**人工智能实验报告**

**——关于解数独和生成数独的算法**

# 1. 程序背景

1. 解数独算法所完成的功能
2. 通过在MFC中输入要解的数独，可以完成对数独部分初值的初始化
3. 能够判断可以放置在数独row行col列的数字，即空白处的候选数字
4. 能够判断是否可以成功的在row行col列放置数字num
5. 能够求解一个正确的数独

生成数独算法所完成的功能

1. 随机生成一个完整的数独
2. 将生成的数独对应显示到MFC中的每一个空格中
3. 算法的背景与意义

之所以选择编写数独算法，是因为我本身就是一名数独爱好者，经常在闲暇的时间里玩数独游戏，也会参加学校举办的数独比赛，对数独的解法也掌握了一定的技巧，但是对于解数独的方法，人脑终究有别于机器。对于数独的解法，人脑可以眼观六路，每一行每一列以及每一个3×3方格内的数字一目了然，因而在解数独的时候靠得是脑力，眼力等各方面因素；而机器就有所不同，他们没法做到像人脑一样智能，只能在每一次填入数字的时候通过遍历每一行每一列以及所在3×3的方格来决定是否填入数字，因而每一个空格的数字都需要遍历几十甚至上百次才能最终得出，在我们看来，这需要耗费巨大的时间，但是实际上机器的运行速度使得整个程序仍然能够在短时间内完成。而对于生成完整的随机数独的算法，只需要随机生成数独的第一行，后面的通过解数独算法即可得出。

我一直希望能够亲手编写一个数独的算法，一来可以加深对数独算法的理解，使得自己的数学思维能力有所提升；二来也可以感受一下机器强大的运行速度；第三就是通过这个还算比较复杂的算法来提高自己的专业能力和专业素养；第四也算是满足自己的小小愿望。

数独算法的意义就在于，可以通过机器来完成数独的求解，不需要经过人脑，使得数独这种看似比较难的数学题目在机器的帮助下变得十分简单，减少了人的脑力劳动，也可以节约大量本应花费在解数独上的时间。使得数独变成一种“人尽可解”的题目。

# 2. 关键技术

1. 使用的C++相关知识
2. 数独算法中，数组的使用是必不可少的。在整个程序中，大部分数据的存储都是通过数组来实现的。
3. 实现遍历和条件判断的过程中大量使用了循环结构和选择结构，整个程序按照顺序结构执行。
4. 递归函数的大量使用是实现回溯法的关键因素。
5. 通过使用MFC做出数独界面，而不是通过控制台进行操作，使得界面更加友好和人性化。
6. 开发程序使用的关键技术

回溯技术：回溯技术是一种选优搜索技术，按选优条件向前搜索，以达到目标。

但当探索到某一步时，发现原先选择并不优或达不到目标，就退回一

步重新选择，这种走不通就退回再走的技术为回溯技术，而满足回溯

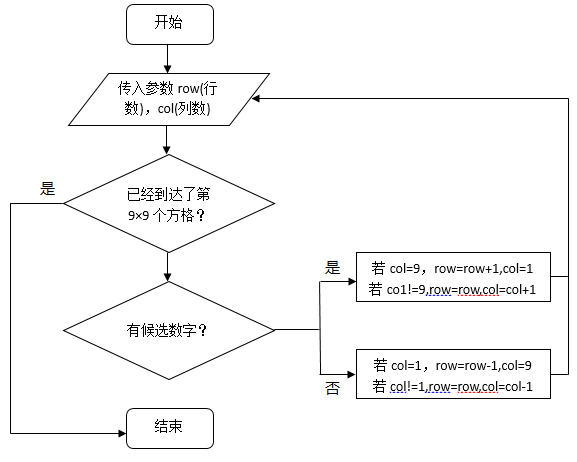
条件的某个状态的点称为“回溯点”。

# 3. 程序设计

1. 开发思路

首先，编写解数独算法，就要知道机器在解数独时的逻辑。解数独的顺序是自上而下，从左往右。因而，向空格内填入候选数字时也应该先从第一个空格开始填起。那么向数独内填数的时候应该填哪些数呢？因而需要一个能够判断该空格的候选数字的方法并且需要一个数组存储这些候选数字。如此向一个一个空格内填入候选数字，直到某一个空格内没有候选数字，则返回上一个空格填入下一个候选数字，直到遍历到第9×9个方格，退出函数，输出最终的答案。而生成数独的算法只需要在解数独的基础上稍作改变：使用rand()和srand()方法随机生成数组的第一行，后续生成则由解数独算法来完成。

1. 流程图

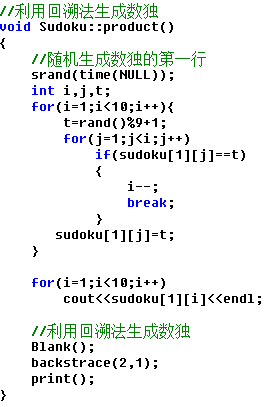


1. 数据结构以及算法思路

数独算法使用的核心方法为回溯法，回溯法的基本思想是：从一条路往前走，能进则进，不能进则退回来，换一条路再试，以此反复，最终可以得出一条最优路径，在数独中来说，即可以得到数独的解。用回溯法解决问题的一般步骤为：针对所给问题，定义问题的解空间，它至少包含问题的一个解—确定易于搜索的解空间结构，使得能用回溯法方便地搜索整个解空间—以深度优先的方式搜索解空间，并在搜索过程中用剪枝函数避免无效搜索。在回溯法中，问题的解空间是在搜索问题解的过程中动态产生的，这是回溯算法的一个重要特性。我采用的算法思路为：将开始节点（第一个填入候选数字的节点）作为扩展节点，向纵深方向移至一个新节点，这个新节点就成为新的扩展节点，当当前扩展节点不能向纵深方向移动，则该节点就成为死节点，此时，回溯至最近的一个活结点处，并将这个活节点作为当前的扩展节点。最终，以这种工作方式递归地在解空间中搜索，直至求解出该数独。

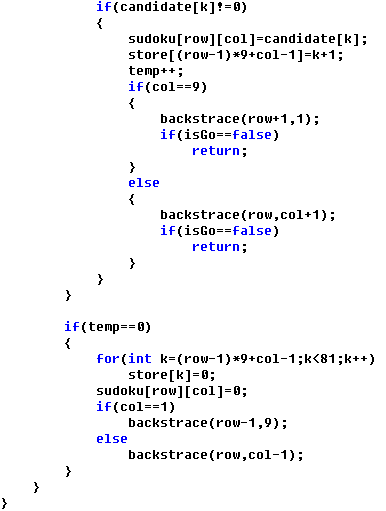
# 4. 程序实现

1. 关键代码
2. 生成数独的算法是建立在解数独算法之上的，但是还需稍加改进。如下代码的前面部分是使用rand()和srand()方法来随机产生数独的第一行，后续部分由解数独算法得出。



1. 由于回溯是通过调用递归函数来实现的，所以函数的第一条if语句是判断是否结束递归函数的重要标志。第二部分（即第二条最外层的if语句）是判断当前方格是否在数独的初始状态就已经被赋值。如果是，则此时引入了一个全局静态变量temp，用来判断本次回溯是向前回溯得到的还是向后回溯得到的，如果是向后回溯得到的，则继续向后回溯；如果是向前回溯得到的，则继续向前回溯。如果在数独的初始状态未被赋值，则到函数的第三部分（即第三条最外层的if语句），进入该if语句后，首先判断该方格内有哪些候选数字，此时，需要一个全局静态变量来存储for循环开始的值，当本次回溯是被上一个空格调用（即向后回溯）时，则循环每次从0开始；当本次回溯是被下一个空格调用（即向前回溯）时，则循环每次从上一次循环到的位置继续进行。我采用的全局静态变量为store[81]，存储每一个方格的循环变量值，初始值全部为0。如果存在候选数字，则向后回溯；如果不存在候选数字，则向前回溯。（注：向前回溯是指向之前的空格处回溯，向后回溯是指向下一个空格回溯）





1. 实现中遇到的问题和解决方案
2. 解数独算法

由于未定义静态全局变量store[81]，而导致程序运行时逻辑出错，形成死循

环。在刚开始写选择候选数字的for循环时，每次循环从0开始，在Compile

和Build时都不会报错，但是程序一旦开始运行，就无法停止，然后会出现对

话框提示程序运行过程出现了错误。然后在调试过程中，我发现，当程序向前

回溯时，每次仍然选择的是第一个候选数字，导致后面的程序又重复进行了一

遍，如此反复，最终导致了程序无法终止。因而，我选择引入一个静态全局数

组，存储每一个格子的循环变量，当向前回溯时，该空格以及之后的循环变量

全部清零；当向后回溯时，前面的循环变量维持当前状态的下一个值。这样一

来，就不会使得重复运行代码。

由于未定义静态全局变量temp，而导致程序运行时逻辑出错，形成死循环。未

定义temp时，当程序向前回溯遇到初始状态已经赋值的方格，就会按照刚开

始的定义（遇到已经赋值的方格，则继续向后回溯）继续运行代码，所以此处

又出现了一直不断的向前回溯向后回溯的问题，导致程序变成死循环。因而，

我定义了全局静态变量temp，它用来存储一个值，以判断当前程序是向前回溯

得到的还是向后回溯得到的。在上一层中，如果存在候选数字，则temp!=0，

接下来向后回溯；如果不存在候选数字，则temp=0，接下来向前回溯。因为

temp是全局静态的，在每次判断之后使用之前都会初始化为0，因而在下次回

溯算法的中判断temp是否为0，即可知道本次程序是如何得到的，从而可以确

定程序接下来应该向前回溯还是向后回溯。

由于使用了错误的return语句，导致递归函数无法终止，程序进入死循环。在

代码接近成功的时候又遇到了一个问题，那就是我无法成功退出递归函数。刚

开始我判断递归函数的终止条件if(isGo==false)return;放在了程序的第一句，但

是发现程序运行到9×9的方格之后就会报错。后来我把判断语句放在了每一次

调用递归函数的下一句，程序运行成功。然后我又将第二个if和最后一个if语

句中的判断语句注释掉运行代码，仍然可以成功。最终的版本就是现在的程序

代码。

1. 生成数独算法

第一次的生成算法中，在backstrace(2,1);之前我并没有调用Blank()函数，程序编译可以通过，但是由于逻辑出错，导致我的程序在运行之后就会异常停止。在比对了之前的数独解法和现在的数独生成之后，我发现，唯一不同的地方是我在随机生成数独的第一行之后，并没有使用Blank()函数为blank[][]数组赋值，而blank[][]数组在回溯法解数独的过程中又是很重要的判断因素，所以导致运行失败。在生成第一行之后，使用backstrace(2,1);之前我调用了Blank()函数初始化了blank[][]数组，程序便成功运行生成一个随机数独。

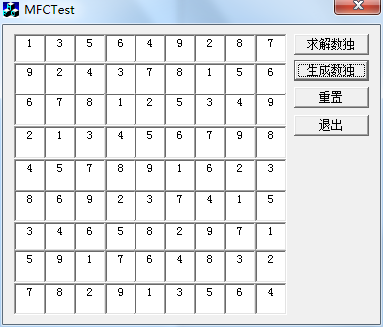
# 5. 结果展示与分析

解数独算法：

输入： 输出：

生成数独算法：



# 6. 心得体会

本次实验中我主要用到的方法是回溯法，在决定做数独的时候我已经对数独的解法有了大概的想法，但是想法果然还是天真美好的。虽然回溯算法说起来很简单，这条路走不通就换一条走，但是不得不说，在思考的时候要考虑的因素还是很多，刚开始我对数独算法列出了大概的流程图，然后一步一步实现，到最后回溯解数独的时候出现了很多的问题，通过调试器一步一步进行调试，有时候确实很没有耐心，因为对数独本身来说就是一个9×9的数组，在遍历过程的中还有行列遍历以及3×3的方格遍历，并且回溯主要是通过递归实现的，这就导致我在调试的过程中可能需要经历几十次循环才可能找到问题的根源。但是，功夫不负有心人，在我经历了数十次的调试几百次的循环遍历之后，终于解决了所有的bug，完成了整个所有代码的编写。

通过这次实验，我学到了很多的知识。其中，最令我感触深刻的就是回溯法，虽然理解起来并不那么难，但是实际操作起来却也不会太简单。其次，通过这次作业，我也得到了教训，就是在做实验之前，应该细致的考虑好每一个函数，每一个步骤所需要考虑的因素，这才能导致后续过程中不会花费过多的时间在调试上面。

最后，感谢老师的悉心教导，使得我对人工智能这门课程有了初步的了解。在这次实验中，我最大的收获就是学会了从问题的本质看待问题：通过使用调试器去观察每一个步骤的运行情况，从而了解程序运行的机理，这将使我受益终生。