suduku_solution

什么是suduku?

`suduku`(中译:数独)是一款基于数字的逻辑位置的益智游戏，源自18世纪瑞士，也有说法是起源于日本。游戏的目标是用数字填充9x9的宫格，让每一列，每一行和每个3x3小九宫部分都包含1到9之间的数字。在游戏开始时，9x9的宫格中会有一些方格已填上数字。数独不需要计算和特殊的数学技能，只需要智慧和专注。

程序介绍

此程序能够解决`9*9`的suduku，基本思想是基于深度优先搜索来实现

`Solution`类

- 构造函数`Solution()`初始化数独板的大小为9。
- 成员函数`isValid(int x, int y, char k, vector<string>& board)`检查在给定位置`(x, y)`填入数字`k`是否有效。它检查当前行、列和3x3子网格中是否已存在该数字。
- 成员函数`bt(vector<string>& board)`是一个递归的回溯算法，用于尝试填入每个空格（用'.'表示）的值，并检查是否满足数独的规则。如果找到一个解，则返回`true`；否则，尝试下一个可能的数字。
- 成员函数`solveSudoku(vector<string>& board)`使用`bt`函数来解决数独问题。如果找到解，则打印出来；如果没有解，则打印一条消息。
- 成员函数`print()`打印输入要求，提示用户如何输入数独板。

完整代码

`main.cpp`

```cpp
#include"methods.h"

int main() {
    Solution temp = Solution();

    temp.print();

    for (int i = 0; i < 9; ++i) {
        cin >> temp.board[i];
    }
    temp.solveSudoku(temp.board);
    return 0;
}
```

`methods.cpp`

```cpp
#include"methods.h"

Solution::Solution() {
    this->board.resize(9);
}
```

```
bool Solution::isValid(int x, int y, char k, vector<string>& board) {
    for (int i = 0; i < y; ++i) {
        if (board[x][i] == k) return false;
    }
    for (int i = y + 1; i < 9; ++i) {
        if (board[x][i] == k) return false;
    }
    for (int i = 0; i < x; ++i) {
        if (board[i][y] == k) return false;
    }
    for (int i = x + 1; i < 9; ++i) {
        if (board[i][y] == k) return false;
    }
    for (int i = x / 3 * 3; i <= x / 3 * 3 + 2; ++i) {
        for (int j = y / 3 * 3; j <= y / 3 * 3 + 2; ++j) {
            if (board[i][j] == k) return false;
        }
    }
    return true;
}
bool Solution::bt(vector<string>& board) {
    for (int i = 0; i < 9; ++i) {
        for (int j = 0; j < 9; ++j) {
            if (board[i][j] == '.') {
                for (char k = '1'; k <= '9'; ++k) {
                    if (isValid(i, j, k, board)) {
                        board[i][j] = k;
                        if (bt(board)) return true;
                        board[i][j] = '.';
                    }
                }
                return false;
            }
        }
    }
    return true;
}

void Solution::solveSudoku(vector<string>& board) {
    if (bt(board)) {
        for (int i = 0; i < 9; ++i) {
            for (int j = 0; j < 9; ++j) {
                cout << board[i][j];
                if (j == 2 || j == 5) cout << "  ";
            }
            if (i == 2 || i == 5) cout << endl;
            cout << endl;
        }
    }
    else {
        printf("此数独无解！！！");
    }
}
void Solution::print() {
```

```
    cout << "-----9*9 数独破解器-----" << endl;
    cout << "输入要求：" << endl << "1. 空白地方用'.'号代替（忽略引号）" << endl
<< "2. 不空格的连续输入九个数，然后回车，重复此操作" << endl;
    cout << endl << endl << "↓开始输入↓" << endl;
}
```

`methods.h`

```
#include<vector>
#include<string>
#include<iostream>
using namespace std;

#pragma once
#ifndef _METHODS_H_
#define _METHODS_H_

class Solution {
public:
    vector<string> board;

    Solution();
    bool isValid(int x, int y, char k, vector<string>& board);
    bool bt(vector<string>& board);
    void print();
    void solveSudoku(vector<string>& board);
};

#endif
```

使用方法

1. 运行程序
2. 跟着要求输入

> 空白地方使用 `.` 号代替
>
> 不空格的连续输入九个数(包含`.`在内), 然后点击回车, 重复九次操作

3. 返回结果

使用教程

输入案例:

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

输入:

```
53..7....
6..195...
.98....6.
8...6...3
4..8.3..1
7...2...6
.6....28.
...419..5
....8..79
```

输出:

```
534  678  912
672  195  348
198  342  567

859  761  423
426  853  791
713  924  856

961  537  284
287  419  635
345  286  179
```