第 1 章　习题解答

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>
int main(void)
{
    printf("%d\n", 123+456);
    return 0;
}
```

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　说明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〗　（1）C　（2）B　（3）A　（4）B

（5）D　（6）D　（7）D　（8）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）$\left|(a+b)(c+d)+2\right|$　　（3）$(\ln x+\sin y)/2$

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

〖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，否则若 y 为角度，则应写成

```
(log(x)+ sin(3.14/180*y))/2 )
```

（4）

```
2*3.1415*r
```

或

```
2*PI*r（其中 PI 被定义为符号常量）
```

（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　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_____。

```
int main(void)
{
    int  a = 2, c = 5;
    printf("a = %%d, b = %%d\n", a, c);
    return 0;
}
```

A）a = %2, b = %5　　B）a = 2, b = 5　　C）a=%%d, b=%%d　D）a=%d, b=%d

（3）有以下程序

```
 #include  <stdio.h>
int main(void)
{
    int a, b, c, d;
    scanf("%c,%c,%d,%d", &a, &b, &c, &d);
    printf("c,%c,%c,%c\n", a, b, c, d);
    return 0;
}
```

若在 VC6 下运行时从键盘上输入：6,5,65,66<回车>，则输出结果是

A）6,5,A,B　　B）6,5,65,66

C）6,5,6,5　　D）6,5,6,6

〖参考答案〗　（1）C　　（2）D　　（3）A

3.2　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）
```
int main(void)
{
    int  a, b;
    scanf("a = %d, b = %d", &a, &b);
```

```
        printf("a = %d, b = %d\n", a, b);
        return 0;
    }
```

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 和 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 ↙

3.3 分析下列程序，写出程序运行结果。

（1）

```
#include <stdio.h>
int main(void)
{
    char  c1 = 'a', c2 = 'b', c3 = 'c';
    printf("a%cb%cc%cabc\n", c1, c2, c3);
    return 0;
}
```

（2）

```
#include <stdio.h>
int main(void)
{
    int  x = 12 , y = 8;
    printf("\n%5d%5d%5d", !x, x || y, x && y);
     return 0;
}
```

（3）

```
#include <stdio.h>
int main(void)
{
    int  x, y;
    scanf("%2d%*2s%2d", &x, &y);
    printf("%d", x + y);
    return 0;
}
```

程序执行时从键盘输入：1234567↙

（4）

```
#include <stdio.h>
int main(void)
{
    int  a = 2, b = 3 ;
    float  x = 3.5, y = 2.5 ;
    printf("%f", (float)(a+b) / 2 + (int)x % (int)y);
    return 0;
}
```

（5）

```
#include <stdio.h>
int main(void)
{
    int  x = 12, y = 8;
    printf("%d %d\n", x++, ++y);
    printf("%d %d\n", x, y);
    return 0;
}
```

（6）

```
#include <stdio.h>
int main(void)
{
    int  x = 12, y = 8, p, q;
    p = x++;
    q = ++y;
    printf("%d %d\n", p, q);
    printf("%d %d\n", x, y);
    return 0;
}
```

〖参考答案〗

（1）aabbccabc　　（2）0　1　1　　（3）68

（4）3.500000　　（5）12　　（6）12　13　13

9　　9　9　9

3.4　从键盘输入三角形的三边长为 *a,b,c*，按下面公式计算并输出三角形的面积。

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

〖参考答案〗　程序运行时应保证输入的 *a*，*b*，*c* 的值满足三角形成立的条件，这样计算得到的三角形面积才有意义。

另外，将面积计算的数学公式写成 area = sqrt(s*(s–a)*(s–b)*(s–c))是正确的，但写成 area = sqrt(s(s–a)(s–b)(s–c))则是错误的。此外，当 *a*、*b*、*c* 被声明为整型变量时，将数学公式

$$s=\frac{1}{2}(a+b+c)$$

写成 s = 0.5*(a+b+c)或 s = (a+b+c)/2.0 都是正确的。而写成 s = 1/2*(a+b+c)或 s= (a+b+c)/2，虽然合法，但结果是错误的。前者是因为 1/2 的值为 0，使得整个表达式的值为 0，从而导致 *s* 的值为 0。因为整型数的相除结果仍为整型，所以后者的最终计算结果中的小数位被截断。

参考程序如下：

```
#include <stdio.h>
#include  <math.h>
int main(void)
{
    float  a, b, c;                          // a,b,c 表示三角形的三边
    float  s, area;
    printf("Input a,b,c:");
    scanf("%f,%f,%f",&a,&b,&c);
    s = 1.0 / 2 * (a + b + c);               // 注意，这里不能写成 1/2，否则值为 0
    area = sqrt(s * (s - a) * (s - b) * (s - c));
    printf("area=%.2f\n", area);
    return 0;
}
```

程序的运行结果如下：

```
Input a,b,c:3,4,5↙
area=6.00
```

3.5　编程从键盘输入圆的半径 *r*，计算并输出圆的周长和面积。

〖参考答案〗　将计算圆周长和面积公式中的π定义为符号常量。

```
#include <stdio.h>
#define      PI       3.14
int main(void)
{
    float  r ;                               // r 为半径
    printf("Input r:");
    scanf("%f", &r);
    printf("circum=%.2f,area=%.2f\n", 2*PI*r, PI*r*r);
    return 0;
}
```

程序的运行结果如下：

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

1.4　习题 4 及参考答案

4.1　选择题。

（1）在下列条件语句中，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结果为_____。

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

（5）以下能判断 ch 是数字字符的选项是_____。

A）if (ch >= '0' && ch<='9')　　B）if (ch >= 0 && ch<=9)

C）if ('0'<= ch <='9')　　D）if (0<= ch <=9)

（6）为了避免嵌套的条件语句 if-else 的二义性，C 语言规定 else 总是与_____配对。

A）同一行上的 if　　B）缩排位置相同的 if

C）其之前最近的未曾配对的 if　　D）其之后最近的未曾配对的 if

（7）下列说法中错误的是_____。

A）嵌套循环的内层和外层循环的循环控制变量不能同名

B）执行嵌套循环时是先执行内层循环，后执行外层循环

C）如果内外层循环的次数是固定的，则嵌套循环的循环次数等于外层循环的循环次数与内层循环的循环次数之积

D）如果一个循环的循环体中又完整地包含了另一个循环，则称为嵌套循环

（8）有以下程序

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

程序执行后的输出结果是_____。

A）自然数 1～9 的累加和　　B）自然数 1～10 的累加和

C）自然数 1～9 中的奇数之和　　D）自然数 1～10 中的偶数之和

（9）有以下程序

```
#include <stdio.h>
int main(void)
{
    int n = 0,p;
    do{
        scanf("%d", &p);
        n++;
    } while (p!=12345 && n<3);
    return 0;
}
```

此处 do-while 循环的结束条件是_____。

A）p 的值不等于 12345 并且 n 的值小于 3

B）p 的值等于 12345 并且 n 的值大于等于 3

C）p 的值不等于 12345 或者 n 的值小于 3

D）p 的值等于 12345 或者 n 的值大于等于 3

（10）有以下程序

```
#include <stdio.h>
int main(void)
{
    int  k = 0;
    while (k = 1)
    {
        k++;
    }
    return 0;
}
```

while 循环执行的次数是_____。

A）无限次　　B）有语法错，不能执行

C）一次也不执行　　D）执行一次

〖参考答案〗

（1）D　（2）B　（3）D　（4）D　（5）A

（6）C　（7）B　（8）D　（9）D　（10）A

4.2　写出程序的运行结果。

（1）下面程序运行结果为_____。

```
#include <stdio.h>
int main(void)
{
    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");
        return 0;
}
```

（2）若从终端上由第一列开始输入：right?↙，则程序运行结果为_____。

```
#include <stdio.h>
int main(void)
{
    char  c;
    c = getchar();
    while (c != '?')
    {
        putchar(c);
        c = getchar();
    }
    return 0;
}
```

（3）对下面程序，若输入数据同上，则程序运行结果为_____。

```
#include <stdio.h>
int main(void)
{
    char  c;
    while ((c = getchar()) != '?')
    {
        putchar(c);
    }
    return 0;
}
```

（4）对下面程序，若输入数据同上，则程序运行结果为_____。

```
#include <stdio.h>
int main(void)
{
    while (putchar (getchar()) != '?') ;
    return 0;
}
```

（5）对下面程序，若运行时输入：abcdefg$abcdefg↙，则程序运行结果分别为_____。

```
#include <stdio.h>
int main(void)
{
    char  c;
    while ((c = getchar()) != '\n')
    {
        putchar(c);
    }
    printf("End!\n");
    return 0;
}
```

（6）对下面程序，若输入数据同上，则程序的运行结果为_____。

```
#include <stdio.h>
int main(void)
{
    char  c;
    while ((c = getchar()) != '$')
    {
        putchar(c);
    }
    printf("End!\n");
    return 0;
}
```

（7）下面程序的运行结果为_____。

```
#include <stdio.h>
int main(void)
{
    int  n;
    for (n=1; n<=5; n++)
    {
        if (n % 2)
            printf("*");
        else
            continue;
        printf("#");
    }
    printf("$\n");
    return 0;
}
```

〖参考答案〗

（1）

```
over!
```

（2）

```
right
```

（3）

```
right
```

（4）

```
right?
```

（5）

```
abcdefg$abcdefgEnd!
```

（6）

```
abcdefgEnd!
```

（7）

```
*#*#*#$
```

4.3　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。

（1）从键盘任意输入一个年号，判断它是否是闰年。若是闰年输出“Yes”，否则输出“No”。

已知符合下列条件之一者是闰年：（a）能被 4 整除，但不能被 100 整除。（b）能被 400 整除。

```
#include <stdio.h>
int main(void)
{
    int  year, flag;
    printf("Enter year:");
    scanf("%d",&year );
    if (   ①   )
    {
        flag = 1;                          // 如果 year 是闰年，则标志变量 flag 置 1
    }
    else
    {
       flag = 0;                           // 否则，标志变量 flag 置 0
    }
    if (   ②   )
    {
        printf("%d is a leap year!\n",year);          // 输出 "是闰年"
    }
    else
    {
       printf("%d is not a leap year!\n",year);       // 输出 "不是闰年"
    }
    return 0;
}
```

〖参考答案〗

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

② flag

（2）判断从键盘输入的字符是数字字符、大写字母、小写字母、空格还是其他字符。

```
#include <stdio.h>
int main(void)
{
    char  ch;
    ch = getchar();
    if (   ①   )
    {
       printf("It is an English character!\n");
    }
    else if (   ②   )
    {
           printf("It is a digit character!\n");
    }
    else if (   ③   )
    {
           printf("It is a space character!\n");
    }
    else
```

```
        {
            printf("It is other character!\n");
        }
        return 0;
    }
```

〖参考答案〗

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

② ch<='9' && ch>='0'

③ ch==' '

（3）华氏和摄氏温度的转换公式为 $C=5/9\times(F-32)$。式中，C 表示摄氏温度，F 表示华氏温度。要求：华氏 0℉～300℉，每隔 20℉输出一个华氏温度对应的摄氏温度值。

```
#include <stdio.h>
int main(void)
{
    int  upper = 300, step = 20;
    float  fahr = 0, celsius;
    while (    ①    < upper)
    {
        ___②___;
        printf("%4.0f\t%6.1f\n", fahr, celsius);
        ___③___;
    }
    return 0;
}
```

〖参考答案〗

① fahr

② celsius = 5.0 / 9 * (fahr - 32)

③ fahr = fahr + step

4.4　编程判断输入整数的正负性和奇偶性。

〖参考答案 1〗

```
#include <stdio.h>
int main(void)
{
    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，则是正偶数
        {
            printf("%d is a positive even\n", m);
        }
        else                                      // 不能被 2 整除，则是正奇数
        {
            printf("%d is a positive odd\n", m);
```

```
        }
    }
    else if (m < 0)                                    // 判断是否为负数
    {
        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
        }
    }
    else
    {
        printf("%d is zero.It is an even\n", m);
    }
    return 0;
}
```

〖参考答案 2〗

```
#include <stdio.h>
int main(void)
{
    int  m;
    printf("Input m:");
    scanf("%d", &m);                                   // 输入一个整数
    if (m > 0)                                         // 判断是否为正数
    {
        printf("%d is a positive ", m);                // 是正数
    }
    else if (m < 0)                                    // 判断是否为负数
    {
        printf("%d is a negative ", m);                // 是负数
    }
    else
    {
        printf("%d is zero. It is an ", m);
    }
    printf(m%2==0 ? "even\n" : "odd\n");               // 是偶数输出"even",否则输出"odd"
    return 0;
}
```

程序的 5 次测试结果如下：

① Input m:6↙
6 is a positive even

② Input m:-7↙
6 is a negative odd

③ Input m:-6↙
-6 is a negative even
④ Input m:7↙
7 is a positive odd
⑤ Input m:0↙
0 is zero. It is an even

4.5　编程计算下面的分段函数，根据从键盘输入的 x 值，在屏幕上输出 y 值。

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

〖参考答案〗

```
#include <stdio.h>
#include <math.h>
int main(void)
{
    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);
    return 0;
}
```

程序的三次测试结果如下：

① Input x:4↙
y=0.018316
② Input x:0↙
y=1.000000
③ Input x:-4↙
y=-0.018316

4.6　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）。

〖参考答案〗

```
#include <stdio.h>
#include <math.h>
#define EPS  1e-1
int main(void)
{
    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)<=EPS||fabs(b-c)<=EPS||fabs(c-a)<=EPS)
        {
            printf("等腰");
            flag = 0;
        }
        if (fabs(a*a+b*b-c*c) <= EPS || fabs(a*a+c*c-b*b) <= EPS
                                     || fabs(c*c+b*b-a*a) <= EPS)
        {
            printf("直角");
            flag = 0;
        }
        if (flag)
        {
            printf("一般");
        }
        printf("三角形\n");
    }
    else
    {
        printf("不是三角形\n");
    }
    return 0;
}
```

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

4.7　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〗 使用 printf()函数输出一个简单的菜单，通过 switch 语句进行菜单项选择。

```
#include <stdio.h>
int main(void)
{
    int  choice;
    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
    scanf("%d", &choice);                          // 输入选项
    switch (choice)                                // 通过 switch 语句选择
    {
        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");
    }
     return 0;
}
```

程序的两次测试结果如下：

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:4↙
Selection error!
```

4.8　读入一个年份和月份，输出该月有多少天（考虑闰年），用 switch 语句编程。

〖参考答案〗

```
#include <stdio.h>
int main(void)
{
    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");
    }
    return 0;
}
```

程序的三次测试结果如下：

```
① Input year,month：1988,5↙
            31 days
② Input year,month：1988,2↙
            29 days
③ Input year,month：1989,2↙
            28 days
```

4.9　编程计算 1+3+5+7+…+99+101 的值。

〖参考答案 1〗

利用 for 循环语句实现，在循环体外为 sum 赋初值 0。

```
#include <stdio.h>
int main(void)
{
    int  i, sum = 0;
    for (i=1; i<=101; i=i+2)
    {
        sum = sum + i;
    }
    printf("sum=%d\n", sum);
    return 0;
}
```

〖参考答案 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>
int main(void)
{
    int  i = 1, sum = 0;
    while (i <= 101)
    {
        sum = sum + i;
        i = i + 2;
    }
    printf("sum = %d\n", sum);
    return 0;
}
```

程序的运行结果如下：

```
sum = 2601
```

4.10　编程计算 1×2×3+3×4×5+…+99×100×101 的值。

〖参考答案〗

用累加求和算法，通项公式为 term=i*(i+1)*(i+2)（i=1, 3, …, 99）或 term=(i−1)*i*(i+1)（i=2, 4, …, 100），步长为 2。

```
#include <stdio.h>
int main(void)
{
    long  i;                                    // 注意，这里的 i 必须被定义为 long 类型
    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);
    return 0;
}
```

程序的运行结果如下：

```
sum = 13002450
```

4.11　编程计算 1!+2!+3!+4!+…+10！的值。

〖参考答案 1〗 用累加求和算法，累加项为 term=term*i（i=1, 2, …, 10），term 初值为 1。

```
#include <stdio.h>
int main(void)
{
    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);
    return 0;
}
```

〖参考答案 2〗 采用双重循环计算，用内层循环求阶乘 *i*!，外层循环控制累加的项数。

```
#include <stdio.h>
int main(void)
{
    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);
    return 0;
}
```

程序的运行结果如下：

```
1!+2!+…+10! = 4037913
```

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

〖参考答案〗 用累加求和算法，累加项为 term=term*10+a（i=1, 2, …, n），term 初值为 0。

```
#include <stdio.h>
int main(void)
{
    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;                  // 求累加项
        sum = sum + term;                      // 进行累加
    }
    printf("sum = %ld\n", sum);
    return 0;
}
```

程序的运行结果如下：

```
Input a,n:2,4↙
```

```
sum = 2468
```

4.13　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 1〗 采用累乘求积算法，累乘项为 term=n*n/((n-1)*(n+1))（n=2, 4, …, 100），步长为 2。

```
#include  <stdio.h>
int main(void)
{
    double  term, result = 1;                                   // 累乘项初值应为 1
    int  n;
    for (n=2; n<=100; n=n+2)
    {
        term = (double)( n * n)/(( n - 1) * ( n + 1));          // 计算累乘项
        result = result * term;
    }
    printf("result = %f\n", 2*result);
    return 0;
}
```

〖参考答案 2〗

采用累乘求积算法，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>
int main(void)
{
    double  term, result = 1;
    int  n;
    for (n=1; n<=50; n++)
    {
        term = (double)(2*n*2*n)/((2*n-1)*(2*n+1));             // 计算累乘项
        result = result * term;
    }
    printf("result = %f\n", 2*result);
    return 0;
}
```

程序的运行结果如下：

```
result = 3.126078
```

4.14　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 1〗

采用累加求和算法 e=e+term。利用前项计算后项寻找累加项的构成规律：由$\frac{1}{2!}=\frac{1}{1!}\div 2$，$\frac{1}{3!}=\frac{1}{2!}\div 3$，…可以发现前后项之间的关系是 $term_n=term_{n-1}\div n$，写成 C 语言的语句为“term=term/n; ”，term 的初值为 1.0，n 的初值也为 1，n 按 n→n+1 变化。统计累加项数的计数

器变量为 count，初值为 0。

```
#include  <math.h>
#include <stdio.h>
int main(void)
{
    int  n = 1, count = 1;
    double  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);
    return 0;
}
```

〖参考答案 2〗

先计算 1!，2!，3!，…，再将其倒数作为累加项 term。

```
#include  <math.h>
#include <stdio.h>
int main(void)
{
    int  n = 1, count = 1;
    double  e = 1.0, term = 1.0;
    long  fac = 1;
    for (n=1; fabs(term)>=1e-5; n++)
    {
        fac = fac * n;
        term = 1.0 / fac;
        e = e + term;
        count++;
    }
    printf("e = %f, count = %d\n", e, count);
    return 0;
}
```

程序的运行结果如下：

```
e = 2.718282, count = 10
```

4.15　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〗 累加项为 term=sign/n，分子 sign=−sign，初值为 1，分母 n=n+1，初值为 1。

```
#include  <stdio.h>
#include  <math.h>
int main(void)
```

```
{
    int  n = 1;
    float  term = 1, sign = 1, sum = 0;
    while (fabs(term) >= 1e-4)                          // 判末项大小
    {
        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);
    return 0;
}
```

程序的运行结果如下：

```
sum = 0.693092
```

4.16　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〗 x 由键盘输入，采用累加求和算法，sum=sum+term，sum 初值为 x，利用前项求后项的方法计算累加项 term= −term*x*x/((n+1)*(n+2))，term 初值为 x，n 初值为 1，n=n+2。

```
#include  <math.h>
#include <stdio.h>
int main(void)
{
    int  n = 1, count = 1;
    double  x, sum, term;
    printf("Input x: ");
    scanf("%lf", &x);
    sum = x;
    term = x;                                                   // 累加项赋初值
    do{
        term = -term * x * x / ((n + 1) * (n + 2));             // 计算累加项
        sum = sum + term;                                       // 累加
        n = n + 2;
        count++;
    }while (fabs(term) >= 1e-5);
    printf("sin(x) = %f, count = %d\n", sum, count);
    return 0;
}
```

程序的运行结果如下：

```
Input x:3↙  (输入弧度值)
sin(x) = 0.141120, count = 9
```

4.17　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 $153=1^3+5^3+3^3$。

〖参考答案 1〗

首先确定水仙花数 n 可能存在的范围，因为 n 是一个三位数，所以范围确定为 n 从 100 变化到 999，分离出 n 的百位 i、十位 j、个位 k 后，只要判断 n 是否等于 i*i*i+j*j*j+k*k*k 即可知道 n 是否是水仙花数。

```
#include <stdio.h>
int main(void)
{
    int  i, j, k, n;
    printf("result is:");
    for (n=100; n<1000; n++)
    {
        i = n / 100;                                    // 分离出百位
        j = (n - i * 100) / 10;                         // 分离出十位
        k = n % 10;                                     // 分离出个位
        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");
    return 0;
}
```

〖参考答案 2〗

```
#include <stdio.h>
int main(void)
{
    int  i, j, k;
    printf("result is:");
    for (i=1; i<=9; i++)
        for (j=1; j<=9; j++)
            for (k=0; k<=9; k++)
            {
                if (i*i*i+j*j*j+k*k*k == 100*i+10*j+k)
                {
                    printf("%d\t", 100*i+10*j+k);
                }
            }
    printf("\n");
    return 0;
}
```

程序的运行结果如下：

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

4.18　参考例 4.2，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〗 对输入的整数取绝对值（用函数 fabs()），即可实现忽略整数前的正负号。

```
#include <stdio.h>
#include <math.h>
int main(void)
{
    int  i1, i2, i3, i4, k, n;
    printf("Input data is:");
    scanf("%d", &n);
    k = abs(n);                                          // 对整数取绝对值
    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);
    return 0;
}
```

程序的两次测试结果如下：

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

4.19　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？

〖参考答案 1〗

设兵数为 x，按题意 x 应满足关系式：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。

```
#include <stdio.h>
int main(void)
{
    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)
        {
            printf("x = %d\n", x);
            find = 1;
        }
        x++;
    }
    return 0;
```

```
}
```

〖参考答案 2〗

```
#include <stdio.h>
int main(void)
{
    int  x = 1;
    while (1)
    {
        if (x%5==1 && x%6==5 && x%7==4 && x%11==10)
        {
            printf("x = %d\n", x);
            break;
        }
        x++;
    }
    return 0;
}
```

〖参考答案 3〗

```
#include <stdio.h>
int main(void)
{
    int  x = 0, find = 0;
    do{
        x++;
        find = x%5==1 && x%6==5 && x%7==4 && x%11==10;
    } while (!find);
    printf("x = %d\n", x);
    return 0;
}
```

〖参考答案 4〗

```
#include <stdio.h>
int main(void)
{
    int  x = 0;
    do{
        x++;
    } while (!(x%5==1 && x%6==5 && x%7==4 && x%11==10));
    printf("x = %d\n", x);
    return 0;
}
```

程序的运行结果如下：

```
x = 2111
```

4.20　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 1〗

设阶梯数为 x，按题意阶梯数应满足关系式：x%2 == 1 && x%3 == 2 && x%5 == 4 && x%6 == 5 && x%7 == 0，采用穷举法对 x 从 1 开始试验，可计算出阶梯共有多少阶。

```
#include <stdio.h>
int main(void)
{
    int  x = 1, find = 0;
    while (!find)
    {
        if (x%2==1 && x%3==2 && x%5==4 && x%6==5 && x%7==0)
        {
            printf("x = %d\n", x);
            find = 1;
        }
        x++;
    }
    return 0;
}
```

〖参考答案 2〗

```
#include <stdio.h>
int main(void)
{
    int  x = 1;
    while (1)
    {
        if (x%2==1 && x%3==2 && x%5==4 && x%6==5 && x%7==0)
        {
            printf("x = %d\n", x);
            break;
        }
        x++;
    }
    return 0;
}
```

〖参考答案 3〗

```
#include <stdio.h>
int main(void)
{
    int  x = 0, find = 0;
    do {
```

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

〖参考答案 4〗

```
#include <stdio.h>
int main(void)
{
    int  x = 0;
    do {
        x++;
    } while (!(x%2==1 && x%3==2 && x%5==4 && x%6==5 && x%7==0));
    printf("x = %d\n", x);
    return 0;
}
```

程序的运行结果如下：

```
x = 119
```

4.21　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？

〖参考答案〗 设任取的红、白、黑球个数分别为 i、j、k。依题意，红、白、黑球个数的取值范围分别为 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。

```
#include <stdio.h>
int main(void)
{
    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);
                }
            }
        }
    }
    return 0;
}
```

程序的运行结果如下：

```
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
```

4.22　鸡兔同笼，共有 98 个头，386 只脚，编程求鸡、兔各多少只。

〖参考答案〗 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 和 y 同时满足条件 $2x+4y=386$，则输出 x、y 的值。

```
#include <stdio.h>
int main(void)
{
    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);
        }
    }
    return 0;
}
```

程序的运行结果如下：

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

4.23　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？

〖参考答案〗 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。注意：程序中的 z/3.0 不能写成 z/3，因为 z/3 是整数除法运算，以 z=75 为例，75/3 的结果与 76/3 和 77/3 的结果一样，因此这样将导致输出结果中多出几组不合理的解。

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

```
    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);
            }
        }
    }
    return 0;
}
```

程序的运行结果如下：

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

4.24　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？

〖参考答案〗 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 29 枚，2 分硬币最多可换 72 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，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。

```
#include <stdio.h>
int main(void)
{
    int  x, y, z, count = 0;
    for (x=1; x<=29; x++)
    {
        for (y=1; y<=72; 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);
    return 0;
}
```

程序的运行结果如下：

```
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
```

4.25　编程输出如下上三角形式的九九乘法表。

```
1  2  3  4  5  6  7  8  9
   4  6  8  10 12 14 16 18
      9  12 15 18 21 24 27
         16 20 24 28 32 36
            25 30 35 40 45
               36 42 48 54
                  49 56 63
                     64 72
                        81
```

〖参考答案〗 根据题意，第 1 行从第 1 列开始输出 9 列数字，第 2 行从第 2 列开始输出 8 列数字，…，第 9 行从第 9 列开始输出 1 列数字，为了输出上三角形式，每列数字占 4 位字符宽度，需要在第 n 行先输出 $4\times n-4$ 个空格，再按%4d 格式输出两数相乘结果。

```
#include <stdio.h>
int main(void)
{
    int  m, n, i;
    for (n=1; n<10; 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 n 行 m 列中的 m*n 的值
        }
        printf("\n");                       // 输出换行符,准备输出下一行
    }
    return 0;
}
```

4.26　按如下格式输出 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

〖参考答案〗 表内第 1 行输出 0～9 的平方根，第 2 行输出 10～19 的平方根，第 3 行输出 20～29 的平方根，…，第 10 行输出 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)。

```
#include <stdio.h>
#include <math.h>
int main(void)
{
    int  m, n;
    for (m=0; m<10; m++)
    {
        printf("%7d", m);                          // 输出表头
    }
    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("%7.3f", sqrt(n*10+m));
        }                                          // 输出第 m 行 n 列中的值
        printf("\n");                              // 输出换行符,准备输出下一行
    }
    return 0;
}
```

4.27　分别编程输出以下 4 种不同的图案。

（1）
```
******
  ******
    ******
      ******
```

（2）
```
    ******
      ******
        ******
          ******
```

（3）
```
*
***
*****
*******
```

（4）
```
    *
   ***
  *****
 *******
```

〖参考答案〗

（1）

```
#include <stdio.h>
int main(void)
```

```
{
     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");
    }
    return 0;
}
```

（2）

```
#include <stdio.h>
int main(void)
{
    int  i, j, k;
    for (i=1; i<=4; i++)                    // i 控制行数
    {
        for (j=1; j<=4-i; j++)              // 随着行数的增加，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
    }
    return 0;
}
```

（3）

```
#include <stdio.h>
int main(void)
{
    int  i , 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*的个数
        {
```

```
            printf("*");
        }
        printf("\n");                                    // 输出一行后换行
    }
    return 0;
}
```

（4）

```
#include <stdio.h>
int main(void)
{
    int  i , j, k ;
    for (i=1; i<=4; i++)                    // 控制行数
    {
        for (j=1; j<=8-i; j++)              // 随着行数的增加，输出递减数目的空格
        {
            printf(" ");
        }
        for (k=1; k<=(2*i-1); k++)
        {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

1.5　习题 5 及参考答案

5.1　多项选择题。

（1）下列关于调试的说法中，正确的是_____。

A）可以一条语句一条语句地执行

B）调试过程中如果修改了源代码，不需要重新编译就能继续运行

C）可以随时查看变量值

D）可以跟踪进入用户自己编写的函数内部

（2）下面所列举的函数名正确且具有良好风格的是_____。

A）abcde()　　B）GetNumber()　　C）change_directory()

D）gotofirstline()　　E）Find@()　　F）2_power()

〖参考答案〗

（1）ACD　　（2）BC

5.2　单项选择题。

（1）阅读程序，选择程序的运行结果。

```
#include <stdio.h>
int Try(int n)
```

```
{
    if (n > 0)
        return(n*Try(n-2));
    else
        return 1;
}
int main(void)
{
    int  x;
    x = Try(5);
    printf("%d\n", x);
    return 0;
}
```

A）15　　B）120　　C）1　　D）前面三个答案都是错的

（2）下面对函数 Fun 有两次调用，请选择程序的运行结果。

```
#include <stdio.h>
int Fun(int m)
{
    static int  n = 0;
    m /= 2;
    m = m * 2;
    if (m)
    {
        n *= m;
        return(Fun(m-2));
    }
    else    return n;
}
int main(void)
{
    int  a, i, j;
    for (i=0; i<2; i++)
    {
        a = Fun(4+i);
        printf("%d\n", a);
    }
     return 0;
}
```

① 第 1 次输出的结果　　A）8　　B）0　　C）64　　D）4

② 第 2 次输出的结果　　A）8　　B）0　　C）64　　D）4

〖参考答案〗

（1）A　　（2）① B　② B

5.3　指出下面这段程序中的错误。

```
#include <stdio.h>
int main(void)
{
    printf("Delay()=%d\n", Delay());
```

```
    return 0;
}
int Delay()
{
    int  i, times;
    for (i=0; i<times; i++)
        ;
}
```

〖参考答案〗

（1）Delay()未先定义就使用；（2）没有参数的函数应该注明 void；（3）变量 times 未赋值就使用；（4）Delay()函数中没有 return 语句。

5.4　写出下面程序的运行结果。

（1）

```
#include <stdio.h>
int Square(int i)
{
    return i * i;
}
int main(void)
{
    int  i = 0;
    i = Square(i);
    for ( ; i<3; i++)
    {
        static int  i = 1;
        i += Square(i);
        printf("%d,", i);
    }
    printf("%d\n", i);
    return 0;
}
```

〖参考答案〗　2，6，42，3

补充说明：函数 main()中 for 语句内部定义的静态变量 i 和 for 语句前面定义的自动变量 i 是两个不同的变量，因为它们各自的作用域不同，所以虽然同名，但互不干扰。for 语句内部定义的静态变量 i 只在 for 语句的一对“{}”内起作用，在此“{}”之外，是 for 语句前面定义的自动变量 i 的作用域，这两个变量的值的变化情况如下所示：

for 循环	自动变量 i 的值	循环开始时静态变量 i 的值	循环结束时静态变量 i 的值
第 1 次 for 循环	0	1	2
第 2 次 for 循环	1	2	6
第 3 次 for 循环	2	6	42
for 循环结束时	3	—	—

（2）

```
#include <stdio.h>
void Bin(int x)
{
    if (x/2 > 0)
```

```
        Bin(x/2);
    printf("%d\n", x%2);
}
int main(void)
{
    Bin(12);
     return 0;
}
```

〖参考答案〗

```
1
1
0
0
```

5.5　程序填空。

（1）下面函数是求阶乘的递归函数，请将程序补充完整。

```
long Fact(int n)
{
    if (n < 0)
        return 0;
    if (n==1 || n==0)
        ____①____;
    else
        ____②____;
}
```

（2）Y()是实现 *n* 层嵌套平方根计算的函数，其公式如下，请将程序补充完整。

$$y(x)=\sqrt{x+\cdots+\sqrt{x+\sqrt{x}}}$$

```
double Y(double x, int n)
{
    if (n == 0)
        return 0;
    else
        return (square(x + ____①____));
}
```

（3）函数 Sum(int n)是用递归方法计算 $\sum_{i=1}^{n} i$ 的值，请补充程序中缺少的内容。

```
int Sum(int n)
{
    if (n <= 0)
    printf("data error\n");
    if (n == 1)
        ____①____;
    else
        return ____②____;
}
```

〖参考答案〗

（1）①　return 1　　②　return n*Facto(n-1)

（2）①　Y(x, n-1)

（3）①　return 1　　②　n+Sum(n-1)

5.6　设计一个函数，用来判断一个整数是否为素数。

〖参考答案 1〗

```
#include <math.h>
#include <stdio.h>
int IsPrimeNumber(int number);
int main(void)
{
    int  n, ret;
    printf("Input n:");
    scanf("%d", &n);
    ret = IsPrimeNumber(n);
    if (ret != 0)
    {
        printf("%d is a prime number\n", n);
    }
    else
    {
        printf("%d is not a prime number\n", n);
    }
     return 0;
}
// 函数功能：判断 number 是否是素数，函数返回非 0 值表示是素数，否则不是素数
int IsPrimeNumber(int number)
{
    int  i;
    if (number <= 1)
        return 0;                                   // 负数、0 和 1 都不是素数
    for (i=2; i<=sqrt(number); i++)
    {
        if (number % i == 0)                        // 被整除，不是素数
            return 0;
    }
    return 1;
}
```

〖参考答案 2〗

函数 IsPrimeNumber()还可以用如下方法编程。

```
int IsPrimeNumber(int number)
{
    int  i, flag = 1;                               // 标志变量置为真，假设是素数
    if (number <= 1)
        flag = 0;                                   // 负数、0 和 1 都不是素数
    for (i=2; i<=sqrt(number); i++)
    {
```

```
            if (number%i == 0)                    // 被整除，不是素数
            {
                flag = 0;
            }
        }
        return flag;
    }
```

程序的三次测试结果如下：

① Input n:4↙
4 is not a prime number
② Input n:7↙
7 is a prime number
③ Input n:1↙
1 is not a prime number

5.7　编程计算组合数 $p=C_m^k=\frac{m!}{k!(m-k)!}$ 的值。

〖参考答案〗 此公式中用到三次阶乘，所以可把计算阶乘设计为一个函数。因为负数没有阶乘，所以函数的参数应采用无符号类型。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 被定义为 double 类型。注意，0 的阶乘为 1。

```
#include <stdio.h>
unsigned long Factorial(unsigned int number);
int main(void)
{
    int  m, k;
    double  p;
    do{
        printf("Please input m,k (m>=k>0):");
        scanf("%d, %d", &m, &k);
    } while (m<k || m<0 || k<0) ;
    p = (double)Factorial(m) / (Factorial(k) * Factorial (m-k));
    printf("p=%.0f\n", p);
    return 0;
}
// 函数功能：计算无符号整型数 number 的阶乘
unsigned long Factorial(unsigned int number)
{
    unsigned long  i, result = 1;
    for (i=2; i<=number; i++)
    {
        result *= i;
    }
    return result;
}
```

程序的三次测试结果如下：

① Please input m,k (m>=k>0): 3,2↙

```
    p=3
② Please input m,k (m>=k>0): 2,3↙
    Please input m,k (m>=k>0): 3,3↙
    p=1
③ Please input m,k (m>=k>0): -2,-4↙
    Please input m,k (m>=k>0): 4,2↙
    p=6
```

5.8　设计一个函数 MinCommonMultiple()，计算两个正整数的最小公倍数。

〖参考答案〗 对于 a 和 b 两个正整数，a 和 b 的公倍数可以从 a 的倍数中寻找 b 的倍数，或者从 b 的倍数中寻找 a 的倍数，因为 $b\times a$ 一定是 a 和 b 的公倍数，所以寻找 a 和 b 最小公倍数的范围不会超过 $b\times a$。因此，计算 a 和 b 的最小公倍数的方法为：在 a，$2\times a$，$3\times a$，…，$b\times a$ 中，从前往后依次判断是否能被 b 整除，第一个能被 b 整除的数就是 a 和 b 的最小公倍数。

```
#include <stdio.h>
int MinCommonMultiple(int a, int b);
int main(void)
{
    int  a, b, x;
    printf("Input a,b:");
    scanf("%d,%d", &a, &b);
    x = MinCommonMultiple(a, b);
    if (x != -1)
        printf("MinCommonMultiple = %d\n", x);
    else
        printf("Input error!\n");
    return 0;
}
// 函数功能：计算两个正整数的最小公倍数，-1 表示没有最小公倍数
int MinCommonMultiple(int a, int b)
{
    int  i;
    if (a<=0 || b<=0)
        return -1;                                  // 保证输入的参数为正整数
    for (i=1; i<b; i++)
    {
        if ((i * a) % b == 0)
            return i * a;
    }
    return b * a;
}
```

程序的两次测试结果如下：

```
① Input a,b: 16,24↙
  MinCommonMultiple = 48
② Input a,b: -16,24↙
  Input error!
```

5.9　设计一个函数 MaxCommonFactor()，利用欧几里德算法（也称为辗转相除法）计算两

个正整数的最大公约数。

〖参考答案〗欧几里德算法是计算两个数最大公约数的传统算法。假设有两个整数 *m* 和 *n*，通过连续运用求余运算，直到余数为 0 为止，最后非 0 的余数就是最大公约数。用 *f* 表示计算最大公约数函数，那么 *m* 和 *n* 最大公约数的求解过程可表示为 *f*(*m*, *n*)=*f*(*n*, *m*%*n*)。例如，15 和 50 的最大公约数的求解过程可表示为 *f*(15,50)=*f*(50,15)=*f*(15,5)=*f*(5,0)=5。

```
#include <stdio.h>
int MaxCommonFactor(int a, int b);
int main(void)
{
    int  a, b, x;
    printf("Input a,b:");
    scanf("%d,%d", &a, &b);
    x = MaxCommonFactor(a, b);
    if (x != -1)
    {
        printf("MaxCommonFactor = %d\n", x);
    }
    else
    {
        printf("Input error!\n");
    }
    return 0;
}
// 函数功能： 计算两个正整数的最大公约数，-1 表示没有最大公约数
int MaxCommonFactor(int a, int b)
{
    int  r;
    if (a<=0 || b<=0)
        return -1;                                    // 保证输入的参数为正整数
    do{
        r = a % b;
        a = b;
        b = r;
    } while (r != 0);
    return  a;
}
```

程序的两次测试结果如下：

```
① Input a,b: 16,24↙
  MaxCommonFactor = 8
② Input a,b: -16,24↙
  Input error!
```

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

〖参考答案〗 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三个性质，直到 *a* 和 *b* 相等为止，这时 *a* 或 *b* 就是它

们的最大公约数。这三个性质用公式表示如下：

- 性质 1：当 $a>b$ 时，计算 a 与 b 的公约数等价于计算 $a–b$ 与 b 的公约数。
- 性质 2：当 $a<b$ 时，计算 a 与 b 的公约数等价于计算 $b–a$ 与 b 的公约数。
- 性质 3：当 $a=b$ 时，a 与 b 的公约数等于 a 或 b。

```
#include <stdio.h>
int MaxCommonFactor(int a, int b);
int main(void)
{
    int  a, b, x;
    printf("Input a,b:");
    scanf("%d,%d", &a, &b);
    x = MaxCommonFactor(a, b);
    if (x != -1)
    {
        printf("MaxCommonFactor = %d\n", x);
    }
    else
    {
        printf("Input error!\n");
    }
    return 0;
}
// 函数功能：计算两个正整数的最大公约数，函数返回最大公约数；-1 表示没有最大公约数
int MaxCommonFactor(int a, int b)
{
    if (a <=0 || b <=0)
        return -1;                                  // 保证输入的参数为正整数
    while (a != b)
    {
        if (a > b)
        {
            a = a - b;
        }
        else if (b > a)
        {
            b = b - a;
        }
    }
    return a;
}
```

程序的两次测试结果如下：

① Input a,b: 16,24↙
MaxCommonFactor = 8

② Input a,b: -16,24↙
Input error!

5.11　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。

〖参考答案〗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。

```
#include <stdio.h>
int CommonFactors(int a, int b);
int main(void)
{
    int  sub;
    while((sub=CommonFactors(100, 50)) > 0)
    {
        static int counter = 1;
        printf("Common factor %d is %d\n", counter++, sub);
    }
     return 0;
}
// 函数功能：指明计算哪两个数的公约数
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
    }
    // 因从大到小求公约数，所以从 a,b 的最小值开始查找公约数，直到全部找到为止
    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;
}
```

程序的运行结果如下：

```
Common factor 1 is 50
Common factor 2 is 25
Common factor 3 is 10
Common factor 4 is 5
```

```
Common factor 5 is 2
Common factor 6 is 1
```

5.12　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。

〖参考答案〗 CalcCommonFactorOf()函数把参数保存到模块自己的全局变量内，由 NextCommonFactor()函数使用。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 后开始继续寻找。文件 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
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，函数返回下一个公约数；-1 表示再也没有新的公约数
int NextCommonFactor(void)
{
    if (a <=0 || b <=0)
        return -1;                          // 保证输入的参数正确
    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;
}
```

文件 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"
int main(void)
{
    int  sub;
    CalcCommonFactorOf(100, 50);
    while((sub=NextCommonFactor()) > 0)
```

```
    {
        static int  counter = 1;
        printf("Common factor %d is %d\n", counter++, sub);
    }
    return 0;
}
```

1.6 习题 6 及参考答案

6.1 选择题。

（1）以下能将外部一维数组 a（含有 10 个元素）正确初始化为 0 的语句是_____。

A）int a[10] = (0,0,0,0,0);　　B）int a[10] = {};

C）int a[10] = {0};　　D）int a[10] = {10*1};

（2）以下能对外部二维数组 a 进行正确初始化的语句是_____。

A）int a[2][] = {{1,0,1},{5,2,3}};

B）int a[][3] = {{1,2,1},{5,2,3}};

C）int a[2][4] = {{1,2,1},{5,2},{6}};

D）int a[][3] = {{1,0,2},{},{2,3}};

（3）若二维数组 a 有 m 列，则在 a[i][j]之前的元素个数为_____。

A）j*m+i　　B）i*m+j　　C）i*m+j-1　　D）i*m+j+1

（4）已知有语句“static int a[3][4];，”则数组 a 中各元素_____ 。

A）可在程序运行阶段得到初值 0　　B）可在程序编译阶段得到初值 0

C）不能得到确定的初值　　D）可在程序的编译或运行阶段得到初值 0

（5）判断字符串 s1 是否大于字符串 s2，应当使用_____。

A）if (s1 > s2)　　B）if (strcmp(s1, s2))

C）if (strcmp(s2, s1) > 0)　　D）if (strcmp(s1, s2) > 0)

（6）若用数组名作为函数调用时的实参，则实际上传递给形参的是_____。

A）数组的首地址　　B）数组的第一个元素值

C）数组中全部元素的值　　D）数组元素的个数

（7）在函数调用时，以下说法中正确的是_____。

A）在 C 语言中，实参与其对应的形参各占独立的存储单元

B）在 C 语言中，实参与其对应的形参共占同一个存储单元

C）在 C 语言中，只有当实参与其对应的形参同名时，才共占同一个存储单元

D）在 C 语言中，形参是虚拟的，不占存储单元

（8）C 语言中形参的默认存储类别是_____。

A）自动（auto）　　B）静态（static）

C）寄存器（register）　　D）外部（extern）

（9）C 语言规定，简单变量作为实参时，它与对应形参之间数据的传递方式为_____。

A）地址传递　　B）单向值传递

C）由实参传给形参，再由形参传回给实参　　D）由用户指定传递方式

（10）以下关于数组的描述中正确的是_____。

A）数组的大小是固定的，但可以有不同类型的数组元素
B）数组的大小是可变的，但所有数组元素的类型必须相同
C）数组的大小是固定的，但所有数组元素的类型必须相同
D）数组的大小是可变的，但可以有不同类型的数组元素

（11）下列说法中正确的是_____。
A）C 语言中，不带下标的数组名代表数组的首地址，即第一个元素在内存中的地址
B）C 语言中，数组的下标都是从 1 开始的
C）C 语言中，二维数组在内存中是按列存储的
D）在定义完数组以后，在程序运行过程中也可以改变数组的大小

（12）下列说法中正确的是_____。
A）数组名做函数参数时，修改形参数组元素值会导致实参数组元素值的修改
B）在声明函数的二维数组形参时，通常不指定数组的大小，而用另外的形参来指定数组的大小
C）在声明函数的二维数组形参时，可省略数组第二维的长度，但不能省略数组第一维的长度
D）数组名做函数参数时，是将数组中所有元素的值传给形参

（13）下列说法中错误的是_____。
A）C 语言中，二维数组在内存中是按列存储的
B）C 语言中，数组的下标都是从 0 开始的
C）C 语言中，不带下标的数组名代表数组的首地址，即第一个元素在内存中的地址
D）C89 规定，不能使用变量定义数组的大小，但是在访问数组元素时在下标中可以使用变量或表达式

（14）下列说法中错误的是_____。
A）字符数组可以存放字符串
B）字符数组中的字符串可以进行整体输入输出
C）可以在赋值语句中通过赋值运算符=对字符数组进行整体赋值
D）字符数组中第一个字符的地址就是字符数组中字符串的地址

〖参考答案〗

（1）C　（2）B　（3）B　（4）B　（5）D　（6）A　（7）A
（8）A　（9）B　（10）C　（11）A　（12）A　（13）A　（14）C

6.2　阅读程序，写出运行结果。

（1）

```
#include <stdio.h>
int main(void)
{
    int  a[6][6], i, j;
    for (i=1; i<6; i++)
    {
        for (j=1; j<6; j++)
        {
            a[i][j] = (i / j) * (j / i);
        }
```

```
        }
        for (i=1; i<6; i++)
        {
            for (j=1; j<6; j++)
            {
                printf("%2d", a[i][j]);
            }
            printf("\n");
        }
        return 0;
    }
```

（2）

```
    #include <stdio.h>
    void Func(int x)
    {
        x = 20;
    }
    int main(void)
    {
        int  x = 10;
        Func(x);
        printf("%d", x);
        return 0;
    }
```

（3）

```
    #include <stdio.h>
    void Func(int b[])
    {
        int  j;
        for (j=0; j<4; j++)
        {
            b[j] = j;
        }
    }
    int main(void)
    {
        static int  a[] = {5,6,7,8},i;
        Func(a);
        for (i=0; i<4; i++)
        {
            printf("%d", a[i]);
        }
         return 0;
    }
```

〖参考答案〗

（1）

```
 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
```

（2）10

（3）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。

（1）下面程序模拟了骰子的 6 000 次投掷，用函数 rand()产生 1～6 之间的随机数 face，然后统计每一面 1～6 出现的机会（概率）存放到数组 frequency 中。

```
#include <stdlib.h>
#include <time.h>
#include <stdio.h>
int main(void)
{
    int  face, roll, frequency[7] = {0};
    srand(time (NULL));
    for (roll=1; roll <= 6000; roll++)
    {
        face = ___①___;
        ++___②___;
    }
    printf("%4s%17s\n", "Face", "Frequency");
    for (face=1; face<=6; face++)
    {
        printf("%4d%17d\n", face, frequency[face]);
    }
     return 0;
}
```

〖参考答案〗

①　rand()%6 + 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

（2）输入 10 个整型数据，存入数组 a 中，计算其最大值、最小值及其所在元素的下标位置。

```
#include <stdio.h>
int main(void)
{
    int  a[10], n, max, min, maxPos, minPos;
    for (n=0; n<10; n++)
    {
        scanf("%d",&a[n]);
    }
    max = min = a[0];
    maxPos = minPos = 0;
    for (n=0; n<10; n++)
    {
        if ( ___①___ )
        {
```

```
            max = a[n];
            maxPos = ___②___ ;
        }
        else if ( ___③___ )
        {
            min = a[n];
            minPos = ___④___;
        }
    }
    printf("max=%d, pos=%d\n",max, maxPos);
    printf("min=%d, pos=%d\n",min, minPos);
    return 0;
}
```

〖参考答案〗

① a[n] > max

② n

③ a[n] < min

④ n

（3）下面程序从键盘输入一行字符并统计其中有多少单词。假设单词之间以空格分开。

```
#include <stdio.h>
int main(void)
{
    char  str[20];
    int  i, num;
    gets(str);
    num = str[0] != ' '? 1 : 0;
    for (i=1; str[i]  ①       ; i++)
    {                  // 若当前被字符不是空格，而前一字符是空格，则表示有新单词出现
        if (str[i] ___②___ && str[i-1] ___③___ )
        {
            num++;
        }
    }
    printf("num=%d\n", num);
    return 0;
}
```

〖参考答案〗

① != '\0'

② != ' '

③ ==' '

（4）下面函数的功能是删除字符串 s 中所出现的与变量 c 相同的字符。

```
void  Squeeze(char s[], char c)
{
    int  i, j;
    for (i=j=0; ___①___ ; i++)
    {
```

```
            if (s[i] != c)
            {
                 ___②___ ;
                 j++;
             }
        }
        s[j] = '\0';
    }
```

〖参考答案〗

①　s[i] != '\0'

②　s[j] = s[i]

（5）下面函数 MrStrcmp()用于实现函数 strcmp()的功能，将两个字符串 s 和 t 进行比较，然后将两个字符串中第一个不相同字符的 ASCII 码值之差作为函数值返回。

```
    int MyStrcmp(char s[], char t[])
    {
        int  i;
        for (i=0; s[i] == t[i]; i++ )
        {
            if (s[i] == ___①___  )
                return 0 ;
        }
        return ( ___②___ );
    }
```

〖参考答案〗

①　'\0'

②　s[i] - t[i]

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>
    int main(void)
    {
        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++;                       // 非负数个数计算
            }
        }
        printf("sum = %d, counter = %d\n", sum, counter);
        return 0;
    }
```

程序的运行结果如下：

```
Input 20 Numbers:
1   2     3      4     5      6     7      8     -9     -10↙
12  -45   56     -99   34     87    90     -23   0      65↙
sum = 380, 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
void  MaxMinExchang(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;
}
int main(void)
{
    int  a[ARR_SIZE], i, n;
    printf("Input n(n<=10):");
    scanf("%d", &n);
    printf("Input %d Numbers:\n", n);
    for (i=0; i<n; i++)
    {
        scanf("%d", &a[i]);
    }
    MaxMinExchang(a, n);
    printf("After MaxMinExchange:\n");
    for (i=0; i<n; i++)
    {
        printf("%4d", a[i]);
```

```
    }
    printf("\n") ;
    return 0;
}
```

程序的运行结果如下：

```
Input n(n<=10):10↙
Input 10 Numbers:
12  45  23  0  -22  46  67  33  99  11 ↙
After MaxMin Exchange:
12  45  23  0  99  46  67  33  -22  11
```

6.6　输入 5×5 阶的矩阵，编程计算：（1）两条对角线上的各元素之和。（2）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 10
int main(void)
{
    int  a[ARR_SIZE][ARR_SIZE], i, j, n, sum = 0;
    long  product = 1;
    printf("Input n:");
    scanf("%d", &n) ;
    printf("Input %d*%d matrix:\n", n, n);
    for (i=0; i<n; i++)
    {
        for (j=0; j<n; j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    for (i=0; i<n; i++)
    {
        for (j=0; j<n; j++)
        {
            if (i == j || i+j == n-1)
                sum += a[i][j];
            if ((i == j || i+j == n-1) && i%2 == 0 && j%2 == 0)
                product *= a[i][j];
        }
    }
    printf("sum = %d\nproduct = %ld\n", sum, product);
    return 0;
}
```

程序的运行结果如下：

```
Input n:5↙
Input 5*5 matrix:
```

```
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 1〗

```
#include<stdio.h>
#define  ARR_SIZE  11
void  YHTriangle(int a[][ARR_SIZE], int  n);
void  PrintYHTriangle(int a[][ARR_SIZE], int  n);
int main(void)
{
    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);
    return 0;
}
// 函数功能：计算 n 行杨辉三角形中各元素数值
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 n 行杨辉三角形
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");
    }
}
```

〖参考答案 2〗

参考答案 1 中的函数 YHTriangle()还可以用以下方法编写。

```
void YHTriangle(int a[][ARR_SIZE], int n)
{
    int  i, j;
    for (i=1; i<=n; i++)
     {
        for (j=1; j<=i; j++)
        {
            if (j==1 || i==j)
            {
                 a[i][j] = 1;
            }
            else
            {
                a[i][j] = a[i-1][j-1] + a[i-1][j];
            }
        }
    }
}
```

程序的运行结果如下：

```
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

〖参考答案〗 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将移动后的元素仍放在数组 v 中。

```
#include<stdio.h>
#define  ROW 2
#define  COL 3
int main(void)
{
    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");
    }
    return 0;
}
```

程序的运行结果如下：

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

6.9　用二维数组作为函数参数，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,…,m; j=1,2,…,n）。

〖参考答案〗

```
#include<stdio.h>
#define      ROW      2
#define      COL      3
// 函数功能：输入矩阵元素，存于数组 a 中
void  InputMatrix(int a[ROW][COL])
{
    int  i , j ;
    for (i=0; i<ROW; i++)
```

```
    {
        for (j=0; j<COL; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }
}
// 函数功能：计算矩阵 a 与 b 之和，即计算矩阵 a、b 对应元素之和，结果存于数组 c 中
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];
        }
    }
}
// 函数功能：输出矩阵 a 中的元素
void  PrintMatrix(int a[ROW][COL])
{
    int  i , j ;
    for (i=0; i<ROW; i++)
    {
        for (j=0; j<COL; j++)
        {
            printf("%6d", a[i][j]);
        }
        printf("\n");
    }
}
int main(void)
{
    int  a[ROW][COL], b[ROW][COL], c[ROW][COL];
    printf("Input 2*3 matrix a:\n");
    InputMatrix(a);
    printf("Input 2*3 matrix b:\n");
    InputMatrix(b);
    AddMatrix(a, b, c);
    printf("Results:\n");
    PrintMatrix(c);
    return 0;
}
```

程序的运行结果如下：

```
Input 2*3 matrix a:
1  2  3 ↙
```

```
4  5  6
Input 2*3 matrix 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，…，m；j=1，2，…，n），b_{ij} 为 $n\times m$ 阶矩阵 ***B*** 的元素（i=1，2，…，n；j=1，2，…，m），c_{ij} 为 $m\times m$ 阶矩阵 ***C*** 的元素（i=1，2，…，m；j=1，2，…，m）。

〖参考答案〗 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 的双重循环中，设置 k 的循环，进行累加求和运算 c[i][j]=c[i][j]+term，累加项为 term=a[i][j]*b[i][j]。

```
#include<stdio.h>
#define      ROW      2
#define      COL      3
// 函数功能：计算矩阵 a 与 b 之积，结果存于数组 c 中
void  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;                    // 一定要在这里将 c[i][j]初始化为 0 值
            for (k=0; k<COL; k++)
            {
                c[i][j] = c[i][j] + a[i][k] * b[k][j];
            }
        }
    }
}
// 函数功能：输出矩阵 a 中的元素
void  PrintMatrix(int a[ROW][ROW])
{
    int  i , j ;
    for (i=0; i<ROW; i++)
    {
        for (j=0; j<ROW; j++)
        {
            printf("%6d", a[i][j]);
        }
        printf("\n");
    }
}
int main(void)
```

```
{
    int  a[ROW][COL], b[COL][ROW], c[ROW][ROW], i, j;
    printf("Input 2*3 matrix a:\n");
    for (i=0; i<ROW ;i++)
    {
        for (j=0; j<COL; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }
    printf("Input 3*2 matrix 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");
    PrintMatrix(c);
    return 0;
}
```

程序的运行结果如下：

```
Input 2*3 matrix a:
1  2  3 ↙
4  5  6 ↙
Input 3*2 matrix 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、空格和其他字符的个数。

〖参考答案〗

```
#include <stdio.h>
#include <string.h>
#define      ARR_SIZE     80
int main(void)
{
    char str[ARR_SIZE];
    int  i, letter = 0, digit = 0, space = 0, others = 0;
    printf("Please input a  string:");
    gets(str);
    for (i=0; str[i]!='\0'; i++)
    {
```

```
        if (str[i]>='a' && str[i]<='z' || str[i]>='A' && str[i]<='Z')
            letter ++;                          // 统计英文字符个数
        else if (str[i] >= '0' && str[i] <= '9' )
            digit ++;                           // 统计数字字符个数
        else if (str[i] == ' ' )
            space ++;                           // 统计空格数
        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);
    return 0;
}
```

程序的运行结果如下：

```
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 1〗

利用两个数组实现字符串的逆序存放。用数组 a 存放逆序存放前的数组元素，用数组 b 存放逆序存放后的数组元素。

```
#include <stdio.h>
#include <string.h>
#define      ARR_SIZE     80
void Inverse(char str[], char ptr[]);
int main(void)
{
    char a[ARR_SIZE], b[ARR_SIZE];
    printf("Please enter a string: ");
    gets(a);
    Inverse(a, b);
    printf("The inversed string is: ");
    puts(b);
    return 0;
}
// 函数功能：实现将字符数组 str 中的字符串逆序存放，结果存于数组 ptr 中
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';
    }
```

〖参考答案 2〗

利用一个数组实现字符串的逆序存放。借助于一个中间变量 temp，将数组中首尾对称位置的元素互换。i 指向数组首部的元素，从 0 依次加 1 变化；j 指向数组尾部的元素，从 n−1 依次减 1 变化，变化到 i>j 时，结束元素互换操作。

```
#include<string.h>
#include<stdio.h>
#define      ARR_SIZE     80
void Inverse(char str[]);
int main(void)
{
    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
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;
    }
}
```

程序的运行结果如下：

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

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

〖参考答案〗 用 i 和 j 分别作为字符数组 srcStr 和字符数组 dstStr 的下标，先将 i 和 j 同时初始化为 0，然后移动 i 使其位于字符 dstStr 的尾部，即字符串结束标志处，再将字符数组 srcStr 中的字符依次复制到字符数组 dstStr 中。

```
#include <stdio.h>
#include <string.h>
#define      ARR_SIZE     80
void MyStrcat(char dstStr[], char srcStr[]);
int main(void)
{
    char  s[ARR_SIZE], t[ARR_SIZE];
    printf("Please enter source string: ");
    gets(s);
    printf("Please enter destination string: ");
    gets(t);
    MyStrcat(t,s);
    printf("The concatenate string is: ");
    puts(t);
    return 0;
}
// 函数功能：将字符串 srcStr 连接到字符串 dstStr 后面
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];
    }
}
```

函数 MyStrcat()的第 2 种实现方法如下。

```
void MyStrcat(char dstStr[], char srcStr[])
{
    int  i = 0, j;                     // 数组下标初始化为 0
    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';              // 在字符串 dstStr 的末尾添加一个字符串结束标志
}
```

程序的运行结果如下：

```
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。

（1）下列对字符串的定义中，错误的是_____。

A）char　str[7] = "FORTRAN";

B）char　str[] = "FORTRAN";

C）char　*str = "FORTRAN";

D）char　str[] = {'F','O','R','T','R','A','N',0};

（2）以下程序段的输出结果是_____。

```
char  a[] =  "ABCDE" ;
char  *p = NULL;
for (p=a; p<a+5; p++)
{
    printf("%s\n", p);
}
```

A）ABCDE　　B）A　　C）E　　D）ABCDE

B　　D　　BCDE

C　　C　　CDE

D　　B　　DE

E　　A　　E

（3）下列程序是对字符串的相关操作，正确的是_____。

A）

```
#include <stdio.h>
int main(void)
{
    static char  a[5];
       a = "abcde";
    printf("%s\n", a);
    return 0;
}
```

B）

```
#include <stdio.h>
int main(void)
{
    static char  a[7] = "goodbye!";
    printf("%s\n", a);
     return 0;
}
```

C）

```
#include <stdio.h>
int main(void)
{
    char  a[5] = "abcde";
    printf("%s\n", a);
```

```
    return 0;
}
```

D）

```
#include <stdio.h>
int main(void)
{
    static  char a[] = "abcde";
    printf("%s\n", a);
    return 0;
}
```

（4）下列函数的功能为_____。

```
void  Exchange(int *p1, int *p2)
{
    int  p;
    p = *p1;
    *p1 = *p2;
    *p2 = p;
}
```

A）交换*p1 和*p2 的值　　B）正确，但无法改变*p1 和*p2 的值

C）交换*p1 和*p2 的地址　　D）可能造成系统故障

（5）设有语句“int array[3][4];”，则在下面几种引用下标为 i 和 j 的数组元素的方法中，不正确的引用方式是_____。

A）array[i][j]　　B）*(*(array + i) + j)

C）*(array[i] + j)　　D）*(array + i*4 + j)

（6）声明语句 int (*p)();的含义是_____。

A）p 是一个指向一维数组的指针变量

B）p 是指针变量，指向一个整型数据

C）p 是一个指向函数的指针，该函数的返回值是一个整型

D）以上都不对

（7）声明语句“int　*f();”中的 f 的含义是_____。

A）一个用于指向整型数据的指针变量　　B）一个用于指向一维数组的行指针

C）一个用于指向函数的指针变量　　D）一个返回值为指针类型的函数名

（8）有 int *p[10];以下说法错误的是_____。

A）p 是数组名　　B）p 是一个指针数组

C）p 中每个元素都是一个指针变量　　D）p++操作是合法的

〖参考答案〗

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

7.2　阅读程序，写出运行结果。

（1）

```
#include <stdio.h>
int main(void)
{
    static int x[]  =  {1,2,3};
```

```
    int  s = 1, i, *p = x;
    for (i=0; i<3; i++)
    {
        s*= *(p + i);
    }
    printf("%d\n", s);
    return 0;
}
```

（2）

```
#include <stdio.h>
int main(void)
{
    int  a[] = {1,2,3,4,5};
    int  *p = a;
    printf("%d, ", *p);
    printf("%d, ", *(++p));
    printf("%d, ", *++p);
    printf("%d, ", *(p--));
    printf("%d, ", *p++);
    printf("%d, ", *p);
    printf("%d, ", ++(*p));
    printf("%d\n", *p);
    return 0;
}
```

（3）

```
#include  <stdio.h>
int main(void)
{
    int  i = 0;
    char  b[] = "program";
    char  *a = "PROGRAM";
    printf("%c%s\n", *a, b + 1);
    while (putchar(*(a + i)))
    {
        i++;
    }
    printf("i = %d\n", i);
    while (--i)
    {
         putchar (*(b + i));
    }
    printf("\n%s\n", &b[3]);
    return 0;
}
```

（4）当命令行参数为“demo.exe This is a program”时，下面程序（已知程序文件名为 demo.c，假设 demo.c 及其可执行程序 demo.exe 位于目录 E:\demo 下）的运行结果是什么？

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

```
int main(int argc, char *argv[])
{
    int  i;
    for (i=0; i<argc; i++)
    {
        printf("%s\n", argv[i]);
    }
    return 0;
}
```

〖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。

（1）下面函数实现函数 strlen()的功能，即计算指针 p 所指向的字符串中的实际字符个数。

```
unsigned int  MyStrlen(char *p)
{
    unsigned int  len;
    len = 0;
    for (; *p!= ___①___ ; p++)
    {
        len ___②___ ;
    }
    return ___③___ ;
}
```

〖参考答案〗

①　'\0'

②　++

③　len

（2）下面函数也实现函数 strlen()的功能，但计算方法与（1）有所不同。

```
unsigned int  MyStrlen(char s[])
{
    char  *p = s;
    while (*p!= ___①___ )                    // 移动指针 p 使其指向字符串结束标志
```

```
    {
        p++;
    }
    return ___②___ ;        // 指针 p 与字符串首地址间的差值即为字符串的实际字符个数
}
```

〖参考答案〗

① '\0'

② (p-s)

（3）下面函数实现函数 strcmp()的功能，即比较两个字符串的大小，将两个字符串中第一个出现的不相同字符的 ASCII 码值之差作为比较的结果返回。返回值大于 0，表示第一个字符串大于第二个字符串；返回值小于 0，表示第一个字符串小于第二个字符串；当两个字符串完全一样时，返回值为 0。

```
int  MyStrcmp(char *p1, char *p2)
{
    for (; *p1 == *p2; p1++,p2++)
    {
        if (*p1 == '\0')
            return ___①___ ;
    }
    return ___②___ ;
}
```

〖参考答案〗

① 0

② (*p1-*p2)

（4）下面的函数用于计算两个整数之和，并通过指针形参 z 得到 x 和 y 相加后的结果。

```
void Add(int x,int y, ___①___ z)
{
    ___②___ = x+y;
}
```

〖参考答案〗

① int *

② *z

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。

```
#include <stdio.h>
void  Print(char *arr[], int len);
int main(void)
{
    char  *pArray[] = {"Fred","Barrey","Wilma","Betty"};
    int  num = sizeof(pArray) / sizeof(char);
    printf("Total string numbers = %d\n", num);
    Print(pArray, num);
    return 0;
}
void  Print(char *arr[], int len)
```

```
{
    int  i;
    for (i=0; i<len; i++)
    {
        printf("%s,  ", arr[i]);
    }
    printf("\n");
}
```

程序希望得到的运行结果如下：

```
Total string numbers = 4
Fred, Barrey, Wilma, Betty
```

〖参考答案〗

程序第 5 行语句计算指针数组的大小的方法是错误的。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)。程序第 5 行语句应修改为：

```
int  num = sizeof(pArray) / sizeof(char*);
```

此外，为了不在所有字符串的最后输出逗号，第 13 行语句应修改为：

```
for (i=0; i<len-1; i++)
```

第 17 行语句应修改为;

```
printf("%s\n", arr[len-1]);
```

7.5　参考例 7.2，用指针变量作函数参数实现两数交换函数，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);
void Exchange(int a[], int b[], int n);
void InputArray(int a[],int n);
void PrintArray(int a[],int n);
int main(void)
{
    int  a[ARRAY_SIZE], b[ARRAY_SIZE], n;
    printf("Input array lenth n<=10: ");
    scanf("%d", &n);
    printf("Input array a:\n");
    InputArray(a, n);
    printf("Input array b:\n");
    InputArray(b, n);
    Exchange(a,b,n);
    printf("After swap:\n");
    printf("Array a:\n");
    PrintArray(a, n);
    printf("Array b:\n");
    PrintArray(b, n);
    return 0;
}
void Swap(int *x, int *y)                    // 交换两个整型数据
```

```
{
    int  temp;
    temp = *x;
    *x = *y;
    *y = temp;
}
void Exchange(int a[], int b[], int n)          // 交换两个数组元素的值
{
    int i;
    for (i=0; i<n; i++)
    {
        Swap(&a[i], &b[i]);                     // 交换两个整型数据
    }
}
void InputArray(int a[], int n)                 // 输入数组元素
{
    int i;
    for (i=0; i<n; i++)
    {
        scanf("%d", &a[i]);
    }
}
void PrintArray(int a[], int n)                 // 输出数组元素
{
    int i;
    for (i = 0;i < n;i++)
    {
        printf("%d ", a[i]);
    }
    printf("\n");
}
```

程序的运行结果如下：

```
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　参考例 7.3，从键盘任意输入 10 个整数，用指针变量作函数参数编程计算最大值和最小值，并返回它们所在数组中的位置。

〖参考答案 1〗

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

```
int FindMax(int num[], int n, int *pMaxPos);
int FindMin(int num[], int n, int *pMinPos);
int main(void)
{
    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);        // 找最大值及其所在下标位置
    minValue = FindMin(num, 10, &minPos);        // 找最小值及其所在下标位置
    printf("Max=%d, Position=%d, Min=%d, Position=%d\n",
                                maxValue, maxPos, minValue, minPos);
    return 0;
}
// 函数功能：求有 n 个元素的整型数组 num 中的最大值及其所在下标位置，函数返回最大值
int FindMax(int num[], int n, int *pMaxPos)
{
    int  i, max;
    max = num[0];                              // 假设 num[0]为最大值
    *pMaxPos = 0;                              // 假设最大值在数组中的下标位置为 0
    for (i=1; i<n; i++)
    {
        if (num[i] > max)
        {
            max = num[i];
            *pMaxPos = i;                      // pMaxPos 指向最大值数组元素的下标位置
        }
    }
    return max ;
}
// 函数功能：求有 n 个元素的整型数组 num 中的最小值及其所在下标位置，函数返回最小值
int FindMin(int num[], int n, int *pMinPos)
{
    int  i, min;
    min = num[0];                              // 假设 num[0]为最小
    *pMinPos = 0;                              // 假设最小值在数组中的下标位置为 0
    for (i=1; i<10; i++)
    {
        if (num[i] < min)
        {
            min = num[i];
            *pMinPos = i;                // pMinPos 指向最小值数组元素的下标位置
        }
    }
    return min;
}
```

〖参考答案 2〗

```
#include <stdio.h>
void FindMax(int num[], int n, int *pMax, int *pMaxPos);
void FindMin(int num[], int n, int *pMin, int *pMinPos);
int main(void)
{
    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
    }
    FindMax(num, 10, &maxValue, &maxPos);        // 找最大值及其所在下标位置
    FindMin(num, 10, &minValue, &minPos);        // 找最小值及其所在下标位置
    printf("Max=%d, Position=%d, Min=%d, Position=%d\n",
                                maxValue, maxPos, minValue, minPos);
    return 0;
}
// 函数功能：求有 n 个元素的整型数组 num 中的最大值及其所在下标位置
void FindMax(int num[], int n, int *pMax, int *pMaxPos)
{
    int i;
    *pMax = num[0];                              // 假设 num[0]为最大
    *pMaxPos = 0;                                // 假设最大值在数组中的下标位置为 0
    for (i=1; i<n; i++)
    {
        if (num[i] > *pMax)
        {
            *pMax = num[i];                      // pMax 指向最大值数组元素
            *pMaxPos = i;                        // pMaxPos 指向最大值数组元素的下标位置
        }
    }
}
// 函数功能：求有 n 个元素的整型数组 num 中的最小值及其所在下标位置
void FindMin(int num[], int n, int *pMin, int *pMinPos)
{
    int  i;
    *pMin = num[0];                              // 假设 num[0]为最小
    *pMinPos = 0;                                // 假设最小值在数组中的下标位置为 0
    for (i=1; i<10; i++)
    {
        if (num[i] < *pMin)
        {
            *pMin = num[i];                      // pMin 指向最小值数组元素
            *pMinPos = i;                        // pMinPos 指向最小值数组元素的下标位置
        }
    }
}
```

程序的运行结果如下：

```
Input 10 numbers:
1 2 3 45 67 8 9 12 7 8↙
Max=67, Position=4, Min=1, Position=0
```

7.7　参考例 7.5 和习题 6.13，不使用函数 strcat()，用字符指针变量作函数参数编程实现字符串连接函数 strcat()的功能，将字符串 srcStr 连接到字符串 dstStr 的尾部。

〖参考答案〗

```
#include <stdio.h>
int main(void)
{
    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);
    printf("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
    return 0;
}
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
}
```

程序的运行结果如下：

```
Please enter the source string:abcd↙
Please enter the other string:efgh↙
The concat is:
abcdefgh
```

7.8　从键盘输入一个字符串，编程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 1〗

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

```
void Inverse(char *pStr);
int main(void)
{
    char  str[80];
    printf("Input a string:\n");
    gets(str);                          // 输入字符串
    Inverse(str);                       // 将存于 str 数组中的字符串逆序存放
    printf("The inversed string is:\n");
    puts(str);                          // 输出字符串
    return 0;
}
// 函数功能： 实现字符串逆序存放
void Inverse(char *pStr)
{
    int  len;
    char temp;
    char *pStart;                       // 指针变量 pStart 指向字符串的第一个字符
    char *pEnd;                         // 指针变量 pEnd 指向字符串的最后一个字符
    len = strlen(pStr);                 // 求出字符串长度
    for (pStart=pStr,pEnd=pStr+len-1; pStart<pEnd; pStart++,pEnd--)
    {
        temp = *pStart;
        *pStart = *pEnd;
        *pEnd = temp;
    }
}
```

〖参考答案 2〗

参考答案 1 程序中的函数 Inverse()还可以用如下方法编写。

```
void Inverse(char *pStr)
{
    int  len = 0;
    char  temp;
    char  *pStart = pStr;               // 指针变量 pStart 指向字符串的第一个字符
    char  *pEnd;                        // 指针变量 pEnd 指向字符串的最后一个字符
    for (; *pStart!='\0'; pStart++)
        len++;                          // 求出字符串长度
    for (pStart=pStr,pEnd=pStr+len-1; pStart<pEnd; pStart++,pEnd--)
    {
        temp = *pStart;
        *pStart = *pEnd;
        *pEnd = temp;
    }
}
```

程序的运行结果如下：

```
Input a string:
abcdef↙
```

```
The inversed string is:
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"和"ABCCBA"都是回文。

【算法思想】 由题意可知，回文就是一个对称的字符串，利用这一特点可采用如下算法：

（1）设置两个指针 pStart 和 pEnd，让 pStart 指向字符串首部，让 pEnd 指向字符串尾部。

（2）利用循环从字符串两边对指针所指字符进行比较，当对应的两字符相等且两指针未超越对方时，使指针 pStart 向前移动一个字符位置（加 1），使指针 pEnd 向后移动一个字符位置（减 1），一旦发现两字符不等或两指针已互相超越（不可能是回文），则立即停止循环。

（3）根据退出循环时两指针的位置，判断字符串是否为回文。

〖参考答案〗

```
#include <stdio.h>
#include <string.h>
int main(void)
{
    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");
    }
    return 0;
}
```

程序的两次测试结果如下：

```
① Input string:abccba↙
  Yes!
② Input string:abccbd↙
  No!
```

*7.10　参考例 7.9，用指针变量作函数参数编程计算任意 $m \times n$ 阶矩阵的转置矩阵。

〖参考答案 1〗

用指向一维数组的指针变量即二维数组的行指针作函数参数。

```
#include <stdio.h>
#define      ROW      3
#define      COL      4
void Transpose(int (*a)[COL], int (*at)[ROW], int row, int col);
void InputMatrix(int (*s)[COL], int row, int col);
void PrintMatrix(int (*s)[ROW], int row, int col);
int main(void)
{
    int  s[ROW][COL];                  // s 代表原矩阵
    int  st[COL][ROW];                 // st 代表转置后的矩阵
    printf("Please enter matrix:\n");
    InputMatrix(s, ROW, COL);          // 输入原矩阵，s 指向矩阵 s 的第 0 行，行指针
    Transpose(s, st, ROW, COL);        // 对矩阵 s 进行转置，结果存放于 st 中
    printf("The transposed matrix is:\n");
    PrintMatrix(st, COL, ROW);         // 输出转置矩阵，*st 指向 st 的第 0 行，行指针
    return 0;
}
// 函数功能：对任意 row 行 col 列的矩阵 a 转置，转置后的矩阵为 at
void Transpose(int (*a)[COL], int (*at)[ROW], int row, int col)
{
    int  i, j;
    for (i=0; i<row; i++)
    {
        for (j=0; j<col; j++)
        {
            *(*(at+j)+i) = *(*(a+i)+j);
        }
    }
}
void InputMatrix(int (*s)[COL], int row, int col)          // 输入矩阵元素
{
    int i, j;
    for (i=0; i<row; i++)
    {
        for (j=0; j<col; j++)
        {
            scanf("%d", *(s+i)+j);              // 这里*(s+i)+j 等价于&s[i][j]
        }
    }
}
void PrintMatrix(int (*s)[ROW], int row, int col)          // 输入矩阵元素
{
    int  i, j;
    for (i=0; i<row; i++)
    {
        for (j=0; j<col; j++)
```

```
        {
            printf("%d\t", *(*(s+i)+j)); //这里*(*(s+i)+j)等价于 s[i][j]
        }
        printf(" \n");
    }
}
```

〖参考答案 2〗

用指向整型数组元素的指针变量即二维数组的列指针作函数参数。

```
#include <stdio.h>
#define      ROW      3
#define      COL      4
void Transpose(int *a, int *at, int row, int col);
void InputMatrix(int *s, int row, int col);
void PrintMatrix(int *s, int row, int col);
int main(void)
{
    int  s[ROW][COL];                       // s 代表原矩阵
    int  st[COL][ROW];                      // st 代表转置后的矩阵
    printf("Please enter matrix:\n");
    InputMatrix(*s, ROW, COL);       // 输入原矩阵，*s 指向矩阵 s 的 0 行 0 列，列指针
    Transpose(*s, *st, ROW, COL);           // 对矩阵 s 进行转置，结果存放于 st 中
    printf("The transposed matrix is:\n");
    PrintMatrix(*st, COL, ROW);     // 输出转置矩阵，*st 指向 st 的 0 行 0 列，列指针
    return 0;
}
// 函数功能：对任意 row 行 col 列的矩阵 a 转置，转置后的矩阵为 at
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);
        }
    }
}
void InputMatrix(int *s, int row, int col)     // 输入矩阵元素
{
    int i, j;
    for (i=0; i<row; i++)
    {
        for (j=0; j<col; j++)
        {
            scanf("%d", s+i*col+j);             // 这里 s+i*col+j 等价于&s[i][j]
        }
```

```
    }
}
void PrintMatrix(int *s, int row, int col)     // 输入矩阵元素
{
    int  i, j;
    for (i=0; i<row; i++)
    {
        for (j=0; j<col; j++)
        {
            printf("%d\t",*(s+i*col+j));        // 这里*(s+i*col+j)等价于s[i][j]
        }
        printf(" \n");
    }
}
```

程序的运行结果如下：

```
Please enter 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〗 定义一个指针数组存放月份的英文表示，然后输出相应月份。

```
#include <stdio.h>
int main(void)
{
    int  n;
    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
    }
    return 0;
}
```

程序的三次测试结果如下：

```
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
```

7.12　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，编程输出得到 3 种不同颜色的球的所有可能取法。

〖参考答案〗 用三重循环模拟取球过程，但每次取出的球如果与前面的球颜色相同就抛弃。

```
#include <stdio.h>
int main(void)
{
    char  *bColor[] = {"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, bColor[i], bColor[j], bColor[k]);
            }
        }
    }
    return 0;
}
```

程序的运行结果如下：

```
1:RED,YELLOW,BLUE
2:RED,YELLOW,WHITE
3:RED,YELLOW,BLACK
4:RED,BLUE,WHITE
5:RED,BLUE,BLACK
6:RED,WHITE,BLACK
7:YELLOW,BLUE,WHITE
8:YELLOW,BLUE,BLACK
9:YELLOW,WHITE,BLACK
10:BLUE,WHITE,BLACK
```

1.8　习题 8 及参考答案

8.1　选择题。

（1）已知有如下共用体变量的定义，则 sizeof(test) 的值是_____。

```
union
```

```
{
    int  i;
    char  c;
    float  a;
} test;
```

A）4　　B）5　　C）6　　D）7

（2）若有以下说明，则_____的叙述是正确的（已知 int 占 2 字节）。

```
struct st
{
    int  a;
    int  b[2];
} a;
```

A）结构体变量 a 与结构体成员 a 同名，定义是非法的

B）程序只在执行到该定义时才为结构体 st 分配存储单元

C）程序运行时为结构体变量 a 分配 6 字节存储单元

D）类型名 struct st 可以通过 extern 关键字提前引用

（3）若有以下结构体定义，则选择_____赋值是正确的。

```
struct s
{
    int  x;
    int  y;
}vs;
```

A）s.x = 10　　B）s.vs.x = 10

C）struct s va; va.x = 10　　D）struct s va = {10};

（4）已知学生记录描述为

```
struct student
{
    int  no;
    char  name[20];
    char  sex;
    struct
    {
        int  year;
        int  month;
        int  day;
     }birth;
} s;
```

设变量 s 中的“生日”是 1984 年 11 月 11 日，下列对“生日”的正确赋值方式是_____。

A）year = 1984;month = 11;day = 11;

B）birth.year = 1984;birth.month = 11;birth.day = 11;

C）s.year = 1984;s.month = 11; s.day = 11;

D）s.birth.year = 1984;s.birth.month = 11;s.birth.day = 11;

（5）已知有如下结构体定义，且有 p=&data，则对 data 中的成员 a 的正确引用是_____。

```
struct sk
{
```

```
    int a;
    float b;
}data, *p;
```

A）(*p).data　　B）(*p).a　　C）p->data.a　　D）p.data.a

（6）下面语句中引用形式非法的是_____。

```
struct student
{
    int num;
    int    age;
} stu[3]={{1001,20},{1002,19},{1003,21}};
struct student *p = stu;
```

A）(p++)->num　　B）p++　　C）(*p).num　　D）p = &stu.age

〖参考答案〗

（1）A　　（2）C　　（3）D　　（4）D　　（5）B　　（6）D

8.2　填空题。

已知有如下定义，则表达式 ++p->x 的值为__①__，表达式(++p)->x 的值为__②__。

```
struct
{
    int  x;
    int  y;
} s[2] = {{1,2},{3,4}},*p=s;
```

〖参考答案〗

①　2

②　3

8.3　阅读程序，选择程序运行结果。

（1）程序如下：

```
#include "stdio.h"
struct s
{
    int  x;
    int  y;
} cnum[2] = {1,3,2,7};
int main(void)
{
    printf("%d\n", cnum[0].y * cnum[1].x);
    return 0;
}
```

A）0　　B）1　　C）3　　D）6

（2）运行下列程序后，选择全局变量 t.x 和 t.s 的正确输出结果。

```
#include "stdio.h"
struct tree
{
    int  x;
    char  s[20];
}t;
```

```
void Func(struct tree t)
{
    t.x = 10;
    strcpy(t.s, "computer");
}
int main(void)
{
    t.x = 1;
    strcpy(t.s, "Microcomputer");
    Func(t);
    printf("%d,%s\n", t.x, t.s);
    return 0;
}
```

A）10,computer　　B）1,Microcomputer

C）1,computer　　D）10,Microcomputer

〖参考答案〗

（1）D　　（2）B

8.4　判断下列对结构体的声明是否正确。如果不正确，写出正确方法。

```
struct  STUDENT
{
    char  name[10];
    int  age;
}
STUDENT  student;
student->age = 20;
union val
{
    char  w;
    float  x;
    int  m;
}v = {1.2};
```

〖参考答案〗

正确的写法应为：

```
struct  STUDENT
{
    char  name[10];
    int  age;
};
struct  STUDENT  student;
student.age = 20;
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，它可以理解为一个筒结构，先放进筒中的数据被后放进筒中的数据“压住”，只有在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“压入堆栈”，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”，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。试着用链表的方法实现堆栈结构。

【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。即每次“压入堆栈”意味着在链表的尾部增加一个节点；而每次“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：压入堆栈，STACK *PushStack(int num)；弹出堆栈，int PopStack(void)。

“压入堆栈”包括两个操作过程：① 建立一个新节点；② 将要保存的数据放到新的节点的数据区。注意，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一致。

“弹出堆栈”包括三个操作过程：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;                                    // 堆栈节点数计数器
// 函数功能：生成一个新节点，用整型变量 num 的值为其赋初值，函数返回指向新节点的指针
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，输出错误信息并退出程序
    {
        printf("No enough memory!\n");
        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：将整型变量 num 的值压入堆栈，函数返回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++;                   // 堆栈节点数计数器加 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++;                   // 堆栈节点数计数器加 1
    }
    return pr;
}
// 函数功能：弹出堆栈，函数返回弹出堆栈的位于栈顶的数据
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--;               // 堆栈节点数计数器减 1
        }
    }
    pr->next = NULL;                    // 将最后一个节点的前一个节点置成末尾节点
    result = p->data;
    free(p);                            // 释放被弹出堆栈的节点数据占用的内存
    return result;
}
int main(void)
{
    int  pushNum[5] = {111,222,333,444,555 };
    int  popNum[5];
    int  i;
    for (i=0; i<5; i++)
```

```
    {
        PushStack(pushNum[i]);
        printf("Push %dth Data :  %d\n", i+1, pushNum[i]);
    }
    for (i=0; i<5; i++)
    {
        popNum[i] = PopStack();
        printf("Pop %dth Data :  %d\n", 5-i, popNum[i]);
    }
     return 0;
}
```

程序的运行结果如下：

```
Push 1th Data :  111
Push 2th Data :  222
Push 3th Data :  333
Push 4th Data :  444
Push 5th Data :  555
Pop 5th Data :  555
Pop 4th Data :  444
Pop 3th Data :  333
Pop 2th Data :  222
Pop 1th Data :  111
```

1.9 习题 9 及参考答案

9.1 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。

〖参考答案〗 在没有参数时提示语法错误，有参数时就把所有的参数当做文件名输出。

```
#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;
    }
    for (i=1; i<argc; i++)                  // 所有参数逐一输出
    {
        if ( type(argv[i]) == 0 )
            perror(argv[i]);
    }
```

```
    return 0;
}
// 函数功能：把文件 filename 输出到 stdout，函数返回为非 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;
}
```

9.2　用基本文件操作编写与例 9.4 有同样文件复制功能的程序。

〖参考答案〗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。只要修改函数 CopyFile()即可，这也体现了函数的优越性。

```
// 函数功能：把 srcName 文件内容复制到 dstName，函数返回值为非 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;
    fhSrc = open(srcName, O_RDONLY | O_BINARY);         // 打开文件
    if (fhSrc == -1)
        goto ERROR;
    fhDst = open(dstName, O_WRONLY | O_CREAT | O_TRUNC | O_BINARY);
    if (fhDst == -1)
        goto ERROR;
    while ((rtn = read(fhSrc, buf, BUF_SIZE)) > 0)      // 复制文件
    {
        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;
    }
```

9.3　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;
    }
    for (i=1; i<argc; i++)                       // 逐一判断所有参数
    {
        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，函数返回非 0 值表示成功，否
//           则失败
int MakeTypeTable(void)
{
    FILE*  fp = NULL;
    int  rval = 1, i;
    fp = fopen(CONFIG_FILENAME, "r");
    if (fp == NULL)
        goto ERROR;
    if (fscanf(fp, "%d\n", &typeCount) != 1)    // 读类型数及其结尾的'\n'
        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));
    for (i=0; i<typeCount; i++)                     // 读表
    {
        char*  pRtn;
        pRtn = fgets(typeTable[i].character, CHARARCTER_LEN, fp);    // 读特征
        if (pRtn == NULL)
            goto ERROR;
        TrimNewLineChar(typeTable[i].character);
        pRtn = fgets(typeTable[i].name, NAME_LEN, fp);           // 读类型名
        if (pRtn == NULL)
            goto ERROR;
        TrimNewLineChar(typeTable[i].name);
    }
    rval = 1;                                       // 成功
    goto EXIT;                                      // 转出口处理
ERROR:    rval = 0;
    FreeTypeTable();
EXIT:    if (fp != NULL)    fclose(fp);
    return rval;
}
// 函数功能：判断打开文件名 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
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
void FreeTypeTable(void)
{
    if (typeTable != NULL)
        free(typeTable);
    typeTable = NULL;
}
// 函数功能：去掉字符串 str 最后的'\n'
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文件 truetype.ini 来扩充数据，支持更多类型的判断，而程序不需要修改。