

3. 编程题

(1) 编写一程序要求任意输入 4 位十六进制整数，以反序的方式输出该十六进制数。例如：输入 9AF0，则输出 0FA9。

【设计思想】 对输入的 4 位十六进制数，分别取其最低、次低、次高和最高的一位十六进制数，具体方法可以将该数分别与 0X000F、0X00F0、0X0F00 和 0XF000 进行按位与的运算，然后分别进行左移 12 位、左移 4 位、右移 4 位和右移 12 位，再把所得到的每一步的值进行求和即可得到该十六进制数的反序结果。例如， $(0X9AF0 \& 0X000F) \ll 12$ 得到 0X0000， $(0X9AF0 \& 0X00F0) \ll 4$ 得到 0X0F00， $(0X9AF0 \& 0X0F00) \gg 4$ 得到 0X00A0， $(0X9AF0 \& 0XF000) \gg 12$ 得到 0X0009，再把得到的结果进行求和，即 $0X0000 + 0X0F00 + 0X00A0 + 0X0009$ 就得到 0X0FA9。

【参考答案】

```
#include <stdio.h>
void main ( )
{
    unsigned short a, b;

    scanf ("%4X", &a);           // 输入 4 位十六进制数给变量 a
    b = (a & 0x000F) << 12;     // 取 a 的最低一位十六进制数并左移 12 位后赋值给 b
    b += (a & 0x00F0) << 4;      // 取 a 的次低一位十六进制数并左移 4 位后与 b 相加再赋值给 b
    b += (a & 0X0F00) >> 4;      // 取 a 的次高一位十六进制数并右移 4 位后与 b 相加再赋值给 b
    b += (a & 0XF000) >> 12;     // 取 a 的最高一位十六进制数并左移 12 位后与 b 相加再赋值给 b
    printf ("%4X\n", b);         // 输出变换后的十六进制数
}
```

程序运行结果（假设输入为 9AF0）：

(4) 编程计算 $a+aa+aaa+\dots+a\dots a$ (n 个 a) 的值, n 和 a 的值由键盘输入。

【设计思想】用累加和算法, 累加项为 $\text{term}=\text{term}*10+a$; $i=1,2,\dots,n$; term 初值为 0。

【参考答案】

```
#include <stdio.h>

void main ()
{
    long term = 0, sum = 0;
    int a, i, n;
    void main () { }

    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 += term;                  // 进行累加
    }
    printf ("sum = %ld\n", sum);
}
```

程序运行结果 (假如 a 和 n 的值输入为 2,4):

```
sum = 2468
```

(5) 利用 $\frac{\pi}{2} \approx \frac{2}{1} \times \frac{2}{3} \times \frac{4}{3} \times \frac{4}{5} \times \frac{6}{5} \times \frac{6}{7} \times \dots$ 前 100 项之积计算 π 的值。

【设计思想】采用累乘积算法, 累乘项为 $\text{term}=n*n/((n-1)*(n+1))$; $n=2,4,\dots,100$;

步长为 2。

【参考答案】

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

程序运行结果：

```
result = 3.126079
```

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

【设计思想】 x 由键盘输入, 采用累加和算法 $sum = sum + term$, sum 的初值为 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, $n=n+2$ 。

【参考答案】

```
#include <stdio.h>
#include <math.h>
void main ( )
{
    int n = 1, count = 0;
    float x;
    double sum, term;
    printf ("Input x: ");
    scanf ("%f", &x);
    sum = x;
    term = x; // 赋初始值
    do
    {
        term = -term * x * x / ((n+1) * (n+2)); // 计算相应项, 并改相应符号
        sum += term; // 累加
        n += 2;
        count++;
    } while (fabs(term) >= 1e-5); // 条件表达式
    printf ("Result is: %f\n", sum);
}
```

```

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
sin(x) = 0.141120, count = 8

```

(7) 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 $153 = 1^3 + 3^3 + 5^3$ 。

【设计思想】 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 $i^3 + j^3 + k^3$ 即可知道 n 是否是水仙花数。

【参考答案】

```

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

```

程序运行结果:

```

result is: 153 370 371 407

```

(8) 从键盘上任意输入一个整数 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 1、2、3、4 四个数字，然后计算 $1+2+3+4=10$ ，并输出 10。

【设计思想】 对输入的整数取绝对值，即可实现忽略整数前的正负号。

【参考答案】

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

void main ( )
{
    int i1, i2, i3, i4, k, n;
    printf ("Input data is: ");
    scanf ("%d", &n);
    k = abs (n);
    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);
}
```

程序运行结果：

```
Input data is: 1234
The sum of the total bit is 10
```

(9) 从键盘上输入任意正整数，编程判断该数是否为回文数。所谓回文数就是从左到右读这个数与从右到左读这个数是一样的。例如，12321、4004都是回文数。

【设计思想】 将该整数按照从最低位到最高位进行分离，然后重新组合成一整数，再将该整数与原来的整数比较，如果相等，则为回文数；否则不是。

【参考答案】

```
#include <stdio.h>

void main ( )
{
    int n, m = 0, s, r;
    printf ("Input data is: ");
    scanf ("%d", &n);
    s = n;
    while (s != 0)
    {
        r = s % 10; //从低位到高位逐一分离
        m = 10 * m + r; //重新组合一整数
        s = s / 10; //求其商
    }
    if (m == n)
        printf ("yes\n");
}
```

```

    else
        printf ("no\n");
}

```

程序运行结果:

```

Input data is: 12321✓
yes

```

- (10) 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？

【设计思想】 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 只需满足第二个方程即可打印，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>

void 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 ("%02d, %02d, %02d\n", x, y, z);
                if (count % 6 == 0)
                    printf ("\n");
            }
        }
    printf ("count = %d\n", count);
}

```

程序运行结果:

```

01, 46, 53  02, 42, 56  03, 38, 59  04, 34, 62  05, 30, 65  06, 26, 68
07, 22, 71  08, 18, 74  09, 14, 77  10, 10, 80  11, 06, 83  12, 02, 86
count = 12

```

- (11) 某学校有四位同学中的一位做了好事，不留名，表扬信来了之后，校长问这四位是谁做的好事。四个人的回答是：

A 说：不是我。

B 说：是 C。

C 说：是 D。

D 说：他胡说。

已知三个人说的是真话，一个人说的是假话。现在问做好事者到底是谁？

【设计思想】 将 A、B、C、D 四个人的回答用一条件表达式来表示，对于 A 的回答：
thisman !='A'; 对于 B 的回答：thisman =='C'; 对于 C 的回答：thisman =='D'; 对于 D 的回答：thisman !='D'; 然后，采用枚举的方法，一个人一个人的去试，如果这四个条件表达式中有三个为真，即四个条件表达式的值相加为 3，则可判定是其中某人做的好事。

【参考答案】

```
#include <stdio.h>

void main ()
{
    int k = 0, sum = 0, g = 0;
    char thisman = ' ';
    for (k = 0; k <= 3; k++)
    {
        thisman = 'A' + k;
        sum = (thisman != 'A') + (thisman == 'C') + (thisman == 'D') + (thisman != 'D');
        if (sum == 3)
        {
            printf ("This man is %c\n", thisman);
            g = 1;
        }
        printf ("Can't found\n");
    }
}
```

程序运行结果：

```
This man is C
```

sum = 76

(2) 输入 10 个整数，将这 10 个整数按升序排列输出，并且奇数在前，偶数在后。比如，如果输入的 10 个数是 10 9 8 7 6 5 4 3 2 1，则输出 1 3 5 7 9 2 4 6 8 10。

【设计思想】 将输入的 10 个整数，按其奇偶性分别放在数组 a 的左部和右部。其具体方法就是设置两个整型变量 odd 和 even，分别表示奇数和偶数存放在数组 a 中元素的下标，odd 的初始值为 0，每存放一个奇数 odd 增 1，even 的初始值为 9，每存放一个偶数 even 减 1，然后通过选择排序的方法分别对数组 a 左边的奇数和右边的偶数进行排序。

【参考答案】

```
#include <stdio.h>

void main ()
{
    int i, j, odd, even, n, t, a[10];
    odd = 0;
    even = 9;
    for (i = 0; i < 10; i++) // 将键盘输入的数据存入数组中
    {
        scanf ("%d", &n);
        if (n % 2 != 0) // 将奇数放置在数组的左边
            a[i] = n;
        else
            a[i] = n;
    }
    for (i = 0; i < 10; i++)
    {
        for (j = i + 1; j < 10; j++)
        {
            if (a[i] > a[j])
            {
                t = a[i];
                a[i] = a[j];
                a[j] = t;
            }
        }
    }
    for (i = 0; i < 10; i++)
        printf ("%d", a[i]);
}
```

```

【参考】 a[odd++] = n;
else
    a[even--] = n;           //将偶数放置在数组的右边
}

void main ()
{
    int i, j, t, a[10];
    n = i;
    for (j = i+1; j < odd; j++)
        if (a[j] < a[n])
            n = j;
    if (n != i)
    {
        t = a[i];
        a[i] = a[n];
        a[n] = t;
    }
    for (i = odd; i < 9; i++)      //通过选择排序对偶数进行升序排列
    {
        n = i;
        for (j = i+1; j < 10; j++)
            if (a[j] < a[n])
                n = j;
        if (n != i)
        {
            t = a[i];
            a[i] = a[n];
            a[n] = t;
        }
    }
    for (i = 0; i < 10; i++)      //输出排序结果
        printf ("%d ", a[i]);
    printf ("\n");
}

```

程序运行结果 (假设输入为 10 9 8 7 6 5 4 3 2 1):

1	3	5	7	9	2	4	6	8	10
---	---	---	---	---	---	---	---	---	----

再输出调

```

printf ("Input 6 integer number: ");
for (i = 0; i < 6; i++)
    scanf ("%d", &a[i]);
printf ("\n");
for (i = 0; i < 6; i++)
{
    for (j = 0; j < 6; j++)
        printf ("%d ", a[j]);
    printf ("\n");
    t = a[5];
    for (j = 5; j > 0; j--)
        a[j] = a[j-1];
    a[0] = t;
}

```

程序运行结果（假设输入为 5 7 4 8 9 1）：

5	7	4	8	9	1
1	5	7	4	8	9
9	1	5	7	4	8
8	9	1	5	7	4
4	8	9	1	5	7
7	4	8	9	1	5

S	B	P	V	S	Q
Y	F	T	C	E	Q
Y	T	O	S	E	R
Y	E	S	B	B	S
L	E	S	B	S	S

(5) 输入 5×5 阶的矩阵，编程实现：

- 求两条对角线上的各元素之和；
- 求两条对角线上行、列下标均为偶数的各元素之积。

【设计思想】 通过两重 for 循环将键盘输入的 5×5 的矩阵元素存放在二维数组 a 中，采用累加和算法： $sum = sum + a[i][i]$ ($i=0,1,2,3,4,5$)，累计左对角线上的元素之和，同时用 $sum = sum + a[i][4-i]$ ($i=0,1,3,4,5$)，累计右对角线上的元素之和（除去两对角线上的交叉元素 $a[2][2]$ ）。采用累乘算法： $mul = mul * a[i][i]$ ($i=0,2,4$)，累乘左对角线上行、列下标均为偶数的元素之积，同时用 $mul = mul * a[i][4-i]$ ($i=0,4$)，累乘右对角线上行、列下标均为偶数的元素之积（除去两对角线上的交叉元素 $a[2][2]$ ）。

【参考答案】

```

#include <stdio.h>

void main ( )

```

```

{  

    int i, j, sum = 0, mul = 1, a[5][5];  

    printf ("Input 5*5 array:\n");      //输入 5×5 的矩阵  

    for (i = 0; i < 5; i++)  

        for (j = 0; j < 5; j++)  

            scanf ("%d", &a[i][j]);  

    int i, max, min, a[10];  

    printf ("\n");  

    for (i = 0; i < 5; i++)  

    {  

        sum += a[i][i];                //对左对角线元素进行累加  

        if (i != 2)                    //对右对角线上的元素进行累加(对角线中间元素除外)  

            sum += a[i][4-i];  

        if (i % 2 != 0)                //如果行下标为奇数,进入下一次循环  

            continue;  

        mul *= a[i][i];               //对左对角线上行、列下标均为偶数的元素进行累乘  

        if (i != 2)                  //对右对角线上行、列下标均为偶数的元素进行累乘(中间元素除外)  

            mul *= a[i][4-i];  

    }  

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

}

```

程序运行结果:

```

Input 5*5 array:  

7 2 7 4 8 ↵  

9 3 5 7 4 ↵  

8 3 5 6 7 ↵  

4 8 5 3 5 ↵  

2 4 8 9 1 ↵  

sum = 44    mul = 560

```

(6) 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

```

【设计思想】 对于有 6 行的杨辉三角形, 可以用一个 6 行 6 列的二维数组 $a[6][6]$ 来表示。对于第 i 行的元素: $a[i][0]=1$, $a[i][i]=1$ ($i=0,1,2,\dots,5$), $a[i][j]=a[i-1][j-1]+a[i-1][j]$ ($j=1,2,\dots,i-1$)。然后显示二维数组中计算的结果, 其中第 1 行显示 1 个数据, 第 2 行显示 2 个数据, ..., 第 6 行显示 6 个数据。

【参考答案】该程序，将字符数组 s2 中的全部字符复制到字符数组 s1 中，不用 strcpy。如果此时复制的字符串为“”，则循环结束，输出字符串 s1。

```
#include <stdio.h>
#define N 6
void main ( )
{
    int i, j, a[N][N];
    for (i = 0; i < N; i++)
    {
        a[i][0] = 1;
        a[i][i] = 1;
        for (j = 1; j < i; j++)
            a[i][j] = a[i-1][j-1] + a[i-1][j];
    }

    for (i = 0; i < N; i++) //显示结果
    {
        for (j = 0; j < N-i-1; j++)
            printf ("  ");
        for (j = 0; j <= i; j++)
            printf ("%2d ", a[i][j]);
        printf ("\n");
    }
}
```

程序运行结果：

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

程序运行结果 (数据输入为 1 3 5 7 9 auto <input> 4 <input> 10 <input> 0 <input> 20 <input>)

(11) 有一个已排好序(升序)的整型数组, 要求从键盘输入一整数按原来排序的规律将它插入数组中。并输出结果。比如, 原来数据为 1 3 5 7, 需插入 4, 插入后为 1 3 4 5 7。

【设计思想】 假设插入前数组元素的个数为 num, 首先要找到在整型数组中要插入的位置, 其具体方法是: 通过循环将插入的整数 n 与数组中的元素 a[i] ($i = num - 1, \dots, 0$) 从右到左进行逐一比较, 如果此时数组元素 $a[i] > n$, 则插入的位置在元素 $a[i]$ 之前的某一位位置, 将 $a[i]$ 往右移一个位置, 循环继续; 如果 $a[i] \leq n$, 则插入的位置刚好就在 $a[i]$ 之后, 即 $a[i+1]$ 的位置, 循环结束, 执行 $a[i+1] = n$ 就实现了插入的任务。

【参考答案】

```
#include <stdio.h>

void main ( )
{
    int a[10] = {1, 3, 5, 7, 9}, num = 5, i;
    printf ("Before insert: "); // 显示插入之前数组元素的值
    for (i = 0; i < num; i++)
        printf ("%d ", a[i]);
    printf ("\n");
    printf ("Input a number: "); // 输入一插入的整数 n
    scanf ("%d", &n);
    for (i = num-1; i >= 0; i--) // 从右到左将与 n 进行比较
        if (a[i] > n) // 数组元素比 n 大
            a[i+1] = a[i]; // 将该数组元素往右移一个位置
        else // 否则, 退出循环, 要插入的位置就是第 i 个元素之后
            break;
    a[i+1] = n;
    printf ("After insert: "); // 显示插入后数组元素的值
    for (i = 0; i < num+1; i++)
        printf ("%d ", a[i]);
    printf ("\n");
}
```

程序设计实验指导 (第二版) 第 3 章 整型数据

程序运行结果:

```
Before insert: 1 3 5 7 9 auto <input> 4 <input> 10 <input> 0 <input> 20 <input> )
Input a number: 6</pre>
```

```
After insert: 1 3 5 6 7 9
```

后数组 a 中的元素为 5 4 3 2 1。所以正确答案为 A。

3. 编程题

(1) 设计一个函数, 用来判断一个整数是否为素数。

【设计思想】 按照素数的定义, 可以用 $2 \sim \text{number}-1$ 的所有整数去除 number , 只要能整除的, 便说明 number 不是素数。不过对任意值的 number 检验它是否为素数时, 不必使用比其平方根大的整数去整除它。

【参考答案】

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

int IsPrimeNumber (int number);

void main ( )
{
    int a;

    printf ("Input a integer number: ");
    scanf ("%d", &a);
}
```

```
if (IsPrimeNumber(a)) { // 判断 a 是否为素数
    printf ("%d is prime number.\n", a);
} else {
    printf ("%d isn't prime number.\n", a);
}

void GetData (int a)
{
    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);
}
```

程序运行结果 (假设输入的整数为 5):

```
5 is prime number
```

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

【设计思想】 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 $a - b$ 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】

```
#include <stdio.h>

int MaxCommonFactor (int a, int b);

void main ( )
{
    int a, b, c;

    printf ("Input two integer number: ");
    scanf ("%d%d", &a, &b);
    c = MaxCommonFactor (a, b);
    if (c != -1)
        printf ("The biggest common factor of %d and %d is %d\n", a, b, c);
    else
        printf ("The biggest common factor of %d and %d isn't exist\n", a, b);
}
```

C 语言程序设计教程 (第二版) 习题解答与实验指导

8

```
// 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);
}
```

程序运行结果 (假设输入的整数为 8 20):

```
The biggest common factor of 8 and 20 is 4
```

(3) 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。

【设计思想】 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：

- 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。
- 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 向后移动一个字符位置（即加 1），使指针 pEnd 向前移动一个字符位置（即减 1），一旦发现对应的两个字符不等或两指针已相互超越（不可能是回文），则立即停止循环。
- 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()
{
    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");
}
```

程序运行结果：

```
Input string: abcba
Yes!
```

(6) 编写一个函数 fun, 它的功能是: 删除字符串中的数字字符。例如输入字符串 48CTYP9E6, 则输出 CTYPE。

【设计思想】 要删除字符串 s 中的数字字符, 可预先设置一字符指针变量 pstr, pstr 首先指向 s 的第一个字符, 然后通过循环使用 pstr 来遍历该字符串的每个字符, 如果 pstr 所指字符是数字字符, 则利用 strcpy 函数将 pstr+1 所指向的后续字符串复制到 pstr 所指向的字符串, 也就是将原来的数字字符覆盖掉, 直到 pstr 所指向的字符是非数字字符, 接着 pstr 加 1, 即右移一个字符位置, 重复上述过程, 直到 pstr 所指字符为字符串结尾符'\0'为止。

【参考答案】

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

void func (char *s);
```

C 语言程序设计教程 (第二版) 习题解答与实验指导

```
void main ()
{
    char str[80];

    printf ("Input string: ");
    gets (str);
    func (str);
    printf ("After delete digital char: %s\n", str);
}
```

```
//删除字符串 s 中的数字字符
void func (char *s)
{
    char *pstr;

    for (pstr = s; *pstr != '\0'; pstr++)
    {
        while (*pstr >= '0' && *pstr <= '9')
            strcpy (pstr, pstr+1);
    }
}
```

程序运行结果:

```
Input string: 48CTYP9E6
After delete digital char: CTYPE
```

(5) 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。

【设计思想】 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>

void 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++)
                printf ("%d: %s, %s, %s\n", m, ballcolor[i], ballcolor[j], ballcolor[k]);
    m++;
}
```

什指针。

3. 编程题

- (1) 编写一程序，其功能是显示指定的文本文件，在显示文件内容时同时加上行号

【设计思想】 命令行的第二个参数 argv[1] 为待显示的文件名，显示前以读方式打开，然后通过 fgets 函数从文件中读出一行信息到临时字符串 string 中，再利用 printf 函数显示该行内容（通过变量 i 来显示行号），直到文件结束，即读出的字符串指针为 NULL 为止。

【参考答案】

```

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

#define MAX 256

void main (int argc, char *argv[ ])
{
    FILE *fp;
    char string[MAX];
    int i;

    if(argc != 2)
    {
        printf("the number of arguments not correct\n");
        printf("\n Usage: 可执行文件名 dispfilename");
        exit(0);
    }

    if ((fp = fopen(argv[1], "r")) == NULL) //打开显示文件失败
    {
        printf("can not open file\n");
        exit(0);
    }

    //显示文件内容
    i = 1;
    while (fgets(string, MAX, fp) != NULL)
        printf ("%d %s", i++, string);

    fclose(fp);
}

```

