

# 数独控制台程序

---

- 姓名：管昀玫、石家琪
- 学号：2013750、2011739
- 专业：计算机科学与技术

前言：

在我们的程序中，已实现所有6个要求的功能，包括：

- 生成数独游戏
- 求解数独游戏
- 不同难度，保证唯一解，限制挖空范围

除此之外，我们的程序允许参数乱序，即不固定 `-n/-m/-u` 或 `-n/-r/-u` 的相对位置，为参数的输入提供了便利。

项目的GitHub链接为：<https://github.com/civilizwa/shudu/tree/master>

## 数独控制台程序

代码简略说明

输入部分

数独生成部分

数独求解部分

用户手册

1 程序简介

2 安装与运行

3 界面介绍

4 使用指南

4.1 生成数独终盘

4.2 读取游戏并给出解答

4.3 批量生成数独游戏

4.4 生成游戏的难度

4.5 控制挖空数量范围

4.6 生成唯一解游戏

4.7 参数乱序与多个参数

4.8 错误与恢复

5 保存与加载

质量分析

1 消除警告

2 静态代码分析

3 代码规范检查

单元测试

1 定义输入

2 测试用例设计

覆盖率报告

性能分析

1 性能探查器

2 VTune Profiler

总结

## 代码简略说明

---

我们的代码主要分为三个部分：

- GenerateHandler: 数独生成
- InputHandler: 输入参数控制
- SolverHandler: 数独求解

## 输入部分

该部分的功能由 `InputHandler` 类来完成，主要用于输入的参数识别，`check` 函数是它的主体。

```
class InputHandler {
public:
    void check(int argc, char** argv);
    void getFinal(int num);
    // 设置绝对路径
    void setAbsPath(string abs) {
        absolutePath = abs;
    }
    GenerateHandler generator; // 定义数独题目生成器
    SolverHandler board;
    int GetNum();
    char GetType1();
    char GetType2();
    char GetType3();
    int GetRange1();
    int GetRange2();
    int GetLevel();

private:
    int isNum(const string& s);
    string absolutePath = "D:\\LessonProjects\\shudu\\shudu";
    string FinalPath = "final.txt"; // 存储终局
    string AnsPath = "ans.txt"; // 存储求解答案
    string QuexPath = "question.txt"; // 存储数独题目
    char type1 = 'y';
    char type2 = 'y';
    char type3 = 'y';
    int num = 20;
    int range1 = 20;
    int range2 = 55;
    int level = 1;
};
```

`check` 函数负责识别各个函数，并进行相应的函数调用。

```

void InputHandler::check(int argc, char** argv) {
    // 只有一个参数:
    // -c num:生成num个终盘
    // -s game.txt:从game.txt读取若干数独游戏, 并给出解答, 存储到shudu.txt中
    // -n num:生成num个数独游戏, 存储到game.txt中
    // -r 挖空: a-b
    generator.setAbsPath(absolutePath);
    if (argc == 3) {
        string parameter1 = argv[1];
        string parameter2 = argv[2];
        if (parameter1 == "-c") { // done
            int n = isNum(parameter2);
            if (n <= 0 || n > 1000000) {
                cout << "不满足0<n<=1000000!" << endl;
            }
            else {
                // FianlMaker fm;
                // fm.make(n);
                type1 = 'c';
                num = n;
                getFinal(n);
                cout << "已生成" << parameter2 << "个数独终盘" << endl;
            }
        }
        else if (parameter1 == "-s") {
            fstream infile(absolutePath + parameter2, ios::in);
            fstream outfile(absolutePath + AnsPath, ios::out);
            type1 = 's';
            if (!infile.is_open()) {
                cout << "文件打开失败!" << endl;
                return;
            }
            cout << "正在求解, 请稍候..." << endl;
            int i = 1;
            while (infile.peek() != EOF) {
                board.input(infile);
                int result = board.solve();
                if (result == 0) {
                    board.output(outfile);
                }
                else {
                    cout << "第" << i << "个数独无解!" << endl;
                }
            }
        }
    }
}

```

由于逻辑大致相同, 此处不再赘述。

## 数独生成部分

主要数独生成算法的思路来源于<https://www.jianshu.com/p/4b0d08e19e93>

i	g	h	c	a	b	f	d	e
c	a	b	f	d	e	i	g	h
f	d	e	i	g	h	c	a	b
g	h	i	a	b	c	d	e	f
a	b	c	d	e	f	g	h	i
d	e	f	g	h	i	a	b	c
h	i	g	b	c	a	e	f	d
b	c	a	e	f	d	h	i	g
e	f	d	h	i	g	b	c	a

步骤2中首先生成中间的方格，然后按照如下图对应变换行、列获得其它位置的方格，如图可见生成的结果符合数独要求。周围的小方格都可以由中间的方格交换行列变换而来。

步骤3中行列交换是一整行一整列（9个）元素的交换，为了使得交换后的数独仍然是合法的，我们限定交换只发生组内，这样可以证明，交换涉及的行/列和3\*3块都仍然是合法的。非严格证明：经过步骤3交换后生成的数独不重复：

1. 如果是由同一个步骤2中生成的数独变换而成，则显然被交换的行/列分别和原本不一样，又因为只交换两行/列，所以双方对应行/列也相互区别。
2. 如果是由不同步骤2中生成的数独变换而成，显然每次至少有三个（一行/一列）方格不一样，对应的是双方都没有交换的那一组行/列方格。

数独生成部分的函数主要在 `GenerateHandler` 类中。

```

class Generatehandler {
private:
    string FinalPath = "final.txt";
    string outputPath = "question.txt";
    string absolutePath = "D:\\LessonProjects\\shudu\\shudu";
    int FinalNum = 0; // 终局数目
    int current_HoleNum = 0; // 当前挖空数
    int current_selectFinal = 0; // 当前使用的棋盘终局
    int matrix[100][9][9] = { 0 }; // 开辟一个大数组，存储最多100个终局
    int holeboard[9][9] = { 0 }; // 当前挖空的棋盘位置

public:
    bool generate(int num, int beginNum, int endNum, bool isUnion);
    void holehole(); // 挖呀挖呀挖
    void input(fstream& f);
    void output(fstream& f, vector<std::vector<int>>& board);
    void SelectFinal();
    int isNum(const string &s);
    // 检查在给定位置 (row, col) 是否可以放置数字 num
    bool isValid(const vector<std::vector<int>>& board, int row, int col, int num);
    // 使用回溯算法生成数独游戏的唯一解
    bool solveSudoku(vector<std::vector<int>>& board);
    // 生成数独游戏题目
    void generateSudoku(vector<std::vector<int>>& board);
    // 生成在min-max范围间的随机数
    int generateRandomNumber(int min, int max);
    // 设置绝对路径
    void setAbsPath(string abs) {
        absolutePath = abs;
    }
};

```

重要的函数思路如下所示：

```

bool Generatehandler::generate(int num, int beginNum, int endNum, bool isUnion)
{
    /*思路：从beginNum开始挖空，
    如果需要判断union：
    判断不成立后，重新迭代100次，
    超过100次则挖空数+1后重新生成
    直至endNum*/
    fstream infile(absolutePath+FinalPath, ios::in);
    fstream outfile(absolutePath+outputPath, ios::out);
    if (!infile.is_open()) {
        cout << "未找到输入文件的路径!" << endl;
    }
    if (!outfile.is_open()) {
        cout << "文件打开失败!" << endl;
        return false;
    }
    cout << "-----正在生成" << num << "个数独题目-----" << endl;
    input(infile); // 将终局读入matrix数组
    infile.close();
    if (isUnion == false) {
        for (int i = 0; i < num; i++) {
            // cout << "generating" << endl;
            current_HoleNum = generateRandomNumber(beginNum, endNum); // 在范围内
            // 随机生成挖空个数
            holehole(); // 挖current_HoleNum个洞
            SelectFinal(); // 随机挑选一个终局
            for (int row = 0; row < 9; row++) { // 将终局挖空后的结果输出到文件

```

```

        if (holeboard[row][0] == 1) {
            outfile << " $";
        } else {
            outfile << " " << matrix[current_selectFinal][row][0];
        }
        for (int col = 1; col < 9; col++) {
            if (holeboard[row][col] == 1) {
                outfile << " $";
            } else {
                outfile << " " << matrix[current_selectFinal][row][col];
            }
        }
        outfile << endl;
    }
    outfile << endl;
}
outfile.close();
cout << "生成完成!" << endl;
} else { // 使用回溯法生成唯一解的数独
    vector<std::vector<int>>> board;
    for (int t = 0; t < num; t++) { // 在范围内随机生成挖空个数
        // cout << "generating" << endl;
        current_HoleNum = generateRandomNumber(beginNum, endNum);
        generateSudoku(board);
        holehole();
        output(outfile, board);
    }
    outfile.close();
}
cout << "生成完成!";
return true;
}

int Generatehandler::generateRandomNumber(int min, int max) {
    // 设置随机数引擎和分布
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_int_distribution<int> dis(min, max);

    // 生成随机数
    int randomNumber = dis(gen);
    return randomNumber;
}

void Generatehandler::generateSudoku(std::vector<std::vector<int>>& board) {
    std::srand(static_cast<unsigned int>(std::time(0))); // 设置随机数种子
    // 清空数独游戏
    board.clear();
    board.resize(9, std::vector<int>(9, 0));
    // 随机填充第一行
    for (int col = 0; col < 9; ++col) {
        board[0][col] = col + 1;
    }
    // 混洗第一行的数字
    std::random_shuffle(board[0].begin(), board[0].end());
    // 生成唯一解的数独游戏
    solveSudoku(board);
}

```

```

void GenerateHandler::holehole() {
    // 置零
    for (int i = 0; i < 9; i++) {
        for (int j = 0; j < 9; j++) {
            holeboard[i][j] = 0;
        }
    }
    // 先挖掉，每行挖两个，再从剩余的空中挖
    for (int i = 0; i < 9; i++) {
        // 设置种子，确保每次运行生成的随机数序列不同
        std::srand(static_cast<unsigned int>(std::time(nullptr)));
        // 生成1到9之间的随机数
        int hole1 = generateRandomNumber(0, 8);
        int hole2 = generateRandomNumber(0, 8);
        while (hole1 == hole2) {
            hole2 = generateRandomNumber(0, 8);
        }
        // 将这两个位置设置为挖去状态
        holeboard[i][hole1] = 1;
        holeboard[i][hole2] = 1;
    }

    int restHole = current_HoleNum - 18;
    while (restHole-->0) {
        int row = generateRandomNumber(0, 8);
        int col = generateRandomNumber(0, 8);
        while (holeboard[row][col] == 1) {
            row = generateRandomNumber(0, 8);
            col = generateRandomNumber(0, 8);
        }
        holeboard[row][col] = 1;
    }
    return;
}

```

## 数独求解部分

暴力搜索-->剪枝-->启发式搜索

考虑代码的可理解性和可维护性，参考了[这篇博客](#)以及[这篇博客](#)中的回溯和DFS思路，决定采用回溯法求数独。将解空间定义为所有解的空格的所有0-9的所有组合。依次尝试每个空格的1-9的所有取值，看能否生成合法的数独，这样算法的复杂度为 $n^9$ ，n为数独中的空格数目。显然改算式仍然有很大的改进空间，有待后续优化。

优化：参考上述博客，在填空进入下一层搜索前，先判断是否是合法数独，如果不合法直接跳过，这样可以有效剪枝。但是需要使得判断数独合法算法能够处理数独不完整时的情况。

再进一步的优化为：记录当前行号，尝试数字时快速跳过本行已有的数字；按照数字在空格中出现的比例来顺序尝试，但是需要额外的数据结构和计算比例。

由于时间关系，优化尚未实现，今后可以进一步完善。

`SolveHandler` 类定义如下所示：

```

struct SolverHandler {
public:
    int matrix[9][9] = { 0 };
    stack<pair<int, int>> blank; // blank coordinates
    int row[9] = { 0 };
    int col[9] = { 0 };
    int patch[9] = { 0 };
    void insert(int i, int j, int num);
    void remove(int i, int j);
    void replace(int i, int j, int newNum);
    int getMask(int i, int j);
    void input(fstream& f);
    void output(fstream& f);
    int solve();
    void clean();
};

```

重要函数如下所示:

```

void SolverHandler::insert(int i, int j, int num) {
    matrix[i][j] = num;
    if (num == '$') {
        blank.push(make_pair(i, j));
    } else {
        row[i] |= (1 << num);
        col[j] |= (1 << num);
        patch[(i / 3) * 3 + j / 3] |= (1 << num);
    }
}

void SolverHandler::remove(int i, int j) {
    int num = matrix[i][j];
    if (num != '$') {
        // matrix[i][j] = (int) '$';
        matrix[i][j] = static_cast<int>('$');
        row[i] ^= (1 << num);
        col[j] ^= (1 << num);
        patch[(i / 3) * 3 + j / 3] ^= (1 << num);
    }
    blank.push(make_pair(i, j));
}

void SolverHandler::replace(int i, int j, int newNum) {
    int oldNum = matrix[i][j];
    if (oldNum == '$') {
        insert(i, j, newNum);
    } else {
        matrix[i][j] = newNum;
        int mask = (1 << oldNum) | (1 << newNum);
        row[i] ^= mask;
        col[j] ^= mask;
        patch[(i / 3) * 3 + j / 3] ^= mask;
    }
}

int SolverHandler::getMask(int i, int j) {
    int mask = 0;
    mask |= row[i];
    mask |= col[j];
}

```



```

        mask |= patch[(i / 3) * 3 + j / 3];
        return mask;
    }

    void SolverHandler::clean() {
        // 清空数组
        // 清空row、col、patch
        for (int i = 0; i < 9; i++) {
            row[i] = 0;
            col[i] = 0;
            patch[i] = 0;
            for (int j = 0; j < 9; j++) {
                matrix[i][j] = 0;
            }
        }
        // 清空栈
        while (!blank.empty()) {
            blank.pop();
        }
    }

    int SolverHandler::solve() {
        if (blank.empty()) {
            // No blank positions, success
            return 0;
        }
        pair<int, int> coord = blank.top();
        blank.pop();
        int mask = getMask(coord.first, coord.second);
        for (int i = 1; i <= 9; i++) {
            if ((mask & (1 << i)) == 0) {
                // Not used, search down
                replace(coord.first, coord.second, i);
                int result = solve();
                if (result == 0) {
                    return 0;
                }
            }
        }
        // coords pushed back here
        remove(coord.first, coord.second);
        return -1;
    }
}

```

## 用户手册

### 1 程序简介

欢迎使用我们的数独控制台程序！该程序为数独爱好者提供了多项实用功能，包括生成数独终盘、求解数独、批量生成数独游戏以及灵活的游戏定制选项。

通过生成数独终盘功能，您可以获得一个全新的数独谜题的终盘，即完整的已填数字的数独板。这个终盘可以作为一个随机数独游戏的基础，让您在每次游戏时都能体验到不同的挑战和解谜乐趣。

同时，该程序也提供了求解数独的功能。如果您遇到难题或想验证自己的解答是否正确，只需将数独谜题输入程序，它 will 为您快速求解并给出答案。

除此之外，我们的控制台程序还支持批量生成数独游戏，让您一次性获得多个数独谜题。您可以指定生成游戏的难度级别，选择简单、中等或困难的谜题，以适应不同的游戏水平和挑战需求。

定制化选项也是我们程序的亮点之一。您可以指定生成游戏中挖空的数量范围，从较少的空格数到更多的空格数，根据自己的喜好和难度偏好定制游戏体验。此外，您还可以选择生成游戏时的解唯一要求，确保每个数独谜题都有唯一的解决方案。

我们希望通过使用我们的数独控制台程序，您可以尽情享受数独的乐趣，锻炼逻辑思维和推理能力。如果您有任何问题或需要支持，请随时与我们联系。祝您玩得愉快并挑战成功！

## 2 安装与运行

1. 下载程序：请前往我们的[GitHub](#)，获取数独控制台程序的exe文件。
2. 解压文件：将下载的安装文件解压到您希望安装程序的目录。
3. 打开控制台：打开操作系统的命令提示符（Windows系统为命令提示符，Linux和macOS系统为终端）。
4. 切换目录：使用命令提示符（或终端）中的 `cd` 命令，切换到数独控制台程序的exe所在目录。默认情况下，目录为：`shudu\x64\Release`
5. 运行程序：在命令提示符（或终端）中输入程序的可执行文件名称，并指定参数信息，按下回车键运行程序。
6. 指定文件位置参数：在运行程序时，使用命令行参数指定数独谜题文件的位置。根据程序的要求，可能需要输入文件的完整路径或相对路径。请根据程序的使用说明提供正确的文件位置参数。

如果您在安装和运行过程中遇到任何问题，请参阅程序的用户手册或联系我们的支持团队获取帮助。

祝您愉快地安装和运行数独控制台程序，并享受解谜的乐趣！

## 3 界面介绍

我们的数独控制台程序提供了简洁而直观的界面，使您能够轻松操作和享受数独游戏的乐趣。

1. 命令输入：命令输入是数独游戏中最重要的部分，具体参数详见后文介绍。通过命令输入，可以指定需要的数独终盘数量、需要解的数独棋盘文件路径、需要的游戏数量、生成的游戏难度、生成游戏的难度、唯一解等。
2. 提示信息：程序通常会在界面中显示一些操作提示，指导您如何与数独进行交互，如提示输入路径、输入错误警告、开始生成若干数独终盘等。请仔细阅读和遵循这些提示，以正确地操作数独游戏。
3. 结果显示：在求解数独或执行其他操作后，程序会在界面中显示相应的结果。这可能是数独的解答、生成的数独游戏、操作的成功或失败消息等。请注意仔细阅读结果显示，以获取所需的信息。

## 4 使用指南

在 `shudu.exe` 目录下打开命令行窗口，输入如下格式的命令：

```
shudu.exe [parameters]
```

注意，每一次运行，都要填入txt文件所在或生成的绝对路径，下面给出示例：

### 4.1 生成数独终盘

使用参数 `-c`，并指定数目，即可生成数独终盘

[para]	-c
mean	需要生成的数组终盘数量
range	1-1000000
example	shudu.exe -c 20 【生成20个数独终盘】

```
PS D:\LessonProjects\shudu\x64\Debug> .\shudu.exe -c 20
请输入存储文件的绝对路径:
D:\LessonProjects\shudu\
开始生成20个数独终盘!
生成完成!
已生成20个数独终盘
PS D:\LessonProjects\shudu\x64\Debug> █
```

4.2 读取游戏并给出解答

使用参数 `-s`，读取若干数独游戏并给出解答

[para]	-s
mean	需要解的数组棋盘文件路径
limitation	绝对或者相对路径
example	shudu.exe -s game.txt 【从game.txt读取若干个数独游戏，并给出解答，生成到 sudoku.txt中】

question.txt 示例：使用 \$ 符号代表待填的空



```
question.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
$ 5 8 4 9 $ 1 2 7
1 2 $ $ 6 7 4 8 $
4 7 9 1 2 $ $ 5 6
2 1 4 3 $ 6 $ 9 8
3 $ 5 $ 8 9 2 $ 4
8 9 7 $ $ 4 5 6 3
5 3 $ 8 4 1 $ 7 $
7 8 1 9 $ 2 6 4 $
$ $ 2 6 7 5 8 3 1

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

命令程序进行求解：

```
PS D:\LessonProjects\shudu\x64\Debug> .\shudu.exe -s question.txt
请输入存储文件的绝对路径:
D:\LessonProjects\shudu\shudu\
正在求解,请稍候...
完成求解!
PS D:\LessonProjects\shudu\x64\Debug>
```

求解完成后, 该目录下出现 `ans.txt`:

	ans.txt	2023/6/28 20:18	文本文档	2 KB
	final.txt	2023/6/28 12:05	文本文档	4 KB
	.	-----	-----	-----

`ans.txt` 内容为求解完成的数独解答:

```
ans.txt - 记事本
文件(F)  编辑(E)  格式(O)  查看(V)  帮助(H)
6 5 8 4 9 3 1 2 7
1 2 3 5 6 7 4 8 9
4 7 9 1 2 8 3 5 6
2 1 4 3 5 6 7 9 8
3 6 5 7 8 9 2 1 4
8 9 7 2 1 4 5 6 3
5 3 6 8 4 1 9 7 2
7 8 1 9 3 2 6 4 5
9 4 2 6 7 5 8 3 1

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

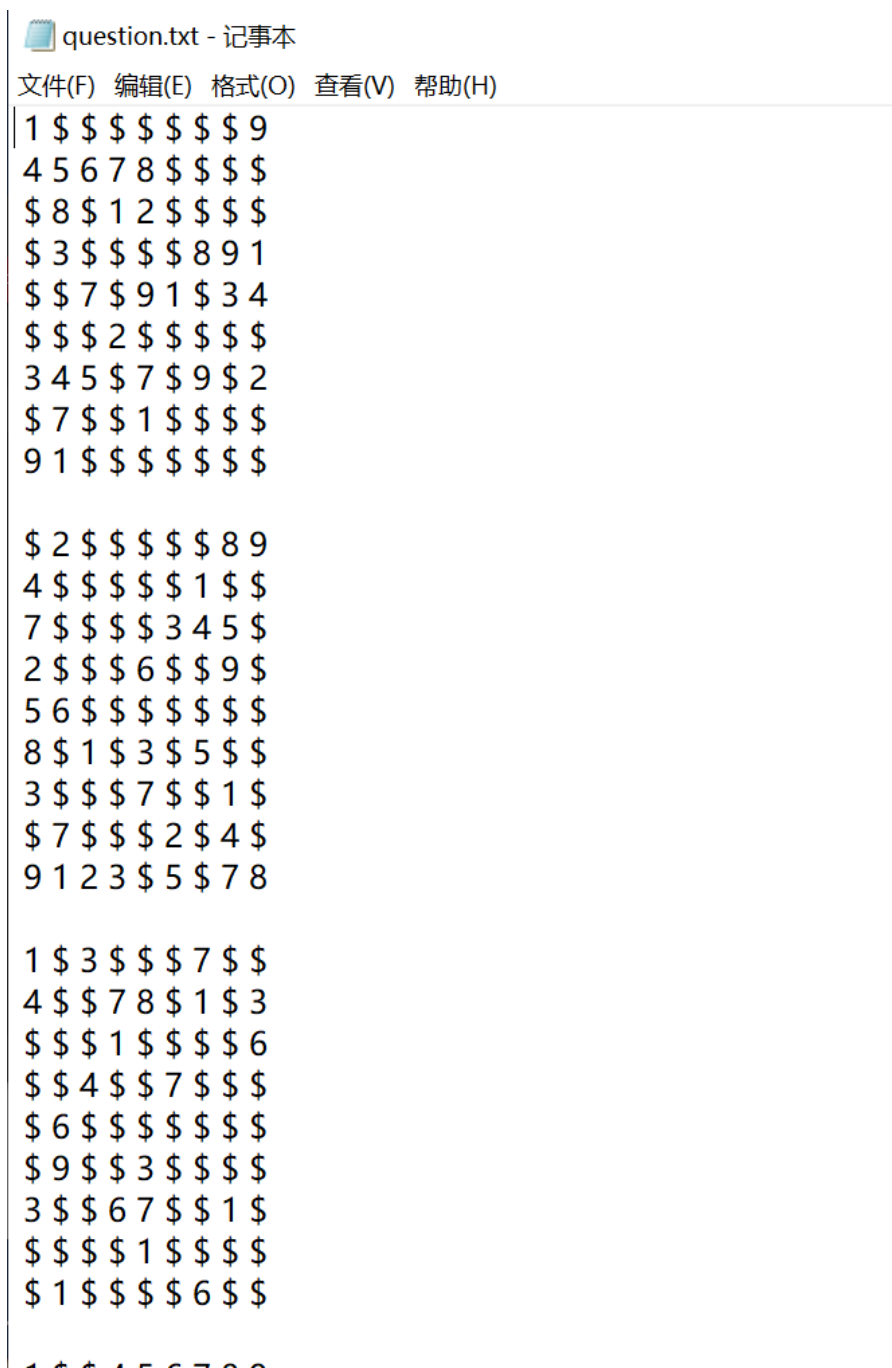
### 4.3 批量生成数独游戏

使用参数 `-n`, 并指定需要的游戏数量, 即可批量生成数独

[para]	-n
mean	需要的游戏数量
range	1-10000
example	shudu.exe -n 1000 【生成1000个数独游戏】

```
-aa87590fcd05be6b4feb0e6f5f05648daa195649\x64\Debug> .\shudu.exe -n 30
请输入存储文件的绝对路径:
C:\Users\civilizwa\Downloads\
-----正在生成30个数独题目-----
生成结束!
```

生成的 `question.txt` 如下所示, 以 `$` 来代替待填的数字:



#### 4.4 生成游戏的难度

使用参数 `-n` 和 `-m`，需要指定生成游戏的数量和难度，难度为数字1~3之间，数字越大代表越难

[para]	-m
mean	生成游戏的难度
range	1-3
example	shudu.exe -n 1000 -m 1 【生成1000个简单数独游戏，只有m和n一起使用时才认为参数无误，否则请报错】

尝试生成难度1：

```
-aa87590fcd05be6b4feb0e6f5f05648daa195649\x64\Debug> ./shudu.exe -n 10 -m 1
请输入存储文件的绝对路径：
C:\Users\civilizwa\Downloads\
level:1
-----正在生成10个数独题目-----
生成结束！
```

question.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

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

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

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

尝试生成难度2:

```
-aa87590fcd05be6b4feb0e6f5f05648daa195649\x64\Debug> ./shudu.exe -n 10 -m 2
请输入存储文件的绝对路径:
C:\Users\civilizwa\Downloads\
level:2
-----正在生成10个数独题目-----
生成结束!
```

尝试生成难度3:

```
-aa87590fcd05be6b4feb0e6f5f05648daa195649\x64\Debug> ./shudu.exe -n 20 -m 3
请输入存储文件的绝对路径:
C:\Users\civilizwa\Downloads\
level:3
-----正在生成20个数独题目-----
生成结束!
```

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

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

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

求解后的难度3为:

ans.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

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

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

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

## 4.5 控制挖空数量范围

同时使用参数 `-n` 和 `-r`，指定生成数独游戏的数量和挖空范围。注意，挖空范围的两个数应用 `-` 符号链接。

[para]	-r
mean	生成游戏中挖空的数量范围
range	20-55
example	shudu.exe -n 20 -r 20-55 【生成20个挖空数在20~55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】

```
PS D:\LessonProjects\shudu\x64\Debug> .\shudu.exe -n 2 -r 20-55
请输入存储文件的绝对路径:
D:\LessonProjects\shudu\
-----正在生成2个数独题目-----
生成完成! 已生成2个数独游戏, 挖空范围在[20, 55]之间
PS D:\LessonProjects\shudu\x64\Debug>
```





#### 4.6 生成唯一解游戏

同时使用参数 `-n` 与 `-u`，即可指定生成游戏的数量，且它们具有唯一解

[para]	-u
mean	生成游戏的解唯一
example	shudu.exe -n 20 -u 【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】

```
PS D:\LessonProjects\shudu\x64\Debug> .\shudu.exe -n 2 -u
请输入存储文件的绝对路径:
D:\LessonProjects\shudu\
-----正在生成2个数独题目-----
生成完成!
PS D:\LessonProjects\shudu\x64\Debug>
```

成功生成了 question.txt:

名称	上次写入	大小	类型
question.txt	2023/6/28 21:15	1 KB	文本文档
shudu.sln	2023/6/28 16:56	3 KB	Visual Studio Sol...

```

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

```

```

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

```

## 4.7 参数乱序与多个参数

在我们的程序中，参数的位置可以并不固定，且允许三个参数结合使用，例如：

- `\shudu.exe -r 20-30 -u -n 20`
- `\shudu.exe -n 20 -m 1 -u`

```

PS D:\LessonProjects\shudu> D:\LessonProjects\shudu\x64\Debug\shudu.exe -r 20-30 -u -n 20
请输入存储文件的绝对路径:
D:\LessonProjects\shudu\
-----正在生成20个数独题目-----
生成完成!已生成20个具有唯一解数独游戏,挖空范围在[20,30]之间
PS D:\LessonProjects\shudu>

```

## 4.8 错误与恢复

当错误输入参数时，程序会提示相应的错误，如下所示：

```

PS D:\LessonProjects\shudu\x64\Debug> .\shudu.exe -x 100
请输入存储文件的绝对路径:
D:\LessonProjects\shudu\
输入错误,请重新输入!
PS D:\LessonProjects\shudu\x64\Debug>

```

```

PS D:\LessonProjects\shudu\x64\Debug> .\shudu.exe -n 20 -m 6
请输入存储文件的绝对路径:
D:\LessonProjects\shudu\
输入的困难不符合规范,应为1-3之间的整数!
PS D:\LessonProjects\shudu\x64\Debug>

```

```
PS D:\LessonProjects\shudu\x64\Debug> .\shudu.exe -n 20 -r 10
请输入存储文件的绝对路径:
D:\LessonProjects\shudu\
[-r]项参数不规范, 应输入a-b形式的参数, 请重新输入!
PS D:\LessonProjects\shudu\x64\Debug> █
```

此时只需要根据提示重新进行输入即可。

```
PS D:\LessonProjects\shudu\x64\Debug> .\shudu.exe -n 20 -r 10
请输入存储文件的绝对路径:
D:\LessonProjects\shudu\
[-r]项参数不规范, 应输入a-b形式的参数, 请重新输入!
PS D:\LessonProjects\shudu\x64\Debug> .\shudu.exe -n 20 -r 20-55
请输入存储文件的绝对路径:
D:\LessonProjects\shudu\
-----正在生成20个数独题目-----
生成完成!
生成完成! 已生成20个数独游戏, 挖空范围在[20, 55]之间
PS D:\LessonProjects\shudu\x64\Debug> █
```

## 5 保存与加载

数独生成与求解的各种文件名解释如下:

- `final.txt`: 默认终局文件
- `question.txt`: 默认题目存储文件
- `ans.txt`: 默认求解结果文件

建议用户使用一个固定目录来存储与管理这些文件, 以免发生混乱。

## 质量分析

我们使用微软的CppCoreCheck进行代码质量分析。

### 1 消除警告

我们首先分析警告:

#### 1. 算数溢出

```
⚠ C26451 算术溢出: 使用 4 字节值上的运算符 +, 然后将结果转换到 8 字节值。在调用运算符 + 之前将值强制转换为宽类型可避免溢出(io.2)。
⚠ C26451 算术溢出: 使用 4 字节值上的运算符 +, 然后将结果转换到 8 字节值。在调用运算符 + 之前将值强制转换为宽类型可避免溢出(io.2)。
```

原代码为:

```
if (board[startRow + i][startCol + j] == num) {
    return false;
}
```

该警告指的是在计算 `startRow + i` 和 `startCol + j` 时, 可能会发生整数溢出。这可能是因为编译器把 `startRow + i` 和 `startCol + j` 这两个表达式的结果从 `int` (4字节) 转换为 `size_t` (在64位系统是8字节) 的过程中发生的。然而, 由于 `startRow`, `startCol`, `i`, 和 `j` 的值都在 0 到 9 之间, 所以这里实际上不可能发生溢出。

但是为了消除, 我们应该在做加法运算之前, 就先把操作数转换为 `size_t`。这样可以确保加法运算的结果不会超过 `size_t` 可以表示的范围。

```
// 检查小九宫格是否有重复数字
int startRow = row - row % 3;
int startCol = col - col % 3;
for (int i = 0; i < 3; ++i) {
    for (int j = 0; j < 3; ++j) {
        size_t rowIndex = static_cast<size_t>(startRow) + i;
        size_t colIndex = static_cast<size_t>(startCol) + j;
        if (board[rowIndex][colIndex] == num) {
            return false;
        }
        /*if (board[startRow + i][startCol + j] == num) {
            return false;
        }*/
    }
}
```

而另一处算数溢出是发生在打印进度条时:

**C26451** 算术溢出: 使用 4 字节值上的运算符 -, 然后将结果转换到 8 字节值。在调用运算符 - 之前将值强制转换为宽类型可避免溢出(io.2)。  
**C4244** “参数”: 从“time\_t”转换到“unsigned int”, 可能丢失数据

源代码为

```
// 打印已完成和未完成进度条
std::cout << std::setw(barWidth) << std::setfill('=') << "=";
std::cout << std::setw(this->ncols - barWidth) << std::setfill(' ') << "|";
```

同理, 我们使用 `static_cast<size_t>` 即可消除警告。

```
// 打印已完成和未完成进度条
std::cout << std::setw(barWidth) << std::setfill('=') << "=";
//std::cout << std::setw(this->ncols - barWidth) << std::setfill(' ') << "|";
std::cout << std::setw(static_cast<size_t>(this->ncols) - static_cast<size_t>(barWidth)) << std::setfill(' ') << "|";
```

## 2. `time_t` 转换到 `unsigned int`, 可能丢失数据

“参数”: 从“time\_t”转换到“unsigned int”, 可能丢失数据

该问题的原代码为:

```
void GenerateHandler::SelectFinal() {
    //随机挑选一个终局
    std::srand(std::time(0)); // 设置种子, 确保每次运行生成的随机数序列不同
    current_selectFinal = generateRandomNumber(0, FinalNum); // 生成1到100之间的随机数
}
```

这个警告是由于将 `time_t` 类型的值转换为 `unsigned int` 类型时可能会丢失数据导致的。同样, 我们可以使用 `static_cast` 进行显式的类型转换。

修改代码为:

```
void GenerateHandler::SelectFinal() {
    //随机挑选一个终局
    std::srand(static_cast<unsigned int>(std::time(0))); // 设置种子, 确保每次运行生成的随机数序列不同
    current_selectFinal = generateRandomNumber(0, FinalNum); // 生成1到100之间的随机数
}
```

即可消除警告。

## 3. `_Rep` 转换到 `int`, 可能丢失数据

**C4244** “参数”: 从“\_Rep”转换到“int”, 可能丢失数据  
**C4244** “初始化”: 从“double”转换到“int”, 可能丢失数据

发生警告的代码为:

```
void ProgressBar::start() {
    // 记录开始时间，并初始化定时器
    this->beginTime = steady_clock::now();
    this->lastTime = this->beginTime;
    // 定时器用于定时调用重绘函数
    this->timer.start(this->interval.count(), std::bind(&ProgressBar::show, this));
}
```

本质上也是数据类型转换的问题，使用 `static_cast` 解决该问题。修改代码如下：

```
this->lastTime = this->beginTime;
// 定时器用于定时调用重绘函数
this->timer.start(static_cast<int>(this->interval.count()), std::bind(&ProgressBar::show, this));
}
```

即可消除警告。


另一处同类型的警告的代码为：

```
// 之后的两部分内容分别为打印已过的时间和剩余时间
int timeFromStartCount = duration<double>(timeFromStart).count();
```

修改为：

```
// 之后的两部分内容分别为打印已过的时间和剩余时间
int timeFromStartCount = static_cast<int>(duration<double>(timeFromStart).count());
```

#### 4. `double` 转换到 `int`，可能丢失数据

 **C4244** “初始化”：从“double”转换到“int”，可能丢失数据

发生警告的代码为：




```
// 计算应该绘制多少=符号
int barWidth = present * this->colsRatio;
// 打印已完成和未完成进度条
```

对于将 `double` 转换为 `int`，使用 `static_cast<int>` 来消除警告。

修改代码为：

```
// 计算应该绘制多少=符号
int barWidth = static_cast<int>(present * this->colsRatio);
// 打印已完成和未完成进度条
```

另一处同类型的数据转换问题的警告为：

 **C4244** “初始化”：从“double”转换到“int”，可能丢失数据  
 **C4244** “初始化”：从“\_Rep”转换到“int”，可能丢失数据  
 **C4244** “=”：从“double”转换到“int”，可能丢失数据

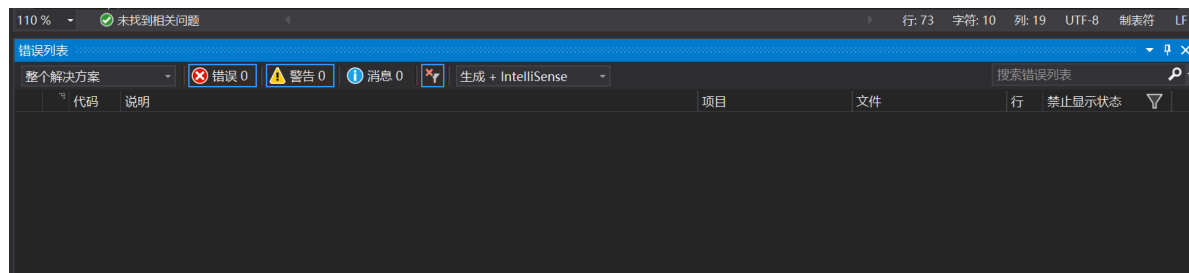
发生警告的代码为：

```
if (rate != 0) {
    // 剩余时间的估计是用这次的速度和未完成的数量进行估计
    timeLast = (this->totalNum - tmpFinished) / rate;
}
```

同样使用 `static_cast` 即可消除警告。

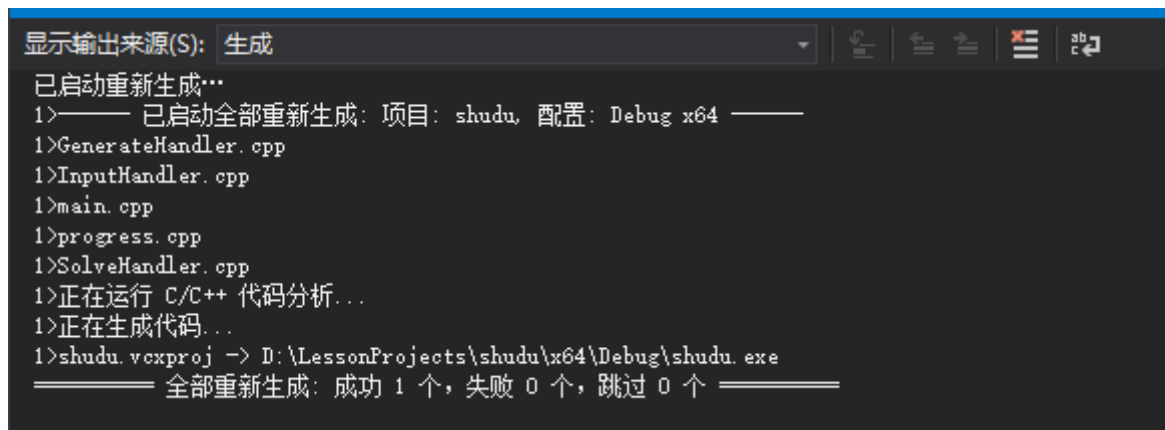
```
if (rate != 0) {
    // 剩余时间的估计是用这次的速度和未完成的数量进行估计
    timeLast = static_cast<int>((this->totalNum - tmpFinished) / rate);
}
```

至此，所有警告已全部消除。



## 2 静态代码分析

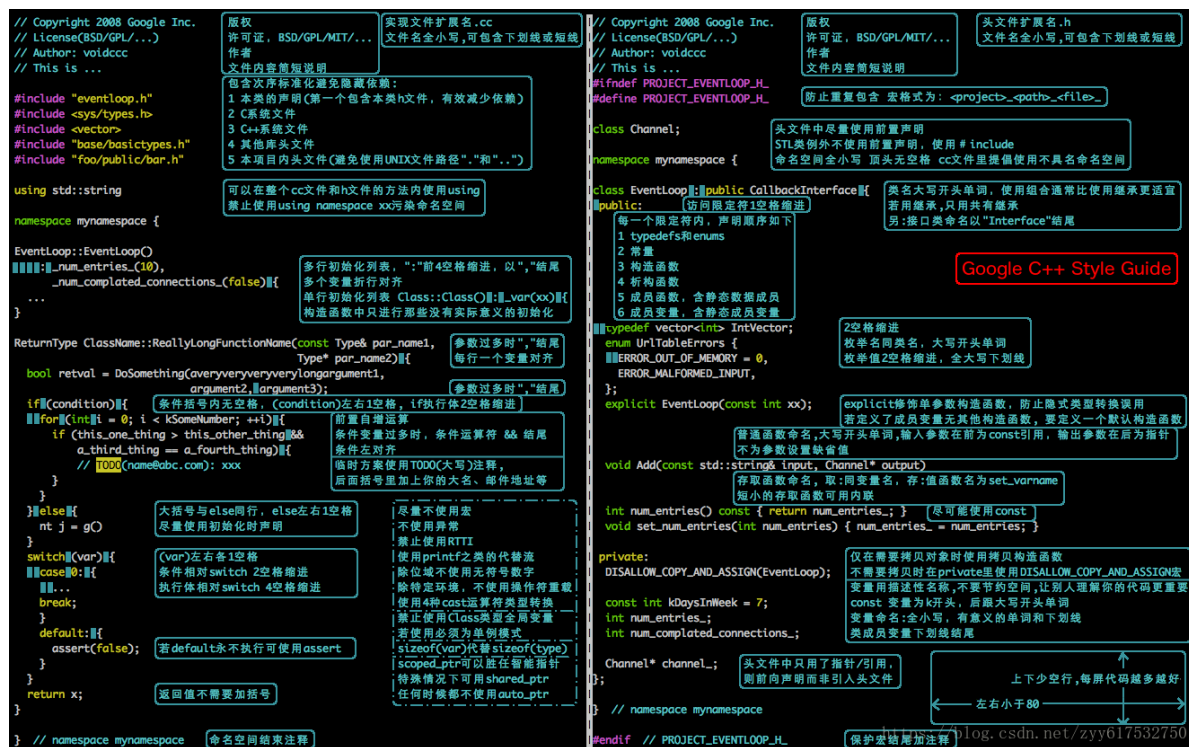
使用VS2019中的Code Analysis进行分析，分析结果如下所示：



成功运行，没有警告。

## 3 代码规范检查

这里遵循的是谷歌的C++编程规范，链接为：<https://zh-google-styleguide.readthedocs.io/en/latest/google-cpp-styleguide/contents/>，大致规范为：



图源<https://blog.csdn.net/zyy617532750/article/details/81264648>

我们使用的代码规范检查工具为cpplint。



## 1. GenerateHandler.cpp

修改前:

```
PS D:\LessonProjects\shudu\shudu> cpplint --verbose=3 .\GenerateHandler.cpp
.\GenerateHandler.cpp:0: No copyright message found. You should have a line: "Copyright [year] <Copyright Owner>" [legal/copyright] [5]
.\GenerateHandler.cpp:5: Found C++ system header after other header. Should be: GenerateHandler.h, c system, c++ system, other. [build/include_order] [4]
.\GenerateHandler.cpp:6: Do not use namespace using-directives. Use using-declarations instead. [build/namespaces] [5]
.\GenerateHandler.cpp:24: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.cpp:28: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.cpp:30: Line ends in whitespace. Consider deleting these extra spaces. [whitespace/end_of_line] [4]
.\GenerateHandler.cpp:31: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.cpp:33: Line ends in whitespace. Consider deleting these extra spaces. [whitespace/end_of_line] [4]
.\GenerateHandler.cpp:34: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.cpp:36: Line ends in whitespace. Consider deleting these extra spaces. [whitespace/end_of_line] [4]
.\GenerateHandler.cpp:37: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.cpp:42: An else should appear on the same line as the preceding } [whitespace/newline] [4]
.\GenerateHandler.cpp:42: If an else has a brace on one side, it should have it on both [readability/braces] [5]
.\GenerateHandler.cpp:49: An else should appear on the same line as the preceding } [whitespace/newline] [4]
.\GenerateHandler.cpp:49: If an else has a brace on one side, it should have it on both [readability/braces] [5]
.\GenerateHandler.cpp:59: An else should appear on the same line as the preceding } [whitespace/newline] [4]
.\GenerateHandler.cpp:59: If an else has a brace on one side, it should have it on both [readability/braces] [5]
.\GenerateHandler.cpp:60: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.cpp:63: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.cpp:67: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.cpp:70: Redundant blank line at the end of a code block should be deleted. [whitespace/blank_line] [3]
.\GenerateHandler.cpp:97: An else should appear on the same line as the preceding } [whitespace/newline] [4]
.\GenerateHandler.cpp:97: If an else has a brace on one side, it should have it on both [readability/braces] [5]
.\GenerateHandler.cpp:104: An else should appear on the same line as the preceding } [whitespace/newline] [4]
.\GenerateHandler.cpp:104: If an else has a brace on one side, it should have it on both [readability/braces] [5]
.\GenerateHandler.cpp:189: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.cpp:195: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.cpp:205: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.cpp:213: Missing spaces around = [whitespace/operators] [4]
.\GenerateHandler.cpp:214: Missing spaces around = [whitespace/operators] [4]
.\GenerateHandler.cpp:225: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.cpp:227: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.cpp:227: Missing spaces around = [whitespace/operators] [4]
.\GenerateHandler.cpp:249: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.cpp:251: Line ends in whitespace. Consider deleting these extra spaces. [whitespace/end_of_line] [4]
.\GenerateHandler.cpp:253: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.cpp:264: Using deprecated casting style. Use static_cast<int>(...) instead [readability/casting] [4]
.\GenerateHandler.cpp:274: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.cpp:277: Could not find a newline character at the end of the file. [whitespace/ending_newline] [5]
```

总结错误如下:

1. 没有版权信息: 应该在文件中添加版权信息行, 例如: "Copyright [year] ". [legal/copyright]
2. 头文件包含顺序问题: C++系统头文件应该放在其他头文件之前, 正确顺序应为: GenerateHandler.h、C系统头文件、C++系统头文件、其他头文件。[build/include\_order]
3. 命名空间错误: 不要使用命名空间的using-directives, 应使用using-declarations。[build/namespaces]
4. 空格与注释问题: 应在注释符 "//" 和注释之间添加空格, 删除行末多余的空格, 确保代码的一致性和可读性。[whitespace/comments] [whitespace/end\_of\_line]
5. 大括号和else语句问题: else语句应与前面的"}" 在同一行上, 如果有大括号, 应该在两边都使用。这样可以提高代码的可读性。[whitespace/newline] [readability/braces]
6. 多余的空白行问题: 应删除代码块末尾多余的空白行。[whitespace/blank\_line]
7. 等号周围的空格问题: 等号周围应该添加空格, 提高代码的可读性。[whitespace/operators]
8. 强制类型转换问题: 使用过时的C风格强制类型转换, 建议使用 static\_cast<int>(...) 进行类型转换。[readability/casting]
9. 文件末尾缺少换行符: 文件末尾应包含一个换行符。[whitespace/ending\_newline]

根据提示, 我们逐行对代码进行修改, 修改之后无error, 如下图所示:

```
PS D:\LessonProjects\shudu\shudu> cpplint --verbose=3 .\GenerateHandler.cpp
Done processing .\GenerateHandler.cpp
PS D:\LessonProjects\shudu\shudu>
```

## 2. GenerateHandler.h

对于 GenerateHandler.h, cpplint提示信息如下:

```

PS D:\LessonProjects\shudu\shudu> cpplint --verbose=3 .\GenerateHandler.h
.\GenerateHandler.h:9: Found C system header after C++ system header. Should be: GenerateHandler.h, c system, c++ system, other. [build/include_order] [4]
.\GenerateHandler.h:10: Include the directory when naming header files [build/include_subdir] [4]
.\GenerateHandler.h:11: Found C++ system header after other header. Should be: GenerateHandler.h, c system, c++ system, other. [build/include_order] [4]
.\GenerateHandler.h:12: Found C++ system header after other header. Should be: GenerateHandler.h, c system, c++ system, other. [build/include_order] [4]
.\GenerateHandler.h:13: Found C++ system header after other header. Should be: GenerateHandler.h, c system, c++ system, other. [build/include_order] [4]
.\GenerateHandler.h:15: Do not use namespace using-directives. Use using-declarations instead. [build/namespaces] [5]
.\GenerateHandler.h:18: private: should be indented +1 space inside class Generatehandler [whitespace/indent] [3]
.\GenerateHandler.h:22: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.h:23: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.h:24: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.h:25: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.h:26: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.h:27: public: should be indented +1 space inside class Generatehandler [whitespace/indent] [3]
.\GenerateHandler.h:27: "public:" should be preceded by a blank line [whitespace/blank_line] [3]
.\GenerateHandler.h:29: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.h:40: Should have a space between // and comment [whitespace/comments] [4]
.\GenerateHandler.h:42: Should have a space between // and comment [whitespace/comments] [4]
Done processing .\GenerateHandler.h
Total errors found: 17

```

以上errors总结为：

1. 头文件包含顺序问题：C系统头文件应该在C++系统头文件之前。[build/include\_order]
2. 头文件命名问题：在命名头文件时，应包含目录信息。[build/include\_subdir]
3. 命名空间错误：不要使用命名空间的using-directives，应使用using-declarations。  
[build/namespaces]
4. 类中的private部分缩进问题：在GenerateHandler类中的private部分应该缩进一个空格。  
[whitespace/indent]
5. 注释与斜线问题：注释符 "//" 和注释之间应添加一个空格。[whitespace/comments]
6. 类中的public部分缩进问题：在GenerateHandler类中的public部分应该缩进一个空格。  
[whitespace/indent]
7. 类中的public部分前的空白行问题：在GenerateHandler类中的public部分前应有一个空白行。  
[whitespace/blank\_line]

逐一解决以上问题，最终完全解决所有报错：

```

PS D:\LessonProjects\shudu\shudu> cpplint --verbose=3 .\GenerateHandler.h
Done processing .\GenerateHandler.h
PS D:\LessonProjects\shudu\shudu>

```

### 3. InputHandler.cpp

```

PS D:\LessonProjects\shudu\shudu> cpplint --verbose=3 .\InputHandler.cpp
.\InputHandler.cpp:10: Include the directory when naming header files [build/include_subdir] [4]
.\InputHandler.cpp:12: Do not use namespace using-directives. Use using-declarations instead. [build/namespaces] [5]
.\InputHandler.cpp:40: If an else has a brace on one side, it should have it on both [readability/braces] [5]
.\InputHandler.cpp:77: If an else has a brace on one side, it should have it on both [readability/braces] [5]
.\InputHandler.cpp:110: Should have a space between // and comment [whitespace/comments] [4]
.\InputHandler.cpp:114: Missing space before ( in if( [whitespace/parens] [5]
.\InputHandler.cpp:114: Missing space before { [whitespace/braces] [5]
.\InputHandler.cpp:115: Missing spaces around = [whitespace/operators] [4]
.\InputHandler.cpp:131: Should have a space between // and comment [whitespace/comments] [4]
.\InputHandler.cpp:155: Missing space after , [whitespace/comma] [3]
.\InputHandler.cpp:156: Missing spaces around << [whitespace/operators] [3]
.\InputHandler.cpp:167: Missing space after , [whitespace/comma] [3]
.\InputHandler.cpp:170: If an else has a brace on one side, it should have it on both [readability/braces] [5]
.\InputHandler.cpp:170: Missing spaces around == [whitespace/operators] [3]
.\InputHandler.cpp:170: Missing space before ( in if( [whitespace/parens] [5]
.\InputHandler.cpp:171: { should almost always be at the end of the previous line [whitespace/braces] [4]
.\InputHandler.cpp:171: Missing space before { [whitespace/braces] [5]
.\InputHandler.cpp:172: Missing space after , [whitespace/comma] [3]
.\InputHandler.cpp:176: Missing space after , [whitespace/comma] [3]
.\InputHandler.cpp:183: Line ends in whitespace. Consider deleting these extra spaces. [whitespace/end_of_line] [4]
.\InputHandler.cpp:190: Should have a space between // and comment [whitespace/comments] [4]
.\InputHandler.cpp:225: Should have a space between // and comment [whitespace/comments] [4]
.\InputHandler.cpp:276: Line ends in whitespace. Consider deleting these extra spaces. [whitespace/end_of_line] [4]
.\InputHandler.cpp:280: Missing space after , [whitespace/comma] [3]
.\InputHandler.cpp:281: If an else has a brace on one side, it should have it on both [readability/braces] [5]
.\InputHandler.cpp:282: { should almost always be at the end of the previous line [whitespace/braces] [4]
.\InputHandler.cpp:282: Missing space before { [whitespace/braces] [5]
.\InputHandler.cpp:283: Missing space after , [whitespace/comma] [3]
.\InputHandler.cpp:285: Missing space after , [whitespace/comma] [3]
.\InputHandler.cpp:298: Line ends in whitespace. Consider deleting these extra spaces. [whitespace/end_of_line] [4]
.\InputHandler.cpp:310: Using deprecated casting style. Use static_cast<int>(...) instead [readability/casting] [4]
.\InputHandler.cpp:313: Line ends in whitespace. Consider deleting these extra spaces. [whitespace/end_of_line] [4]
.\InputHandler.cpp:379: Could not find a newline character at the end of the file. [whitespace/ending_newline] [5]
Done processing .\InputHandler.cpp
Total errors found: 33

```

以上报错大致与前两个文件报错内容相同，逐一解决即可：



```
PS D:\LessonProjects\shudu\shudu> cpplint --verbose=3 .\InputHandler.cpp
Done processing .\InputHandler.cpp
PS D:\LessonProjects\shudu\shudu>
```

#### 4. InputHandler.h

```
PS D:\LessonProjects\shudu\shudu> cpplint --verbose=3 .\InputHandler.h
.\InputHandler.h:11: Do not use namespace using-directives. Use using-declarations instead. [build/namespaces] [5]
.\InputHandler.h:14: public: should be indented +1 space inside class InputHandler [whitespace/indent] [3]
.\InputHandler.h:17: Should have a space between // and comment [whitespace/comments] [4]
.\InputHandler.h:21: Should have a space between // and comment [whitespace/comments] [4]
.\InputHandler.h:29: private: should be indented +1 space inside class InputHandler [whitespace/indent] [3]
.\InputHandler.h:29: "private:" should be preceded by a blank line [whitespace/blank_line] [3]
.\InputHandler.h:32: Should have a space between // and comment [whitespace/comments] [4]
.\InputHandler.h:33: Should have a space between // and comment [whitespace/comments] [4]
.\InputHandler.h:34: Should have a space between // and comment [whitespace/comments] [4]
Done processing .\InputHandler.h
Total errors found: 9
```

常规错误，解决即可：

```
Total errors found: 2
PS D:\LessonProjects\shudu\shudu> cpplint --verbose=3 .\InputHandler.h
Done processing .\InputHandler.h
PS D:\LessonProjects\shudu\shudu>
```

#### 5. main.cpp

```
PS D:\LessonProjects\shudu\shudu> cpplint --verbose=3 .\main.cpp
.\main.cpp:9: Do not use namespace using-directives. Use using-declarations instead. [build/namespaces] [5]
.\main.cpp:19: Could not find a newline character at the end of the file. [whitespace/ending_newline] [5]
Done processing .\main.cpp
Total errors found: 2
PS D:\LessonProjects\shudu\shudu>
```

消除以上错误：

```
PS D:\LessonProjects\shudu\shudu> cpplint --verbose=3 .\main.cpp
Done processing .\main.cpp
PS D:\LessonProjects\shudu\shudu> █
```

#### 6. progree.cpp

```
PS D:\LessonProjects\shudu\shudu> cpplint --verbose=3 .\progress.cpp
Done processing .\progress.cpp
PS D:\LessonProjects\shudu\shudu>
```

#### 7. progress.h

```
PS D:\LessonProjects\shudu\shudu> cpplint --verbose=3 .\progress.h
Done processing .\progress.h
PS D:\LessonProjects\shudu\shudu> █
```

#### 8. SolveHandler.cpp

大部分错误我们之前已经遇到过，逐行修改即可：

```
Total errors found: 1
PS D:\LessonProjects\shudu\shudu> cpplint --verbose=3 .\SolveHandler.cpp
Done processing .\SolveHandler.cpp
PS D:\LessonProjects\shudu\shudu>
```

#### 9. SolveHandler.h

```
PS D:\LessonProjects\shudu\shudu> cpplint --verbose=3 .\SolveHandler.h
Done processing .\SolveHandler.h
PS D:\LessonProjects\shudu\shudu> █
```

## 单元测试

使用Visual Studio自带的测试项目模板，关联Sudoku项目进行测试。需要创建一个新项目，使用本机单元测试项目模板，然后添加现有项，关联 main.obj 即可。

# 1 定义输入

首先我们定义一连串的字符串，以模拟正常可能的输入。正常输入如下所示：

参数名字	参数意义	范围限制	用法示例
-c	需要的数独终盘数量	1-1000000	示例: 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	示例: 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	示例: sudoku.exe -n 1000 [表示生成1000个数独游戏]
-m	生成游戏的难度	1-3	示例: 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	示例: 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		示例: 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]

```
int argc1, argc2, argc3, argc4;
char** argv1, ** argv2, ** argv3, ** argv4, ** argv5, ** argv6, ** argv7;
string path = "D:\\LessonProjects\\shudu\\";

argc1 = 3;
argc2 = 4;
argc3 = 5;
argc4 = 6;
argv1 = new char* [3]; //c
argv2 = new char* [3]; //s
argv3 = new char* [3]; //n
argv4 = new char* [4]; //n u
argv5 = new char* [5]; //n m
argv6 = new char* [5]; //n r
argv7 = new char* [6]; //n r u

for (int i = 0; i < 3; i++) {
    argv1[i] = new char[30];
    argv2[i] = new char[30];
    argv3[i] = new char[30];
}
for (int i = 0; i < 4; i++) {
    argv4[i] = new char[30];
}
for (int i = 0; i < 5; i++) {
    argv5[i] = new char[30];
    argv6[i] = new char[30];
}
for (int i = 0; i < 6; i++) {
    argv7[i] = new char[30];
}

strcpy_s(argv1[0], 30, "shudu.exe");
strcpy_s(argv1[1], 30, "-c");
strcpy_s(argv1[2], 30, "100");

strcpy_s(argv2[0], 30, "shudu.exe");
strcpy_s(argv2[1], 30, "-s");
strcpy_s(argv2[2], 30, "question.txt");

strcpy_s(argv3[0], 30, "shudu.exe");
strcpy_s(argv3[1], 30, "-n");
strcpy_s(argv3[2], 30, "100");
```

```

strcpy_s(argv4[0], 30, "shudu.exe");
strcpy_s(argv4[1], 30, "-n");
strcpy_s(argv4[2], 30, "10");
strcpy_s(argv4[3], 30, "-u");

strcpy_s(argv5[0], 30, "shudu.exe");
strcpy_s(argv5[1], 30, "-n");
strcpy_s(argv5[2], 30, "100");
strcpy_s(argv5[3], 30, "-m");
strcpy_s(argv5[4], 30, "3");

strcpy_s(argv6[0], 30, "shudu.exe");
strcpy_s(argv6[1], 30, "-n");
strcpy_s(argv6[2], 30, "100");
strcpy_s(argv6[3], 30, "-r");
strcpy_s(argv6[4], 30, "20-55");

strcpy_s(argv7[0], 30, "shudu.exe");
strcpy_s(argv7[1], 30, "-n");
strcpy_s(argv7[2], 30, "100");
strcpy_s(argv7[3], 30, "-r");
strcpy_s(argv7[4], 30, "20-55");
strcpy_s(argv7[3], 30, "-u");

```

之后便可以开始各个部分的测试。

## 2 测试用例设计

1. 当参数为 `-c` 时，默认的数量为100，即生成100个游戏终盘。

我们使用 `InputHandler` 的 `GetNum` 和 `GetType` 函数来测试 `setAbsPath` 和 `check` 函数是否正常。使用 `setAbsPath` 设置路径，并使用 `check` 函数将参数传入。首先验证是否正确接收到数量100这个数字，其次验证是否正确接收到参数为 `-c`。如果都正确接收，函数会返回 `true`，失败将显示错误信息，并返回 `false`。

```

//测试InputHandler,参数为-c时
TEST_METHOD(TestMethod1)
{
    InputHandler inputs;
    inputs.setAbsPath(path);
    inputs.check(argc1, argv1);
    Assert::AreEqual(inputs.GetNum() == 100, true);
    Assert::AreEqual(inputs.GetType1() == 'c', true);
}

```

2. 当参数为 `-s` 时，求解游戏。

仍然使用 `InputHandler` 的 `GetNum` 和 `GetType` 函数进行测试，期望正确识别参数 `-s` 和文件路径。

```
//测试InputHandler, 参数为-s时
TEST_METHOD(TestMethod2)
{
    InputHandler inputs;
    inputs.setAbsPath(path);
    inputs.check(argc1, argv2);
    Assert::AreEqual(inputs.GetType1() == 's', true);
}
```

### 3. 当参数为 -n 时, 生成数独游戏

此时我们需要使用 `inputs.generator` 的 `generate` 函数来生成数独游戏。如果生成成功, 将会返回 `true`。且能正常识别参数 `-n`。

```
TEST_METHOD(TestMethod3)
{
    InputHandler inputs;
    inputs.setAbsPath(path);
    inputs.check(argc1, argv3);
    bool abc = inputs.generator.generate(100, 18, 64, false);
    Assert::AreEqual(inputs.GetType1() == 'n', true);
    Assert::AreEqual(abc, true);
}
```

### 4. 命令行参数只有两个

正常来说, 命令行的参数至少为3个。若只传入两个参数, 那么程序本身会输出错误提示, 且参数类型被初始化后不能被修改, 即一直为初始化的 `y`。

```
//测试InputHandler, 命令行参数不是2个
TEST_METHOD(TestMethod4)
{
    argc1 = 2;
    InputHandler inputs;
    inputs.setAbsPath(path);
    inputs.check(argc1, argv1);
    Assert::AreEqual(inputs.GetType1() == 'y', true);
}
```

### 5. 参数不在给定范围中

假设参数不是给定的6个可使用参数中的任何一个, 为 `x`, 则参数的type不能被正确地赋值, 和第四个相同:

```
//测试InputHandler, 参数不是-c\s\n
TEST_METHOD(TestMethod5)
{
    argv1[1][1] = 'x';
    InputHandler inputs;
    inputs.setAbsPath(path);
    inputs.check(argc1, argv1);
    Assert::AreEqual(inputs.GetType1() == 'y', true);
}
```

### 6. 测试生成数量

```
TEST_METHOD(TestMethod6)
{
    argv1[2][0] = '-';
    argv1[2][1] = '1';
    InputHandler inputs;
    inputs.setAbsPath(path);
    inputs.check(argc1, argv1);
    Assert::AreEqual(inputs.GetNum() == 1, true);
}
```

## 7. 测试能否生成数独终局

调用generate函数，以生成数独终局。如果成功生成，则会返回 true，失败将显示错误信息，并返回 false。

```
//测试生成数独终局
TEST_METHOD(TestMethod7)
{
    strcpy_s(argv1[2], 30, "1");
    InputHandler inputs;
    inputs.setAbsPath(path);
    inputs.check(argc1, argv1);
    bool a = inputs.generator.generate(inputs.GetNum(), 18, 64,
false);
    Assert::AreEqual(true, a);
}
```

## 8. 测试生成唯一解

同上，需要调用generate函数。但是这里需要在 generate 函数中传入 true，表明生成具有唯一解的游戏，成功生成将返回 true，失败将显示错误信息，并返回 false。

```
//测试生成唯一解
TEST_METHOD(TestMethod8)
{
    InputHandler inputs;
    inputs.setAbsPath(path);
    inputs.check(argc2, argv4);
    Assert::AreEqual(inputs.GetNum() == 10, true);
    Assert::AreEqual(inputs.GetType1() == 'n', true);
    Assert::AreEqual(inputs.GetType2() == 'u', true);
    bool a = inputs.generator.generate(inputs.GetNum(),s 18, 64,
true);
    Assert::AreEqual(true, a);
}
```

## 9. 测试指定level

同样，测试 -n 和 -m 参数能否被正确识别，数字是否被正确赋值。当参数 -m 为3时，挖空数量为 33~64，成功创建游戏将返回 true，失败将显示错误信息，并返回 false。

```
//测试指定level
TEST_METHOD(TestMethod9)
{
    InputHandler inputs;
    inputs.setAbsPath(path);
    inputs.check(argc3, argv5);
    Assert::AreEqual(inputs.GetNum() == 100, true);
    Assert::AreEqual(inputs.GetType1() == 'n', true);
    Assert::AreEqual(inputs.GetType2() == 'm', true);
    Assert::AreEqual(inputs.GetLevel() == 3, true);
    bool a = inputs.generator.generate(inputs.GetNum(), 33, 64,
true);

    Assert::AreEqual(true, a);
}
```

#### 10. 测试指定挖空数量

同样调用generate函数，但是挖空范围由 GetRange 获取，成功生成游戏将返回 true，失败将显示错误信息，并返回 false。

```
//测试指定挖空数量
TEST_METHOD(TestMethod10)
{
    InputHandler inputs;
    inputs.setAbsPath(path);
    inputs.check(argc3, argv6);
    Assert::AreEqual(inputs.GetNum() == 100, true);
    Assert::AreEqual(inputs.GetType1() == 'n', true);
    Assert::AreEqual(inputs.GetType2() == 'r', true);
    Assert::AreEqual(inputs.GetRange1() == 20, true);
    Assert::AreEqual(inputs.GetRange2() == 30, true);
    bool a = inputs.generator.generate(inputs.GetNum(),
inputs.GetRange1(), inputs.GetRange2(), true);
    Assert::AreEqual(true, a);
}
```

#### 11. 测试该数独可以解

测试 solveSudoku 函数，确保对于一个给定的数独游戏，它能否正确地找到解。

```
//测试该数独可以解
TEST_METHOD(TestMethod11)
{
    Generatehandler handler;
    vector<vector<int>> board = {
        {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}
    };
    Assert::AreEqual(handler.solveSudoku(board), true); // 该数独游戏
可以解
```

```
}
```

## 12. 确保生成9\*9游戏且有解

测试 `generateSudoku` 函数，确保它能生成一个9x9的数独游戏，并且这个游戏有解。

```
TEST_METHOD(TestMethod12)
{
    GenerateHandler handler;
    vector<vector<int>> board;
    handler.generateSudoku(board);

    // 检查是否生成的是9x9的数独游戏
    size_t nine = 9;
    Assert::AreEqual(board.size(), nine);
    for (const auto& row : board) {
        Assert::AreEqual(row.size(), nine);
    }

    // 检查是否有解
    Assert::AreEqual(handler.solveSudoku(board), true);
}
```

## 13. 测试能否放置数字

测试 `isValid` 函数，确保在给定的行、列和数字情况下，能否正确地验证是否可以放置这个数字。

```
//测试能否放置数字
TEST_METHOD(TestMethod13)
{
    GenerateHandler handler;
    vector<vector<int>> board = {
        {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}
    };

    Assert::AreEqual(handler.isValid(board, 0, 2, 1), true); // 5x5位
    // 置可以放1
    Assert::AreEqual(handler.isValid(board, 0, 0, 6), false); // 5x5
    // 位置不能放6
}
```

## 14. 6个参数

当输入的参数为: `-n 100 -r 20-55 -u` 时

```
TEST_METHOD(TestMethod14)
{
    InputHandler inputs;
    inputs.setAbsPath(path);
}
```

```

        inputs.check(argc4, argv7);
        Assert::AreEqual(inputs.GetNum() == 100, true);
        Assert::AreEqual(inputs.GetType1() == 'n', true);
        Assert::AreEqual(inputs.GetType2() == 'r', true);
        Assert::AreEqual(inputs.GetType3() == 'u', true);
        assert(inputs.GetRange1() == 20, true);
        assert(inputs.GetRange2() == 55, true);
        bool a = inputs.generator.generate(inputs.GetNum(),
inputs.GetType1(), inputs.GetType2(), true);
        Assert::AreEqual(true, a);

    }

```

## 15. 参数乱序

当输入的参数为: `-u -n 100 -m 1` 时

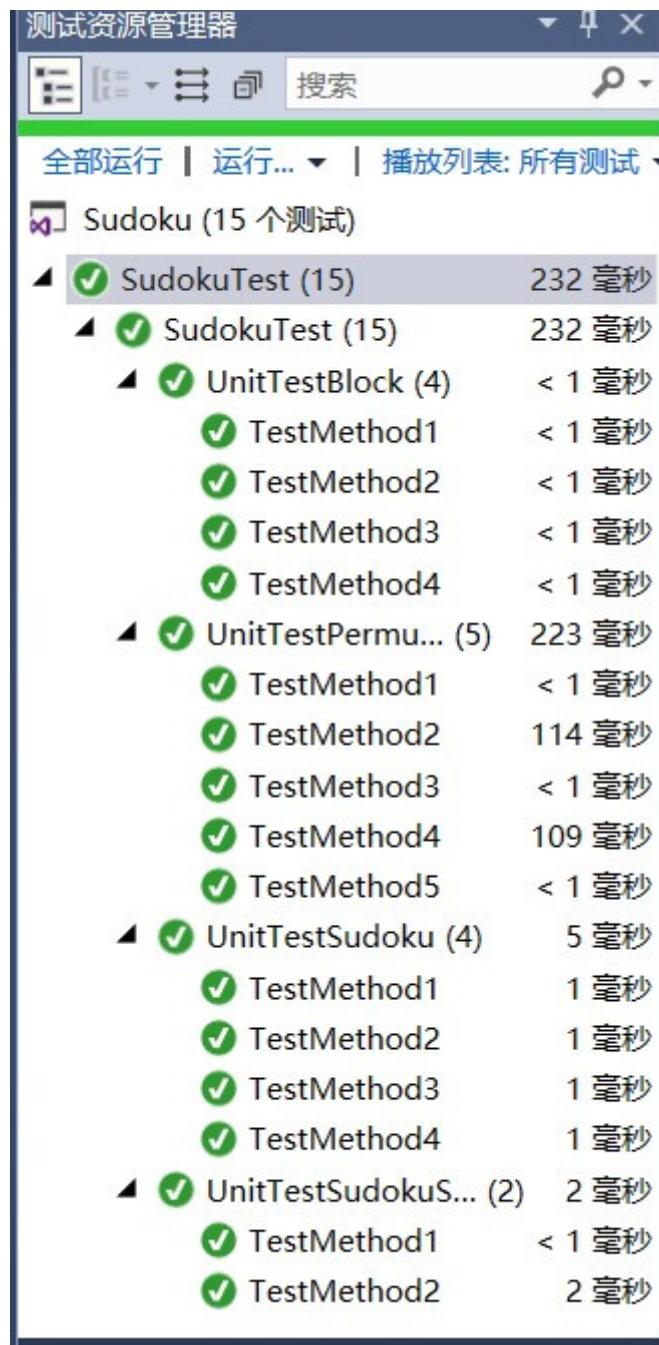
```

TEST_METHOD(TestMethod16)
{
    strcpy_s(argv7[0], 30, "shudu.exe");
    strcpy_s(argv7[1], 30, "-n");
    strcpy_s(argv7[2], 30, "100");
    strcpy_s(argv7[3], 30, "-r");
    strcpy_s(argv7[4], 30, "20-55");
    strcpy_s(argv7[3], 30, "-u");
    InputHandler inputs;
    inputs.setAbsPath(path);
    bool abc = inputs.generator.generate(1, 20, 30, false);
    Assert::AreEqual(abc, true);
}

```

单元测试全部通过:





除此之外，我们还通过命令行对代码进行了测试，以下是各种情况的执行情况：

编号	命令行参数	结果
1	-c 1	已生成1个游戏终盘
2	-c 100	已生成100个游戏终盘
3	-c 1000000	已生成1000000个游戏终盘
4	-c 1000001	不满足 $0 < n \leq 1000000$ !
5	-c 0	不满足 $0 < n \leq 1000000$ !
6	-cc 12	输入有误!
7	-c x	不满足 $0 < n \leq 1000000$ !
8	-s question.txt	完成求解!
9	-s illegalsudoku.txt	第 i 个数独无解
10	-s unexistfile.txt	文件打开失败!
11	-n 100	生成成功!
12	-n 0	不满足 $0 < n \leq 1000000$ !
13	-n 1001	不满足 $0 < n \leq 1000000$ !
14	-x	输入错误, 请重新输入!
15	-n 100 -u	生成成功
16	-n 100 -m	输入参数为三个, 但不存在-u选项!
17	-n 10000001 -u	生成数独题库数量不规范( $0 < n < 1000000$ )!请重新输入生成数
18	-n 100 -r 20-55	生成成功
19	-n 100 -r 4	[-r]项参数不规范, 应输入a-b形式的参数, 请重新输入!
20	-n 100 -r 20-e	[-r]项参数不规范, 应输入a-b形式的正整数, 请重新输入!
21	-n 100 -m 1	生成成功
22	-n 100 -m 6	输入的难度不符合规范, 应为1-3之间的整数!
23	-n 100 -x 6	输入有误!存在未定义的选项
24	-n 100 -m 1 -u	生成成功
25	-n 100 -m 1 -x	输入命令行格式错误, 出现单数个参数但未出现[-u], 请重新输入!
26	-n 100 -r 10-15 - u	存在-r项不规范问题: 可能原因1.范围设置有误2.该范围无法生成唯一解(请将范围设置在18-64)
27	-n 100 -r 22 -u	[-r]项参数不规范, 应输入a-b形式的参数, 请重新输入!
28	-m 1 -n 100 -u	生成成功

编号	命令行参数	结果
29	-r 20-55 -u -n 100	生成成功

生成个数	预期输出	实际输出
1	1	1
20	20	20
100	100	100
1000	1000	1000
1000000	1000000	1000000
数独求解 编号	预期输出	实际输出
1	正确	正确
2	正确	正确
3	正确	正确
4	正确	正确
5	正确	正确

经检查，数独生成可解，生成求解正确，代码逻辑无误。

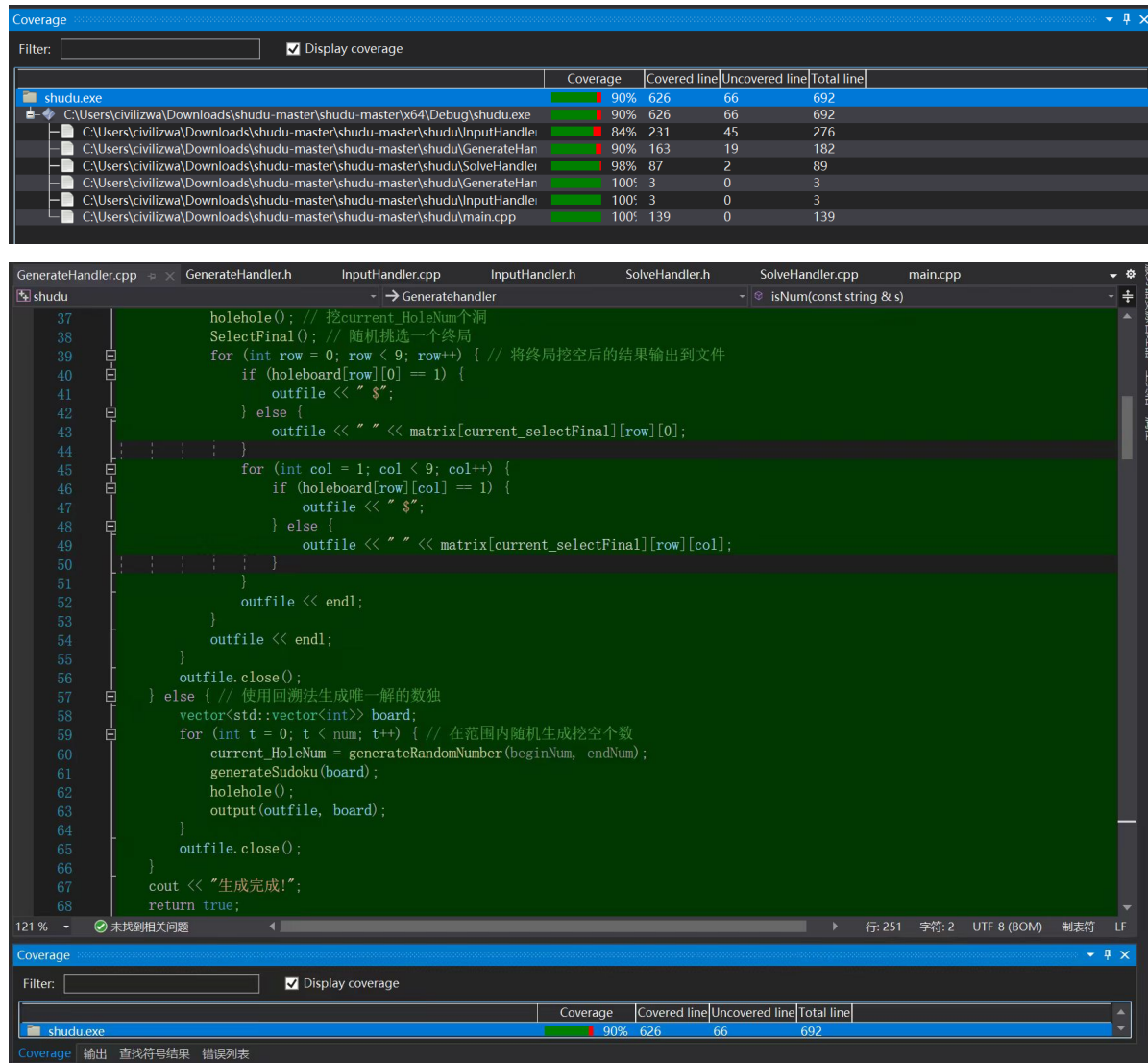
## 覆盖率报告

在VS2019中使用OpenCppCoverage进行分析，使用的指令为：

- shudu.exe -c 100
- shudu.exe -s question.txt
- shudu.exe -n 100
- shudu.exe -n 100 -u
- shudu.exe -u -n 100
- shudu.exe -n 100 -m 3
- shudu.exe -n 100 -m 2
- shudu.exe -n 100 -m 1
- shudu.exe -n 100 -m -6
- shudu.exe -n 0 -m 2
- shudu.exe -n 100 -r 20-55
- shudu.exe -n 0
- shudu.exe -r dd
- shudu.exe -n 100 -r 20-55 -u
- shudu.exe -n 100 -r 1-6 -u
- shudu.exe -n 100 -r 1 -u
- shudu.exe -u -r 20-55 -n 10
- shudu.exe -u -r 0-5 -n 10
- shudu.exe -u -r 11 -n 10

- shudu.exe -u -m 1 -n 10
- shudu.exe -u -m 2 -n 10
- shudu.exe -u -m 3 -n 10
- shudu.exe -u -m 6 -n 10
- shudu.exe -m 1 -u -n 10
- shudu.exe -m -u 1 -n 10

分析结果如下所示：



GenerateHandler.cpp   GenerateHandler.h   InputHandler.cpp   InputHandler.h   SolveHandler.h   SolveHandler.cpp   main.cpp

shudu

```
106     if (n <= 0 || n > 1000000) {
107         num = n;
108         cout << "生成数独题库数量不规范(0<n<1000000)!请重新输入生成数" << endl;
109         return;
110     }
111     generator.generate(n, 18, 64, true);
112 } else if(argc == 5){
113     string arg1 = argv[1];
114     string param1 = argv[2];
115     string arg2 = argv[3];
116     string param2 = argv[4];
117     if (arg1 == "-n") {
118         int n = isNum(param1);
119         num = n;
120         type1 = 'n';
121         if (n <= 0 || n > 1000000) {
122             cout << "生成数独题库数量不规范(0<n<1000000)!请重新输入生成数" << endl;
123             return;
124         }
125     }
126     if (arg2 == "-r") {
127         type2 = 'r';
128         string begin, end;
129         bool isBegin = true;
130         for (int i = 0; i < param2.length(); i++) {
131             if (param2[i] == '-') {
132                 isBegin = false;
133             } else {
134                 if (isBegin) {
135                     begin += param2[i];
136                 } else {
137                     end += param2[i];
138                 }
139             }
140         }
141     }
142 }
```

121 %   未找到相关问题   行: 61   字符: 18   列: 33   UTF-8   制表符   LF

Coverage

Filter:   ☒ Display coverage

	Coverage	Covered line	Uncovered line	Total line
shudu.exe	90%	626	66	692

GenerateHandler.cpp   GenerateHandler.h   InputHandler.cpp   InputHandler.h   SolveHandler.h   SolveHandler.cpp   main.cpp

shudu

```
25 }
26
27 void SolverHandler::remove(int i, int j) {
28     int num = matrix[i][j];
29     if (num != '$') {
30         // matrix[i][j] = (int)'$';
31         matrix[i][j] = static_cast<int>('$');
32         row[i] ^= (1 << num);
33         col[j] ^= (1 << num);
34         patch[(i / 3) * 3 + j / 3] ^= (1 << num);
35     }
36     blank.push(make_pair(i, j));
37 }
38
39 void SolverHandler::replace(int i, int j, int newNum) {
40     int oldNum = matrix[i][j];
41     if (oldNum == '$') {
42         insert(i, j, newNum);
43     } else {
44         matrix[i][j] = newNum;
45         int mask = (1 << oldNum) | (1 << newNum);
46         row[i] ^= mask;
47         col[j] ^= mask;
48         patch[(i / 3) * 3 + j / 3] ^= mask;
49     }
50 }
51
52 int SolverHandler::getMask(int i, int j) {
53     int mask = 0;
54     mask |= row[i];
55     mask |= col[j];
56     mask |= patch[(i / 3) * 3 + j / 3];
57 }
```

121 %   未找到相关问题   行: 1   字符: 1   UTF-8 (BOM)   空格   LF

Coverage

Filter:   ☒ Display coverage

	Coverage	Covered line	Uncovered line	Total line
shudu.exe	90%	626	66	692

```
130         if (param2[i] == '-') {
131             isBegin = false;
132         } else {
133             if (isBegin) {
134                 begin += param2[i];
135             } else {
136                 end += param2[i];
137             }
138         }
139     }
140     if (begin.length() == 0 || end.length() == 0) {
141         cout << "[+]项参数不规范，应输入a-b形式的参数，请重新输入!" << endl;
142         return;
143     }
144     int begin_num = isNum(begin);
145     int end_num = isNum(end);
146     range1 = begin_num;
147     range2 = end_num;
148     if (begin_num <= 0 || end_num <= 0 || begin_num > end_num) {
149         cout << "[+]项参数不规范，应输入a-b形式的正整数，请重新输入!" << endl;
150         return;
151     } else {
152         generator.generate(n, begin_num, end_num, false);
153         cout << "已生成" << param1 << "个数独游戏，挖空范围在[" << begin_num << ", " << end_num << "之间" << endl;
154         return;
155     }
156 } else if (arg2 == "-m") {
157     // 选择难度的时候，分为3档
158     // 第一档：挖空在5-18之间(因此不能要求唯一解)
159     // 第二档：挖空在18-32之间
160     // 第三档：挖空在33-64之间
161     level = isNum(param2);
```

	Coverage	Covered line	Uncovered line	Total line
shudu.exe	90%	626	66	692

```
172     return;
173 } else if (level == 3) {
174     generator.generate(n, 48, 64, false);
175     cout << "成功生成!" << endl;
176     return;
177 } else {
178     cout << "输入的难度不符合规范，应为1-3之间的整数!" << endl;
179     return;
180 }
181 } else {
182     cout << "输入有误!存在未定义的选项" << endl;
183     return;
184 }
185 }
186 } else if (argc == 6) {
187     // 首先需要确定-u的位置
188     int upos = 1;
189     bool isUnion = false;
190     string arg1, param1, arg2, param2;
191     for (int i = 0; i < 6; i++) {
192         string p = argv[i];
193         if (p == "-u") {
194             upos = i;
195             isUnion = true;
196         }
197     }
198     if (!isUnion) {
199         cout << "输入命令行格式错误，出现单个参数但未出现[-u]，请重新输入!" << endl;
200         return;
201     }
202     if (upos == 1) {
```

	Coverage	Covered line	Uncovered line	Total line
shudu.exe	90%	626	66	692

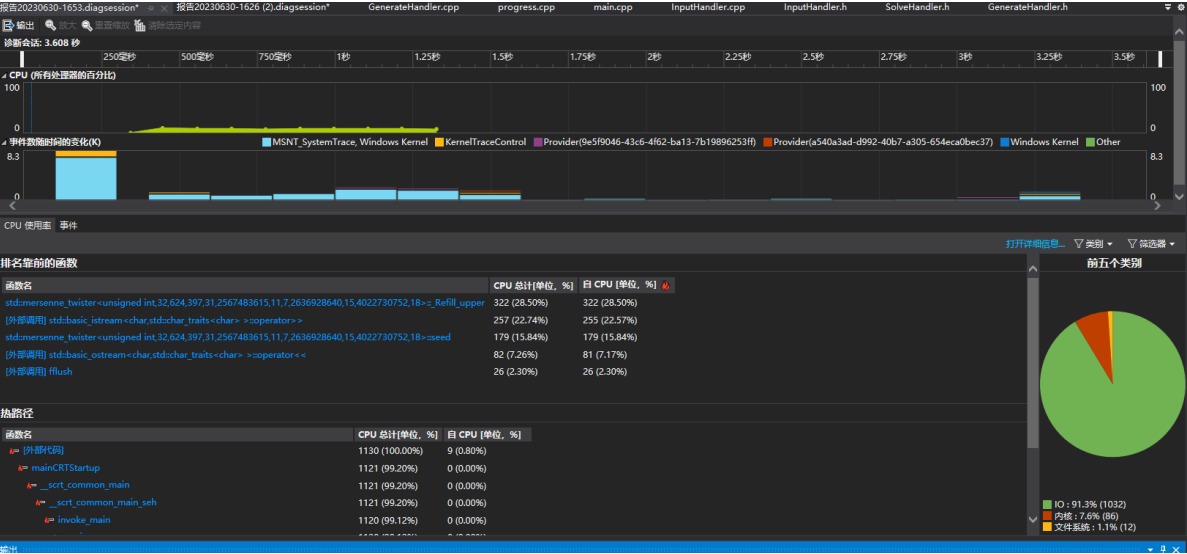
可以看到，这些指令几乎已经覆盖了所有的代码，只有少数else循环表示错误的语句没有进入，整体覆盖率达到90%，表明代码的测试质量较高。

## 性能分析



# 1 性能探查器

我们VS2019的性能探查器来进行分析。



函数名	CPU 总计[单位, %]	自 CPU [单位, %]
std::mersenne_twister<unsigned int,32,624,397,31,2567483615,11,7,2636928640,15,4022730752,18>::_Refill_upper	322 (28.50%)	322 (28.50%)
[外部调用] std::basic_istream<char,std::char_traits<char> >::operator>	257 (22.74%)	255 (22.57%)
std::mersenne_twister<unsigned int,32,624,397,31,2567483615,11,7,2636928640,15,4022730752,18>::seed	179 (15.84%)	179 (15.84%)
[外部调用] std::basic_ostream<char,std::char_traits<char> >::operator<<	82 (7.26%)	81 (7.17%)
[外部调用] fflush	26 (2.30%)	26 (2.30%)

函数名	CPU 总计[单位, %]	自 CPU [单位, %]
[外部代码]	1130 (100.00%)	9 (0.80%)
mainCRTStartup	1121 (99.20%)	0 (0.00%)
_srt_common_main	1121 (99.20%)	0 (0.00%)
_srt_common_main_seh	1121 (99.20%)	0 (0.00%)
invoke_main	1120 (99.12%)	0 (0.00%)
main	1120 (99.12%)	0 (0.00%)
UnitTest	1120 (99.12%)	0 (0.00%)
InputHandler::check	1120 (99.12%)	0 (0.00%)
Generatehandler::generate	1089 (96.37%)	1 (0.09%)
Generatehandler::holehole	537 (47.52%)	1 (0.09%)
Generatehandler::input	356 (31.50%)	1 (0.09%)

当前视图: 函数			
函数名	CPU 总计[单位...	自 CPU [单位, %]	模块
shudu.exe (PID: 15824)	1130 (100.00%)	0 (0.00%)	shudu.exe
[外部代码]	1130 (100.00%)	9 (0.80%)	多个模块
_srt_common_main	1121 (99.20%)	0 (0.00%)	shudu.exe
_srt_common_main_seh	1121 (99.20%)	0 (0.00%)	shudu.exe
mainCRTStartup	1121 (99.20%)	0 (0.00%)	shudu.exe
InputHandler::check	1120 (99.12%)	0 (0.00%)	shudu.exe
invoke_main	1120 (99.12%)	0 (0.00%)	shudu.exe
main	1120 (99.12%)	0 (0.00%)	shudu.exe
UnitTest	1120 (99.12%)	0 (0.00%)	shudu.exe
Generatehandler::generate	1089 (96.37%)	1 (0.09%)	shudu.exe
Generatehandler::generateRandomNumber	538 (47.61%)	11 (0.97%)	shudu.exe
Generatehandler::holehole	537 (47.52%)	1 (0.09%)	shudu.exe
Generatehandler::input	356 (31.50%)	1 (0.09%)	shudu.exe
std::uniform_int<int>::_Eval<std::mersenne_t...	332 (29.38%)	1 (0.09%)	shudu.exe

D:\LessonProjects\shudu\shudu\main.cpp: 14

```

86     strcpy_s(argv4[3], 30, "10");
87
88
89     strcpy_s(argv5[0], 30, "shudu.exe");
90     strcpy_s(argv5[1], 30, "-n");
91     strcpy_s(argv5[2], 30, "100");
92     strcpy_s(argv5[3], 30, "-m");
93     strcpy_s(argv5[4], 30, "3");
94     //-n 100 -m 3
165 (14.60%) 95     inputs.check(argc3, argv5);
96     //-n 100 -m 1
74 (6.55%) 97     strcpy_s(argv5[4], 30, "1");
98     inputs.check(argc3, argv5);
99     //-n 100 -m 2
118 (10.44%) 100    strcpy_s(argv5[4], 30, "2");
101    inputs.check(argc3, argv5);
102    //-n 100 -m 6
103    strcpy_s(argv5[4], 30, "6");
104    inputs.check(argc3, argv5);

```

可以看到，总体来说，`std::mersenne_twister<unsigned int ...>::_Refill_upper` 这个函数是耗时最多的函数。

`std::mersenne_twister<unsigned int ...>::_Refill_upper` 函数是 C++ 标准库中的 `std::mersenne_twister` 类模板的私有成员函数之一。这个函数主要用于内部生成随机数序列。

`std::mersenne_twister` 是一个伪随机数生成器，它基于梅森旋转算法（Mersenne Twister Algorithm），被广泛用于生成高质量的随机数序列。该算法的实现分为几个关键步骤，其中 `_Refill_upper` 函数在其中扮演了重要的角色。

具体而言，`_Refill_upper` 函数的功能是填充 Mersenne Twister 算法的内部状态数组（state array）中的上半部分（即高位部分）。Mersenne Twister 算法使用一个很长的状态数组作为内部状态，在生成每个随机数时，需要对这个数组进行更新和变换。`_Refill_upper` 函数负责更新数组中的上半部分，以确保生成的随机数具有良好的统计特性和周期性。

由于 `_Refill_upper` 函数是私有成员函数，意味着它只能在类内部使用，而无法直接从外部调用。它在 Mersenne Twister 算法的内部执行，并在需要时自动被调用，以满足随机数的生成需求。

其次便是IO，在我们的代码中，输入输出较为频繁。

对具体函数耗时进行分析，可以发现 `Generatehandler::generate` 函数耗时较多，这是因为几乎每一次调用都需要使用 `generate` 函数进行生成，这是一个总的入口。其次就是 `generateRandomNumber` 和 `holehole`，分别为生成随机数函数和挖空函数。

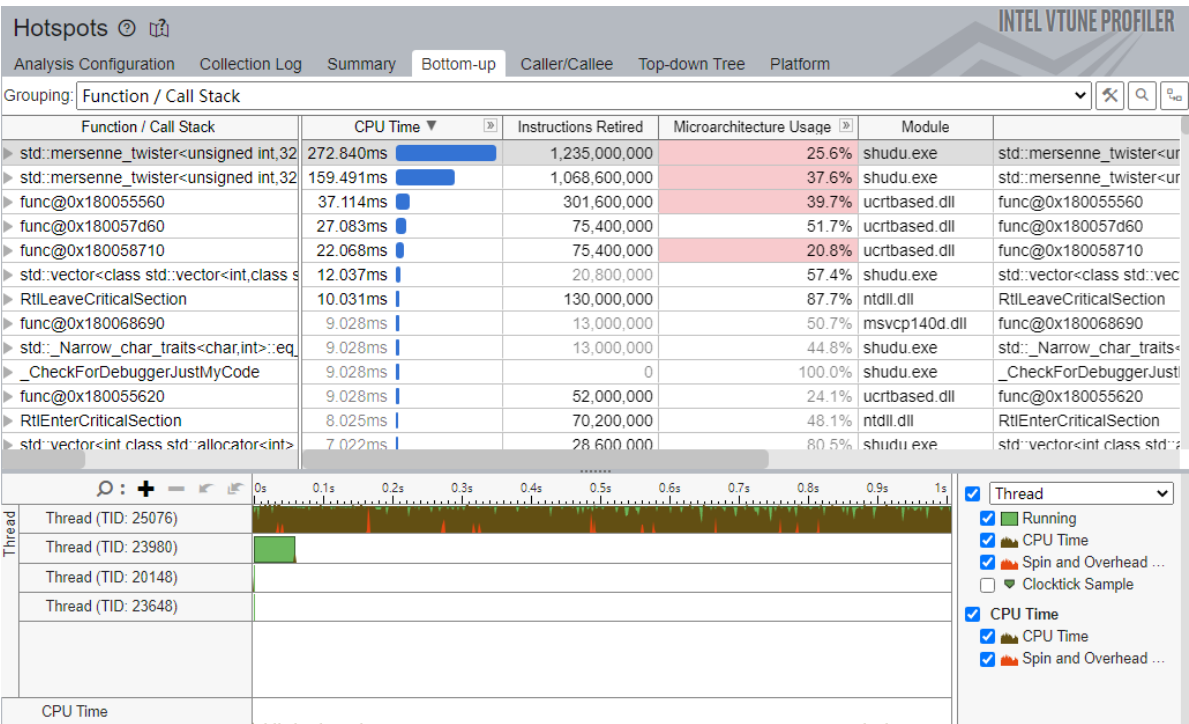
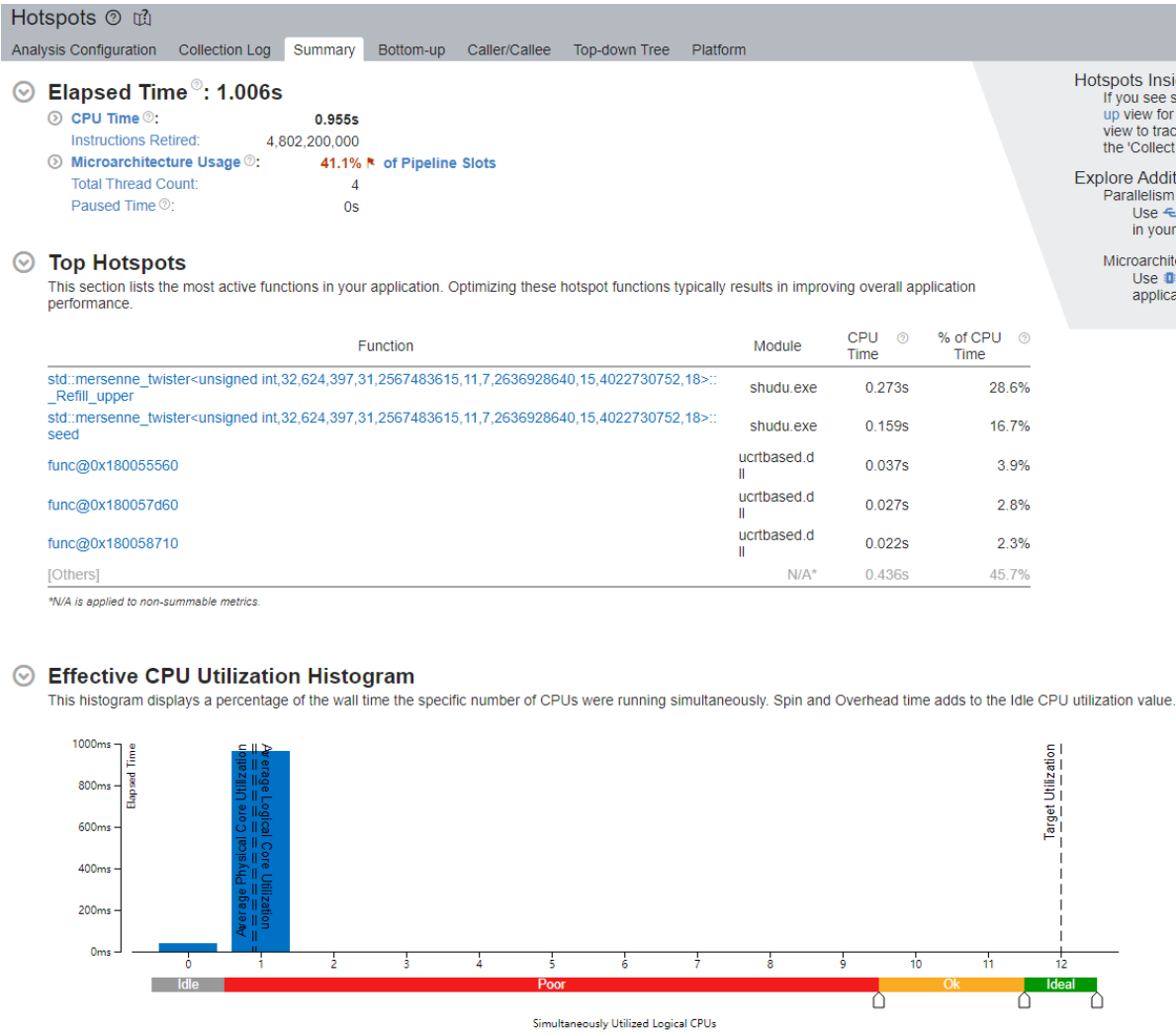
通过性能分析，我们发现了算法的瓶颈所在，为之后进一步提升提供了方向。



## 2 VTune Profiler

除此之外，我们还尝试使用VTune进行性能分析。

我们使用Hotspots进行分析。



分析的结果与性能探测器相同，在产生随机数方面耗费了大量时间。未来我们将进一步改进这个问题，以提升程序的性能。

# 总结

---

通过这次作业，我们学到了以下内容：

1. 学习和应用算法：在编写数独程序的过程中，我们学到了并应用了许多算法。生成数独游戏和求解数独游戏都需要复杂的算法，例如回溯算法或者约束传播算法。我们通过实际编码和调试的过程，加深了对这些算法的理解和掌握。
2. 挖空难度控制：挖空范围的限制是确保数独游戏具有一解且不易解决的重要一步。我们成功地实现了这个功能，并提供了不同难度级别供用户选择。
3. 错误处理和异常情况：在实际应用中，不可避免地会出现各种错误和异常情况。我们遇到了一些挑战，例如输入错误的数独游戏或无解的情况。我们的程序正确地处理这些异常情况，并向用户提供有用的错误信息。
4. 代码质量和可维护性：开发一个功能完整的程序不仅仅意味着它能够正确运行，还要考虑到代码的质量和可维护性。我们思考了如何组织代码、使用合适的数据结构、遵循良好的编码规范等方面。这些都是提高软件质量和可维护性的重要因素。

总的来说，完成数独命令行程序是一个很好的实践项目，我们通过这个过程积累了许多宝贵的经验和技能。无论是算法设计、用户交互、错误处理还是代码质量，我们都有许多心得体会。我们将会吸收这次作业带来的经验，争取在实际的软件工程开发中做得更出色。