

Project 2 : Futoshiki Puzzle

How to run the project:

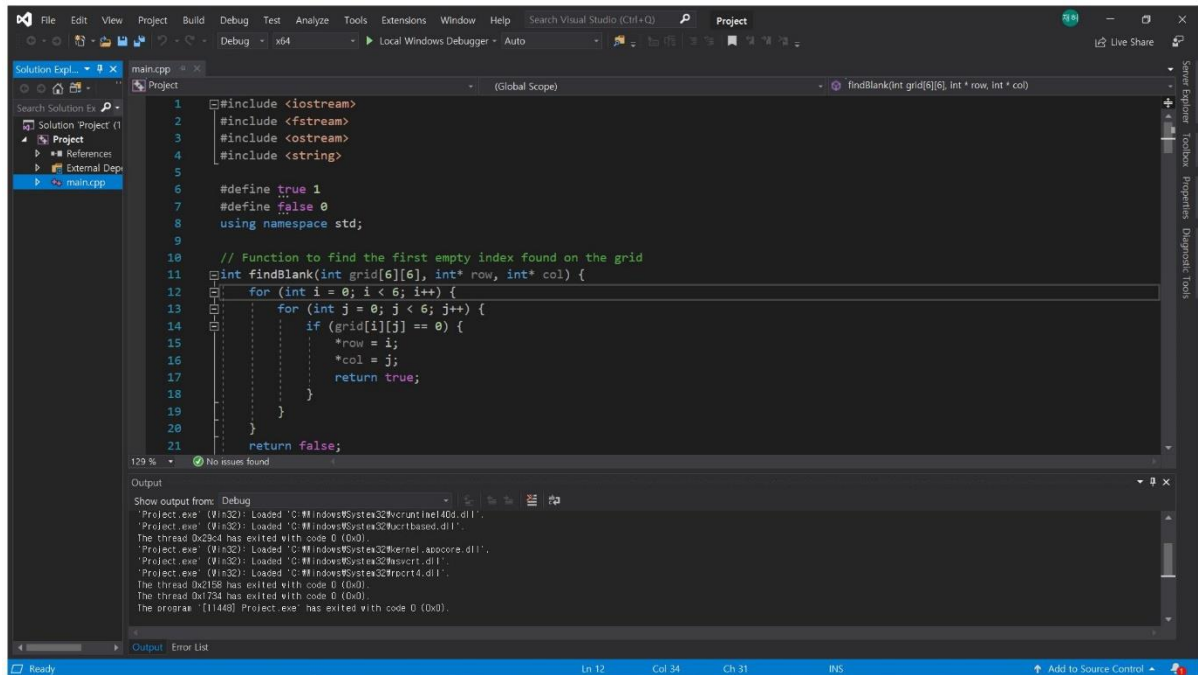
1. Unpack the AI Project2.zip file that I submitted.
2. Check if the input files are in the project folder before run Project.sin

Debug	5/18/2021 10:32 AM	파일 폴더	
x64	5/18/2021 10:32 AM	파일 폴더	
Input1.txt	4/30/2021 12:33 PM	텍스트 문서	1KB
Input2.txt	5/6/2021 7:40 PM	텍스트 문서	1KB
Input3.txt	5/6/2021 7:40 PM	텍스트 문서	1KB
main.cpp	5/18/2021 10:22 AM	C++ Source	6KB
Project.vcxproj	5/6/2021 9:51 PM	VC++ Project	7KB
Project.vcxproj.filters.txt	6/8/2020 8:58 PM	텍스트 문서	1KB
Project.vcxproj.user	6/8/2020 6:06 PM	Per-User Project ...	1KB

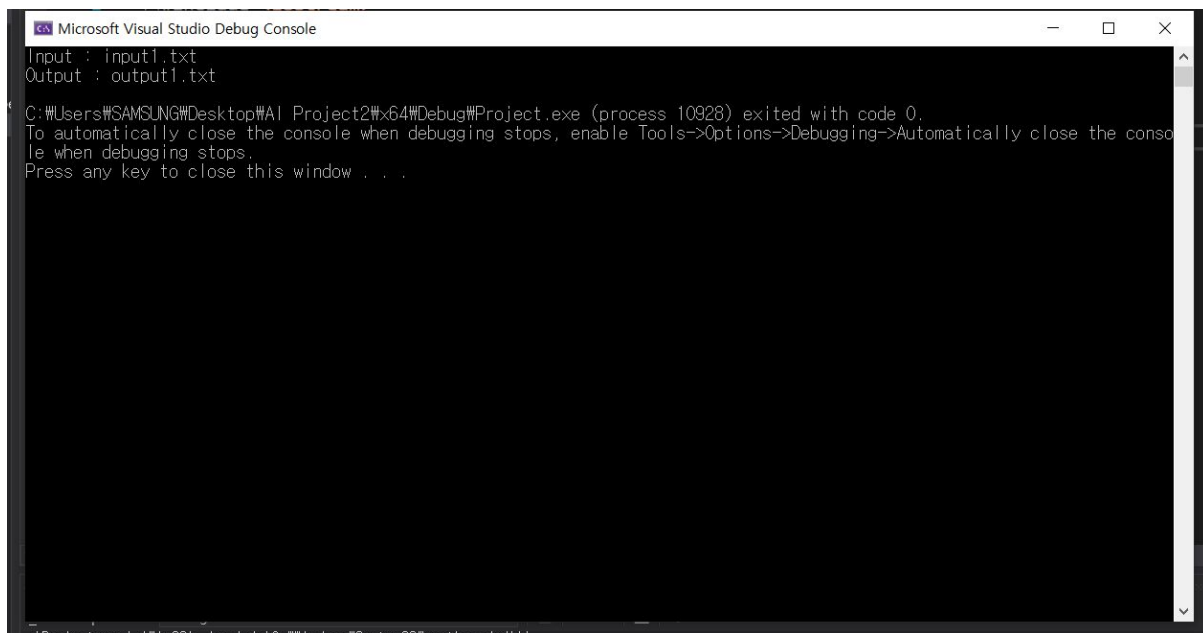
3. Run Project.sin in AI Project2 folder.

Debug	5/18/2021 10:32 AM	파일 폴더	
Project	5/18/2021 10:33 AM	파일 폴더	
x64	5/18/2021 10:32 AM	파일 폴더	
Project.sin	6/8/2020 6:06 PM	Visual Studio Sol...	2KB

4. Run the main.cpp



5. Enter the correct input file name and the name of the output file you want to create.



6. check the output file in the project folder.

Debug	5/18/2021 10:32 AM	파일 폴더	
x64	5/18/2021 10:32 AM	파일 폴더	
Input1.txt	4/30/2021 12:33 PM	텍스트 문서	1KB
Input2.txt	5/6/2021 7:40 PM	텍스트 문서	1KB
Input3.txt	5/6/2021 7:40 PM	텍스트 문서	1KB
main.cpp	5/18/2021 10:22 AM	C++ Source	6KB
output1.txt	5/18/2021 10:42 AM	텍스트 문서	1KB
Project.vcxproj	5/6/2021 9:51 PM	VC++ Project	7KB
Project.vcxproj.filters.txt	6/8/2020 8:58 PM	텍스트 문서	1KB
Project.vcxproj.user	6/8/2020 6:06 PM	Per-User Project ...	1KB

Source codes:

```

main.cpp x
Project (Global Scope) findBlank(int grid[6][6], int * row, int * col)
1 #include <iostream>
2 #include <fstream>
3 #include <ostream>
4 #include <string>
5
6 #define true 1
7 #define false 0
8 using namespace std;
9
10 // Function to find the first empty index found on the grid
11 int findBlank(int grid[6][6], int* row, int* col) {
12     for (int i = 0; i < 6; i++) {
13         for (int j = 0; j < 6; j++) {
14             if (grid[i][j] == 0) {
15                 *row = i;
16                 *col = j;
17                 return true;
18             }
19         }
20     }
21     return false;
22 }
129 % No issues found

```

```

main.cpp x
Project (Global Scope) findBlank(int grid[6][6], int * row, int * col)
22 }
23
24 // Function to check the grid to see if the specified row already contains a number
25 int usedInRow(int grid[6][6], int row, int num) {
26     for (int col = 0; col < 6; col++) {
27         if (grid[row][col] == num) return true;
28     }
29     return false;
30 }
31
32 // Function to check the grid to see if the specified column already contains a number
33 int usedInCol(int grid[6][6], int col, int num) {
34     for (int row = 0; row < 6; row++) {
35         if (grid[row][col] == num) return true;
36     }
37     return false;
38 }
39
40
41 // inequality check function
42 int check(char h[6][5], char v[5][6], int grid[6][6]) {
129 % No issues found

```

```
main.cpp * X
Project (Global Scope) findBlank(int grid[6][6], int * row, int * col)
40
41 // inequality check function
42 int check(char h[6][5], char v[5][6], int grid[6][6]) {
43     // Check for horizontal inequality.
44     for (int i = 0; i < 6; i++) {
45         for (int j = 0; j < 5; j++) {
46             // However, if at least one of the left and right spaces of the inequality sign is not padded, it is skipped without check.
47             if (grid[i][j] == 0 || grid[i][j + 1] == 0) continue;
48             switch (h[i][j]) {
49                 case '>':
50                     if (grid[i][j] <= grid[i][j + 1]) return false;
51                     break;
52                 case '<':
53                     if (grid[i][j] >= grid[i][j + 1]) return false;
54                     break;
55                 default:
56                     break;
57             }
58         }
59     }
60     // Check for vertical inequality.
129 % No issues found
```

```
main.cpp * X
Project (Global Scope) findBlank(int grid[6][6], int * row, int * col)
61     for (int i = 0; i < 5; i++) {
62         for (int j = 0; j < 6; j++) {
63             // However, if at least one of the up and down spaces of the inequality sign is not padded, it is skipped without check.
64             if (grid[i][j] == 0 || grid[i + 1][j] == 0) continue;
65             switch (v[i][j]) {
66                 case '^':
67                     if (grid[i][j] >= grid[i + 1][j]) return false;
68                     break;
69                 case 'v':
70                     if (grid[i][j] <= grid[i + 1][j]) return false;
71                     break;
72                 default:
73                     break;
74             }
75         }
76     }
77
78     // Returns true if none of the above is detected.
79     return true;
80
81
129 % No issues found
```

```
main.cpp * X
Project (Global Scope) findBlank(int grid[6][6], int * row, int * col)
82 int isSafe(int grid[6][6], int row, int col, int num) {
83     // Check that the Futoshiki rule is satisfied when number is placed in the specified row and column.
84     // The Futoshiki rule means that 1,2,3,4,5,6 should be used only once in horizontal and vertical lines.
85
86     return !usedInRow(grid, row, num) && !usedInCol(grid, col, num);
87 }
88
89 //Problem solving function (recursive call)
90 int solve(char h[6][5], char v[5][6], int grid[6][6]) {
91     // Backtracking algorithm is used.
92     // A method of proceeding by inserting the numbers 1-6 in order and checking whether the rules of the Futoshiki Puzzle are satisfied.
93
94     int row, col;
95     // Find the next blank.
96     if (findBlank(grid, &row, &col) == 0) {
97         return true;
98     }
99
100     for (int num = 1; num <= 6; num++) {
101         // When number is in the corresponding cell, it checks whether 1,2,3,4,5,6 are duplicated in a row or column,
102         if (isSafe(grid, row, col, num)) {
129 % No issues found
```

```
main.cpp * x
Project (Global Scope) findBlank(int grid[6][6], int * row, int * col)
103 // If not duplicated, add number.
104 grid[row][col] = num;
105 // Then check if the inequality sign is satisfied,
106 if (!check(h, v, grid)) {
107     // If not satisfied, it returns to zero.
108     grid[row][col] = 0;
109 } else {
110     // If satisfied, fix it to number and recursively call solve for the next cell.
111     if (solve(h, v, grid)) {
112         return true;
113     }
114 }
115 // If false is returned during recursion and has been backtracked to this side, it is returned to 0 and re-challenged to
116 grid[row][col] = 0;
117 }
118 }
119
120 // If you are here after all the loops are over, there is no correct answer.
121 return false;
122 }
123
```

```
main.cpp * x
Project (Global Scope) findBlank(int grid[6][6], int * row, int * col)
124 // Reads a character from the filestream that is not a blank or newline character.
125 char readChar(ifstream& file) {
126     while (true) {
127         char tmp;
128         file.get(tmp);
129         if (tmp != ' ' && tmp != '\n') return tmp;
130     }
131 }
132 int main() {
133     int grid[6][6];
134     char h[6][5];
135     char v[5][6];
136
137     string filename;
138     cout << "Input : ";
139     cin >> filename;
140
141     ifstream file;
142     file.open(filename);
143
144     // Read file
145
```

```
main.cpp * x
Project (Global Scope) findBlank(int grid[6][6], int * row, int * col)
145 if (file.is_open()) {
146     // fill the grid first
147     for (int i = 0; i < 6; i++) {
148         for (int j = 0; j < 6; j++) {
149             file >> grid[i][j];
150         }
151     }
152
153     // Fill in the horizontal inequality sign
154     for (int i = 0; i < 6; i++) {
155         for (int j = 0; j < 5; j++) {
156             h[i][j] = readChar(file);
157         }
158     }
159
160     // Fill in the vertical inequality sign
161     for (int i = 0; i < 5; i++) {
162         for (int j = 0; j < 6; j++) {
163             v[i][j] = readChar(file);
164         }
165     }
166 }
```

```
main.cpp * X
Project (Global Scope) findBlank(int grid[6][6], int * row, int * col)
166 }
167 else {
168     cout << "File open failed." << endl;
169 }
170
171 file.close();
172
173 // Get the name of the output file
174 cout << "Output : ";
175 cin >> filename;
176
177 ofstream outFile;
178 outFile.open(filename);
179
180 if (solve(h, v, grid)) {
181     // When an answer is found, write the result to a file.
182     for (int i = 0; i < 6; i++) {
183         for (int j = 0; j < 6; j++) {
184             outFile << grid[i][j] << " ";
185         }
186         outFile << endl;
187     }
188 }
189
190 // If you can't find an answer, write the sentence below to a file..
191 outFile << "There is no solution!";
192
193 outFile.close();
194
195 return 0;
196 }
```

```
main.cpp * X
Project (Global Scope) findBlank(int grid[6][6], int * row, int * col)
187 }
188 else {
189     // If you can't find an answer, write the sentence below to a file..
190     outFile << "There is no solution!";
191 }
192
193 outFile.close();
194
195 return 0;
196 }
```

Input and output files:

1.

Input1.txt - Windows 메모장

— □ ×

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

000000
300000
030064
000040
000000
000013

00>00
00000
00000
000<0
0>00>
>0000

v00000
000000
000^00
00000^
v00000

output1.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
4 6 3 1 2 5
3 1 2 4 5 6
1 3 5 2 6 4
2 5 6 3 4 1
6 4 1 5 3 2
5 2 4 6 1 3
```


2.



```
Input2.txt - Windows 메모장
파일(F)  편집(E)  서식(O)  보기(V)  도움말(H)
5 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 2 0 0 0 0
0 0 5 0 3 0
3 5 4 0 0 0

0 < 0 0 0
0 0 0 0 0
0 0 0 0 0
0 < 0 < 0
0 < 0 0 0
0 0 0 0 <

0 0 0 0 0 0
^ 0 0 0 0 0
0 0 v 0 0 v
v 0 0 0 0 0
0 0 0 ^ 0 0
```

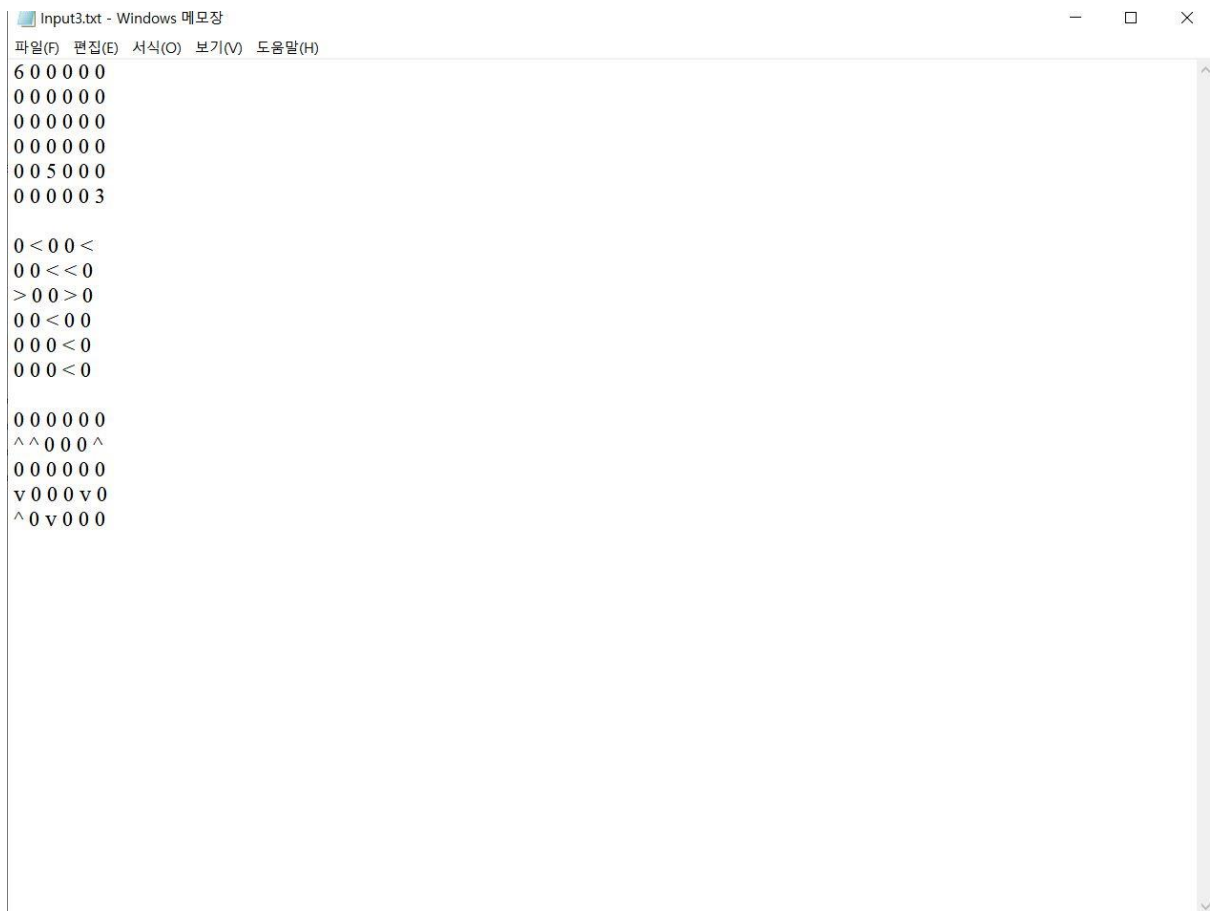
output2.txt - Windows 메모장

— □ ×

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

5 1 2 3 6 4
2 6 1 5 4 3
4 3 6 1 2 5
6 2 3 4 5 1
1 4 5 2 3 6
3 5 4 6 1 2

3.



```
Input3.txt - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
600000
000000
000000
000000
005000
000003
0<00<
00<<0
>00>0
00<00
000<0
000<0
000000
^000^
000000
v000v0
^0v000
```

output3.txt - Windows 메모장

— □ ×

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
6 3 4 1 2 5
4 1 3 5 6 2
5 2 6 3 1 4
3 5 2 6 4 1
1 4 5 2 3 6
2 6 1 4 5 3
```