1.编程打印形如图所示的n\*n方阵。

1 2 3 4 5 6 7

24 25 26 27 28 29 8

23 40 41 42 43 30 9

22 39 48 49 44 31 10

21 38 47 46 45 32 11

20 37 36 35 34 33 12

19 18 17 16 15 14 13

**程序如下：**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int i, j, k, s, n;

int a[10][10];

printf("请输入矩阵的阶n=\n");

scanf("%d", &n);

k = 1, s = 0, i = 0;

while (k <= n\*n){

for (j = s; j<n - s; j++){

a[i][j] = k++;

}

j--;

for (i = s + 1; i<n - s; i++){

a[i][j] = k++;

}

i--;

for (j = n - s - 2; j >= s; j--){

a[i][j] = k++;

}

j++;

for (i = n - s - 2; i >= s + 1; i--){

a[i][j] = k++;

}

i++;

s++;

}

for (i = 0; i<n; i++){

for (j = 0; j<n; j++){

printf("%4d", a[i][j]);

}

printf("\n");

}

system("pause");

return 0;

}

2.写出计算ackermann函数ack(m,n)的递归计算函数。对于m>=0,n>=0，ack(m,n)定义为：

ack(0,n)=n+1;

ack(m,0)=ack(m-1,1);

ack(m,n)=ack(m-1,ack(m,n-1));

**递归计算函数：**

#include <stdio.h>

#include<stdio.h>

int Ack(int m, int n);

int main()

{

int m, n;

scanf("%d %d", &m, &n);

printf("%d\n", Ack(m, n));

system("pause");

return 0;

}

int Ack(int m, int n)

{

if (m == 0)

return n + 1;

if (n == 0 && m > 0)

return Ack(m - 1, 1);

if (m > 0 && n > 0)

return Ack(m - 1, Ack(m, n - 1));

}

3.请编程求1\*2\*3\*4\*……\*n所得的数末尾有多少个0?(n由键盘输入，1000<n<10000)

**程序如下：**

#include <stdio.h>

#include<stdlib.h>

int main()

{

int n, N, sum = 0;

scanf("%d", &n);

N = n;

for (n; n>5; n = n / 5)

{

sum = sum + n / 5;

}

printf("%d!末尾0的个数是 :%d\n", N, sum);

system("pause");

return 0;

}

4. A,B,C,D,E5人为某次竞赛的前五名,他们在公布名次前猜名次。  
A说:B得第三名,C得第五名。  
B说:D得第二名,E得第四名。  
C说:B得第一名,E得第四名。  
D说:C得第一名,B得第二名。  
A说:D得第二名,A得第三名。  
结果每个人都猜对了一半,实际名次是什么呢?

**程序如下：**  
#include<stdio.h>

#include<stdlib.h>

int main()

{

int a, b, c, d, e;

for (a = 1; a <= 5; a++)

{

for (b = 1; b <= 5; b++)

{

if (a != b)

for (c = 1; c <= 5; c++)

{

if (c != a&&c != b)

for (d = 1; d <= 5; d++)

{

if (d != a&&d != b&&d != c)

{

e = 15 - a - b - c - d;

if ((b == 3) + (c == 5) == 1 && (d == 2) + (e == 4) == 1 && (b == 1) + (e == 4) == 1 && (c == 1) + (b == 2) == 1 && (d == 2) + (a == 3) == 1)

printf("A是第%d名 B是第%d名 C是第%d名 D是第%d名 E是第%d名\n", a, b, c, d, e);

}

}

}

}

}

system("pause");

return 0;

}

5.编写一个算法对输入的一个整数，判断它能否被4,7,9整除，并输出以下信息之一：

能同时被4,7,9，整除；

能被其中两个数（要指出哪两个）整除；

能被其中一个数（要指出哪一个）整除；

不能被4,7,9,任一个整除。

**程序如下：**

#include<stdio.h>

#include<stdlib.h>

void print(int num)

{

if (num % 4 == 0) printf("4 ");

if (num % 7 == 0) printf("7 ");

if (num % 9 == 0) printf("9 ");

}

int main()

{

int num;

int count = 0;

printf("请输入一个数：\n");

scanf("%d", &num);

if (num % 4 == 0)

{

count++;

}

if (num % 7 == 0)

{

count++;

}

if (num % 9 == 0)

{

count++;

}

switch (count)

{

case 0:

printf("%d不能被4，7，9任一个整数除\n", num);

break;

case 1:

printf("%d只能被其中一个数整除\n", num);

print(num);

printf("\n");

break;

case 2:

printf("%d能同时被其中的2个数整除\n", num);

print(num);

printf("\n");

break;

case 3:

printf("%d能同时被4，7，9整除\n", num);

break;

default:

exit(0);

break;

}

system("pause");

return 0;

}

6.求这样的两个数据：5位数=2\*4位数，9个数字互不相同。

**程序如下：**

#include<stdio.h>

#include<stdlib.h>

int judge(int n, int m){

int a[10] = { 0 };

int count = 0;

int i;

do{

a[n % 10] = 1;

n = n / 10;

a[m % 10] = 1;

m = m / 10;

} while (n || m); //以数组下标作为对应数字进行统计标记

for (i = 0; i <= 9; i++){

if (a[i] == 1) count++;

}

return count - 1;

}

int main()

{

int i, j;

int count;

for (i = 5000; i<9999; i++) //2需要乘上至少5000的数才能等于五位数

for (j = 10000; j<20000; j++){

count = judge(i, j);

if (count == 9 && 2 \* i == j)

printf("四位数为%d 五位数为%d\n", i, j);

}

system("pause");

return 0;

}

7.输入一个5位数以内的正整数，完成以下操作：

①判断它是一个几位数。

②请按序输出其各位数字。

③逆序输出其各位数字。

**程序如下：**

#include<stdlib.h>

#include<stdio.h>

int main()

{

int x, y, s[6], i = 0, j;

scanf("%d", &x);

if (x / 10000 != 0)

y = 5;

else if (x / 1000 != 0)

y = 4;

else if (x / 100 != 0)

y = 3;

else if (x / 10 != 0)

y = 2;

else y = 1;

printf("这是一个%d位数字\n", y);

while (y--)

{

s[i] = x % 10;

x = x / 10;

i++;

}

for (j = i - 1; j >= 0; j--) //正序输出

{

printf("%d ", s[j]);

}

printf("\n");

for (j = 0; j < i; j++) //逆序输出

{

printf("%d ", s[j]);

}

printf("\n");

system("pause");

return 0;

}