Sudoku 簡易製作

Jia-Yin Wang

December 19, 2017

Abstract

這份講義提供一個簡單的Sudoku應用程式。使用界面一樣是使用rlutil，使用者可以用上下左右鍵輸入數字，系統會記錄解答問題所耗費的時間。遊戲的目的是希望在最短的時間內解出問題，使用者在填答的過程當中，允許最多兩次失誤，如果填錯數字5次，該次挑戰即宣告失敗。

解答Sudoku問題有很多方法，這邊主要是使用簡單的遞迴來處理，這種方法對於一般的Sudoku問題，大致可以在很短的時間內完成，但是對於最複雜的Sudoku問題，則要耗費相當時間。有興趣的讀者，可以自行爬文或設計更好的解答方法。

在本篇講義中，會使用到結構以及開檔的處理，讀者可以藉此複習相關指令。

本篇講義完整程式碼可以從 i-learning 下載，以下分成數個段落介紹程式。

1 結構和全域變數

Sudoku問題檔為二進位檔，其檔頭宣告如下：

```
struct SudokuDataHeader {
    int numbers; // 總共有多少個問題
    int datasize; // 每個問題佔多少空間 (bytes)
};
```

每個Sudoku問題，使用以下結構：

```
struct SudokuProblem {
    int id;
    int data[9][9];
};
```

其中id是問題的編號，data則是問題本身，共有81個數字，其中0代表空格，其他代陣列子內的數字。

另外我們宣告了以下全域變數：

```
SudokuProblem CurProblem, Solution; // 問題和解答
int CurPos = 18; // 目前游標位置
int Lives = 5;   // 五次失誤機會
```

2 開場

2.1 清畫面、開檔案、讀檔頭

```
cls();
fp = fopen("sudoku.dat", "rb");
fread((void*)&sdh, sizeof(sdh),  1, fp); // Read header
printf("\n\nMy Sudoku Game\n\n");
printf("\nTotal Problems = %d\n", sdh.numbers);
```

2.2 選取問題

由使用者選取問題，使用者可輸入問題編號，如果編號不在範圍內，例如輸入0，則由系統隨機選取。

```
printf("Select problem ID (Input 0 for random) : ");
scanf("%d", &k);
srand(time(NULL)); // Randomize the seed
if (k<1 || k>sdh.numbers) k = rand() % sdh.numbers;
printf("\n\nOK, Problem ID = %d\n", k);
fseek(fp, (k-1)*sdh.datasize, SEEK_CUR); // Jump to k-th records
fread((void*)&CurProblem, sizeof(CurProblem), 1, fp);
```

2.3 印出使用說明

```
printf("Use arrow keys to move. Type number for the blank position.\n");
printf("If you make 5 mistakes, then you are out!\n");
anykey("Press any key to start. Good luck!");
```

3 遊戲主體

3.1 計算問題答案

先算出問題答案：

```
Solution = CurProblem;
solve(Solution.data, 0);
```

注意，問題和答案都是相同的結構，不同的是，問題的結構資料中，會有數字0，表示未知的空格，解答的結構資料則沒有0，全部都是數字1到9。解答函式為遞迴函式，見以下說明。

3.2 解答函式

解答函式為遞迴函式，基本上有兩個參數，第1個參數表示陣列資料，第2個參數表示目前的步數，依次由左上角開始，先橫著走到底，接著走第2列，繼續走到底，然後第3列，依次類推，走完81步即表示問題解答完成。

```
int solve(int puzzle[][9], int pos)
{
    if (pos==81) return 1; // 已走完

    int row=pos/9, col=pos%9; // 步數換成行列值

    if (puzzle[row][col]) return (solve(puzzle, pos+1)); // 跳過空格

    for (int nextNum = 1; nextNum<10; nextNum++) {
        if(isValid(nextNum, puzzle, row, col)) { // 檢查可用的數字
            puzzle[row][col] = nextNum; // 可用的話先填入
            if (solve(puzzle, pos+1)) return 1; // 然後繼續走
        }
    }
    // 走完還沒回傳1，表示從pos往下找不到解，回上一層，數字先歸0
    puzzle[row][col] = 0; // reset before goback
    return 0;
}
```

在解答函式中，我們使用isValid函式，檢查在問題陣列的某行列位置填入特定的數是否允許，詳述如下。

3.3 檢查是否可填

檢查函式目的是檢查在問題陣列的某行列位置填入特定的數是否允許，判斷的方法是檢查有無違反規則，也就是該位置的是否在同一行、同一列，或者同一個大區塊中已經出現，程式碼如下。

```
int isValid(int number, int puzzle[][9], int row, int col)
{
    int rowStart = (row/3) * 3; // 九大區塊左上的row
    int colStart = (col/3) * 3; // 九大區塊左上的col

    for(int i=0; i<9; ++i) // 如果欲填的數在橫直或大區塊出現則失敗
    {
        if (puzzle[row][i] == number) return 0;
        if (puzzle[i][col] == number) return 0;
        if (puzzle[rowStart + (i%3)][colStart + (i/3)] == number) return 0;
    }
    return 1;
}
```

3.4 走棋

先列印問題陣列，注意，陣列資料為0的部份會保留空白，其餘則印出數字。

```
print_board(CurProblem);
```

列印問題陣列的函式如下，基本上程式碼應該不難了解。

```
void print_board(SudokuProblem sp)
{
    cls();
    printf("\n     Problem ID = %d\n", sp.id);
    setColor(BCOLOR);
    printf("\n +=======================+\n");
    for(int i=1; i<10; ++i) {
        for(int j=1; j<10; ++j) {
            if (j%3==1) printf(" | ");
            else printf(" ");
```

```
            setColor(CCOLOR);
            int t = sp.data[i-1][j-1];
            printf("%c", t ? t+'0' : '_');
            setColor(BCOLOR);
        }
        printf(" |\n");
        if (i == 9) printf(" +======================+\n");
        else if (i%3 == 0) printf(" +----------------------+\n");
    }
}
```

接著進入走棋的迴圈，基本上讓使用者用上下左右鍵移動目前位置，並使用數字1到9落子。

```
if (kbhit()) { // 偵測按鍵
    int k = rlutil::getkey(); // 取得按鍵
    if (k == rlutil::KEY_LEFT) --c; // 往左移
    else if (k == rlutil::KEY_RIGHT) ++c; // 往右移
    else if (k == rlutil::KEY_UP) --r;  // 往上移
    else if (k == rlutil::KEY_DOWN) ++r; // 往下移
    else if (k >= '1' && k <= '9') { // 鍵入數字
        finish = process_input(k, r, c); // 填入並檢查是否完成
        if (finish) break; // 完成則跳出
        if (!Lives) break; // 填錯五次跳出
    } else if (k == rlutil::KEY_ESCAPE) break; // 中斷遊戲
    gotoxy(40, 2); std::cout <<"("<<r+1<<","<<c+1<<")"; // Output player
    gotoxy(40, 3); std::cout <<"Lives = " << Lives;
}
```

3.5 填子函式

填子函式是在特定的row和col填入數字k，如果填完後問題解答完成則回傳1，否則回傳0。另外會檢查所填的數字是否正確，如果不正確的話不會落子。

```
bool process_input(int k, int r, int c)
{
    if (CurProblem.data[r][c]==0) { // 該位置為空格才可填
        if (k-'0' == Solution.data[r][c]) { // 填的數字正確
            printf("%c", k); // 輸出所填數字
```

```
                CurProblem.data[r][c] = k-'0'; // 存入問題陣列
                if (finished()) return true; // 檢查是否完成
            } else { // 填入數字不正確，生命值扣一點，列印警告訊息
                Lives--;
                gotoxy(40, 6);
                printf("Error!\x07");
                msleep(500);
                gotoxy(40, 6);
                printf("           ");
            }
        }
        return false;
    }
```

3.6 檢查是否完成

這個檢查函式主要是檢查是否已解答整個問題，檢查方式很簡單，看看是不是81格都填完了，沒有空格存在。

```
bool finished()
{
    for (int i=0; i<9; i++) {
        for (int j=0; j<9; j++) {
            if(CurProblem.data[i][j]==0) return false;
        }
    }
    return true;
}
```

除了以上的主要函式，其他大體上是用來列印訊息，計算時間，以及移動棋子的部份，這些部份比較次要，不在此介紹，讀者可從程式碼中了解其運作流程。

4 完整程式碼

```
1  #include <cstdio>
2  #include "rlutil.h"
3  #include <cstdlib>
4  #include <ctime>
```

```
5
6   #define BCOLOR 8
7   #define CCOLOR 7
8   #define CURSOR 11
9   using namespace rlutil;
10
11  struct SudokuDataHeader {
12      int numbers;
13      int datasize;
14  };
15
16  struct SudokuProblem {
17      int id;
18      int data[9][9];
19  };
20
21  SudokuProblem CurProblem, Solution;
22
23  void print_board(SudokuProblem sp);
24  int solve(int puzzle[][9], int pos);
25  int isValid(int number, int puzzle[][9], int row, int col);
26  void hline(int r1, int c1, int c2, int color, char c);
27  void gotopos(int k);
28  void gotopos(int r, int c);
29  bool process_input(int k, int r, int c);
30  bool finished();
31
32  int CurPos = 18;
33  int Lives = 5;
34
35  int main()
36  {
37      FILE *fp;
38      SudokuDataHeader sdh;
39      int k;
40
41      cls();
42      fp = fopen("sudoku.dat", "rb");
43      fread((void*)&sdh, sizeof(sdh),  1, fp); // Read header
44      printf("\n\nMy Sudoku Game\n\n");
45      printf("\nTotal Problems = %d\n", sdh.numbers);
```

```
46      printf("Select problem ID (Input 0 for random) : ");
47      scanf("%d", &k);
48      srand(time(NULL)); // Randomize the seed
49      if (k<1 || k>sdh.numbers) k = rand() % sdh.numbers;
50      printf("\n\nOK, Problem ID = %d\n", k);
51      fseek(fp, (k-1)*sdh.datasize, SEEK_CUR); // Jump k records
52      fread((void*)&CurProblem, sizeof(CurProblem), 1, fp);
53
54      printf("Use arrow keys to move. Type number for the blank
        ↪  position.\n");
55      printf("If you make 5 mistakes, then you are out!\n");
56      anykey("Press any key to start. Good luck!");
57
58      Solution = CurProblem;
59      solve(Solution.data, 0);
60      print_board(CurProblem);
61
62      int r=0, c=0;
63      gotopos(r,c);
64      gotoxy(40, 2); std::cout <<"(0,0)"; // Output player
65      gotoxy(40, 3); std::cout <<"Lives = " << Lives;
66      int cnt=0;
67      bool finish=false;
68      while (1) {
69          hidecursor();
70          if (kbhit()) {
71              int k = rlutil::getkey(); // Get character
72              if (k == rlutil::KEY_LEFT) --c;
73              else if (k == rlutil::KEY_RIGHT) ++c;
74              else if (k == rlutil::KEY_UP) --r;
75              else if (k == rlutil::KEY_DOWN) ++r;
76              else if (k >= '1' && k <= '9') {
77                  finish = process_input(k, r, c);
78                  if (finish) break;
79                  if (!Lives) break;
80              } else if (k == rlutil::KEY_ESCAPE) break;
81              gotoxy(40, 2); std::cout <<"("<<r+1<<","<<c+1<<")"; //
                ↪  Output player
82              gotoxy(40, 3); std::cout <<"Lives = " << Lives;
83          }
```

```
84          if (!(++cnt%2000)) { gotoxy(40, 4); std::cout << "Times = " <<
            ↪  cnt/2000; }
85          r = (r+9) % 9;
86          c = (c+9) % 9;
87          showcursor();
88          gotopos(r,c);
89          std::cout.flush();
90      }
91      gotoxy(40, 3); std::cout <<"Lives = " << Lives;
92      gotoxy(30,18);
93      setColor(15);
94      if (finish) printf("You did it! Spent time = %d!\n\n", cnt/2000);
95      else if (!Lives) printf("You lose the game!\n\n");
96      else printf("You abandoned!\n\n");
97
98      return 0;
99  }
100
101 bool finished()
102 {
103     for (int i=0; i<9; i++) {
104         for (int j=0; j<9; j++) {
105             if(CurProblem.data[i][j]==0) return false;
106         }
107     }
108     return true;
109 }
110
111 bool process_input(int k, int r, int c)
112 {
113     if (CurProblem.data[r][c]==0) {
114         if (k-'0' == Solution.data[r][c]) {
115             printf("%c", k);
116             CurProblem.data[r][c] = k-'0';
117             if (finished()) return true;
118         } else {
119             Lives--;
120             gotoxy(40, 6);
121             printf("Error!\x07");
122             msleep(500);
123             gotoxy(40, 6);
```

```
124             printf("          ");
125         }
126     }
127     return false;
128 }
129 
130 void gotopos(int r, int c)
131 {
132     int ypos[] = {5, 6, 7, 9, 10, 11, 13, 14, 15};
133     int xpos[] = {4, 6, 8, 12, 14, 16, 20, 22, 24};
134     setColor(CURSOR);
135     showcursor();
136     locate(xpos[c], ypos[r]);
137 }
138 
139 void gotopos(int k)
140 {
141     gotopos(k/9, k%9);
142 }
143 
144 int solve(int puzzle[][9], int pos)
145 {
146     if (pos==81) return 1;
147 
148     int row=pos/9, col=pos%9;
149 
150     if (puzzle[row][col]) return (solve(puzzle, pos+1));
151 
152     for (int nextNum = 1; nextNum<10; nextNum++) {
153         if(isValid(nextNum, puzzle, row, col)) {
154             puzzle[row][col] = nextNum;
155             if (solve(puzzle, pos+1)) return 1;
156         }
157     }
158     // Failed to find a valid value, go back to previous cell for
      ↪  another try
159     puzzle[row][col] = 0; // reset before goback
160     return 0;
161 }
162 
163 int isValid(int number, int puzzle[][9], int row, int col)
```

```
164 {
165     int rowStart = (row/3) * 3;
166     int colStart = (col/3) * 3;
167 
168     for(int i=0; i<9; ++i)
169     {
170         if (puzzle[row][i] == number) return 0;
171         if (puzzle[i][col] == number) return 0;
172         if (puzzle[rowStart + (i%3)][colStart + (i/3)] == number)
            ↪  return 0;
173     }
174     return 1;
175 }
176 
177 void print_board(SudokuProblem sp)
178 {
179     cls();
180     printf("\n     Problem ID = %d\n", sp.id);
181     setColor(BCOLOR);
182     printf("\n +=======================+\n");
183     for(int i=1; i<10; ++i) {
184         for(int j=1; j<10; ++j) {
185             if (j%3==1) printf(" | ");
186             else printf(" ");
187             setColor(CCOLOR);
188             int t = sp.data[i-1][j-1];
189             printf("%c", t ? t+'0' : '_');
190             setColor(BCOLOR);
191         }
192         printf(" |\n");
193         if (i == 9) printf(" +=======================+\n");
194         else if (i%3 == 0) printf(" +-----------------------+\n");
195     }
196 }
```