

**课程设计报告**

**题目：基于SAT的双数独游戏求解程序**

**课程名称： 程序设计综合课程设计**

**专业班级：计算机科学与技术2106班**

**学 号： U202115512**

**姓 名： 洪炜豪**

**指导教师： 纪俊文**

**报告日期： 2022.9.14**

**计算机科学与技术学院**

目 录

**任务书 3**

**1引言**

**1.1问题背景 4**

**1.2**研究意义 **4**

**1.3本文摘要 5**

**2 系统需求分析与总体设计**

**2.1系统需求分析 6**

**2.2系统总体设计 7**

**3 系统详细设计**

**3.1有关常量和变量定义 8**

**3.2有关数据结构的定义 8**

**3.3 主要算法设计 9**

**3.3.1 DPLL算法思想 9**

**3.3.2 CNF文件处理（CnfParser） 10**

**3.3.3 Sudoku（二进制数独游戏转化SAT问题）算法思想 10**

**3.3.4 Examination（sat求解结果正确性检验）算法思想 11**

**3.3.5 game算法思想 11**

**4系统实现与测试**

**4.1.1 软硬件环境 12**

**4.1.2 系统各部分函数阐述 12**

**4.1.2.1 DPLL 12**

**4.1.2.2 CnfParser 13**

**4.1.2.3 Sudoku 13**

**4.2系统测试 15**

**4.2.1 CNF算例求解（SAT模块） 15**

**4.2.2 数独游戏测试 22**

**4.2.3 答案检验系统测试 26**

**4.3 结果分析 28**

**4.4 复杂度分析 28**

**5 总结与展望**

**5.1全文总结 29**

**5.2工作展望 29**

**6 体会 30**

**参考文献 31**

**[附录A main.cpp 3](#_Toc2235613)2**

**[附录B CnfParser.cpp 3](#_Toc2235614)8**

**[附录C DPLL.cpp 4](#_Toc2235615)0**

**[附录D header.h 4](#_Toc2235616)5**

**[附录E Sudoku.cpp 4](#_Toc2235617)6**

**[附录F game.cpp 6](#_Toc2235617)0**

**[附录G Examination.cpp 6](#_Toc2235617)6**

**任务书**

* **设计内容**

SAT问题即命题逻辑公式的可满足性问题（satisfiability problem），是计算机科学与人工智能基本问题，是一个典型的NP完全问题，可广泛应用于许多实际问题如硬件设计、安全协议验证等，具有重要理论意义与应用价值。本设计要求基于DPLL算法实现一个完备SAT求解器，对输入的CNF范式算例文件，解析并建立其内部表示；精心设计问题中变元、文字、子句、公式等有效的物理存储结构以及一定的分支变元处理策略，使求解器具有优化的执行性能；对一定规模的算例能有效求解，输出与文件保存求解结果，统计求解时间。

* **设计要求**

要求具有如下功能：

**（1）输入输出功能：**包括程序执行参数的输入，SAT算例cnf文件的读取，执行结果的输出与文件保存等。(15%)

**（2）公式解析与验证：**读取cnf算例文件，解析文件，基于一定的物理结构，建立公式的内部表示；并实现对解析正确性的验证功能，即遍历内部结构逐行输出与显示每个子句，与输入算例对比可人工判断解析功能的正确性。数据结构的设计可参考文献[1-3]。(15%)

**（3）DPLL过程：**基于DPLL算法框架，实现SAT算例的求解。(35%)

**（4）时间性能的测量：**基于相应的时间处理函数（参考time.h），记录DPLL过程执行时间（以毫秒为单位），并作为输出信息的一部分。(5%)

**（5）程序优化：**对基本DPLL的实现进行存储结构、分支变元选取策略[1-3]等某一方面进行优化设计与实现，提供较明确的性能优化率结果。优化率的计算公式为：[(t-to)/t]\*100%,其中t 为未对DPLL优化时求解基准算例的执行时间，to则为优化DPLL实现时求解同一算例的执行时间。(15%)

**（6）SAT应用：**将数双独游戏[5]问题转化为SAT问题[6-8]，并集成到上面的求解器进行数独游戏求解，游戏可玩，具有一定的/简单的交互性。应用问题归约为SAT问题的具体方法可参考文献[3]与[6-8]。(15%)

# 1 引 言

## 问题背景

SAT问题又称命题逻辑公式的可满足性问题（satisfiability problem），是判断对合取范式形式给出的命题逻辑公式是否存在一个真值指派使得该逻辑公式为真。SAT问题是计算机科学与人工智能基本问题，是一个典型的NP完全问题。看似简单，却可广泛应用于许多实际问题如人工智能、电子设计自动化、自动化推理、硬件设计、安全协议验证等，具有重要理论意义与应用价值。对于SAT问题的研究从没有停止过，在1997年和2003年，H.Kautz与B.Selman两次列举出SAT搜索面临的挑战性问题，并于2011年和2007年，两度对当时的SAT问题研究现状进行了全面的综述。黄文奇提出的Solar算法在北京第三届SAT问题快速算法比赛中获得第一名。对SAT问题的求解主要有完备算法和不完备算法两大类。不完备算法主要是局部搜索算法，这种算法不能保证一定找到解，但是求解速度快，对于某些SAT问题的求解，局部搜索算法要比很多完备算法更有效。完备算法出现的时间更早，优点是可以正确判断SAT问题的可满足性，在算例无解的情况下可以给出完备的证明。对于求解SAT问题的优化算法主要有启发式算法、冲突子句学习算法、双文字监视法等。

## 研究意义

SAT问题是第一个被证明的NP完全问题，而NP完全问题由于其极大的理论价值和困难程度，破解后将会在许多领域得到广泛应用，从而在计算复杂性理论中具有非常重要的地位。由于所有的NP完全问题都能够在多项式时间内进行转换，那么如果SAT问题能够得到高效解决，所有的NP完全问题都能够在多项式时间内得到解决。对SAT问题的求解，可用于解决计算机和人工智能领域内的CSP问题（约束满足问题）、语义信息的处理和逻辑编程等问题，也可用于解决计算机辅助设计领域中的任务规划与设计、三维物体识别等问题。SAT问题的应用领域非常广泛，还能用于解决数学研究和应用领域中的旅行商问题和逻辑算数问题。许多实际问题，例如数据库检索、积木世界规划、超大规模集成电路设计、人工智能等都可以转换成SAT问题进而进行求解。可见对SAT问题求解的研究，具有重大意义。

## 本文摘要

依据Davis和Putnam在1960年提出的DPLL算法来求解合取范式，并基于原始的DPLL算法提出优化方案，并对优化前后的DPLL算法进行比较，做出一些思考。本文首先介绍关于SAT问题的背景、研究意义等，接着介绍系统需求分析与总体设计，然后说明系统详细设计，再者阐述系统实现与测试，最后作出总结与展望。

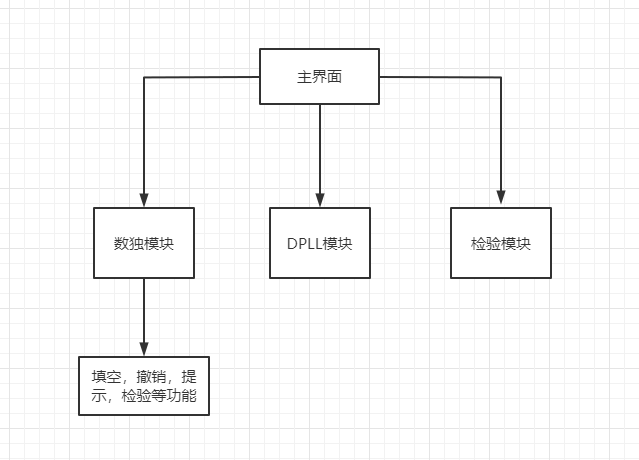
1. **系统需求分析与总体设计**

**2.1系统需求分析**

我们需要精心设计问题中变元、文字、子句、公式等有效的物理存储结构，基于DPLL过程实现一个高效SAT求解器，对于给定的中小规模算例进行求解，输出求解结果，统计求解时间。

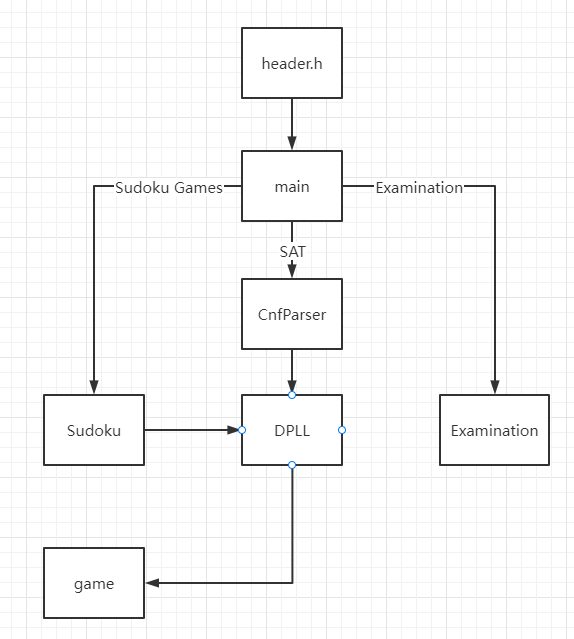
并基于数独问题转化为SAT问题的思路，使此高效的SAT求解器用于实现数独游戏生成与求解时间的优化，系统通过随机生成的棋盘进行求解，得到一个完整的数独解，据此通过挖洞法得到一个具有唯一解的数独棋盘。

要求设计基于SAT的双数独游戏求解程序，故应具备三种功能，一是可交互的数独游戏模块，二是对sat算例进行求解的dpll模块，三是对求解结果进行正确性检验的检验模块。



**2.2系统总体设计**

整个工程包含main.cpp、CnfParser.cpp、DPLL.cpp、header.h、Sudoku.cpp、game.cpp、Examination.cpp七个文件，其中main.cpp用于程序主界面的显示和主要函数的调用,header.h是声明关于求解SAT问题有关的函数以及一些变量的声明、数据元素类型的定义,DPLL.cpp是求解SAT问题有关函数的实现,Sudoku.cpp是关于求解数独以及数独游戏函数的声明,CnfParser.cpp是将cnf文件读入十字链表的实现,game.cpp是数独游戏的实现，Examination.cpp是对解析正确性的验证功能的实现。



系统模块结构图

**3系统详细设计**

## 3.1有关常量和变量定义

有关常量的定义：

#define ROW 9

#define COL 9

const string filePath ;

(统一的文件路径，位置改变时需要修改)

有关变量的定义：（仅列出主要变量）

int choice，holes，varnum;

string filename；

clock\_t startTime,endTime;

bool flag；

int \*result = new int[varnum];

Head\* LinkedList；

**3.2有关数据结构的定义**

typedef struct Data {

    int data = 0;

    Data \*nextData{};

}Data;

typedef struct Head{

    int num = 0;

    Data \*nextData{};

    Head \*nextHead{};

}Head;

//十字链表定义

**3.3 主要算法设计**

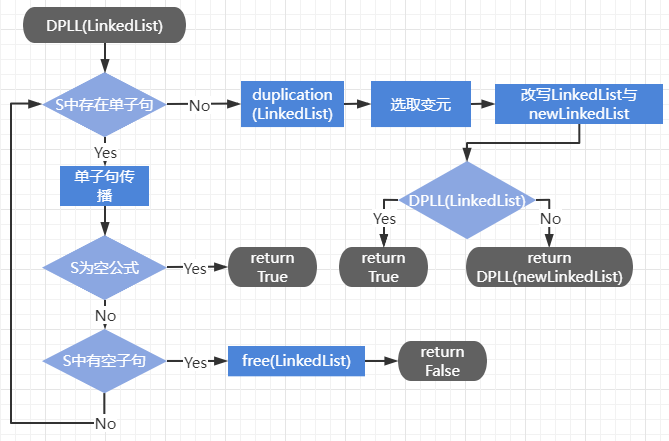
## 3.3.1 DPLL算法思想

DPLL算法是一种基于树的回溯算法，主要使用两种基本处理策略：

单子句规则。如果子句集*S*中有一个单子句*L*,那么*L*一定取真值，于是可以从*S*中删除所有包含*L*的子句（包括单子句本身），得到子句集*S*1，如果它是空集，则*S*可满足。否则对*S*1中的每个子句，如果它包含文字*¬L*,则从该子句中去掉这个文字，这样可得到子句集合*S*2。*S*可满足当且仅当*S*2可满足。单子句传播策略就是反复利用单子句规则化简*S*的过程。

分裂策略。按某种策略选取一个文字*L*.如果*L*取真值，则根据单子句传播策略，可将*S*化成*S*2；若*L*取假值（即*¬L*成立）时，*S*可化成*S*1.

根据上述规则可不断对公式化简，并最终达到终止状态，如下图所示：



基于单子句传播与分裂策略的DPLL算法可以描述为一个如后所示的递归过程DPLL( *S* ),为了优化执行效率，一般用非递归实现。

DPLL( *S*) :

/\* *S*为公式对应的子句集。若其满足，返回TURE；否则返回FALSE. \*/

{

while(*S*中存在单子句) {//单子句传播

在*S*中选一个单子句*L*；

依据单子句规则，利用*L*化简*S*；

if *S* = Φ return(TRUE);

else if (*S*中有空子句 ) return（FALSE）；

}//while

基于某种策略选取变元*v*；//策略对DPLL性能影响很大

if DPLL（*S* ∪*v* ）return(TURE);

return DPLL(*S* ∪¬*v*);

}

先采取的变元选取策略为选取十字链表的第一个子句的第一个变元，优化后采取的策略是选取最后一个最短子句的第一个变元。

采取的递归分裂策略是复制一个十字链表，加上变元的假值再进行化简。

## 3.3.2 CNF文件处理（CnfParser）

程序执行参数的输入，SAT算例cnf文件的读取。建立一个读取CNF文件函数Head\* initCnf(int &varNum, string &filename)，利用ifstream类的构造函数创建一个文件输入流对象，从文件中读取数据，最后将CNF文件中的子句存入十字链表中，返回十字链表表头指针。

## 3.3.3 Sudoku（二进制数独游戏转化SAT问题）算法思想

1.初始双数独生成算法：

先生成左上方数独：先通过随机数生成第一行，然后从第二行第一列开始，依次填入。设立标记数组vis[10]，遍历同行、同列、同3x3格子，若有出现i则给vis[i]标记。然后i从1开始遍历到9，当vis[i]未被标记时则将该格子填入数字i。直到第九行第九列填完，9x9数独生成完成。

然后声明第二个9x9数组，并传入重复部分的数据，再用同样的算法生成右下方数独。

2.初始游戏格局生成算法（挖洞法保证唯一解）：

建立一个调用DPLL算法判断挖洞法生成的数独棋盘是否有解的函数void putHole(int a[][COL], int b[][COL], int holeNum)。

设置三个随机变量whichSudoku、x、y来决定挖左上或者右下方数独的哪个洞，挖洞后的格子记为0，若要挖的洞已被挖过，则重新随机。成功挖洞后，首先，从1到9遍历该洞，当满足数独填入规则（行，列，9x9）且不与初始值重复时选取该值，否则跳过。其次，将该值带入要判断是否可以挖洞的位置，然后调用DPLL算法来检验填入该数后数独棋盘是否有解。若有解则恢复当前位置、挖洞数不变，若无解则累计挖洞数加一，当累计挖洞数达到目标挖洞数时，while循环结束，标准双数独游戏格局完成。

3.数独格局转换成对应的SAT问题的CNF文件算法

建立将数独问题转化成SAT问题的主函数string getCnf(int a[][COL], int b[][COL])。

然后分别依据每空已填入的值、每空取值约束、每行取值约束、每列取值约束、每小方块取值约束和公共部分约束共六个部分，将数独问题转换成对应的SAT问题的CNF文件。

SAT公式CNF文件中，一般变元是从1进行连续编码的，可以将语义编码转换为自然顺序编码，公式为：ij n → (i-1)\*81+(j-1)\*9+n。当按自然编码对数独游戏对应的CNF公式求解后，可设计逆变换公式将解解析为对应的游戏填充方案，完成填充，或给游戏玩家给予每一步填充的正误提示。

## 3.3.4 Examination（sat求解结果正确性检验）算法思想

为了检验sat所得出答案的正确性，我们需要将所得出的解代回原子句中进行检验。故先将res文件中的解读入到result数组中，再将cnf文件的子句读入数组bv[][]中，且当子句中有变元的值存在于result数组中，将clause\_value[第几子句]标记为true。最后遍历输出clause\_value，若均为真值，说明解析所得答案完全正确。

## 3.3.5 game算法思想

从“sudoku.txt”读入数独游戏初始格局至数组a,b；从result数组通过逆变公式得到答案数组a1,b1。done[2][10][10]数组记录玩家已填入的位置，方便查阅。玩家写入或撤回答案时改变a,b数组相应位置的值，通过比对玩家已填入的值和答案数组对应位置的值检查答案的正误。

**4系统实现与测试**

**4.1系统实现**

## 4.1.1 软硬件环境

1 硬件环境：

处理器 11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz 2.69 GHz

机带 RAM 16.0 GB (15.8 GB 可用)

系统类型 64 位操作系统, 基于 x64 的处理器

2 软件环境：

版本 Windows 10 家庭中文版

版本号 21H2

安装日期 ‎2021/‎8/‎5

操作系统内部版本 19044.2006

编辑环境 Visual Studio Code: 使用Makefile编译

Electron: 19.0.12

Chromium: 102.0.5005.167

Node.js: 16.14.2

V8: 10.2.154.15-electron.0

OS: Windows\_NT x64 10.0.19044

## 4.1.2 系统各部分函数阐述

## 4.1.2.1 DPLL

void DestroyLinkedList(Head\* LinkedList)

化简失败时，释放十字链表所占空间，优化内存。

bool isEmptyClause(Head\* LinkedList)

判断子句集中是否有空子句，若有，则返回true。

Head\* findSingleClause(Head\* LinkedList)

查找子句集中单子句，若找到则返回头结点指针，否则返回空指针。

Head\* addSingleClause(Head\* LinkedList,int var)

在子句集的第一句前插入数据为var的单子句。

void deleteHead(Head\* Clause,Head \*&LinkedList)

在子句集中删除子句Clause。

void deleteData(int object,Head \*&LinkedList)

在子句集中删除含有该变元的子句或者在含有该变元假值的子句中删除该变元假值。

Head\* duplication(Head\* LinkedList)

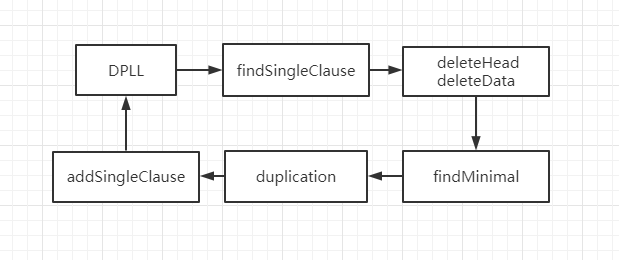
复制一个完整的十字链表并返回对应指针。

Data\* findMinimal(Head\* LinkedList)

返回子句集中最短子句。

bool DPLL(Head\* LinkedList,int\* result,int varNum)

DPLL函数，函数调用过程如下图，算法思想见上文：



## 4.1.2.2 CnfParser

ifstream openFile(string file)

利用ifstream类的构造函数创建一个文件输入流对象

Head\* buildLinkedList(ifstream& fis,int clauseNum)

将cnf文件输入十字链表

void printLinkedList(Head\* HEAD)

输出读入的cnf字句

Head\* initCnf(int &varNum, string &filename)

打开cnf文件，先读入变元、子句数量，再将子句读入链表中，并返回头指针

## 4.1.2.3 Sudoku

void randomFirstRow(int a[], int n)

随机生成9x9数独的第一行

bool fillLast(int a[][COL], int row, int col)

用递归的方式，从第二行第一格开始，遵循数独规则填完9x9数独的剩余8行

void createSudoku(int a[][COL])

生成9x9数独的函数，调用randomFirstRow和fillLast

void putHole(int a[][COL], int b[][COL], int holeNum)

挖洞法生成数独初始格局的函数，算法见上文

string getCnf(int a[][COL], int b[][COL])

将初始数独格局转化为cnf算例并保存的函数，算法见上文

void printSudoku(int a[][COL], int b[][COL])

将初始数独格局存入文件的函数，便于后续游戏模块的读取

string initSudoku(int holes)

{

    int sudoku1[ROW][COL] = {0};

    int sudoku2[ROW][COL] = {0};

    int holeNum = holes;

    createSudoku(sudoku1);

    for (int i = 0; i < 3; i++){

        for (int j = 0; j < 3; j++){

            sudoku2[i][j] = sudoku1[i + 6][j + 6];

        }

    }

    createSudoku(sudoku2);

    putHole(sudoku1, sudoku2, holeNum);

    printSudoku(sudoku1, sudoku2);

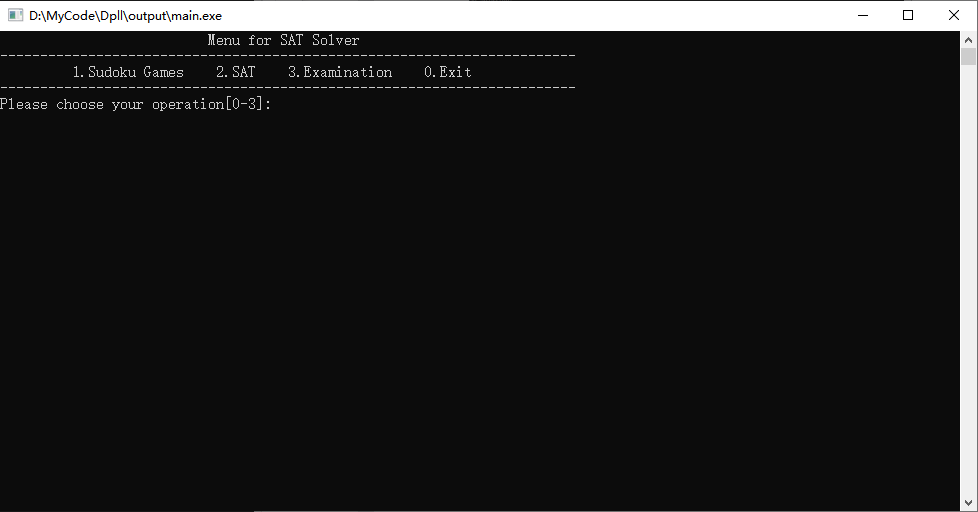
    return getCnf(sudoku1, sudoku2);

}

数独生成的总调用函数，先生成第一个数独，再来生成第二个，返回双数独的cnf文件名

**4.2系统测试**

程序采用简易界面，如图所示：



## 4.2.1 CNF算例求解（SAT模块）

该部分要实现的功能如下：

**（1）输入输出功能：**包括程序执行参数的输入，SAT算例cnf文件的读取，执行结果的输出与文件保存等。

**（2）公式解析：**读取cnf算例文件，解析文件，基于一定的物理结构，建立公式的内部表示。

**（3）DPLL过程：**基于DPLL算法框架，实现SAT算例的求解。

**（4）时间性能的测量：**基于相应的时间处理函数（参考time.h），记录DPLL过程执行时间（以毫秒为单位），并作为输出信息的一部分。

**（5）程序优化：**对基本DPLL的实现进行存储结构、分支变元选取策略[1-3]等某一方面进行优化设计与实现，提供较明确的性能优化率结果。优化率的计算公式为：[(t-to)/t]\*100%,其中t 为未对DPLL优化时求解基准算例的执行时间，to则为优化DPLL实现时求解同一算例的执行时间。

CNF基准算例 表4-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 变元数 | 子句数 | 算例文件夹 | 算例名称 | 优化前 | 优化后 |
| 20 | 91 | 功能测试 | sat-20.cnf | 0ms | 0ms |
| 30 | 420 | 功能测试 | unsat-5cnf-30.cnf | 92ms | 36ms |
| 181 | 3151 | 性能测试 | ais10.cnf | 1940ms | 103ms |
| 303 | 2851 | 性能测试 | sud00009.cnf | 153ms | 187ms |

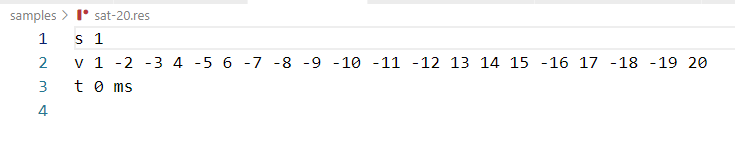


图4-1 算例sat-20.cnf未优化

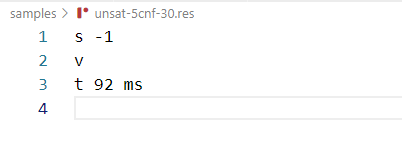


图4-2 unsat-5cnf-30.cnf算例未优化

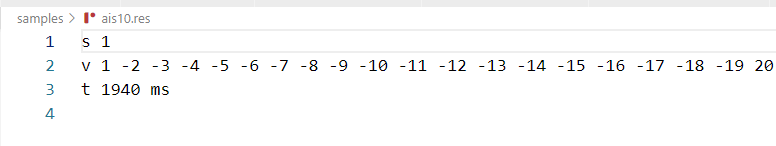


图4-3 ais10.cnf算例未优化

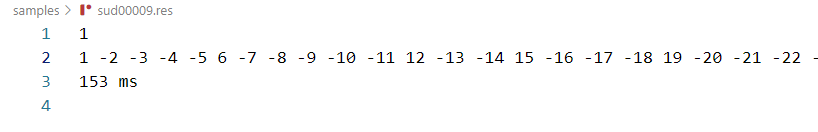


图4-4 sud00009.cnf算例未优化

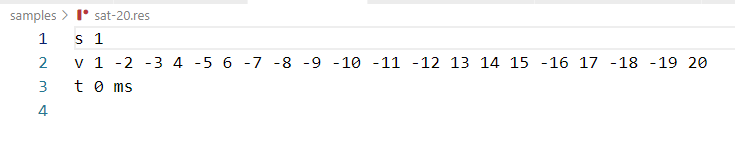


图4-5 算例sat-20.cnf优化后

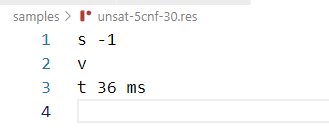


图4-6 unsat-5cnf-30.cnf算例优化后

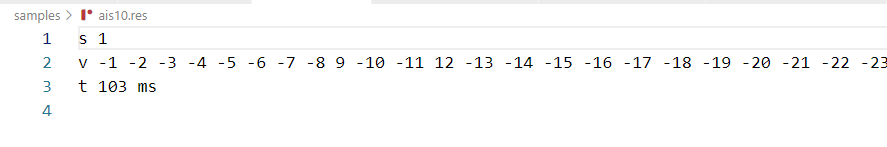


图4-7 ais10.cnf算例优化后

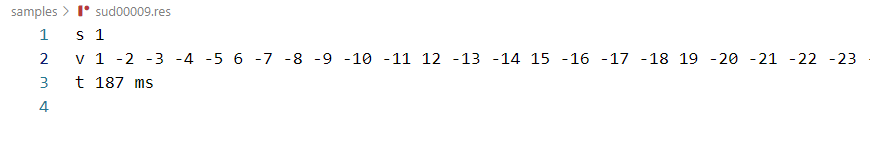


图4-8 sud00009.cnf算例优化后

小型CNF算例 表4-2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 变元数 | 子句数 | 算例属性 | 算例名称 | 优化前（ms） | 优化后（ms） |
| 20 | 91 | S | problem1-20.cnf | 1 | 1 |
| 50 | 80 | S | problem2-50.cnf | 0 | 0 |
| 100 | 340 | S | problem3-100.cnf | 9 | 14 |
| 50 | 100 | S | problem6-50.cnf | 1 | 3 |
| 50 | 300 | S | problem8-50.cnf | 2 | 1 |
| 100 | 200 | S | problem9-100.cnf | 7 | 0 |
| 100 | 600 | S | problem11-100.cnf | 6 | 25 |
| 25 | 100 | S | tst\_v25\_c100.cnf | 0 | 1 |
| 20 | 1532 | S | 7cnf20\_90000\_90000\_7  .shuffled-20.cnf | 48 | 11 |

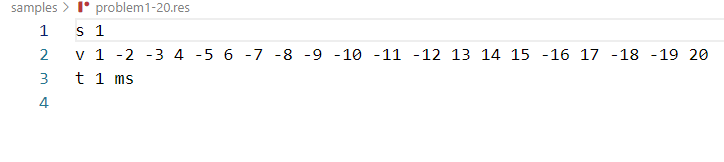


图4-9 problem1-20.cnf算例优化后

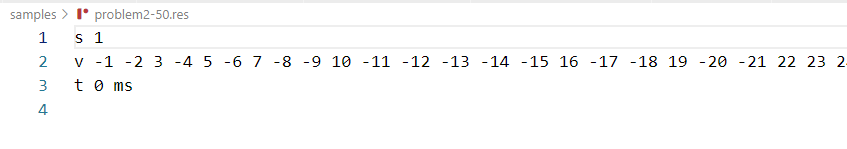


图4-10 problem2-50.cnf算例优化后

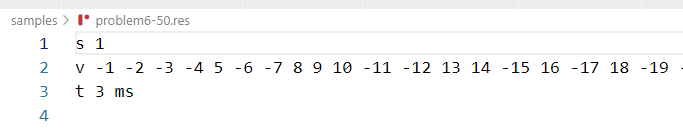


图4-11 problem6-50.cnf算例优化后

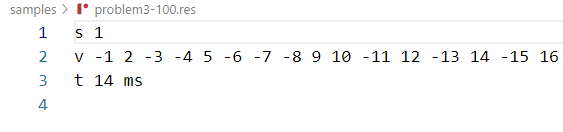


图4-12 problem3-100.cnf算例优化后

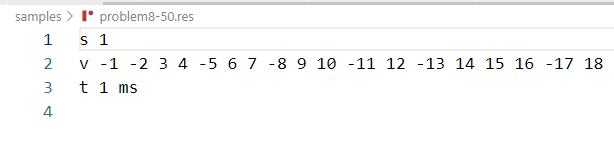


图4-13 problem8-50.cnf算例优化后

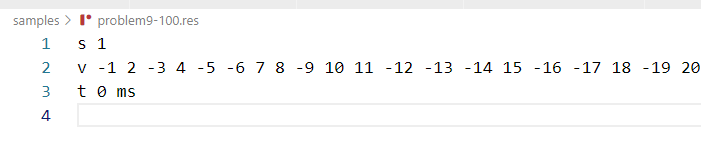


图4-14 problem9-100.cnf算例优化后

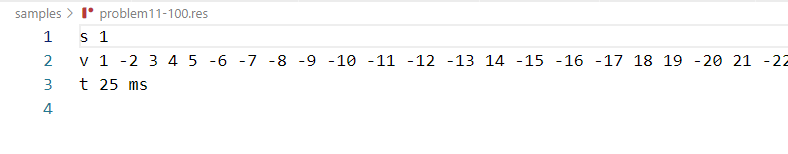


图4-15 problem11-100.cnf算例优化后

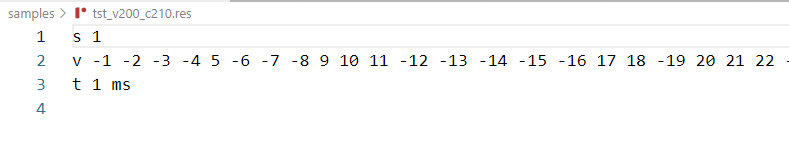


图4-16 tst\_v25\_c100.cnf算例优化后

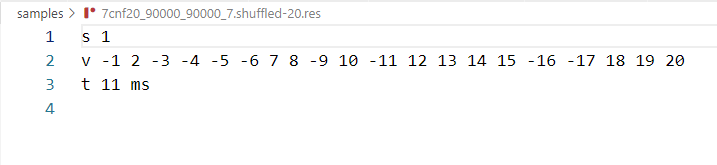


图4-17 7cnf20\_90000\_90000\_7.shuffled-20.cnf算例优化后

中型CNF算例 表4-3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 变元数 | 子句数 | 算例属性 | 算例名称 | 优化前（ms） | 优化后（ms） |
| 200 | 320 | M | problem5-200.cnf | 14 | 10 |
| 200 | 1200 | M | problem12-200.cnf | 26 | 53 |
| 301 | 2780 | M | sud00001.cnf | 70 | 71 |
| 303 | 2851 | M | sud00009.cnf | 160 | 184 |
| 232 | 1901 | M | sud00012.cnf | 31 | 93 |
| 308 | 2911 | M | sud00021.cnf | 911 | 196 |
| 301 | 2810 | M | sud00079.cnf | 110 | 27 |
| 224 | 1762 | M | sud00082.cnf | 10 | 9 |
| 297 | 2721 | M | sud00861.cnf | 51 | 32 |
| 200 | 210 | M | tst\_v200\_c210.cnf | 1 | 1 |
| 231 | 1166 | M | bart17.shuffled-231.cnf | 10 | 17 |

截图略

大型CNF算例 表4-4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 变元数 | 子句数 | 属性 | 算例名称 | 优化前（ms） | 优化后（ms） |
| 1584 | 16587 | L | ec-iso-ukn009.shuffled-as.sat05-3632-1584.cnf | 未解出 | 1996 |
| 1075 | 3152 | L | eh-dp04s04.shuffled-1075.cnf | 未解出 | 992 |

## 

图4-18 ec-iso-ukn009.shuffled-as.sat05-3632-1584.cnf算例



图4-19 eh-dp04s04.shuffled-1075.cnf算例

不满足CNF算例 表4-5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 变元数 | 子句数 | 算例名称 | 优化前（ms） | 优化后（ms） |
| 30 | 420 | unsat-5cnf-30.cnf | 未解出 | 92 |
| 80 | 370 | php-010-008.shuffled-as.sat05-1171.cnf | 未解出 | 3294 |
| 60 | 936 | u-5cnf\_3900\_3900\_060.shuffled-60.cnf | 未解出 | 31178 |

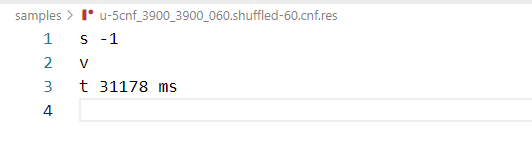


图4-20 u-5cnf\_3900\_3900\_060.shuffled-60.cnf算例

## 4.2.2 数独游戏测试

该部分要实现的功能如下：

将数双独游戏问题转化为SAT问题，并集成到上面的求解器进行数独游戏求解，游戏可玩，具有一定的的交互性。设计目标是实现填空、撤销、帮助填入、检查正误、查看答案等功能。

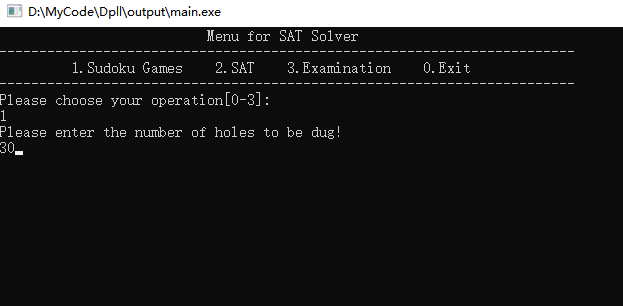




图4-21 数独生成截图

进入数独模块时，会先询问玩家要挖空的数量，再用挖洞法进行初始格局的生成。



图4-22 数独功能1 每填入一个数字都会显示最新填入信息



图4-23 数独功能3 直接填入所指示空位

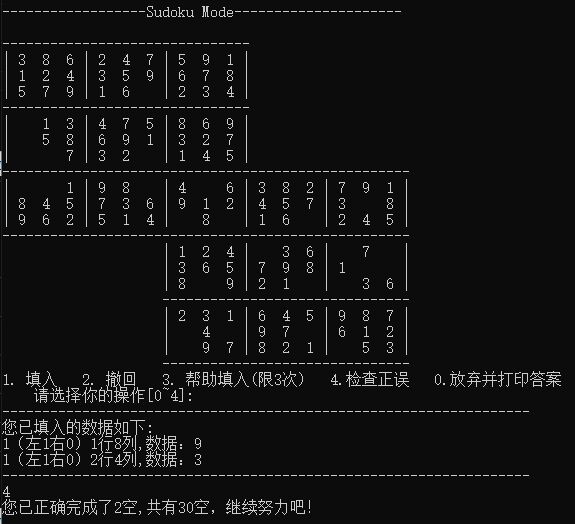


图4-24 数独功能4

可以统计正确数量，显示错误填入的位置

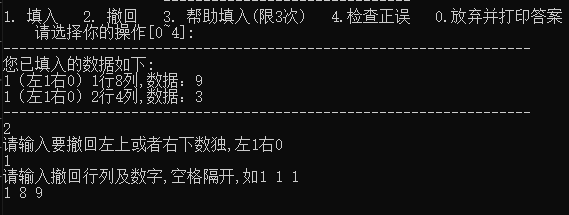


图4-25 数独功能2

可以撤回所填入的空格内数字

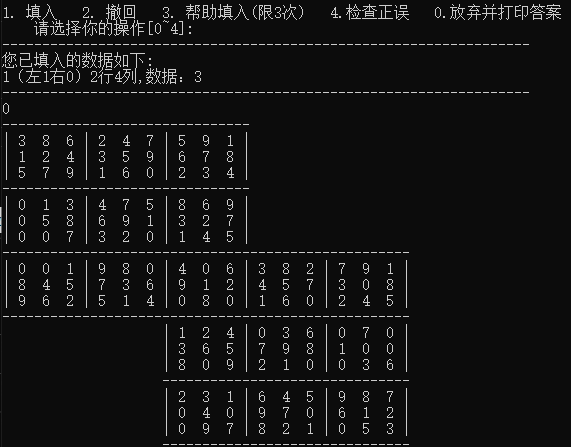


图4-26 数独功能0



图4-27 数独避免错误填入功能

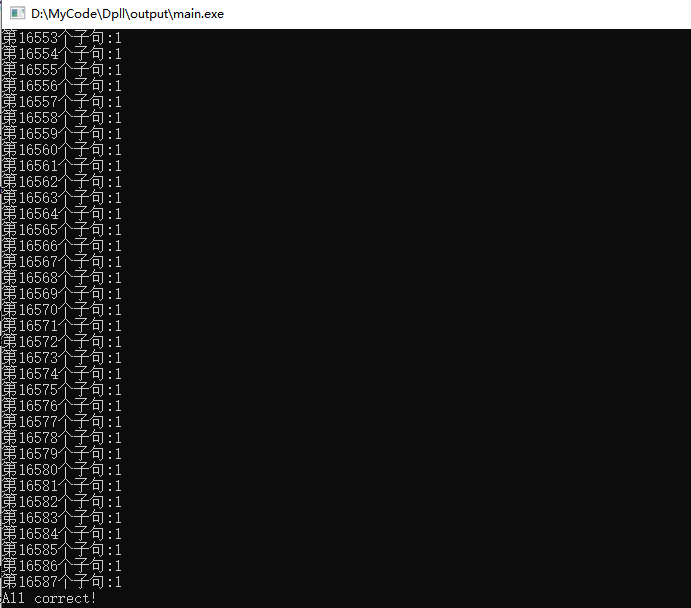
## 4.2.3 答案检验系统测试

该部分要实现的功能如下：

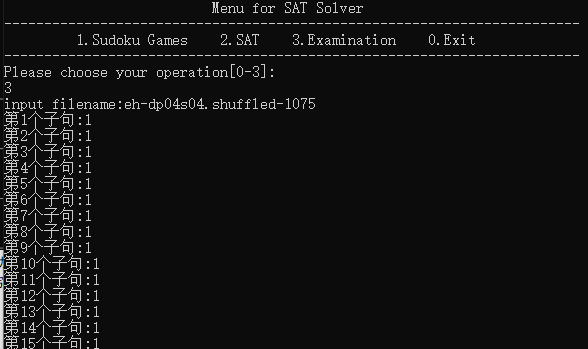
实现对解析正确性的验证功能，即遍历内部结构逐行输出与显示每个子句，与输入算例对比可判断解析功能的正确性，即将所得出的解代回原子句中进行逐句检验。

将答案按顺序保存在数组中，遍历输出输出即可全览每个字句的正误。

检查大型算例ec-iso-ukn009.shuffled-as.sat05-3632-1584.cnf：



检查大型算例eh-dp04s04.shuffled-1075.cnf：



## 

## 4.3 结果分析

（1）对于SAT问题：

通过上面的表格可以大致看出：优化后的算法在小算例中优势并不明显。而随着变元数目与子句数目的增多，优化过后的算法更占优势。当子句的数目不是很多时，优化之后的算法优势不明显，因为优化需要遍历子句，找到最短子句，在这种情况下，子句数目在较小的范围时会出现优化等于没优化，或者优化之后更慢的情况。经检验，该求解器均能在规定时间内求解大、中、小算例，满足该模块的设计目标。不足之处：仍有部分大型算例无法解出。

（2）对于数独游戏：

对于数独游戏的测试，不难发现挖洞数越多，生成初始棋盘的时间较长，那是由于随着挖洞数增加，每挖一个洞生成多解数独的可能性越大，需要进行dpll的次数越多。我尽量对dpll进行了优化，使得解数独的时间尽可能短。经统计，挖30空平均需要10秒，50空平均需要25秒。经检验，数独模块完成了预计目标，并有应对不合法输入的策略，满足该模块的设计目标。

（3）不同的算法适应不同的算例。优化之前的算法更适合小型算例，子句与变元数目均不多的情况。优化之后的算法更适合子句长度比较悬殊且变元数目和子句数目较大的情况。

## 4.4 复杂度分析

DPLL；最坏情况为测试这些变量赋值的每种可能的排列方式（如全部赋为True 、其一为 True ，其他全为False ……），若存在一种赋值排列使得公式的结果为True ，那么就可以说明这条公式是可满足的。但很显然，最坏情况下这种方法需要我们测试 2^n 种（n为变量数）赋值排列，而用于检查每种赋值排列最终的运算结果（Examination）也是不可忽略的。故时间复杂度（最坏）O（2^n）。

Examination：所需时间即为遍历数组时间2\*O（布尔变元数）+O（所有变元数量）

假设公式的变元数为v,子句数为C,公式的总文字数N,子句平均长度L=N/C,短时间内其变化可忽略不计。DPLL 算法可以等价于一颗深度为变元数量V的二叉树上的搜索算法。到达某--结点时，可能会发现当前结点已经不满足目标条件(出现空子句)，说明没有继续向深处搜索的必要，于是折返继续搜索。假定搜索的平均深度为h,那么需要遍历的结点数就为2^ h的量级，时间复杂度为0(2 ^h)。

**5.总结与展望**

**5.1全文总结**

主要工作如下：

（1）完成sat求解器的编写并且不断优化；

（2）完成基于SAT的双数独游戏求解程序；

（3）完成sat求解答案的解析验证程序；

（4）完成实验报告的撰写。

**5.2工作展望**

在今后的研究中，围绕着如下几个方面开展工作：

（1）学而不思则罔，思而不学则殆。过去的学习中，我总是拘泥于书本，并没有过多重视实践，我将多研究算法，多尝试一些实际应用的程序。

（2）这次课设使我意识到，学习的过程中需要同伴，互相监督学习，互相鼓励支持，一个人是很难克制自己的，希望以后寻找一些学习上的同伴，共同进步。

# 6 体会

通过本次程序设计综合课程设计，我进一步正确理解与应用了专业知识，增强和提高了分析问题与解决问题的综合能力，加深了对于求解实际问题的基本科研步骤的体会与理解，增强和提升了信息搜索和分析技能，培养了技术总结的基本技能，锻炼了课程设计报告的撰写的能力。

这门课程只进行了八次，共两个星期，但我觉得这门课的重量不止这些学时，在我看来虽然学时少，但是收获大。这次课设与以往的实验不同，以往都是一个文件就可以搞定，但是这次试验更偏向于实际，注重工程中的模块化编程，这一点是我的收获之一。当然，对dpll算法进行优化也是一个既痛苦又快乐的过程。

这次的课程设计不同与以往的是单数独改为了双数独。因此更加需要我们进行独立思考和算法设计，而没有办法去参照往年的代码。说明老师对我们的要求提高了。很多人都是从暑假看到题目就开始构思撰写的，我刚看到题目的时候完全看不懂要干嘛，所以先问了一些代码基础比较好的同学，学习他们的思路。在学习的过程中，有一点是必须要具有的，那就是勤于请教，不懂就要问，我在这些天里问了很多人关于课设中遇到的难题，他们也都乐于回答，这使我很感激。

在我完成课程设计的期间里，我遇到了很多困难，在我陷入困境无法解决的时候，有很多人给予了我无私的帮助。更重要的是，老师和助教给予了我很多建议，指正了我一些错误。在此，我向在课程设计期间帮助过我的老师、同学表示最真挚的感谢，谢谢你们对我的帮助！

# 参考文献

[1] 张健著. 逻辑公式的可满足性判定—方法、工具及应用. 科学出版社，2000

[2]Tanbir Ahmed. An Implementation of the DPLL Algorithm. Master thesis, Concordia University,Canada,2009

[3] 陈稳. 基于DPLL的SAT算法的研究与应用.硕士学位论文，电子科技大学，2011

[4]Carsten Sinz.Visualizing SAT Instances and Runs of the DPLL Algorithm.J Autom Reasoning (2007) 39:219–243

[5] 360百科：数独游戏<https://baike.so.com/doc/3390505-3569059.html>

[6] Tjark Weber. A sat-based sudoku solver. In 12th International Conference on Logic forProgramming, Artificial Intelligence and Reasoning, LPAR 2005, pages 11–15, 2005.

[7]Ins Lynce and Jol Ouaknine. Sudoku as a sat problem.In Proceedings of the 9th InternationalSymposium on Artificial Intelligence and Mathematics, AIMATH 2006, Fort Lauderdale.Springer,2006.

[8] Uwe Pfeiffer, Tomas Karnagel and Guido Scheffler. A Sudoku-Solver for Large Puzzles using SAT. LPAR-17-short (EPiC Series, vol. 13), 52–57

[9] Sudoku Puzzles Generating: from Easy to Evil.

http://zhangroup.aporc.org/images/files/Paper\_3485.pdf

[10] Robert Ganian and Stefan Szeider. Community Structure Inspired Algorithms for SAT and #SAT. International Conference on Theory and Applications of Satisfiability Testing(SAT 2015),223-237360

[11] 赵伟楠. 对可满足性（SAT）问题求全解的算法研究及实现. 硕士学位论文. 北京交通大学

# 附录A main.cpp

#include "header.h"

void display()

{

    printf("                          Menu for SAT Solver                           \n");

    printf("------------------------------------------------------------------------\n");

    printf("         1.Sudoku Games    2.SAT    3.Examination    0.Exit             \n");

    printf("------------------------------------------------------------------------\n");

    printf("Please choose your operation[0-3]:\n");

}

void printAnswer(clock\_t startTime, clock\_t endTime, bool flag, int \*result, int varNum, string filename)

{

    string outputFile = filePath + filename;

    ofstream fos(outputFile);

    if (!fos.is\_open())

    {

        cout << "error! Can't open a new file\n"

             << endl;

        exit(-1);

    }

    if (flag)

    {

        fos << "s " << 1 << endl;

        fos << "v ";

        for (int i = 0; i < varNum; i++)

        {

            if (result[i] == 1)

            {

                fos << i + 1 << " ";

            }

            else if (result[i] == 0)

            {

                fos << -(i + 1) << " ";

            }

            else

            {

                fos << i + 1 << " ";

            }

        }

        fos << endl;

    }

    else

    {

        fos << "s " << -1 << endl;

        fos << "v " << endl;

    }

    fos << "t " << (double)(endTime - startTime) / CLOCKS\_PER\_SEC \* 1000.0 << " ms" << endl;

    fos.close();

}

void printCompleteSudoku(int \*result, int varNum)

{

    ofstream fos(filePath + "sudokuAnswer.out");

    int a[9][9] = {0},b[9][9] = {0};

    for (int i = 0; i < varNum / 2; i++) {

        if (result[i] > 0) {

            int t = i;

            int x = t / 81;

            int y = (t - x \* 81) / 9;

            a[x][y] = t - x \* 81 - y \* 9 + 1;

        }

    }

    for (int i = varNum / 2; i < varNum; i++) {

        if (result[i] > 0) {

            int t = i - 729;

            int x = t / 81;

            int y = (t - x \* 81) / 9;

            b[x][y] = t - x \* 81 - y \* 9 + 1;

        }

    }

    for (int i = 0; i < 6; i++)

    {

        if (i % 3 == 0)

            cout << "-------------------------------" << endl;

        ;

        for (int j = 0; j < 9; j++)

        {

            if (j % 3 == 0)

                cout << "|";

            cout << " " << a[i][j] << " ";

        }

        cout << "|" << endl;

    }

    for (int i = 6; i < 9; i++)

    {

        if (i % 3 == 0)

            cout << "---------------------------------------------------" << endl;

        for (int j = 0; j < 9; j++)

        {

            if (j % 3 == 0)

                cout << "|";

            cout << " " << a[i][j] << " ";

        }

        cout << "|";

        for (int j = 3; j < 9; j++)

        {

            cout << " " << b[i - 6][j] << " ";

            if ((j - 2) % 3 == 0)

                cout << "|";

        }

        cout << endl;

    }

    for (int i = 3; i < 9; i++)

    {

        if (i == 3)

            cout << "---------------------------------------------------" << endl;

        if (i == 6)

            cout << "                    -------------------------------" << endl;

        for (int j = 1; j <= 20; j++)

        {

            cout << " ";

        }

        for (int j = 0; j < 9; j++)

        {

            if (j % 3 == 0)

                cout << "|";

            cout << " " << b[i][j] << " ";

        }

        cout << "|" << endl;

    }

    cout << "                    -------------------------------" << endl;

    if (!fos.is\_open())

    {

        cout << "Error! Can't open file" << endl;

        exit(-1);

    }

    for (int i = 0; i < 6; i++)

    {

        for (int j = 0; j < 9; j++)

        {

            fos << a[i][j] << " ";

        }

    }

    for (int i = 6; i < 9; i++)

    {

        for (int j = 0; j < 9; j++)

        {

            fos << a[i][j] << " ";

        }

        for (int j = 3; j < 9; j++)

        {

            fos << b[i - 6][j] << " ";

        }

    }

    for (int i = 3; i < 9; i++)

    {

        for (int j = 0; j < 9; j++)

        {

            fos << b[i][j] << " ";

        }

    }

}

int main()

{

    display();

    int choice = 0, holes = 0;

    cin >> choice;

    while (choice)

    {

        if (choice == 1)

        {

            cout << "Please enter the number of holes to be dug!" << endl;

            cin >> holes;

            string filename = initSudoku(holes);

            int varnum = 0;

            Head \*LinkedList = initCnf(varnum, filename);

            int \*result = new int[varnum];

            memset(result, 0, sizeof(int) \* varnum);

            clock\_t startTime, endTime;

            startTime = clock();

            bool flag = DPLL(LinkedList, result, varnum);

            endTime = clock();

            printAnswer(startTime, endTime, flag, result, varnum, "sudoku.res");

            games(holes, result);

            printCompleteSudoku(result, varnum);

        }

        else if (choice == 2)

        {

            int varnum;

            string file = " ";

            cin >> file;

            string suffix = ".cnf";

            string filename = file + suffix;

            Head \*LinkedList = initCnf(varnum, filename);

            int \*result = new int[varnum];

            memset(result, 0, sizeof(int) \* varnum);

            clock\_t startTime, endTime;

            string outputFile = file + ".res";

            startTime = clock();

            bool flag = DPLL(LinkedList, result, varnum);

            endTime = clock();

            printAnswer(startTime, endTime, flag, result, varnum, outputFile);

        }

        else if (choice == 3)

        {

            Examination();

        }

        else if (choice == 0)

        {

            break;

        }

        else

        {

            cout << "Please input the right num! " << endl;

            exit(-1);

        }

        display();

        cin >> choice;

    }

    return 0;

}

# 附录B CnfParser.cpp

#include"header.h"

ifstream openFile(string file) {  //利用ifstream类的构造函数创建一个文件输入流对象

    ifstream fis(file.c\_str());

    return fis;

}

Head\* buildLinkedList(ifstream& fis,int clauseNum) {   //将cnf文件输入十字链表

    Head\* HEAD = new Head;

    Head\* headRear = HEAD;

    Head\* END = new Head;

    for (int i = 0; i < clauseNum; i++) {

        int temp;

        fis >> temp;

        Data\* front = new Data;

        front -> data = temp;

        headRear -> nextData = front;

        headRear -> num++;

        fis >> temp;

        while (temp != 0) {

            Data\* rear = new Data;

            front -> nextData = rear;

            rear -> data = temp;

            front = front -> nextData;

            headRear -> num++;

            fis >> temp;

        }

        front -> nextData = NULL;

        fis.get();  //换行

        Head\* tp = new Head;

        headRear -> nextHead = tp;

        END = headRear;

        headRear = headRear -> nextHead;

    }

    END -> nextHead = NULL;

    return HEAD;

}

void printLinkedList(Head\* HEAD) {    //输出读入的cnf字句

    Head\* Phead = HEAD;

    Data\* front = NULL;

    cout << "CnfParser" << endl;

    while (Phead != NULL) {

        front = Phead -> nextData;

        while (front != NULL) {

            cout << front -> data << " ";

            front = front -> nextData;

        }

        cout << endl;

        Phead = Phead -> nextHead;

    }

}

Head\* initCnf(int &varNum, string &filename) {

    //打开cnf文件

    string file = filePath + filename;

    ifstream fis = openFile(file);

    char ch,readbuf[1145];

    fis >> ch;

    while (ch != 'p') {

        fis.getline(readbuf,1145);

        fis >> ch;

    }//此时已读完字符p

    string cnf;

    int clauseNum;

    fis >> cnf >> varNum >> clauseNum;

    fis.get();

    //构建链表

    Head\* HEAD = buildLinkedList(fis,clauseNum);

    //输出整个链表

    //printLinkedList(HEAD);

    return HEAD;

}

# 附录C DPLL.cpp

#include "header.h"

void DestroyLinkedList(Head\* LinkedList)

{

    Head\* p = LinkedList ;

    Head\* p1=nullptr;

        while(p){

            Data\* e=p->nextData;

            Data\* e1=nullptr;

            while(e){

                e1=e;

                e=e1->nextData;

                free(e1);

            }

            p1=p;

            p=p1->nextHead;

            free(p1);

        }

}

bool isEmptyClause(Head\* LinkedList) {

    Head\* PHead = LinkedList;

    while (PHead != nullptr) {

        if (PHead -> num == 0) {

            return true;

        }

        PHead = PHead -> nextHead;

    }

    return false;

}

Head\* findSingleClause(Head\* LinkedList) {

    while (LinkedList != nullptr) {

        if (LinkedList -> num == 1) {

            return LinkedList;

        }

        LinkedList = LinkedList -> nextHead;

    }

    return nullptr;

}

Head\* addSingleClause(Head\* LinkedList,int var) {

    Head\* addHead = new Head;

    Data\* addData = new Data;

    addData -> data = var;

    addData -> nextData = nullptr;

    addHead -> nextData = addData;

    addHead -> num = 1;

    addHead -> nextHead = LinkedList;

    LinkedList = addHead;

    return LinkedList;

}

void deleteHead(Head\* Clause,Head \*&LinkedList) {

    if (!Clause)

        return;

    if (Clause == LinkedList) {

        LinkedList = Clause -> nextHead;

    } else {

        for (Head\* front = LinkedList; front != nullptr; front = front -> nextHead) {

            if (front -> nextHead == Clause) {

                front -> nextHead = Clause -> nextHead;

            }

        }

    }

}

void deleteData(int object,Head \*&LinkedList) {

    for (Head\* pHead = LinkedList; pHead != nullptr; pHead = pHead -> nextHead) {

        for (Data\* rear = pHead -> nextData; rear != nullptr; rear = rear -> nextData) {

            if (rear -> data == object) {  //删除整行

                deleteHead(pHead,LinkedList);

            } else if (abs(rear -> data) == abs(object)) {

                //绝对值相同，只删除这个节点

                if (rear == pHead -> nextData) {

                    //如果是头部节点

                    pHead -> nextData = rear -> nextData;

                    pHead -> num--;

                    free(rear);

                } else {

                    for (Data\* front = pHead -> nextData; front != nullptr; front = front -> nextData) {

                        if (front -> nextData == rear) {

                            front -> nextData = rear -> nextData;

                            pHead -> num--;

                        }

                    }

                }

            }

        }

    }

}

void printResult(int \*result,int varNum) {

    cout << "V ";

    for (int i = 0; i < varNum; i++) {

        if (result[i] == 1) {

            cout << i + 1 << " ";

        } else if (result[i] == 0) {

            cout << -(i + 1) << " ";

        } else {  //第三种情况是变元，可true可false

            cout << i + 1 << " ";

        }

    }

    cout << endl;

}

Head\* duplication(Head\* LinkedList) {

    //复制，先创建第一行

    Head\* lastHead = LinkedList;

    Head\* newHead = new Head;

    newHead -> num = lastHead -> num;

    Head\* pHead = newHead;

    //第一个Data

    Data\* newData = new Data;

    Data\* firstLastData = lastHead -> nextData;

    newData -> data = firstLastData -> data;

    pHead -> nextData = newData;

    //第一行

    for (firstLastData = firstLastData -> nextData; firstLastData != nullptr; firstLastData = firstLastData -> nextData) {

        Data\* temp = new Data;

        temp -> data = firstLastData -> data;

        newData -> nextData = temp;

        newData = newData -> nextData;

    }

    //后面若干行

    for (lastHead = lastHead ->nextHead; lastHead != nullptr; lastHead = lastHead -> nextHead) {

        Head\* tempHead = new Head;

        Data\* tempData = new Data;

        tempHead -> num = lastHead -> num;

        pHead -> nextHead = tempHead;

        pHead = pHead -> nextHead;

        Data\* lastData = lastHead -> nextData;

        tempData -> data = lastData -> data;

        pHead -> nextData = tempData;

        for (lastData = lastData -> nextData; lastData != nullptr; lastData = lastData -> nextData) {

            Data\* temp = new Data;

            temp -> data = lastData -> data;

            tempData -> nextData = temp;

            tempData = tempData -> nextData;

        }

        tempData -> nextData = nullptr;

    }

    pHead -> nextHead = nullptr;

    return newHead;

}

Data\* findMinimal(Head\* LinkedList) {

    Data\* result = nullptr;

    int miniNum = 0xffff;

    for (Head\* head = LinkedList; head != nullptr; head = head -> nextHead) {

        if (head -> num <= miniNum) {

            miniNum = head -> num;

            result = head -> nextData;

        }

    }

    return result;

}

bool DPLL(Head\* LinkedList,int\* result,int varNum) {

    Head\* pFind = LinkedList;

    Head\* singleClause = findSingleClause(pFind);

    while (singleClause != nullptr) {

        int data = singleClause -> nextData -> data;

        if (data > 0) {

            result[abs(data) - 1] = 1;

        } else {

            result[abs(data) - 1] = 0;

        }

        deleteHead(singleClause,LinkedList);  //删除单子句

        deleteData(data,LinkedList);   //化简包含相反值的子句

        if (!LinkedList) {

            return true;

        } else if (isEmptyClause(LinkedList)) {

            DestroyLinkedList(LinkedList);

            return false;

        }

        pFind = LinkedList;

        singleClause = findSingleClause(pFind);

    }

    Data\* minimalList = findMinimal(LinkedList);

    int data = minimalList -> data;

    //int data = LinkedList -> nextData -> data;

    Head\* last = duplication(LinkedList);

    Head\* newLinkedList = addSingleClause(LinkedList,data);

    //分裂策略

    if (DPLL(newLinkedList,result,varNum)) {

        return true;

    } else {

        Head\* newLinkedList2 = addSingleClause(last,-data);

        return DPLL(newLinkedList2,result,varNum);

    }

}

# 附录D header.h

#include <cstdio>

#include <cstdio>

#include <cstring>

#include <iostream>

#include <fstream>

#include <ctime>

#include <algorithm>

#define ROW 9

#define COL 9

using namespace std;

const string filePath = R"(C:\Users\cy\Desktop\文件\数据结构课设\Dpll\samples\)";

typedef struct Data {

    int data = 0;

    Data \*nextData{};

}Data;

typedef struct Head{

    int num = 0;

    Data \*nextData{};

    Head \*nextHead{};

}Head;

//十字链表定义

void games(int holes ,int \*result);

void Examination(void);

Head\* initCnf(int &, string &);

bool DPLL(Head\*,int\*,int);

string initSudoku(int holes);

# 附录E Sudoku.cpp

#include "header.h"

string getCnf(int a[][COL], int b[][COL]);

void randomFirstRow(int a[], int n)

{

    srand((unsigned)time(NULL));

    int vis[10];

    memset(vis, 0, sizeof(vis));

    for (int i = 0; i < n; i++)

    {

        if (a[i] != 0)

        { //如果已经填过了

            vis[a[i]] = 1;

        }

        else

        {

            a[i] = rand() % 9 + 1;

            while (vis[a[i]])

            {

                a[i] = rand() % 9 + 1;

            }

            vis[a[i]] = 1;

        }

    }

}

bool fillLast(int a[][COL], int row, int col)

{

    //当前要填第row行col列的格子

    if (row == ROW)

    {

        return true;

    }

    if (col == COL)

    {

        return fillLast(a, row + 1, 0);

    }

    if (a[row][col] != 0)

    {

        //如果已经填过了

        return fillLast(a, row, col + 1);

    }

    int vis[10];

    memset(vis, 0, sizeof(vis));

    for (int i = 0; i < 9; i++)

    {

        vis[a[row][i]] = 1;

        vis[a[i][col]] = 1;

    }

    for (int i = row / 3 \* 3; i < row / 3 \* 3 + 3; i++)

    {

        for (int j = col / 3 \* 3; j < col / 3 \* 3 + 3; j++)

        {

            vis[a[i][j]] = 1;

        }

    }

    for (int i = 1; i <= 9; i++)

    {

        if (vis[i])

        {

            continue;

        }

        a[row][col] = i;

        if (fillLast(a, row, col + 1))

        {

            return true;

        }

    }

    a[row][col] = 0;

    return false;

}

void createSudoku(int a[][COL])

{

    randomFirstRow(a[0], COL);

    fillLast(a, 1, 0);

}

void print(const int a[][COL])

{

    for (int i = 0; i < ROW; i++)

    {

        for (int j = 0; j < COL; j++)

        {

            cout << a[i][j] << " ";

        }

        cout << endl;

    }

    cout << endl;

}

void putHole(int a[][COL], int b[][COL], int holeNum)

{

    srand((unsigned)time(NULL));

    int ii = 0, a1 = 0, b1 = 0;

    while (ii < holeNum)

    {

        int whichSudoku = rand() % 2, x = rand() % 9, y = rand() % 9;

        if (whichSudoku == 0)

        {

            //挖第一个数独上的洞

            while (a[x][y] == 0)

            { //已经被挖过了

                x = rand() % 9;

                y = rand() % 9;

            }

            a1 = a[x][y];

            a[x][y] = 0;

            if (x >= 6 && y >= 6)

            {

                b[x - 6][y - 6] = 0;

            }

        }

        else

        {

            while (b[x][y] == 0)

            {

                x = rand() % 9;

                y = rand() % 9;

            }

            b1 = b[x][y];

            b[x][y] = 0;

            if (x <= 2 && y <= 2)

            {

                a[x + 6][y + 6] = 0;

            }

        }

        if (whichSudoku == 0){

            for (int p = 1; p <= 9; p++){

                int k1=0;

                if (p == a1) continue;

                int vis[10];

                memset(vis, 0, sizeof(vis));

                for (int i1 = 0; i1 < 9; i1++)

                {

                   vis[a[x][i1]] = 1;

                   vis[a[i1][y]] = 1;

                }

                for (int i = x / 3 \* 3; i < x / 3 \* 3 + 3; i++)

                {

                    for (int j = y / 3 \* 3; j < y / 3 \* 3 + 3; j++)

                    {

                        vis[a[i][j]] = 1;

                    }

                }

                for (int i = 1; i <= 9; i++)

                {

                    if (vis[p])

                    {

                        k1=1;break;

                    }

                }

                if (k1 == 1) continue;

                a[x][y] = p;

                string q = getCnf(a, b);

                int varnum = 0;

                Head \*LinkedList = initCnf(varnum, q);

                int \*result = new int[1458];

                memset(result, 0, sizeof(int) \* varnum);

                bool flag = DPLL(LinkedList, result, 1458);

                if (flag == true){

                    if (whichSudoku == 0){

                        a[x][y] = a1;

                        if (x >= 6 && y >= 6){

                            b[x - 6][y - 6] = a1;

                        }

                    }

                    if (whichSudoku != 0){

                        b[x][y] = b1;

                        if (x <= 2 && y <= 2){

                            a[x + 6][y + 6] = b1;

                        }

                    }

                    ii--;

                    break;

                }

            }

        }

        else{

            for (int p = 1; p <= 9; p++){

                int k1=0;

                if (p == b1) continue;

                int vis[10];

                memset(vis, 0, sizeof(vis));

                for (int i1 = 0; i1 < 9; i1++)

                {

                   vis[b[x][i1]] = 1;

                   vis[b[i1][y]] = 1;

                }

                for (int i = x / 3 \* 3; i < x / 3 \* 3 + 3; i++)

                {

                    for (int j = y / 3 \* 3; j < y / 3 \* 3 + 3; j++)

                    {

                        vis[b[i][j]] = 1;

                    }

                }

                for (int i = 1; i <= 9; i++)

                {

                    if (vis[p])

                    {

                        k1=1;break;

                    }

                }

                if (k1 == 1) continue;

                b[x][y] = p;

                string q = getCnf(a, b);

                int varnum = 0;

                Head \*LinkedList = initCnf(varnum, q);

                int \*result = new int[1458];

                memset(result, 0, sizeof(int) \* varnum);

                bool flag = DPLL(LinkedList, result, 1458);

                if (flag == true){

                    if (whichSudoku == 0){

                        a[x][y] = a1;

                        if (x >= 6 && y >= 6){

                            b[x - 6][y - 6] = a1;

                        }

                    }

                    if (whichSudoku != 0){

                        b[x][y] = b1;

                        if (x <= 2 && y <= 2){

                            a[x + 6][y + 6] = b1;

                        }

                    }

                    ii--;

                    break;

                }

            }

        }

        ii++;

    }

}

string getCnf(int a[][COL], int b[][COL])

{

    ofstream fos((filePath + "sudoku.cnf").c\_str());

    if (!fos.is\_open())

    {

        cout << "Error! Can't open file1" << endl;

        exit(-1);

    }

    int tot = 0, cnt = 0;

    for (int i = 0; i < ROW; i++)

    {

        for (int j = 0; j < COL; j++)

        {

            if (a[i][j] != 0)

            {

                tot++;

            }

            if (b[i][j] != 0)

            {

                tot++;

            }

        }

    }

    fos << "p cnf 1458 " << 29970 + tot << " " << endl;

    //对于已经填了的，直接生成单语句

    for (int i = 0; i < ROW; i++)

    {

        for (int j = 0; j < COL; j++)

        {

            if (a[i][j] != 0)

            {

                fos << i \* 81 + j \* 9 + a[i][j] << " " << 0 << endl;

            }

            if (b[i][j] != 0)

            {

                fos << 729 + i \* 81 + j \* 9 + b[i][j] << " " << 0 << endl;

            }

        }

    }

    //单个格子

    for (int i = 1; i <= 9; i++)

    {

        for (int j = 1; j <= 9; j++)

        {

            for (int k = 1; k <= 9; k++)

            {

                fos << (i - 1) \* 81 + (j - 1) \* 9 + k << " ";

            }

            fos << 0 << endl;

            cnt++;

            for (int k = 1; k <= 9; k++)

            {

                fos << 729 + (i - 1) \* 81 + (j - 1) \* 9 + k << " ";

            }

            fos << 0 << endl;

            cnt++;

            for (int k = 1; k <= 9; k++)

            {

                for (int p = k + 1; p <= 9; p++)

                {

                    fos << -((i - 1) \* 81 + (j - 1) \* 9 + k) << " " << -((i - 1) \* 81 + (j - 1) \* 9 + p) << " " << 0 << endl;

                    fos << -(729 + (i - 1) \* 81 + (j - 1) \* 9 + k) << " " << -(729 + (i - 1) \* 81 + (j - 1) \* 9 + p) << " " << 0 << endl;

                    cnt += 2;

                }

            }

            // cnt += 2;

        }

    }

    //行

    for (int i = 1; i <= 9; i++)

    {

        for (int k = 1; k <= 9; k++)

        {

            for (int j = 1; j <= 9; j++)

            {

                fos << ((i - 1) \* 81 + (j - 1) \* 9 + k) << " ";

            }

            fos << 0 << endl;

            for (int j = 1; j <= 9; j++)

            {

                fos << (729 + (i - 1) \* 81 + (j - 1) \* 9 + k) << " ";

            }

            fos << 0 << endl;

            cnt += 2;

            for (int p = 1; p <= 9; p++)

            {

                for (int q = p + 1; q <= 9; q++)

                {

                    fos << -((i - 1) \* 81 + (p - 1) \* 9 + k) << " " << -((i - 1) \* 81 + (q - 1) \* 9 + k) << " " << 0 << endl;

                    fos << -(729 + (i - 1) \* 81 + (p - 1) \* 9 + k) << " " << -(729 + (i - 1) \* 81 + (q - 1) \* 9 + k) << " " << 0 << endl;

                    cnt += 2;

                }

            }

        }

    }

    //列

    for (int j = 1; j <= 9; j++)

    {

        for (int k = 1; k <= 9; k++)

        {

            for (int i = 1; i <= 9; i++)

            {

                fos << ((i - 1) \* 81 + (j - 1) \* 9 + k) << " ";

            }

            fos << 0 << endl;

            for (int i = 1; i <= 9; i++)

            {

                fos << (729 + (i - 1) \* 81 + (j - 1) \* 9 + k) << " ";

            }

            fos << 0 << endl;

            cnt += 2;

            for (int p = 1; p <= 9; p++)

            {

                for (int q = p + 1; q <= 9; q++)

                {

                    fos << -((p - 1) \* 81 + (j - 1) \* 9 + k) << " " << -((q - 1) \* 81 + (j - 1) \* 9 + k) << " " << 0 << endl;

                    fos << -(729 + (p - 1) \* 81 + (j - 1) \* 9 + k) << " " << -(729 + (q - 1) \* 81 + (j - 1) \* 9 + k) << " " << 0 << endl;

                    cnt += 2;

                }

            }

        }

    }

    // 3\*3

    for (int x = 0; x < 3; x++)

    {

        for (int y = 0; y < 3; y++)

        {

            int x1 = x \* 3 + 1, x2 = x \* 3 + 3;

            int y1 = y \* 3 + 1, y2 = y \* 3 + 3;

            for (int k = 1; k <= 9; k++)

            {

                for (int i = x1; i <= x2; i++)

                {

                    for (int j = y1; j <= y2; j++)

                    {

                        fos << (i - 1) \* 81 + (j - 1) \* 9 + k << " ";

                    }

                }

                fos << 0 << endl;

                for (int i = x1; i <= x2; i++)

                {

                    for (int j = y1; j <= y2; j++)

                    {

                        fos << 729 + (i - 1) \* 81 + (j - 1) \* 9 + k << " ";

                    }

                }

                fos << 0 << endl;

                cnt += 2;

                for (int i = x1; i <= x2; i++)

                {

                    for (int j = y1; j <= y2; j++)

                    {

                        for (int p = x1; p <= x2; p++)

                        {

                            for (int q = y1; q <= y2; q++)

                            {

                                if (i == p && j == q)

                                {

                                    continue;

                                }

                                fos << -((i - 1) \* 81 + (j - 1) \* 9 + k) << " " << -((p - 1) \* 81 + (q - 1) \* 9 + k) << " " << 0 << endl;

                                fos << -(729 + (i - 1) \* 81 + (j - 1) \* 9 + k) << " " << -(729 + (p - 1) \* 81 + (q - 1) \* 9 + k) << " " << 0 << endl;

                                cnt += 2;

                            }

                        }

                    }

                }

            }

        }

    }

    //公共部分

    for (int i = 0; i < 3; i++)

    {

        for (int j = 0; j < 3; j++)

        {

            for (int k = 1; k <= 9; k++)

            {

                fos << 0 - ((i + 6) \* 81 + (j + 6) \* 9 + k) << " " << (729 + i \* 81 + j \* 9 + k) << " " << 0 << endl;

                fos << ((i + 6) \* 81 + (j + 6) \* 9 + k) << " " << 0 - (729 + i \* 81 + j \* 9 + k) << " " << 0 << endl;

                cnt += 2;

            }

        }

    }

    fos.close();

    return "sudoku.cnf";

}

void printSudoku(int a[][COL], int b[][COL])

{

    ofstream fos((filePath + "sudoku.txt").c\_str());

    if (!fos.is\_open())

    {

        cout << "Error! Can't open file" << endl;

        exit(-1);

    }

    for (int i = 0; i < 9; i++)

    {

        for (int j = 0; j < 9; j++)

        {

            fos << (a[i][j]) << " ";

        }

    }

    for (int i = 0; i < 9; i++)

    {

        for (int j = 0; j < 9; j++)

        {

            fos << (b[i][j]) << " ";

        }

    }

}

string initSudoku(int holes)

{

    int sudoku1[ROW][COL] = {0};

    int sudoku2[ROW][COL] = {0};

    int holeNum = holes;

    //先生成第一个数独，再来生成第二个

    createSudoku(sudoku1);

    for (int i = 0; i < 3; i++)

    {

        for (int j = 0; j < 3; j++)

        {

            sudoku2[i][j] = sudoku1[i + 6][j + 6];

        }

    }

    createSudoku(sudoku2);

    putHole(sudoku1, sudoku2, holeNum);

    printSudoku(sudoku1, sudoku2);

    return getCnf(sudoku1, sudoku2);

}

# 附录F game.cpp

#include "header.h"

void games(int holes ,int \*result)

{

    int varNum=1458;

    int a1[9][9] = {0},b1[9][9] = {0};

    for (int i = 0; i < varNum / 2; i++) {

        if (result[i] > 0) {

            int t = i;

            int x = t / 81;

            int y = (t - x \* 81) / 9;

            a1[x][y] = t - x \* 81 - y \* 9 + 1;

        }

    }

    for (int i = varNum / 2; i < varNum; i++) {

        if (result[i] > 0) {

            int t = i - 729;

            int x = t / 81;

            int y = (t - x \* 81) / 9;

            b1[x][y] = t - x \* 81 - y \* 9 + 1;

        }

    }

    int a[ROW][COL] = {0};

    int b[ROW][COL] = {0};

    int done[2][10][10] = {0};

    string file=filePath + "sudoku.txt" ;

    ifstream f(file);

    if (!f) {

        cout << "Error! File1 can't open!" << endl;

        exit(-1);

    }

    for(int i=0;i<9;i++){

        for(int j=0;j<9;j++){

            f >> a[i][j];

        }

    }

    for(int i=0;i<9;i++) {

        for(int j=0;j<9;j++){

            f >> b[i][j];

        }

    }

    int op=100,i,j,count=0,x,y,d=100,counts=0;

    while(op){

        system("cls"); printf("\n\n");

        printf("------------------Sudoku Mode--------------------- \n");

        cout << endl;

        for (int i = 0; i < 6; i++) {

            if(i % 3 == 0) cout << "-------------------------------" << endl; ;

            for (int j = 0; j < 9; j++) {

            if(j % 3 == 0) cout << "|" ;

            if(a[i][j] != 0) cout << " " << a[i][j] << " " ;

            else cout << "   ";

            }

            cout << "|" << endl;

        }

        for (int i = 6; i < 9; i++) {

            if(i % 3 == 0) cout << "---------------------------------------------------" << endl;

            for (int j = 0; j < 9; j++) {

                if(j % 3 == 0) cout << "|" ;

                if(a[i][j] != 0) cout << " " << a[i][j] << " " ;

                else cout << "   ";

            }

            cout << "|";

            for (int j = 3; j < 9; j++) {

                if(b[i - 6][j] != 0) cout << " " << b[i - 6][j] << " ";

                else cout << "   ";

                if((j-2) % 3 == 0) cout << "|" ;

            }

            cout  << endl;

        }

        for (int i = 3; i < 9; i++) {

            if (i == 3) cout << "---------------------------------------------------" << endl;

            if (i == 6) cout << "                    -------------------------------" << endl;

            for (int j = 1; j <= 20; j++) {

                cout << " ";

            }

            for (int j = 0; j < 9; j++) {

                if(j % 3 == 0) cout << "|" ;

                if(b[i][j] != 0) cout << " " << b[i][j] << " ";

                else cout << "   ";

            }

            cout << "|" << endl;

        }

        cout << "                    -------------------------------" << endl;

        printf("1. 填入   2. 撤回   3. 帮助填入(限3次)   4.检查正误   0.放弃并打印答案\n");

        printf("    请选择你的操作[0~4]:\n");

        printf("------------------------------------------------------------------\n");

        printf("您已填入的数据如下:\n");

        for(int i=0;i<=1;i++){

            for(int j=1;j<=9;j++){

                for(int k=1;k<=9;k++){

                    if(done[i][j][k]==1){

                        cout << i << "（左1右0）" << j << "行" << k << "列,数据：" ;

                        if(i==1) cout << a[j-1][k-1] ;

                        else cout << a[j-1][k-1];

                        cout << endl ;

                    }

                }

            }

        }

        printf("------------------------------------------------------------------\n");

        cin >> op;

        switch(op){

            case 0:break;

            case 1:

                cout << "请输入填入左上或者右下数独,左1右0" << endl ;

                cin >> d;

                cout << "请输入填入行列及数字,空格隔开,如1 1 1" <<endl ;

                cin >> x >> y >> i;

                while((d==1&&a[x-1][y-1]!=0) || (d==0&&b[x-1][y-1]!=0)){

                    cout << "此空已填入固定数字，请重新输入！" << endl;

                    cin >> x >> y >> i;

                }

                if(d==1){

                    a[x-1][y-1]=i;

                    if (x >= 6 && y >= 6) {

                        b[x - 6][y - 6] = i;

                    }

                }

                else if(d==0){

                    b[x-1][y-1]=i;

                    if (x <= 2 && y <= 2) {

                        a[x + 6][y + 6] = i;

                    }

                }

                done[d][x][y]=1;

                getchar();getchar();

                break;

            case 2:

                cout << "请输入要撤回左上或者右下数独,左1右0" << endl ;

                cin >> d;

                cout << "请输入撤回行列及数字,空格隔开,如1 1 1" << endl ;

                cin >> x >> y >> i;

                if(d==1){

                    a[x-1][y-1]=0;

                    if (x >= 6 && y >= 6) {

                        b[x - 6][y - 6] = 0;

                    }

                }

                else if(d==0){

                    b[x-1][y-1]=0;

                    if (x <= 2 && y <= 2) {

                        a[x + 6][y + 6] = 0;

                    }

                }

                done[d][x][y]=0;

                getchar();getchar();

                break;

            case 3:

                if(counts>=3){

                    cout << "帮助次数已达上限" <<endl ;

                    break;

                }

                cout << "请输入要帮助左上或者右下数独,左1右0" << endl ;

                cin >> d;

                cout << "请输入行列,空格隔开,如1 1" <<endl ;

                cin >> x >> y ;

                if(d==1){

                    a[x-1][y-1]=a1[x-1][y-1];

                    if (x >= 6 && y >= 6) {

                        b[x - 6][y - 6] = b1[x - 6][y - 6];

                    }

                }

                else if(d==0){

                    b[x-1][y-1]=b1[x-1][y-1];

                    if (x <= 2 && y <= 2) {

                        a[x + 6][y + 6] = a1[x + 6][y + 6];

                    }

                }

                done[d][x][y]=1;counts++;

                getchar();getchar();

                break;

            case 4:

                count=0;

                for(int i=0;i<=1;i++){

                    for(int j=1;j<=9;j++){

                        for(int k=1;k<=9;k++){

                            if(done[i][j][k]==1){

                                if((i==1&&a[j-1][k-1]==a1[j-1][k-1])||(i==0&&b[j-1][k-1]==b1[j-1][k-1])) count++;

                                else{

                                    cout << "错误填入：" << i << "（左1右0）" << j << "行" << k << "列" << endl;

                                }

                            }

                        }

                    }

                }

                if(count<holes) cout << "您已正确完成了" << count << "空,共有" << holes << "空，继续努力吧！" << endl;

                else cout << "恭喜你完成了数独！压力马斯内！" << endl;

                getchar();getchar();

                break;

        }

    }

}

# 附录G Examination.cpp

#include "header.h"

#include "header.h"

void Examination(void)

{

    string file = " ";

    int varNum, clauseNum, i, j, solution, ans = 1;

    static int bv[35000][40];

    bool clause\_value[35000] = {0};

    printf("input filename:");

    cin >> file;

    string outputFile = file + ".res";

    string filename = file + ".cnf";

    string file1 = filePath + filename;

    string file2 = filePath + outputFile;

    ifstream fi(file1.c\_str());

    if (!fi)

    {

        cout << "Error! File can't open!" << endl;

        exit(-1);

    }

    ifstream fii(file2.c\_str());

    if (!fii)

    {

        cout << "Error! File can't open!" << endl;

        exit(-1);

    }

    char ch, readbuf[1145];

    fi >> ch;

    while (ch != 'p')

    {

        fi.getline(readbuf, 1145);

        fi >> ch;

    } //此时已读完字符p

    string cnf;

    fi >> cnf >> varNum >> clauseNum;

    fi.get();

    fii >> ch;

    while (ch != 'v')

    {

        fi.getline(readbuf, 1145);

        fii >> ch;

    } //此时已读完字符v

    int \*result = new int[varNum];

    memset(result, 0, sizeof(int) \* varNum);

    for (int i = 0; i < varNum; i++)

    {

        fii >> solution;

        result[abs(solution)] = solution;

    }

    result[0] = 0;

    for (i = 0; i < clauseNum; i++)

    { //读取所有子句

        clause\_value[i] = false;

        j = 0;

        fi >> bv[i][j];

        do

        {

            if (result[abs(bv[i][j])] == bv[i][j])

                clause\_value[i] = true;

            j++;

            fi >> bv[i][j];

        } while (bv[i][j] != 0);

    }

    for (i = 0; i < clauseNum; i++)

    { //输出代入后的

        cout << "第" << i + 1 << "个子句:" << clause\_value[i] << endl;

        if (0 == clause\_value[i])

            ans = 0;

    }

    for (i = 0; i < clauseNum; i++)

    {

        if (0 == clause\_value[i])

        cout << "第" << i + 1 << "个子句:" << clause\_value[i] << endl;

    }

    if (ans == 1)

        cout << "All correct!" << endl;

    else

        cout << "Error!" << endl;

}