



南開大學  
Nankai University

计算机学院  
软件工程实验报告

结对编程

姓名：平世龙，姚翔载

年级：2020 级

专业：计算机科学与技术

2023 年 6 月 29 日

# 目录

<b>1</b>	<b>用户手册</b>	<b>2</b>
1.1	引言	2
1.1.1	编写目的	2
1.1.2	项目背景	2
1.2	软件概述	2
1.2.1	目标	2
1.2.2	功能	3
1.2.3	运行环境与安装流程	3
1.3	数据	3
1.3.1	数据背景	3
1.3.2	输入	3
1.3.3	输出	4
1.4	运行方法	4
1.5	运行参数	4
1.6	出错与恢复	4
1.7	源文件一览	4
<b>2</b>	<b>编程规范</b>	<b>5</b>
<b>3</b>	<b>质量分析</b>	<b>6</b>
<b>4</b>	<b>单元测试</b>	<b>7</b>
<b>5</b>	<b>覆盖统计</b>	<b>7</b>

# 1 用户手册

## 1.1 引言

### 1.1.1 编写目的

本用户手册旨在为用户提供使用数独程序的详细说明和指导。该程序旨在帮助用户解决数独难题，提供数独游戏的娱乐和挑战。通过本手册，您将了解程序的功能、界面和使用方法，以便能够充分利用数独程序并提高解决数独问题的效率。无论您是打算使用数独程序解决难题还是寻求数独游戏的乐趣，本手册都将提供您所需的信息和指导。

通过本手册，您将了解以下内容：

1. 程序的安装和配置要求
2. 软件功能与目标的介绍
3. 软件具体输入输出的格式
4. 程序运行方法与参数说明
5. 源文件的说明

### 1.1.2 项目背景

1. 项目名称：数独控制台小程序
2. 开发者：本软件工程小组
3. 用户：南开大学师生

## 1.2 软件概述

### 1.2.1 目标

本软件的目标是提供一个简单轻量的数独游戏控制台程序，以满足软件工程课程对程序设计作业的需求。通过该程序，用户可以通过控制台输入参数并运行程序，实现数独游戏的生成、终盘的生成以及游戏的求解等主要功能。为了达到目标，本软件注重以下几个方面：

1. 简单轻量：本软件设计追求简洁性和轻量化，使得用户可以快速上手并且能够顺利完成所需的数独游戏操作。用户不需要安装复杂的图形界面或其他依赖项，只需在控制台输入相应的命令即可进行游戏操作。
2. 数独功能：软件主要聚焦于数独游戏的核心功能。用户可以通过输入参数来生成新的数独游戏，包括选择游戏的难度级别、指定已知数字的数量等。程序还提供了生成数独终盘的功能，用户可以在生成的游戏基础上进行游戏求解。
3. 用户友好性：尽管本软件是一个控制台程序，但仍致力于提供用户友好的操作体验。通过清晰的命令提示和详细的说明，用户可以轻松理解和使用程序的各项功能。同时，程序会对用户输入进行合理性检查，并给出相应的错误提示，以确保用户能够顺利完成操作。

4. 符合作业需求：本软件旨在满足软件工程课程对程序设计作业的要求，包括输入参数的处理、游戏生成和求解算法的实现等。通过使用本软件，用户能够学习和实践相关的程序设计概念和技术，提高编程能力和理解算法的能力。

### 1.2.2 功能

本软件的功能包括：

1. 生成数独终盘：用户可以通过软件生成指定数量的数独终盘，并将其输出到文本文件中。这些生成的终盘可作为数独游戏的基础，供用户挑战和解答。
2. 生成数独游戏：软件支持通过挖空法生成数独游戏。用户可以指定挖空的数量范围，从而控制游戏的难度。此外，用户还可以选择生成不同难度级别的游戏，包括简单、中等和困难。对于需要解唯一的游戏，软件也提供了相应的选项。
3. 数独游戏求解：用户可以从文本文件中读取数独游戏，并使用软件进行求解。软件将自动解析数独游戏并找到解，然后将解输出到另一个文本文件中。这个功能可以帮助用户解答难题或验证自己的解答是否正确。

通过这些功能，用户可以充分享受数独游戏的乐趣和挑战。无论是生成终盘、创建自定义难度的游戏，还是使用软件来解答难题，本软件都为用户提供了便捷的操作方式和功能选项。用户可以根据自己的喜好和需求，灵活地使用软件，并享受数独游戏带来的智力训练和娱乐体验。

### 1.2.3 运行环境与安装流程

Win10 系统下测试正常运行。本软件不需要复杂的安装流程，只需要保证软件的可执行文件所在位置下，所需的相对输入输出路径均存在即可。

## 1.3 数据

### 1.3.1 数据背景

在本软件下，数据的输入与输出均以 txt 文件存储。数据完全按照数独游戏的数字进行存储，并且数据的存储适当穿插了空格与空行以增加对用户的可读性。

### 1.3.2 输入

输入文件只包含 txt 文件，其中：

1. 每个数独游戏为 9 行 9 列的数字 + 符号 + 空格组合，其中数字代表游戏中的数字，符号 ‘\$’ 代表数独游戏需要填入的空白格。如果是生成的终局或解，则没有符号 ‘\$’。每个数字与空白格左右均与相邻的数字或空白格以空格隔开，这样组成了一个数独游戏。注意每行最右边以空格结尾。例如：

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

2. 一个文件中有若干个游戏，每个游戏之间以空行分割，文件以空行结束。

### 1.3.3 输出

除了参数'-s' 之外，输出文件与输入文件的组织形式相同。

'-s' 的参数意义为读取游戏、生成解并输出到文件中，其输出文件也为 txt 文件，其中输出文件的每个游戏中只包含求解后的所有数字，不包含符号 '\$'。此外，为了处理一个数独游戏有多个解的情况，'-s' 输出文件总的组织形式如下：对应读取的文件中的每一个数独游戏，首先输出：'Number of solutions: \$n'，其中 n 为该游戏的解的个数；然后在第二行，依次按照与输入文件格式相同的方法输出所有的解。循环上述步骤直到读取的文件中所有游戏的解的个数和所有解都输出完毕。

## 1.4 运行方法

进入到包含可执行文件的路径，唤起 windows 命令行，输入带参数的指令运行可执行文件即可。

## 1.5 运行参数

运行参数包含-c/-m/-r/-u/-s 所示，说明如表1和2所示。几种可能的用法示例：

生成 20 个挖空数在 20 到 55 之间的数独游戏：sudoku.exe -n 20 -r 20 55

表示生成 1000 个简单数独游戏：sudoku.exe -n 1000 -m 1

从 game.txt 读取若干个数独游戏，并给出其解答，生成到 sudoku.txt 中：sudoku.exe -s game.txt

几种错误的用法示例：

参数超出范围：sudoku.exe -c 9999999

部分参数未正确组合使用：sudoku.exe -u

参数名	参数意义	说明	范围限制
-c	数独终盘的数量	不符合范围限制则报错	1-1000000
-m	需要生成游戏的难度	不符合范围限制则报错，只能与 n 组合使用	'1-3，整数
-r	生成游戏中挖空的数量范围，为 "a~b" 形式，其中 a、b 为整数，表示生成游戏的挖空数在 a 到 b 之间	不符合范围限制则报错，只能与 n 组合使用	$20 \leq a < b \leq 55$ ，整数
-u	生成游戏的解唯一	不符合范围限制则报错，只能与 n 组合使用	无
-s	需要解的数独棋盘文件路径	绝对或相对路径，路径不存在则报错	存在的绝对或相对路径

表 1: 参数说明 1

## 1.6 出错与恢复

若出现“运行参数”一节所述的报错情况，不需要特殊的出错恢复手段，确定路径等无误后命令行重新输入指令、运行程序即可。

## 1.7 源文件一览

源文件的结构如图1.1所示：

其中，files 为默认的 txt 文件输出路径；sudoku 文件夹下存放着源文件与可执行文件。其中：

参数名	执行成功说明	报错说明
-c	成功输出到 txt 文件后， 命令行显示成功信息： "save{rel_path}"，其中 rel_path 为存储到的 txt 文件的相对路径	超出范围：跳过生成步骤，无生成成功提示
-m	同上	同上，若未组合使用，报错"n error"
-r	同上	同上，若未组合使用，报错"n error"
-u	同上	同上，若未组合使用，报错"n error"
-s	成功输出到 txt 文件则无任何提示	若存储的 txt 文件父级路径不存在，则报错

表 2: 参数说明 2

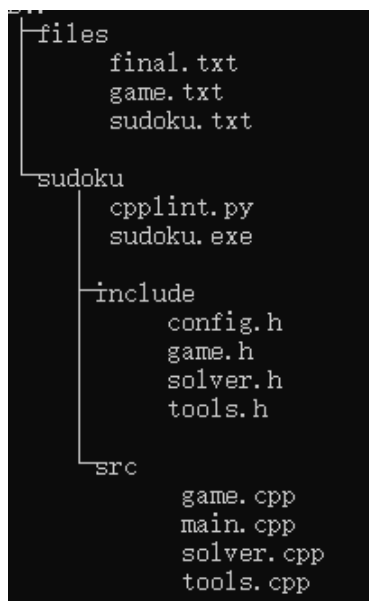


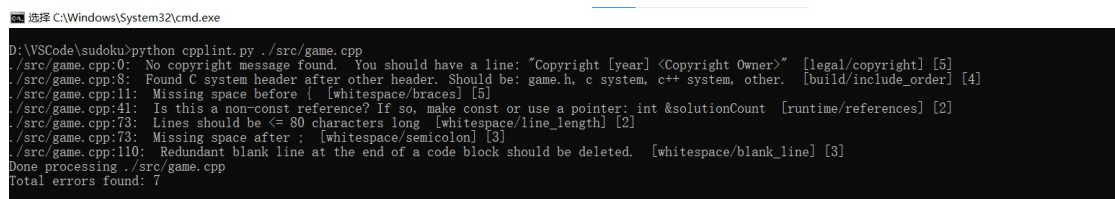
图 1.1: 文件结构

1. files 为默认的 txt 文件输出路径
2. sudoku 文件夹下存放着可执行文件，为程序的主体
3. cpplint 文件是用于静态代码分析
4. include 文件夹下包含着编译所需要的头文件，其中：config.h 为各种宏定义等的声明；game.h 为生成游戏与终局的函数声明；solver.h 为求解游戏的相关函数声明；tools.h 为其他函数可能用到的一些辅助函数声明。
5. src 文件夹下，game、solver 与 tools 三个文件为头文件声明函数的具体定义，包含了生成游戏、求解游戏、输出到文件等功能的具体逻辑。main.cpp 中则主要是运行参数的接收与处理，从而实现不同的功能。

## 2 编程规范

使用 CppLint 检测并消除代码中的不规范。

使用 CppLint 时检测出代码一系列不规范。以 game.cpp 为例（为使结果清楚简洁已将同一类型不规范消除至一个）：



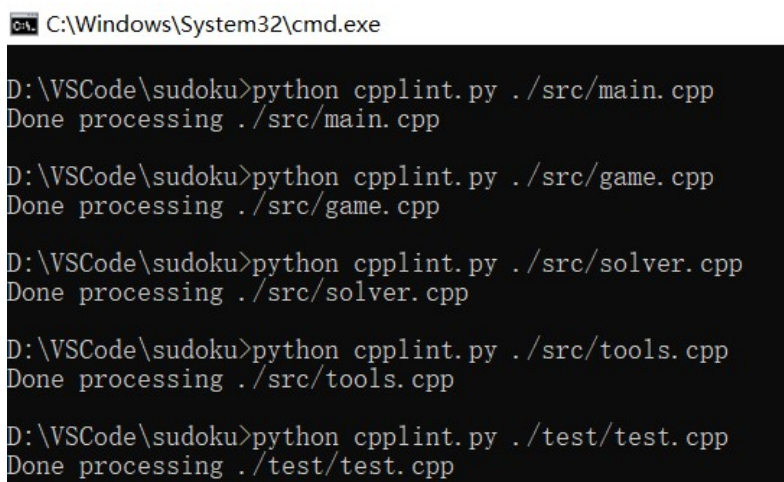
```
C:\Windows\System32\cmd.exe
D:\VSCode\sudoku>python cpplint.py ./src/game.cpp
./src/game.cpp:0: No copyright message found. You should have a line: "Copyright [year] <Copyright Owner>" [legal/copyright] [5]
./src/game.cpp:8: Found C system header after other header. Should be: game.h, c system, c++ system, other. [build/include_order] [4]
./src/game.cpp:11: Missing space before { [whitespace/braces] [5]
./src/game.cpp:41: Is this a non-const reference? If so, make const or use a pointer: int &solutionCount [runtime/references] [2]
./src/game.cpp:73: Lines should be <= 80 characters long [whitespace/line_length] [2]
./src/game.cpp:73: Missing space after ; [whitespace/semicolon] [3]
./src/game.cpp:110: Redundant blank line at the end of a code block should be deleted. [whitespace/blank_line] [3]
Done processing ./src/game.cpp
Total errors found: 7
```

图 2.2: 编程不规范

包括:

1. 无版权信息
2. 缺少空格
3. 代码行过长
4. include 头文件顺序错误
5. 不允许函数参数为 non-const 的 reference
6. 代码块末尾的多余空行未删除

规范编程后实验结果如下:



```
C:\Windows\System32\cmd.exe
D:\VSCode\sudoku>python cpplint.py ./src/main.cpp
Done processing ./src/main.cpp

D:\VSCode\sudoku>python cpplint.py ./src/game.cpp
Done processing ./src/game.cpp

D:\VSCode\sudoku>python cpplint.py ./src/solver.cpp
Done processing ./src/solver.cpp

D:\VSCode\sudoku>python cpplint.py ./src/tools.cpp
Done processing ./src/tools.cpp

D:\VSCode\sudoku>python cpplint.py ./test/test.cpp
Done processing ./test/test.cpp
```

图 2.3: 编程规范

### 3 质量分析

使用 CppCheck 完成质量分析。使用 CppCheck 对目录或文件进行分析后, 会得到代码中可能需要优化的地方, 例如:

代码中变量 r 的作用域可缩减至 for 循环中。CppCheck 同时会提供信息帮助理解。

使用 CppCheck 质量分析代码并消除警告后结果如下 (结果中消除引入标准头文件之外的所有警告和错误):

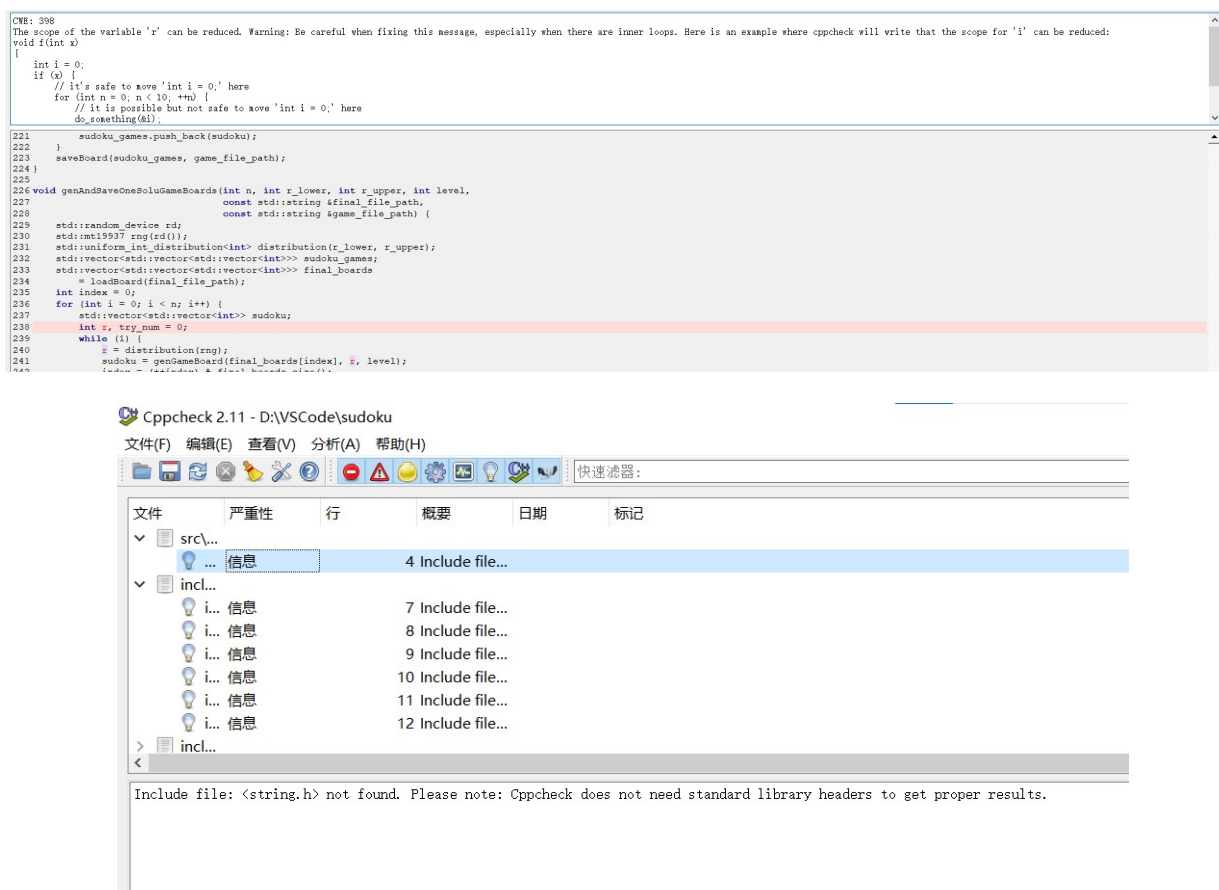


图 3.4: 质量分析

## 4 单元测试

使用 GoogleTest 完成单元测试。GoogleTest 是 Google 开源的一个跨平台 C++ 单元测试框架。代码分别就生成游戏的正确性、挖空范围、游戏难度、唯一解展开测试。

1. 正确性。分别生成不同数量的唯一游戏，求解后测试解的正确性。
2. 挖空范围。分别生成不同挖空数量的游戏，生成后测试挖空数量是否在设置的范围内。
3. 游戏难度。分为生成不同难度的游戏，求解后测试游戏难度是否与初始设置一致。
4. 唯一解。分别生成不同数量的唯一解游戏，求解后测试解的数量是否唯一。

实验结果如下：



```
C:\Windows\System32\cmd.exe

D:\VSCode\sudoku\cmake-build-debug>sudoku.exe
[=====] Running 11 tests from 1 test suite.
[-----] Global test environment set-up.
[-----] 11 tests from TestCases
[ RUN     ] TestCases.correctness_case_1
[ OK      ] TestCases.correctness_case_1 (5 ms)
[ RUN     ] TestCases.correctness_case_5
[ OK      ] TestCases.correctness_case_5 (921 ms)
[ RUN     ] TestCases.correctness_case_10
[ OK      ] TestCases.correctness_case_10 (2817 ms)
[ RUN     ] TestCases.range_case_1
[ OK      ] TestCases.range_case_1 (4 ms)
[ RUN     ] TestCases.range_case_2
[ OK      ] TestCases.range_case_2 (4 ms)
[ RUN     ] TestCases.level_case_1
[ OK      ] TestCases.level_case_1 (416 ms)
[ RUN     ] TestCases.level_case_2
[ OK      ] TestCases.level_case_2 (23 ms)
[ RUN     ] TestCases.level_case_3
[ OK      ] TestCases.level_case_3 (28 ms)
[ RUN     ] TestCases.uniqueness_case_1
[ OK      ] TestCases.uniqueness_case_1 (4 ms)
[ RUN     ] TestCases.uniqueness_case_5
[ OK      ] TestCases.uniqueness_case_5 (548 ms)
[ RUN     ] TestCases.uniqueness_case_10
[ OK      ] TestCases.uniqueness_case_10 (25056 ms)
[-----] 11 tests from TestCases (29901 ms total)

[-----] Global test environment tear-down
[=====] 11 tests from 1 test suite ran. (29912 ms total)
[ PASSED ] 11 tests.
```

图 4.5: 单元测试

## 5 覆盖统计