**问题描述**

|  |  |
| --- | --- |
| 试题编号： | 201604-2 |
| 试题名称： | 俄罗斯方块 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　俄罗斯方块是俄罗斯人阿列克谢·帕基特诺夫发明的一款休闲游戏。 　　游戏在一个15行10列的方格图上进行，方格图上的每一个格子可能已经放置了方块，或者没有放置方块。每一轮，都会有一个新的由4个小方块组成的板块从方格图的上方落下，玩家可以操作板块左右移动放到合适的位置，当板块中某一个方块的下边缘与方格图上的方块上边缘重合或者达到下边界时，板块不再移动，如果此时方格图的某一行全放满了方块，则该行被消除并得分。 　　在这个问题中，你需要写一个程序来模拟板块下落，你不需要处理玩家的操作，也不需要处理消行和得分。 　　具体的，给定一个初始的方格图，以及一个板块的形状和它下落的初始位置，你要给出最终的方格图。  **输入格式**  　　输入的前15行包含初始的方格图，每行包含10个数字，相邻的数字用空格分隔。如果一个数字是0，表示对应的方格中没有方块，如果数字是1，则表示初始的时候有方块。输入保证前4行中的数字都是0。 　　输入的第16至第19行包含新加入的板块的形状，每行包含4个数字，组成了板块图案，同样0表示没方块，1表示有方块。输入保证板块的图案中正好包含4个方块，且4个方块是连在一起的（准确的说，4个方块是四连通的，即给定的板块是俄罗斯方块的标准板块）。 　　第20行包含一个1到7之间的整数，表示板块图案最左边开始的时候是在方格图的哪一列中。注意，这里的板块图案指的是16至19行所输入的板块图案，如果板块图案的最左边一列全是0，则它的左边和实际所表示的板块的左边是不一致的（见样例）  **输出格式**  　　输出15行，每行10个数字，相邻的数字之间用一个空格分隔，表示板块下落后的方格图。注意，你不需要处理最终的消行。  **样例输入**  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 1 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 0 3  **样例输出**  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0 |
| 代码程序： | #include<stdio.h>  bool isCT(int ELSFK[15][10],int pos,int hang,int MBK[4][4],int h);  int main()  {  int ELSFK[15][10];  int MBK[4][4];  for(int i =0;i<15;i++)  {  for(int j = 0; j<10;j++)  {  scanf("%d",&ELSFK[i][j]);  }  }  for(int i = 0; i<4;i++)  {  for(int j = 0; j<4;j++)  {  scanf("%d",&MBK[i][j]);  }  }  int pos;  scanf("%d",&pos);  // 模拟  int h= 4;  for(int i = 3; i>=0;i--)  {  int flag = true;  for(int j = 0;j<4;j++)  {  if(MBK[i][j] !=0)  {  flag = false;  break;  }  }  if(flag){  continue;  }else{  h = i+1;  break;  }  }  int i;  for(i = 0; i<=15-h;i++)  {  if(isCT(ELSFK,pos-1,i,MBK,h))  {  break;  }  }  int Zhang = i-1;  for(int i = Zhang;i<Zhang+h;i++)  {  for(int j = pos-1;j<pos-1+4;j++)  {  ELSFK[i][j] = ELSFK[i][j] + MBK[i-Zhang][j-pos+1];  }  }  for(int i = 0; i<15;i++)  {  for(int j = 0; j<10;j++)  {  printf("%d",ELSFK[i][j]);  if(j<10-1)  {  printf(" ");  }  }  printf("\n");  }  return 0;  }  bool isCT(int ELSFK[15][10],int pos,int hang,int MBK[4][4],int h)  {  int flag = false;  for(int i = hang;i<hang+h;i++)  {  for(int j = pos;j<pos+4;j++)  {  if(ELSFK[i][j] + MBK[i-hang][j-pos] == 2)  {  flag = true;  break;  }  }  }  return flag;  } |