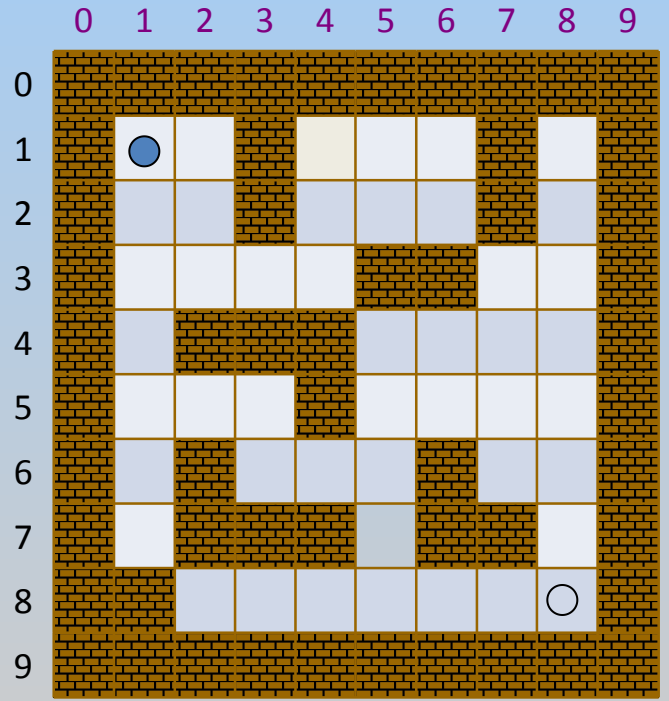
本节主题：

队列的应用-迷宫问题

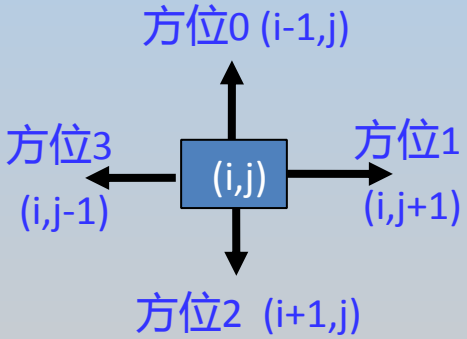
问题

- 迷宫问题：求出从入口到出口的路径。
- 设置一个数组mg表示迷宫，方块为0表示对应方块是通道，为1时表示对应方块为墙：

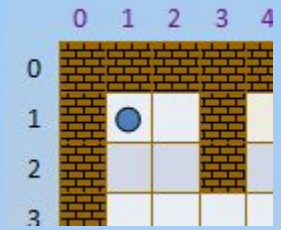
```
//M=8,N=8
int mg[M+2][N+2]={
{1,1,1,1,1,1,1,1,1,1},
{1,0,0,1,0,0,0,1,0,1},
{1,0,0,1,0,0,0,1,0,1},
{1,0,0,0,0,1,1,0,0,1},
{1,0,1,1,1,0,0,0,0,1},
{1,0,0,0,1,0,0,0,0,1},
{1,0,1,0,0,0,1,0,0,1},
{1,0,1,1,1,0,1,1,0,1},
{1,1,0,0,0,0,0,0,0,1},
{1,1,1,1,1,1,1,1,1,1}};
```



运行过程中，mg[i][j]=-1时，将表示对应方格不能再考察。



采用队列求解迷宫问题



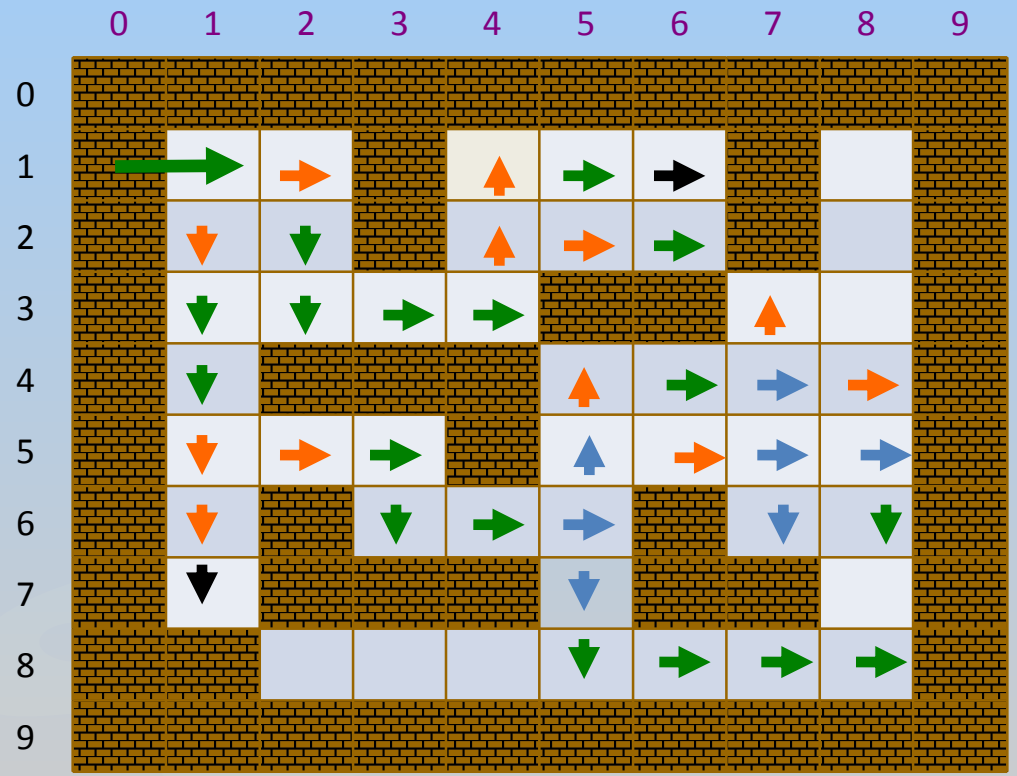

```
typedef struct
{
    int i,j; //方块的位置
    int pre; //本路径中上一方块在队列中的下标
} Box;   //方块类型
typedef struct
{
    Box data[MaxSize];
    int front,rear; //队头指针和队尾指针
} QuType;           //定义顺序队类型
QuType qu;
```

	i	j	pre
[0]	1	1	-1
[1]	1	2	0
[2]	2	1	0
[3]	2	2	1
[4]	3	1	2
[5]	3	2	3
[6]	4	1	4
[7]	3	3	5
[8]	5	1	6
[9]	3	4	7
[10]	5	2	8
[11]	6	1	8
[12]	2	4	9
[13]	5	3	10
[14]	7	1	11
[15]	1	4	12
[16]	2	5	12
[17]	6	3	13
[18]	1	5	15
[19]	2	6	16
[20]	6	4	17

	i	j	pre
[21]	1	6	18
[22]	6	5	20
[23]	5	5	22
[24]	7	5	22
[25]	4	5	23
[26]	5	6	23
[27]	8	5	24
[28]	4	6	25
[29]	5	7	26
[30]	8	6	27
[31]	8	4	27
[32]	4	7	28
[33]	5	8	29
[34]	6	7	29
[35]	8	7	30
[36]	8	3	31
[37]	3	7	32
[38]	4	8	32
[39]	6	8	33
[40]	8	8	35

求解过程

- 搜索
- 输出结果



	i	j	pre		i	j	pre
[0]	1	1	-1	[21]	1	6	18
[1]	1	2	0	[22]	6	5	20
[2]	2	1	0	[23]	5	5	22
[3]	2	2	1	[24]	7	5	22
[4]	3	1	2	[25]	4	5	23
[5]	3	2	3	[26]	5	6	23
[6]	4	1	4	[27]	8	5	24
[7]	3	3	5	[28]	4	6	25
[8]	5	1	6	[29]	5	7	26
[9]	3	4	7	[30]	8	6	27
[10]	5	2	8	[31]	8	4	27
[11]	6	1	8	[32]	4	7	28
[12]	2	4	9	[33]	5	8	29
[13]	5	3	10	[34]	6	7	29
[14]	7	1	11	[35]	8	7	30
[15]	1	4	12	[36]	8	3	31
[16]	2	5	12	[37]	3	7	32
[17]	6	3	13	[38]	4	8	32
[18]	1	5	15	[39]	6	8	33
[19]	2	6	16	[40]	8	8	35
[20]	6	4	17				

搜索从(1,1)到(M,N)的路径

（1）首先将(1,1)入队；

（2）在队列qu不为空时循环：出队一次（由于不是环形队列，该出队元素仍在队列中），称该出队的方块为当前方块，front为该方块在qu中的下标。

① 如果当前方块是出口，则输出路径并结束。

② 否则按顺时针方向找出当前方块的四个方位中可走的相邻方块（对应的mg数组值为0），将这些可走的相邻方块均插入到队列qu中，其pre设置为本搜索路径中上一方块在qu中的下标值，也就是当前方块的front值，并将相邻方块对应的mg数组值置为-1，以避免回过来重复搜索。

(3)若队列为空仍未找到出口，即不存在路径。

搜索算法

```
int mgpath(int xi,int yi,int xe,int ye)
{
   int i,j,find=0,di;
   QuType qu;
   qu.front=qu.rear=-1;
   qu.rear++;
   qu.data[qu.rear].i=xi;
   qu.data[qu.rear].j=yi;
   qu.data[qu.rear].pre=-1;
   mg[xi][yi]=-1;
   while (qu.front!=qu.rear && !find)
   {
      qu.front++;
      i=qu.data[qu.front].i;
      j=qu.data[qu.front].j;

      //若到达终点，输出结果并结束

      //将当前方块四周可以到达的方块加入队列
   }
   return(0);
}
```

```
if (i==xe && j==ye)
{
   find=1;
   print(qu,qu.front);
   return(1);
}
```

```
for (di=0; di<4; di++)
{
   switch(di)
   {
   case 0:
      i=qu.data[qu.front].i-1;  j=qu.data[qu.front].j;break;
   case 1:
      i=qu.data[qu.front].i; j=qu.data[qu.front].j+1; break;
   case 2:
      i=qu.data[qu.front].i+1;  j=qu.data[qu.front].j;  break;
   case 3:
      i=qu.data[qu.front].i, j=qu.data[qu.front].j-1;  break;
   }
   if (mg[i][j]==0)
   {
      qu.rear++;
      qu.data[qu.rear].i=i;
      qu.data[qu.rear].j=j;
      qu.data[qu.rear].pre=qu.front;
      mg[i][j]=-1;
   }
}
```

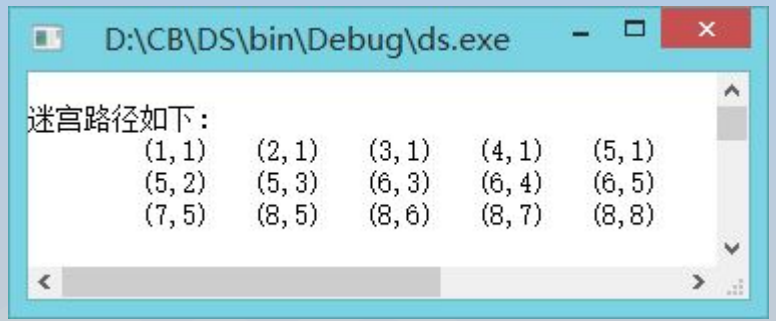
```
void print(QuType qu,int front)
{
   int k=front,j,ns=0;
   printf("\n");
   do  //反向找到最短路径,将该路径上的方块的pre成员设置成-1
   {
      j=k;
      k=qu.data[k].pre;
      qu.data[j].pre=-1;
   }
   while (k!=0);
   printf("迷宫路径如下:\n");
   k=0;
   while (k<=front)  //正向搜索到pre为-1的方块,即构成正向的路径
   {
      if (qu.data[k].pre==-1)
      {
         ns++;
         printf("\t(%d,%d)",qu.data[k].i,qu.data[k].j);
         if (ns%5==0)
            printf("\n");       //每输出每5个方块后换一行
      }
      k++;
   }
   printf("\n");
}
```

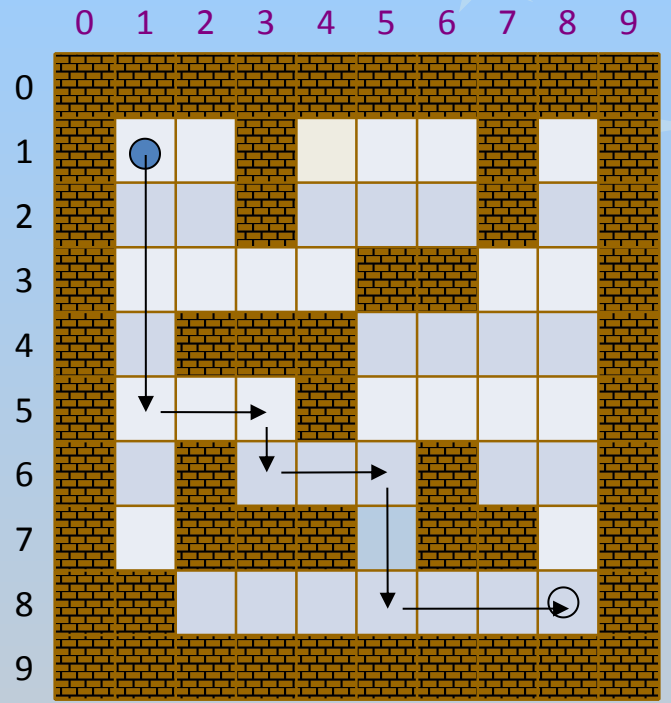
输出结果

	i	j	pre		i	j	pre
[0]	1	1	-1	[21]	1	6	18
[1]	1	2	0	[22]	6	5	20
[2]	2	1	0	[23]	5	5	22
[3]	2	2	1	[24]	7	5	22
[4]	3	1	2	[25]	4	5	23
[5]	3	2	3	[26]	5	6	23
[6]	4	1	4	[27]	8	5	24
[7]	3	3	5	[28]	4	6	25
[8]	5	1	6	[29]	5	7	26
[9]	3	4	7	[30]	8	6	27
[10]	5	2	8	[31]	8	4	27
[11]	6	1	8	[32]	4	7	28
[12]	2	4	9	[33]	5	8	29
[13]	5	3	10	[34]	6	7	29
[14]	7	1	11	[35]	8	7	30
[15]	1	4	12	[36]	8	3	31
[16]	2	5	12	[37]	3	7	32
[17]	6	3	13	[38]	4	8	32
[18]	1	5	15	[39]	6	8	33
[19]	2	6	16	[40]	8	8	35
[20]	6	4	17				

求解结果

```
int main()
{
    mgpath(1,1,M,N);
    return 0;
}
```





实施广度优先搜索，
得到最优解！

思考题

- 用队列和用栈求解迷宫问题有什么不同？