杨辉三角

/\*打印出以下的杨辉三角形 ，要求用户输入行数。

1:1

2:1 1

3:1 2 1

4:1 3 3 1

5:1 4 6 4 1

6:1 5 10 10 5 1

（注：输出数据之间用一个空格分隔，每行最后没有空格，行之间不插入空行）\*/

#include <stdio.h>

**int** main()

{

**int** i,j,n;

**int** a[20][20];

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

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

    {

        a[i][i]=1;           //对角线元素全是1

        a[i][0]=1;           //第一个元素也是1

    }

**for**(i=2;i<n;i++)           //从第三行开始处理

**for**(j=1;j<=i-1;j++)   //最后一列已经取了1

        a[i][j]=a[i-1][j-1]+a[i-1][j];

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

    {

**for**(j=0;j<=i;j++)

**printf**("%d%c",a[i][j],' ');

**printf**("\n");

    }

**return** 0;

}

方法2

#include <stdio.h>

**int** main()

{

**int** n, i, j, coefficient;

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

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

    {

        coefficient = 1;

**for** (j = 0; j <= i; j++)

        {

**printf**("%d ", coefficient);

            coefficient = coefficient \* (i - j) / (j + 1);

        }

**printf**("\n");

    }

**return** 0;

}

加密

/\*【题目描述】

输入4字母的小写英文单词和整数k，将单词中的每个字母用其后的第k个字母替换(字母表首尾相接)。

【输入输出样例】

输入：

like 4

输出:

pmoi \*/

#include <stdio.h>

**int** main()

{

**int** a1,a2,b1,b2,c1,c2,d1,d2,i;

**scanf**("%c%c%c%c%d",&a1,&b1,&c1,&d1,&i);

    i=i%26;    //避免下标越界，找到后移位数

**if**(i>('z'-a1)) a2=a1-(26-i);  //判断是否会超出字母表（超出就要重开），a1+i-26;

**else** a2=a1+i;

**if**(i>('z'-b1)) b2=b1-(26-i);

**else** b2=b1+i;

**if**(i>('z'-c1)) c2=c1-(26-i);

**else** c2=c1+i;

**if**(i>('z'-d1)) d2=d1-(26-i);

**else** d2=d1+i;

**printf**("%c%c%c%c",a2,b2,c2,d2);

**return** 0;

}

字符串转整数

/\*输入一个三位字符串形式的数，比如”123”、”976”，现在要将他们变为int型数字，然后判断它是否为素数。

【算法】

设sum = 0;

sum = 10 \* sum + p[i] - '0';\*/

#include <stdio.h>

**int** main()

{

**int** i,j,k,n;

**char** p[3];    //注意定义的是字符型

**int** sum=0;

**int** flag=0;   //使用标志判断

**for**(i=0;i<3;i++)

**scanf**("%c",&p[i]);

**for**(i=0;i<3;i++)

    sum=sum\*10+ p[i]- '0';   //记住这种转化方法

    n=sum;

**for**(j=2;j<sum;j++)

    {

        k=sum%j;

**if**(k==0) flag=1;

    }

**if**(flag==1) **printf**("%d %s",n,"No.");

**else** **printf**("%d %s",n,"Yes.");

**return** 0;

}

Getchar 和 strlen用法

/\*编写函数将一个字符按照字符顺序插入到一个有序字符数组中（数组长20, 输入的有序字符串长度<10）。函数原型：

void StrInsert(char a[], char c);

在主函数输入有序串以及一个字符，调用函数实现插入，最后在主函数输出改变后的串。\*/

/\*思路：扩展一维数组

（1）使数组大小比已有数组大1

（2）将待排序数存入扩展数组的最后一位

（3） 整体排序\*/

#include <stdio.h>

#include <string.h>

**void** StrInsert(**char** a[], **char** c)

{

**int** i,j,n,t;

    n=**strlen**(a);   //strlen只能以char做参数，且需以'\0'结尾

    a[n]=c;

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

    {

**for**(j=i+1;j<n+1;j++)

        {

**if**(a[i]>a[j])

            {

                t=a[i];

                a[i]=a[j];

                a[j]=t;

            }

        }

     }

}

**int** main()

{

**char** a[20],c;

**scanf**("%s",a);

**getchar**(); ////读取回车符

**scanf**("%c",&c);

    StrInsert(a,c);

**printf**("%s",a);

**return** 0;

}

窗体顶端

窗体底端

#include<stdio.h>

**char**\* shuchu(**char**\* A)

{

**char** B[100]={'\0'};

**int** i=0,j=0,word=0;

**while**(1)

    {

**if**(A[i]=='\0')

        {

**printf**("%s",B);

**return** B;

        }

**if**((A[i]<=57&&A[i]>=49)&&(word==0))

        {

            word=1;

        }

**if**((A[i]==45)&&(A[i+1]<=57&&A[i+1]>=48))

        {

            B[j]=A[i];

            j++;

        }

**if**(word==1)

        {

            B[j]=A[i];

            j++;

**if**(A[i+1]>57||A[i+1]<48)

            {

                B[j]=' ';

                j++;

                word=0;

            }

        }

        i++;

    }

}

**int** main()

{

**char** A[100]={'\0'};

**gets**(A);

**char**\* B=shuchu(A);

}

/\*题目内容：

编写两个函数分别实现整型数组从小到大排序、二分法在有序数组中查找（返回下标, 找不到时结果为-1）。函数原型：

void Sort(int \*p, int n);

int Search(int \*p, int n, int key); // n - 数组长度， key - 待查数字

在主函数输入若干正整数（不超过100个，输入-9999表示结束）以及5个待查找数字，调用排序和查找函数，输出查找结果。\*/

#include <stdio.h>

**void** Sort(**int** \*p,**int** n)     //p等价于数组

{

**int** i,j,t;

**for**(i=0;i<n-1;i++) //选择排序的内容，非常重要！

**for**(j=i+1;j<n;j++)

    {

**if**(\*(p+i)>\*(p+j))

        {

            t=\*(p+i);

            \*(p+i)=\*(p+j);

            \*(p+j)=t;

        }

    }

}

**int** Search(**int** \*p, **int** n, **int** key)

{

**int** flag=-1;

**int** start=0;

**int** end=n-1;

**int** mid;

**do**

    {

        mid=(start+end)/2;

**if**(key>\*(p+mid)) start=mid+1;  //注意这里mid要+1

**else** **if**(key<\*(p+mid)) end=mid-1; //else if！

**else**

        {

            flag=mid+1;

**break**;

        }

    }**while**(start<=end);

**return** flag;

}

**int** main()

{

**int** i,a[1000],n,x,key,\*p; //主函数要定义数组，然后才可通过指针操作

    p=a; //指针指向数组的方式

**for**(i=0;i<1000;i++)

    {

**scanf**("%d",&\*(p+i));

**if**(\*(p+i)==-9999)

        {

            n=i;

**break**;

        }

    }

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

    {

**scanf**("%d",&key);

        x=Search(p,n,key);

**printf**("%d",x);

**printf**("\n");

    }

**return** 0;

}

结构体（考点，要记得套路）

/\*定义表示平面点的结构体类型，成员有x坐标和y坐标，数据类型用float。

定义计算两个点的距离的函数，计算两个平面点的距离，返回float。

主函数中定义两个平面点的结构体变量，输入点的坐标，调用函数求距离，显示它们的距离(输出%f)。

输入：四个实数，前两个表示一个点的x,y坐标；后两个表示另一个点的x,y坐标.

输出：两个点的距离。

注意，距离函数的参数是两个结构体类型！！！\*/

#include <stdio.h>

#include <math.h>

**struct** points

    {

**float** x,y;

    }p1,p2;

**float** dist(**struct** points p1, **struct** points p2)

{

**float** d1, d2, dis;

d1=p2.x-p1.x;

d2=p2.y-p1.y;

dis=**sqrt**((d1\*d1)+(d2\*d2));

**return** dis;

}

**int** main()

{

**float** dis;

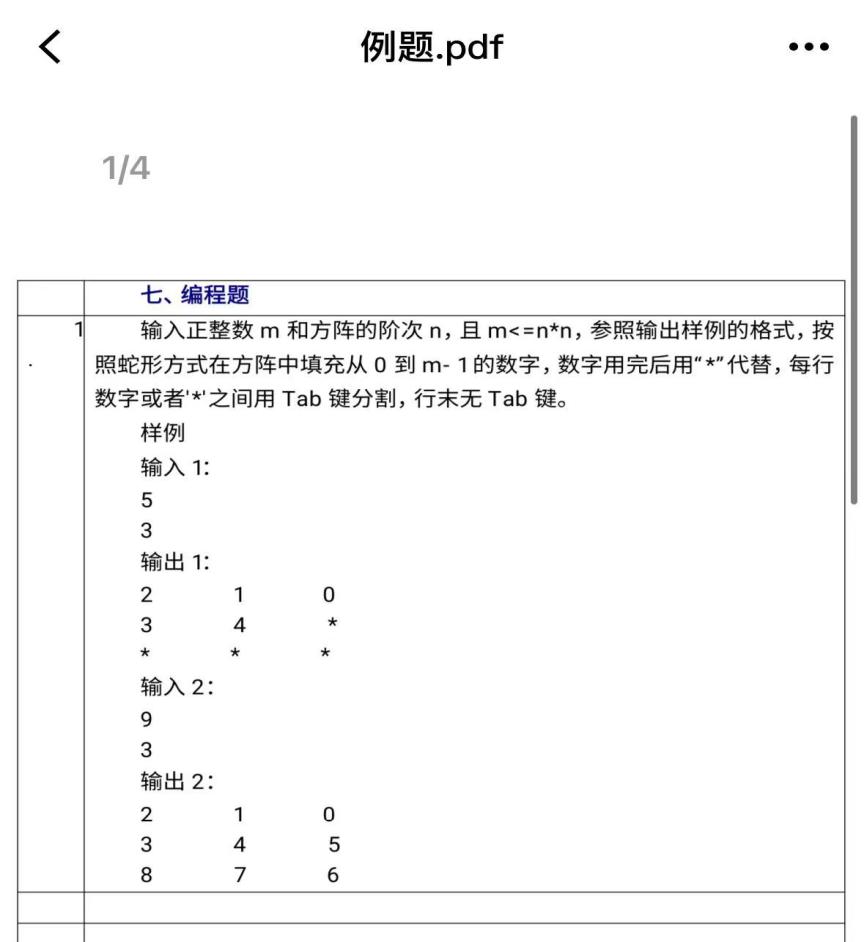
**scanf**("%f%f%f%f",&p1.x,&p1.y,&p2.x,&p2.y);

    dis=dist(p1,p2);

**printf**("%f",dis);

**return** 0;

}



/\*输入正整数m和方阵的阶次n,且m<=n\*n，参照输出样例的格式，按照蛇形方式在方阵中填充从0到m-1的数字

数字用完后用“\*”代替，每行数字或者\*之间用 Tab 键分割，行末无 Tab 键。\*/

#include <stdio.h>

int main()

{

int i=0,x=0;

int j,k,m,n,a[100][100];

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

//把要输出的东西先存到二维数组

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

{

if(j%2==0) //0和偶数行是逆时针排序

{

for(k=n-1;k>=0;k--)

{

if(i<m)

{

a[j][k]=i;

i++;

}

else break; //超出m-1就跳出循环

}

}

else //顺时针排序

{

for(k=0;k<n;k++)

{

if(i<m)

{

a[j][k]=i;

i++;

}

else break;

}

}

}

//下面是输出二维数组过程

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

{

for(k=0;k<n;k++)

{

if(x<m) //未超出m-1，按照二维数组输出

{

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

x++;

printf("\t");

}

else

{

printf("\*\t");

}

}

printf("\n"); //一行结束，换行

}

return 0;

}