数独程序-质量分析报告

小组成员：牟迪 宋彦艳

学号：2011360 2013471

班级：徐思涵老师班

专业：计算机科学与技术

本次实验中，完成了一个能够生成数独游戏，并求解数独问题的控制台程序。参照实验指导，完成了对于原始程序的编写。并对程序进行了质量分析，并加以改进。程序质量分析的具体过程与结果将在后文进行介绍。

一.实验环境

本次实验的相关环境说明如下：

- 编程语言：C++14标准
- 开发环境：VisualStudio 2022 Community 2022 (64 位) 17.2.6
- 运行环境：64bit，Windows10家庭中文版

二.代码审查

本次实验中对代码进行质量分析的第一个环节是人工进行代码的review，代码review可以帮助提高代码的可读性，可维护性，可测试性和可扩展性，提高软件的质量，效率和安全性。本次实验中小组规定由一人完成的代码必须由另一人进行评审检查后，方能合并到git分支。在进行代码评审的过程中，一方面检查代码的可读性，确保评审人能够理解代码含义。同时也对于代码中的部分安全隐患进行了初步的处理。对于不当的函数调用进行了更正。经过代码审查，得到了正确且可运行的初版程序代码，修正了由Visual Studio默认代码分析中给出的潜在数值溢出，错误类型转换等警告信息。

三.静态分析

在进行代码审查之后，采用sonarlint工具对代码进行进一步的静态分析。sonarlint可以在开发环境中提供对代码的实时检测和提供代码质量问题反馈。本次实验中，将sonarlint集成至visualstudio环境中，采用sonarSource提供的静态代码分析规则对程序代码进行检查，并对于发现的警告问题进行处理。在这一过程中，解决的问题包括但不限于：

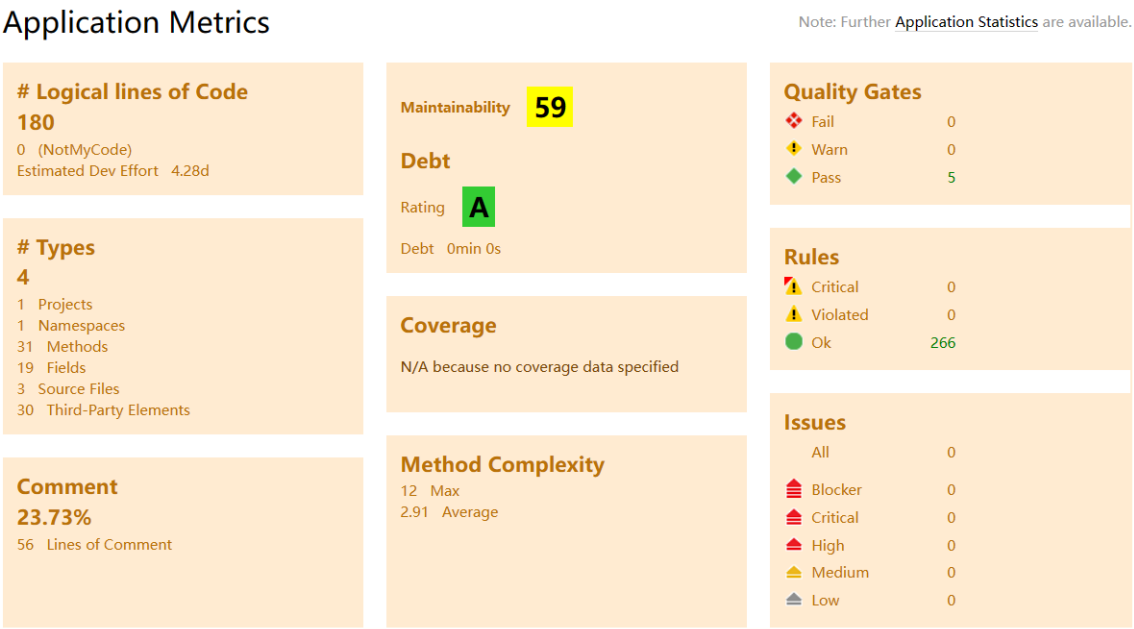
- 以类的单例模式替代必须声明为const的全局变量
- 更改数组的声明形式，采用C++中新的数组标准而非C语言中的标准
- 采用更规范的随机数生成引擎来进行随机数的生成，避免多线程下的风险
- 简化代码逻辑，避免同一区域出现三层以上的判断逻辑
- 采用智能指针来替代new语法，将内存的分配与回收转换为自动行为
- 将部分类对象声明时的显示类型声明转换为auto
- 简化并拆分部分函数逻辑，降低单个函数的复杂程度

在根据质量分析结果进行代码的修正之后，将程序中的警告数量由59减少至0

四.采用Cppdepend进行代码度量

在使用sonarlint进行静态分析后，已经消除了程序中所有的警告信息。接下来采用cppdepend工具对程序进行进一步的衡量。cppdepend中提供了对于代码复杂性，依赖性，重复性，代码注释等相关的规则。可以对程序代码的质量进行量化分析。

应用程序指标



如上为程序的各项综合指标分析结果。其中涉及了代码的总行数，代码的注释率，在右侧信息中，程序可以完全通过质量门已经全部分析规则的拣择，且不存在issues，程序的初始可维护性评分为54分，经过了进一步的结构调整与优化，将可维护性评分提高至59分。在综合的debt rating中，在5级评分中获得了A的评级。程序代码中共包含了33个方法。其中最高的复杂度为12，所有方法的平均复杂度为2.91.

质量门评估

Quality Gates summary

11 0 0

Name	Value	Group
Percentage Code Coverage	N/A because no coverage data	Project Rules \ Quality Gates
Percentage Coverage on New Code	N/A because no coverage data	Project Rules \ Quality Gates
Percentage Coverage on Refactored Code	N/A because no coverage data	Project Rules \ Quality Gates
Blocker Issues	0 issues	Project Rules \ Quality Gates
Critical Issues	0 issues	Project Rules \ Quality Gates
New Blocker / Critical / High Issues	N/A because no coverage data	Project Rules \ Quality Gates
Critical Rules Violated	0 rules	Project Rules \ Quality Gates
Percentage Debt	0 %	Project Rules \ Quality Gates
New Debt since Baseline	N/A because no coverage data	Project Rules \ Quality Gates
Debt Rating per Namespace	0 namespaces	Project Rules \ Quality Gates
New Annual Interest since Baseline	N/A because no coverage data	Project Rules \ Quality Gates

Showing 1 to 11 of 11 entries

在cppdepend，默认进行了一些质量门的设置。在暂未部署程序测试的情况下，其中的部分质量门并没有起到作用，而在剩下的5项质量门评估中，程序可以通过全部的测试。

规则检查

Rules summary

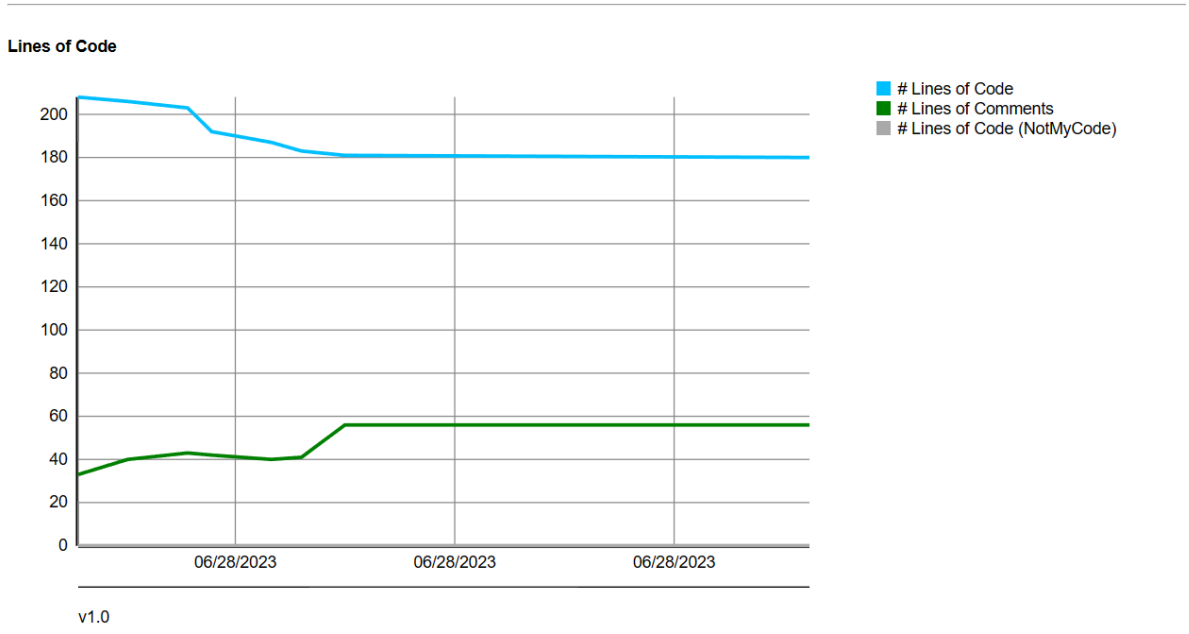
287 0 0

- Number of Rules or Queries with Error (syntax error, exception thrown, time-out): **0**
- Number of Rules violated: **0**

根据cppdepend中默认的规则匹配，当前的程序代码可以完全匹配266条预先设定的代码规范。无语法错误，抛出异常与超时警告。具体的代码规范参照附件1：代码检查规范.html

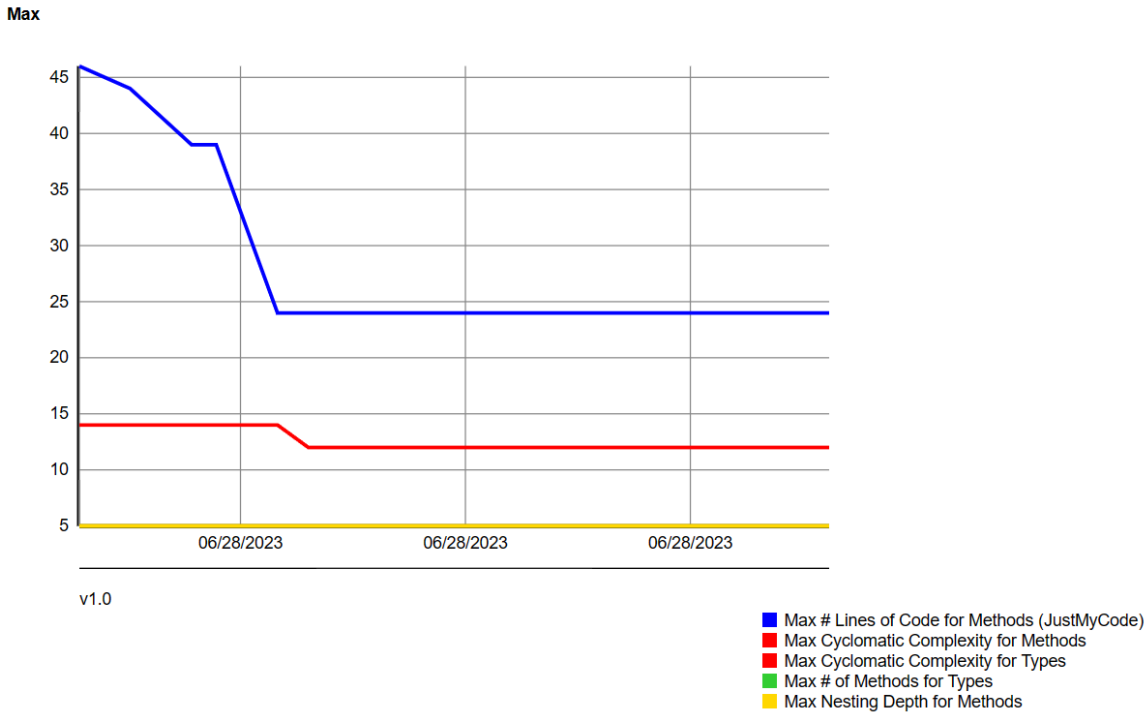
程序优化过程

代码总行数



在首次进行质量分析时，代码的总函数超过200行，在随后的改进中，对部分代码进行了优化，提高了代码的可读性，可维护性，并增加了注释说明。

最大复杂度

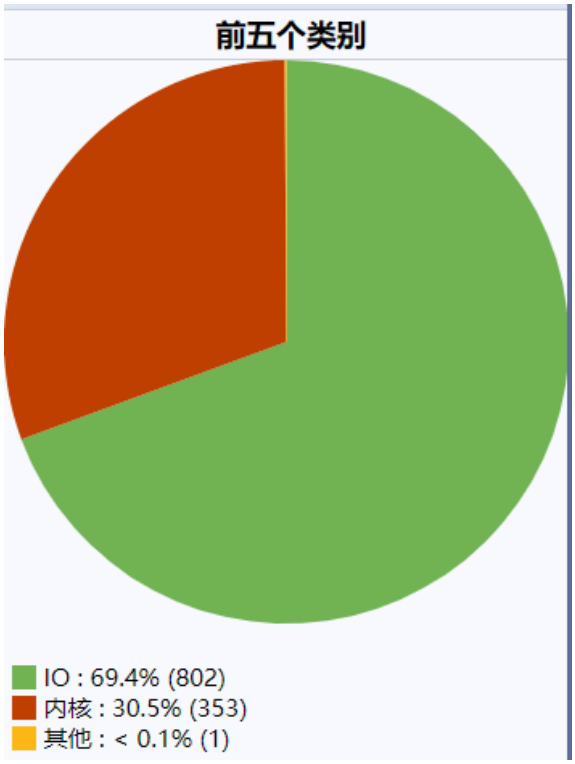


如上为历次分析下代码中最大复杂度的变化。其中蓝色部分代码了各个方法中的代码行数，红色部分代表了各个方法的循环复杂度。在进行质量分析后，对代码进行了优化，二者均有一定程度的下降。

五.性能剖析

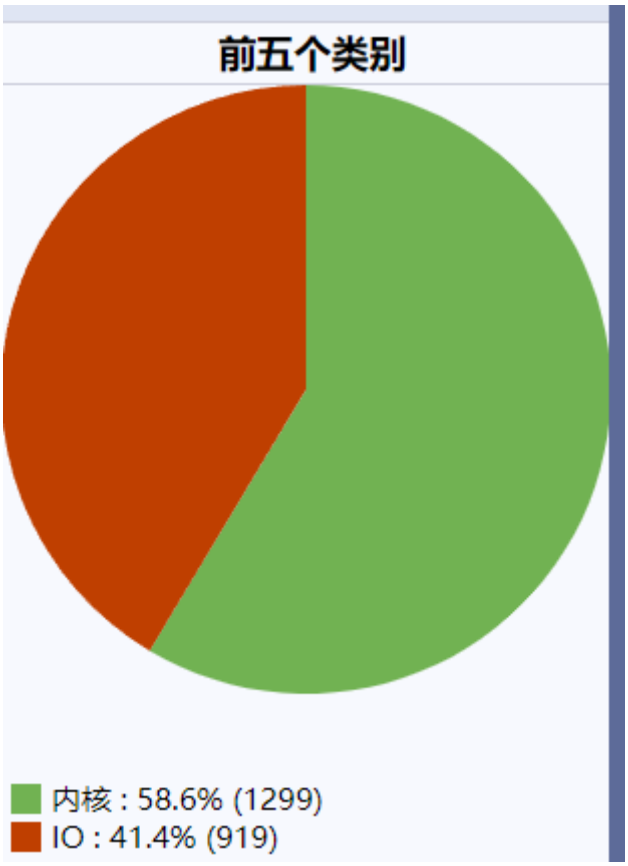
对于程序的性能剖析采用Visual Studio中性能探查器，对于程序运行时CPU使用率进行分析。测试过程中模拟两种使用场景，分别是创建1000个数独游戏与求解1000个数独游戏。其CPU使用情况如下所示：

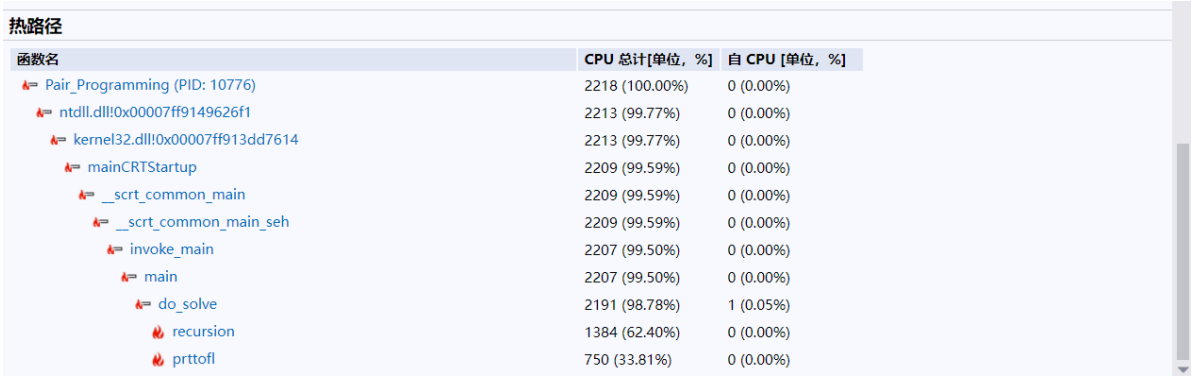
创建数独游戏



在执行创建1000个数独游戏的任务时，其总的CPU使用率维持在6%，而在程序执行种，可以发现，对于文件的IO操作占据了70%左右的CPU使用，说明此过程的性能瓶颈位于文件输出环节。

求解数独游戏





热路径

函数名	CPU 总计[单位，%]	自 CPU [单位，%]
Pair_Programming (PID: 10776)	2218 (100.00%)	0 (0.00%)
ntdll.dll!0x00007ff9149626f1	2213 (99.77%)	0 (0.00%)
kernel32.dll!0x00007ff913dd7614	2213 (99.77%)	0 (0.00%)
mainCRTStartup	2209 (99.59%)	0 (0.00%)
__scrt_common_main	2209 (99.59%)	0 (0.00%)
__scrt_common_main_seh	2209 (99.59%)	0 (0.00%)
invoke_main	2207 (99.50%)	0 (0.00%)
main	2207 (99.50%)	0 (0.00%)
do_solve	2191 (98.78%)	1 (0.05%)
recursion	1384 (62.40%)	0 (0.00%)
prttofl	750 (33.81%)	0 (0.00%)

在执行求解1000个数独游戏的过程中，程序CPU使用率维持在6%，根据扇形统计中可以得出，在执行求解任务下，占用CPU最高的部分转为了内核。在热路径统计下，也可以发现，与求解相关的do_solve函数与recursion函数排序较为靠前。