南开2020软件工程结对编程

👍数独程序源代码已上传至github（https://github.com/Lunaticsky-tql/sudoku.git）仓库。

小组成员：田翔宇 2011748 , 田佳业 2013599

用户手册

功能

输入

参数名字	参数意义	范围限制	用法示例	用法解释
-c	需要的数独终盘数量	1-1000000	sudoku.exe -c 20	表示生成20个数独终盘
-s	需要解的数独棋盘文件路径	绝对或相对路径	sudoku.exe -s game.txt	表示从game.txt读取若干个数独游戏，并给出其解答，生成到sudoku.txt中
-n	需要的游戏数量	1-10000	sudoku.exe -n 1000	表示生成1000个数独游戏
-m	生成游戏的难度	1-3	sudoku.exe -n 1000 -m 1	表示生成1000个简单数独游戏，只有m和n一起使用才认为参数无误，否则请报错
-r	生成游戏中挖空的数量范围	20-55	sudoku.exe -n 20 -r 20~55	表示生成20个挖空数在20到55之间的数独游戏，只有 r 和 n 一起使用才认为参数无误，否则请报错
-u	生成游戏的解唯一		sudoku.exe -n 20 -u	表示生成20个解唯一的数独游戏，只有u和 n 一起使用才认为参数无误，否则请报错

输出

- 在exe程序的文件夹下生成对应的.txt文件
- 棋盘按照9行9列形势输出
- 不同棋盘之间空一行
- 同一个棋盘内部每个数字之间空一格

运行

1. 直接运行sudoku.exe,所得结果将出现在该文件夹下，生成的文件名称将输出到命令行中
2. 从github仓库下载源代码，在根目录下按照通常编译Cmake的方式进行编译。即

```
cmake -B <build tree> -s <source tree>
cmake --build <build tree>
```

`<source tree>`为项目路径，`<build tree>`为生成文件的路径如`build`。

实现思路

利用C++面向对象的特性，我们可以在不对性能造成过多影响的同时编写工程上干净整洁的代码，降低模块之间的耦合度，方便进行测试。如本次实验在框架中将解析参数，任务分发，文件输入输出等进行分离，并根据实际情况为实验的主要对象--棋盘添加成员方法，如将整个棋盘输出到指定输出流。

单元测试

本次实验采用Googletest框架进行单元测试。对其源代码进行编译，取出gtest部分和build生成的链接库导入cmake依赖，即可在项目中使用GoogleTest，将依赖文件加入git后无需配置环境即可实现两人配合进行代码编写和测试。

同时在本次实验中采用边编写边测试的策略稳步推进。测试用例的设计也是按照数据流的递进进行增加，下面列举了帮助定位到程序代码错误的若干测试用例，可以参看源代码获取完整的测试用例。

参数解析

测试用例

```
    const char *argv10[5] = {"sudoku.exe", "-n", "1000", "-r", "20~60"};
    EXPECT_THROW(parseArgs(5, argv10), ParseArgException);
```

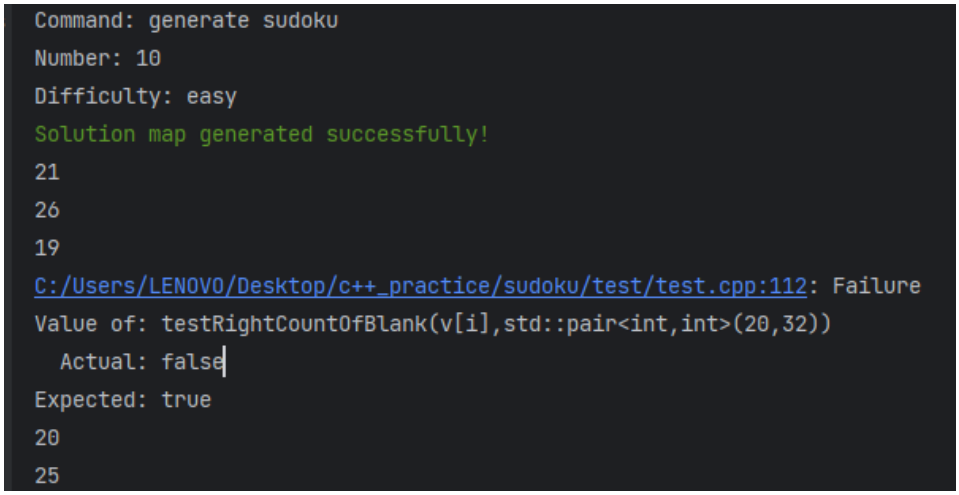
发现一开始忘记对数据范围进行检验，要求里写的清清楚楚。。。

```
    const char *argv8[5] = {"sudoku.exe", "-n", "1000", "-m", "0"};
    EXPECT_THROW(parseArgs(3, argv8), ParseArgException);
```

上面这个更蠢，把测试用例写错了。。

游戏生成

这个测试测试生成的数独挖空数是否正确，一开始测试不通过




```
TEST(GenerateGameTest,TestLevel){
    Command c1(GENERATE_SUDOKU,10,EASY);
    c1.showCommand();
    EXPECT_TRUE(distributeTask(c1));
    std::vector<Board> v;
    v=readBoardsFromFile(GAME_MAP_PATH);
    EXPECT_EQ(v.size(),10);
    for (int i = 0; i < 10; i++) {
        EXPECT_TRUE(testRightCountOfBlank(v[i],std::pair<int,int>(20,32)));
    }
}
```

发现了下面的错误:

挖洞过程的循环应为`while (holeNum--)`

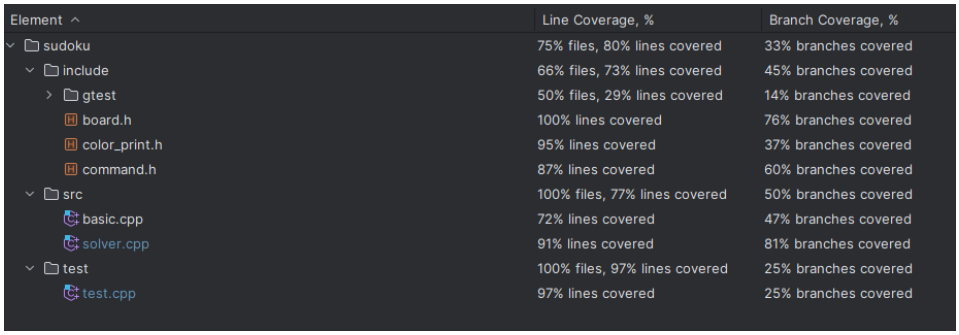
```
    for (auto &board: boards) {
        holeNum = dis(gen);
        while (--holeNum) {
            int x = rd0to8();
            int y = rd0to8();
            while (board[x][y] == 0) {
                x = rd0to8();
                y = rd0to8();
            }
            board[x][y] = 0;
        }
    }
```

单元测试通过记录

```
78  }
79  TEST(ExceptionTest, TestInvalidArgs) {...}
102 TEST(ExceptionTest, TestValidArgs) {...}
136 TEST(SolutionMapTest, TestSolutionMap) {...}
144
145 TEST(GenerateMapTest, TestMapWriteAndRead) {...}
160
161 TEST(GenerateMapTest, TestMapBatchRead) {...}
174
175 TEST(GenerateGameTest, TestLevel) {...}
207
208 TEST(SolverTest, TestSolve) {...}
223
224 TEST(SolverTest, TestSolveAll){...}
241 TEST(VectorTest, TestVector) {...}
251 TEST(GenerateGameTest,TestUnique){...}
269 int main(int argc, char **argv)
270 {
271     ::testing::InitGoogleTest(&argc, argv);
272     return RUN_ALL_TESTS();
273 }
```

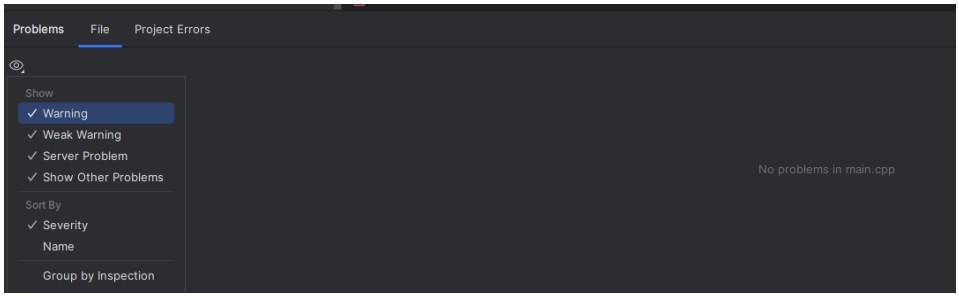
覆盖率测试

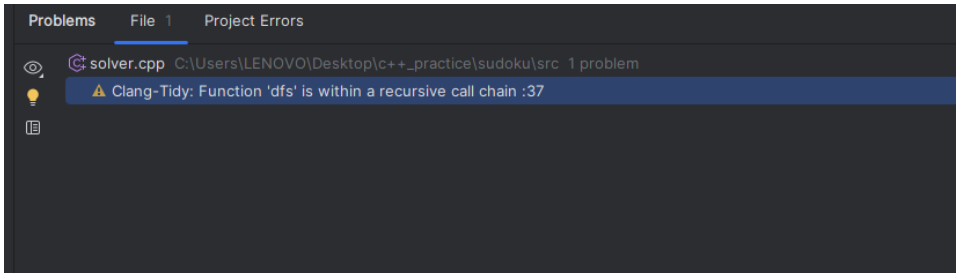
对`test.cpp`运行所有测试用例，得到结果如下图所示。



质量分析

代码除`dfs`递归调用外通过了所有的Clang-Tidy代码检查，消除警告，保证代码质量。




Problems
File 1
Project Errors
solver.cpp C:\Users\LENOVO\Desktop\c++_practice\sudoku\src 1 problem
Clang-Tidy: Function 'dfs' is within a recursive call chain :37