1.1 习题 1 及参考答案

1.1 列举几种读者所知道的计算机硬件和软件。

【参考答案】

硬件：CPU、内存、硬盘、光盘、键盘、鼠标等。
软件：Windows，QQ，Internet Explorer，Word 等。

1.2 冯·诺依曼机模型有哪几个基本组成部分？

【参考答案】 控制器、运算器、存储器、输入设备和输出设备。

1.3 尝试把例 1.4 改为计算“123+456”。

```
#include <stdio.h>
main()
{
    printf("%d\n", 123+456);
}
```

1.4 列举几种程序设计语言。

【参考答案】 C，C++，Java，C#，PHP，ASP，Pascal 等。

1.5 列举几个在生活和学习中成功应用 IT 技术的例子。

【参考答案】 可编程的微波炉、洗衣机、手机、文曲星、电子游戏机等。

1.2 习题 2 及参考答案

2.1 下列变量名中哪些是合法的？

π　2a　a#　C$　t3　_var　θ　int

【参考答案】 合法的为　t3　_var

2.2 选择填空

（1）C 语言中用_____表示逻辑值“真”。

A）true　　B）整数值 0　　C）非零整数值　　D）T

（2）下列合法的字符常量为_____。

A）"a"　　B）'\n'　　C）'china'　　D）a

（3）设有语句 char c='\72';，则变量 c_____。

A）包含 1 个字符　　B）包含 2 个字符　　C）包含 3 个字符　　D）不合法

（4）字符串常量"\t\"Name\\Address\n"的长度为_____。

A）19　　B）15　　C）18　　D）不合法

（5）设 a,b,c 为 int 型变量，且 a = 3 ,b = 4 , c = 5，下面表达式值为 0 的是_____。

A）'a' && 'b'　　B）a <= b

C）a || b+c && b-c　　D）!((a<b) && !c || 1)

（6）若有以下定义：

```
char a ;    int b ;    float c ;    double d ;
```

则表达式 a * b + d - c 的值的类型为_____。

A）float　　B）int　　C）char　　D）double

（7）设有语句 int a = 3;，执行语句 a += a -= a * a; 后，变量 a 的值是_____。

A）3　　B）0　　C）9　　D）-12

（8）设有语句 int a = 3;，执行语句 printf("%d", -a ++); 后，输出的结果是_____，变量 a 的值是_____。

A）3　　B）4　　C）-3　　D）-12

【参考答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
答案	C	B	A	B	D	D	D	C B

2.3　将下列数学表达式表示为合法的 C 语言表达式。

（1）$\dfrac{\sqrt{a^2+b^2}}{2c}$　　（2）$|(a+b)(c+d)+2|$　　（3）(lnx+siny)/2

（4）$2\pi r$　　（5）$\dfrac{1}{1+\dfrac{1}{x}}$　　（6）$\dfrac{\sin 30^\circ + 2e^x}{2y+y^x}$

【参考答案】

（1）sqrt(a*a+b*b)/(2*c)或 sqrt(pow(a,2)+pow(b,2))/(2*c)

（2）fabs((a+b)*(c+d)+2)

（3）(log(x)+sin(y))/2　（注：y 应为弧度）

例如，若 y 值为 30°，sin(y)应写成 sin(3.14/180*30)，不能直接写成 sin(30)。

（4）2*3.1415*r

或 #define PI 3.1415

表达式写为： 2*PI*r

（5）1/(1+1.0/x)

（6）(sin(3.14/180*30)+2*exp(x))/(2*y+pow(y,x))

1.3 习题3及参考答案

3.1 C语句分为哪几类?

【参考答案】 表达式语句、函数调用语句、控制语句、空语句和复合语句。

3.2 选择填空

(1) 下列可作为C语言赋值语句的是_____。

A) x = 3, y = 5　　B) a = b = c

C) i -- ;　　D) y = int (x) ;

(2) 以下程序的输出结果为_____。

```
main()
{
    int  a = 2, c = 5;

    printf("a = %%d, b = %%d\n", a, c);
}
```

A) a = %2, b = %5　　B) a = 2, b = 5

C) a=%%d, b=%%d　　D) a=%d, b=%d

【参考答案】 (1) C　　(2) D

3.3 分析下面程序段，指出错误的原因和程序错在哪里，并将其改正。

(1)
```
int  a, b;
scanf("%d,%d", a, b);
```

(2)
```
float  f = 2.39;
printf("%d", f);
```

(3)
```
double  var;
long    a;
scanf("%f%d", &var, &a);
```

(4)
```
int  a, b;
scanf("%d,%d\n", a, b);
```

(5)
```
float  f;
scanf("%5.2f", &f);
```

(6)
```
main()
{
 int  a. b;
 scanf("a = %d, b = %d", &a, &b);
 printf("a = %d, b = %d\n", a, b);
}
```

程序运行时输入：6, 2 ↙

【参考答案】

（1）错误：在 scanf 函数中，参数应是 a,b 两个变量的地址。

改正：scanf("%d,%d",&a,&b);

（2）错误：数据输出格式与数据类型不匹配。

改正：printf("%f", f);

（3）错误：数据输入格式与数据类型不匹配。

改正：scanf("%lf%ld",&var, &a);

（4）错误：在 scanf 函数输入格式控制串中多了'\n',a,b 前面少了'&'。

改正：scanf("%d,%d",&a,&b);

（5）错误：%f 的输入格式不应有精度控制。

改正：scanf("%5f",&f);

（6）错误：程序输入错误使得变量 a,b 的值不是 6,2。

改正：应输入　a=6,b=2 ↙

3.4　分析下列程序，写出程序运行结果。

（1）

```
main()
{
    char  c1 = 'a', c2 = 'b', c3 = 'c';
    printf("a%cb%cc%cabc\n", c1, c2, c3);
}
```

（2）

```
main()
{
    int  x = 12 , y = 8;
    printf("\n%5d%5d%5d", !x, x || y, x && y);
}
```

（3）

```
main()
{
    int  x, y;
    scanf("%2d%*2s%2d", &x, &y);
    printf("%d", x + y);
}
```

程序执行时从键盘输入：1234567↙

（4）

```
main()
{
    int    a = 2, b = 3 ;
    float  x = 3.5, y = 2.5 ;
    printf("%f", (float)(a+b) / 2 + (int)x % (int)y) ;
}
```

(5)

```
main()
{
   int  x = 12 , y = 8;
   printf("%d %d\n", x++, ++y);
   printf("%d %d\n", x, y);
}
```

(6)

```
main()
{
   int  x = 12, y = 8, p, q;
   p = x++;
   q = ++y;
   printf("%d %d\n", p, q);
   printf("%d %d\n", x, y);
}
```

【参考答案】

(1) aabbccabc

(2)　　0　　1　　1

(3) 68

(4) 3.500000

(5) 12 9

　　13 9

(6) 12 9

　　13 9

3.5　已知三角形的三边长 a,b,c，计算三角形面积的公式为

$$s=\frac{1}{2}(a+b+c),\quad \text{area}=\sqrt{s(s-a)(s-b)(s-c)}$$

要求编写程序，从键盘输入 a,b,c 的值，计算并输出三角形的面积。

```
#include <stdio.h>
#include  <math.h>
main()
{
    float  a, b, c;                /*a,b,c为三边变量*/
    float  s, area;
    printf("Input a,b,c:");
    scanf("%f,%f,%f",&a,&b,&c);
    s = 1.0 / 2 * (a + b + c);
    area = sqrt(s * (s - a) * (s - b) * (s - c));
    printf("area=%.2f\n", area);
}
```

程序运行结果：

```
Input a,b,c:3,4,5↙
area=6.00
```

3.6 编程从键盘输入圆的半径r，计算并输出圆的周长和面积。
[提示：将计算圆周长和面积公式中的π定义为符号常量。]

【参考答案】

```
#include <stdio.h>

#define PI  3.14

main()
{
    float  r ;                  /*r为半径变量*/
    float  circum, area;

    printf("Input r:");
    scanf("%f", &r);

    circum = 2*PI*r;
    area = PI*r*r;

    printf("circum=%.2f,area=%.2f\n", circum, area);
}
```

程序运行结果：

```
Input r:5↙
circum=31.40,area=78.50
```

1.4 习题4及参考答案

4.1 简答题

（1）什么是算法？算法在程序设计中的重要作用是什么？

（2）什么是结构化程序设计？其基本思想是什么？

（3）什么是“自顶向下、逐步求精”的程序设计方法？

【参考答案】（1）

所谓算法，就是一个有穷规则的集合，其中的规则确定了一个解决某一特定类型问题的运算序列。简单地说，就是为解决一个具体问题而采取的确定的有限的操作步骤。当然这里所说的算法仅指计算机算法，即计算机能执行的算法。

每个程序都要依靠算法和数据结构，在某些特殊领域，如计算机图形学、语法分析、数值分析、人工智能和模拟仿真等，解决问题的能力几乎完全依赖于最新的算法和数据结构。因此，针对某一应用领域，要想开发出优质高效的程序，开发人员除了要熟练掌握程序设计语言这种工具和必要的程序设计方法以外，更重要的是要多了解、多积累并逐渐学会自己设计一些好的算法。

【参考答案】（2）

结构化程序设计是一种实现程序设计的原则和方法，按照这种原则和方法设计的程序具有结构清晰，容易阅读，容易修改，容易验证等特点。

结构化程序设计的基本思想归纳起来有以下几点：

- 采用顺序、选择和循环三种基本结构作为程序设计的基本单元，避免无限制地使用goto语句而使流程任意转向。
- 三种基本结构应具有如下良好特性：
  ① 只有一个入口。
  ② 只有一个出口。
  ③ 无死语句，即不存在永远都执行不到的语句。
  ④ 无死循环，即不存在永远都执行不完的循环。
- 程序设计采用“自顶向下、逐步求精”和模块化的方法。

【参考答案】（3）

自顶向下方法是先写出结构简单、清晰的主程序来表达整个问题，在此问题中所包含的复杂子问题用子程序来实现。若子问题中还包含复杂的子问题，再用另外一个子程序来解决，直到每一细节都可以用高级语言清楚表达为止。逐步求精技术可以理解为一种不断地由自底向上修正所补充的自顶向下的程序设计方法。

4.2 选择题

(1) 在下面的条件语句中，只有一个在功能上与其他三个语句不等价（其中 s1 和 s2 表示某个 C 语句），这个不等价的语句是_____。

A) `if (a) s1; else s2;`

B) `if (!a) s2; else s1;`

C) `if (a != 0) s1; else s2;`

D) `if (a == 0) s1; else s2;`

(2) 设有声明语句 int a=1,b=0;，则执行以下语句后输出为_____。

```
switch (a)
{
   case 1:switch (b)
          {
               case 0: printf("**0**");break;
               case 1: printf("**1**");break;
          }
   case 2: printf("**2**");break;
}
```

A) **0**　　B) **0****2**　　C) **0****1****2**　　D) 有语法错误

(3) 在 while (x)语句中的 x 与下面条件表达式等价的是_____。

A) x == 0　　B) x == 1　　C) x != 1　　D) x != 0

(4) 若有语句 int x;，下面程序段的输出结果为_____。

```
for (x=3; x<6; x++)
{
    printf((x%2) ? "**%d" : "##%d\n", x);
}
```

A)
```
**3
##4
**5
```

B)
```
##3
**4
##5
```

C)
```
##3
**4##5
```

D)
```
**3##4
**5
```

【参考答案】

题号	(1)	(2)	(3)	(4)
答案	D	B	D	D

4.3 写出程序的运行结果。

(1) 下面程序运行结果为_____。

```
#include <stdio.h>
main()
{
    int  a = 2, b = 3, c = 1;
    if (a > b)
        if (a > c)
            printf("%d\n",a);
        else
            printf("%d\n",b);
    printf("over!\n");
}
```

【参考答案】 (1) over!

(2) 若从终端上由第一列开始输入以下数据:

```
right?↙
```

则程序运行结果为_____。

```
#include <stdio.h>
main()
{
    char  c;

    c = getchar();
    while (c != '?')
    {
        putchar(c);
        c = getchar();
    }
}
```

【参考答案】 (2) right

(3) 对下面程序，若输入数据同上，则程序运行结果为_____。

```
#include <stdio.h>
main()
{
    char  c;

    while ((c = getchar()) != '?')
    {
        putchar(c);
    }
}
```

【参考答案】 (3) right

(4) 对下面程序，若输入数据同上，则程序运行结果为_____。

```
#include <stdio.h>
main()
{
    char  c;

    while (putchar (getchar()) != '?') ;
}
```

【参考答案】 (4) right?

（5）请读者上机运行下面的程序，并仔细体会两个程序运行结果的不同。

程序 1

```
#include <stdio.h>
main()
{
    char  c;

    while ((c = getchar()) != '\n')
    {
        putchar(c);
    }
    printf("End!\n");
}
```

程序 2

```
#include <stdio.h>
main()
{
    char  c;

    while ((c = getchar()) != '$')
    {
        putchar(c);
    }
    printf("End!\n");
}
```

运行时输入：abcdefg$abcdefg↙

则两个程序的运行结果分别为_____。

【参考答案】 （5）程序 1 运行结果：abcdefg$abcdefgEnd!

程序 2 运行结果：abcdefgEnd!

（6）下面程序的运行结果为_____。

```
#include <stdio.h>
main()
{
    int   i, j, k;
    char  space = ' ';
    for (i=1;i<=4;i++)
    {
        for (j=1; j<=i; j++)
        {
            printf("%c",space);
        }

        for (k=1; k<=6; k++)
        {
            printf("*");
        }
        printf("\n");
    }
}
```

【参考答案】 （6）

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

(7) 下面程序的运行结果为_____。

```
#include <stdio.h>
main()
{
    int  n;

    for (n=1; n<=5; n++)
    {
        if (n % 2)
        {
            printf("*");
        }
        else
        {
            continue;
        }
        printf("#");
    }
    printf("$\n");
}
```

【参考答案】 *#*#*#$

4.4 阅读程序，按要求在空白处填写适当的表达式或语句，使程序完整并符合题目要求。

【参考答案】 (1)

① (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)

② flag

【参考答案】 (2)

① (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')

② ch <= '9' && ch >= '0'

③ ch == ' '

【参考答案】 (3)

① fahr

② celsius = 5.0 / 9 * (fahr - 20)

③ fahr = fahr + step

4.5　编程判断输入整数的正负性和奇偶性。流程图如图 1-1 所示。

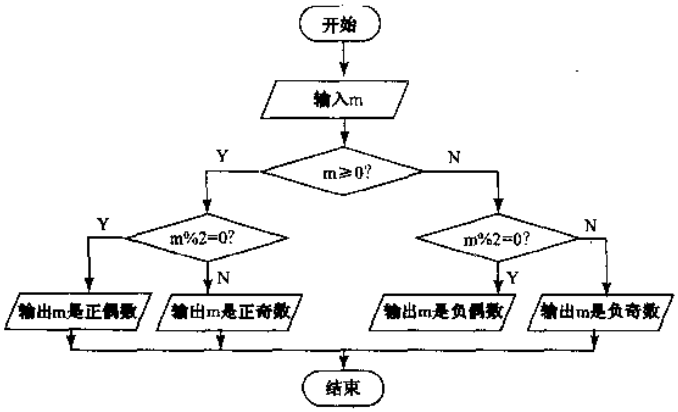

图 1-1　习题 4.5 的流程图

【参考答案】

```
#include <stdio.h>
main()
{
    int m;

    printf("Input m: ");
    scanf("%d", &m);    /*输入一个整数*/
    if (m >= 0)             /*是否为正数*/
    {
        if (m%2 == 0)   /*是正数，且能被2整除，是正偶数*/
        {
            printf("%d is a positive even\n", m);
        }
        else            /*不能被2整除，是正奇数*/
        {
            printf("%d is a positive odd\n", m);
        }
    }
    else
    {
        if (m % 2 == 0)
        {
            printf("%d is a negative even\n", m);  /*是负偶数*/
        }
        else
        {
            printf("%d is a negative odd\n", m);   /*是负奇数*/
        }
    }
}
```

程序的 2 次测试结果如下：

① Input m:6↙

6 is a positive even

② Input m:-7↙

-7 is a negative odd

4.6 编程计算分段函数

$$y=\begin{cases} e^{-x} & x>0 \\ 1 & x=0 \\ -e^{x} & x<0 \end{cases}$$

输入 x，打印出 y 值。流程图如图 1-2 所示。

【参考答案】

```
#include <stdio.h>
#include <math.h>
main()
{
    int x;
    double y;

    printf("Input x: ");
    scanf("%d", &x);        /* 输入一个整数*/
    if (x > 0)
    {
        y = exp(-x);  /*如果大于0，计算y=exp(-x)的值 */
    }
    else if (x == 0)
    {
        y = 1;   /*x=0，则y=1*/
    }
    else
    {
        y = -exp(x);  /*x<0，则y=-exp(x)*/
    }
    printf("y=%f\n", y);
}
```

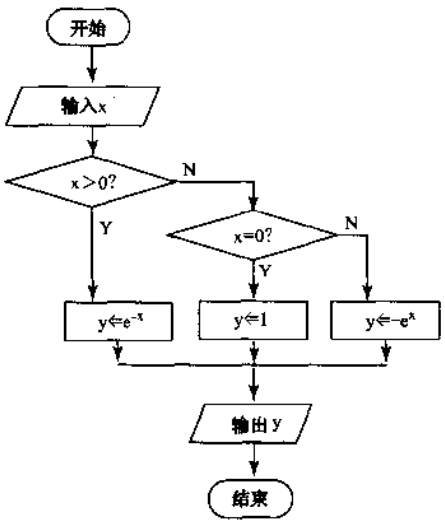

图 1-2 习题 4.6 的流程图

程序的 3 次测试结果如下：

① Input x:4↙
y=0.018316

② Input x:0↙
y=1.000000

③ Input x:-4↙
y=-0.018316

4.7　输入三角形的三条边 a,b,c，判断它们能否构成三角形。若能构成三角形，指出是何种三角形（等腰三角形、直角三角形、一般三角形）。流程图如图 1-3 所示。

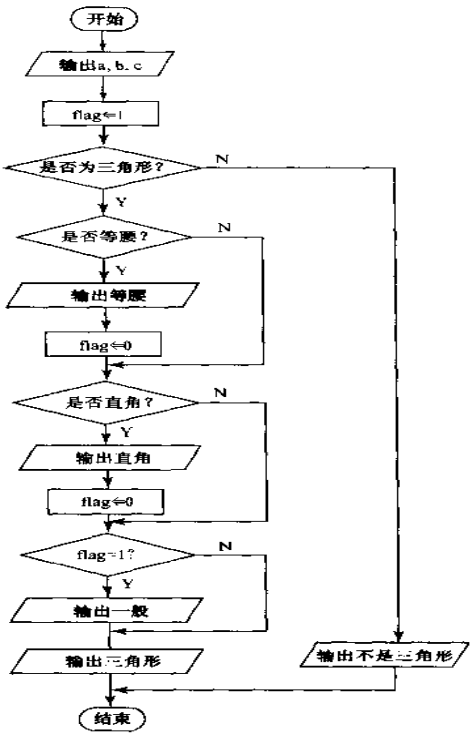

图 1-3　习题 4.7 的流程图

【参考答案】

```
#include <stdio.h>
#include <math.h>
#define LIMIT  1e-1

main()
{
    float a, b, c;
    int flag = 1;

    printf("Input the three edge length: ");
    scanf("%f, %f, %f", &a, &b , &c);        /*输入三角形的三条边*/
    if ((a + b) > c && ((b + c) > a) && ((a + c) > b))
                                             /*三角形的基本条件*/
    {
        if (fabs(a-b) <= LIMIT  || fabs(b-c) <= LIMIT
            || fabs(c-a) <= LIMIT)           /*等腰三角形的条件*/
        {
            printf("等腰");
            flag = 0;
        }
        if (fabs(a * a + b * b - c * c) <= LIMIT
            || fabs(a * a + c * c - b * b) <= LIMIT
            || fabs(c * c + b * b - a * a) <= LIMIT) /*直角三角形的条件 */
        {
            printf("直角");
            flag = 0;
        }
        if (flag)
        {
            printf("一般");
        }
        printf("三角形\n");
    }
    else
    {
        printf("不是三角形\n");
    }
}
```

程序的 4 次测试结果如下：

① Input the three edge:3,4,5↙
直角三角形

② Input the three edge:4,4,5↙
等腰三角形

③ Input the three edge:10,10,14.14↙
等腰直角三角形

④ Input the three edge:3,4,9↙
不是三角形

4.8 在屏幕上显示一张如下所示的时间表：

```
*****Time*****
1  morning
2  afternoon
3  night
Please enter your choice:
```

操作人员根据提示进行选择，程序根据输入的时间序号显示相应的问候信息，选择 1 时显示"Good morning"， 选择 2 时显示"Good afternoon"， 选择 3 时显示"Good night"，对于其他选择显示"Selection error!"，用 switch 语句编程实现。

【参考答案】

```
#include <stdio.h>
main()
{
    char  c;

    printf("*****Time*****\n");
    printf("1 morning \n");
    printf("2 afternoon \n");
    printf("3 night \n");
    printf("please enter your choice");     /*建立相应的菜单 */

    c = getchar();                          /*输入选项*/
    switch (c)                              /*通过switch选择 */
    {
        case '1':
                printf("Good morning \n");
                break;
        case '2':
                printf("Good afternoon \n");
                break;
        case '3':
                printf("Good night\n");
                break;
        default:
                printf("Selection error!\n");
    }
}
```

程序的 2 次测试结果如下：

①
```
*****Time*****
1  morning
2  afternoon
3  night
Please enter your choice:1↙
Good moning
```

②
```
*****Time*****
1  morning
2  afternoon
3  night
Please enter your choice:3↙
Good night
```

4.9 读入一个年份和月份，打印出该月有多少天（考虑闰年），用 switch 语句编程。[提示：闰年的 2 月有 29 天，平年的 2 月有 28 天。]

【参考答案】

```
#include <stdio.h>
main()
{
    int year, month;

    printf("Input year,month: ");
    scanf("%d, %d", &year, &month); /*输入相应的年和月*/
    switch (month)
    {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:
                printf("31 days\n");
                break;
        case 2:
                if((year % 4== 0 && year % 100 != 0)||(year % 400 == 0))
                {
                    printf("29 days\n");   /*闰年的2月有29天 */

                }
                else
                {
                   printf("28 days\n");   /*平年的2月有28天 */
                }
                break;
        case 4:
        case 6:
        case 9:
        case 11:
                printf("30 days\n");
                break;
        default:
                printf("Input error!\n");
    }
}
```

程序的 3 次测试结果如下：

① Input year,month: 1988,5↙
31 days

② Input year,month: 1988,2↙
29 days

③ Input year,month: 1989,2↙
28 days

4.10　编程计算 1+3+5+7+…+99+101 的值。

【参考答案】

算法 1　利用 for 循环语句实现，在循环体外为 sum 赋初值 0。流程图如图 1-4 所示。

```
#include <stdio.h>
main()
{
    int i, sum = 0;

    for (i = 1; i <= 101; i = i+2)
    {
        sum = sum + i;
    }

    printf("sum=%d\n", sum);
}
```

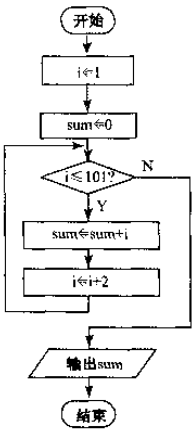

图 1-4　习题 4.10 的流程图

算法 2　利用 while 循环语句实现，在循环体外为 i 和 sum 赋初值。

```
#include <stdio.h>
main()
{
    int i = 1, sum = 0;

    while (i <= 101)
    {
        sum = sum + i;
        i = i + 2;
    }

    printf("sum=%d\n", sum);
}
```

程序运行结果：

```
sum=2601
```

4.11　编程计算 1*2*3+3*4*5+…+99*100*101 的值。

【算法思想】　用累加和算法，通项公式为 term=i*(i+1)*(i+2)；i=1,3,…,99；或者公式为 term=(i-1)*i*(i+1)；i=2,4,…,100；步长为 2。流程图如图 1-5 所示。

【参考答案】

```
#include <stdio.h>
main()
{
    int i ;
    long term, sum = 0;

    for (i = 1; i <= 99; i = i + 2)
    {
        term = i * (i + 1) * (i + 2);
        sum = sum + term;
    }
    printf("sum=%ld",sum);
}
```

程序运行结果：

```
sum=13002450
```

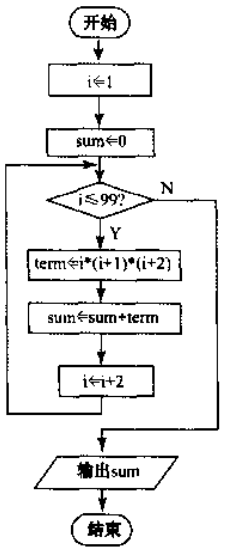

图 1-5　习题 4.11 的流程图

4.12　编程计算 1!+2!+3!+4!+…+10！的值。

【参考答案】

算法 1　用累加和算法，累加项为 term=term*i;i=1，2，…，10。term 初值为 1，使用单重循环完成。流程图如图 1-6 所示。

```
#include <stdio.h>
main()
{
    long term = 1,sum = 0;
    int i;

    for (i = 1; i <= 10; i++)
    {
        term = term * i;
        sum = sum + term;
    }

    printf("1!+2!+...+10! = %ld \n", sum);
}
```

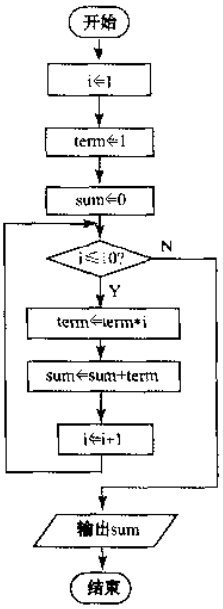

图 1-6　习题 4.12 的流程图

算法 2　用内层循环求阶乘，外层循环控制累加的项数。

```
#include <stdio.h>
main()
{
    long term ,sum = 0;
    int i, j;

    for (i = 1; i <= 10; i++)
    {
        term = 1;
        for (j = 1; j <= i; j++)
        {
            term = term * j;
        }
        sum = sum + term;
    }

    printf("1!+2!+…+10! = %ld \n", sum);
}
```

程序运行结果：

```
1!+2!+…+10! = 4037913
```

4.13　编程计算 a+aa+aaa+…+aa…a（n 个 a）的值，n 和 a 的值由键盘输入。

【算法思想】　用累加和算法，累加项为 term=term*10+a; i=1，2，…，n；term 初值为 0。流程图如图 1-7 所示。

【参考答案】

```
#include <stdio.h>
main()
{
    long term = 0,sum = 0;
    int a , i, n;

    printf("Input a,n: ");
    scanf("%d,%d", &a, &n);      /*输入a,n的值*/

    for (i = 1; i <= n; i++)
    {
        term = term * 10 + a;    /*求出累加项*/
        sum = sum + term;        /*进行累加*/
    }
    printf("sum=%ld\n",sum);
}
```

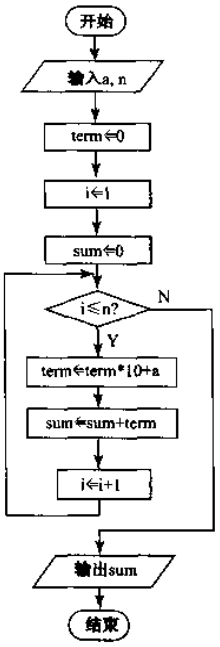

图 1-7　习题 4.13 的流程图

程序运行结果：

```
Input a,n:2,4↙
sum=2468
```

4.14　利用 $\frac{\pi}{2}=\frac{2}{1}\times\frac{2}{3}\times\frac{4}{3}\times\frac{4}{5}\times\frac{6}{5}\times\frac{6}{7}\times\cdots$ 的前 100 项之积计算π的值。

【参考答案】

算法 1　采用累乘积算法，累乘项为 term=n*n/((n-1)*(n+1))；n=2,4,…,100；步长为 2。流程图如图 1-8 所示。

```
#include <stdio.h>
main()
{
    float term, result = 1;  /*累乘项初值应为1*/
    int n;

    for (n = 2; n <= 100; n = n + 2)
    {
        term = (float)( n * n)/(( n - 1) * ( n + 1));
        result = result * term;
    }
    printf("result = %f\n", 2*result);
}
```

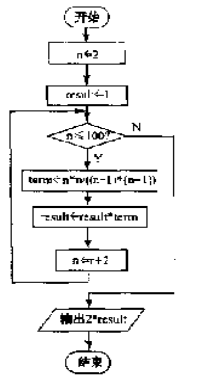

图 1-8　习题 4.14 的流程图

算法 2　采用累乘积算法，累乘项为 term=2*n*2*n/((2*n-1)*(2*n+1)) ;n=1,2,…,50；步长为 1。

```
#include <stdio.h>
main()
{
    float result = 1,term;
    int n;

    for (n = 1; n <= 50; n++)
    {
        term = (float)(2*n*2*n)/((2*n-1)*(2*n+1));  /*计算累乘项*/
        result = result * term;
    }
    printf("result = %f\n", 2*result);
}
```

程序运行结果：

```
result = 3.126078
```

4.15 利用泰勒级数 $e=1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\cdots+\frac{1}{n!}$，计算 e 的近似值。当最后一项的绝对值小于 10^{-5} 时认为达到了精度要求，要求统计总共累加了多少项。

【参考答案】

```
#include <math.h>
main()
{
    int n = 1, count =1;
    float e = 1.0, term = 1.0;
    while (fabs(term) >= 1e-5)
    {
        term = term / n;
        e = e + term;
        n++;
        count++;
    }
    printf("e = %f, count = %d\n", e, count);
}
```

程序运行结果：

```
e = 2.178282, count = 10
```

4.16 计算 $1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\cdots+\frac{1}{99}-\frac{1}{100}+\cdots$，直到最后一项的绝对值小于 10^{-4} 为止。

【参考答案】

```
#include <stdio.h>
#include <math.h>
main()
{
    int n = 1;
    float term = 1.0, sign = 1,sum = 0;
    while (fabs(term) >= 1e-4)  /*判断末项大小*/
    {
        term = sign / n;        /*求出累加项*/
        sum = sum + term;       /*累加*/
        sign = -sign;           /*改变项的符号*/
        n++;                    /*分母加1*/
    }
    printf("sum = %f\n", sum);
}
```

程序运行结果：

```
sum = 0.693092
```

4.17　利用泰勒级数 $\sin(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots$，计算 sin（$x$）的值。要求最后一项的绝对值小于 10^{-5}，并统计出此时累加了多少项。

【参考答案】

```
#include <math.h>
main()
{
    int n = 1,count = 0;
    float x;
    double sum , term;   /*因为位数多，所以定义为双精度 */
    printf("Input x: ");
    scanf("%f", &x);
    sum = x;
    term = x;          /*赋初值*/
    do
    {
        /*计算相应项，并改相应符号*/
        term = -term * x * x / ((n + 1) * (n + 2));
        sum = sum + term;            /*累加 */
        n = n + 2;
        count++;
    }while (fabs(term) >= 1e-5);

    printf("sin(x) = %f, count = %d\n", sum, count);
}
```

程序运行结果：

```
Input x:3↙（输入的值为弧度）
sin(x) = 0.141120, count = 8
```

4.18　打印所有的“水仙花数”。所谓“水仙花数”是指一个三位数，其各位数字的立方和等于该数本身。例如，153 是“水仙花数”，因为 $153=1^3+3^3+5^3$。

【参考答案】

```
#include <stdio.h>
main()
{
    int i, j, k, n;
    printf("result is:");
    for (n = 100; n < 1000; n++)
    {
        i = n / 100;                  /*分出百位*/
        j = (n - i * 100) / 10;       /*分出十位*/
        k = n % 10;                   /*分出个位*/
        if (i*100 + j*10 + k == i*i*i + j*j*j + k*k*k)
        {
            printf("%d\t ",n);       /*输出结果*/
        }
    }
    printf("\n");
}
```

程序运行结果：

```
result is:153  370  371  407
```

4.19　从键盘任意输入一个 4 位数 x，编程计算 x 的每一位数字相加之和（忽略整数前的正负号）。例如，输入 x 为 1234，则由 1234 分离出其千位 1、百位 2、十位 3、个位 4，然后计算 1+2+3+4=10，并输出 10。

【参考答案】

```
#include <stdio.h>
#include <math.h>
main()
{
    int i1, i2, i3, i4, k, n;
    printf("Input data is:");
    scanf("%d", &n);
    k = fabs(n);              /*取绝对值*/
    i1 = k / 1000;            /*分离出千位*/
    i2 = (k - i1 * 1000) / 100;     /*分离出百位*/
    i3 = (k - i1 * 1000 - i2 * 100) / 10;   /*分离出十位*/
    i4 = k % 10;              /*分离出个位*/
    printf("The sum of the total bit is %d\n", i1+i2+i3+i4);
}
```

程序的 2 次测试结果如下：

```
① Input data is :1234↙
  The sum of the total bit is 10
② Input data is :-5678↙
  The sum of the total bit is 26
```

4.20　韩信点兵。韩信有一队兵，他想知道有多少人，便让士兵排队报数。按从 1 至 5 报数，最末一个士兵报的数为 1；按从 1 至 6 报数，最末一个士兵报的数为 5；按从 1 至 7 报数，最末一个士兵报的数为 4；最后再按从 1 至 11 报数，最末一个士兵报的数为 10。你知道韩信至少有多少兵吗？

【算法思想】　设兵数为 x，则按题意 x 应满足下述关系式：

```
x%5 ==1 && x%6==5 &&x %7==4 && x%11==10
```

采用穷举法对 x 从 1 开始试验，可得到韩信至少有多少兵。流程图如图 1-13 所示。

【参考答案】

```
#include <stdio.h>
main()
{
    int  x = 1
    int find = 0;              /*设置找到标志为假*/

    while (!find)
    {
        if (x % 5 == 1 && x % 6 == 5 && x % 7 == 4 && x % 11 == 10)
        {
            find = 1;
        }
        x++;
    }
    printf(" x = %d\n", x);
}
```

程序运行结果：

```
x = 2112
```

4.21　爱因斯坦数学题。爱因斯坦曾出过这样一道数学题：有一条长阶梯，若每步跨 2 阶，最后剩下 1 阶；若每步跨 3 阶，最后剩下 2 阶；若每步跨 5 阶，最后剩下 4 阶；若每步跨 6 阶，最后剩下 5 阶；只有每步跨 7 阶，最后才正好 1 阶不剩。请问，这条阶梯共有多少阶？

【参考答案】　采用穷举法对 x 从 1 开始试验，可计算出这条阶梯共有多少阶。

```
main()
{
    int  x = 1, find = 0;
    while (!find)
    {
        if (x % 2 == 1 && x % 3 == 2 && x % 5 == 4 && x % 6 == 5
            && x % 7 == 0)
        {
            find = 1;
        }
        x++;
    }
    printf(" x = %d\n", x);
}
```

程序运行结果：

```
x = 120
```

4.22　三色球问题。若一个口袋中放有 12 个球，其中有 3 个红色的，3 个白色的，6 个黑色的，从中任取 8 个球，问共有多少种不同的颜色搭配？

【算法思想】　设任取的红球个数为 i，白球个数为 j，黑球个数为 k。根据题意，红、白黑球个数的取值范围应分别为 0<=i<=3，0<=j<=3，0<=k<=6。只要满足 I+j+k=8，则 i，j，k 的组合即为所求。流程图如图 1-15 所示。

【参考答案】

```
#include <stdio.h>
main()
{
    int i, j, k;
    for (i = 0; i <= 3; i++)
    {
       for (j = 0; j <= 3; j++)
       {
           for (k = 0; k <= 6; k++)
           {
               if (i + j + k == 8)
               {
                   printf("i=%d, j=%d, k=%d\n", i, j, k);
               }
           }
       }
    }
}
```

程序运行结果：

```
i=0,j=2,k=6
i=0,j=3,k=5
i=1,j=1,k=6
i=1,j=2,k=5
i=1,j=3,k=4
i=2,j=0,k=6
i=2,j=1,k=5
i=2,j=2,k=4
i=2,j=3,k=3
i=3,j=0,k=5
i=3,j=1,k=4
i=3,j=2,k=3
i=3,j=3,k=2
```

4.23　鸡兔同笼，共有98个头，386只脚，编程求鸡、兔各多少只。

【算法思想】　设鸡数为x，兔数为y，据题意有x+y=98，2x+4y=386。采用穷举法，x从1变化到97，y取98-x，如果x，y同时满足条件2x+4y=386，则打印x，y的值。

【参考答案】

```
#include <stdio.h>
main()
{
   int x, y;

   for (x = 1; x <= 97; x++)
   {
       y = 98 - x;
       if (2 * x + 4 * y == 386)
       {
           printf("x = %d, y = %d", x, y);
       }
   }
}
```

程序运行结果：

```
x = 3, y = 95
```

4.24　我国古代的《张丘建算经》中有这样一道著名的百鸡问题："鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一。百钱买百鸡，问鸡翁、母、雏各几何？"其意为：公鸡每只5元，母鸡每只3元，小鸡3只1元。用100元买100只鸡，问公鸡、母鸡和小鸡各能买多少只？

【算法思想】　设公鸡、母鸡、小鸡数分别为x，y，z，依题意列出方程组x+y+z=100，5x+3y+z/3=100，采用穷举法求解，因100元买公鸡最多可买20只，买母鸡最多可买33只，所以，x从0变化到20，y从0变化到33，则z=100-x-y，只要判断第2个条件是否满足即可。

【参考答案】

```
#include <stdio.h>
main()
{
    int x, y, z;
    for (x = 0; x <= 20; x++)
    {
        for (y = 0; y <= 33; y++)
        {
            z = 100 - x - y;
            if (5*x + 3*y + z/3.0 == 100)
            {
                printf("x=%d, y=%d, z=%d\n", x, y, z);
            }
        }
    }
}
```

程序运行结果：

```
x=0,y=25,z=75
x=4,y=18,z=78
x=8,y=11,z=81
x=12,y=4,z=84
```

4.25　用1元5角钱人民币兑换5分、2分和1分的硬币（每一种都要有）共100枚，问共有几种兑换方案？每种方案各换多少枚？

【算法思想】 设5分、2分和1分的硬币各换x，y，z枚，依题意有x+y+z=100，5x+2y+z=150，由于每一种硬币都要有，故5分硬币最多可换28枚，2分硬币最多可换73枚，1分硬币可换100-x-y枚，x，y，z只需满足第2个方程即可打印，对每一组满足条件的x，y，z值用计数器计数即可得到兑换方案的数目。

【参考答案】

```
#include <stdio.h>
main()
{
    int x, y, z, count = 0;
    for (x = 1; x <= 28; x++)
    {
        for (y = 1; y <= 73; y++)
        {
            z = 100 - x - y;
            if (5*x + 2*y + z == 150)
            {
                count++;
                printf("%d, %d, %d\n", x, y, z);
            }
        }
    }
    printf("count = %d\n", count);
}
```

程序运行结果：

```
1,46,53
2,42,56
3,38,59
4,34,62
5,30,65
6,26,68
7,22,71
8,18,74
9,14,77
10,10,80
11,6,83
12,2,86
count = 12
```

4.26　编程输出如下上三角形式的九九乘法表。

【算法思想】　根据题意，第 1 行打 9 列，第 2 行打 8 列，……，最后一行打 1 列，为了打印右上三角形式，需要在第 n 列先打印 $4*n-4$ 列空格，再打印两数相乘结果。

【参考答案】

```
#include <stdio.h>
main()
{
    int m, n, i;
    for (m = 1; m < 10; m++)
    {
        printf("%4d", m);
    }           /*打印表头*/
    printf("\n");
    for (m = 1; m < 10; m++)
    {
        printf("   -");
    }
    printf("\n");
    for (n = 1; n <= 9; n++)                    /*被乘数n从1变化到9*/
    {
        for (i = 1; i <= 4*n-4; i++)
        {
            printf(" ");                        /*输出相应空格使数字右对齐*/
        }
        for (m = n; m < 10; m++)                /*乘数m从n变化到9*/
        {
            printf("%4d", m * n);               /*输出第m行n列中的m*n的值*/
        }
        printf("\n");                           /*输出换行符,准备打印下一行*/
    }
}
```

4.27　编程打印以下图案。

（1）

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

（2）

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

（3）

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

【参考答案】　（1）

```
#include <stdio.h>
main()
{
    int i, j, k;
    for (i = 1; i< = 4; i++)            /*i控制行数*/
    {
        for (j = 1; j <= 4-i; j++)  /* 随行数的增加，输出递减数目的空格*/
        {
            printf(" ");
        }
        for (k = 1; k <= 6; k++)        /*每行输出6个*字符*/
        {
            printf("*");
        }
        printf("\n");                   /*将光标移到下一行起始位置处*/
    }
}
```

【参考答案】　（2）

```
#include <stdio.h>
main()
{
    int  i , j , k;
    for (i = 1; i <= 4; i++)                /*控制行数*/
    {
        for (k = 1; k <= (2 * i - 1); k++) /*控制每行输出的*号个数*/
        {
            printf("*");
        }
        printf("\n");                       /*输出一行后换行*/
    }
}
```

【参考答案】　（3）

```
#include <stdio.h>
main()
{
    int  i , j, k ;
    for (i = 1; i <= 4; i++)            /* 控制行数*/
    {
        for (j = 1; j <= 8-i; j++)  /* 随行数的增加，输出递减数目的空格*/
        {
            printf(" ");
        }
        for (k = 1;k <= (2*i-1);k++)
        {
            printf("*");
        }
        printf("\n");
    }
```

4.28　按如下格式打印 100 以内整数的平方根表。

	0	1	2	3	4	5	6	7	8	9
0	0.000	1.000	1.414	1.732	2.000	2.236	2.449	2.646	2.828	3.000
1	3.162	3.317	3.464	3.606	3.742	3.873	4.000	4.123	4.243	4.359
2	4.472	4.583	4.690	4.796	4.899	5.000	5.099	5.196	5.292	5.385
3	5.477	5.568	5.657	5.745	5.831	5.916	6.000	6.083	6.164	6.245
4	6.325	6.403	6.481	6.557	6.633	6.708	6.782	6.856	6.928	7.000
5	7.071	7.141	7.211	7.280	7.348	7.416	7.483	7.550	7.616	7.681
6	7.746	7.810	7.874	7.937	8.000	8.062	8.124	8.185	8.246	8.307
7	8.367	8.426	8.485	8.544	8.602	8.660	8.718	8.775	8.832	8.888
8	8.944	9.000	9.055	9.110	9.165	9.220	9.274	9.327	9.381	9.434
9	9.487	9.539	9.592	9.644	9.695	9.747	9.798	9.849	9.899	9.950

【算法思想】　表内第 1 行打印 0~9 的平方根，第 2 行打印 10~19 的平方根，第 3 行打印 20~29 的平方根，……，第 10 行打印 90~99 的平方根。设表的行数为 x，表的列数为 y，则表内对应第 x 行第 y 列的表值为 sqrt(x*10+y)。

【参考答案】

```
#include <stdio.h>
#include <math.h>
main()
{
    int  m, n, i;
    for (m = 0; m < 10; m++)
    {
        printf("%7d", m);                    /*打印表头*/
    }
    printf("\n");
    for (n = 0; n < 10; n++)                 /*乘数n从1变化到9*/
    {
        printf("%d", n);                     /*输出每行的开头数字*/
        for (m = 0; m < 10; m++)                 /*被乘数m从1变化到9*/
        {
            printf(" %4.3f ", sqrt(n * 10 + m));
        }                                        /*输出第m行n列中的值*/
        printf("\n");                        /*输出换行符,准备打印下一行*/
    }
}
```

1.5 习题5及参考答案

5.1 多项选择题 【参考答案】 （1）ACD （2）BC

5.2 指出下面这段程序中的错误。

【参考答案】

（1）Delay()未先声明或定义就使用 （2）没有参数的函数应该注明 void

（3）变量 times 未赋值就使用 （4）没有 return 语句

5.3 写出下面程序的执行结果。【参考答案】 2，6，42，3

5.4 设计一个函数，判断一个整数是否为素数。

【参考答案】

```
#include <math.h>
/*
   函数功能:      判断参数是否是素数
   函数入口参数:  整型数，要求为正整数
   函数返回值:    非0值表示是素数，否则不是素数
*/
int IsPrimeNumber(int number)
{
    int i;
    if (number <= 1)     /* 负数、0和1都不是素数 */
        return 0;
    for (i=2; i<sqrt(number); i++)
    {
        if ((number % i) == 0)  /* 被整除，不是素数 */
            return 0;
    }
    return 1;
}
```

5.5 编程计算 $p=\frac{k!}{(m-k)!}$ 的值。

【算法思想】 此公式中用到两次阶乘，所以可以把阶乘设计为一个函数。因为负数没有阶乘，所以此函数的参数应该采用无符号类型。而 0 的阶乘为 1，这一点在编程时要注意。阶乘的值都非常大，所以用无符号长整型作为返回值。p 的运算结果可能是浮点数，所以 p 定义为 double 类型。

【参考答案】

```
#include <stdio.h>
unsigned long Factorial(unsigned int number);
main()
{
    unsigned int m, k;
    double p;
    printf("Please input m, k:");
    scanf("%u, %u", &m, &k);
    p = (double)Factorial(k) / Factorial (m-k);
    printf("p=%f\n", p);
}
/*
   函数功能:      计算参数的阶乘
   函数入口参数:  无符号整型
   函数返回值:    运算结果
*/
unsigned long Factorial(unsigned int number)
{
    unsigned long i, result = 1;
    for (i=2; i<=number; i++)
        result *= i;
    return result;
}
```

5.6 设计函数 MaxCommonFactor()，计算两个正整数的最大公约数。

【算法思想】 对于 a 和 b 两个数，当 a>b 时，如果 a 中含有与 b 相同的公约数，则 a 中去掉 b 后剩余的部分 a-b 中也应该含有与 b 相同的公约数，对 a-b 和 b 计算公约数就相当于对 a 和 b 计算公约数。反复使用最大公约数的 3 个性质，直到 a 和 b 相等为止，这时 a 或 b 就是它们的最大公约数。

【参考答案】

```
/*
  函数功能:      计算两个正整数的最大公约数
  函数入口参数:  两个整型数
  函数返回值:    最大公约数; -1表示没有最大公约数
*/
int MaxCommonFactor(int a, int b)
{
    if (a <=0 || b <=0)      /* 保证输入的参数正确 */
        return -1;
    while(a != b)
    {
        if (a > b)
        {
            a = a - b;
        }
        else if (b > a)
        {
            b = b - a;
        }
    }
    return a;
}
```

5.7 用下面给定的代码调用，实现函数 int CommonFactors(int a, int b)，计算 a 和 b 的所有公约数。第一次调用，返回最大公约数。以后只要再使用相同参数调用，每次返回下一个小一些的公约数。无公约数时返回-1。

【算法思想】 只要用户这次输入的参数与上次的不同，就重新开始计算。算法采用穷举法，用静态变量实现。

【参考答案】

```
#include <stdio.h>
int CommonFactors(int a, int b);
main()
{
    int sub;
    while((sub=CommonFactors(100, 50)) > 0)
    {
        static int counter=1;
        printf("Common factor %d is %d\n", counter++, sub);
    }
}
/*
   函数功能:      指明计算哪两个数的公约数
   函数入口参数:  两个整型数
   函数返回值:    公约数
*/
int CommonFactors(int a, int b)
{
    static int num1=-1;
    static int num2=-1;
    static int curFactor;

    if (a < 0 || b < 0)
        return -1;

    if (num1 != a || num2 != b)     /* 使用了新的参数 */
    {
        num1 = a;
        num2 = b;
        curFactor = a > b ? b : a;  /* curFactor置为两个数中较小的那个 */
    }

    while(curFactor>0)
    {
        if ( a%curFactor == 0 && b%curFactor == 0 )
        {
            return curFactor--;     /* 如果不减1，则下次还会测试这个数 */
        }
        curFactor--;
    }

    return -1;
}
```

5.8 设计一个模块 cmnfctr，计算给定的两个整数的所有公约数。CalcCommonFactorOf()用来设定参与计算的两个整数，然后每调用一次 NextCommonFactor()得到一个公约数，按照从大到小的顺序给出。用下面给定的代码调用此模块。

【算法思想】 CalcCommonFactorOf()把参数保存到模块自己的全局变量内，由NextCommonFactor()使用。NextCommonFactor()采用从大到小逐个整数试验的方法寻找公约数，每次调用都是从上一次调用找到的公约数减 1 的地方开始继续寻找。

【参考答案】

cmnfctr.c 如下：

```
static int a, b;
static int curFactor;

/*
   函数功能:      指明计算哪两个数的公约数
   函数入口参数:  两个整型数
   函数返回值:    无
*/
void CalcCommonFactorOf(int num1, int num2)
{
    a = num1;
    b = num2;
    curFactor = a > b ? b : a;      /* curFactor置为两个数中较小的那个 */
}

/*
   函数功能:      得到下一个公约数
   函数入口参数:  无
   函数返回值:    下一个公约数；-1表示再也没有新的公约数
*/
int NextCommonFactor(void)
{
    if (a <=0 || b <=0)     /* 保证输入的参数正确 */
        return -1;

    while(curFactor>0)
    {
        if ( a%curFactor == 0 && b%curFactor == 0 )
        {
            return curFactor--;     /* 如果不减1，则下次还会测试这个数 */
        }
        curFactor--;
    }

    return -1;
}
```

cmnfctr.h 如下：

```
void CalcCommonFactorOf(int num1, int num2);
int NextCommonFactor(void);
```

调用代码如下：

```
#include <stdio.h>
#include "cmnfctr.h"
main()
{
    int sub;
    CalcCommonFactorOf(100, 50);
    while((sub=NextCommonFactor()) > 0)
    {
        static int counter=1;
        printf("Common factor %d is %d\n", counter++, sub);
    }
}
```

1.6 习题6及参考答案

6.1 选择题

【参考答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
答案	C	B	B	B	D	A	A	A	B

6.2 阅读程序，写出运行结果。

【参考答案】 (1) 运行结果：

```
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
```

【参考答案】 (2) 运行结果：10

【参考答案】 (3) 运行结果：0 1 2 3

6.3 阅读程序，按要求在空白处填写适当的表达式或语句，使程序完整并符合题目要求。

【参考答案】 (1)

① rand()*6.0/32767 + 1; /*利用随机函数产生1-6之间的随机数*/

② frequency[face]; /*计数器分类计数*/

【参考答案】 (2)

① a[n] > max /*用数组中的数据与数列中最大数比较*/

② n /*记载最大数在数列中的位置*/

③ a[n] < min /*用数组中的数据与数列中最小数比较*/

④ n /*记载最小数在数列中的位置*/

【参考答案】 (3)

① !='\0' /*判断字符串是否结束*/

② != ' ' /*判断当前字符不为空格*/

③ ==' ' /*判断前一个字符是空格*/

【参考答案】 (4)

① s[i] != '\0' /*判断字符串是否结束*/

② s[j] = s[i] /*将与c不同的字符存回原数组*/

【参考答案】 (5)

① '\0' /*判断字符串是否结束*/

② s[i] - t[i] /*返回第一个不同字符的ASCII码值之差*/

【参考答案】 (6)

① int * /*定义一个能传回结果的整型指针变量*/

② *z /间接访问变量z，改变相应值/

6.4 编程实现从键盘任意输入20个整数，统计非负数个数，并计算非负数之和。

【参考答案】

```
#include <stdio.h>
main()
{
    int i, n, sum = 0, counter = 0;
    printf("Input 20 Numbers:\n");
    for (i=0; i < 20; i++)
    {
        scanf("%d", &n);
        if (n >= 0)                    /*判断是否为非负数*/
        {
            sum += n;                  /*非负数求和*/
            counter++;                 /*非负数个数计数*/
        }
    }
    printf("sum=%d,counter=%d\n", sum,counter);
}
```

程序运行结果：

```
Input 20 Numbers:
12  -45  56  -99  34  87  90  -23  0  65 ↙
11  53  67  -88  -51  25  78  45  123  78↙
sum=824,counter=15
```

6.5 从键盘任意输入 10 个整数，用函数编程实现将其中最大数与最小数的位置对换后，再输出调整后的数组。

【参考答案】

```
#include <stdio.h>
#define ARR_SIZE 10
/*  函数功能:  找出n个数中的最大数与最小数并将其位置对换
    函数参数:  整型数组a，存放待处理数据
               整型变量n，代表数据个数
    函数返回值: 无
*/
void MaxMinExchange(int a[], int n)
{
    int maxValue = a[0], minValue = a[0], maxPos = 0, minPos = 0;
    int i, temp;
    for (i=1; i<n; i++)
    {
        if (a[i] > maxValue)
        {
            maxValue = a[i];
            maxPos = i;
        }
        if (a[i] < minValue)
        {
            minValue = a[i];
            minPos = i;
        }
    }
    temp = a[maxPos];
    a[maxPos] = a[minPos];
    a[minPos] = temp;
}
main()
{
    int a[ARR_SIZE], i;

    printf("Input Numbers (n<10):\n");
    for (i=0; i<ARR_SIZE; i++)
    {
        scanf("%d",&a[i]);
    }

    MaxMinExchange(a, i);

    for (i=0; i<ARR_SIZE; i++)
    {
        printf("%4d", a[i]);
    }
}
```

程序运行结果：

```
Input Numbers (n<10):
12  45  23  0  -22  45  67  33  99  11 ↙
12  45  23  0  99  45  67  33  -22  11
```

6.6 输入5×5阶的矩阵，编程实现：

（1）求两条对角线上的各元素之和。

（2）求两条对角线上行、列下标均为偶数的各元素之积。

【参考答案】

```
#include <stdio.h>
#define ARR_SIZE 5
main()
{
   int     a[ARR_SIZE][ARR_SIZE], i, j, sum = 0;
   long    product = 1;

   for (i=0; i<ARR_SIZE; i++)
   {
       for (j=0; j<ARR_SIZE; j++)
       {
           scanf("%d",&a[i][j]);
       }
   }
   for (i=0; i<ARR_SIZE; i++)
   {
       for (j=0; j<ARR_SIZE; j++)
       {
           if (i == j || i+j == ARR_SIZE-1)
               sum += a[i][j];
           if ((i == j || i+j == ARR_SIZE-1) && i%2 == 0 && j%2 == 0)
               product *= a[i][j];
       }
   }
   printf("sum = %d\n product = %ld\n", sum, product);
}
```

程序运行结果：

```
1  2  3  4  5 ↙
2  3  4  5  6 ↙
3  4  5  6  7 ↙
4  5  6  7  8 ↙
5  6  7  8  9 ↙
sum = 45
product = 1125
```

6.7 编程打印如下形式的杨辉三角形。

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

【算法思想】 用二维数组存放杨辉三角形中的数据，这些数据的特点是：第 0 列全为 1，对角线上的元素全为 1，其余的左下角元素 a[i][j] = a[i-1][j-1] + a[i-1][j]，用数组元素作为函数参数编程实现计算，并存放这些元素的值。

【参考答案】

```
#include<stdio.h>
#define  ARR_SIZE  11
void  YHTriangle(int a[][ARR_SIZE], int  n);
void  PrintYHTriangle(int a[][ARR_SIZE], int  n);
main()
{
    int  a[ARR_SIZE][ARR_SIZE], n;
    printf("input  n (n<=10):");
    scanf("%d",&n);     /*根据要求输入杨辉三角形的行数*/
    YHTriangle(a,n);
    PrintYHTriangle(a,n);
}
/*函数功能：  计算杨辉三角形中各元素数值
  函数参数：  整型数组a，存放计算得到的杨辉三角形数据
              整型变量n，代表杨辉三角形的行数
  函数返回值：无
*/
void YHTriangle(int a[][ARR_SIZE], int n)
{
    int  i, j ;
    for (i=1; i<=n; i++)
    {
        a[i][1] = 1;
        a[i][i] = 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];
        }
    }
}
/*  函数功能：  输出杨辉三角形
    函数参数：  整型数组a，存放杨辉三角形数据
                整型变量n，代表杨辉三角形的行数
    函数返回值：无
*/
void PrintYHTriangle(int a[][ARR_SIZE], int n)
{
    int i , j ;

    for (i=1; i<=n; i++)
    {
        for (j=1; j<=i; j++)
        {
            printf("%4d", a[i][j]);
        }
        printf("\n");
    }
}
```

程序运行结果：

```
input  n (n<=10):6↙
1
1  1
1  2   1
1  3   3   1
1  4   6   4   1
1  5   10  10  5   1
```

6.8 编程将下列矩阵中的元素向右移动一列，最右一列移至第一列。

1 4 6

8 10 12

【算法思想】 用二维数组 v 存放矩阵中元素，数组 v 可在定义时初始化；有两种方法实现这种移动：一种方法是将移动后的元素放在另一个二维数组中；另一种方法是利用一个中间变量仍将移动后的元素放在数组 v 中。

【参考答案】

```
#include<stdio.h>
#define  ROW 2
#define  COL 3
main()
{
    int a[ROW][COL] = {1,4,6,8,10,12};
    int i, j, temp;
    for (i=0; i<ROW; i++)
    {
        temp = a[i][COL-1];      /*将当前行最后一列暂存*/
        for (j=COL-2; j>=0; j--)
        {
            a[i][j+1] = a[i][j];          /*将当前行其他列后移*/
        }
        a[i][0] = temp;                  /*将暂存数据赋予当前行0列*/
    }
    for (i=0; i<ROW; i++)
    {
        for (j=0; j<COL; j++)
        {
            printf("%6d", a[i][j]);
        }
        printf("\n");
    }
}
```

程序运行结果：

```
     6     1     4
    12     8    10
```

6.9　利用公式 $c_{ij}=a_{ij}+b_{ij}$ 计算 $m\times n$ 阶矩阵 $\boldsymbol{A}$ 和 $m\times n$ 阶矩阵 $\boldsymbol{B}$ 之和。已知 a_{ij} 为矩阵 $\boldsymbol{A}$ 的元素，b_{ij} 为矩阵 $\boldsymbol{B}$ 的元素，c_{ij} 为矩阵 $\boldsymbol{C}$ 的元素，$i=1,2,\cdots,m$; $j=1,2,\cdots,n$。

【算法思想】　用二维数组元素作为函数参数编程实现矩阵相加。

【参考答案】

```
#include<stdio.h>
#define  ROW 2
#define  COL 3
/*  函数功能:   计算矩阵之和，即计算数组a，b对应位置数据相加之和，结果存于数组c中
    函数参数:   整型数组a,b，分别存放两个待求和的矩阵元素
                整型数组c，存放矩阵求和结果
    函数返回值：无
*/
void  AddMatrix(int a[ROW][COL], int b[ROW][COL], int c[ROW][COL])
{
    int i , j ;

    for (i=0; i<ROW; i++)
    {
        for (j=0; j<COL; j++)
        {
            c[i][j] = a[i][j] + b[i][j];
        }
    }
}
main()
{
    int a[ROW][COL], b[ROW][COL], c[ROW][COL], i, j;
    printf("Input array a:\n");
    for (i=0; i<ROW; i++)
    {
        for (j=0; j<COL; j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    printf("Input array b:\n");
    for (i=0; i<ROW; i++)
    {
        for (j=0; j<COL; j++)
        {
            scanf("%d",&b[i][j]);
        }
    }
    AddMatrix(a, b, c);
    printf("Results:\n");
    for (i=0; i<ROW; i++)
    {
        for (j=0; j<COL; j++)
        {
            printf("%6d",c[i][j]);
        }
        printf("\n");
    }
}
```

程序运行结果：

```
Input array a:
1  2  3 ↙
4  5  6 ↙
Input array b:
7  8  9 ↙
10  11  12 ↙
Results:
8    10    12
14   16    18
```

***6.10　利用公式 $c_{ij}=\sum_{k=1}^{n} a_{ik} * b_{kj}$ 计算矩阵 *A* 和矩阵 *B* 之积。**已知 a_{ij} 为 $m\times n$ 阶矩阵 *A* 的元素，$i=1,2,\cdots,m$；$j=1,2,\cdots,n$。b_{ij} 为 $n\times m$ 阶矩阵 *B* 的元素，$i=1,2,\cdots,n$；$j=1,2,\cdots m$。c_{ij} 为 $m\times m$ 阶矩阵 *C* 的元素，$i=1,2,\cdots,m$；$j=1,2,\cdots,m$。

【算法思想】　用二维数组元素作为函数参数编程实现矩阵相乘。在 i 和 j 的二重循环中，设置 k 的循环，进行累加和运算 c[i][j]=c[i][j]+term，累加项为 term=a[i][j]*b[i][j]。注意，c[i][j] 要在 k 循环体外的前面赋初值 0。

【参考答案】

```
#include<stdio.h>

#define  ROW 2
#define  COL 3

/* 函数功能：  计算矩阵相乘之积，结果存于数组c中
   函数参数：  整型数组a，b，分别存放两个待求乘积的矩阵元素
               整型数组c，存放矩阵相乘的结果
   函数返回值：无
*/
MultiplyMatrix(int a[ROW][COL], int b[COL][ROW], int c[ROW][ROW])
{
    int i, j, k;

    for (i=0; i<ROW; i++)
    {
        for (j=0; j<ROW; j++)
        {
            c[i][j] = 0;
            for (k=0; k<COL; k++)
            {
                c[i][j] = c[i][j] + a[i][k] * b[k][j];
            }
        }
    }
}

main()
{
    int a[ROW][COL], b[COL][ROW], c[ROW][ROW], i, j;

    printf("Input array a:\n");
    for (i=0; i<ROW ;i++)
    {
        for (j=0; j<COL; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }
```

```
    printf("Input array b:\n");
    for (i=0; i<COL; i++)
    {
        for (j=0; j<ROW; j++)
        {
            scanf("%d", &b[i][j] );
        }
    }

    MultiplyMatrix(a, b, c);

    printf("Results:\n");
     for (i=0; i<ROW; i++)
     {
        for (j=0; j<ROW; j++)
        {
            printf("%6d", c[i][j]);
        }
        printf("\n");
     }
}
```

程序运行结果：

```
Input array a:
1  2  3 ↙
4  5  6 ↙
Input array b:
7  8 ↙
9  0 ↙
1  2 ↙
Results:
    28    14
    79    44
```

6.11 输入一行字符，统计其中的英文字符、数字字符、空格及其他字符的个数。

【参考答案】

```
#include <stdio.h>
#include <string.h>
#define ARR_SIZE 80
main()
{
    char str[ARR_SIZE];
    int  len, i, letter = 0, digit = 0, space = 0, others = 0;
    printf("Please input a  string:");
    gets(str);
    len = strlen(str);
    for (i=0; i<len; i++)
    {
        if (str[i] >= 'a' && str[i] <= 'z' || str[i] >= 'A' && str[i] <= 'Z')
            letter ++;              /*统计英文字符*/
        else if (str[i] >= '0' && str[i] <= '9' )
            digit ++;               /*统计数字字符*/
        else if (str[i] == ' ' )
            space ++;            /*统计空格*/
        else
            others ++;           /*统计其他字符的个数*/
    }
    printf("English character:  %d\n", letter);
    printf("digit character:  %d\n", digit);
    printf("space:  %d\n", space);
    printf("other character:  %d\n", others);
}
```

程序运行结果：

```
Please input a string: *****c language.***** ↙
English character:  9
digit character:  0
space:  1
other character:  11
```

6.12 编写一个函数 Inverse，实现将字符数组中的字符串逆序存放的功能。

【参考答案】

方法 1 利用两个数组实现字符串的逆序存放。

```
#include <stdio.h>
#include <string.h>
#define ARR_SIZE 80
void Inverse(char str[], char ptr[]);
main()
{
    char a[ARR_SIZE], b[ARR_SIZE];
    printf("Please enter a string: ");
    gets(a);
    Inverse(a, b);
    printf("The inversed string is: ");
    puts(b);
}
/*函数功能： 实现将字符数组中的字符串逆序存放
  函数参数： 字符数组a，存放源字符串
             字符数组b，存放逆序字符串
  函数返回值：无
*/
void Inverse(char str[], char ptr[])
{
    int  i = 0, j;
    j = strlen(str) - 1;
    while (str[i] != '\0')
    {
        ptr[j] = str[i];
        i++;
        j--;
    }
    ptr[i]='\0';
}
```

方法 2 利用一个数组实现字符串的逆序存放。

```
#include<string.h>
#include<stdio.h>
#define ARR_SIZE 80
void Inverse(char str[]);
main()
{
    char a[ARR_SIZE] ;
    printf("Please enter a string: ");
    gets(a);
    Inverse(a);
    printf("The inversed string is: ");
    puts(a);
}
/*函数功能： 实现将字符数组中的字符串逆序存放
  函数参数： 字符数组a，存放源字符串，逆序后的字符串最终也被存放在此数组中
  函数返回值：无
*/
void Inverse(char str[])
{
    int   len, i, j;
    char  temp;
    len = strlen(str);
    for (i=0, j=len-1; i<j; i++, j--)
    {
        temp = str[i];
        str[i] = str[j];
        str[j] = temp;
    }
}
```

程序运行结果：

```
Please enter a string: ABCDEFGHI ↙
The inversed string is: IHGFEDCBA
```

6.13 不用 strcat 函数，编程实现字符串连接函数 strcat 的功能，将字符串 dstStr 连接到字符串 srcStr 的尾部。

【参考答案】

```
#include <stdio.h>
#include <string.h>
#define ARR_SIZE 80
void MyStrcat(char dstStr[], char srcStr[]);
main()
{
    char  s[ARR_SIZE], t[ARR_SIZE];
    printf("Please enter source string: ");
    gets(s);
    printf("Please enter destination string: ");
    gets(t);
    MyStrcat(s,t);
    printf("The concatenate string is: ");
    puts(s);
}
/*  函数功能：将源字符串srcStr中的字符串连接到目的字符串dstStr之后
    函数参数：字符数组srcStr，存放源字符串
              字符数组srcStr，存放目的字符串
              源字符串srcStr和目的字符串dstStr连接后的字符串也存于此数组中
    函数返回值：无
*/
void MyStrcat(char dstStr[], char srcStr[])
{
    unsigned int  i, j;
    i = strlen(dstStr);              /*将下标移动到目的字符串末尾*/
    for (j=0; j<=strlen(srcStr); j++, i++)
    {
       dstStr[i] = srcStr[j];
    }
}
```

MyStrcat 函数的第 2 种实现方法。

```
void MyStrcat(char dstStr[], char srcStr[])
{
    int i = 0, j;
    while (dstStr[i] != '\0')  /*将下标移动到目的字符串末尾*/
    {
        i++;
    }
    for (j=0; srcStr[j]!='\0'; j++, i++)
    {
        dstStr[i] = srcStr[j];
    }
    dstStr[i] = '\0';
}
```

程序运行结果：

```
Please enter source string: Hello ↙
Please enter destination string: China! ↙
The concatenate string is: Hello China!
```

1.7 习题7及参考答案

7.1 选择题

【参考答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	A	D	D	A	D

7.2 阅读程序，写出运行结果。

【参考答案】

(1) 6

(2) 1,2,3,3,2,3,4,4

(3) Program
PROGRAM i=7
margor
gram

(4) demo.exe
This
is
a
program

7.3 阅读程序，按要求在空白处填写适当的表达式或语句，使程序完整并符合题目要求。

【参考答案】 (1) ① '\0' ② ++ ③ len

【参考答案】 (2) ① '\0' ② (p-s)

【参考答案】 (3) ① 0 ② (*p1-*p2)

7.4 在下面使用指针数组的程序中存在一个错误，试分析这个程序，并上机运行，观察运行结果，找到这个错误，并分析出错的原因。

【参考答案】 错误之处在于求指针数组的大小上。指针数组的元素类型是字符指针，需要的字节数为 sizeof(char*)，不是 sizeof(char)。正确程序如下：

```
#include <stdio.h>
void Print(char *arr[], int len);
void main()
{
    char *pArray[] = {"Fred", "Barrey", "Wilma", "Betty"};
    int num = sizeof(pArray) / sizeof(char*);  /* 修改的语句*/
    Print(pArray, num);
}
void Print(char *arr[], int len)
{
    int i;
    for (i = 0;i < len;i++)
    {
        printf("%s\n", arr[i]);
    }
}
```

7.5 编写一个交换变量值的函数，利用该函数交换数组 a 和数组 b 中的对应元素值。

【参考答案】

```
#include <stdio.h>
#define ARRAY_SIZE 10
void  Swap(int *x, int *y);
main()
{
    int  a[ARRAY_SIZE], b[ARRAY_SIZE], i, n;
    printf("Input array length n<=10: ");
    scanf("%d", &n);
    printf("Input array a:\n");
    for (i = 0; i < n; i++)
    {
        scanf("%d ", &a[i]);
    }
    printf("Input array b:\n");
    for (i = 0; i < n; i++)
    {
        scanf("%d ", &b[i]);
    }                                   /*输入两个数组内容*/
    for (i = 0; i < n; i++)
    {
        Swap(&a[i], &b[i]);             /*调用交换函数*/
    }
    printf("After swap:\n");
    printf("Array a:\n");
    for (i = 0; i < n; i++)
    {
        printf("%d ", a[i]);
    }
    printf("\n");
    printf("Array b:\n");
    for (i = 0; i < n; i++)
    {
        printf("%d ", b[i]);
    }
    printf("\n");
}
/* 函数功能:   交换两个整型数的值
   函数参数:   整型指针x和y分别指向两个待交换的整型数
   函数返回值: 无
*/
void Swap(int *x, int *y)    /*交换函数*/
{
    int  temp;
    temp = *x;
    *x = *y;
    *y = temp;
}
```

程序运行结果：

```
Input array length n<=10: 5↙
Input array a:
1 3 5 7 9↙
Input array b:
2 4 6 8 10↙
After swap:
Array a:
2 4 6 8 10
Array b:
1 3 5 7 9
```

7.6 从键盘任意输入 10 个整数，用函数编程实现计算最大值和最小值，并返回它们所在数组中的位置。

【算法思想】 用指针变量作为函数参数，得到最大值和最小值及在数组中的位置。

【参考答案】

```
#include <stdio.h>
int FindMax(int num[], int n, int *pMaxPos);
int FindMin(int num[], int n, int *pMinPos);    /*函数声明*/
main()
{
    int num[10], maxValue, maxPos, minValue, minPos, i;
    printf("Input 10 numbers:\n ");
    for (i = 0; i <= 10; i++)
    {
        scanf("%d", &num[i]);                   /* 输入10个数*/
    }
    maxValue = FindMax(num, 10, &maxPos);  /* 找最大值及位置 *
    minValue = FindMin(num, 10, &minPos);  /* 找最小值及位置 *
    printf("Max=%d, Position=%d, Min=%d, Position=%d\n",
            maxValue, maxPos, minValue, minPos);
}
/*  函数功能:       求n个数中的最大值
    函数入口参数:   整型数组num，存放n个整数
                    整型变量n，表示数组元素个数
    函数出口参数:   整型指针变量pMaxPos，指向存放最大值的整型变量
    函数返回值:     最大值
*/
int FindMax(int num[], int n, int *pMaxPos)
{
    int i, max;
    max = num[0];              /*假设num[0]为最大*/
    *pMaxPos = 0;              /*给指针赋初值 */
    for (i = 1; i < n; i++)
    {
        if (num[i] > max)
        {
            max = num[i];
            *pMaxPos = i;
        }
    }
    return max;         /*返回最大值*/
}
/*  函数功能:       求n个数中的最小值
    函数入口参数:   整型数组num，存放n个整数
                    整型变量n，表示数组元素个数
    函数出口参数:   整型指针变量pMinPos，指向存放最小值的整型变量
    函数返回值:     最小值
*/
int FindMin(int num[], int n, int *pMinPos)
{
    int i, min;
    min = num[0];                   /*假设num[0]为最小*/
    *pMinPos = 0;
    for (i = 1; i < n; i++)
    {
        if (num[i] < min)
        {
            min = num[i];
            *pMaxPos = i;
        }
    }
    return min;                     /*返回最小值*/
}
```

程序运行结果：

```
Input 10 numbers:
1 2 3 45 67 8 9 12 7 8↙
Max=67, Position=5, Min=1, Position=1
```

7.7 不用 strcat 函数，编程实现字符串连接函数 strcat 的功能，将字符串 t 连接到字符串s 的尾部。

【参考答案】 函数 MyStrcat 用下面两种方法实现。

方法 1 用字符数组编程实现函数 MyStrcat。

```
/*  函数功能:   连接两个字符串
    函数参数:   字符型数组dstStr存储连接后的字符串
                字符型数组srcStr存储待连接的字符串
    函数返回值: 无
*/
void  MyStrcat(char dstStr[], char srcStr[])  /*用字符数组作为函数参数*/
{
    int  i=0, j=0;                    /*数组下标初始化为0*/
    while (dstStr[i] != '\0')
    {
        i++;
    }
    while (srcStr[j] != '\0')
    {
        dstStr[i] = srcStr[j];
        i++;
        j++;
    }
    dstStr[i] = '\0';          /*在字符串dstStr的末尾添加一个字符串结束标志*/
}
```

方法 2 用字符指针编程实现函数 MyStrcat。

```
void  MyStrcat(char *dstStr, char * srcStr)   /*用字符指针作为函数参数*/
{
    while (*dstStr != '\0')
    {
        dstStr++;
    }
    while (*srcStr != '\0') /*若srcStr所指字符不是字符串结束标志*/
    {
        *dstStr = * srcStr; /*将srcStr所指字符复制到dstStr所指的存储单元中*/
        srcStr++;           /*使srcStr指向下一个字符*/
        dstStr++;           /*使dstStr指向下一个存储单元*/
    }
    *dstStr = '\0';         /*在字符串dstStr的末尾添加一个字符串结束标志*/
}
```

主函数程序如下：

```
#include <stdio.h>
main()
{
    char  s[80];          /*源字符串*/
    char  t[80];          /*待连接字符串*/
    printf("Please enter the source string: \n");
    gets(s);
    printh("Please enter the other string: ");
    gets(t);              /*输入字符串*/
    MyStrcat(s, t);       /*将字符数组t中的字符串连到s的尾部*/
    printf("The concat is:\n");
    puts(s);              /*输出连接后的字符串s*/
}
```

7.8 编程实现从键盘输入一个字符串，将其字符顺序颠倒后重新存放，并输出这个字符串。

【算法思想】 定义两个指针分别指向字符串的两端，同时向前和向后边移动边交换。

【参考答案】

```
#include <stdio.h>
#include <string.h>
main()
{
    char *pStr, temp, str[80];
    char *pStart, *pEnd;
    int  len;
    pStr = str;
    printf("Input string:\n");
    gets(pStr);                          /*输入字符串*/
    len = strlen(pStr);                  /*求出字符串长度*/
    for (pStart = pStr, pEnd=pStr + len-1; pStart < pEnd ; pStart++,pEnd--)
    {   /*pStart,pEnd分别指向串的首和尾*/
        temp = *pStart;
        *pStart = *pEnd;
        *pEnd = temp;                /*交换pStart和pEnd指向的串中的字符*/
    }
    puts(pStr);                          /*输出交换后的结果*/
}
```

程序运行结果：

```
Input string:
abcdef↙
fedcba
```

*7.9 编程判断输入的一串字符是否为“回文”。所谓“回文”是指顺读和倒读都一样的字符串。如"level"，"ABCCBA"都是回文。

【参考答案】

```
#include <stdio.h>
#include <string.h>
main()
{
    char str[80], *pStart, *pEnd;
    int len;
    printf("Input string: ");
    gets(str);
    len = strlen(str);
    pStart = str;
    pEnd = str + len - 1;
    while ((*pStart == *pEnd) && (pStart < pEnd))
    {
        pStart++;
        pEnd--;
    }
    if (pStart < pEnd)
    {
        printf("No!\n");
    }
    else
    {
        printf("Yes!\n");
    }
}
```

程序的 2 次测试结果如下：

① Input string:abccba↙

Yes!

② input string:abccbd↙

No!

*7.10　编写一个能对任意 $m \times n$ 阶矩阵进行转置运算的函数 Transpose。

【参考答案】

```
#define ROW_LEN 3
#define COL_LEN 4
void Transpose(int *a, int *at, int row, int col);    /*函数声明*/
main()
{
    int s[ROW_LEN][COL_LEN];          /*s代表原矩阵*/
    int st[COL_LEN][ROW_LEN];         /*st代表转置后的矩阵*/
    int i, j;
    printf("Please enter the source matrix:\n");
    for (i = 0; i < ROW_LEN; i++)
    {
        for (j = 0; j < COL_LEN; j++)
        {
            scanf("%d ", &s[i][j];    /*输入矩阵*/
        }
    }
    Transpose(*s,*st, ROW_LEN, COL_LEN);   /*调用函数*/
    Printf("The transposed matrix is:\n");
    for (i = 0; i < COL_LEN; i++)
    {
        for (j = 0; j < ROW_LEN; j++)
        {
            printf("%d", st[i][j]);      /*输出转置后的矩阵*/
        }
        printf(" \n");
    }
}
/*  函数功能：对任意row行col列的矩阵转置
    函数入口参数：整型指针变量a，指向一个二维整型数组的第0行第0列
                  整型变量row，二维整型数组的行数
                  整型变量col，二维整型数组的列数
    函数出口参数：整型指针变量at，指向转置后二维数组的第0行第0列
    函数返回值：无
*/
void Transpose(int *a, int *at, int row, int col)
{
    int i, j;
    for (i = 0; i < row; i++)
    {
        for (j = 0; j < col; j++)
        {
            *(at + j * row + i) = *(a + i * col + j); /*对应元素交换*/
        }
    }
}
```

程序运行结果：

```
Please enter the source matrix:
1 2 3 4↙
1 2 3 4↙
1 2 3 4↙
The transposed matrix is:
1 1 1
2 2 2
3 3 3
4 4 4
```

*7.11　用指针数组编程实现：从键盘任意输入一个数字表示月份值 n，程序输出该月份的英文表示，若 n 不在 1~12 之间，则输出“Illegal month”。

【参考答案】

```
#include <stdio.h>
main()
{
    int n;
    static char *monthName[]={"Illegal month", "January", "February",
                        "March", "April", "May", "June", "July", "August",
                        "September", "October", "November", "December"};
    printf("Input month number:");
    scanf("%d", &n);    /*输入月份*/
    if ((n <= 12) && (n >= 1))
    {
        printf("month %d is %s\n", n, monthName[n]);  /*输出相应月份*/
    }
    else
    {
        printf("%s\n", monthName[0]);            /*输出错误*/
    }
}
```

程序的 3 次测试结果如下：

① Input month number:5↙

  month 5 is May

② Input month number:12↙

  month 12 is December

③ Input month number:13↙

  Illegal month

1.8 习题8及参考答案

8.1 选择题

【参考答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
答案	A	C	D	C	D	B	D	D

8.2 填空题

【参考答案】 (1) ① 0x34 ② 0x12

【参考答案】 (2) ① 2 ② 3

【参考答案】 (3) ① struct node *next;

8.3 阅读程序，选择程序运行结果。

【参考答案】 (1) D

【参考答案】 (2) B

【参考答案】 (3) 4

8.4 判断题

【参考答案】 (1) ① 错 ② 错 ③ 对 ④ 错 ⑤ 错 ⑥ 对

【参考答案】 (2)

① 错，正确的方式为

```
struct  STUDENT
{
    char name[10];
    int age;
} ;
```

② 错，正确的方式为

```
struct STUDENT student;
```

③ 错，正确的方式为

```
student.age=20;
```

④ 错，正确的方式为

```
union val
{
    char w;
    float x;
    int m;
}v={'0'};
```

8.5　在计算机中有一个重要的概念——堆栈。堆栈是指这样一段内存，它可以理解为一个简结构，先放进筒中的数据被后放进筒中的数据“压住”，只有后放进筒中的数据都取出后，先放进去的数据才能被取出，这称为“后进先出”。堆栈的长度可以随意增加。堆栈结构可以用链表实现。设计一个链表结构需包含两个成员：一个存放数据，一个为指向下一个节点的指针。当每次有一个新数据要放入堆栈时，称为“压栈”，这时动态建立一个链表的节点，并连接到链表的结尾；当每次从堆栈中取出一个数据时，称为“弹出堆栈”，这意味着从链表的最后一个节点中取出该节点的数据成员，同时删除该节点，释放该节点所占的内存。堆栈不允许在链表中间添加、删除节点，只能在链表的结尾添加和删除节点。试用链表方法实现堆栈结构。

【算法思想】　从对堆栈的描述来看，堆栈是一个特殊的链表，其特殊之处在于，不会在任意两个节点之间插入一个节点，每一个新的节点必须插入到链表的尾部；也不会从链表中间删除任意一个节点，每一次的删除必须从链表的最后一个节点删除。即每一次“压栈”意味着在链表的尾部增加一个节点；而每一次“弹出堆栈”意味着读出链表尾部节点的数据，并且将该节点删除。所以需要编写两个函数：

（1）压栈　*PushStack(int num);

（2）弹出堆栈　int PopStack(void);

“压栈”意味着两个操作过程：① 产生一个新的节点；② 将要保存的数据放到新的节点的数据区。前一个过程请参照教材第 8 章的讲解，而后一个过程实际上就是一个赋值过程，链表结构中的数据区的数据类型与 PushStack 函数的参数的数据类型应该一致。

“弹出堆栈”意味着 3 个操作过程：① 找到链表的最后一个节点；② 将节点数据区的数据内容读出，放在一个临时变量中；③ 删除该节点，返回临时变量中的值。

【参考答案】

```
#include <stdio.h>
#include <stdlib.h>
struct stack
{
    int data;
    struct stack *next;
};
typedef struct stack STACK;
STACK *head,*pr;
int nodeNumber = 0;      /* 堆栈节点数寄存器 */
/*函数功能：生成一个新的节点，并为该节点赋初值
  函数参数：整型变量unm，是要给新节点赋的初值
  函数返回值：指向新的节点的指针
  */
  STACK *CreateNode(int num)
  {
      STACK *p;
      p = (STACK *)malloc(sizeof(STACK));  /* 动态申请一段内存 */
      if(p == NULL) /* 如果返回空指针，申请失败，打印错误信息并退出程序 */
      {
          printf("No enough memory to alloc");
          exit(0);                          /*结束程序运行*/
      }
      p->next = NULL;                       /* 为新建的节点指针域赋空指针 */
      p->data = num;                        /* 为新建的节点数据区赋值 */
      return p;
  }
  /*  函数功能：压栈函数
      函数参数：整型变量num，表示要保存到栈里的数据
      函数返回值：指向链表新节点的指针
  */
  STACK *PushStack(int num)
  {
      if(nodeNumber==0)
      {
          /* 如果是第一个节点，保留该节点首地址在head中*/
          head = CreateNode(num);
          pr = head;
          nodeNumber++;                 /* 堆栈节点数寄存器+1 */
      }
      else
      {
          /* 不是第一个节点，将新建节点连到链表的结尾处 */
          pr->next = CreateNode(num);
          pr = pr->next;
          nodeNumber++;                 /* 堆栈节点数寄存器+1 */
      }
  }
```

```
/*  函数功能：弹出堆栈
    函数参数：无
    函数返回值：当前堆栈中存储的数据
*/
int PopStack(void)
{
    STACK *p;
    int result;
    p = head;
    for(;;)
    {
        if(p->next == NULL)              /* 查找最后一个节点 */
        {
            break;
        }
        else
        {
            pr = p;                      /* 记录最后一个节点的前一个节点的地址 */
            p = p->next;
            nodeNumber--;                /*堆栈节点数寄存器-1 */
        }
    }
    if(nodeNumber == 0) return -1; /* 如果是最后一个节点,返回错误代码*/
    pr->next = NULL;    /*将最后一个节点的前一个节点的地址的指针域赋空指针*/
    result = p->data;
    free(p);                             /*释放内存*/
    return result;
}
main ()
{
    int pushNum[10]={111,222,333,444,555,666,777,888,999,10000};
    int popNum[10];
    int i;
    for(i=0;i<10;i++)
    {
        PushStack(pushNum[i]);
        printf("Push %dth Data :  %d\n",i,pushNum[i]);
    }
    for(i=0;i<10;i++)
    {
        popNum[i] = PopStack();
        printf("Pop %dth Data :  %d\n",9-i,popNum[i]);
    }
}
```

8.6 请读者自己编写一个程序，该程序可以完成个人财务管理。每个人的财务项目应当包括姓名、年度、收入、支出等。为了叙述简单，以一个财政年度为统计单位，程序中可以计算每个人的每个财政年度的收入总额、支出总额、存款余额等，并能够打印出来。需要注意的是，收入总额不可能只输入一次，而可能是多次收入的总和；同样地，支出总额也不可能是一次支出，应是多次支出的总和。

【算法思想】 本程序是一个财务管理程序，涉及收入和支出，虽然是个人财务管理程序，但最好能够按照一种标准的财务管理软件来考虑，所以在程序设计时，需要考虑如下几个因素：

（1）每一笔收入或支出都可以理解为一笔交易，那么程序最多可以容纳多少笔交易决定着数组的元素个数，必须有一个预测，不妨先假定为 50 笔。

（2）确定结构体形式时需要认真考虑，它关系到程序实现的思路和方式。一个人的收入和支出显然是多次输入的，而每一笔交易必须记录交易的日期和交易的具体时间，所以结构体应该包含日期和时间信息。从银行存款单我们可以知道，每一次存款和取款的数目必须记录，而且是分别记录，所以结构体考虑增加收入和支出两个元素，这样就形成了如下描述每一笔交易的结构体。

```
struct deal
{
    struct date dt;                          /*每笔交易的日期*/
    struct time ti;                          /*每笔交易的时间*/
    double earning;                          /*每笔交易的收入额度*/
    double payout;                           /*每笔交易的支出额度*/
};
```

（3）这样的设计能够满足题目要求吗？首先，每一笔交易的记录可以存储起来，而且记录了交易的时间和金额，但是计算机如何知道是收入还是支出？此处必须设计一种输入方式，需要用户在输入每一笔交易金额的前面添加“+”或“-”，“+”代表收入，“-”代表支出，程序根据金额前面的符号判断存储在结构体的哪个元素中。其次，题目要求可以输出年度的收入总额、支出总额、余额等信息，从结构体的设计可以看到，通过计算可以获得这些信息。收入总额为当年每一笔交易的收入元素之和；支出总额为当年每一笔交易的支出元素之和；余额为当年收入总额减去当年支出总额，这些数据可以在需要时随时通过程序计算得到。

（4）当进行输入操作时，如果由于操作人员的失误，出现输入错误，程序如何处理？输入错误会有三类：其一，数字错敲成字符，可以通过程序禁止接收字符来滤掉此类错误；其二，数字或小数点键入错误，如本来是 123.23，却敲成了 12.233，程序该如何处理？其三，本来是收入应该敲“+”号，却敲成了“-”号，变成了支出，程序该如何处理？从财务角度讲，每一笔交易都要记录，即使错了也要记录，而且不能修改，如果要将错误修正过来，惟一的办法是再进行一笔交易，将错误修正过来，使收入和支出平衡。所以从这个角度讲，第 1 种错误要限制，第 2 和第 3 种错误程序不需要处理，实际上也没有办法处理。

【参考答案】

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <dos.h>
#define DEAL    50                       /*设计最大交易次数*/
struct deal
{
    struct date dt;                      /*每笔交易的日期*/
    struct time ti;                      /*每笔交易的时间*/
    double earning;                      /*每笔交易的收入额度*/
    double payout;                       /*每笔交易的支出额度*/
};
typedef struct deal FINANCE;
FINANCE person[DEAL]={0};

/*  函数功能：求账面上的余额
    函数参数：结构体指针变量per，代表指向结构体数组的指针
    函数返回值：账面余额
*/
float Balance(FINANCE *per)
{
    int i;
    float sum1 = 0.0,sum2 = 0.0;

    for(i=0;i<DEAL;i++)
    {
        sum1 += (per+i)->earning;
        sum2 += (per+i)->payout;
    }
    return sum1+sum2;
}
/*  函数功能：年度统计，输出年度收入总额、支出总额、余额
    函数参数：结构体指针变量per，代表指向结构体数组的指针
              整型变量year，代表年度，如2004
    函数返回值：无
 */
void OneYearBalance(FINANCE *per,int year)
{
    int i;
    float sum1 = 0.0,sum2 = 0.0;
    for(i=0;i<DEAL;i++)
    {
        if((per+i)->dt.da_year != year)continue;

        sum1 += (per+i)->earning;
        sum2 += (per+i)->payout;
    }
    printf("param of %d  :  ",year);
    printf("Sum of earning       Sum of payout       Balance\n");
    printf("%28.2f%21.2f%19.2f\n",sum1,sum2,sum1+sum2);
}
```

```
/*  函数功能：打印所有的交易信息
    函数参数：结构体指针变量per，代表指向结构体数组的指针
    函数返回值：无
*/
void PrintBalance(FINANCE *per)
{
    int i;
    printf("  Date      Time      Earning     Payout    Balance \n");
    for(i=0;i<DEAL;i++)
    {
        if((per+i)->dt.da_year != 0)
        {
            printf("%d/%d/%d  ",(per+i)->dt.da_year,
                    (per+i)->dt.da_mon,(per+i)->dt.da_day);
            printf("%d:%d:%d :",(per+i)->ti.ti_hour,
                    (per+i)->ti.ti_min,(per+i)->ti.ti_sec);
        }
        if((per+i)->earning!=0)
            printf("%8.2f\n",(per+i)->earning);
        if((per+i)->payout!=0)
            printf("           %8.2f\n",(per+i)->payout);
    }
    printf("                                          %8.2f\n",
            Balance(per));
}
/*  函数功能：打印某一年的交易信息
    函数参数：结构体指针变量per，代表指向结构体数组的指针
    函数返回值：无
*/
void PrintOneYear(FINANCE *per)
{
    int i,year;
    printf("Please Input one year :");
    scanf("%d",&year);
    printf("  Date      Time      Earning     Payout    Balance \n");
    for(i=0;i<DEAL;i++)
    {
        if((per+i)->dt.da_year != year) continue;

        if((per+i)->dt.da_year!=0)
        {
            printf("%d/%d/%d  ",(per+i)->dt.da_year,
                    (per+i)->dt.da_mon,(per+i)->dt.da_day);
            printf("%d:%d:%d :",(per+i)->ti.ti_hour,
                    (per+i)->ti.ti_min,(per+i)->ti.ti_sec);
        }
        if((per+i)->earning!=0)
            printf("%8.2f\n",(per+i)->earning);
        if((per+i)->payout!=0)
            printf("           %8.2f\n",(per+i)->payout);

    }
    OneYearBalance(per,year);
}
```

```
/*  函数功能：获得计算机的系统日期、时间
    函数参数：结构体指针变量per，代表指向结构体数组的指针
    函数返回值：无
*/
void GetDateTime(FINANCE *per)
{
    printf("Please input one deal:\n");
    getdate(&per->dt);
    printf("%d/%d/%d :",per->dt.da_year,per->dt.da_mon,
            per->dt.da_day);
    gettime(&per->ti);
    printf("%d:%d:%d :",per->ti.ti_hour,per->ti.ti_min,
            per->ti.ti_sec);
}
/*  函数功能：每一笔交易输入模块
    函数参数：结构体指针变量per，代表指向结构体数组的指针
    函数返回值：无
*/
void InputOneDeal(FINANCE *per)
{
    char string[10];
    printf("Please input deal (+/-)\n");
    GetDateTime(per);
    scanf("%s",&string);
    if(string[0] == '-')
        per->payout = atof(string);
    else
        per->earning = atof(string);
}
/*  函数功能：菜单模块
    函数参数：无
    函数返回值：无
*/
void Menu()
{
    printf("1.Input One deal\n");
    printf("2.Print All the Balance\n");
    printf("3.Print One year Balance\n");
    printf("4.Exit to DOS\n");
}
main()
{
    char key;
    int i;
    i = 0;
    while(1)
    {
        Menu();
        key=bioskey(0);
        switch(key)
        {
            case '1':  InputOneDeal(person+i);
                       i++;
                       break;
            case '2':  PrintBalance(person);
                      break;
            case '3':  PrintOneYear(person);
                      break;
            case '4':  exit(0);
            default :  break;
        }
    }
}
```

8.7　口袋中有若干红、黄、蓝、白、黑 5 种颜色的球，每次从口袋中取出 3 个球，编程打印出得到 3 种不同颜色的球的所有可能取法。

【算法思想】　利用三重循环分别模拟取球过程，但每次取出的球需要与前面的球比较颜色，颜色相同的球要抛弃。

【参考答案】

```
#include <stdio.h>
main ()
{
    char *ballColor[]={"RED","YELLOW","BLUE","WHITE","BLACK"};
    int i,j,k,m=0;
    for(i=0;i<5;i++)
    {
        for(j=i+1;j<5;j++)
        {
            for(k=j+1;k<5;k++)
            {
                m++;
                printf("%d: %s , %s , %s\n",m,
                        ballColor[i],ballColor[j],ballColor[k]);
            }
        }
    }
}
```

8.9　程序语句填空。

【参考答案】

① break;

② pre->next = newp;

③ newp->next = suc;

8.8　25 个人围成一个圈，从第 1 个人开始顺序报号，凡报号为 3 和 3 的倍数者退出圈子，找出最后留在圈子中的人原来的序号。

【算法思想】　可以用链表方式实现。首先按照 1～25 号的顺序建立链表，1～25 号存在链表的数据区内，从链表头开始数，数到 3 或 3 的倍数的节点，删除该节点，一直到链表结束；重复上述过程一直到链表中只剩下两个节点，读出最后两个节点的数据区的数据就是最后的答案。

编程中会遇到如下几个问题：

（1）在第一次从链表头开始数每个节点时，可以很方便地确定每个节点的顺序，当到该链表的结尾时，结尾节点的顺序号应该是 25 号，下面应该数哪个节点？显然是头节点，也就是说，需要将该链表的首尾相接形成一个环形，如何形成一个环形？程序怎么做？

（2）删除节点操作中包括三种不同的节点删除方式：中间节点、头节点、结尾节点。每一种删除方式都会不同，在访问和删除过程中必须分清要删除节点的类型。

（3）必须考虑循环何时结束。

【参考答案】

```
#include <stdio.h>
#include <stdlib.h>
struct Link
{
    int data;
    struct Link *next;
};
struct Link *head;              /*建立一个指向链表头的全局变量*/
/*  函数功能：建立一个新的节点，并为该节点赋初值
    函数参数：整型变量nodeNumbers，表示建立的节点个数
    函数返回值：指向该节点的指针
*/
struct Link *CreateNode(int nodeNumbers)
{
    struct Link *p;
    p = (struct Link *)malloc(sizeof(struct Link));  /* 动态申请一段内存 */
    if(p == NULL)           /* 如果返回空指针，申请失败，打印错误信息并退出程序 */
    {
        printf("No enough memory to alloc");
        exit(0);                            /*结束程序运行*/
    }
    p->next = NULL;                         /* 为新建的节点指针域赋空指针 */
    p->data = nodeNumbers+1;                /* 为新建的节点数据区赋值 */
    printf("\nCreate a new node!");
    return p;
}
/*  函数功能：显示所有已经建立的节点的节点号和该节点中数据项内容
    函数参数：结构体指针变量head，表示指向链表的头指针
    函数返回值：无
*/
void DispLink(struct Link *head)
{
    struct Link *p;
    int j=1;
    p = head;
    do{
        printf("\n%5d%10d\n",j,p->data);
        p = p->next;
        j++;
    }while(p != NULL);
}
/*  函数功能：删除节点，释放内存
    函数参数：结构体指针变量p，表示指向链表的当前节点的指针
              结构体指针变量pr，表示指向链表的当前节点的前一个节点的指针
    函数返回值：返回指向当前节点的指针
*/
struct Link *DelNode(struct Link *p, struct Link *pr)
{
    if(p == head){
        head = p->next;             /*头节点的删除*/
        free(p);
        return head;
    }
    if(p->next == NULL){                /*尾节点的删除*/
        pr->next = NULL;
        free(p);
        return head;                /*指针指向头节点*/
    }
    else
    {
        pr->next = p->next;         /*中间节点的删除*/
        free(p);
        return pr->next;
    }
}
```

```
main()
{
    int i=0,nodenum=25;
    struct Link *p,*pr;
    char c;
    head = NULL;
    for(i=0;i<25;i++)
    {
        if (i == 0)
        {
            /* 如果是第一个节点，在head中保留该节点的首地址*/
            head = CreateNode(i);
            pr = head;
        }
        else
        {
            /* 如果不是第一个节点，将新建节点连到链表的结尾处 */
            pr->next = CreateNode(i);
            pr = pr->next;
        }
    }
    DispLink(head);
    i=1;
    p = head;
    for(;;)
    {
        if((i%3)==0)                    /*如果是3的倍数*/
        {
            p = DelNode(p,pr);
            i++;
            nodenum--;                  /*节点数-1*/
            if(nodenum<3) break;        /*如果节点数<3，退出循环*/
        }
        else
        {
            pr = p;                     /*链表走到下一个节点*/
            p = p->next;
            if(p == NULL)               /*如果到尾节点，下一个节点连接到头节点*/
               p = head;
            i++;                        /*计数器+1*/
        }
    }
       DispLink(head);
}
```

1.9 习题9及参考答案

9.1 选择题

【参考答案】 (1) A

【参考答案】 (2) ① B ② B

9.2 阅读程序，给出程序结果。

【参考答案】 1 1 0 0

9.3 程序填空。

【参考答案】 (1) ① return 1 ② return n*Facto(n-1)

【参考答案】 (2) ① Y(x,n-1)

【参考答案】 (3) ① return 1 ② n+Sum(n-1)

9.4 用递归方法实现快速排序。对一个一维数组进行快速排序的基本算法如下：

(1) 分割 取未排序数组中的第一个元素，确定它在最终排序的数组中的位置。这时数组中的所有小于该元素的元素都位于它的左边，所有大于该元素的元素都位于它的右边。至此，有一个元素已经处于它该处的位置，另外还有两个未排序的子数组。

(2) 递归 对每一个未排序的数组执行步骤(1)。

【算法思想】 首先确定数组中第一个元素的最后位置。看下面一组数据（带圆括号的元素为分割元素，将被放到它在最终排序的数组中应处的位置）：

(37) 2 6 4 89 8 10 12 68 39

(1) 从数组的右端元素开始，依次与37进行比较，直到发现小于37的元素，将此元素与37交换位置。第一个小于37的元素是12，将37与12交换位置，则数组变为

[12] 2 6 4 89 8 10 (37) 68 39

(2) 从数组的左边元素开始（不包括12，即从12后边的元素开始），依次与37比较，直到发现有大于37的元素，上面的序列为89，将37与89交换位置，数组变为

12 2 6 4 (37) 8 10 [89] 68 39

(3) 从数组的右端元素开始（不包括89，即从89前边的元素开始），依次与37比较，直到发现有小于37的元素，上面的序列为10，将37与10交换位置，数组变为

12 2 6 4 [10] 8 (37) 89 68 39

(4) 从数组的左边元素开始（即从10后边的元素开始），依次与37比较，直到发现有大于37的元素，因为再也没有大于37的元素了，因此37与其自身比较时，已经处于最终排序的数组中的位置。

12 2 6 4 10 8 (37) 89 68 39

(5) 上面数组的分割结果产生了两个未排序的子数组。小于37的子数组为12，2，6，4，10，8；大于37的子数组包含89，68，35。对两个子数组重复上述分割步骤，对数组继续排序。

根据上面的分析，设计一个函数QuickSort，入口参数包括一个指向整数数组的指针、数组的起始下标、数组的终止下标。如：

```
int *QuickSort(int *num, int StartPos, int EndPos);
```

【参考答案】

```
#include <stdio.h>
/*  函数功能：交换两个地址中的数据
    函数参数：整型指针变量first，表示指向第一个数据的地址
              整型指针变量second，表示指向第二个数据的地址
    函数返回值：无
*/
void Exchange(int *first,int *second)
{
    int *temp=0;
    *temp = *first;
    *first = *second;
    *second = *temp;
}
```

```
/*  函数功能：实现排序的递归函数
    函数参数：整型指针变量num,表示指向数组的指针
              整型变量startPos,表示子数组第一个元素在原数组中的位置
              整型变量endPos,表示子数组最后一个元素在原数组中的位置
    函数返回值：无
*/
int *QuickSort(int *num,int startPos,int endPos)
{
    int temp,i;
    int nowPos=startPos;
    int start=startPos,end=endPos;
    temp = *(num+start);
    if(start == end)
        return;
    else
    {
        while(1)
        {
            for(i=end;i>nowPos;i--) /*从右向左开始寻找*/
                if(temp > *(num+i)) /*如果找到比temp小的数据，循环终止*/
                    break;
            Exchange(num+nowPos,num+i); /*交换两个地址中的数据*/
            start = nowPos+1;           /*确定从左到右开始寻找的起始地址*/
            nowPos = i;                 /*保存当前地址*/
            if(start>nowPos)break;
            for(i=start;i<nowPos;i++)   /*从左向右开始寻找*/
                if(temp < *(num+i)) /*如果找到比temp大的数据，循环终止*/
                    break;
            Exchange(num+nowPos,num+i); /*交换两个地址中的数据*/
            end = nowPos-1;             /*确定从右到左开始寻找的起始地址*/
            nowPos = i;                 /*保存当前地址*/
            if(start==end || nowPos>end || nowPos<start)break;
        }
        QuickSort(num,startPos,nowPos-1);
        if(nowPos+1>endPos)
            nowPos = endPos-1;
        QuickSort(num,nowPos+1,endPos);
    }
}
```

```
/*  函数功能：实现排序的递归函数
    函数参数：整型指针变量num,表示指向数组的指针
              整型变量startPos,表示子数组第一个元素在原数组中的位置
              整型变量endPos,表示子数组最后一个元素在原数组中的位置
    函数返回值：无
*/
int *QuickSort(int *num,int startPos,int endPos)
{
    int temp,i;
    int nowPos=startPos;
    int start=startPos,end=endPos;
    temp = *(num+start);
    if(start == end)
        return;
    else
    {
        while(1)
        {
            for(i=end;i>nowPos;i--) /*从右向左开始寻找*/
                if(temp > *(num+i)) /*如果找到比temp小的数据，循环终止*/
                    break;
            Exchange(num+nowPos,num+i); /*交换两个地址中的数据*/
            start = nowPos+1;           /*确定从左到右开始寻找的起始地址*/
            nowPos = i;                 /*保存当前地址*/
            if(start>nowPos)break;
            for(i=start;i<nowPos;i++)   /*从左向右开始寻找*/
                if(temp < *(num+i)) /*如果找到比temp大的数据，循环终止*/
                    break;
            Exchange(num+nowPos,num+i); /*交换两个地址中的数据*/
            end = nowPos-1;             /*确定从右到左开始寻找的起始地址*/
            nowPos = i;                 /*保存当前地址*/
            if(start==end || nowPos>end || nowPos<start)break;
        }
        QuickSort(num,startPos,nowPos-1);
        if(nowPos+1>endPos)
            nowPos = endPos-1;
        QuickSort(num,nowPos+1,endPos);
    }
}
main()
{
    int number[10] = {37,2,6,4,9,28,10,12,68,39};
    int i;
    printf("\nBefore Sort:  ");
    for(i=0;i<10;i++)
        printf("%5d",number[i]);
    printf("\n");
    QuickSort(number,0,9);
    printf("\nAfter Sort:   ");
    for(i=0;i<10;i++)
        printf("%5d",number[i]);
    printf("\n");
}
```

1.10 习题 10 及参考答案

10.1 用基本文件操作编写 DOS 下的 type 命令，即把文件内容以 ASCII 码字符方式向标准输出设备输出。

【算法思想】 type 命令在没有参数时提示语法错误，有参数时就把所有的参数当作文件名输出。

【参考答案】

```
#include <io.h>
#include <stdio.h>
#include <fcntl.h>
int type(const char* filename);
int main(int argc, char *argv[])
{
    int i;
    if (argc < 2)
    {
        printf("The syntax of the command is incorrect.\n");
        return -1;
    }
    /* 把所有参数逐一输出 */
    for (i=1; i<argc; i++)
    {
        if ( type(argv[i]) == 0 )
            perror(argv[i]);
    }
    return 0;
}
/*
   函数功能：把文件filename输出到stdout
   函数返回值：非0表示成功，否则出错
*/
int type(const char* filename)
{
#define BUF_SIZE    1024
    int fh, rtn, val=1;
    char buf[BUF_SIZE];
    fh = open(filename, O_RDONLY | O_TEXT);
    if (fh == -1)
        return 0;
    while((rtn = read(fh, buf, BUF_SIZE-1)) > 0)
    /* BUF_SIZE-1为'\0'留个空间 */
    {
        buf[rtn] = '\0';    /* 字符串终结符 */
        printf(buf);
    }
    if (rtn == -1)
        val = 0;
    close(fh);
#undef BUF_SIZE
    return val;
}
```

10.2 用基本文件操作编写与例 10.4 有同样文件复制功能的程序。

【算法思想】 用基本文件操作函数一个字符一个字符地读写，效率太低了，应该整块整块地复制。只要修改 CopyFile 函数即可，这也体现了函数的优越性。

【参考答案】

```
/*
   函数功能：把srcName文件内容复制到dstName
   函数入口参数：文件路径
   函数返回值：非0值表示复制成功，否则出错
*/
int CopyFile(const char* srcName, const char* dstName)
{
#define BUF_SIZE    1024
    char buf[BUF_SIZE];
    int fhSrc = -1;
    int fhDst = -1;
    int rval=1;
    int rtn;
    /* 打开文件 */
    fhSrc = open(srcName, O_RDONLY | O_BINARY);
    if (fhSrc == -1)
        goto ERROR;
    fhDst = open(dstName, O_WRONLY | O_CREAT | O_TRUNC | O_BINARY);
    if (fhDst == -1)
        goto ERROR;
    /* 复制文件 */
    while((rtn = read(fhSrc, buf, BUF_SIZE)) > 0)
    {
        if (write(fhDst, buf, rtn) == -1)
            goto ERROR;
    }
    if (rtn == 0)
        goto EXIT;
ERROR:
    rval=0;
EXIT:
    if (fhSrc != -1)
        close(fhSrc);
    if (fhDst != -1)
        close(fhDst);
#undef BUF_SIZE
    return rval;
}
```

10.3　已知文件的前若干字符与文件类型的对应关系为

前若干字符	文 件 类 型
MZ	EXE
Rar!	RAR
PK	ZIP
%PDF	PDF
BM	BMP
GIF	GIF
RIFF	AVI 或 WAV 等
MThd	MID

有些软件通过改变文件的扩展名隐藏文件的真实类型。比如，有些游戏的音乐和动画其实就是标准的 mid 和 avi 文件，只要把扩展名改回来，就能直接播放。现在编写一个程序，使它能从一个配置文件中获得字符串与文件类型的对应表，然后判断用户指定的文件的真实类型。

【参考答案】

```
#include <io.h>
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
/*  配置文件名。格式为
 *
 *  3
 *  MZ
 *  EXE
 *  Rar!
 *  RAR
 *  RIFF
 *  AVI or WAV etc.
 *
 *  第一行的数字表示类型数。其他行的偶数行是特征，奇数行是类型名
 */
#define CONFIG_FILENAME     "truetype.ini"
#define CHARARCTER_LEN      10
#define NAME_LEN            22
typedef struct
{
    char character[CHARARCTER_LEN];
    char name[NAME_LEN];
} FILETYPE;
int typeCount;                /* 类型总数 */
FILETYPE* typeTable=NULL;     /* 类型表 */
int MakeTypeTable(void);
void FreeTypeTable(void);
int TrueType(const char* filename);
const char* TypeName(int type);
void TrimNewLineChar(char* str);
int main(int argc, char *argv[])
{
    int i;
    if (argc < 2)
    {
        printf("You must specify one filename at least.\n");
        return 1;
    }
    if ( !MakeTypeTable() )
    {
        printf("Error on config file.\n");
        return 2;
    }
    /* 逐一判断所有参数*/
    for (i=1; i<argc; i++)
    {
        int type = TrueType(argv[i]);
        if ( type == -1 )
            perror(argv[i]);
        else if ( type == -2)
            printf("Unknown type.\n");
        else
            printf("File %s's true type is: %s\n",
                    argv[i],
                    TypeName(type));
    }
    FreeTypeTable();
    return 0;
}
```

```
/*
   函数功能：从CONFIG_FILENAME读入文件特征类型对应表
   函数返回值：非0表示成功，否则失败
*/
int MakeTypeTable(void)
{
     FILE* fp=NULL;
     int rval=1, i;
     fp = fopen(CONFIG_FILENAME, "r");
     if (fp == NULL)
          goto ERROR;
     /* 读类型数及其结尾的'\n' */
     if (fscanf(fp, "%d\n", &typeCount) != 1)
          goto ERROR;
     /* 申请表空间 */
     FreeTypeTable();          /*如果已经建立过类型表，要先删除 */
     typeTable = (FILETYPE*)malloc( typeCount * sizeof(FILETYPE) );
     if (typeTable == NULL)
          goto ERROR;
     memset(typeTable, 0, typeCount * sizeof(FILETYPE));
     /* 读表 */
     for (i=0; i<typeCount; i++)
     {
          char* pRtn;
          /* 读特征 */
          pRtn = fgets(typeTable[i].character, CHARARCTER_LEN, fp);
          if (pRtn == NULL)
               goto ERROR;
          TrimNewLineChar(typeTable[i].character);
          /* 读类型名 */
          pRtn = fgets(typeTable[i].name, NAME_LEN, fp);
          if (pRtn == NULL)
               goto ERROR;
          TrimNewLineChar(typeTable[i].name);
     }
     /* 成功 */
     rval = 1;
     goto EXIT;
     /* 出口处理 */
ERROR:
     rval = 0;
     FreeTypeTable();
EXIT:
     if (fp != NULL)
          fclose(fp);
     return rval;
}
/*
   函数功能：判断filename的类型
   函数返回值：类型ID，从0开始计数
               -1表示文件操作出错，-2表示无法得到真实类型
*/
int TrueType(const char* filename)
{
     int fh=-1, rval;
     int rtn, i;
     char buf[NAME_LEN];
     fh = open(filename, O_RDONLY | O_BINARY);
     if (fh == -1)
          goto ERROR;
     rtn = read(fh, buf, NAME_LEN);
     if (rtn == -1)
          goto ERROR;
     /* 开始比较 */
     rval = -2;
     for (i=0; i<typeCount; i++)
     {
          /* 比较typeTable[i].character和buf中等长的字符串 */
          int cmp = strncmp(typeTable[i].character,
                            buf,
                            strlen(typeTable[i].character));
          if (cmp == 0)
          {
               rval = i;
               break;
          }
     }
     goto EXIT;
ERROR:
     rval = -1;
EXIT:
     if (fh != -1)
          close(fh);
     return rval;
}
```

```
/*
   函数功能：返回类型ID对应的类型名
   函数参数：整型的类型ID
   函数返回值：类型名
*/
const char* TypeName(int type)
{
    if (type < 0 || type > typeCount-1 )
        return "Unknown";
    else
        return typeTable[type].name;
}
/*
   函数功能：释放类型表空间
   函数参数：无
   函数返回值：无
*/
void FreeTypeTable(void)
{
    if (typeTable != NULL)
        free(typeTable);
    typeTable = NULL;
}
/*
   函数功能：去掉字符串最后的'\n'
   函数参数：被处理的字符串
   函数返回值：无
*/
void TrimNewLineChar(char* str)
{
    char* pRtn = strrchr(str, '\n');
    if (pRtn != NULL)
        *pRtn = '\0';
}
```

此程序利用高级文件操作函数按行读方式处理配置文件，利用基本文件操作按字节读方式读取被判断的文件。可以说这两种文件处理方法，各得其所。

只要对此程序稍加修改，就可将其包装成一个判断文件真实类型的模块。可以用任意文本编辑器编辑 truetype.ini 文件来扩充数据，支持更多类型的判断，而程序不需要修改一个字节。

10.4　习题 8.6 是一个很实用的小程序。如果能够把用户输入的数据存盘，下次运行时读出，就更有用了，编程尝试增加此项功能。

【算法思想】　每次启动程序时从数据文件读出数据，退出程序时保存所有数据（包括本次运行输入的数据）。

【参考答案】　为习题 8.6 的答案增加两个函数，并修改 main()如下：

```
/*
   函数功能：读取数据文件
   函数参数：无
   函数返回值：读入的记录条数
*/
int LoadData(void)
{
    int i;
    FILE* fp=fopen("deal.dat", "rb");
    if (fp == NULL)
    {
        return 0;
    }
    i = fread(person, sizeof(FINANCE), DEAL, fp);
    fclose(fp);
    return i;
}
/*
   函数功能：保存数据文件
   函数参数：保存几条数据
   函数返回值：无
*/
void SaveData(int i)
{
    int written;
    FILE* fp=fopen("deal.dat", "wb");
    if (fp == NULL)
    {
        perror("Open deal.dat error:");
        return;
    }
    written = fwrite(person, sizeof(FINANCE), i, fp);
    if (written != DEAL)
        perror("Write deal.dat error:");
    fclose(fp);
}
main()
{
    char key;
    int i;
    i = LoadData();
    while(1)
    {
        Menu();
        key=bioskey(0);
        switch(key)
        {
            case '1':  InputOneDeal(person+i);
                       i++;
                       break;
            case '2':  PrintBalance(person);
                       break;
            case '3':  PrintOneYear(person);
                       break;
            case '4':  SaveData(i);
                       exit(0);
            default :  break;
         }
    }
}
```