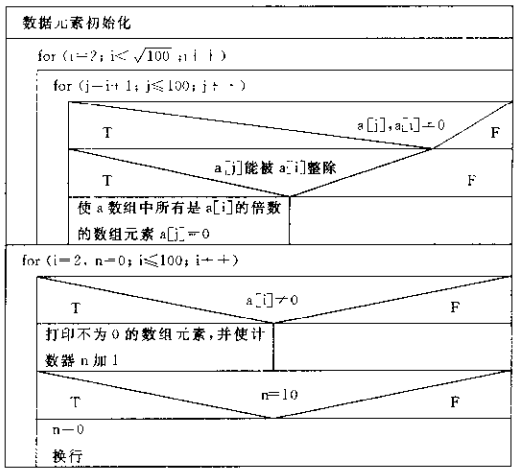


图 7.1

程序如下：

```
#include <math.h>
main()
{int i,j,n,a[101];
 for (i=1;i<=100;i++)
   a[i]=i;
 for (i=2;i<=sqrt(100);i++)
   for (j=i+1;j<=100;j++)
     if(a[i]!=0 && a[j]!=0)
       if (a[j]%a[i]==0)
         a[j]=0;}
 printf("\n");
 for (i=2,n=0;i<=100;i++)
```

```
    {if (a[i]! =0)
       {printf("%5d",a[i]);
        n++;}
     if(n==10)
       {printf("\n");
        n=0;}
    }
}
```

运行结果：

```
  2    3    5    7   11   13   17   19   23   29
 31   37   41   43   47   53   59   61   67   71
 73   79   83   89   97
```

7.2 用选择法对 10 个整数排序(从小到大)。

解：选择排序的思路如下：

设有 10 个元素 a[1]～a[10]，将 a[1]与 a[2]～a[10] 比较，若 a[1]比 a[2] ～a[10] 都小，则不进行交换，即无任何操作。若 a[2]～a[10]中有一个以上比 a[1]小，则将其中最大的一个(假设为 a[i]) 与 a[1] 交换，此时 a[1]中存放了 10 个中最小的数。第二轮将 a[2]与 a[3]～a[10]比较，将剩下 9 个数中的最小者 a[i] 与a[2]对换，此时 a[2]中存放的是 10 个中第 2 小的数。依此类推，共进行 9 轮比较，a[1]到 a[10]就已按由小到大顺序存放。N-S 图如图 7.2。

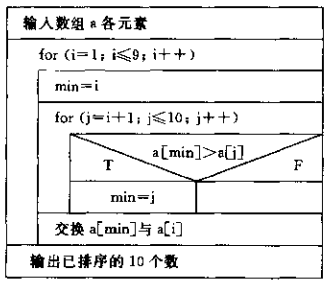


图 7.2

程序如下：

```
main()
{int i,j,min,temp,a[11];
 printf("Enter data:\n");
 for (i=1;i<=10;i++)
   {printf("a[%d]=",i);
    scanf("%d",&a[i]);        /* 输入 10 个数 */
```

```
    }
  printf("\n");
  for (i=1;i<=10;i++)
    printf("%5d",a[i]);          /* 输出这 10 个数 */
  printf("\n");
  for (i=1;i<=9;i++)             /* 以下 8 行是对 10 个数排序 */
    {min=i;
     for (j=i+1;j<=10;j++)
       if (a[min]>a[j]) min=j;
     temp=a[i];                  /* 以下 3 行将 a[i+1]~a[10]中最小者与 a_i_对换
     a[i]=a[min];
     a[min]=temp;
    }
  printf("\nThe sorted numbers:\n");
  for (i=1;i<=10;i++)            /* 输出已排好序的 10 个数 */
    printf("%5d",a[i]);
}
```

运行结果：

```
Enter data:
a[1]=6↙
a[2]=90↙
a[3]=45↙
a[4]=56↙
a[5]=1↙
a[6]=15↙
a[7]=44↙
a[8]=78↙
a[9]=58↙
a[10]=101↙
   6   90   45   56    1   15   44   78   58  101
The sorted numbers:
   1    6   15   44   45   56   58   78   90  101
```

说明：定义 a 数组有 11 个元素：a[0]~a[10]，但实际上只对 a[1]~a[10] 这 10 个元素输入值并排序。a[0] 未用到，这样符合人们的习惯。

7.3 求一个 3×3 矩阵对角线元素之和。

解：

```
main()
{int a[3][3],sum=0;
 int i,j;
 printf("Enter data:\n");
```

```
 for (i=0;i<3;i++)
   for (j=0;j<3;j++)
     scanf("%d",&a[i][j]);
 for (i=0;i<3;i++)
   sum=sum+a[i][i];
 printf("sum =%5d\n",sum);
}
```

运行情况如下：

```
Enter data:
1  2  3  4  5  6  7  8  9↙
sum = 15
```

此程序中用的是整型数组，运行结果是正确的。如果用是实型数组、程序应修改为：

```
main()
{float a[3][3],sum=0;
 int i,j;
 printf("Enter data:\n");
 for (i=0;i<3;i++)
   for (j=0;j<3;j++)
     scanf("%f",&a[i][j]);   /* 注意：在%f前有一空格，否则无法输入数据 */
 for (i=0;i<3;i++)
   printf("sum = %6.2f\n",sum);
   sum=sum+a[i][i];
}
```

在 Turbo C 2.0 环境下运行此程序时，出现运行错误，在输入数据后系统显示出错信息：

```
scanf: floating point formats not linked
Abnormal program termination
```

经过检查，程序的逻辑和语法都是正确的，而且在其他的 C 系统中（例如 Borland C++）可以正常运行。出现这种情况是所用的 C 编译系统不完善，处理的方法有两个：一是把程序移到其他 C 系统上运行（但往往不方便）；二是迁就所用的 C 系统，修改程序，避开其缺陷，这就需要通过多次试验掌握其规律。对以上程序来说，可以有多种代替的方案，例如可以把程序中第 5～7 行改为：

```
for (i=0;i<3;i++)
  scanf("%f %f %f",&a[i][0],&a[i][1],&a[i][2]);
```

它在效果上应与原来的第 5～7 行等价，但上机运行时发现仍不能正常运行，再改为：

```
    for (j=0;j<3;j++)
```

```
        scanf("%f %f %f",&a[0][j],&a[1][j],&a[2][j]);
```

它也是与原来的第 5～7 行等价的，上机运行时发现可以正常运行。程序如下：

```
main()
{float a[3][3],sum=0;
 int i,j;
 printf("Enter data:\n");
   for (j=0;j<3;j++)
     scanf("%f %f %f",&a[0][j],&a[1][j],&a[2][j]);
   for (i=0;i<3;i++)
     sum=sum+a[i][i];
   printf("sum=%6.2f\n",sum);
}
```

运行况如下：

```
Enter data:
1.1  2.2  3.3  4.4  5.5  6.6  7.7  8.8  9.9↵
```

应注意数据与数组元素的对应关系：1.1 是 a[0][0]，2.2 是 a[1][0]，3.3 是 a[2][0]……

通过此例，可以锻炼我们处理问题的能力。在上机调试程序时出错而又经反复检查没有错误时，只好采取"迂回"的办法，使程序能正常运行。

7.4 有一个已排好序的数组，今输入一个数，要求按原来排序的规律将它插入数组中。

解： N-S 图如图 7.3。

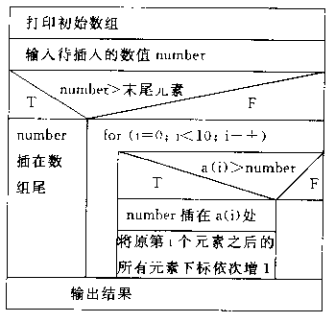


图 7.3

程序如下：

```
main()
{int a[11]={1,4,6,9,13,16,19,28,40,100};
 int temp1,temp2,number,end,i,j;
 printf("array a:\n");
```

```
    for (i=0;i<10;i++)
      printf("%5d",a[i]);
    printf("\n");
    printf("Insert data:");
    scanf("%d",&number);
    end=a[9];
    if (number>end)
      a[10]=number;
    else
      {for (i=0;i<10;i++)
        {if (a[i]>number)
          {temp1=a[i];
           a[i]=number;
           for (j=i+1;j<11;j++)
             {temp2=a[j];
              a[j]=temp1;
              temp1=temp2;
             }
           break;
          }
        }
      }
    printf("Now, array a:\n");
    for (i=0;i<11;i++)
      printf("%6d",a[i]);
}
```

运行情况如下：

```
array a:
    1    4    6    9    13    16    19    28    40    100
Insert data:5↙
Now,array a:
    1    4    5    6    9    13    16    19    28    40    100
```

7.5 将一个数组中的值按逆序重新存放。例如原来顺序为：8、6、5、4、1。要求改为：1、4、5、6、8。

解：N-S 图见图 7.4。

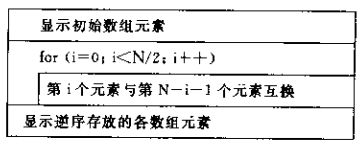


图 7.4

程序如下：

```
#define N 5
main()
{int a[N],i,temp;
 printf("Enter array a:\n");
 for (i=0;i<N;i++)
   scanf("%d",&a[i]);
 printf("array a:\n");
 for (i=0;i<N;i++)
   printf("%4d",a[i]);
 for (i=0;i<N/2;i++)
   {temp=a[i];
    a[i]=a[N-i-1];
    a[N-i-1]=temp;
   }
 printf("\n Now,array a:\n");
 for (i=0;i<N;i++)
   printf("%4d",a[i]);
 printf("\n");
}
```

运行情况如下：

```
Enter array a:
8  6  5  4  1↙
array a:
  8  6  5  4  1
Now, array a:
  1  4  5  6  8
```

7.6 打印出以下的杨辉三角形(要求打印出 10 行)。

```
1
1  1
1  2   1
1  3   3   1
1  4   6   4  1
1  5  10  10  5  1
⋮  ⋮   ⋮   ⋮  ⋮  ⋮
```

解：杨辉三角形是$(a+b)^n$展开后各项的系数。例如：

$(a+b)^0$ 展开后为	1	系数为 1
$(a+b)^1$ 展开后为	$a+b$,	系数为 1，1
$(a+b)^2$ 展开后为	$a^2+2ab+b^2$	系数为 1，2，1
$(a+b)^3$ 展开后为	$a^3+3a^2b+3ab^2+b^3$	系数为 1，3，3，1
$(a+b)^4$ 展开后为	$a^4+4a^3b+6a^2b^2+4ab^3+b^4$	系数为 1，4，6，4，1

以上就是杨辉三角形的前5行。杨辉三角形各行的系数有以下的规律：

(1) 各行第一个数都是1。

(2) 各行最后一个数都是1。

(3) 从第3行起，除上面指出的第一个数和最后一个数外，其余各数是上一行同列和前一列两个数之和。例如第4行第2个数(3)是第3行第2个数(2)和第3行第1个数(1)之和。可以这样表示：a[i][j]=a[i-1][j]+a[i-1][j-1]，其中i为行数，j为列数。

N-S图见图7.5。

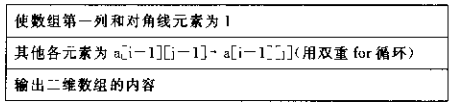


图 7.5

```
#define N 11
main()
{int i,j,a[N][N];
 for (i=1;i<N;i++)
   {a[i][i]=1;
    a[i][1]=1;
   }
 for (i=3;i<N;i++)
   for (j=2;j<=i-1;j++)
     a[i][j]=a[i-1][j-1]+a[i-1][j];
 for (i=1;i<N;i++)
   {for (j=1;j<=i;j++)
      printf("%6d",a[i][j]);
    printf("\n");
   }
 printf("\n");
}
```

运行结果：

```
1
1     1
1     2     1
1     3     3     1
1     4     6     4     1
1     5    10    10     5     1
1     6    15    20    15     6     1
1     7    21    35    35    21     7     1
1     8    28    56    70    56    28     8     1
1     9    36    84   126   126    84    36     9     1
```

7.7 打印"魔方阵"。所谓魔方阵是指这样的方阵，它的每一行、每一列和对角线之和均相等。例如，三阶魔方阵为

8 1 6

3 5 7

4 9 2

要求打印出由 $1\sim n^2$ 的自然数构成的魔方阵。

解：魔方阵中各数的排列规律如下：

（1）将 1 放在第一行中间一列；

（2）从 2 开始直到 $n\times n$ 止各数依次按下列规则存放：每一个数存放的行比前一个数的行数减 1，列数加 1（例如上面的三阶魔方阵，5 在 1 的上一行后一列）；

（3）如果上一数的行数为 1，则下一个数的行数为 n（指最下一行）。例如 1 在第 1 行，则 2 应放在最下一行，列数同样加 1；

（4）当上一个数的列数为 n 时，下一个数的列数应为 1，行数减 1。例如 2 在第 3 行最后一列，则 3 应放在第 2 行第 1 列；

（5）如果按上面规则确定的位置上已有数，或上一个数是第 1 行第 n 列时，则把下一个数放在上一个数的下面。例如按上面的规定，4 应该放在第 1 行第 2 列，但该位置已被 1 占据，所以 4 就放在 3 的下面。由于 6 是第 1 行第 3 列（即最后一列），故 7 放在 6 下面。按此方法可以得到任何阶的魔方阵。

N-S 图如图 7.6 所示。

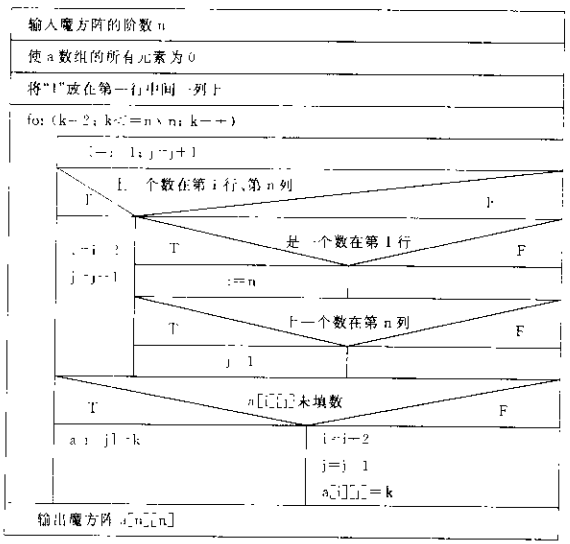


图 7.6

```
main()
{int a[16][16],i,j,k,p,m,n;
 p=1;
 while(p==1)                    /* 要求阶数为1~15的奇数 */
   {printf("Enter n(n=1~15):)");
    scanf("%d",&n);
    if ((n!=0) && (n<=15) && (n%2!=0))
    p=0;
   }
 /* 初始化 */
 for (i=1;i<=n;i++)
   for (j=1;j<=n;j++)
     a[i][j]=0;
 /* 建立魔方阵 */
 j=n/2+1;
 a[1][j]=1;
 for (k=2;k<=n*n;k++)
   {i=i-1;
    j=j+1;
    if ((i<1) && (j>n))
      {i=i+2;
       j=j-1;
      }
    else
      {if (i<1) i=n;
       if (j>n) j=1;
      }
    if (a[i][j]==0)
      a[i][j]=k;
    else
      {i=i+2;
       j=j-1;
       a[i][j]=k;
      }
   }
 for (i=1;i<=n;i++)                 /* 输出魔方阵 */
   {for (j=1;j<=n;j++)
      printf("%4d",a[i][j]);
    printf("\n");
   }
 }
```

运行结果：

```
Enter n(n=1~15): 5↙
```

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

说明：魔方阵的阶数应为奇数，程序指定其最大值为 15。今定义数组 a 为 16 行 16 列，第 0 行第 0 列不用来放数据，只用第 1～15 行，以符合人们的习惯。

7.8 找出一个二维数组中的鞍点，即该位置上的元素在该行上最大，在该列上最小。也可能没有鞍点。

解：N-S 图如图 7.7 所示。

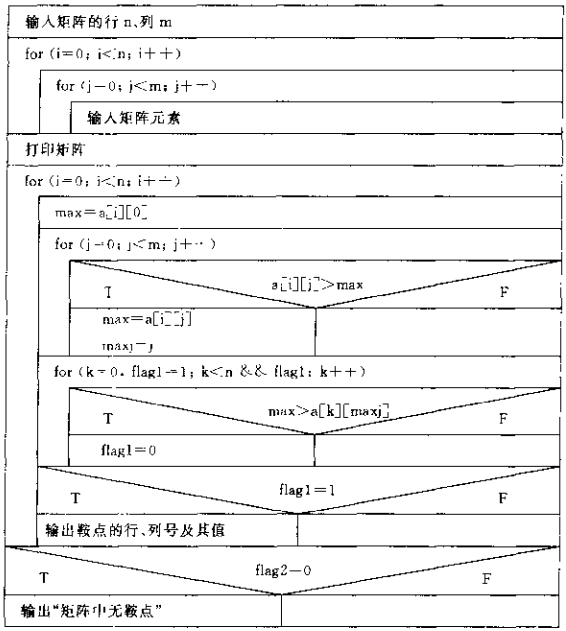


图 7.7

程序如下：

```
#define N 10
#define M 10
main()
{ int i,j,k,m,n,flag1,flag2,a[N][M],max,maxi,maxj;
```

```
    printf("\n 输入行数 n:");
    scanf("%d",&n);printf("\n 输入列数 m:");
    scanf("%d",&m);
    for (i=0;i<n;i++)
      {printf("第%d 行? \n",i);
       for (j=0;j<m;j++)
       scanf("%d",&a[i][j]);
      }
    for (i=0;i<n;i++)
      {for (j=0;j<m;j++)
         printf("%5d",a[i][j]);
       printf("\n");
      }
    flag2=0;
    for (i=0;i<n;i++)
    {max=a[i][0];
     for (j=0;j<m;j++)
       if (a[i][j]>max)
       {max=a[i][j];
        maxj=j;
       }
     for (k=0,flag1=1;k<n && flag1;k++)
       if (max>a[k][maxj])
         flag1=0;
       if (flag1)
         {printf("\n 第%d 行,第%d 列的%d 是鞍点\n",i,maxj,max);
          flag2=1;
         }
    }
      if(! flag2)
        printf("\n 矩阵中无鞍点! \n");
}
```

运行结果:

① 输入行数n:3↙
  输入列数m:4↙
  第 0 行?
  1 2 3 4↙
  第 1 行 ?
  4 5 5 6↙
  第 2 行 ?
  3 5 6 7↙

```
1    2    3    1
1    5    3    6
3    5    6    7
第0行,第3列的1是鞍点
② 输入行数n:3↙
  输入列数m:4↙
第0行?
2 4 90 7↙
第1行?
3 4 5 8↙
第2行?
9 1 2 3↙
  2    4   90    7
  3    4    5    8
  9    1    2    3
矩阵中无鞍点!
```

7.9 有15个数按由大到小的顺序存放在一个数组中,输入一个数,要求用折半查找法找出该数是数组中第几个元素的值。如果该数不在数组中,则打印出"无此表"。

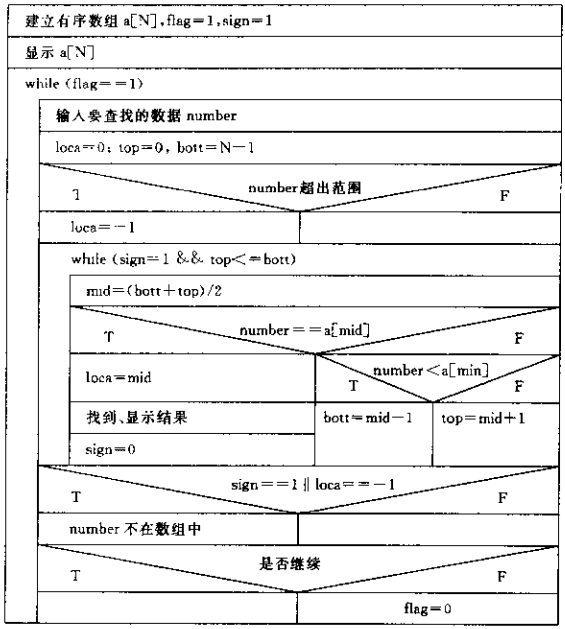
解:从表列中查一个数最简单的方法是从第1个数开始顺序查找,将要找的数与表列中的数一一比较,直到找到为止(如果表列中无此数,则应找到最后一个数,然后判定"找不到")。但这种"顺序查找法"效率较低。如果表列中有1000个数,且要找的数恰恰是第1000个数,则要进行1000次比较才能得到结果。平均比较次数为500次。

折半查找法是效率较高的一种方法。基本思路如下:

假如有已按由小到大排好序的9个数,a[1]~a[9],其值分别为:1、3、5、7、9、11、13、15、17。

若输入一个数3,想查3是否在此数列中,先找出表列中居中的数,即a[5],将要找的数3与a[5]比较,a[5]的值是9,发现a[5]>3,显然3应当在a[1]到a[5]范围内,而不会在a[6]到a[9]范围内。这样就可以缩小查找范围,甩掉a[6]到a[9]这一部分,即将查找范围缩小为一半。再找a[1]到a[5]范围内的居中的数,即a[3],将要找的数3与a[3]比较,a[3]的值是5,发现a[3]>3,显然3应当在a[1]到a[3]范围内。这样又将查找范围缩小一半,再将3与a[1]到a[3]范围内的居中的数a[2]比较,发现要找的数3等于a[2],查找结束。一共比较了3次。如果表列中有n个数,则最多比较的次数为int(log$n$)+1。

N-S图如图7.8所示。



top,bott：查找区间两端点的下标；loca：查找成功与否的开关变量。

图 7.8

程序如下：

```
#include <stdio.h>
#define N 15
main()
{ int i,j,number,top,bott,mid,loca,a[N],flag=1,sign=1;
  char c;
  printf("Enter data:\n");
  scanf("%d",&a[0]);
  i=1;
  while(i<N)
    {scanf("%d",&a[i]);
     if (a[i]>=a[i-1])
        i++;
     else
        printf("Enter this data again:");
```

```
    printf("\n");
    for (i=0;i<N;i++)
      printf("%4d",a[i]);
    printf("\n");
    flag=1;
    while(flag)
      {printf("Input number to look for:");
       scanf("%d",&number);
       loca=0;
       top=0;
       bott=N-1;
       if ((number<a[0])||(number>a[N-1]))
         loca=-1;
       while ((sign==1)&&(top<=bott))
         {mid=(bott+top)/2;
          if (number==a[mid])
            {loca=mid;
             printf("Find %d, its position is %d \n",number,loca+1);
             sign=0;
            }
          else if (number<a[mid])
                 bott=mid-1;
               else
                 top=mid+1;
         }
       if(sign==1||loca==-1)
         printf("%d is not found. \n",number);
       printf("Continue or not(Y/N)?");
       scanf(" %c",&c);
       if (c=='N'||c=='n')
          flag=0;
      }
}
```

运行情况如下：

```
Enter data:
1↙
3↙
2↙
Enter this data again:4↙
5↙
6↙
```

8↙
12↙
23↙
34↙
44↙
45↙
56↙
57↙
58↙
68↙
1 3 4 5 6 8 12 23 34 44 45 56 57 58 68
Input number to look for:7↙
7 is not found.
Continue or not(Y/N)? y↙
Input number to look for:12↙
Find 12, its position is 7
Continue or not(Y/N)? n↙
(运行结束)

7.10 有一篇文章,共有3行文字,每行有80个字符。要求分别统计出其中英文大写字母、小写字母、数字、空格以及其他字符的个数。

解:N-S图如图7.9。

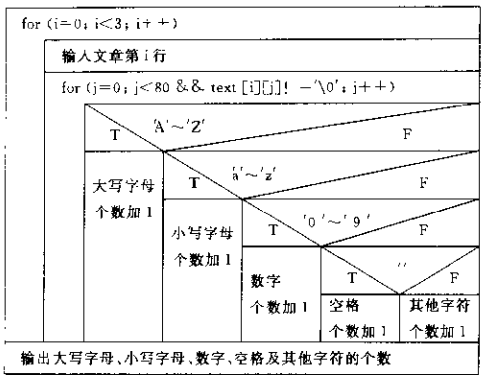


图 7.9

程序如下:

```
main()
{int i,j,upp,low,dig,spa,oth;
 char text[3][80];
 upp=low=dig=spa=oth=0;
```

```
for (i=0;i<3;i++)
  {printf("\n Please input line %d:\n",i+1);
   gets(text[i]);
   for (j=0;j<80 && text[i][j]!='\0';j++)
     {if (text[i][j]>='A'&& text[i][j]<='Z')
        upp++;
      else if (text[i][j]>='a' && text[i][j]<='z')
        low++;
      else if (text[i][j]>='0' && text[i][j]<='9')
        dig++;
      else if (text[i][j]==' ')
        spa++;
      else
        oth++;
     }
  }
for (i=0;i<3;i++)
  printf("%s\n",text[i]);
printf("upper case: %d\n",upp);
printf("lower case: %d\n",low);
printf("digit : %d\n",dig);
printf("space : %d\n",spa);
printf("other : %d\n",oth);
}
```

运行情况如下：

```
Please input line 1:
I am a student.↙

Please input line 2:
123456↙

Please input line 3:
ASDFG↙
upper case :6
lower case:10
digit      : 6
space      : 3
other      :1
```

说明：数组 text 的行号为 0～2，但在提示用户输入各行数据时，要求用户输入第 1 行、第 2 行、第 3 行，而不是第 0 行、第 1 行、第 2 行，这完全是照顾人们的习

惯。为此，在程序第 6 行中输出行数时用 i+1，而不用 i。这样并不影响程序对数组的处理，程序其他地方数组的第 1 个下标值仍然是 0～2。

7.11 打印以下图案：

```
 * * * * *
  * * * * *
   * * * * *
    * * * * *
     * * * * *
```

解：程序如下：

```
main()
{ char a[5]={'*','*','*','*','*'};
  int i,j,k;
  char space=' ';
  for (i=0;i<5;i++)                    /* 输出 5 行 */
    { printf("\n");                    /* 输出每行前先换行 */
      printf(" ");                     /* 每行前面留 4 个空格 */
      for (j=1;j<=i;j++)
        printf("%c",space);            /* 每行再留 1 个空格 */
      for (k=0;k<5;k++)
        printf("%c",a[k]);             /* 每行输出 5 个 * 号 */
    }
}
```

运行结果：

```
 * * * * *
  * * * * *
   * * * * *
    * * * * *
     * * * * *
```

7.12 有一行电文，已按下面规律译成密码：

A→Z　　a→z

B→Y　　b→y

C→X　　c→x

⋮　　⋮

即第一个字母变成第 26 个字母，第 i 个字母变成第(26－i+1)个字母。非字母字符不变。要求编程序将密码译回原文，并打印出密码和原文。

解：N-S 图如图 7.10。

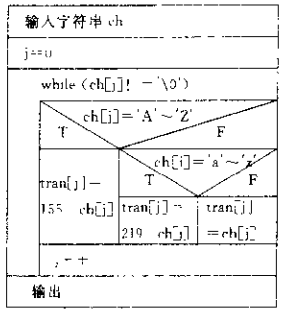


图 7.10

说明：字符 ch[j]如果是大写字母，则它是第(ch[j]-64)个大写字母。若 ch[j]的值是大写字母'B'，它的 ASCII 码为 66，它应是第(66－64)＝2 个大写字母。按密码规定应将它转换为第(26－i+1)个大写字母，即第(26－2+1)＝25 个大写字母。而 26－i+1＝26－(ch[j]-64)+1＝26+64－ch[j]+1，即 91－ch[j]（如 ch[j]等于'B'，91－'B'＝91－66＝25，C[j]应与第 25 个大写字母对换）。该字母的 ASCII 码为 91－ch[j]+64，即 25＋64＝89，89 是'Y'的 ASCII 码。可以表示为 155－ch[j]。小写字母情况与此相似，但由于'a'的 ASCII 码为 97，因此公式应改为 26+96－ch[j]+1+96＝123－ch[j]+96＝219－ch[j]。若 ch[j]的值为'b'，则其交换对象为 219－'b'＝219－98＝121，它是'y'的 ASCII 码。

由于此密码的规律是对称交换，即第一个字母与最后一个字母交换，第二个字母与最后第二个字母交换……因此从原文译为密码和从密码译为原文，都是用同一个公式。

程序如下：

```
#include <stdio.h>
main()
{int j,n;
 char ch[80],tran[80];
 printf("\nInput cipher code:");
 gets(ch);
 printf("\n\ncipher code:%s",ch);
 j=0;
 while (ch[j]! ='\0')
 { if ((ch[j]>='A') && (ch[j]<='Z'))
       tran[j]=155-ch[j];
   else if ((ch[j]>='a') && (ch[j]<='z'))
         tran[j]=219-ch[j];
       else
```

```
            tran[j]=ch[j];
        j++;
    }
    n=j;
    printf("\noriginal text:");
    for (j=0;j<n;j++)
      putchar(tran[j]);
    printf("\n");
}
```

运行情况如下：

```
Input cipher code:R droo erhrg Xsrmz mvcg dvvp.↙

cipher code : R droo erhrg Xsrmz mvcg dvvp.
original text: I will visit China next week.
```

7.13 编一个程序，将两个字符串连接起来，不要用 strcat 函数。

解： N-S 图如图 7.11。

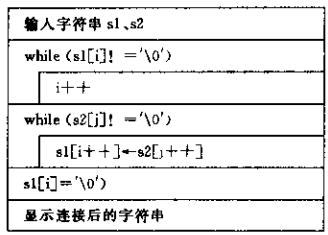


图 7.11

```
main()
{char s1[80],s2[40];
 int i=0,j=0;
 printf("\nInput string1:");
 scanf("%s",s1);
 printf("Input string2:");
 scanf("%s",s2);
 while (s1[i]! ='\0')
   i++;
 while(s2[j]! ='\0')
   s1[i++]=s2[j++];
 s1[i]='\0';
 printf("The new string is:%s",s1);
}
```

运行结果：

```
Input string1: country↙
Input string2: side↙
The new string is:countryside
```

7.14 编一个程序，将两个字符串 s1 和 s2 进行比较。如果 s1>s2，输出一个正数；s1=s2，输出 0；s1<s2，输出一个负数。不要用 strcmp 函数。两个字符串用 gets 函数读入。输出的正数或负数的绝对值应是相比较的两个字符串相应字符的 ASCII 码的差值。例如，'A'与'C'相比，由于'A'<'C'，应输出负数，由于'A'与'C'的 ASCII 码的差值为 2，因此，应输出"-2"。同理："And"和"Aid"比较，根据第 2 个字符比较结果，'n'比'i'大 5，因此应输出'5'。

解：N-S 图如图 7.12。

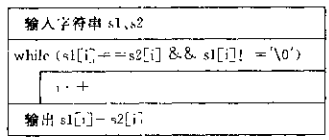


图 7.12

程序如下：

```
#include <stdio.h>
main()
{ int i,resu;
  char s1[100],s2[100];
  printf("\n input string1:");
  gets(s1);
  printf("\n Input string2:");
  gets(s2);
  i=0;
  while ((s1[i]==s2[i]) && (s1[i]!='\0')) i++;
  if (s1[i]=='\0' && s2[i]=='\0') resu=0;
  else
    resu=s1[i]-s2[i];
  printf("\n result:%d.\n",resu);
}
```

运行情况如下：

```
Input string1:aid↙
Input string2:and↙
result:-5
```

7.15 编写一个程序，将字符数组 s2 中的全部字符拷贝到字符数组 s1 中，不用 strcpy 函数。拷贝时，'\0'也要拷贝过去，'\0'后面的字符不拷贝。

解：程序如下：

```
#include "stdio.h"
main()
{ char s1[80],s2[80];
  int i;
  printf("Input s2:");
  scanf("%s",s2);
  for (i=0;i<=strlen(s2);i++)
    s1[i]=s2[i];
  printf("s1: %s\n",s1);
}
```

运行情况如下：

```
Input s2:student↙
s1: student
```

第8章 函 数

8.1 写两个函数，分别求两个整数的最大公约数和最小公倍数，用主函数调用这两个函数，并输出结果，两个整数由键盘输入。

解：设两个整数为 u 和 v，用辗转相除法求最大公约数的算法见图 8.1 所示。

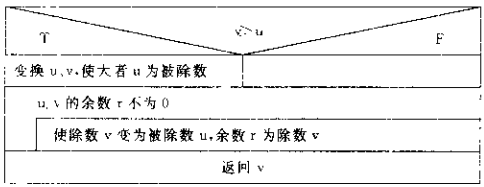


图 8.1

最小公倍数＝uv/最大公约数。据此写出程序：

```
int hcf(int u,int v)
{int t,r;
 if (v,>u)
    {t  u;u=v;v= t;}
 while ((r=u,%v)! =0)
     u=v;
     v= r;}
 return(v);
}

int lcd(int u,int v,int h)
{
return(u * v/ h);
}

main()
{int u,v,h,l;
 scanf("%d,%d",& u,& v);
 h  hcf(u,v);
 printf("H. C. F=%d\n",h);
 l  lcd(u,v,h);
 printf("L. C. D=%d\n",l);
}
```

运行结果如下：

```
24,16↙
H. C. F=8
L. C. D=48
```

8.2 求方程 $ax^2+bx+c=0$ 的根，用 3 个函数分别求当 b^2-4ac 大于 0、等于 0 和小于 0 时的根，并输出结果。从主函数输入 a、b、c 的值。

解：程序如下：

```
#include <math.h>
float x1,x2,disc,p,q;
greater_than  zero(float a,float b)          /* 定义一个函数,用来求 disc>0 时方程的根 */
  {x1=(-b+sqrt(disc))/(2*a);
   x2=(-b-sqrt(disc))/(2*a);
  }
equal_to_zero(float a,float b)               /* 定义一个函数,用来求 disc=0 时方程的根 */
  {
   x1=x2=(-b)/(2*a);
  }
smaller  than_zero(float a,float b)          /* 定义一个函数,用来求 disc<0 时方程的根 */
  {
   p=-b/(2*a);
   q=sqrt(disc)/(2*a);
  }
main()
{
 float a,b,c;
 printf("\nInput a,b,c:");
 scanf("%f,%f,%f",&a,&b,&c);
 printf("\nequation: %5.2f*x*x+%5.2f*x+%5.2f=0\n",a,b,c);
 disc=b*b-4*a*c;
 printf("root:\n");
 if (disc>0)
   {
    greater_than_zero(a,b);
    printf("x1=%5.2f\tx2=%5.2f\n\n",x1,x2);
   }
 else if (disc==0)
       {equal_to_zero(a,b);
        printf("x1=%5.2f\tx2=%5.2f\n\n",x1,x2);
       }
      else
```

```
    {smaller_than_zero(a,b);
     printf("x1=%5.2f+5.2fi\tx2=%5.2f-%%5.2fi\n",p,q,p,q);
    }
}
```

运行结果：

```
Input a,b,c:1,2,1↵
equation: 1.00*x*x+2.00*x+1.00=0
root:
x1=-1.00x2=-1.00
```

8.3 写一个判素数的函数，在主函数输入一个整数，输出是否素数的信息。

解：

```
main()
{int prime(int);                    /* 函数原型声明 */
 int n;
 printf("\nInput an integer:");
 if (prime(n))
   printf("\n %d is a prime.",n);
 else
   printf("\n %d is not a prime.",n);
}

int prime(int n)
{int flag=1,i;
 for (i=2;i<n/2 && flag==1;i++)
   if (n%i==0)
     flag=0;
 return(flag);
}
```

运行结果：

```
Input an integer:17↵
17 is a prime.
```

8.4 写一函数，使给定的一个二维数组(3×3)转置，即行列互换。

解：

```
#define N 3
int array[N][N];
convert(int array[3][3])            /* 定义转置数组的函数 */
  {int i,j,t;
   for (i=0;i<N;i++)
     for (j=i+1;j<N;j++)
       {t=array[i][j];
```

```
            array[i][j]=array[j][i];
            array[j][i]=t;
           }
    }
main()
{int i,j;
 printf("Input array:\n");
 for (i=0;i<N;i++)
   for (j=0;j<N;j++)
     scanf("%d",&array[i][j]);
 printf("\noriginal array :\n");
 for (i=0;i<N;i++)
   {for (j=0;j<N;j++)
      printf("%5d",array[i][j]);
    printf("\n");
   }
 convert (array);
 printf ("convert array:\n");
 for (i=0;i<N;i++)
   {for (j=0;j<N;j++)
      printf ("%sd",array[i][j]);
    printf ("\n");
   }
}
```

运行结果：

```
Input array:
1 2 3 4 5 6 7 8 9↙
original array:
  1  2  3
  4  5  6
  7  8  9
convert array:
  1  4  7
  2  5  8
  3  6  9
```

8.5 写一函数，使输入的一个字符串按反序存放，在主函数中输入和输出字符串。

解：

```
main()
{int inverse(char str[]);                    /* 函数原型声明 */
```

```
 char str[100];
 printf("Input string:");
 scanf("%s",str);
 inverse(str);
 printf("Inverse string:%s\n",str);
}
int inverse(char str[])                    /* 函数定义 */
{char t;
 int i,j;
 for (i=0,j=strlen(str);i<strlen(str)/2;i++,j--)
   {t=str[i];
    str[i]=str[j-1];
    str[j-1]=t;
   }
}
```

运行结果：

```
Input string:abcdefg↙
Inverse string:gfedcba
```

8.6 写一函数，将两个字符串连接。

解：

```
char concatenate(char string1[],char string2[],char string[])
{int i,j;
 for (i=0;string1[i]!='\0';i++)
   string[i]=string1[i];
 for(j=0;string2[j]!='\0';j++)
   string[i+j]=string2[j];
 string[i+j]='\0';
}
main()
{char s1[100],s2[100],s[100];
 printf("\nInput string1:");
 scanf("%s",s1);
 printf("Input string2:");
 scanf("%s",s2);
 concatenate(s1,s2,s);
 printf("The new string is %s\n",s);
}
```

运行结果：

```
Input string1:country↙
```

```
Input string2:side↙
The new string is countryside
```

8.7 写一函数，将一个字符串中的元音字母复制到另一字符串，然后输出。

解：

```
main()
{void cpy(char s[],char c[]);
 char sty[80],c[80];
 printf("\nInput string:");
 gets(sty);
 cpy(sty,c);
 printf("\nThe vowel letters are :%s",c);
}
void cpy(char s[],char c[])
{int i,j;
 for (i=0,j=0;s[i]! ='\0';i++)
   if (s[i]=='a'||s[i]=='A'.|s[i]=='e'||s[i]=='E'||s[i]=='i'||
     s[i]=='I'||s[i]=='o'||s[i]=='O'||s[i]=='u'||s[i]=='U')
     {c[j]=s[i];
      j++;
     }
 c[j]='\0';
}
```

运行结果：

```
Input string:abcdefghijklmn↙
The vowel letters are: aei
```

8.8 写一函数，输入一个 4 位数字，要求输出这 4 个数字字符，但每两个数字间有一个空格。如输入 1990，应输出“1␣9␣9␣0”。

解：

```
main()
{char str[80];
 void insert(char str[]);
 printf("\nInput four digits:");
 scanf("%s",str);
 insert(str);
}
void insert(char str[])
{int i;
 for (i=strlen(str);i>0;i--)
 {str[2*i]=str[i];
```

```
  str[2*i-1]=' ';
 }
 printf("\nOutput:\n%s",str);
}
```

运行结果：

```
Input four digits:1357↙
Output:
1 3 5 7
```

8.9 编写一函数，由实参传来一个字符串，统计此字符串中字母、数字、空格和其他字符的个数，在主函数中输入字符串以及输出上述的结果。

解：

```
int letter,digit,space,others;          /* 全局变量 */
main()
{int count(char str[]);                 /* 函数声明 */
 char text[80];
 printf("\nInput string:\n");
 gets(text);
 printf("string:");
 puts(text);
 letter=0;
 digit=0;
 space=0;
 others=0;
 count(text);
 printf("letter:%d,digit:%d,space:%d,others:%d\n",letter,digit,space,others);
}
int count(char str[])
{int i;
 for (i=0;str[i]!='\0';i++)
 if ((str[i]>='a'&& str[i]<='z')||(str[i]>='A' && str[i]<='Z'))
   letter++;
 else if (str[i]>='0' && str [i]<='9')
      digit++;
 else if (strcmp(str[i],' ')==0)
      space++;
 else
      others++;
}
```

运行结果：

```
Input string:
My address is #123 Shanghai Road, Beijing,100045.↙
string:My address is #123 Shanghai Road, Beijing,100045.
letter:30,digit:9,space:5,others:4
```

8.10 写一函数，输入一行字符，将此字符串中最长的单词输出。

解：认为单词是全由字母组成的字符串，程序中设 longest 函数，作用是找最长单词的位置。此函数的返回值是该行字符中最长单词的起始位置。longest 函数的 N-S 图见图 8.2。

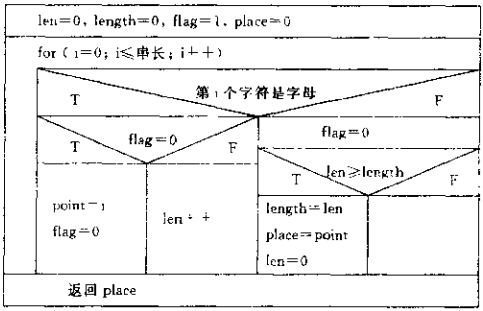


图 8.2

图中用 flag 表示单词是否已开始，flag=0 表示未开始，flag=1 表示单词开始；len 代表当前单词已累计的字母个数；length 代表先前单词中最长单词的长度；point 代表当前单词的起始位置(用下标表示)；place 代表最长单词的起始位置。

函数 alphabetic 的作用是判断当前字符是否字母，若是则返回 1，否则返回 0。

程序如下：

```
int alphabetic(char c)          /* 判断当前字符是否字母,若是则返回1,否则返回0 */
{if ((c>='a' && c<='z')||(c>='A'&&c<='z'))
   return(1);
 else
   return(0);
}

int longest(char string[])              /* 寻找最长单词的起始位置 */
{int len=0,i,length=0,flag=1,place=0,point;
  for (i=0;i<=strlen(string);i++)
    if (alphabetic(string[i]))
      if (flag)
        {point=i;
         flag=0;
```

```
            }
          else
            len = 0;
        else
        {flag = 1;
          if (len >= length)
          {length=len;
           place=point;
           len = 0;
          }
        }
    return(place);
}
main()
{int i;
 char line[100];
 printf("Input one line:\n");
 gets(line);
 printf("\nThe longest word is :");
 for (i=longest(line);alphabetic(line[i]);i++)
   printf("%c",line[i]);
 printf("\n");
}
```

运行结果：

```
Input one line:
I am a student.↙
The longest word is :student
```

8.11 写一函数，用“起泡法”对输入的10个字符按由小到大顺序排列。

解： 主函数main()的N-S图如图8.3(a)。sort()函数的作用是排序，其N-S图如图8.3(b)。

程序如下：

```
#define N 10
char str[N];
main()
{void sort(char str[]);
 int i,flag;
 for (flag=1;flag==1;)
   {printf("\nInput string:\n");
    scanf("%s",str);
    if (strlen(str)>N)
      printf("String too long,input again!");
```

```
    else
      flag=0;
  }
  sort(str);
  printf("string sorted:\n");
  for (i=0;i<N;i++)
    printf("%c",str[i]);
}
void sort(char str[])
{int i,j;
 char t;
 for(j=1;j<N;j++)
   for (i=0;(i<N-j)&&(str[i]! ='\0');i++)
     if(str[i]>str[i+1])
       {t=str[i];
        str[i]=str[i+1];
        str[i+1]=t;
       }
}
```

运行结果：

```
Input string:
reputation↙
string sorted:
aeionprttu
```

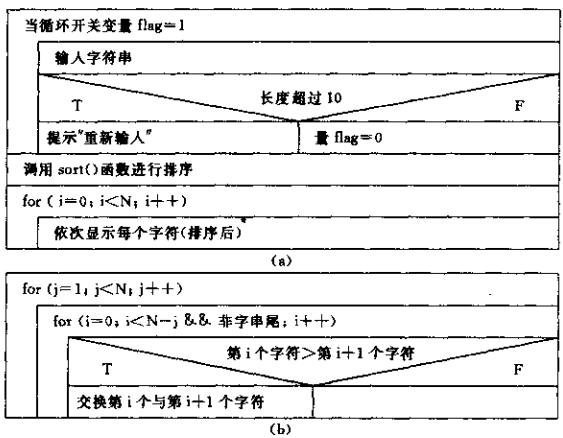


图 8.3

8.12 用牛顿迭代法求根。方程为 $ax^3+bx^2+cx+d=0$，系数 a、b、c、d 的值依次为 1、2、3、4。求 x 在 1 附近的一个实根。求出根后由主函数输出。

解：牛顿迭代法的公式是 $x=x_0-f(x)/f'(x)$，设迭代到 $|x-x_0|\leqslant 10^{-3}$ 时结束。用牛顿迭代法求根的函数 solut 的 N-S 图如图 8.4。

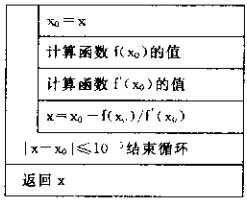


图 8.4

程序如下：

```
#include <math.h>
float solut(float a,float b,float c,float d)
{float x=1,x0,f,f1;
 do
   {x0=x;
    f=((a*x0+b)*x0+c)*x0+d;
    f1=(3*a*x0+2*b)*x0+c;
    x=x0-f/f1;
   }
   while(fabs(x-x0)>=1e-3);
 return(x);
}
main()
{
  float a,b,c,d;
  printf("\nInput a,b,c,d:");
  scanf("%f,%f,%f,%f",&a,&b,&c,&d);
  printf("Equation is:%5.2fx^3+%5.2fx^2+%5.2fx+%5.2f=0",a,b,c,d);
  printf("\nx=%10.7f\n",solut(a,b,c,d));
}
```

运行结果：

```
Input a,b,c,d: 1,2,3,4↙
Equation is:1.00x^3+ 2.00x^2+ 3.00x+ 4.00=0
X=-1.6506292
```

8.13 用递归方法求 n 阶勒让德多项式的值，递归公式为

$$P_n(x)=\begin{cases}1 & (n=0)\\ x & (n=1)\\ ((2n-1)xP_{n-1}(x)-(n-1)P_{n-2}(x))/n & (n>1)\end{cases}$$

解：递归函数 p()的 N-S 图如图 8.5。

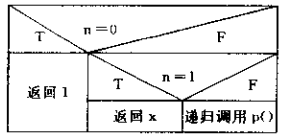


图 8.5

```
main()
{int x,n;
 float p(int,int);
 printf("\nInput n & x:");
 scanf("%d,%d",&n,&x);
 printf("n=%d,x=%d \n",n,x);
 printf("P%d(%d)=%6.2f",n,x,p(n,x));
}

float p(int n,int x)
{if (n==0)
   return(1);
 else if (n==1)
        return(x);
      else
        return(((2*n-1)*x*p((n-1),x)-(n-1)*p((n-2),x))/n);
}
```

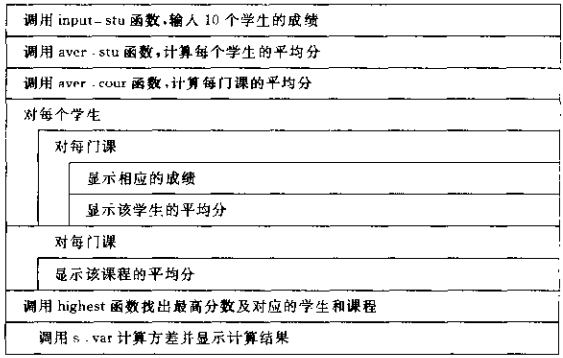
运行结果：

① Input n & x:0,7↙
   n=0,x=7
   P0(7)=1.00
② Input n & x:1,2↙
   n=1,x=2
   P1(2)=2.00
③ Input n & x:3,4↙
   n=3,x=4
   P3(4)=154.00

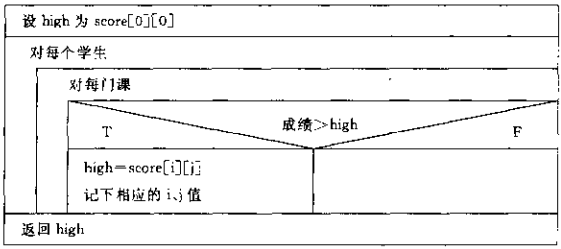
8.14 输入 10 个学生 5 门课的成绩，分别用函数求：(1) 每个学生的平均分；(2) 每门课的平均分；(3) 找出最高的分数所对应的学生和课程；(4) 求出平均分方差：$\sigma=$

$\frac{1}{n}\sum x_i^2-\left(\frac{\sum x_i}{n}\right)^2$，$x_i$ 为某一学生的平均分。

解：主函数的 N-S 图见图 8.6(a)。



(a)



(b)

图 8.6

函数 input_stu 的执行结果是给全程变量学生成绩数组 score 各元素输入初值。

函数 aver_stu 的作用是计算每个学生的平均分，并将结果赋给全程变量数组 a_stu 中各元素。

函数 aver_cour 的作用是计算每门课的平均成绩，计算结果存入全程变量数组 a_cour。

函数 highest 的返回值是最高分，r、c 是两个全局变量，分别代表最高分所在的行、列号。该函数的 N-S 图见图 8.6(b)。

函数 s_var 的返回值是平均分的方差。

程序如下：

```
#define N 10
#define M 5
```

```
float score[N][M];                    /* 全局数组 */
float a_stu[N],a_cour[M];             /* 全局数组 */
int r,c;                              /* 全局变量 */

main()
{ float s_var(void);                  /* 函数原型声明 */
  float highest();                    /* 函数原型声明 */
  void input_stu(void);               /* 函数原型声明 */
  void aver_stu(void);                /* 函数原型声明 */
  void aver_cour(void);               /* 函数原型声明 */
  int i,j;
  float h;
  input_stu();                        /* 函数调用,输入 10 个学生成绩 */
  aver_stu();                         /* 函数调用,计算 10 个学生平均成绩 */
  aver_cour();                        /* 函数调用,计算 5 门课平均成绩 */
  printf("\n NO.      cour1  cour2  cour3  cour4  cour5  aver");
  for(i=0;i<N;i++)
    {printf("\n NO. %2d ",i+1);      /* 输出 1 个学生号码 */
     for(j=0;j<M;j++)                 /* 输出 1 个学生各门课的成绩 */
       printf("%8.2f",score[i][j]);
     printf("%8.2f",a_stu[i]);        /* 输出 1 个学生的平均成绩 */
    }
  printf("\naverage:");
  for (j=0;j<M;j++)                   /* 输出 5 门课平均成绩 */
    printf("%8.2f",a_cour[j]);
  h=highest();                        /* 调用函数,求最高分和它属于哪个学生、哪门课 */
  printf("\nhighest:%7.2f  NO.%2d  course%2d\n",h,r,c);
                                      /* 输出最高分和学生号、课程号 */
  printf("\n variance:%8.2f",s_var());      /* 调用函数,计算和输出方差 */
}

void input_stu(void)                  /* 输入 10 个学生成绩的函数 */
{int i,j;
 for (i=0;i<N;i++)
   {printf("\nInput score of student%2d:\n",i+1);      /* 学生号从 1 开始 */
    for (j=0;j<M;j++)
      scanf("%f",&score[i][j]);
   }
}

void aver_stu(void)                   /* 计算 10 个学生平均成绩的函数 */
{int i,j;
 float s;
 for (i=0;i<N;i++)
```

```
    {for (j=0,s=0;j<M;j++)
       s+=score[i][j];
     a_stu[i]=s/5.0;
    }
}
void aver_cour(void)                    /* 计算5门课平均成绩的函数 */
{int i,j;
 float s;
 for (j=0;j<M;j++)
   {s=0;
    for (i=0;i<N;i++)
      s+=score[i][j];
    a_cour[j]=s/(float)N;
   }
}
float highest()                         /* 求最高分和它属于哪个学生、哪门课的函数 */
{float high;
 int i,j;
 high=score[0][0];
 for (i=0;i<N;i++)
   for (j=0;j<M;j++)
     if (score[i][j]>high)
       {high=score[i][j];
        r=i+1;                          /* 数组行号i从0开始,学生号r从1开始,故r=i+1 */
        c=j+1;                          /* 数组列号j从0开始,课程号c从1开始,故c=i+1 */
       }
 return(high);
}
float s_var(void)                       /* 求方差的函数 */
{int i,j;
 float sumx,sumxn;
 sumx=0.0;
 sumxn=0.0;
 for (i=0;i<N;i++)
   {sumx+=a_stu[i]*a_stu[i];
    sumxn+=a_stu[i];
   }
 return(sumx/N-(sumxn/N)*(sumxn/N));
}
```

运行情况如下：

```
Input score of student 1:
```

```
91 79 83 83 91↙
Input score of student 2:
95 71 81 80 83↙
Input score of student 3:
86 82 77 91 88↙
Input score of student 4:
85 76 79 81 82↙
Input score of student 5:
88 76 84 81 80↙
Input score of student 6:
87 72 79 91 84↙
Input score of student 7:
85 77 75 84 93↙
Input score of student 8:
66 62 64 54 70↙
Input score of student 9:
89 67 75 86 93↙
Input score of student 10:
75 73 70 85 82↙
NO.      cour1    cour2    cour3    cour4    cour5    aver
NO.  1   91.00    79.00    83.00    83.00    91.00    85.40
NO.  2   95.00    71.00    81.00    80.00    83.00    82.00
NO.  3   86.00    82.00    77.00    91.00    88.00    84.80
NO.  4   85.00    76.00    79.00    81.00    82.00    80.60
NO.  5   88.00    76.00    84.00    81.00    80.00    81.80
NO.  6   87.00    72.00    79.00    91.00    84.00    82.60
NO.  7   85.00    77.00    75.00    84.00    93.00    82.80
NO.  8   66.00    62.00    64.00    54.00    70.00    63.20
NO.  9   89.00    67.00    75.00    86.00    93.00    82.00
NO.  10  75.00    73.00    70.00    85.00    82.00    77.00
average:  84.70   73.50    76.70    81.60    84.60
highest:  95.00   NO. 2    course 1
variance:  6.92
```

8.15 写几个函数：(1) 输入 10 个职工的姓名和职工号；(2) 按职工号由小到大排序，姓名顺序也随之调整；(3) 要求输入一个职工号，用折半查找法找出该职工的姓名。从主函数输入要查找的职工号，输出该职工姓名。

解：input_e 函数完成 10 个职工的数据的录入。sort 函数的作用是选择法排序，其流程类似于 7.2 题。

search 函数的作用是用折半查找的方法找出指定职工号的职工姓名，其查找的算法参见 7.9 题。

主函数的 N-S 图见图 8.7。

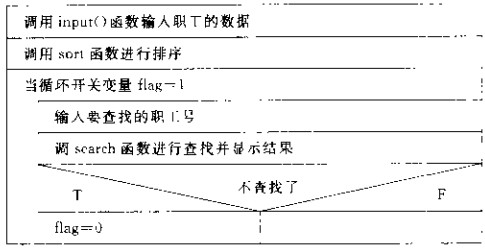


图 8.7

程序如下：

```
#include <stdio.h>
#define N 10
void input(int num[],char name[N][8])          /* 输入数据的函数 */
{int i;
 for (i=0;i<N;i++)
   {printf("\n Input NO. : ");
    scanf("%d",&num[i]);
    printf("Input name:");
    getchar();
    gets(name[i]);
   }
}
void sort(int num[ ],char name[N][8])          /* 排序的函数 */
{int i,j,min,temp1;
 char temp2[8];
 for (i=0;i<N-1;i++)
  {min=i;
   for (j=i;j<N;j++)
     if (num[min]>num[j])  min  j;
   temp1=num[i];
   strcpy(temp2,name[i]);
   num[i]=num[min];
   strcpy (name[i],name[min]);
   num[min]=temp1;
   strcpy(name[min],temp2);
   }
 printf("\n result:\n");
 for (i=0;i<N;i++)
```

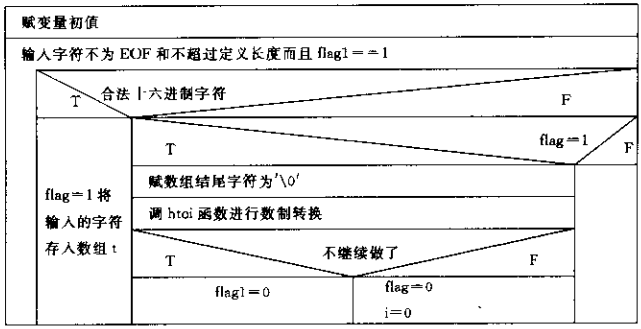
```
  printf("\n %5d%10s",num[i],name[i]);
}
void search(int n,num[],char name[N][8])      /* 折半查找函数 */
{int top,bott,mid,loca;
 loca=0;
 top=0;
 bott=N-1;
 if ((n<num[0])||(n>num[N-1]))
   loca=-1;
 while((sign==1) && (top<=bott))
   {min=(bott+top)/2;
    if (n==num[mid])
      {loca=mid;
       printf("NO. %d , his name is %s . \n",n,name[loca]);
       sign=-1
      }
    else if (n<num[mid])
           bott=mid-1;
         else
           top=mid +1;
   }
 if (sign==-1 || loca==-1)
   printf("Can not find %d. \n",n);
}
main()
{ int num[N],numher,flag=1,c,n;
  char name[N][8];
  input(num,name);
  sort(num,name);
  while (flag==1)
    {printf("\nInput number to look for:");
     scanf("%d",&number);
     search(number,num,name);
     printf("Continue or not(Y/N)?");
     getchar();
     c=getchar();
     if (c=='N'||c=='n')
       flag=0;
    }
}
```

运行结果：

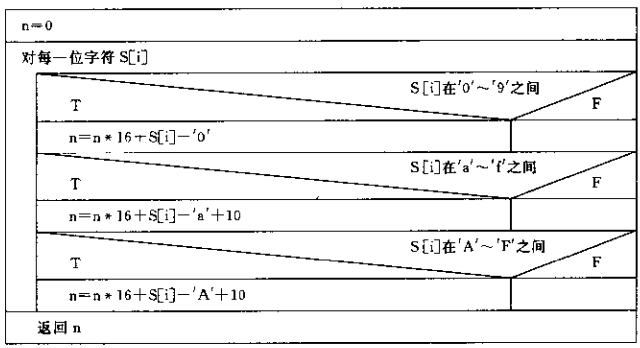
```
Input NO. :1↙
Input name:Li↙
Input NO. :2↙
Input name:Wang↙
Input NO. :5↙
Input name:Liu↙
Input NO. :8↙
Input name:Ma↙
Input NO. :4↙
Input name:Chen↙
Input NO. :10↙
Input name:Zhou↙
Input NO. :12↙
Input name:Zhang↙
Input NO. :6↙
Input name:Xie↙
Input NO. :23↙
Input name:Yuan↙
Input NO. :34↙
Input name:Lu↙
result:
 1    Li
 2    Wang
 4    Chen
 5    Liu
 6    Xie
 8    Ma
10    Zhou
12    Zhang
23    Yuan
34    Lu
Input number to look for:3↙
Can not find 3.
Continue or not(Y/N)? y↙
Input number to look for:6↙
NO. 6, his name is Xie.
Continue or not(Y/N)? n↙
(程序运行结束)
```

8.16 写一函数，输入一个十六进制数，输出相应的十进制数。

解：主函数 main 的 N-S 图见图 8.8(a)。求十进制数的函数 htoi 的 N-S 图见图 8.8(b)。



(a)



(b)

图 8.8

程序如下：

```
#include<stdio.h>
#define MAX 1000
main()
{char t[MAX];
 i=0;
 flag=0;
 flag1=1;
 printf("\nInput a hex number:");
 while((c=getchar())!='\0' && i<MAX&& flag1)
   {if (c>='0' && c<='9'||c>='a' && c<='f'||c>='A' && c<='F')
```

```
        {flag=1;
         t[i++]=c;
        }
        else if (flag)
             {t[i]='\0';
              printf("decimal  number %d:\n",htoi(t));
              printf("Continue or not?");
              c=getchar();
              if (c=='N'||c=='n')
                flag1=0;
              else
                {flag=0;
                 i=0;
                 printf("\nInput a hex number:");
                }
             }
      }
htoi(char s[])
{ int i,n;
  n=0;
  for (i=0;s[i]!='\0';i++)
    {if (s[i]>='0'&& s[i]<='9')
       n=n*16+s[i]-'0';
     if (s[i]>='a' && s[i]<='f')
       n=n*16+s[i]-'a'+10;
     if (s[i]>='A' && s[i]<='F')
       n=n*16+s[i]-'A'+10;
    }
    return(n);
}
```

运行结果：

```
Input a hex number:all
decimal  number:2577
Continue or not? y
Input a hex number:10
decimal  number:16
Continue or not? y
Input a hex number:f
decimal  number:15
Continue or not? n
```

8.17 用递归法将一个整数 n 转换成字符串。例如输入 483，应输出字符串“483”。n 的位数不确定，可以是任意位数的整数。

解： 主函数的 N-S 图见图 8.9。

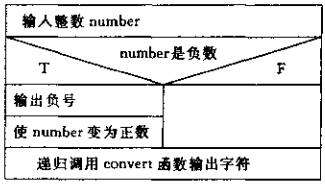


图 8.9

程序如下：

```
#include <stdio.h>
void convert(int n)
{int i;
 if ((i=n/10)!=0)
   convert(i);
 putchar(n%10+'0');
}
main()
{ int number;
  printf("\nInput an integer: ");
  scanf("%d",&number);
  printf("Output: ");
  if (number<0)
    {putchar('-');
     number=-number;
    }
  convert(number);
}
```

运行结果：

```
Input an integer:2345↙
Output:2345
```

8.18 给出年、月、日，计算该日是该年的第几天。

解： 主函数接收从键盘输入的日期，并调用 sum_day 和 leap 函数计算天数。其 N-S 图见图 8.10。sum_day 计算输入日期的天数。leap 函数返回是否闰年的信息。

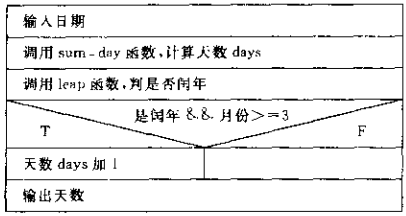


图 8.10

程序如下：

```
main()
{int year,month,day,days;
 printf("\nInput date(year,month,day):");
 scanf("%d,%d,%d",&year,&month,&day);
 printf("\n %d/%d/%d ",year,month,day);
 days=sum_day(month,day);                    /* 调用函数一 */
 if(leap(year) && month>=3)                  /* 调用函数二 */
   days=days+1;
 printf("is the %dth day in this year.\n",days);
}

int day_tab[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};  /* 外部数组 */
int sum_day(int month,int day)                 /* 函数一:计算日期 */
{int i;                                        /* 累加所在月之前的天数 */
 for (i=1;i<month;i++)
   day+=day_tab[i];
 return(day);
}

int leap(int year)                             /* 函数二:判断是否为闰年 */
{int leap;
 leap=year%4==0&&year%100!=0||year%400==0;
 return(leap);
}
```

运行结果：

```
Input date(year,month,day):2000,10,1↙
2000/10/1 is the 275th day in this year.
```

第9章　预处理命令

9.1　定义一个带参数的宏，使两个参数的值互换，并写出程序，输入两个数作为使用宏时的实参。输出已交换后的两个值。

解：

```
#define SWAP(a,b) t=b;b=a;a=t
main()
{int a,b,t;
printf("Input two integers a,b:");
scanf ("%d,%d," * a, * b);
SWAP(a,b);
printf("Now,a=%d,b=%d\n",a,b);
}
```

运行结果如下：

```
Input two integers a,b:3,4↙
Now,a=4,b=3
```

9.2　输入两个整数，求它们相除的余数。用带参的宏来实现，编程序。

解：

```
#define SURPLUS(a,b) a%b
main()
{int a,b;
 printf("Input two integers a,b:");
 scanf("%d,%d",&a,&b);
 printf("Remainder is %d\n",SURPLUS(a,b));
}
```

运行结果如下：

```
Input two integers a,b:60,13↙
Remainder is 8
```

9.3　三角形的面积为：

$$area = \sqrt{s(s-a)(s-b)(s-c)}$$

其中 $s=\frac{1}{2}(a+b+c)$，a、b、c 为三角形的三边。定义两个带参的宏，一个用来求 s，另一个用来求 $area$。写程序，在程序中用带实参的宏名来求面积 $area$。

解：

```
#include <math.h>
#define S(a,b,c)(a+b+c)/2
#define AREA(a,b,c) sqrt(S(a,b,c) * (S(a,b,c)-a) * (S(a,b,c)-b) * (S(a,b,c)-c))
main()
{float a,b,c;
 printf("Input a,b,c:");
 scanf("%f,%f,%f",&a,&b,&c);
 if (a+b>c && a+c>b && b+c>a)
    printf("area:%8.2f.\n",AREA(a,b,c));
 else
    printf("It is not a triangle!");
}
```

运行结果：

```
① Input a,b,c:3,4,5↙
    area:6.00
② Input a,b,c:12,3,5↙
    It is not a triangle!
```

9.4 给年份 year，定义一个宏，以判别该年份是否闰年。

提示：宏名可定义为 LEAP_YEAR，形参为 y，即定义宏的形式为：

```
#define LEAP_YEAR(y)(读者设计的字符串)
```

在程序中用以下语句输出结果：

```
if(LEAP_YEAR(year)) printf("%d is a Leap year",year);
else printf("%d is not a leap year",year);
```

解：

```
#define LEAP_YEAR(y) (y%4==0)&&(y%100!=0)||(y%400==0)
main()
{
  int year;
  printf("\nInput year:");
  scanf("%d",&year);
  if (LEAP_YEAR(year))
     printf("%d is a leap year.\n",year);
  else
     printf("%d is not a leap year.\n",year);
}
```

运行结果：

```
① Input year:1990↙
```

1990 is not a leap year.

② Input year:2000↙

2000 is a leap year.

9.5 请分析以下一组宏所定义的输出格式：

```
#define NL putchar('\n')
#define PR(format,value) printf("value=%format\t",(value))
#define PRINT1(f,x1) PR(f,x1);NL
#define PRINT2(f,x1,x2) PR(f,x1);PRINT1(f,x2)
```

如果在程序中有以下的宏引用：

```
PR(d,x);
PRINT1(d,x);
PRINT2(d,x1,x2);
```

写出宏展开后的情况，并写出应输出的结果，设 x=5、x1=3、x2=8。

解：展开后为：

```
printf("value=%format\t",x);
printf("value=%format\t",x);putchar('\n');
printf("value=%format\t",x1);printf("value=%format\t",x2);putchar('\n');
```

如果运行以下程序：

```
#include <stdio.h>
#define NL putchar('\n')
#define PR(format,value) printf("value=%format\t",(value))
#define PRINT1(f,x1) PR(f,x1);NL
#define PRINT2(f,x1,x2) PR(f,x1);PRINT1(f,x2)
main()
{ float x=5.0,x1=3.0,x2=8.0;
  char d;
  PR(d,x);
  PRINT1(d,x);
  PRINT2(d,x1,x2);
}
```

输出结果如下：

```
value=5.000000ormat    value=5.000000ormat
value=3.000000ormat    value=8.000000ormat
```

通过本习题可以看到：如果用 Turbo C，则不能用这种方法将输出格式和输出项都作为参数。在宏替换时对字符串中的字符不予替换，一律保留原状。

9.6 请设计输出实数的格式，包括：(1) 一行输出一个实数；(2) 一行输出两个实数；(3) 一行输出 3 个实数。实数用"6.2f"格式输出。

解：

```
#define PR printf
#define NL "\n"
#define Fs "%f"
#define F "%6.2f"
#define F1 F NL
#define F2 F "\t" F NL
#define F3 F "\t" F "\t" F NL
main()
{float a,b,c;
 PR("Input three floating numbers a,b,c:\n");
 scanf(Fs,&a);
 scanf(Fs,&b);
 scanf(Fs,&c);
 FR(NL);
 PR("Output one floating number each line:\n");
 PR(F1,a);
 PR(F1,b);
 PR(F1,c);
 PR(NL);
 PR("Output two floating numbers:\n");
 PR(F2,a,b);
 PR(F1,c);
 PR(NL);
 PR("Output three floating numbers:\n");
 PR(F3,a,b,c);
}
```

运行情况如下：

```
Input three floating numbers a,b,c:
2.4  5.9  9.1↙
Output one floating number each line:
  2.40
  5.90
  9.10
Output two floating numbers:
  2.40  5.90
  9.10
Output three floating numbers:
  2.40  5.90  9.10
```

9.7 设计所需的各种各样的输出格式(包括整数、实数、字符串等)，用一个文件名"for-

mat. h",把这些信息都放到此文件内,另编一个程序文件,用 # include"format. h"命令以确保能使用这些格式。

解:

```
/ * format. h 文件 * /
# define INTEGER(d) printf("%d\n",d)        / * 输出整数 * /
# define FLOAT(f) printf("%8. 2f\n",f)      / * 输出实数 * /
# define STRING(s) printf("%s\n",s)         / * 输出字符串 * /
/ * 以下为用户自己编写的程序,其中需要用到"format. h"文件中的输出格式 * /
# include "format. h"
main()
{int d,num;
 float f;
 char s[80];
 printf("Choice data format:1—integer,2—float,3—string:");
 scanf("%d",&num);
 switch(num)
 {case 1 : printf("Input integer:");
           scanf("%d",&d);
           INTEGER(d);
           break;
  case 2: printf("Input float:");
          scanf("%f",&f);
          FLOAT(f);
          break;
  case 3: printf("Input string:");
          scanf("%s",&s);
          STRING(s);
          break;
  default: printf("input error!");
 }
}
```

运行结果如下:

```
① Choice data format:1—integer,2—float,3—string:1↙
  Input integer:20↙
  20              (输出整数 20)
② Choice data format:1—integer,2—float,3—string:2↙
  Input float:3.79↙
  3.79            (输出实数 3.79)
③ Choice data format:1—integer,2—float,3—string:3↙
  Input string:student↙
```

```
   student        (输出字符串student)
④ Choice data format:1--integer,2—float,3—string:4↙
   Input error!
```

本题参考解答只是示意性的，表明如何利用"format.h"文件中的输出格式。读者应在此基础上编出能供实际使用的各种各样的输出格式(例如，一行中输出一个、两个或若干个实数，有的使用%6.2f格式，有的使用%10.2f格式，等等)。

9.8 分别用函数和带参的宏，从3个数中找出最大数。

解：(1) 用函数实现

```
main()
{
 int a,b,c;
 printf("Enter three integers:");
 scanf("%d,%d,%d",&a,&b,&c);
 printf("max   %d\n",max(a,b,c));
}

max(int x,int y,int z)
{int t;
 t=(x>y? x:y);
 return(t>z? t:z);
}
```

运行结果：

```
Input three integers:12,34,9↙
max=34
```

(2) 用带参的宏实现

```
#define MAX(a,b) ((a)>(b)? (a):(b))
main()
{
 int a,b,c;
 printf("Input three integers:");
 scanf("%d,%d,%d",&a,&b,&c);
 printf("max=%d\n",MAX(MAX(a,b),c));
}
```

运行结果：

```
Input three integers:12,34,9↙
max=34
```

9.9 试述"文件包含"和程序文件的连接(link)的概念，二者有何不同？

解："文件包含"是事先将程序中需要用到的信息分别存放在不同的"头文件"中(文

件后缀为.h),用户在编写程序时,利用#include命令将该头文件的内容包含进来,成为程序中的一部分。特别应当注意的是,该头文件与它所在的源文件共同组成一个文件模块(而不是两个文件模块)。在编译时它是作为一个文件进行编译的。

连接则与此不同,它的作用是将多个目标文件连接起来。如果有两个或多个源程序文件,应先对它们分别进行编译,得到两个或多个目标文件(后缀为.obj),在连接阶段,把这些目标文件与系统提供的函数库等文件连接成一个可执行的文件(后缀为.exe)。

9.10 用条件编译方法实现以下功能:

输入一行电报文字,可以任选两种输出:一为原文输出;一为将字母变成其下一字母(如'a'变成'b'……'z'变成'a'。其他字符不变)。用#define命令来控制是否要译成密码。例如:

#define CHANGE 1

则输出密码。若

#define CHANGE 0

则不译成密码,按原码输出。

解:

```
#include "stdio.h"
#define MAX 80
#define CHANGE 1
main()
{
  char str[MAX];
  int i;
  printf("Input text:\n");
  gets(str);
  #if (CHANGE)
    {for (i=0;i<MAX;i++)
      {if (str[i]!='\0')
        if (str[i]>='a'&& str[i]<'z' ||str[i]='A'&& str[i]='Z')
          str[i]+=1;
        else if (str[i]='z' str[i]=='Z')
          str[i]-=25;
      }
    }
  #endif
  printf("Output:\n%s",str);
}
```

运行结果:

```
Input text:
```

A Lazy Brown Fox Jumps Over A Dog. ↙

Output:

B Mbaz Cspxo Gpy Kvnqt Pwfs B Eph.

如果想输出原文，可以将程序第 3 行改为：

#define CHANGE 0

修改后再运行：

Input text:

A Lazy Brown Fox Jumps Over A Dog. ↙

Output:

A Lazy Brown Fox Jumps Over A Dog.

第10章 指　　针

10.1 输入3个整数，按由小到大的顺序输出。

解：

```
main()
{int n1,n2,n3;
 int *p1,*p2,*p3;
 printf("Input three integers n1,n2,n3:");
 scanf("%d,%d,%d",&n1,&n2,&n3);
 p1=&n1;
 p2=&n2;
 p3=&n3;
 if(n1>n2) swap(p1,p2);
 if(n1>n3) swap(p1,p3);
 if(n2>n3) swap(p2,p3);
 printf("Now,the order is:%d,%d,%d\n",n1,n2,n3);
}
swap(int *p1,int *p2)
{int p;
 p=*p1; *p1=*p2; *p2=p;
}
```

运行结果：

```
Input three integers n1,n2,n3:34,21,25↙
Now,the order is:21,25,34
```

10.2 输入3个字符串，按由小到大的顺序输出。

解：

```
main()
{char *str1[20],*str2[20],*str3[20];
 char swap();
 printf("Input three lines:\n");
 gets(str1);
 gets(str2);
 gets(str3);
```

```
 if(strcmp(str1,str2)>0)swap(str1,str2);
 if(strcmp(str1,str3)>0)swap(str1,str3);
 if(strcmp(str2,str3)>0)swap(str2,str3);
 printf("Now,the order is:\n");
 printf("%s\n%s\n%s\n",str1,str2,str3);
}
char swap(char *p1,char *p2)          /* 交换两个字符串 */
{char p[20];
 strcpy(p,p1);strcpy(p1,p2);strcpy(p2,p);
}
```

运行结果：

```
Input three lines:
I study very hard.↙
C language is very interesting.↙
He is a professfor.↙
Now,the order is:
C language is very interesting.
He is a professfor.
I study very hard.
```

10.3 输入 10 个整数，将其中最小的数与第一个数对换，把最大的数与最后一个数对换。写 3 个函数：(1) 输入 10 个数；(2) 进行处理；(3) 输出 10 个数。

解： 输入输出函数的 N-S 图见图 10.1。交换函数的 N-S 图见图 10.2。

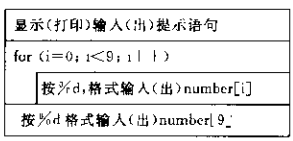


图 10.1

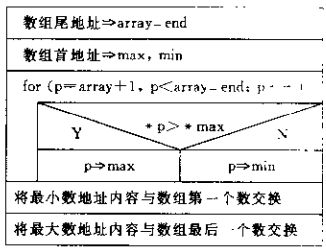


图 10.2

```
main()
{int number[10];
 input(number);                    /* 调用输入 10 个数的函数 */
 max_min_value(number);            /* 调用交换函数 */
 output(number);                   /* 调用输出函数 */
}
```

```
input(int number[10])                          /* 输入 10 个数的函数 */
{int i;
 printf("Input 10 numbers:");
 for (i=0;i<10;i++)
   scanf("%d",&number[i]);
}
max_min_value(int array[10])                   /* 交换函数 */
{ int *max,*min,*p,*array_end;
 array_end=array+10;
 max=min=array;
 for (p=array+1;p<array_end;p++)
   if (*p>*max) max=p;                         /* 将大数地址赋给 max */
   else if (*p<*min) min=p;                    /* 将小数地址赋给 min */
 *p=array[0];array[0]=*min;*min=*p;            /* 将最小数与第一数交换 */
 *p=array[9];array[9]=*max;*max=*p;            /* 将最小数与第十数交换 */
 return;
}
output(int array[10])                          /* 输出函数 */
{int *p;
 printf("Now,they are:");
 for (p=array;p<=array+9;p++)
   printf("%d ",*p);
}
```

运行结果：

```
Input 10 numbers:32 24 56 78 1 98 36 44 29 6↙
Now,they are: 1 24 56 78 32 6 36 44 29 98
```

10.4 有 n 个整数，使其前面各数顺序向后移 m 个位置，最后 m 个数变成最前面 m 个数，见图 10.3。写一函数实现以上功能，在主函数中输入 n 个整数，并输出调整后的 n 个数。

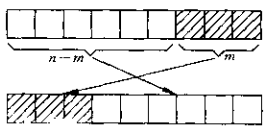


图 10.3

解：

```
main()
```

```
{int number[20],n,m,i;
 printf("How many numbers?");                    /* 共有多少个数 */
 scanf("%d",&n);
 printf("Input %d numbers:\n",n);                /* 输入 n 个数 */
 for (i=0;i<n;i++)
 scanf("%d",&number[i]);
 printf("How many place you want to move?");     /* 后移多少个位置 */
 scanf("%d",&m);
 move(number,n,m);                               /* 调用 move 函数 */
 printf("Now,they are:\n");
 for (i=0;i<n;i++)
     printf("%d",number[i]);
}
move(int array[20],int n,int m)                  /* 循环后移一次的函数 */
{int *p,array_end;
 array_end=*(array+n-1);
 for (p=array+n-1;p>array;p--)
     *p=*(p-1);
 *array=array_end;
 m--;
 if (m>0) move(array,n,m);                       /* 递归调用,当循环次数 m 减至 0 */
                                                 /* 时,停止调用 */
}
```

运行结果：

```
How many numbers? 8↙
Input 8 numbers:
12 43 65 67 8 2 7 11↙
How many place you want to move? 4↙
Now,they are:
8  2  7  11  12  43  65  67
```

10.5 有 n 个人围成一圈，顺序排号。从第一个人开始报数(从 1 到 3 报数)，凡报到 3 的人退出圈子，问最后留下的是原来第几号的那位。

解：N-S 图如图 10.4 所示。

```
main()
{int i,k,m,n,num[50],*p;
 printf("Input number of person: n=");
 scanf("%d",&n);
```

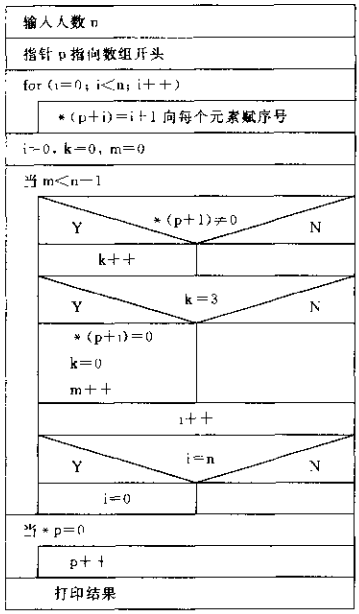


图 10.4

```
p=num;
for (i=0;i<n;i++)
   *(p+i)=i+1;              /* 以 1 至 n 为序给每个人编号 */
i=0;                        /* i 为每次循环时的计数变量 */
k=0;                        /* k 为按 1、2、3 报数时的计数变量 */
m=0;                        /* m 为退出人数 */
while (m<n-1)               /* 当退出人数比 n-1 少时(即未退出人数大于 1 时)执行
                               循环体 */
  {if (*(p+i)!=0) k++;
   if (k==3)                /* 对退出的人的编号置为 0 */
     {*(p+i)=0;
      k=0;
      m++;
     }
   i++;
   if (i==n) i=0;           /* 报数到尾后,i 恢复为 0 */
  }
while(*p==0) p++;
printf("The last one is NO. %d\n",*p);
```

运行结果：

```
Input number of person: n=8↙
The last one is NO.7                    (最后留在圈子内的是 7 号)
```

10.6 写一个函数，求一个字符串的长度。在 main 函数中输入字符串，并输出其长度。

解：

```
main()
{int len;
 char *str[20];
 printf("Input string:");
 scanf("%s",str);
 len=length(str);
 printf("The length of string is %d.",len);
}
length(char *p)                    /*求字符串长度函数*/
{int n;
 n=0;
 while (*p!='\0')
   {n++;
    p++;
   }
 return(n);
}
```

运行结果：

```
Input string:China↙
The length of string is 5.
```

10.7 有一字符串，包含 n 个字符。写一个函数，将此字符串中从第 m 个字符开始的全部字符复制成为另一个字符串。

解：

```
main()
{int m;
 char *str1[20],*str2[20];
 printf("input string:");
 gets(str1);
 printf("Which character that begin to copy?");
 scanf("%d",&m);
 if (strlen(str1)<m)
   printf("input error!");
```

```
  else
    {copystr(str1,str2,m);
     printf("result: %s",str2);
    }
}
copystr(char * p1,char * p2,int m)              /* 字符串部分复制函数 */
{int n;
 n=0;
 while (n<m-1)
   {n++;
    p1++;
   }
 while ( * p1! ='\0')
   { * p2= * p1;
    p1++;
    p2++;
   }
 * p2='\0';
}
```

运行结果：

```
Input string: reading-room↙
Which character,that begin to copy? 9↙
result: room
```

10.8 输入一行文字，找出其中大写字母、小写字母、空格、数字及其他字符各有多少。

解：

```
#include <stdio.h>
main()
{int upper=0,lower=0,digit=0,space=0,other=0,i=0;
 char * p,s[20];
 printf("Input string:");
 while ((s[i]=getchar())!='\n') i++;
 p=&s[0];
 while ( * p! ='\n')
   {if (('A'<= * p) && ( * p<='Z'))
       upper++;
    else if (('a'<= * p) && ( * p<='z'))
       lower++;
    else if ( * p ==' ')
       space++;
    else if (( * p<='9') && ( * p>='0'))
```

```
            ++digit;
        else
            ++other;
        p++;
    }
    printf("upper case:%d lower case:%d",upper,lower);
    printf("space:%d digit:%d other:%d\n",space,digit,other);
}
```

运行结果：

```
Input string:Today is 2000/1/1↙
upper case:1    lower case:6    space:2    digit:6    other:2
```

10.9 写一个函数，将一个 3×3 的矩阵转置。

解：

```
main()
{int a[3][3],*p,i;
 printf("Input matrix:\n");
 for (i=0;i<3;i++)
   scanf("%d %d %d",&a[i][0],&a[i][1],&a[i][2]);
 p=&a[0][0];
 move(p);
 printf("Now,matrix:\n");
 for (i=0;i<3;i++)
 printf("%d %d %d\n",a[i][0],a[i][1],a[i][2]);
}
move(int *pointer)
{int i,j,t;
 for (i=0;i<3;i++)
   for (j=i;j<3;j++)
     {t=*(pointer+3*i+j);
      *(pointer+3*i+j)=*(pointer+3*j+i);
      *(pointer+3*j+i)=t;
     }
}
```

运行结果：

```
Input matrix:
1 2 3↙
4 5 6↙
7 8 9↙
Now,matrix:
```

1 4 7
2 5 8
3 6 9

10.10 将一个 5×5 的矩阵中最大的元素放在中心，4 个角分别放 4 个最小的元素(按从左到右、从上到下的顺序，依次从小到大存放)，写一个函数实现之，并用 main 函数调用。

解：

```
main()
{int a[5][5], *p,i,j;
 printf("Input matrix:\n");
 for (i=0;i<5;i++)                          /* 输入矩阵 */
   for (j=0;j<5;j++)
     scanf("%d",&a[i][j]);
 p=&a[0][0];                                /* 使 p 指向 0 行 0 列元素 */
 change(p);                                 /* 调用函数,实现交换 */
 printf("Now,matrix:\n");
 for (i=0;i<5;i++)                          /* 输出已交换的矩阵 */
   {for (j=0;j<5;j++)
      printf("%d ",a[i][j]);
    printf("\n");
   }
}
change(int *p)                              /*交换函数*/
{int i,j,temp;
 int *pmax,*pmin;
 pmax=p;
 pmin=p;
 for (i=0;i<5;i++)          /*找最大值和最小值的地址,并赋给 pmax,pmin */
   for (j=i;j<5;j++)
     {if (*pmax<*(p+5*i+j)) pmax=p+5*i+j;
      if (*pmin>*(p+5*i+j)) pmin=p+5*i+j;
     }
 temp=*(p+12);                              /*将最大值换给中心元素*/
 *(p+12)=*pmax;
 *pmax=temp;
 temp=*p;                                   /*将最小值换给左上角元素*
 *p=*pmin;
 *pmin=temp;
 pmin=p+1;
```

```
  for (i=0;i<5;i++)                                    /* 找第二最小值的地址赋给 pmin */
    for (j=0;j<5;j++)
      if (((p+5*i+j)!=p) && (*pmin > *(p+5*i+j))) pmin=p+5*i+j;
  temp=*pmin;                                          /* 将第二最小值换给右上角元素 */
  *pmin=*(p+4);
  *(p+4)=temp;
  pmin=p+1;
  for (i=0;i<5;i++)                                    /* 找第三最小值的地址赋给 pmin */
    for (j=0;j<5;j++)
      if (((p+5*i+j)!=(p+4)) && ((p+5*i+j)!=p) &&
          (*pmin>*(p+5*i+j)))pmin=p+5*i+j;
  temp=*pmin;                                          /* 将第三最小值换给左下角元素 */
  *pmin=*(p+20);
  *(p+20)=temp;
  pmin=p+1;
  for (i=0;i<5;i++)                                    /* 找第四最小值的地址赋给 pmin */
    for (j=0;j<5;j++)
      if (((p+5*i+j)!=p) && ((p+5*i+j)!=(p+4)) && ((p+5*i+j)!
          (p+20)) && (*pmin>*(p+5*i+j)))  pmin=p+5*i+j;
  temp=*pmin;                                          /*将第四最小值换给右下角元素*/
  *pmin=*(p+24);
  *(p+24)=temp;
}
```

运行结果：

```
Input matrix:
35 34 33 32 31↙
30 29 28 27 26↙
25 24 23 22 21↙
15 14 13 12 11↙
Now, matrix:
11 34 33 32 12
30 29 28 27 26
25 24 35 22 21
20 19 18 17 16
13 23 15 31 14
```

10.11 在主函数中输入 10 个等长的字符串。用另一个函数对它们排序，然后在主函数输出这 10 个已排好序的字符串。

解：

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

```
main()
{void sort(char s[][]);
 int i;
 char str[10][6];
 printf("Input 10 strings:\n");
 for (i=0;i<10;i++)
   scanf("%s",str[i]);
 sort(str);
 printf("Now,the sequence is:\n");
 for (i=0;i<10;i++)
   printf("%s\n",str[i]);
}
void sort(char s[10][6])
  {int i,j;
  char *p,temp[10];
  p=temp;
  for (i=0;i<9;i++)
    for (j=0;j<9-i;j++)
      if (strcmp(s[j],s[j+1])>0)
        {strcpy(p,s[j]);
         strcpy(s[j],s[j+1]);
         strcpy(s[j+1],p);
        }
}
```

运行结果：

```
Input 10 strings:
China↙
Japan↙
Korea↙
Egypt↙
Nepal↙
Burma↙
Ghana↙
Sudan↙
Italy↙
Libya↙
Now,the sequence is:
Burma
China
Egypt
Ghana
```

Italy
Japan
Korea
Libya
Nepal
Sudan

10.12 用指针数组处理上一题目，字符串不等长。

解：

```
main()
{int i;
 char *p[10],str[10][20];
 for (i=0;i<10;i++)
   p[i]=str[i];          /*将第i个字符串的首地址赋予指针数组p的第i个元素*/
 printf("Input 10 strings:\n");
 for (i=0;i<10;i++)
   scanf("%s",p[i]);
 sotr(p);
 printf("Now,the sequence is:\n");
 for (i=0;i<10;i++)
   printf("%s\n",p[i]);
}
sotr(char *p[])
{int i,j;
 char *temp;
 for (i=0;i<9;i++)
   for (j=0;j<9-i;j++)
     if (strcmp(*(p+j),*(p+j+1))>0)
       {temp=*(p+j);
        *(p+j)=*(p+j+1);
        *(p+j+1)=temp;
       }
}
```

运行情况如下：

```
Input 10 strings:
China↙
Japan↙
Yemen↙
Pakistan↙
Mexico↙
Korea↙
```

```
Brazil↙
Iceland↙
Canada↙
Mongolia↙
Now,the sequence is:
Brazil
Canada
China
Iceland
Japan
Korea
Mexico
Mongolia
Pakistan
Yemen
```

10.13 写一个用矩形法求定积分的通用函数，分别求：

$$\int_0^1 \sin x\,dx,\quad \int_{-1}^1 \cos x\,dx,\quad \int_0^2 e^x\,dx$$

（说明：sin、cos、exp 已在系统的数学函数库中，程序开头要用 #include <math.h>）。

解：

```
#include <math.h>
main()
{float integral(float (*p)(float),float a,float b,int n);
 float a1,b1,a2,b2,a3,b3,c,(*p)(float);
 float fsin(float);                    /* 声明 fsin 函数 */
 float fcos(float);                    /* 声明 fcos 函数 */
 float fexp(float);                    /* 声明 fexp 函数 */
 int n=20;
 printf("Input a1,b1:");               /* 输入求 sin x 定积分的下限和上限 */
 scanf("%f,%f",&a1,&b1);
 printf("Input a2,b2:");               /* 输入求 cos x 定积分的下限和上限 */
 scanf("%f,%f",&a2,&b2);
 printf("Input a3,b3:");               /* 输入求 e^x 定积分的下限和上限 */
 scanf("%f,%f",&a3,&b3);
 p=fsin;
 c=integral(p,a1,b1,n);                /* 求出 sin x 的定积分 */
 printf("The integral of sin(x) is :%f\n",c);
 p=fcos;
 c=integral(p,a2,b2,n);                /* 求出 cos x 的定积分 */
```

```
 printf("The integral of cos(x) is :%f\n",c);
 p=fexp;
 c=integral(p,a3,b3,n);              /* 求出 e^x 的定积分 */
 printf("The integral of exp(x) is :%f\n",c);
}
float integral(float (*p)(float),float a,float b,int n)
                                     /*用矩形法求定积分的通用函数 */
{int i;
 float x,h,s;
 h=(b-a)/n;
 x=a;
 s=0;
 for (i=1;i<=n;i++)
   {x=x+h;
    s=s+(*p)(x)*h;
   }
 return(s);
}
float fsin(float x)                  /* 计算 sin x 的函数 */
  {return sin(x);}
float fcos(float x)                  /* 计算 cos x 的函数 */
  {return cos(x);}
float fexp(float x)                  /* 计算 e^x 的函数 */
  {return exp(x);}
```

运行情况如下：

```
Input a1,b1:0,1↙
Input a2,b2:-1,1↙
Input a3,b3:0,2↙
The integral of sin(x) is : 0.480639
The integral of cos(x) is :1.681539
The integral of exp(x) is : 6.713833
```

说明：sin、cos 和 exp 是系统提供的数学函数，在程序中定义 3 个函数 fsin、fcos 和 fexp 分别用来计算 sin(x)、cos(x)和 exp(x)的值，在 main 函数中要声明这 3 个函数。在 main 函数中定义 p 为指向函数的指针变量，定义形式是"float (*p)(float)"，表示 p 指向的函数有一个实型形参，p 可指向返回值为实型的函数。在 main 函数中有"p=fsin;"，表示将 fsin 函数的入口地址赋给 p。在调用 integral 函数时，用 p 作为实参，把 fsin 函数的入口地址传递给形参 p(形参 p 也定义为指向函数的指针变量)，这样形参 p 也指向 fsin 函数，(*p)(x)就相当于 fsin(x)。fsin(x)的值就是 sin x 的值。因此通过调用 integral 函数求出了 sin x

的定积分。求其余两个函数定积分的情况与此类似。

10.14 将 n 个数按输入顺序的逆序排列，用函数实现。

解：

```
main()
{int i,n;
 char *p,num[20];
 printf("input n:");
 scanf("%d",&n);
 printf("please input these numbers:\n");
 for (i=0;i<n;i++)
   scanf("%d",&num[i]);
 p=&num[0];
 sort(p,n);
 printf("Now,the sequence is:\n");
 for (i=0;i<n;i++)
   printf("%d ",num[i]);
}

sort (char p,int m)                    /*将 n 个数逆序排列函数 */
{int i;
 char temp, *p1, *p2;
 for (i=0;i<m/2;i++)
   {p1=p+i;
    p2=p+(m-1-i);
    temp=*p1;
    *p1=*p2;
    *p2=temp;
   }
}
```

运行结果：

```
input n:10↙
please input these numbers:
10 9 8 7 6 5 4 3 2 1↙
Now,the sequence is:
1 2 3 4 5 6 7 8 9 10
```

10.15 有一个班 4 个学生，5 门课。(1) 求第一门课的平均分；(2) 找出有两门以上课程不及格的学生，输出他们的学号和全部课程成绩及平均成绩；(3) 找出平均成绩在 90 分以上或全部课程成绩在 85 分以上的学生。分别编 3 个函数实现以上 3

个要求。

解：

```
main()
{
 int i,j, * pnum,num[4];
 float score[4][5],aver[4], * psco, * pave;
 char course[5][10], * pcou;
 printf("Input course:\n");
 pcou=course[0];
 for (i=0;i<5;i++)
   scanf("%s",course[i]);
 printf("Input NO. and scores:\n");
 printf("NO.");
 for (i=0;i<5;i++)
   printf(",%s",course[i]);
 printf("\n");
 psco=&score[0][0];
 pnum=&num[0];
 for (i=0;i<4;i++)
   {scanf("%d",pnum+i);
    for (j=0;j<5;j++)
      scanf(",%f",psco+5 * i+j);
   }
 pave=&aver[0];
 printf("\n\n");
 avsco(psco,pave);                    /* 求出每个学生的平均成绩 */
 avcour1(pcou,psco);                  /* 求出第一门课的平均成绩 */
 printf("\n\n");
 fali2(pcou,pnum,psco,pave);          /* 找出 2 门课不及格的学生 */
 printf("\n\n");
 good(pcou,pnum,psco,pave);           /* 找出成绩好的学生 */
}
avsco(float * psco,float * pave)      /* 求每个学生的平均成绩的函数 */
{int i,j;
 float sum,average;
 for (i=0;i<4;i++)
   {sum=0.0;
    for (j=0;j<5;j++)
      sum=sum+(*(psco+5 * i+j));      /* 累计每个学生的各科成绩 */
```

```
        average=sum/5;                          /*计算平均成绩*/
        *(pave+i)=average;
    }
}
avcour1(char *pcou,float *psco)                 /* 求第一门课的平均成绩的函数 */
{int i;
 float sum,average1;
 sum=0.0;
 for (i=0;i<4;i++)
   sum=sum+(*(psco+5*i));                       /*累计每个学生的得分*/
 average1=sum/4;                                /*计算平均成绩*/
 printf("course 1: %s ,average score:%6.2f.\n",pcou,average1);
}

fali2(char course[5][10],int num[],float score[4][5],float aver[4])
                        /* 找两门以上课程不及格的学生的函数 */
{int i,j,k,label;
 printf("==========Student who is fail ========\n");
 printf("NO.");
 for (i=0;i<5;i++)
   printf("%10s",course[i]);
 printf("average\n");
 for (i=0;i<4;i++)
   {label=0;
    for (j=0;j<5;j++)
      if ((score[i][j])<60.0) label++;
    if (label>=2)
      {printf("%5d",num[i]);
       for (k=0;k<5;k++)
         printf("%10.2f",score[i][k]);
       printf("%10.2f\n",aver[i]);
      }
   }
}
good(char course[5][10],int num[4],float score[4][5],float aver[4])
          /* 找成绩优秀的学生(各门85分以上或平均90分以上)的函数 */
{int i,j,k,n;
 printf("=====Student whose score is good======\n");
 printf("NO.");
```

```
  for (i=0;i<5;i++)
    printf("%10s",course[i]);
  printf("average\n");
  for (i=0;i<4;i++)
    {n=0;
     for (j=0;j<5;j++)
       if ((score[i][j])>85.0) n++;
     if ((n==5)||(aver[i]>=90))
       {printf("%5d",num[i]);
        for (k=0;k<5;k++)
          printf("%10.2f",score[i][k]);
        printf("%10.2f\n",aver[i]);
       }
    }
}
```

运行情况如下：

```
Input course:                          (输入课程名称)
English↙
Computer↙
Math↙
Physics↙
Chemistry↙
Input NO. and scores:                  (输入学号和各门课成绩)
NO.,English,Computer,Math,Physics,Chemistry,average  (按此顺序输入)
101,34,56,88,99,89↙
102,77,88,99,67,78↙
103,99,90,87,86,89↙
104,78,89,99,56,77↙
course 1: English ,average score : 72.00.    (第一门课英语的平均成绩)
==========Student who is fail ========(有两门课不及格者)
NO.     English    Computer    Math    Physics    Chemistry    average
101      34.00      56.00      88.00    99.00      89.00       73.20

======Student whose score is good======          (成绩优良者)
NO.     English    Computer    Math    Physics    Chemistry    average
103      99.00      90.00      87.00    86.00      89.00       90.20
```

程序中 num 是存放 4 个学生学号的一维数组，course 是存放 5 门课名称的二维字符数组，score 是存放 4 个学生 5 门课成绩的二维数组，aver 是存放每个学

生平均成绩的数组，pnum 是指向 num 数组的指针变量，pcou 是指向 course 数组的指针变量，psco 是指向 score 数组的指针变量，pave 是指向 aver 数组的指针变量，见图 10.5。

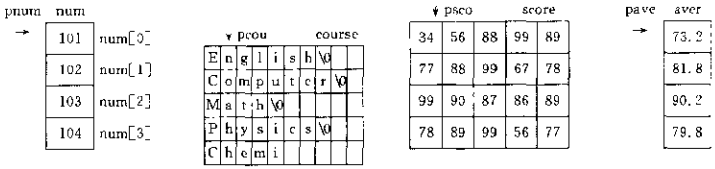


图 10.5

函数的形参用数组，调用函数时的实参用指针变量。形参也可以不用数组而用指针变量，请读者自己分析。

10.16 输入一个字符串，内有数字和非数字字符，如：

a123x456␣17960? 302tab5876

将其中连续的数字作为一个整数，依次存放到一数组 a 中。例如 123 放在 a[0]中，456 放在 a[1]中……统计共有多少个整数，并输出这些数。

解：

```
#include <stdio.h>
main()
{
 char str[50], *pstr;
 int i,j,k,m,e10,digit,ndigit,a[10], *pa;
 printf("Input a string:\n");
 gets(str);
 printf("\n");
 pstr=&str[0];          /* 字符指针 pstr 置于数组 str 首地址 */
 pa=&a[0];              /* 指针 pa 置于 a 数组首地址 */
 ndigit=0;              /* ndigit 代表有多少个整数 */
 i=0;                   /* 代表字符串中字符的位置 */
 j=0;                   /* 代表连续数字的位数 */
 while(*(pstr+i)!='\0')
   {if((*(pstr+i)>='0') && (*(pstr+i)<='9'))
      j++;
    else
      {if (j>0)
         {digit=*(pstr+i-1)-48;          /*将个数位赋予 digit*/
          k=1;
          while (k<j)          /*将含有两位以上数的其他位的数值累计于 digit*
            {e10=1;
```

```
            for (m=1;m<=k;m++)
              e10=e10*10;              /*e10代表该位数所应乘的因子*
            digit=digit+(*(pstr+i-1-k)-48)*e10;    /*将该位数的数值
                                                    累加子digit*
            k++;                       /*位数k自增*/
          }
          *pa=digit;                   /*将数值赋予数组a*/
          ndigit++;
          pa++;                        /*指针pa指向a数组下一元素*/
          j=0;
        }
      }
      i++;
    }
  if (j>0)                             /*以数字结尾字符串的最后一个数据*
    {digit=*(pstr+i-1)-48;             /*将个数位赋予digit*/
     k=1;
     while (k<j)                       /*将含有两位以上数的其他位的数值累加子digit*
       {e10=1;
        for (m=1;m<=k;m++)
          e10=e10*10;                  /*e10代表位数所应乘的因子*/
        digit=digit+(*(pstr+i-1-k)-48)*e10;  /*将该位数值累加子digit*
        k--;                           /*位数k自增*/
       }
     *pa=digit;                        /*将数值赋予数组a*/
     ndigit++;
     j=0;
    }
  printf("There are %d numbers in this line. They are:\n",ndigit);
  j=0;
  pa=&a[0];
  for (j=0;j<ndigit;j++)               /*打印数据*/
    printf("%d ",*(pa+j));
  printf("\n");
}
```

运行情况如下：

```
Input a string:
a123x456 17960? 302tab5876↙
There are 6 numbers in this line. They are:
123 456 17960  302  5876
```

10.17 写一函数，实现两个字符串的比较。即自己写一个 strcmp 函数，函数原型为：

int strcmp(char *p1,char *p2)

设 p1 指向字符串 s1,p2 指向字符串 s2。要求：当 s1=s2 时,返回值为 0。当 s1≠s2 时,返回它们二者的第一个不同字符的 ASCII 码差值(如"BOY"与"BAD",第二个字母不同,"O"与"A"之差为 79-65=14);如果 s1>s2,则输出正值;如果 s1<s2,则输出负值。

解:

```
main()
{int m;
 char str1[20],str2[20],*p1,*p2;
 printf("Input two strings:\n");
 scanf("%s",str1);
 scanf("%s",str2);
 p1=&str1[0];
 p2=&str2[0];
 m=strcmp(p1,p2);
 printf("result:%d,\n",m);
}

strcmp(char *p1,char *p2)          /* 两个字符串比较的函数 */
{int i;
 i=0;
 while(*(p1+i)==*(p2+i))
   if(*(p1+i++)=='\0') return(0);          /* 相等时返回结果 0 */
 return(*(p1+i)-*(p2+i));     /* 不等时返回结果为第一个不等字符 ASCII 码的差
                                 值 */
}
```

运行情况如下：

① Input two strings:
CHINA↙
Chen↙
result:-32
② Input two strings:
hello! ↙
hello! ↙
result:0
③ Input two strings:
dog↙
cat↙
result:1

10.18 编一个程序,打入月份号,输出该月的英文月名。例如,输入"3",则输出

"March",要求用指针数组处理。

解:

```
main()
{char * month_ name[13]={"illegal month","January","February","March","April",
   "May","June","July","August","September","October", "November","December"};
 int n;
 printf("Input month:\n");
 scanf("%d",&n);
 if ((n<=12) && (n>=1))
   printf("It is %s.", * (month_ name+n));
 else
   printf("It is wrong.");
}
```

运行结果:

```
① Input month:2
It is February.
② Input month:8
It is August.
③ Input month:13
It is wrong.
```

10.19 编写一个函数 alloc(n),用来在内存区新开辟一个连续的空间(n 个字节)。此函数的返回值是一个指针,指向新开辟的连续空间的起始地址。再写一个函数 free(p),将地址 p 开始的各单元释放(不能再被程序使用,除非再度开辟)。

提示:先在内存规定出一片相当大的连续空间(例如 1000 个字节)。然后开辟与释放都在此空间内进行。假设指针变量 p 原已指向未用空间的开头,调用 alloc(n)后,开辟了 n 个字节可供程序使用(例如,可以赋值到这些单元中)。现在需要使 p 的值变成 p+n,表示空白未用区从 p+n 地址开始,同时要将新开辟区的起始位置(p)作为函数值返回,以表示可以利用从此点开始的单元。如果更新开辟的区太大(n 大),超过了预设的空间(1000 字符),则 alloc(n)函数返回指针 NULL,表示开辟失败。

alloc(n)应返回一个指向字符数据的指针(因为开辟的区间是以字节为单位被利用的)。

解:

```
#define NULL 0                          /* 当开辟失败时返回标志 */
#define ALLOCSIZE 1000                  /* 可以开辟的最大空间 */
char allocbuf[ALLOCSIZE];               /* 开辟一个字符数组,作为存储区 */
char * allocp=allocbuf;                 /* 指针指向存储区的始端 */
char * alloc(int n)                     /* 开辟存储区函数,开辟存储区后返回指针 *
```

```
{if (allocp+n<=allocbuf+ALLOCSIZE)
   {allocp+=n;
    return(allocp-n);                    /* 返回一个指针,它指向存区的开始位置 */
   }
 else
   return(NULL);                         /* 当存区不够分配时,返回一个空指针 */
}
free(char *p)                            /* 释放存储区函数 */
  {if (p>=allocbuf && p>allocbuf+ALLOCSIZE)
     allocp=p;
  }
```

说明:定义一个全局指针变量 allocp,它指向指定的存储区中下一个可用的元素。开始时,allocp 指向此存储区 allocbuf 的开头,当调用 alloc(n)函数后,allocp指向 allocbuf 中的第 n 个元素(见图 10.6 中的 allocp')。

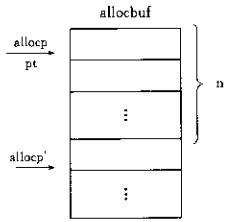


图 10.6

如果调用时用以下语句:

pt=alloc(n);

则 pt 的值为刚才所开辟的空间的首地址(allocp'-n)。

在调用 free 函数时,如果写出以下调用语句:

free(pt);

则把 allocp 的值改成 pt,即使得 allocp 指向刚才开辟空间的开头,恢复 allocp 的原值,就相当于释放此段空间,使这段空间可以做其他用途。

10.20 用指向指针的指针的方法对 5 个字符串排序并输出。

解:

```
#define LINEMAX 20                   /*定义字符串的最大长度*/
main()
{int i;
 char **p, *pstr[5],str[5][LINEMAX];
 for (i=0;i<5;i++)
   pstr[i]=str[i];   /*将第 i 个字符串的首地址赋予指针数组 pstr 的第 i 个元素*
```

```
    printf("Input 5 strings:\n");
    for (i=0;i<5;i++)
      scanf("%s",pstr[i]);
    p=pstr;
    sort(p);
    printf("strings sorted:\n");
    for (i=0;i<5;i++)
      printf("%s\n",pstr[i]);
}
sort(char **p)                    /*冒泡法对5个字符串排序的函数*/
{int i,j;
 char *temp;
 for (i=0;i<5;i++)
   {for (j=i+1;j<5;j++)
      {if (strcmp(*(p+i),*(p+j))>0)          /*比较后交换字符串地址*/
         {temp=*(p+i);
          *(p+i)=*(p+j);
          *(p+j)=temp;
         }
      }
   }
}
```

运行情况如下：

```
Input 5 strings:
China↙
America↙
India↙
Philippines↙
Canada↙
strings sorted:
America
Canada
China
India
Philippines
```

10.21 用指向指针的指针的方法对 n 个整数排序并输出。要求将排序单独写成一个函数。n 和整数在主函数中输入。最后在主函数中输出。

解：

```
main()
{void sort(int * * p,int n);
 int i,n,data[10],* * p,* pstr[10];
 printf("Input n:");
 scanf("%d",&n);
 for (i=0;i<n;i++)
   pstr[i]=&data[i];   /* 将第 i 个整数的地址赋予指针数组 pstr 的第 i 个元素 */
 printf("Input %d integer numbers:\n",n);
 for (i=0;i<n;i++)
   scanf("%d",pstr[i]);
 p=pstr;
 sort(p,n);
 printf("Now,the sequence is:\n");
 for (i=0;i<n;i++)
   printf("%d", * pstr[i]);
 printf("\n");
}
void sort(int * * p,int n)
{int i,j, * temp;
 for (i=0;i<n-1;i++)
   {for (j=i+1;j<n;j++)
      {if ( * * (p+i)> * * (p+j))        /* 比较后交换整数的地址 */
         {temp= * (p+i);
          * (p+i)= * (p+j);
          * (p+j)=temp;
         }
      }
   }
}
```

运行情况如下：

```
Input n:7↙
Input 7 integer numbers:
34 98 56 12 22 65 1↙
Now,the sequence is:
1 12 22 34 56 65 98
```

data 数组用来存放 n 个整数；pstr 是指针数组，每一个元素指向 data 数组中的一个元素；p 是指向指针的指针。请参考图 10.7。图 10.7(a)表示的是排序前的情况，图 10.7(b)表示的是排序后的情况。在图中可以看到，data 数组中数的次序没有变化，而 pstr 指针数组中的各元素的值(也就是它们的指向)改变了。

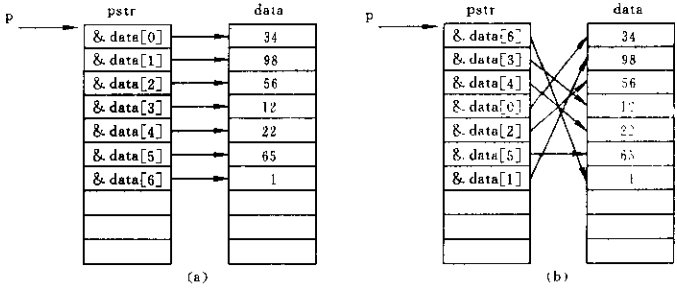

p
pstr
data
&. data[0]
&. data[1]
&. data[2]
&. data[3]
&. data[4]
&. data[5]
&. data[6]
34
98
56
12
22
65
1
(a)
p
pstr
data
&. data[6]
&. data[3]
&. data[4]
&. data[0]
&. data[2]
&. data[5]
&. data[1]
34
98
56
12
22
65
1
(b)

图 10.7

第 11 章　结构体与共用体

11.1　定义一个结构体变量(包括年、月、日)。计算该日在本年中是第几天,注意闰年问题。

解:解法一:见图 11.1。

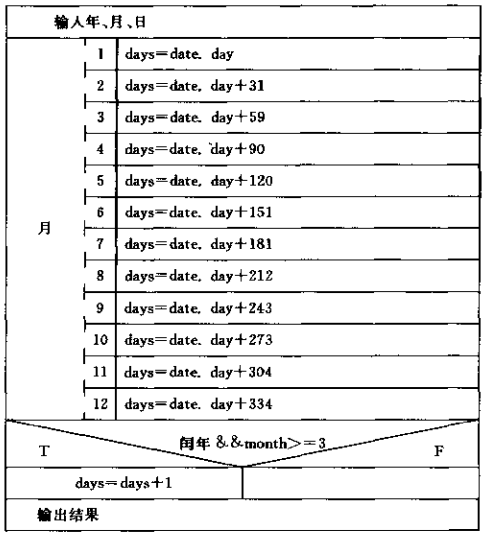


图　11.1

结构体变量 date 中的成员对应于输入的年、月、日。days 为天数。

```
struct
{int year;
 int month;
 int day;
}date;
main()
{int days;
 printf("Input year,month,day:");
 scanf("%d,%d,%d",&date. year,&date. month,&date. day);
 switch(date. month)
```

```
    {case 1: days=date.day;              break;
     case 2: days=date.day+31;           break;
     case 3: days=date.day+59;           break;
     case 4: days=date.day+90;           break;
     case 5: days=date.day+120;          break;
     case 6: days=date.day+151;          break;
     case 7: days=date.day+181;          break;
     case 8: days=date.day+212;          break;
     case 9: days=date.day+243;          break;
     case 10: days=date.day+273;         break;
     case 11: days=date.day+304;         break;
     case 12: days=date.day+334;         break;
    }
 if ((date.year %4== 0 && date.year % 100 ! = 0
      ||date.year % 400 == 0) && date.month >=3)days+=1;
 printf("\n %d / %d is the %dth day in%d. ",date.month,date.day,days,date.year);
}
```

运行情况如下：

```
Input year,month,day:2000,10,1↙
10 / 1 is the 275th day in 2000.
```

解法二：

```
struct date
{int year;
 int month;
 int day;
}
main()
{int i,days;
 int day_tab[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
 printf("Input year,month,day:");
 scanf("%d,%d,%d",&date.year,&date.month,&date.day);
 days=0;
 for (i=1;i<date.month;i++)
   days+=day_tab[i];
 days+=date.day;
 if ((date.year%4==0 && date.year%100! =0 || date.year%400==0) && date.
    month>=3)
      days+=1;
 printf("%d / %d is the %dth day in %d.",date.month,date.day,days,date.year);
}
```

运行情况如下

```
Input year,month,day:2001,10,1↙
10 / 1 is the 274th day in 2000.
```

11.2 写一个函数 days，实现上面的计算。由主函数将年、月、日传递给 days 函数，计算后将日数传回主函数输出。

解：函数 days 的程序结构基本与 11.1 题相同。

解法一：

```
struct y_m_d
{int year;
 int month;
 int day;
}date;
intdays(struct y_m_d date1)        /* 形参 date1 为结构体 struct y_m_d 类型 */
{int sum;
 switch(date1.month)
   {case 1: sum=date1.day;              break;
    case 2: sum=date1.day+31;           break;
    case 3: sum=date1.day+59;           break;
    case 4: sum=date1.day+90;           break;
    case 5: sum=date1.day+120;          break;
    case 6: sum=date1.day+151;          break;
    case 7: sum=date1.day+181;          break;
    case 8: sum=date1.day+212;          break;
    case 9: sum=date1.day+243;          break;
    case 10: sum=date1.day+273;         break;
    case 11: sum=date1.day+304;         break;
    case 12: sum=date1.day+334;         break;
   }
 if ((date1.year % 4 == 0 && date1.year % 100! =0
   || date1.year % 400 == 0) && date1.month >=3)
     sum+=1;
 return(sum);
}
main()
{ printf("Input year,month,day:");
  scanf("%d,%d,%d",&date.year,&date.month,&date.day);
  printf("\n");
  printf("%d / %d is the %dth day in %d. ",date.month,date.day,days(date),date.
        year);
}
```

运行情况如下：

```
Input year,month,day:2002,10,1↙
10 / 1 is the 274th day in 2000.
```

注意：在 main 函数中的 printf 函数用 days(date)调用 days 函数，其返回值就是天数。

解法二：

```
struct y_m_d
{int year;
 int month;
 int day;
} date;
main()
{ int days(int,int,int);          /* 对 days 函数的声明 */
  int i,day_sum;
  printf("input year,month,day:");
  scanf("%d,%d,%d",&date.year,&date.month,&date.day);
  day_sum=days(date.year,date.month,date.day);
  printf("\n%d / %d is the %dth day in %d.",date.month,date.day,day_sum,date.
  year);
}

days(int year,int month,int day)          /* 定义 days 函数 */
{int day_sum,i;
 int day_tab[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
 day_sum=0;
 for (i=1;i<month;i++)
   day_sum+=day_tab[i];
 day_sum+=day;
 if ((year%4==0 && year%100!=0 || year%4==0) && month>=3)
     day_sum+=1;
 return(day_sum);
}
```

运行情况如下：

```
input year,month,day:2005,7,1↙
10 / 1 is the 182th day in 2005.
```

11.3 编写一个函数 print，打印一个学生的成绩数组，该数组中有 5 个学生的数据记录，每个记录包括 num、name、score[3]，用主函数输入这些记录，用 print 函数输出这些记录。

解：

```
#define N 5
struct student
{ char num[6];
  char name[8];
  int score[4];
}stu[N];
main()
{int i,j;
 for (i=0;i<N;i++)
 {printf("\nInput score of student %d:\n",i+1);
  printf("NO.: ");
  scanf("%s",stu[i].num);
  printf("name: ");
  scanf("%s",stu[i].name);
  for (j=0;j<3;j++)
    {printf("score %d:",j+1);
     scanf("%d",&stu[i].score[j]);
    }
  printf("\n");
 }
 print(stu);
}
print(struct student stu[6])
{int i,j;
 printf("\n  NO.     name     score1     score2     score3\n");
 for (i=0;i<N;i++)
   {printf("%5s%10s",stu[i].num,stu[i].name);
      for (j=0;j<3;j++)
     printf("%9d",stu[i].score[j]);
     printf("\n");
   }
}
```

运行情况如下：

```
Input score of student 1:
NO.:101
name:Li
score 1: 90
score 2: 79
score 3: 89

Input score of student 2:
```

```
NO.:102
name:Ma
score 1: 97
score 2: 90
score 3: 68

Input score of student 3:
NO.:103
name:Wang
score 1: 77
score 2: 70
score 3: 78

Input score of student 4:
NO.:104
name:Fun
score 1: 67
score 2: 89
score 3: 56

Input score of student 5:
NO.:105
name:Xue
score 1: 87
score 2: 65
score 3: 69
```

NO.	name	score1	score2	score3
101	Li	90	79	89
102	Ma	97	90	68
103	Wang	77	70	78
104	Fun	67	89	56
105	Xue	87	65	69

11.4 在上题的基础上，编写一个函数 input，用来输入 5 个学生的数据记录。

解： input 函数的程序结构类似于 11.3 题中主函数的相应部分。

```
#define N 5
struct student
{char num[6];
 char name[8];
 int score[4];
} stu[N];
input(struct student stu[])
{int i,j;
 for (i=0;i<N;i++)
```

```
    { printf("input scores of student %d:\n",i+1);
      printf("NO. : ");
      scanf("%s",stu[i].num);
      printf("name:  ");
      scanf("%s",stu[i].name);
      for (j=0;j<3;j++)
        {printf("score %d:",j++);
         scanf("%d",&stu[i].score[j]);
        }
      printf("\n");
    }
}
```

写一个 main 函数，调用 input 函数以及题 11.3 中提供的 print 函数，就可以完成对学生数据的输入和输出。

11.5 有 10 个学生，每个学生的数据包括学号、姓名、3 门课的成绩，从键盘输入 10 个学生的数据，要求打印出 3 门课的总平均成绩，以及最高分的学生的数据（包括学号、姓名、3 门课成绩、平均分数）。

解：N-S 图见图 11.2。

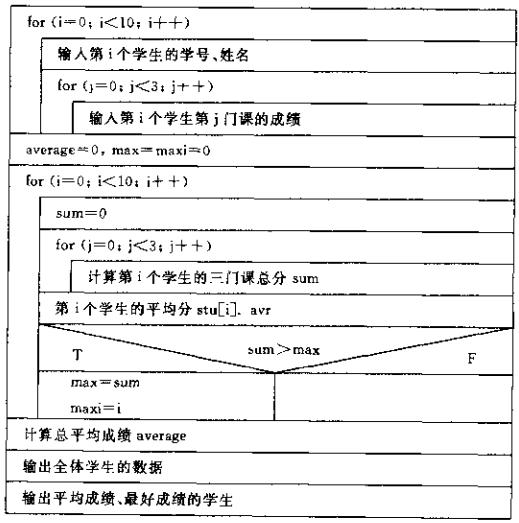


图 11.2

变量说明：

max：当前最好成绩；maxi：当前最好成绩所对应的下标序号；sum：第 i 个学生的

总成绩。

```
# define N 10
struct student
{ char num[6];
  char name[8];
  int score[4];
  float avr;
} stu[N];
main()
{ int i,j,max,maxi,sum;
  float average;
  /* 输入 */
  for (i=0;i<N;i++)
    {printf("\nInput scores of student %d:\n",i+1);
     printf("NO.:");
     scanf("%s",stu[i].num);
     printf("name:");
     scanf("%s",stu[i].name);
     for (j=0;j<3;j++)
       {printf("score %d:",j+1);
        scanf("%d",&stu[i].score[j]);
       }
    }
    /* 计算 */
  average=0;
  max=0;
  maxi=0;
  for (i=0;i<N;i++)
    {sum=0;
     for (j=0;j<3;j++)
       sum+=stu[i].score[j];
     stu[i].avr=sum/3.0;
     average+=stu[i].avr;
     if (sum>max)
       {max=sum;
        maxi=i;
       }
    }
  average/=N;
      /* 打印 */
  printf("  NO.     name     score1     score2     score3     average\n");
```

```
    for (i=0;i<N;i++)
      {printf("%5s%10s",stu[i].num,stu[i].name);
       for (j=0;j<3;j++)
       printf("%9d",stu[i].score[j]);
       printf("%8.2f\n",stu[i].avr);
      }
    printf("average=%6.2f\n",average);
    printf("The highest score is : %s, score total: %d.",stu[maxi].name,max);
}
```

运行情况如下：

```
Input scores of student 1:
NO. :101
name:Wang
score1: 93
score2: 89
score3: 87

Input scores of student 2:
NO. :102
name:Li
score1: 85
score2: 80
score3: 78

Input scores of student 3:
NO. :103
name:Zhao
score1: 65
score2: 70
score3: 59

Input scores of student 4:
NO. :104
name:Ma
score1: 77
score2: 70
score3: 83

Input scores of student 5:
NO. :105
name:Han
score1: 70
score2: 67
score3: 60
```

Input scores of student 6:

NO.:106

name:Zhang

score1: 99

score2: 97

score3: 95

Input scores of student 7:

NO.:107

name:Zhou

score1: 88

score2: 89

score3: 88

Input scores of student 8:

NO.:108

name:Chen

score1: 87

score2: 88

score3: 85

Input scores of student 9:

NO.:109

name:Yang

score1: 72

score2: 70

score3: 69

Input scores of student 10:

NO.:110

name:Liu

score1: 78

score2: 80

score3: 83

NO.	name	score1	score2	score3	average
101	Wang	93	89	87	89.67
102	Li	85	80	78	81.00
103	Zhao	65	70	59	64.67
104	Ma	77	70	83	76.67
105	Han	70	67	60	65.67
106	Zhang	99	97	95	97.00
107	Zhou	88	89	88	88.33
108	Chen	87	88	85	86.67
109	Yang	72	70	69	70.33

```
110        Liu        78        80        83        80.33
average= 80.03
The highest score is : Zhang,score total: 291.
```

11.6 编写一个函数 new，对 n 个字符开辟连续的存储空间，此函数应返回一个指针(地址)，指向字符串开始的空间。new(n)表示分配 n 个字节的内存空间，见图 11.3 示意。

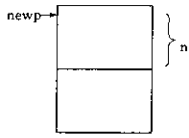


图 11.3

解：new 函数是分配 n 个连续字符的存储区，为此，应先开辟一个足够大的连续存储区。设置字符数组 newbuf[1000]，new 函数将在此上进行操作。newp 是指向可用存储区的起始地址的指针。每当请求 new 开出 n 个字符的存储区时，要先检查一下 newbuf 是否还有足够的可用空间。若有，则返回指针 newp 的当前值，然后修改 newp 为 newp+n，指向下一次可用空间的开始地址；若存储不够分配，则返回 NULL。

new 函数如下：

```
#define NULL 0
#define NEWSIZE 1000
char newbuf[NEWSIZE];
char *newp=newbuf;
char *new(int n)                    /* 定义 new 函数 */
  {if (newp+n <= newbuf+ NEWSIZE)
     {newp=newp+n;
      return(newp-n);
     }
  else
    return(NULL);
  }
```

11.7 写一函数 free，将上题用 new 函数占用的空间释放。free(p)表示将 p(地址)指向的单元以后的内存段释放。

解：free 的作用是将 newp 的值改为 p 的值。

```
#define NULL 0
#define NEWSIZE 1000
char newbuf[NEWSIZE];
char *newp=newbuf;
```

```
free(char *p)                    /* 定义 free 函数 */
  {if ((p>=newbuf) && (p<newbuf+NEWSIZE))
      newp=p;
  }
```

11.8 已有 a、b 两个链表，每个链表中的结点包括学号、成绩。要求把两个链表合并，按学号升序排列。

解：

```
#include <stdio.h>
#define NULL 0
#define LEN sizeof(struct student)
struct student
{long num;
  int score;
  struct student *next;
};
struct student listA,listB;
int n,sum=0;
main()
{struct student *creat(void);                                    /* 函数声明 */
 struct student *insert(struct student*,struct student *);       /* 函数声明 */
 void print(struct student *);                                   /* 函数声明 */
 struct student *ahead, *bhead, *abh;
 printf("\nInput list a:\n");
 ahead=creat();                    /* 调用 creat 函数，输入链表 a */
 sum=sum+n;
 printf("Input list b:\n");
 bhead=creat();                    /* 调用 creat 函数，输入链表 b */
 sum=sum+n;
 abh=insert(ahead,bhead);          /* 调用 insert 函数，将两表合并 */
 print(abh);                       /* 输出合并后的链表 */
}
struct student *creat(void)        /* 建立链表的函数 */
{struct student *p1, *p2, *head;
 n=0;
 p1=p2=(struct student *)malloc(LEN);
 printf("Input number & scores of student:\n");
 printf("If number is 0,stop inputing.\n");
 scanf("%ld,%d",&p1->num,&p1->score);
 head=NULL;
 while(p1->num != 0)
   {n=n+1;
    if (n==1) head=p1;
```

```
        else p2->next=p1;
      p2=p1;
      p1=(struct student *)malloc(LEN);
      scanf("%ld,%d",&p1->num,&p1->score);
    }
  p2->next=NULL;
  return(head);
}

struct student *insert(struct student *ah,struct student *bh)
                                        /*定义 insert 函数,用来合并两个链表 *
{struct student * pa1,* pa2,* pb1,* pb2;
 pa2=pa1=ah;
 pb2=pb1=bh;
 do
   {while((pb1->num>pa1->num) && (pa1->next != NULL))
      {pa2=pa1;
       pa1=pa1->next;
      }
    if (pb1->num<=pa1->num)
      {if (ah==pa1)
       ah=pb1;
       else pa2->next=pb1;
       pb1=pb1->next;
       pb2->next=pa1;
       pa2=pb2;
       pb2=pb1;
      }
   }
 while ((pa1->next!=NULL) || (pa1==NULL && pb1!=NULL));
   if ((pb1->num>pa1->num) && (pa1->next==NULL))
     pa1->next=pb1;
 return(ah);
}

void print(struct student *head)      /*输出函数*
{struct student * p;
 printf("\n There are %d records:\n",sum);
 p=head;
 if (p != NULL)
   do
     {printf("%ld %d\n",p->num,p->score);
      p=p->next;
     }
```

```
    while (p ! =NULL);
}
```

运行情况如下：

```
Input list a:                          (输入链表 a)
Input number & scores of student:      (输入学生的学号和成绩)
If number is 0,stop inputing.          (如果学号为 0,表示输入结束)
101,89↙
103,67↙
107,88↙
109,90↙
0↙
Input list b:                          (输入链表 b)
Input number & scores of student:      (输入学生的学号和成绩)
If number is 0,stop inputing.          (如果学号为 0,表示输入结束)
102,100↙
104,65↙
105,60 ↙
106,88↙
108,100↙
110,71↙
120,90↙
121,94↙
0↙
There are 12 records:
101 89
102 100
103 67
104 65
105 60
106 88
107 88
108 100
109 90
110 71
120 90
121 94
```

11.9 13 个人围成一圈，从第 1 个人开始顺序报号 1、2、3。凡报到“3”者退出圈子。找出最后留在圈子中的人原来的序号。

解： N-S 图见图 11.4。

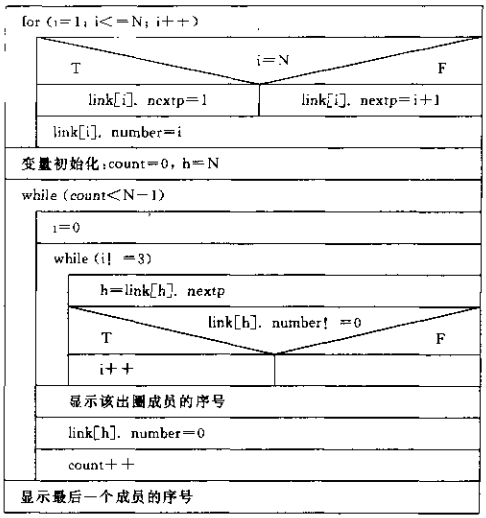


图 11.4

```
#define N 13
struct person
{
 int number;
 int nextp;
}link[N+1];
main()
{int i,count,h;
 for (i=1;i<=N;i++)
   {if (i==N)
      link[i].nextp=1;
    else
      link[i].nextp=i+1;
    link[i].number=1;
   }
 printf("\n");
 count=0;
 h=N;
 printf("sequence that persons leave the circle:\n");
 while(count<N-1)
   {i=0;
```

```
    while(i! =3)
      {h=link[h]. nextp;
       if (link[h]. number)
         i++;
      }
    printf("%4d",link[h]. number);
    link[h]. number=0;
    count++;
   }
 printf("\nThe last one is ");
 for (i=1;i<=N;i++)
   if (link[i]. number)
     printf("%3d", link[i]. number);
}
```

运行结果：

```
sequence that persons leave the circle:
  3  6  9  12  2  7  11  4  10  5  1  8
The last one is 13
```

11.10 有两个链表 a 和 b，设结点中包含学号、姓名。从 a 链表中删去与 b 链表中有相同学号的那些结点。

解：删除操作的 N-S 图见图 11.5。

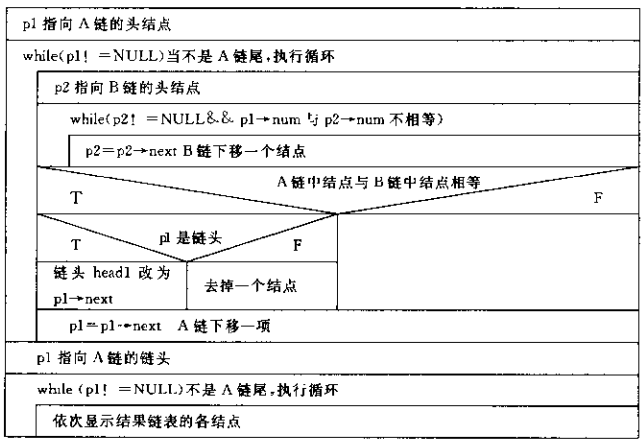


图 11.5

为减少程序运行时的输入量，先设两个结构体数组 a 和 b，并使用初始化的方法使之得到数据。建立链表时就利用这两个数组中的元素作为结点。

```
#define LA 4
#define LB 5
#define NULL 0
struct student
  char num[6];
  char name[8];
  struct student *next;
} A[LA],b[LB];
main()
{struct student a[LA]={{"101","Wang"},{"102","Li"},{"105","Zhang"},{"106","Wei"},
 struct student b[LB]={{"103","Zhang"},{"104","Ma"},{"105","Chen"},{"107","Guo"},
                       {"108","Lui"}};
 int i,j;
 struct student *p,*p1,*p2,*pt,*head1,*head2;
 /* 初始化 */
 head1=a;
 head2=b;
 printf("  list a:  \n");
   for (p1=head1,i=1;p1<a+LA;i++)
     {p=p1;
      p1->next=a+i;
      printf("%8s%8s\n",p1->num,p1->name);
      p1=p1->next;
     }
   p->next=NULL;
   printf("\n list b:\n");
   for (p2=head2,i=1;p2<b+LB;i++)
     {p=p2;
      p2->next=b+i;
      printf("%8s%8s\n",p2->num,p2->name);
      p2=p2->next;
     }
   p->next=NULL;
   printf("\n");
   /* 删除 */
   p1=head1;
   while(p1!=NULL)
     {p2=head2;
      while (p2!=NULL && strcmp(p1->num,p2->num)!=0)
        p2=p2->next;
```

```
        if (strcmp(p1->num,p2->num)==0)
          if (p1==head1)
             head1=p1->next;
          else
            p->next=p1->next;
        p=p1;
        p1=p1->next;
      }
    * 输出 *
  p1=head1;
  printf("\n result:\n");
  while(p1!=NULL)
    {printf("%7s %7s \n",p1->num,p1->name);
     p1=p1->next;
    }
}
```

运行结果：

```
list a:
    101    Wang
    102      Li
    105   Zhang
    106     Wei
list b:
    103   Zhang
    104      Ma
    105   Zhang
    107    Guo
    108    Lui
result:
    101    Wang
    102      Li
    106     Wei
```

11.11 建立一个链表，每个结点包括：学号、姓名、性别、年龄。输入一个年龄，如果链表中的结点所包含的年龄等于此年龄，则将此结点删去。

解：N-S 图如图 11.6 所示。

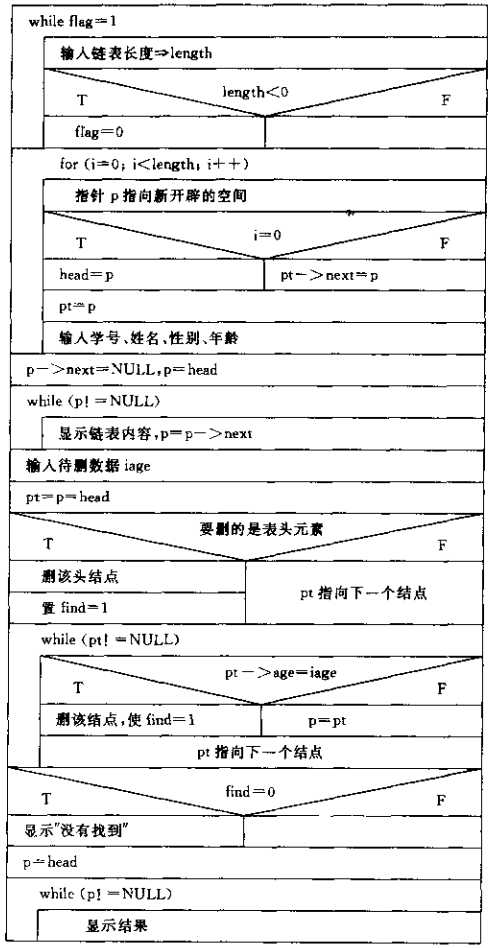


图 11.6

```
#define NULL 0
#define LEN sizeof(struct student)
struct student
{   char num[6];
    char name[8];
    char sex[2];
    int age;
```

```
    struct student *next;
} stu[10];
main()
{struct student *p,*pt,*head;
 int i,length,iage,flag=1;
 int find=0;              /*若找到待删除元素,则 find=1,否则 find=0*/
 while (flag==1)
   {printf("Input length of list(<10):");
    scanf("%d",&length);
    if (length<10)
      flag=0;
   }
 /*建立链表*/
 for (i=0;i<length;i++)
   {p=(struct student *) malloc(LEN);
      if (i==0)
        head=pt=p;
      else
        pt->next=p;
      pt=p;
      printf("NO:");
      scanf("%s",p->num);
      printf("name:");
      scanf("%s",p->name);
      printf("sex:");
      scanf("%s",p->sex);
      printf("age:");
      scanf("%d",&p->age);
   }
 p->next=NULL;
 p=head;
 printf("\n NO.    name    sex    age\n");          /*显示*/
 while(p! =NULL)
   {printf("%4s%8s%6s%6d\n",p->num,p->name,p->sex,p->age);
    p=p->next;
   }
 /*删除*/
 printf("Input age:");                            /*输入待删年龄*/
 scanf("%d",&iage);
 pt=head;
 p=pt;
 if (pt->age==iage)                               /*链头是待删元素*/
   {p=pt->next;
```

```
    head=pt=p;
    find=1;
   }
  else                                    /*链头不是待删元素*/
   pt=pt->next;
  while (pt! =NULL)
   {if (pt->age==iage)
      {p->next=pt->next;
       find=1;
      }
    else                                  /*中间结点不是待删元素*/
      p=pt;
    pt=pt->next;
   }
  if (!find)
   printf(" Not found%d.",iage);
  p=head;
  printf("\n NO.  name sex age\n");       /*显示结果*/
  while (p! =NULL)
   {
    printf("%4s%8s",p->num,p->name);
    printf("%6s%6d\n",p->sex,p->age);
    p=p->next;
   }
}
```

运行情况如下：

```
Input length of list(<10):4↙                    (输入链表长度)
NO. :101↙
Name:Ma↙
Sex:m↙
Age:20↙
NO. :102↙
Name:Li↙
Sex:f↙
Age:23↙
NO. :103↙
Name:Zhang↙
Sex:m↙
Age:19↙
NO. :104↙
Name:Wang↙
Sex:m↙
Age:19↙
```

```
NO.       name      sex     age
101         Ma        m      20
102         Li        f      23
103       Zhang       m      19
104       Wang        m      19
Input age:19↙                          (输入待删年龄)
NO.       name      sex     age
101         Ma        m      20
102         Li        f      23
```

11.12 将一个链表按逆序排列，即将链头当链尾，链尾当链头。

解法一：N-S 图如图 11.7。

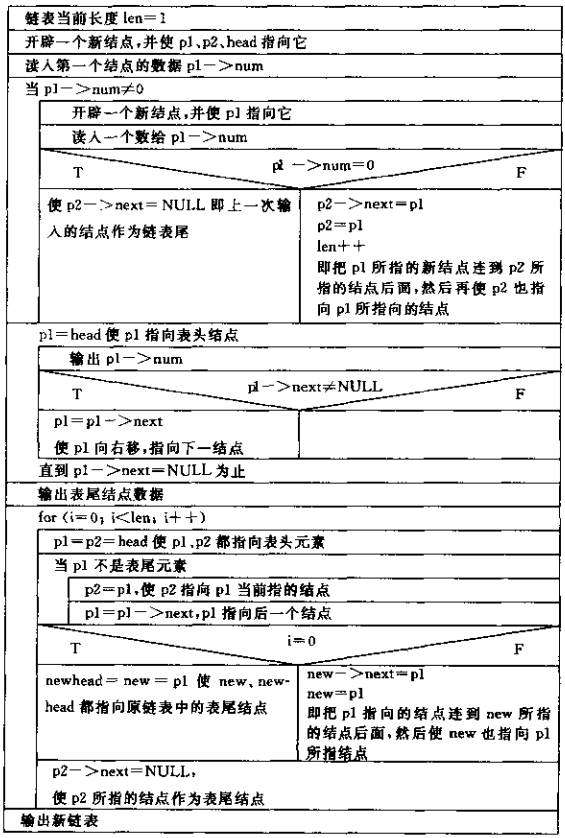


图 11.7

```
#define NULL 0
struct stu
{int num;
 struct stu *next;
}
main()
{int len=1,i;
 struct stu *p1,*p2,*head,*new,*newhead;
 /* 建立链表 */
 p1=p2=head=(struct stu *) malloc(sizeof(struct stu));
 printf("Input number(0:list end):");
 scanf("%d",&p1->num);
 while(p1->num!=0)
   {p1=(struct stu *) malloc(sizeof(struct stu));
    printf("Input number(0:list end):");
    scanf("%d",&p1->num);
    if (p1->num==0)
      p2->next=NULL;
    else
      {p2->next=p1;
       p2=p1;
       len++;
      }
   }
 /* 输出原链表 */
 p1=head;
 printf("\n The original list:\n");
 do
   {printf("%4d",p1->num);
    if (p1->next!=NULL)
      p1=p1->next;
   }
   while (p1->next!=NULL);
 printf("%4d",p1->num);
 /* 输出新链表 */
 for (i=0;i<len;i++)
   {p2=p1=head;
    while (p1->next!=NULL)
      {p2=p1;
       p1=p1->next;
      }
    if (i==0)
```

```
        newhead=new=p1;
      else
        new=new->next=p1;
      p2->next=NULL;
    }
  printf("\n\n The new list:\n");
  p1=newhead;
  for (i=0;i<len;i++)
    {printf("%4d",p1->num);
     p1=p1->next;
    }
  printf("\n");
}
```

运行情况如下：

```
Input number(0:list end):12↙   (输入链表中各结点的数据,输入 0 代表结束)
Input number(0:list end):6↙
Input number(0:list end):9↙
Input number(0:list end):32↙
Input number(0:list end):7↙
Input number(0:list end):0↙
The original list:
  12  6  9  32  7
The new list:
  7  32  9  6  12
```

说明：在建立链表时，当输入完链表中所有结点的数据后输入 0，表示链表至此结束。
最后输入的 0 不包括在链表中。

解法二：

在看懂上面程序基础上，可以参考下面的程序。在下面的程序中分成几个函数分别完成建表、按逆序重新排列和输出的功能。同时不用输入"0"来控制建表的结束，而用输入一个或多个字符(用"end"或其他任意字符串均可)。当一个 scanf 函数成功地执行时返回值为 1，否则为 0，因此当向一个整型变量 p1->num 输入一个字符串时，scanf 函数返回 0，用它来控制建表的结束。

```
#define NULL 0
struct stu
  {int num;
   struct stu *next;
  } *p1,*p2;
struct stu *creat(void)
```

```
{int temp;
 struct stu *head=NULL;
 printf("\ninput number:");
 while (scanf("%d",&temp))
   {p1=(struct stu *) malloc(sizeof(struct stu));
    if(head==NULL)
      head=p1;
    else
    p2->next=p1;
    p1->num=temp;
    printf("Input number:");
    p2=p1;
   }
 p2->next=NULL;
 return (head);
}

void output(struct stu *outhead)
  {
   for (p1=outhead; p1!=NULL; printf("%4d",p1->num),p1=p1->next);
  }

struct stu * turnback(struct stu *head)
  {struct stu *new,*newhead=NULL;
   do { p2=NULL;
        p1=head;
        while(p1->next !=NULL) p2=p1,p1=p1->next;
        if (newhead==NULL) newhead=p1,new=newhead->next=p2;
        new=new->next=p2;
        p2->next=NULL;
      } while(head->next!=NULL);
   return (newhead);
  }

main()
{struct stu *head;
 head=creat();
 printf("\n\nThe original list:");
 output(head);
 head=turnback(head);
 printf("\n\nThe new list:");
 output(head);
}
```

运行情况如下：

```
Input number:12↙                    (输入链表，非整数值表示结束)
Input number:34↙
Input number:9↙
Input number:0↙
Input number:7↙
Input number:6↙
Input number:2↙
Input number:end↙

The original list:
  12  34  9  0  7  6  2

The new list:
  2  6  7  0  9  34  12
```

说明：在 Turbo C 环境中有时在输入实型数据(特别是对实型数组元素的输入)时，常会出现不正常的情况。例如有以下程序：

```
main()
{struct student
 {int num;
  char name[20];
  float score;
 }
 struct student stu [4];
 struct student * p;
 int i,temp=0;
 float max;
 for (i=0;i<4;i++)
   scanf ("%d %s%f",&stu [i]. num,stu [i]. name,&stu [i]. score);
 for (max=stu [0]. score,i=1;i<4;i++)
   if (stu [i]. score>max)
     {max=stu [i]. score;temp=i;}
 p=stu+temp;
 printf ("\nNo. %d\nname:%s\nscore:%4. 1f\n",p->num,p->name,p->score);
}
```

这个程序的作用是：有 4 个学生，每个学生包括学号、姓名、成绩，要求找出成绩最高者的姓名和成绩。

运行程序输入以下数据：

```
101 Li 90↙
```

屏幕上出现以下信息：

```
scanf: floating point formats not linked
```

Abnormal program termination

程序运行中断。

经检查，程序的逻辑和语法都是正确的，而且在其他 C 编译系统环境中能正常运行。在 Turbo C 环境中不能正常运行的原因并不是程序有问题，而是由于 TC 系统不完善引起的。当然可以将程序改在其他 C 编译系统环境下运行，但是对多数人来说是不方便的。如果仍在 Turbo C 环境下运行，希望得到正确的结果，可以采取"妥协迂回"的办法。先用一实型变量接受输入的实型数据，然后再将它转赋给数组元素。将程序修改如下：

```
main()
{struct student
 {int num
  char name [20];
  float score;
 }
 struct student stu [4];
 struct student * p;
 int i,temp=0;
 float max,s;                    /* 设一个实型变量 s */
 for (i=0;i<4;i++)
     {scanf ("%d %s%f",&stu [i]. num,stu [i]. name,&s);  /* 输入一个实数给 s */
      stu [i]. score=s;          /* 再将 s 的值赋给数组元素 stu [I]. score */
     }
 for (max=stu [0]. score,i=1;i<4;i++)
   if (stu [i]. score>max)
    {max=stu [i]. score;temp=i;}
 p=stu+temp;
 printf ("\nNo. %d\nname:%s\nscore:%4. 1f\n",p->num,p->name,p->score);
}
```

运行此程序，按上述方法输入数据，运行正常，结果正确。

在本章的一些习题中，定义结构体时，对成员 score(成绩)指定为整型，运行正常。如果将它指定为实型，在输入数据时可能会出现类似上面的问题，请读者参照上述办法相机处理。

第12章 位 运 算

12.1 编写一个函数 getbits，从一个 16 位的单元中取出某几位（即该几位保留原值，其余位为 0）。函数调用形式为：

getbits(value,n1,n2)

value 为该 16 位（两个字节）单元中的数据值，n1 为欲取出的起始位，n2 为欲取出的结束位。如：

getbits(0101675,5,8)

表示对八进制 101675 这个数，取出它从左面起的第 5 位到第 8 位。

解：

```
main()
{
  unsigned int a;
  int n1,n2;
  printf("Input an octal number:");
  scanf("%o",&a);
  printf("Input n1,n2:");
  scanf("%d,%d",&n1,&n2);
  printf("result:%o\n",getbits(a,n1-1,n2));
}
getbits(unsigned value,int n1,int n2)
{unsigned int z;
 z=~0;
 z=(z>>n1) & (z<<(16-n2));
 z=value & z;
 z=z>>(16-n2);
 return(z);
}
```

运行结果：

```
Input an octal number:173253↙
Input n1,n2:5,8↙
result:6
```

12.2 写一个函数，对一个 16 位的二进制数取出它的奇数位（即从左边起第 1、3、5、…、15 位）。

解：

```
main()
{unsigned getbits(unsigned);
 unsigned int a;
 printf("\nInput an octal number:");
 scanf("%o",&a);
 printf("result:%o\n",getbits(a));
}

unsigned getbits(unsigned value)
{int i,j,m,n;
 unsigned int z,a,q;
 z=0;
 for (i=1;i<=15;i+=2)
   {q=1;
    for (j=1;j<=(16-i-1)/2;j++)
      q=q*2;
    a=value>>(16-i);
    a=a<<15;
    a=a>>15;
    z=z+a*q;
   }
 return(z);
}
```

运行结果：

① Input an octal number: 145432↙
result:263

说明：八进制数 145432 用二进制表示为：1100101100011010
取其奇数位得到：1 0 1 1 0 0 1 1
用八进制表示为：2 6 3

② Input an octal number:043526↙
result:21

说明：八进制数 043526 用二进制表示为：0100011101010110
取其奇数位得到：0 0 0 1 0 0 0 1
用八进制表示为：0 2 1

12.3 编一程序，检查一下你所用的计算机系统的 C 编译在执行右移时是按照逻辑位移的原则，还是按照算术右移的原则。如果是逻辑右移，请编一函数实现算术右移。如果是算术右移，请编写一函数实现逻辑右移。

解：

```
main ()
```

```
{int a,n,m;
 a=~0;
 if ((a>>5)!=a)
   {printf("\nTurbo C ,logical move! \n");
    m=0;
    }
 else
   {printf("\nTurbo C,arithmetic move! \n");
    m=1;
    }
 printf("Input an octal number:");
 scanf("%o",&a);
 printf("How many digit move towards the right:");
 scanf("%d",&n);
 if (m==0)
   printf("Arithmetic right move,result:%o\n",getbits1(a,n));
 else
   printf("Logical right move,result:%o",getbits2(a,n));
}
getbits1(unsigned value,int n)
{unsigned z;
 z=~0;
 z=z>>n;
 z=~z;
 z=z|(value>>n);
 return(z);
}
getbits2(unsigned value,int n)
{unsigned z;
 z=(~(1>>n))&(value>>n);
 return(z);
}
```

运行情况如下：

```
Turbo C,arithmetic move!                    (Turbo C为算术右移)
Input an octal numher:173253↙               (输入八进制数173253)
How many digit move towards the right:4↙    (输入右移几位)
Logical right move,result: 7552             (逻辑右移的结果为7552)
```

说明：开始时并不知道 Turbo C 是算术右移还是逻辑右移，因此先检查它是采用哪一种方式。～0 用二进制表示 16 位 1，即 1111111111111111，将它右移 5 位。如果是逻辑右移，则其结果为 0000011111111111；如果是算术右移，则其结果

为 1111111111111111。因此，若 a>>5 等于 a，就证明是算术右移；若 a>>5 不等于 a，就是逻辑右移。在程序中，设变量 m，m=0 表示逻辑右移，m=1 表示算术右移。

定义两个函数 getbits1 和 getbits2 分别实现算术右移和逻辑右移（在任何 C 系统中都适用）。请读者自己分析。

12.4 编一函数用来实现左右循环移位。函数名为 move，调用方法为：

move(value, n)

其中 value 为要循环位移的数，n 为位移的位数。如 n<0 为左移；n>0 为右移。如 n=4，表示要右移 4 位；n=-3，表示要左移 3 位。

解：

```
main ()
{ unsigned moveright(unsigned,int);
  unsigned moveleft(unsigned,int);
  unsigned a;
  int n;
  printf("\nInput an octal number: ");
  scanf("%o",&a);
  printf("Input n: ");
  scanf("%d",&n);
  if (n>0)
     {moveright(a,n);
      printf("result:%o\n",moveright(a,n));
     }
  else
     {n=-n;
      moveleft(a,n);
      printf("result:%o\n",moveleft(a,n));
     }
}

unsigned moveright(unsigned value,int n)          /* 右循环移位函数 */
  {unsigned z;
   z=(value>>n)|(value<<(16-n));
   return(z);
  }

unsigned moveleft(unsigned value,int n)           /* 左循环移位函数 */
  {unsigned z;
   z=(value>>(16-n))|(value<<n);
   return(z);
  }
```

运行情况如下：

① Input an octal number:152525↙
  Input n:4↙
  result:56525
② Input an octal number:152525↙
  Input n:-4↙
  result:52535

12.5 设计一个函数，使给出一个数的原码，能得到该数的补码。

解：

```
main ()
{ unsigned int a;
  unsigned int getbits(unsigned);
  printf("\nInput an octal number:");
  scanf("%o",&a);
  printf("result:%o\n",getbits(a));
}

unsigned int getbits(unsigned value)          /*求一个二进制的补码函数*/
{ unsigned int z;
  z=value&0100000;
  if (z==0100000)
     z=~value+1;
  else
     z=value;
  return(z);
}
```

运行情况如下：

① Input an octal number:2345↙
  result:2345
② Input an octal number:152525↙
  result:25253

一个正数的补码等于该数原码，一个负数的补码等于该数的反码加 1。

第13章 文　件

13.1 对C文件操作有些什么特点？什么是缓冲文件系统？什么是非缓冲文件系统？这二者的缓冲区有什么区别？

解： 略。

13.2 什么是文件型指针？通过文件指针访问文件有什么好处？

解： 略。

13.3 对文件的打开与关闭的含义是什么？为什么要打开和关闭文件？

解： 略。

13.4 从键盘输入一个字符串，将其中的小写字母全部转换成大写字母，然后输出到一个磁盘文件"test"中保存。输入的字符串以"!"结束。

解：

```
#include <stdio.h>
main ( )
{
 FILE *fp;
 char str[100];
 int i=0;
 if ((fp=fopen("test","w"))==NULL)
     {printf("Can not open the file\n");
      exit(0);
     }
 printf("Input a string:\n");
 gets(str);
 while (str[i]! ='!')
     {if (str[i]>='a'&& str[i]<='z')
         str[i]=str[i]-32;
          fputc(str[i],fp);
          i++;
     }
 fclose(fp);
 fp=fopen("test","r");
 fgets(str,strlen(str)+1,fp);
 printf("%s\n",str);
 fclose(fp);
}
```

运行情况如下：

```
Input a string:
i love china!
I LOVE CHINA
```

13.5 有两个磁盘文件“A”和“B”，各存放一行字母，要求把这两个文件中的信息合并（按字母顺序排列），输出到一个新文件“C”中。

解： 先分别将 A、B 文件的内容读出放到数组 C 中，再对数组 C 排序。最后将数组内容写到 C 文件中。N-S 图如图 13.1 所示。

```
#include <stdio.h>
main ()
{
 FILE *fp;
 int i,j,n,i1;
 char c[100],t,ch;
 if ((fp=fopen("a1","r"))==NULL)
   {
    printf("Can not open the file\n");
    exit(0);
   }
 printf("\n file A:\n");
 for (i=0;(ch=fgetc(fp))!=EOF;i++)
   {
    c[i]=ch;
    putchar(c[i]);
   }
 fclose(fp);
 i1=i;
 if ((fp=fopen("b1","r"))==NULL)
   {printf("\n Can not open the file");
    exit(0);
   }
 printf("\n file B:\n");
 for (i=i1;(ch=fgetc(fp))!=EOF;i++)
   {
    c[i]=ch;
    putchar(c[i]);
   }
 fclose(fp);
 n=i;
 for (i=0;i<n;i++)
```

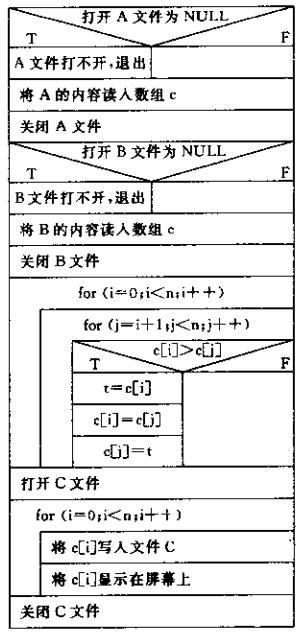


图 13.1

```
    for (j=i+1;j<n;j++)
      if (c[i]>c[j])
      {t=c[i];
       c[i]=c[j];
       c[j]=t;}
  printf("\n file C:\n");
  fp=fopen("c1","w");
     for (i=0;i<n;i++)
      {putc(c[i],fp);
       putchar(c[i]);
      }
  fclose(fp);
}
```

运行结果：

```
file A:
I LOVE CHINA
file B:
I LOVE BEIJING
file C:
␣␣␣␣ABCEEEGHIIIIIJLLNNOOVV
```

13.6 有 5 个学生，每个学生有 3 门课的成绩，从键盘输入以上数据（包括学生号、姓名、三门课成绩），计算出平均成绩，将原有数据和计算出的平均分数存放在磁盘文件 stud 中。

解法一：N-S 图如图 13.2。

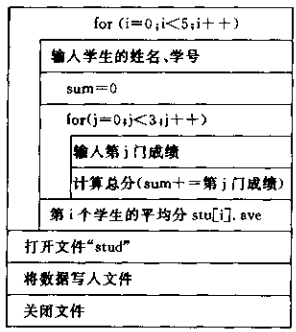


图 13.2

```
#include <stdio.h>
struct student
{char num[10];
 char name[8];
 int score[3];
 float ave;
 } stu[5];
main()
{int i,j,sum;
 FILE *fp;
 for(i=0;i<5;i++)  /* 输入 */
   {printf("\nInput score of student
    %d:\n",i+1);
    printf("NO.:");
    scanf("%s",stu[i].num);
```

```
        printf("name:");
        scanf("%s",stu[i].name);
        sum=0;
        for (j=0;j<3;j++)
          {printf("score %d:",j+1);
           scanf("%d",&stu[i].score[j]);
           sum+=stu[i].score[j];
          }
        stu[i].ave=sum/3.0;
      }
      /*将数据写入文件*/
  fp=fopen("stud","w");
  for (i=0;i<5;i++)
     if (fwrite(&stu[i],sizeof(struct student),1,fp)!=1)
       printf("File write error\n");
  fclose(fp);
  /* 检查文件内容 */
  fp=fopen("stud","r");
  for (i=0;i<5;i++)
    {fread(&stu[i],sizeof(struct student),1,fp);
     printf("%s,%s,%d,%d,%d,%6.2f\n",stu[i].num,stu[i].name,stu[i].score[0],
           stu[i].score[1],stu[i].score[2],stu[i].ave);}
}
```

运行情况如下：

```
Input score of student 1:
NO.:110↙
name:Li↙
score 1: 90↙
score 2: 89↙
score 3: 88↙

Input score of student 2:
NO.:120↙
name:Wang↙
score 1:80↙
score 2:79↙
score 3:78↙

Input score of student 3:
NO.:130↙
name:Chen↙
score 1:70↙
```

```
score 2:69↙
score 3:68↙

Input score of student 4:
NO.:140↙
name:Ma↙
score 1:100↙
score 2:99↙
score 3:98↙

Input score of student 5:
NO.:150↙
name:Wei↙
score 1:60↙
score 2:59↙
score 3:58↙

110,Li,90,89,88, 89.00
120,Wang,80,79,78, 79.00
130,Chen,70,69,68, 69.00
140,Ma,100,99,98. 99.00
150,Wei,60,59,58, 59.00
```

说明：在程序的第一个 for 循环中，有两个 printf 函数语句用来提示用户输入数据，即“printf("\ninput scoreof student %d:\n",i+1);”和“printf("score %d:",j+1);”，其中用“i+1”和“j+1”而不是用 i 和 j 的用意是使显示提示时，序号从 1 起，即学生 1 和成绩 1(而不是学生 0 和成绩 0)，以符合人们习惯，但在内存中，数组元素下标仍从 0 算起。

程序最后 5 行用来检查文件 stud 中的内容是否正确，从结果来看，是正确的。请注意：用 fwrite 函数向文件输出数据时不是按 ASCII 码方式输出的，而是按内存中存储数据的方式输出的(例如一个整数占 2 个字节，一个实数占 4 个字节)，因此不能用 type 命令输出该文件中的数据。

解法二：该题也可以用下面程序来实现：

```
#include <stdio.h>
#define SIZE 5
struct student
{char name[10];
 int num;
 int score[3];
 float ave;
} stud[SIZE];
main()
```

```
{ void save(void);                          /* 函数声明 */
  int i;
  float sum[SIZE];
  FILE *fp1;
  for (i=0;i<SIZE;i++)                      /* 输入数据,并求每个学生的平均分 */
    {scanf("%s %d %d %d %d",stud[i].name,&stud[i].num,&stud[i].score[0],
          &stud[i].score[1],&stud[i].score[2]);
     sum[i]=stud[i].score[0]+stud[i].score[1]+stud[i].score[2];
     stud[i].ave=sum[i]/3;
    }
  save();                                   /* 调用 save 函数,向文件 stu.dat 输出数据 */
  fp1=fopen("stu.dat","rb");                /* 用只读方式打开 stu.dat 文件 */
  printf("\n name   NO.  score1 score2 score3 ave\n");
  for (i=0;i<SIZE;i++)                      /* 从文件读入数据并在屏幕输出 */
    {
     fread(&stud[i],sizeof(struct student),1,fp1);
     printf("%-10s %3d %7d %7d %7d %8.2f\n",stud[i].name,stud[i].num,
     stud[i].score[0],stud[i].score[1],stud[i].score[2],stud[i].ave);
    }
  fclose (fp1);
}

void save(void)                             /* 向文件输出数据的函数 */
  {
   FILE *fp;
   int i;
   if ((fp=fopen("stu.dat","wb"))==NULL)
     {
      printf("Can not open the file\n");
      return;
     }
   for(i=0;i<SIZE;i++)
     if(fwrite(&stud[i],sizeof(struct student),1,fp)!=1)
       {printf("File write error\n");
        return;
       }
   fclose(fp);
  }
```

运行情况如下：

```
Zhang 101 77 78 98↙
Li 102 67 78 88↙
```

Wang 103 89 99 97↙
Wei 104 77 76 98↙
Tan 105 78 89 97↙

name	NO.	score1	score2	score3	ave
Zhang	101	77	78	98	84.33
Li	102	67	78	88	77.67
Wang	103	89	99	97	95.00
Wei	104	77	76	98	83.67
Tan	105	78	89	97	88.00

本程序用 save 函数将数据写到磁盘文件上，再从文件读回，然后用 printf 函数输出，从运行结果可以看到文件中的数据是正确的。

13.7 将上题 stud 文件中的学生数据按平均分进行排序处理，并将已排序的学生数据存入一个新文件 stu_sort 中。

解法一：N-S 图如图 13.3 所示。

```
#include <stdio.h>
#define N 10
struct student
{char num[10];
 char name[8];
 int score[3];
 float ave;
 } st[N],temp;
 main()
{
 FILE *fp;
 int i,j,n;
 /*读文件*/
 if ((fp=fopen("stud","r"))==NULL)
   {printf("Can not open the file.");
    exit(0);
   }
 printf("\nfile 'stud': ");
 for (i=0; fread(&st[i], sizeof(struct
      student),1,fp)!=0;i++)
   {printf("\n%8s%8s",st[i].num,st[i].
          name);
    for (j=0;j<3;j++)
      printf("%8d",st[i].score[j]);
    printf("%10.2f",st[i].ave);
   }
```

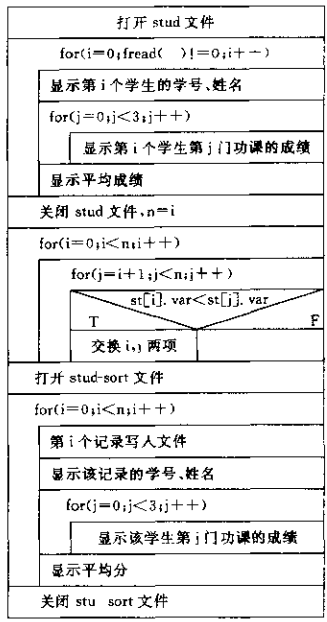


图 13.3

```
    fclose(fp);
    n=i;
    /* 排序 */
    for (i=0;i<n;i++)
      for (j=i+1;j<n;j++)
        if (st[i].ave < st[j].ave)
          {temp=st[i];
           st[i]=st[j];
           st[j]=temp;
          }
    /* 输出 */
    printf("\nnow:");
    fp=fopen("stu_sort","w");
    for (i=0;i<n;i++)
        {fwrite(&st[i],sizeof(struct student),1,fp);
         printf("\n%8s%8s",st[i].num,st[i].name);
         for (j=0;j<3;j++)
            printf ("%8d",st[i].score[j]);
         printf("%10.2f",st[i].ave);
        }
    fclose(fp);
}
```

运行结果：

```
file 'stud':
     110       Li     90     89     88     89.00
     120     Wang     80     79     78     79.00
     130    Chen      70     69     68     69.00
     140      Ma     100     99     98     99.00
     150      Wei     60     59     58     59.00
now:
     140      Ma     100     99     98     99.00
     110       Li     90     89     88     89.00
     120     Wang     80     79     78     79.00
     130    Chen      70     69     68     69.00
     150      Wei     60     59     58     59.00
```

解法二：

与上题解法二相应，也可以使用下面的程序来实现本题要求。

```
#include <stdio.h>
#define SIZE
struct student
```

```
{char name[10];
 int num;
 int score[3];
 float ave;
} stud[SIZE],work;
main()
{void sort(void);
 int i;
 FILE *fp;
 sort();
 fp=fopen("stud_sort.dat","rb");
 printf("Sorted student's scores list is as follows\n");
 printf("------------------------------------------\n");
 printf(" NAME      NO.     SCORE1   SCORE2   SCORE3    AVE   \n");
 printf("------------------------------------------\n");
 for (i=0;i<SIZE;i++)
   {fread(&stud[i],sizeof(struct student),1,fp);
    printf("%-10s %3d %8d %8d %8d %9.2f\n",stud[i].name,stud[i].num,
          stud[i].score[0],stud[i].score[1],stud[i].score[2],stud[i].ave);
   }
 fclose(fp);
}

void sort(void)
{FILE *fp1,*fp2;
 int i,j;
 if ((fp1=fopen("stu.dat","rb"))==NULL)
   {printf("Can not open the file\n");
    exit(0);
   }
 if ((fp2=fopen("stud_sort.dat","wb"))==NULL)
   {printf("The file write error\n");
    exit(0);
   }
 for (i=0;i<SIZE;i++)
   if (fread(&stud[i],sizeof(struct student),1,fp1)!=1)
   {printf("file read error\n");
    exit(0);
   }
 for (i=0;i<SIZE;i++)
   {for (j=i+1;j<SIZE;j++)
      if (stud[i].ave<stud[j].ave)
```

```
            {work=stud[i];
             stud[i]=stud[j];
             stud[j]=work;
            }
        fwrite(&stud[i],sizeof(struct student),1,fp2);
    }
 fclose(fp1);
 fclose(fp2);
}
```

运行情况如下：

Sorted student's scores list is as follows

NAME	NO.	SCORE1	SCORE2	SCORE3	AVE
Wang	103	89	99	97	95.00
Tan	105	78	89	97	88.00
Zhang	101	77	78	98	84.33
Wei	104	77	76	98	83.67
Li	102	67	78	88	77.67

本程序是将已排好序的数据存放到磁盘文件 stud_sort.dat 中，然后再读入内存并输出。

13.8 将上题已排序的学生成绩文件进行插入处理。插入一个学生的 3 门课成绩，程序先计算新插入学生的平均成绩，然后将它按平均成绩高低顺序插入，插入后建立一个新文件。

解：N-S 图如图 13.4。

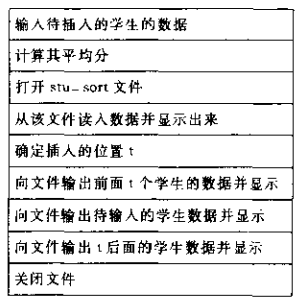


图 13.4

```
#include <stdio.h>
struct student
{char num[10];
 char name[8];
 int score[3];
 float ave;
 }st[10],s;
main()
{FILE *fp,*fp1;
 int i,j,t,n;
 /* 输入待插入的学生 s 的数据 */
 printf("\nNO.:");
 scanf("%s",s.num);
 printf("name:");
 scanf("%s",s.name);
 printf("score1,score2,score3:");
```

```
    scanf("%d,%d,%d",&s.score[0],&s.score[1],&s.score[2]);
    s.ave=(s.score[0]+s.score[1]+s.score[2])/3.0;
    /*从文件读数据*/
    if((fp=fopen("stu_sort","r"))==NULL)
      {printf("can not open file.");
       exit(0);
      }
    printf("original data:\n");
      for (i=0;fread(&st[i],sizeof(struct student),1,fp)!=0;i++)
      {printf("\n%8s%8s",st[i].num,st[i].name);
        for (j=0;j<3;j++)
          printf("%8d",st[i].score[j]);
        printf("%10.2f",st[i].ave);
      }
    n=i;
    for (t=0;st[t].ave>s.ave && t<n;t++);          /*确定插入的位置t*/
    /*向文件写数据*/
    printf("\nnow:\n");
    fp1=fopen("sort1.dat","w");
    for (i=0;i<t;i++)                  /*先输出平均成绩高于学生s的学生的成绩*/
      {fwrite(&st[i],sizeof(struct student),1,fp1);
       printf("\n %8s%8s",st[i].num,st[i].name);
       for (j=0;j<3;j++)
          printf("%8d",st[i].score[j]);
       printf("%10.2f",st[i].ave);
      }
    fwrite(&s,sizeof(struct student),1,fp1);
    printf("\n %8s %7s %7d %7d %7d%10.2f",s.num,s.name,s.score[0],
          s.score[1],s.score[2],s.ave);     /*输出学生s的成绩*/
    for (i=t;i<n;i++)                  /*输出平均成绩低于学生s的学生的成绩*/
      {fwrite(&st[i],sizeof(struct student),1,fp1);
       printf("\n %8s%8s",st[i].num,st[i].name);
       for(j=0;j<3;j++)
          printf("%8d",st[i].score[j]);
       printf("%10.2f",st[i].ave);
      }
    fclose(fp);
    fclose(fp1);
    }
```

运行结果：

```
NO.:160↙
```

```
name:Tan↙
score1,score2,score3:98,97,98↙
original data:
    140      Ma    100    99    98    99.00
    110      Li     90    89    88    89.00
    120    Wang     80    79    78    79.00
    130    Chen     70    69    68    69.00
    150     Wei     60    59    58    59.00
now:
    140      Ma    100    99    98    99.00
    160     Tan     98    97    98    97.67
    110      Li     90    89    88    89.00
    120    Wang     80    79    78    79.00
    130    Chen     70    69    68    69.00
    150     Wei     60    59    58    59.00
```

为节省篇幅，本题和题 13.9 中不再给出上题“解法二”的程序，请读者自己编写程序。

13.9 上题结果仍存入原有的 stu_sort 文件而不另建立新文件。

解：

```
#include <stdio.h>
struct student
{char num[10];
 char name[8];
 int score[3];
 float ave;
} st[10],s;
main()
{FILE *fp,*fp1;
 int i,j,t,n;
 printf("\nNO.:");
 scanf("%s",s.num);
 printf("name:");
 scanf("%s",s.name);
 printf("score1,score2,score3:");
 scanf("%d,%d,%d",&s.score[0],&s.score[1],&s.score[2]);
 s.ave=(s.score[0]+s.score[1]+s.score[2])/3.0;
 /*从文件读数据*/
 if((fp=fopen("stu_sort","r"))==NULL)
   {printf("Can not open the file.");
    exit(0);
   }
 printf ("original data:\n");
```

```
    for (i=0;fread(&st[i],sizeof(struct student),1,fp)!=0;i++)
      {printf("\n%8s%8s",st[i].num,st[i].name);
       for (j=0;j<3;j++)
          printf("%8d",st[i].score[j]);
       printf("%10.2f",st[i].ave);
      }
    fclose(fp);
    n=i;
    for (t=0;st[t].ave>s.ave && t<n;t++);
    /*向文件写数据*/
    printf("\nnow:\n");
    fp=fopen("stu_sort","W");
    for (i=0;i<t;i++)
      {fwrite(&st[i],sizeof(struct student),1,fp);
       printf("\n %8s%8s",st[i].num,st[i].name);
       for (j=0;j<3;j++)
          printf("%8d",st[i].score[j]);
       printf("%10.2f",st[i].ave);
      }
    fwrite(&s,sizeof(struct student),1,fp);
    printf("\n%9s%8s%8d%8d%8d%10.2f",s.num,s.name,s.score[0],
           s.score[1],s.score[2],s.ave);
    for (i=t;i<n;i++)
      {fwrite(&st[i],sizeof(struct student),1,fp);
       printf("\n %8s%8s",st[i].num,st[i].name);
       for(j=0;j<3;j++)
          printf("%8d",st[i].score[j]);
       printf("%10.2f",st[i].ave);
      }
    fclose(fp);
}
```

运行情况如下：

```
NO.:160↙
name:Hua↙
score1,score2,score3:78,89,91↙
original data:
    140      Ma   100   99   98   99.00
    110      Li    90   89   88   89.00
    120    Wang    80   79   78   79.00
    130    Chen    70   69   68   69.00
    150     Wei    60   59   58   59.00
```

now:

140	Ma	100	99	98	99.00
110	Li	90	89	88	89.00
160	Hua	78	89	91	86.00
120	Wang	80	79	78	79.00
130	Chen	70	69	68	69.00
150	Wei	60	59	58	59.00

13.10 有一磁盘文件 emploee，内存放职工的数据。每个职工的数据包括：职工姓名、职工号、性别、年龄、住址、工资、健康状况、文化程度。要求将职工名和工资的信息单独抽出来另建一个简明的职工工资文件。

解：N-S 图如图 13.5 所示。

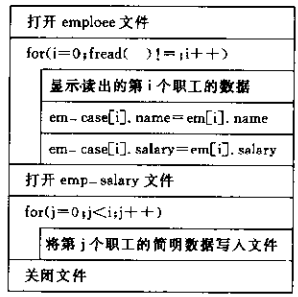


图 13.5

```
#include <stdio.h>
struct emploee
{char    num[6];
 char    name[10];
 char    sex[2];
 int     age;
 char    addr[20];
 int     salary;
 char    health[8];
 char    class[10];
}em[10];
struct emp
{char name[10];
 int salary;
}em_case[10];
main()
{FILE *fp1,*fp2;
 int i,j;
 if ((fp1=fopen("emploee","r"))==NULL)
   {printf("Can not open the file.");
    exit(0);
   }
 printf("\n NO.    name    sex    age    addr    salary    health    class\n");
 for (i=0;fread(&em[i],sizeof(struct emploee),1,fp1)!=0;i++)
   {printf("\n%4s%8s%4s%6d%10s%6d%10s%8s",em[i].num,em[i].name,em[i].
          sex,em[i].age,em[i].addr,em[i].salary,em[i].health,em[i].class);
    strcpy(em_case[i].name,em[i].name);
    em_case[i].salary=em[i].salary;
   }
```

```
    printf("\n\n* * * * * * * * * * * * * * * * * * * * * * * * * * * * *    ");
    if((fp2=fopen("emp_salary","wb"))==NULL)
      {printf("Can not open the file.");
       exit(0);}
    for (j=0;j<i;j++)
      {if(fwrite(&em_case[j],sizeof(struct emp),1,fp2)!=1)
         printf("error!");
       printf("\n   %12s%10d",em_case[j].name,em_case[j].salary);
      }
    printf("\n    * * * * * * * * * * * * * * * * * * * * * * * * * * * * *    ");
    fclose(fp1);
    fclose(fp2);
}
```

运行结果如下：

NO.	name	sex	age	addr	salary	health	class
101	Li	m	23	Beijing	670	good	P. H. D.
102	Wang	f	45	Shanghai	780	had	master
103	Ma	m	32	Taijin	650	good	univ.
104	Liu	f	56	Xian	540	pass	college

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        Li    670
      Wang    780
        Ma    650
       Liu    540
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

说明：数据文件 emploee 是事先建立好的，其中已有职工数据，而 emp_salary文件则是由程序建立的。

建立 emploee 文件的程序如下：

```
#include <stdio.h>
struct emploee
{char    num[6];
 char    name[10];
 char    sex[2];
 int     age;
 char    addr[20];
 int     salary;
 char    health[8];
 char    class[10];
}em[10];
main()
```

```
{ FILE * fp;
  int i,j;
  printf("input NO., name, sex, age, addr,salary,health,class\n");
  for (i=0;i<4;i++)
    scanf(" %s %s %s %d %s %d %s %s",em[i].num,em[i].name,em[i].sex,
          &em[i].age,em[i].addr,&em[i].salary,em[i].health,em[i].class);
     /*将数据写入文件*/
  if((fp=fopen("emploee","w"))==NULL)
    {printf("Can not open the file.");
     exit(0);
    }
  for (i=0;i<4;i++)
     if(fwrite(&em[i],sizeof(struct emploee),1,fp)!=1)
        printf("error\n");
  fclose(fp);
}
```

在运行此程序时从键盘输入 4 个职工的数据，程序将它们写入 emploee 文件。在运行前面一个程序时从 emploee 文件中读出数据并输出到屏幕，然后建立一个简明文件，同时在屏幕上输出。

13.11 从上题的“职工工资文件”中删去一个职工的数据，再存回原文件。

解：N-S 图见图 13.6。

```
#include <stdio.h>
#include <string.h>
struct emploee
{char  name[10];
 int   salary;
}emp[20];
main()
{FILE * fp;
 int i,j,n,flag;
 char name[10];
 int salary;
 if ((fp = fopen ("emp- salary",
      "rb"))==NULL)
      {printf("Can not open file.");
       exit(0);
      }
 printf("\n  original data:");
```

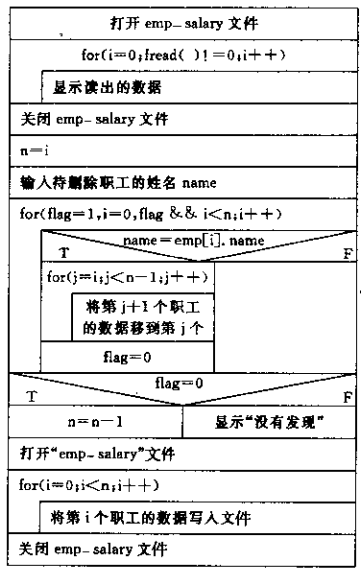


图 13.6

```
  for (i=0;fread(&emp[i],sizeof(struct emploee),1,fp)!=0;i++)
    printf("\n   %8s    %7d",emp[i]. name,emp[i]. salary);
  fclose(fp);
  n=i;
  printf("\n Input name deleted:");
  scanf("%s",name);
  for (flag=1,i=0;flag && i<n;i++)
    {if (strcmp(name,emp[i]. name)==0)
       {for (j=i;j<n-1;j++)
            {strcpy(emp[j]. name,emp[j+1]. name);
             emp[j]. salary=emp[j+1]. salary;
            }
        flag=0;
       }
    }
  if(!flag)
     n=n-1;
  else
    printf("\n Not found!");
  printf("\nNow,the content of file:\n");
  fp=fopen("emp_ salary","wb");
  for (i=0;i<n;i++)
    fwrite(&emp[i],sizeof(struct emploee),1,fp);
  fclose(fp);
  fp=fopen("emp . salary","r");
  for (i=0;fread(&emp[i],sizeof(struct emploee),1,fp)!=0;i++)
    printf("\n%8s %7d",emp[i]. name,emp[i]. salary);
  fclose(fp);
}
```

运行情况如下：

```
original data:
   Li        670
   Wang      780
   Ma        650
   Liu       540
Input name deleted:Ma↙
Now,the content of file:
   Li        670
   Wang      780
   Liu       540
```

13.12 从键盘输入若干行字符(每行长度不等),输入后把它们存储到一磁盘文件中。再从该文件中读入这些数据,将其中小写字母转换成大写字母后在显示屏上输出。

解: N-S 图见图 13.7。

```
#include <stdio.h>
main()
{ int i,flag;
  char str[80],c;
  FILE *fp;
  fp=fopen("text","w");
  flag=1;
  while(flag==1)
    {printf("\nInput string:\n");
     gets(str);
     fprintf(fp,"%s",str);
     printf("\nContinue?");
     c=getchar();
     if ((c=='N')||(c=='n'))
       flag=0;
     getchar();
    }
    fclose(fp);
    fp=fopen("text","r");
    while(fscanf(fp,"%s",str)!=EOF)
     {for (i=0;str[i]!='\0';i++)
        if ((str[i]>='a') && (str[i]<='z'))
          str[i]-=32;
      printf("\n%s\n",str);
     }
  fclose(fp);
}
```

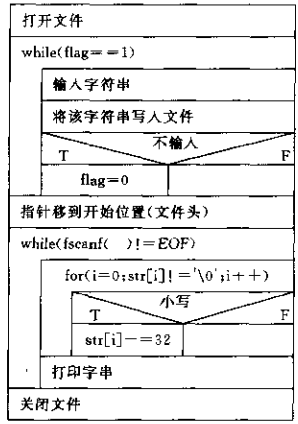


图 13.7

运行情况如下:

```
Input string: abcdef.↙
Continue? y↙
Input string: ghijkl.↙
Continue? y↙
Input string: mnopqrst↙
Continue? n↙

ABCDEF.
GHIJKL.
```

MNOPQRST.

此程序运行结果是正确的，但是如果输入的字符串中包含了空格，就会发生一些问题，例如输入“i am a student.”，得到的结果是：

I
AM
A
STUDENT.

这是因为用 fscanf 函数从文件读入字符串时，把空格作为一个字符串的结束标志，因此把该行作为 4 个字符串来处理，分别输出在 4 行上。请读者考虑怎样解决这个问题。