## 实验一 图搜索与问题求解

### 实验1 启发式搜索

**一 实验目的**

1 熟悉和掌握启发式搜索的定义、估价函数和算法过程；

2 理解和掌握启发式搜索过程，能够用选定的编程语言求解八数码问题，理解求解流程和搜索顺序；

3 比较并分析图搜索策略的实质，通过实验理解启发式搜索的意义。

**二 实验预习内容**

1了解重排九宫/八数码问题、一字棋游戏、八皇后问题；

2 各种图搜索算法及剪枝技术等。

**三 实验内容**

1 八数码问题

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 8 | 3 |  | 1 | 2 | 3 |
| 1 | 6 | 4 | 8 |  | 4 |
| 7 |  | 5 | 7 | 6 | 5 |

初始棋局 目标棋局

图 1 八数码问题示例

在一个3\*3的方格棋盘上放置8个标有1、2、3、4、5、6、7、8数字的将牌，留下一个空格（一般用0表示），规定与空格上下左右相邻的将牌可以移入空格。问题的解是要求寻找一条从某初始状态S0到目标状态Sg的将牌移动路线（如图1所示）

2 问题描述

要求用某种启发式搜索方法（推荐选用A\*算法）求解从给定的初始状态到目标状态的移动路线。

**四 实验要求**

1 自己定义启发式函数,能正确求解出从初始状态到目标状态的移动路线；

A\*算法是一种启发式搜索算法，用于在搜索图中寻找从初始节点到目标节点的最佳路径。在每一步，从所有叶节点中选择一个节点进行扩展。为了尽快找到一条从初始节点到目标节点的路径，A\*算法采用评价函数f(n) = g(n) + h(n)，其中：

- n是待评价的节点。

- g(n)是从初始节点s到节点n的最佳路径耗散值的估计。

- h(n)是从节点n到目标节点t的最佳路径耗散值的估计，称为启发函数。

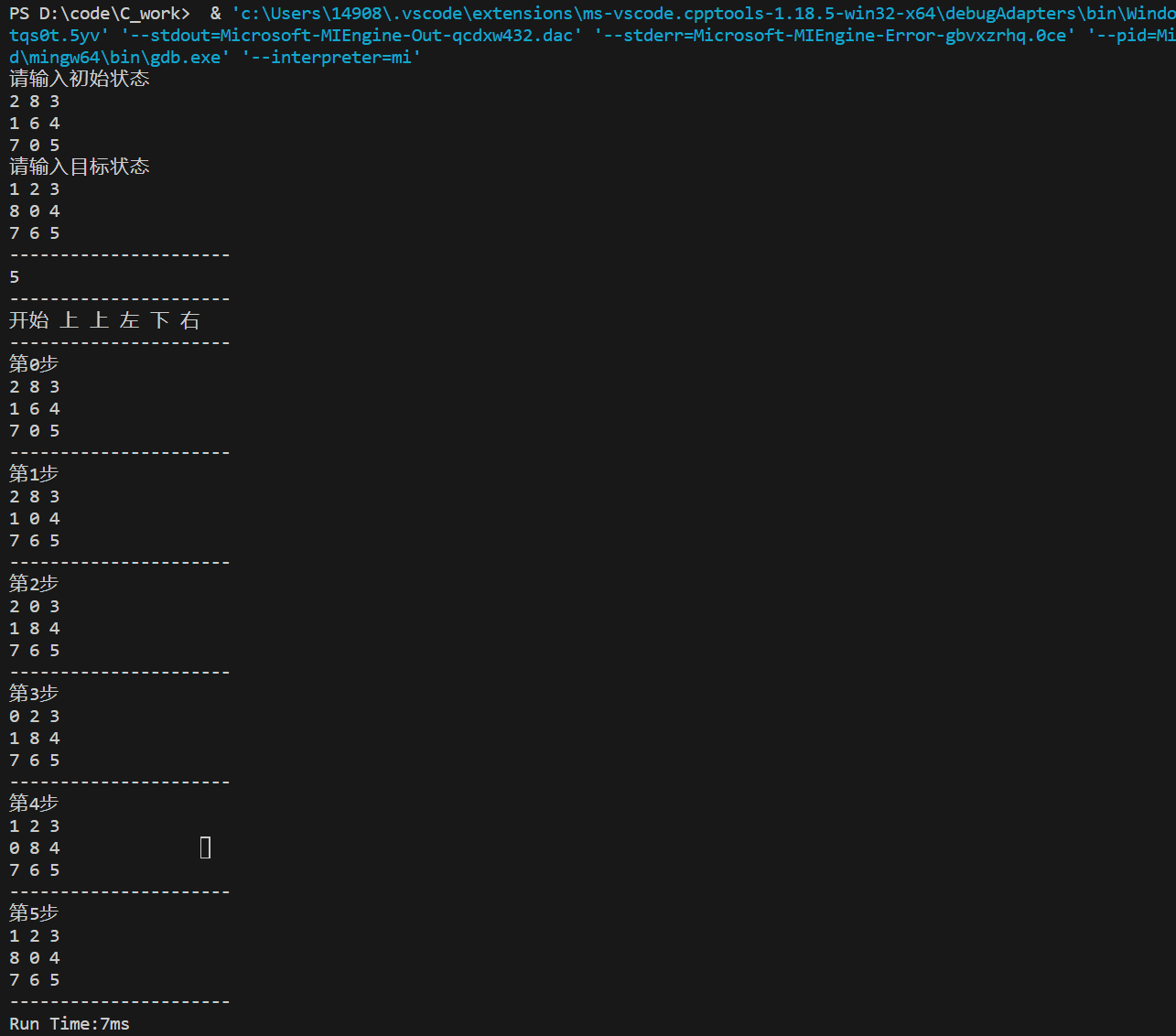
- f(n)是从初始节点s经过节点n到达目标节点的最佳路径耗散值的估计，也是评价函数。

在每次扩展时，A\*算法选择具有最小评价函数值f(n)的节点。其中，g(n)表示已经走过的路径的实际代价，而h(n)是从当前节点到目标节点的估计代价。当启发函数满足h(n) ≤ h\*(n)时，其中h\*(n)是从节点n到目标节点t的最优实际耗散值，可以证明在问题有解的情况下，A\*算法一定会找到一条耗散值最小的路径。这个条件的满足使得A\*算法成为一种优秀的启发式搜索算法。

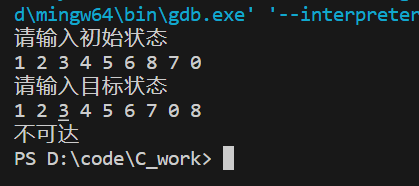
因此我们针对八数码问题使用A\*算法，其中g（n）表示当前结点和初始结点的距离，h(n)为当前状态和目标状态不一致的数字的个数

2 要求界面显示初始状态、目标状态和中间搜索步骤；

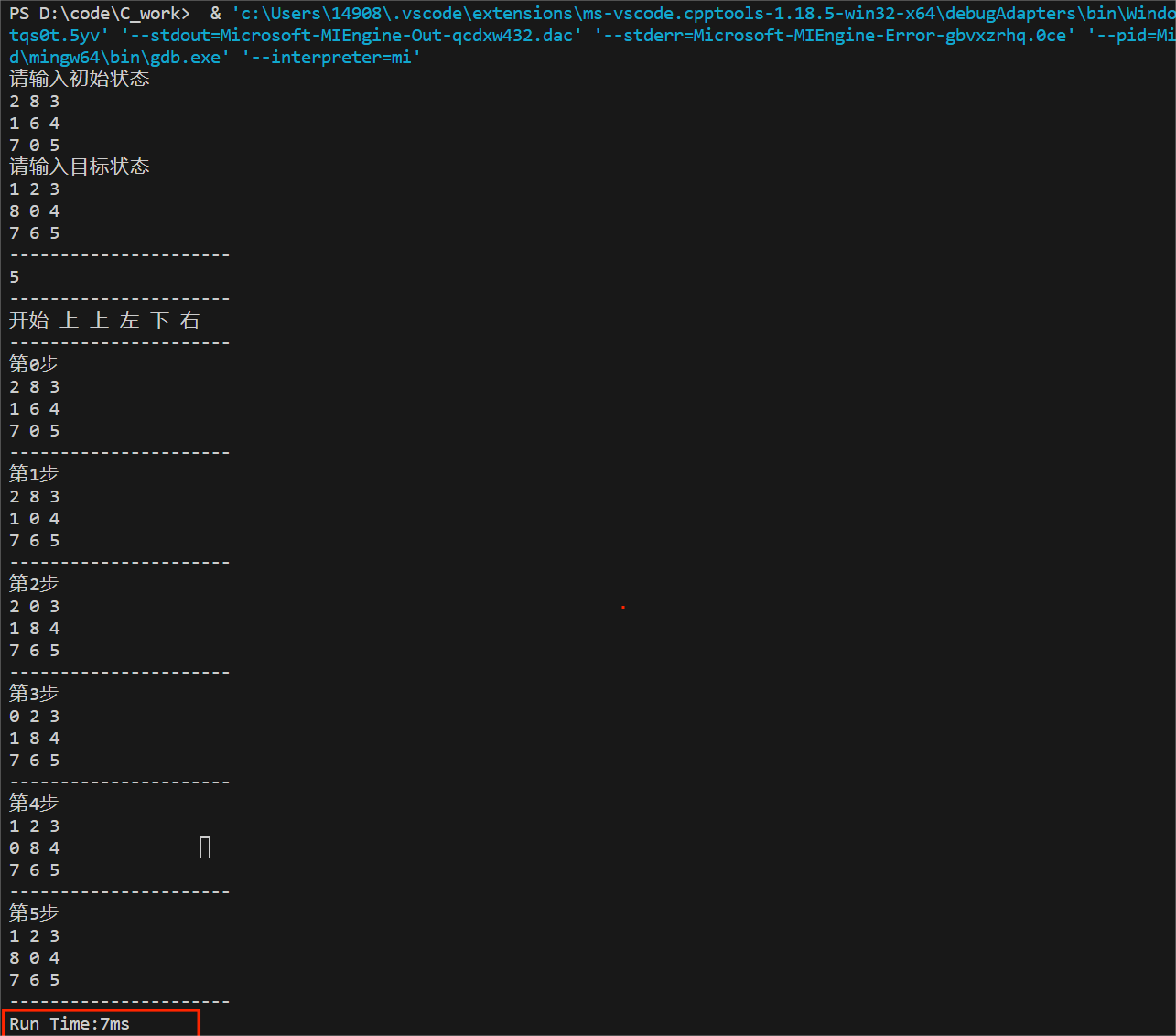
**运行结果：**



3 对不可达状态能进行正确识别；



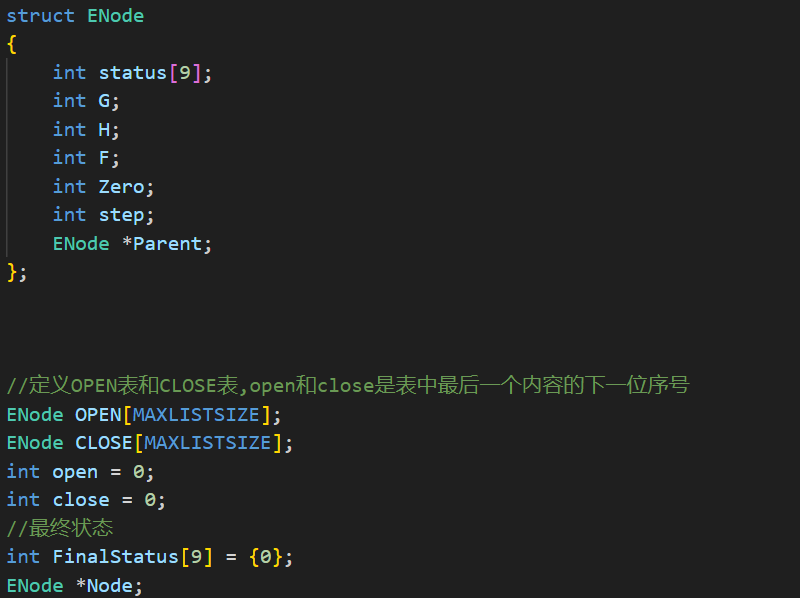
4 对所采用的启发式函数做出性能分析。



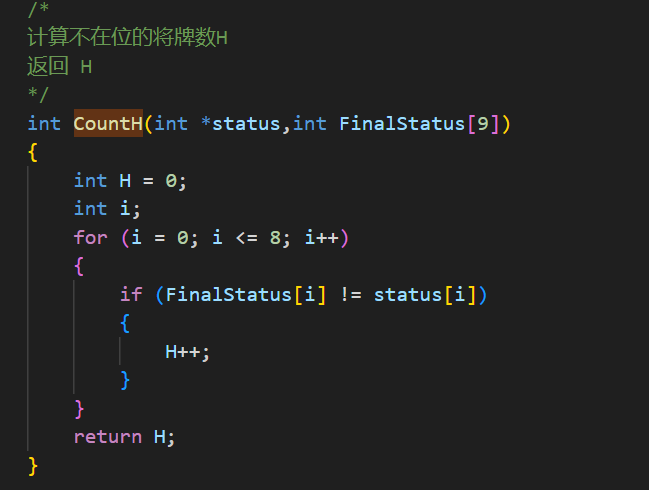
可以看到对于我们实验要求中的示例，我们执行时间仅用了7ms，效率还是非常高的。

**五 代码分析**

1. **定义八数码结构体和全局变量**



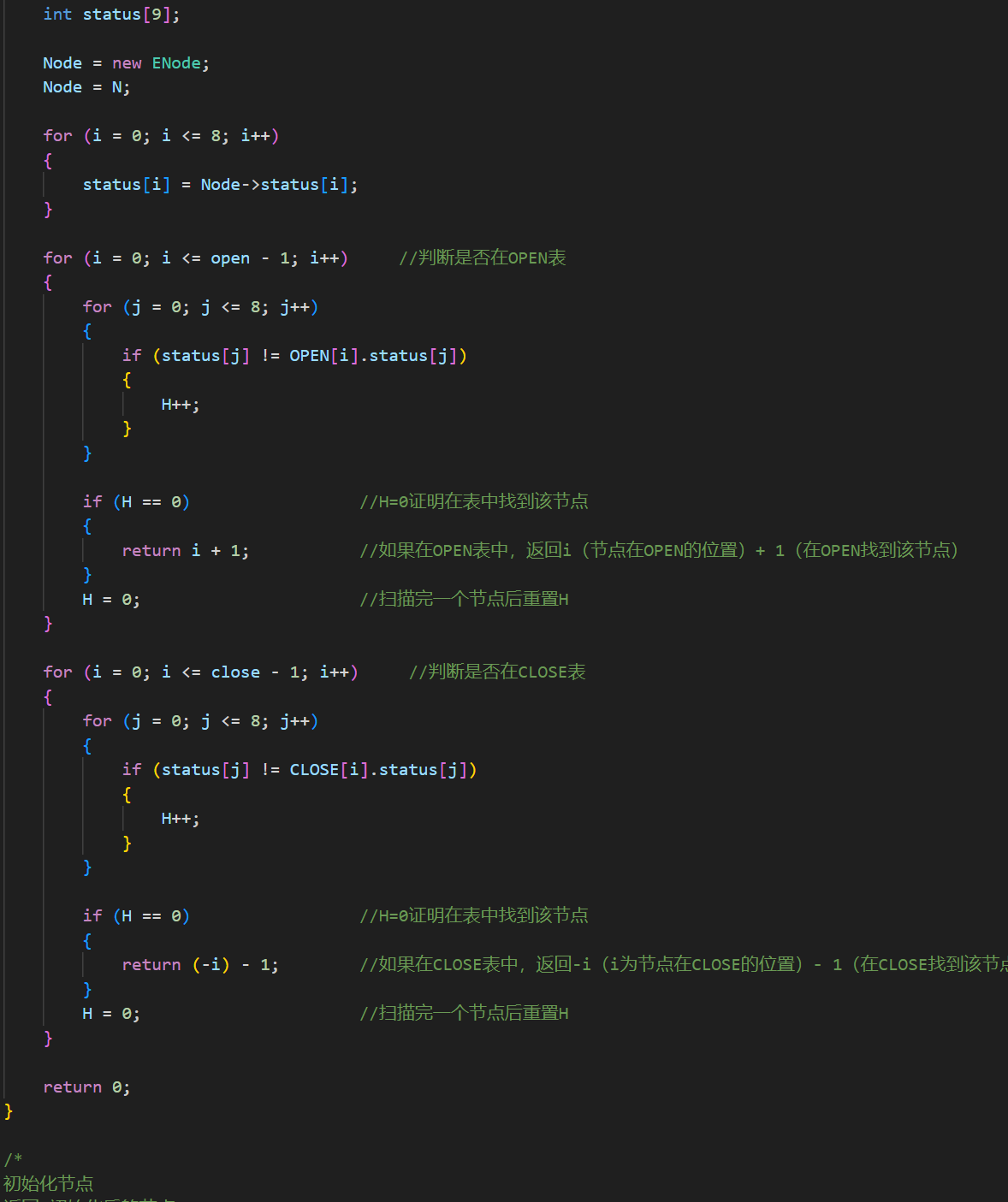
1. ENode 结构体表示八数码问题中的节点，包括状态、步数、不在位的牌数等信息。
2. OPEN 和 CLOSE 分别是存储待扩展和已扩展节点的数组。
3. open 和 close 是记录 OPEN 和 CLOSE 数组的末尾元素位置的指针。
4. FinalStatus 存储目标状态。
5. Node 是一个全局指针，用于辅助操作节点。
6. 计算启发式函数H的数值CountH



CountH 函数计算当前状态与目标状态之间的不在位的牌数，即 H 值。

通过比较每个位置上的牌，不同的数量即为不在位的牌数。

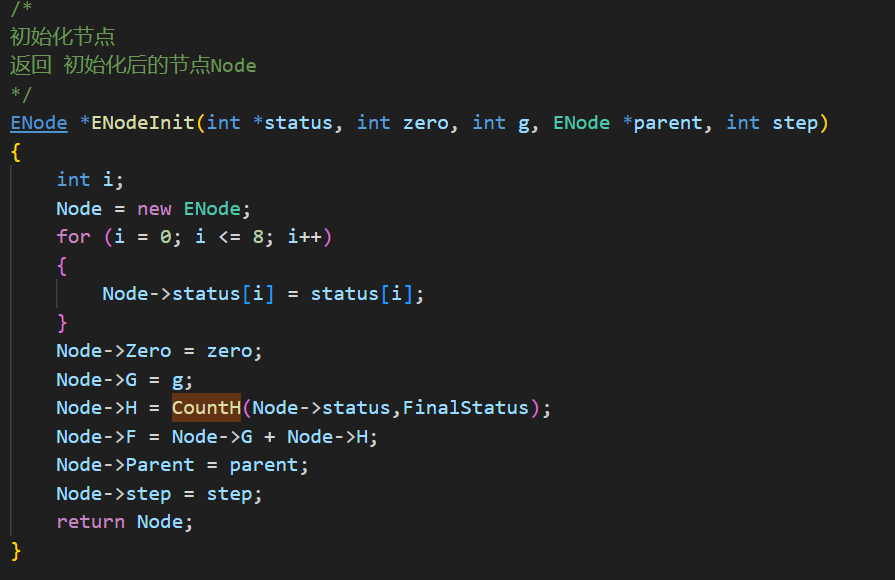
1. 判断结点是否存在与open与close表中

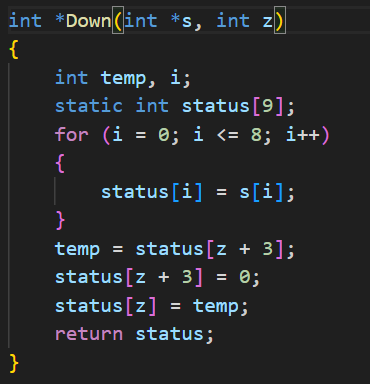
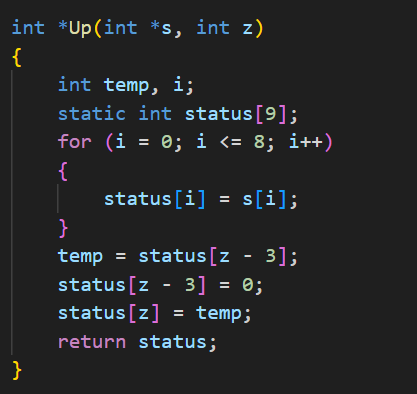
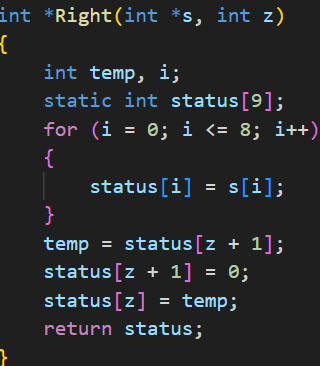
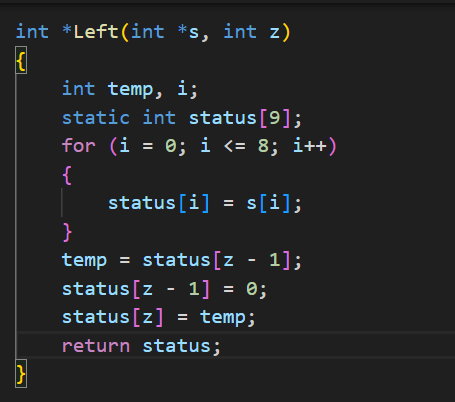


Exist 函数用于判断新生成的节点是否已存在于 OPEN 表或 CLOSE 表中。

通过比较状态数组判断节点是否相同，返回值表示存在于 OPEN 表（正数）、CLOSE 表（负数）或均不在表中（0）。

1. 初始化与状态扩展

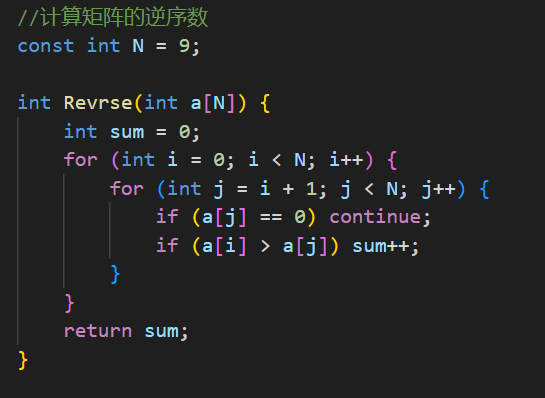




ENodeInit 函数用于初始化新的节点，设置状态、零牌位置、步数、父节点等信息。

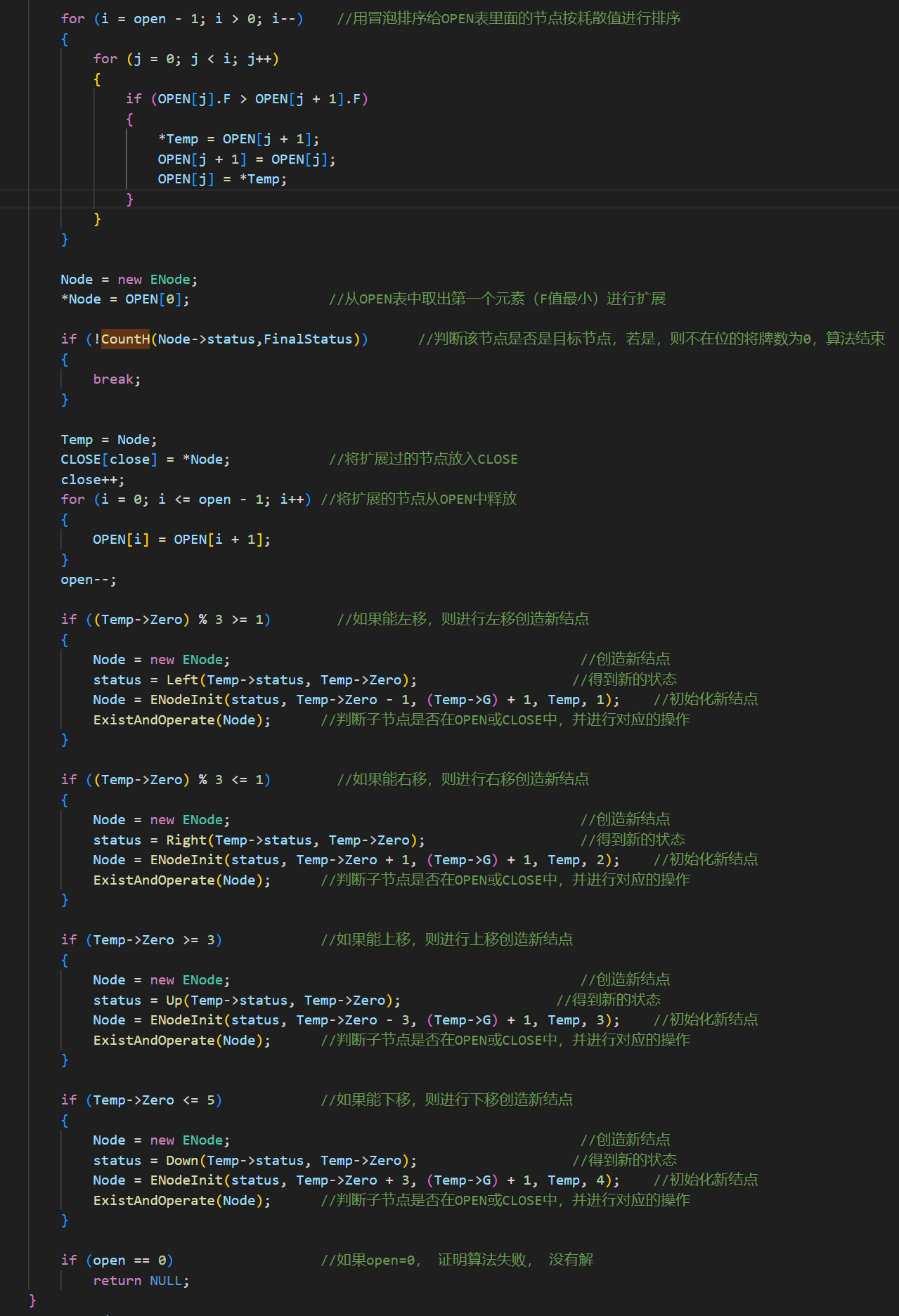
上下左右这些函数分别实现了左移、右移、上移和下移的操作，返回移动后的状态数组。

1. 判断是否存在解，根据数列的奇偶性



Revrse 函数用于计算状态数组的逆序数，用于判断是否有解

1. A\*算法



Astar 函数是 A\* 算法的主体，通过循环选择 F 值最小的节点进行扩展，将扩展后的结点放到open表中进行排序，将H值最小的放在最前面并取出给到closed表中，然后再根据新的结点进行扩展，循环，直到找到目标状态或 OPEN 表为空。

**六 实验检查要求**

1 界面显示要求：

（1）包含初始状态和目标状态的显示，可由指导老师随机输入状态进行检查；

（2）对不可达状态应给出提示；

（3）每次搜索过程的步骤，不要求显示树型结构，只要求（动画）表现，每走一步的状态变化；

（4）每走一次搜索步，需要有步数的累积显示；

（5）最后有完成一次搜索完毕的结果显示。

2 代码要求

要求提供节点定义代码，open表和close表的详细定义，以及搜索过程代码。

3 讲解要求

要求学生讲解自己设计代码的构架，具体实现方法（包含使用了何种数据结构，open表和close表的结构等）；要求学生说明自己所使用的搜索算法，以及程序运行效果。

1. **实验总结**

通过实践，我对A\*算法的工作原理有了更深入的理解。特别是如何利用估价函数来指导搜索，从而有效地减少搜索空间。我学会了如何设计有效的估价函数来估算从当前状态到目标状态的代价，这是启发式搜索中的关键步骤。通过编程实践，我不仅提升了我的编程能力，还学会了如何将理论应用于实际问题解决中。

**八 补充知识**

**1 八数码问题的有解和无解判定**

将九宫格中数字顺序排列后，形成一个包含0在内的9位数字序列，该字串可用来表示九宫格的当前状态，其中0表示空格所在位置。当空格上下、左右移动时，易知序列的逆序值奇偶性不会发生改变。由此可知，九宫问题的362,880种状态被分成逆序值为奇数和逆序值为偶数两部分，每一部分内任意两种状态相互可达。

在八数码问题中，有些状态之间是不可达的。如果我们能在一开始先判断初始状态和目标状态之间是否可达，这样可以避免不可达状态之间的盲目求解。

两个状态之间是否可达可以通过两个状态逆序值的奇偶性进行判断。我们把每个状态看作一个数列，然后计算数列的逆序值，若两个数列逆序值的奇偶性相同，则对应的两个状态是可达的，否则不可达。（注：求逆序值不把空格算在内）。

如图2所示状态：

|  |  |  |
| --- | --- | --- |
| 2 | 3 | 1 |
| 5 | 8 | 4 |
| 6 |  | 7 |

图2 棋局示例

它对应的数列是：23158467（不包括空格）。

对于一个数列，数列中每个数的逆序值是指位于这个数前面的比这个数大的数的个数。数列的逆序值就是数列中每个数的逆序值之和。

逆序值求法：例：23158467的逆序值为6，求解过程为：

0+0+2(1<2,1<3)+0+0+2(4<5,4<8)+1(6<8)+1(7<8)=6

而状态12345678 的逆序值自然就是0。

**2 曼哈顿路径**

曼哈顿距离（Manhattan Distance），又称为出租车距离，是由十九世纪的赫尔曼·闵可夫斯基所创词汇，是种使用在几何度量空间的几何学用语，用以标明两个点上在标准坐标系上的绝对轴距总和。

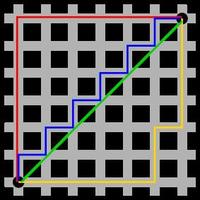


图3 曼哈顿距离示意图

我们可以定义曼哈顿距离的正式意义为L1-距离或城市区块距离，也就是在欧几里德空间的固定直角坐标系上两点所形成的线段对轴产生的投影的距离总和。

　　例如在平面上，坐标（x1, y1）的点P1与坐标（x2, y2）的点P2的曼哈顿距离为：

　 　|x1 - x2| + |y1 - y2|

要注意的是，曼哈顿距离依赖坐标系统的转度，而非系统在坐标轴上的平移或映射。

**3 康托展开**

（1）康托展开的公式:

把一个整数X展开成如下形式：

X=a[n]\*(n-1)!+a[n-1]\*(n-2)!+...+a[i]\*(i-1)!+...+a[2]\*1!+a[1]\*0!

其中，a为整数，并且0<=a[i]<i(1<=i<=n)

（2）康托展开的应用实例

{1,2,3,4,...,n}表示1,2,3,...,n的排列如 {1,2,3} 按从小到大排列一共6个。123 132 213 231 312 321 。 代表的数字 1 2 3 4 5 6 也就是把10进制数与一个排列对应起来。 他们间的对应关系可由康托展开来找到，如想知道321是{1,2,3}中第几个大的数可以这样考虑 ：

　　第一位是3，当第一位的数小于3时，那排列数小于321 如 123、 213 ，小于3的数有1、2 。所以有2\*2!个。再看小于第二位2的：小于2的数只有一个就是1 ，所以有1\*1!=1 所以小于321的{1,2,3}排列数有2\*2!+1\*1!=5个。所以321是第6个大的数。 2\*2!+1\*1!是康托展开。

再举个例子：1324是{1,2,3,4}排列数中第几个大的数：第一位是1小于1的数没有，是0个 0\*3! 第二位是3小于3的数有1和2，但1已经在第一位了，所以只有一个数2 1\*2! 。第三位是2小于2的数是1，但1在第一位，所以有0个数 0\*1! ，所以比1324小的排列有0\*3!+1\*2!+0\*1!=2个，1324是第三个大数。

（3）康托展开的启示

图搜索求解中，对于频繁访问的OPEN表和CLOSE表，有时需要判断两个节点是否相同，求节点序列的康托展开是个不错的方法。