软件工程 结对编程作业

一、成员信息与分工

- 周延霖 2013921
  - 制定项目需求
  - 完成数独程序的命令行参数处理，生成数独终局功能，生成数独游戏功能。
- 林筠皓 2011283
  - 完成数独程序的求解数独游戏功能，生成唯一解的数独游戏功能。
  - 进行代码质量测试及单元测试，完成简易用户手册。

二、简易用户手册

（一）生成数独终局

命令说明

- 命令：`shudu.exe -c <num>`，表示生成`num`个数独终盘
- 范围限制：`num`的范围应该[1, 1000000]之间。
- 输入文件：无
- 输出文件：生成的数独终盘会记录在当前目录下的`end_game.txt`中。若此前没有该文件则会自动生成。

示例

生成2个数独终盘

- 命令：`shudu.exe -c 2`
- 输出文件`end_game.txt`：

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

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

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

（二）生成数独游戏

命令说明

- 命令：`shudu.exe -n <num> [options]`
  - 说明：表示根据`end_game.txt`中的终盘生成`num`个数独游戏
  - 可选参数`options`：可以不给出，也可以为以下的一种（不能同时出现两种及以上）：
    - `-m <difficulty>`
      - 说明：表示生成的数独游戏的难度为`difficulty`。
      - 范围限制：`difficulty`的范围应该[1, 3]之间。
        - `difficulty = 1`: 生成的数独游戏挖空数在20到31之间;
        - `difficulty = 2`: 生成的数独游戏挖空数在32到47之间;
        - `difficulty = 3`: 生成的数独游戏挖空数在48到64之间;
    - `-r <min-max>`
      - 说明：表示生成的数独游戏挖空数在`min`到`max`之间。
      - 范围显示：[`min`, `max`] ∈ [20, 55]
    - `-u`
      - 说明：表示生成的数独游戏具有唯一解。
- 输入文件：`end_game.txt`，程序会自动在当前目录下搜寻并读取，不必指定其路径
- 输出文件：生成的数独游戏会记录在当前目录下的`game.txt`中。若此前没有该文件则会自动生成。

示例

生成2个难度为1的数独游戏

- 命令：`shudu.exe -n 2 -m 1`
- 输出文件`game.txt`：

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

$ 1 2 9 8 7 $ 3 6
9 $ $ $ 3 $ 5 1 2
```

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

（三）求解数独游戏

命令说明

- 命令：`shudu.exe -s <path>`
- 说明
  - 表示从`path`中读取若干个数独游戏，并给出其解答；
  - 若数独游戏有多个解，最多只输出其中的3种。
- 输入文件：`path`
- 输出文件：解出的数独会记录在当前目录下的`sudoku.txt`中。若此前没有该文件则会自动生成。

示例

从`game.txt`读取若干个数独游戏，并给出其解答，生成到`sudoku.txt`中。

- 命令：`shudu.exe -s game.txt`
- 输出文件`sudoku.txt`：

```
Possible solution to Sudoku #1:
5 1 2 9 8 7 4 3 6
9 8 7 4 3 6 5 1 2
4 3 6 5 1 2 9 8 7
1 2 9 8 7 4 3 6 5
8 7 4 3 6 5 1 2 9
3 6 5 1 2 9 8 7 4
2 9 8 7 4 3 6 5 1
7 4 3 6 5 1 2 9 8
6 5 1 2 9 8 7 4 3

Possible solution to Sudoku #2:
5 1 2 9 8 7 4 3 6
9 8 7 4 3 6 5 1 2
4 3 6 5 1 2 9 8 7
1 2 9 8 7 4 3 6 5
8 7 4 3 6 5 1 2 9
3 6 5 1 2 9 8 7 4
2 9 8 7 4 3 6 5 1
6 5 1 2 9 8 7 4 3
7 4 3 6 5 1 2 9 8
```

三、代码质量分析

（一）代码规范检查

使用`cpplint`进行代码规范检查，检查代码是否符合Google C++编码规范。

在修改代码前存在的编码规范问题

在未仔细修改代码前，通过`cpplint`检查当前代码存在的问题：

```
D:\学习资料\22-23-B\软件工程\hw5\sudoku>cpplint shudu.cpp
shudu.cpp:0:  No copyright message found.  You should have a line: "Copyright [year] <Copyright Owner>"  [legal/copyright] [5]
shudu.cpp:14:  At least two spaces is best between code and comments  [whitespace/comments] [2]
shudu.cpp:15:  At least two spaces is best between code and comments  [whitespace/comments] [2]
shudu.cpp:17:  At least two spaces is best between code and comments  [whitespace/comments] [2]
shudu.cpp:23:  Lines should be <= 80 characters long  [whitespace/line_length] [2]
shudu.cpp:23:  At least two spaces is best between code and comments  [whitespace/comments] [2]
shudu.cpp:25:  Do not use namespace using-directives.  Use using-declarations instead.  [build/namespaces] [5]
shudu.cpp:28:  Is this a non-const reference? If so, make const or use a pointer: int& line  [runtime/references] [2]
shudu.cpp:48:  Lines should be <= 80 characters long  [whitespace/line_length] [2]
shudu.cpp:76:  Lines should be <= 80 characters long  [whitespace/line_length] [2]
shudu.cpp:76:  Is this a non-const reference? If so, make const or use a pointer: int& line  [runtime/references] [2]
shudu.cpp:77:  At least two spaces is best between code and comments  [whitespace/comments] [2]
shudu.cpp:81:  At least two spaces is best between code and comments  [whitespace/comments] [2]
shudu.cpp:160:  Lines should be <= 80 characters long  [whitespace/line_length] [2]
shudu.cpp:161:  Lines should be <= 80 characters long  [whitespace/line_length] [2]
shudu.cpp:213:  An else should appear on the same line as the preceding }  [whitespace/newline] [4]
shudu.cpp:213:  If an else has a brace on one side, it should have it on both  [readability/braces] [5]
shudu.cpp:225:  An else should appear on the same line as the preceding }  [whitespace/newline] [4]
shudu.cpp:225:  If an else has a brace on one side, it should have it on both  [readability/braces] [5]
shudu.cpp:241:  An else should appear on the same line as the preceding }  [whitespace/newline] [4]
shudu.cpp:241:  If an else has a brace on one side, it should have it on both  [readability/braces] [5]
shudu.cpp:271:  Lines should be <= 80 characters long  [whitespace/line_length] [2]
shudu.cpp:271:  Is this a non-const reference? If so, make const or use a pointer: int& row  [runtime/references] [2]
shudu.cpp:271:  Is this a non-const reference? If so, make const or use a pointer: int& col  [runtime/references] [2]
shudu.cpp:282:  Lines should be <= 80 characters long  [whitespace/line_length] [2]
shudu.cpp:311:  Lines should be <= 80 characters long  [whitespace/line_length] [2]
shudu.cpp:311:  Is this a non-const reference? If so, make const or use a pointer: vector<vector<int>>& grid  [runtime/references] [2]
shudu.cpp:311:  Is this a non-const reference? If so, make const or use a pointer: int& hasSolution  [runtime/references] [2]
shudu.cpp:431:  Lines should be <= 80 characters long  [whitespace/line_length] [2]
shudu.cpp:446:  Lines should be <= 80 characters long  [whitespace/line_length] [2]
shudu.cpp:461:  Line ends in whitespace.  Consider deleting these extra spaces.  [whitespace/end_of_line] [4]
shudu.cpp:465:  Lines should be <= 80 characters long  [whitespace/line_length] [2]
shudu.cpp:485:  Lines should be <= 80 characters long  [whitespace/line_length] [2]
shudu.cpp:489:  Lines should be <= 80 characters long  [whitespace/line_length] [2]
shudu.cpp:492:  Lines should be <= 80 characters long  [whitespace/line_length] [2]
shudu.cpp:496:  Lines should be <= 80 characters long  [whitespace/line_length] [2]
shudu.cpp:500:  Could not find a newline character at the end of the file.  [whitespace/ending_newline] [5]
Done processing shudu.cpp
Total errors found: 37
```

主要的问题包括：

- 缺少版权信息：没有包含版权信息的声明。
- 注释与代码之间的空格：最好在注释和代码之间留出至少两个空格。
- 行长度超过80个字符：某些行的长度超过了推荐的80个字符。
- 不应使用namespace using-directives：应改用using-declarations来引入命名空间。
- 非const引用参数：应该将非const引用参数改为const或使用指针。
- else语句的位置：else语句应该与前面的}在同一行，并且两边都应该使用大括号。
- 行末尾空格：行末尾存在多余的空格，建议删除。
- 文件结尾缺少换行符：文件结尾处缺少换行符。

编写符合编码规范的代码

将上述问题更正后，再重新使用`cpplint`进行代码规范检查：

```
D:\学习资料\22-23-B\软件工程\hw5\sudoku>cpplint shudu.cpp
Done processing shudu.cpp
```

（二）性能分析

1. 内存泄漏检查

检查生成数独终局是否有内存泄漏：无可能的内存泄漏，安全。

```
jacelin@master:~/project/sf$ valgrind --leak-check=full ./shudu -c 100
==6985== Memcheck, a memory error detector
==6985== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==6985== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==6985== Command: ./shudu -c 100
==6985==
==6985==
==6985== HEAP SUMMARY:
==6985==     in use at exit: 0 bytes in 0 blocks
==6985==   total heap usage: 106 allocs, 106 frees, 101,692 bytes allocated
==6985==
==6985== All heap blocks were freed -- no leaks are possible
==6985==
==6985== For lists of detected and suppressed errors, rerun with: -s
==6985== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

检查生成数独游戏是否有内存泄漏：无可能的内存泄漏，安全。

```
jacelin@master:~/project/sf$ valgrind --leak-check=full ./shudu -n 100
==6986== Memcheck, a memory error detector
==6986== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==6986== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==6986== Command: ./shudu -n 100
==6986==
==6986==
==6986== HEAP SUMMARY:
==6986==     in use at exit: 0 bytes in 0 blocks
==6986==   total heap usage: 8 allocs, 8 frees, 90,256 bytes allocated
==6986==
==6986== All heap blocks were freed -- no leaks are possible
==6986==
==6986== For lists of detected and suppressed errors, rerun with: -s
==6986== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

检查求解数独游戏时是否有内存泄漏：无可能的内存泄漏，安全。

```
jacelin@master:~/project/sf$ valgrind --leak-check=full ./shudu -s game.txt
==8710== Memcheck, a memory error detector
==8710== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==8710== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==8710== Command: ./shudu -s game.txt
==8710==
==8710==
==8710== HEAP SUMMARY:
==8710==     in use at exit: 0 bytes in 0 blocks
==8710==   total heap usage: 42 allocs, 42 frees, 91,655 bytes allocated
==8710==
==8710== All heap blocks were freed -- no leaks are possible
==8710==
==8710== For lists of detected and suppressed errors, rerun with: -s
==8710== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

2. 代码复杂度

生成数独游戏的复杂度

求解数独游戏的核心函数是`GenerateNewGame`。

时间复杂度：

- `GenerateNewGame`函数的时间复杂度主要取决于生成游戏的循环和写入游戏文件的循环。两个循环的迭代次数取决于参数`num`，即要生成的游戏数量。
- 在生成游戏的循环中，有一个嵌套的循环用于读取终局，其时间复杂度为 O(1)，因为终局的大小是固定的。
- 在生成游戏的循环中，还有一个条件语句和一系列的 switch 语句。条件语句和 switch 语句的复杂度可以视为常数级别，不会随输入规模变化而变化。
- 在写入游戏文件的循环中，有两个嵌套的循环用于遍历游戏棋盘，其时间复杂度为 O(1)，因为棋盘的大小是固定的。
- 因此，总的时间复杂度为 O(num)，即线性复杂度。

空间复杂度：

- 代码中使用了一个二维字符数组`thisGame`用于存储生成的游戏棋盘，因此它的空间复杂度为 O(1)，即固定大小的二维数组。
- 另外，代码中打开了输入和输出文件流，并且使用了一些局部变量。这些额外的空间消耗可以忽略不计，因为它们的空间复杂度与输入规模无关。
- 综上所述，总的空间复杂度为 O(1)。

求解数独游戏的复杂度

求解数独游戏的核心函数是`SolveSudoku`。

其核心代码为：

```
bool SolveSudoku(std::vector<std::vector<int>>& grid, int& hasSolution,  //
NOLINT
                 std::ofstream& outputFile, int id = 0) {
  int row, col;
  if (!FindEmptyCell(grid, row, col)) {
    return true;
  }
  for (int num = 1; num <= GRID_SIZE; ++num) {
    if (IsSafe(grid, row, col, num)) {
      grid[row][col] = num;
      if (SolveSudoku(grid, hasSolution, outputFile, id)) {
        hasSolution++;
        if (hasSolution > 3) {
          return false;
        }
        /* 写入当前解到输出文件... */
        grid[row][col] = 0;
      } else {
        grid[row][col] = 0;
```

```
      }
    }
  }
  return false;
}
```

时间复杂度:

最坏情况下，数独问题可能有多个解，因此我们需要找到所有可能的解。如果假设每个空单元格都有 `GRID_SIZE` 个选择，那么回溯过程的次数最多为 `GRID_SIZE` 的指数级别，即 O(`GRID_SIZE`^空单元格数)。在每次回溯操作中，我们需要检查当前数字的放置是否合法，这涉及到对行、列和 3x3 方框进行检查。由于数独的大小为 `GRID_SIZE x GRID_SIZE`，因此这一步的时间复杂度为 O(`GRID_SIZE`)。

因此，总的时间复杂度为 O(`GRID_SIZE`^空单元格数)。

空间复杂度:

代码中使用了一个二维向量 `grid` 来表示数独的布局，因此它的空间复杂度为 O(`GRID_SIZE`^2)。在递归调用 `SolveSudoku` 函数时，会在堆栈上存储每次递归调用的局部变量和参数。最大递归深度取决于数独的空单元格数，因此空间复杂度为 O(空单元格数)。

综上所述，总的空间复杂度为 O(`GRID_SIZE`^2 + 空单元格数)。

（三）错误与异常处理

命令行参数处理

- 未定义的参数选项
  - 报错 `Please re-run this program and enter the correct parameters!`，并且 `main` 函数应返回1表示异常。
- 不合规的参数数量
  - 报错 `Please re-run this program and enter the parameters!`，并且 `main` 函数应返回1表示异常。
- 不合规的参数值范围
  - 报错 `The range of 'xxx' should be within xxx.`，并且 `main` 函数应返回1表示异常。
- 不合规的参数排列方式
  - 报错 `Should not specify both xxx and xxx`，并且 `main` 函数应返回1表示异常。

生成数独终局

- 若捕获到以下异常，则输出对应异常信息并强制返回
  - `std::ofstream::failure`
  - `std::bad_alloc`

生成数独游戏

- 若捕获到以下异常，则输出对应异常信息并强制返回
  - `std::ifstream::failure`，该异常往往由不存在终局文件`end_game.txt`导致。
  - `std::bad_alloc`

求解数独游戏

- 限制求解的迭代深度，防止栈溢出或运行时间过长
  - 考虑到一些空格数量较大的数独游戏有过多种解法，若是要得到所有解法则会大大增加求运行时间，故限制最多求出3个解就强制停止求解。
- 若在读入和输出文件时捕获到以下异常，则输出对应异常信息并强制返回
  - `std::ifstream::failure`
  - `std::ofstream::failure`
  - `std::bad_alloc`

四、单元测试

（一）测试用例设计思路

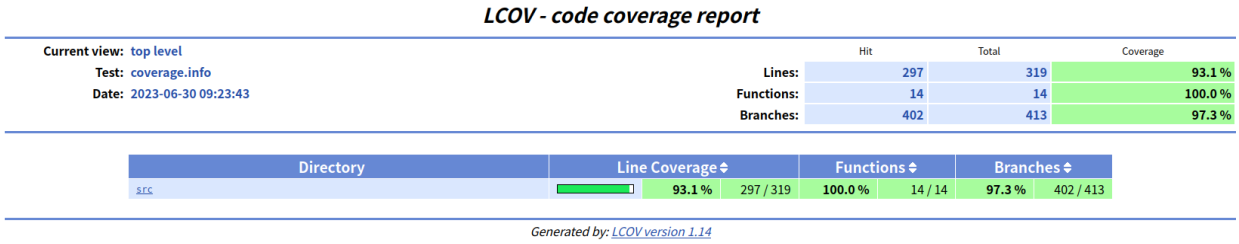
- 命令行参数相关
  - 当输入的参数不符合规范时，`main`函数的返回值应该是1
    - 不合规的参数数量
    - 不合规的参数值范围
    - 不合规的参数排列方式
    - 未定义的参数选项
  - 当输入的参数合法时，`main`函数的返回值应该是0
- 生成终局文件
  - 检查生成的终局文件是否符合数独的规则
    - 每一行包含1到9的所有数字，且没有重复。
    - 每一列包含1到9的所有数字，且没有重复。
    - 每个3x3的小方块中包含1到9的所有数字，且没有重复。
  - 检查生成的数独终局是否有重复的
- 读取棋盘文件并求解
  - 检查求解结果是否符合数独的规则
- 生成数独游戏
  - 检查生成的数独游戏文件是否和输入的终局文件是否是一一对应的
  - 生成指定难度的游戏
    - 枚举所有难度进行生成
  - 生成指定空格数量范围的游戏

        - 指定几组空格数量范围进行生成
    - 生成唯一解的游戏
        - 还需使用求解函数检查生成的数独游戏文件是否只有唯一解

（二）覆盖率与测试通过率

测试通过率为100%。

如下图所示，平均代码覆盖率为96%，其中语句覆盖率为93.1%，函数覆盖率为100%，分支覆盖率为97.3%。



LCOV - code coverage report

Current view:	top level		Hit	Total	Coverage
Test:	coverage.info	**Lines:**	297	319	**93.1 %**
Date:	2023-06-30 09:23:43	**Functions:**	14	14	**100.0 %**
		Branches:	402	413	**97.3 %**

Directory	Line Coverage		Functions		Branches	
src		**93.1 %**	297 / 319	**100.0 %**	14 / 14	**97.3 %**	402 / 413

Generated by: LCOV version 1.14