**课程设计报告**

**题目： 基于SAT的数独游戏求解程序**

**课程名称： 综合程序设计课程设计**

**专业班级： CS1903班**

**学 号： U201914376**

**姓 名： 沈承磊**

**指导教师： 袁凌**

**助 教： 王振江**

**报告日期： 2021年9月29日**

**计算机科学与技术学院**

**任 务 书**

**1.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。

**2.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问题求解，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%)

**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] 严蔚敏等. 数据结构(C语言版). 北京：清华大学出版社

**目 录**

[1引言 1](#_Toc5551118)

[1.1 课题背景与意义 1](#_Toc5551119)

[1.2 国内外研究现状 1](#_Toc5551120)

[1.3 课程设计的主要研究工作 1](#_Toc5551121)

[2 系统需求分析与总体设计 1](#_Toc5551122)

[2.1 系统需求分析 2](#_Toc5551123)

[2.2 系统总体设计 2](#_Toc5551124)

[3 系统详细设计 3](#_Toc5551125)

[3.1 有关数据结构的定义 3](#_Toc5551126)

[3.2 主要算法设计 4](#_Toc5551127)

[4.系统实现与测试 11](#_Toc5551128)

[4.1 系统实现 11](#_Toc5551129)

[4.2 系统测试 19](#_Toc5551130)

[4.3程序优化 23](#_Toc5551131)

[5 总结与展望 24](#_Toc5551132)

[5.1 全文总结 25](#_Toc5551133)

[5.1 工作展望 25](#_Toc5551134)

[6体会 26](#_Toc5551135)

[7附录 27](#_7附录)

[7.1主函数 27](#_Toc5551137)

[7.2各操作函数 31](#各操作函数)

[7.3 头文件 32](#头文件)

**1引言**

**1.1 课题背景与意义**

对于计算机科学与技术、信息安全与物联网专业大二学生，在前三个学期已经学习了C语言程序设计、数据结构两门面向编程知识与技术的基础理论课，已经具备具有初步的程序设计、数据抽象与建模、问题求解与算法设计的能力。

可满足性问题是人工智能领域研究的一个核心问题，其应用范围极广，在 计算机辅助设计，机器视觉，数据库等方面均有涉及。合取范式的可满足性问题是理论计算机科学与人工智能的著名问题，寻求其有效算法，一直是计算机连及实际应用的重要任务。

本文简单的实现了基于DPLL的SAT算法，采用邻接表的数据结构和最短正字句的决策方式，并将算法应用在数独问题的求解过程中。

**1.2 国内外研究现状**

近年来，可满足性问题研究逐渐升温，国内国际上对SAT问题的研究热度持续升温，目前，各种不同的局部搜索算法和回溯搜索算法的提出，使得SAT解决器解决不同领域中的SAT问题的能力不断增强，能解决的问题的规模不断增大。

其中局部搜索算法显示出对于随机的SAT问题特别有用，而回溯搜索算法则被用来解决大规模实际应用领域中的SAT问题。事实上，国际上已提出了一大批采用回溯搜索算法的高效的SAT问题解决器，其中绝大多数提出来的回溯搜索算法是对原始的DPLL回溯搜索算法的改进算法。这些改进措施包括:新的变量决策策略，新的搜索空间剪除技术，新的推理和回溯技术以及新的更快的算法实现方案和数据结构……当前水平的SAT问题求解器已能够轻松解决以前传统 SAT问题解决器完全无法解决的可满足性问题。

尽管当前的SAT问题解决器已取得了相当重要的进步，但是研究的脚步不会停止，我们还可以提出一些值得研究的问题。比如，是否存在新的更高效的SAT 问题处理技术可以集成到DPLL算法框架内；是否可以找到除局部搜索，回溯搜 索之外的其他SAT算法来更有效地解决SAT问题；是否能提出更好的SAT改进算法和实现方案。

**1.2 课程设计的主要研究工作**

本次实验课题为“基于SAT的数独游戏求解程序”，要求实现SAT求解器和数独游戏两个功能。

DPLL算法是经典的SAT完备型求解算法，对给定的一个SAT问题实例，理论上可判定其是否满足，满足时可给出对应的一组解。本设计基于DPLL的算法与程序框架，实现一个完备SAT求解器，包括程序的改进也必须在此算法的基础上进行，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。

数独游戏可转化为SAT问题，用本系统实现的SAT求解器可以快捷地对数独问题转化的CNF文件进行求解，再以变元真值数据转化的数独盘格式输出求解答案。本系统具有一定的交互功能，用户可以利用本系统进行数独游戏，系统将自动判断解的正确性，并输出正确答案。

**2 系统需求分析与总体设计**

**2.1 系统需求分析**

本设计基于DPLL的算法与程序框架，实现一个完备SAT求解器，并基于此完成数独游戏求解的设计。

## 2.1.1 SAT求解器

要求SAT求解器可对输入的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，输出与将求解结果存入同名文件（文件拓展名为.res），统计求解时间。

## 2.1.2 数独游戏

数独部分包括生成可玩的数独游戏及求解数独CNF范式算例文件两部分。数独游戏板块输出含空格的数独游戏盘，并给予操作提示，用户可输入自己求解的答案，系统将检验其答案的正确性，答案错误时可选择直接查看正确答案或继续解题。数独文件求解后系统将以数独终盘的形式输出求解结果。

系统通过菜单显示操作类别，用户可自行选择要进行的操作。

**2.2 系统总体设计**

这部分可根据用户需求，设计和规划一个系统，说明清楚系统应该有哪些功能模块，每个模块做什么。最后给出完整的系统模块结构图。

## 2.2.1 SAT求解器

SAT求解器求解CNF范式算例文件，实现该功能包括4个部分：

（1）CNF范式算例文件解析功能模块，建立其对应的数据存储结构；

（2）基于DPLL算法的SAT求解模块，对文件算例进行求解。该部分模块的功能实现需进行对CNF范式的删除子句，删除文字，判断单子句，判断空字句等操作，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。

（3）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。

（4）进入文件保存模块，将求得的结果保存入CNF算例同名文件（拓展名为.res）中。

## 2.2.2 数独游戏

数独游戏模块包括2个功能部分：

（1）生成可玩的数独游戏：

利用SAT求解器生成随机的数独终盘，再基于挖洞法生成有唯一解的数独游戏盘，输出至用户界面，用户可选择挖洞数量进而自行调整难度，在生成初盘后将数独的CNF文件存入相应位置。

（2）求解数独CNF范式算例文件：

读取文件中数独残局信息，建立变元与数独格取值的对应关系，再利用SAT求解器求解数独文件，此过程用户可选择是否显示读入的CNF文件（解析）。求解完成后将解得的真值信息转换为二维数组并以数独盘的格式输出。

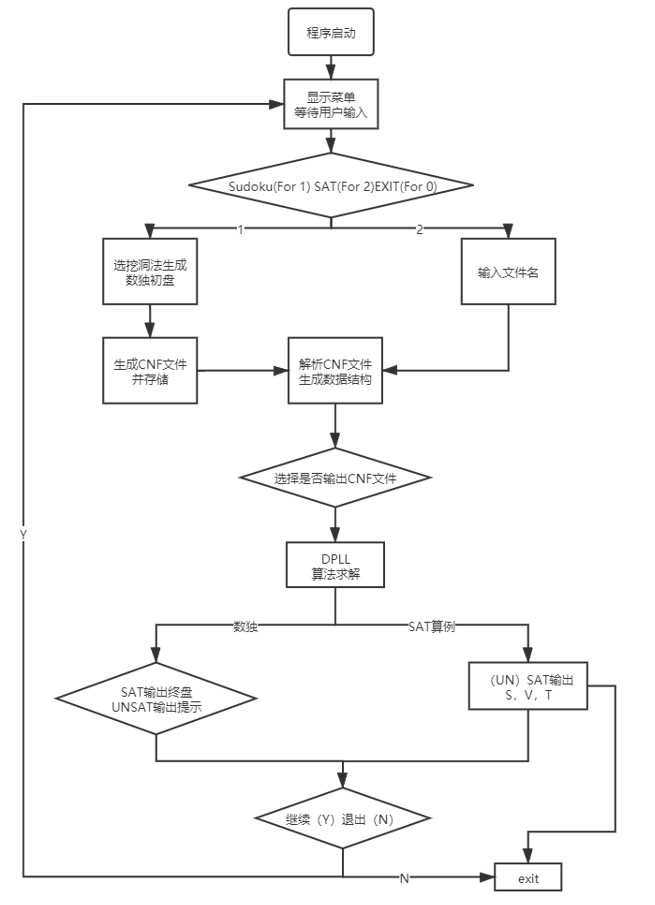
系统模块结构图如图2-1所示

图2-1 系统模块结构图

**3 系统详细设计**

**3.1 有关数据结构的定义**

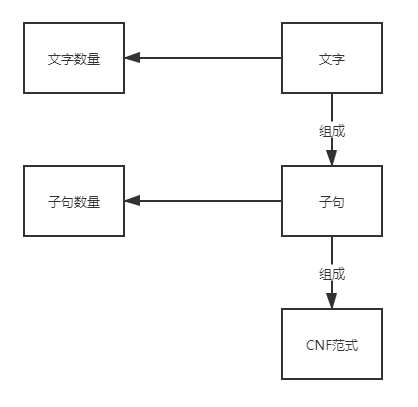
## 3.1.1 数据结构的定义

本次实验主要运用了二维线性链表，一位数组，二维数组三种数据结构来存储信息：二维线性链表用于存储CNF范式的子句和文字信息，一位数组用于存储CNF算例中各变元的真假信息，二维数组则用于存储挖洞数独中“洞”的信息，以及数独本身的信息，以便于输出。

1. 系统中需要处理的数据，以及每种数据所包含的数据项，如表3-1所示。

表3-1 数据以及数据项

|  |  |  |
| --- | --- | --- |
| 需要处理的数据 | 所包含的数据项 | 数据类型 |
| CNF文件 | 文字 | 整型 |
| 文字个数 | 整型 |
| 子句 | 链表 |
| 子句数量 | 整型 |

1. 数据之间的关联，如图所示。
2. 各数据结构类型的定义如下：

（a）存储子句链表文字结点的数据结构类型定义：

typedef struct DataNode {

int data = 0;//数字记录变元编号

DataNode \*next{};//指向同一子句下一结点指针

}DataNode;

（b）存储CNF范式链表结点（即子句表头结点）的数据结构类型定义：

typedef struct HeadNode {

int Num = 0;//记录该子句文字结点个数

DataNode \*right{};//指向该子句第一个文字结点

HeadNode \*down{};//指向下一字句表头结点

}HeadNode;

（c）存储单个变元真假值的数据结构类型定义：

struct consequence {

int value = -1;//存真值 真时为true，假时为false

};//普通CNF文件

struct conse {

int num = 0;

int value = -1;//存真值 真时为true，假时为false

};//数组CNF文件

（d）存储各个变元出现次数的数据结构类型定义：

struct Count {

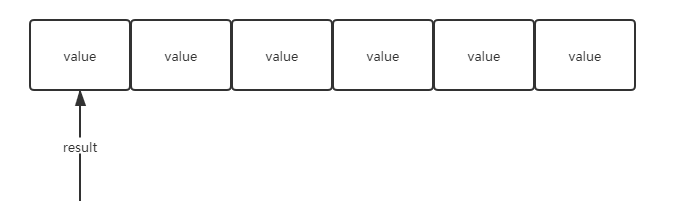
int num = 0;

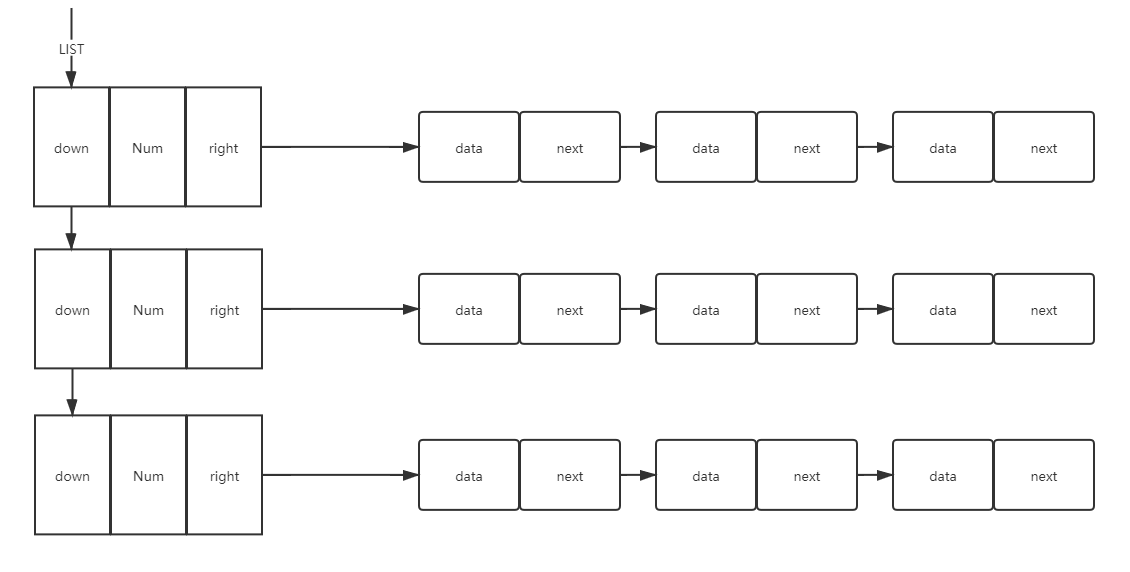
int count = 0;

Count \*next=nullptr;

};

## 3.1.2 数据结构的关联

（a）存放变元真值的数组结构示意图：

（b）存储CNF文件内容的链表结构示意图：

## 3.1.3 相关常量声明

#define TRUE 1

#define FALSE 0

#define ROW 9

#define COL 9

#define NoAnwser -1

## 3.1.4 相关操作说明

1. **HeadNode\* CreateClause(int &,string &);**

**功能：**用文件指针fp打开用户指定的文件，并读取文件内容保存到给定参数中，读取成功返回1，失败返回0。

**算法实现：**依次读取文件内容，读取到cnf时，记录后面的变元数和句子数。再依次开辟十字链表空间，记录每一条句子和文字。

**时空效率分析：**依次遍历每一条句子的每个文字，时间复杂度为O(n\*m)，需给十字链表开辟空间，空间复杂度为O(n\*m)，n和m分别为句子数量和每个句子所含文字数量。

1. **void DeleteHeadNode(HeadNode\*,HeadNode\*&);**

**功能：**销毁链表。

**算法实现：**设置记录变量，遍历销毁十字链表的每一句子和每一个文字。

**时空效率分析：**需遍历每一个句子和每一个文字，时间复杂度为O(n\*m)，空间复杂度为O(1)，n和m分别为句子数量和每个句子所含文字数量。

1. **HeadNode\* IsSingleClause(HeadNode\*)**

**功能：**判断是否为单子句，是返回1，不是返回0。

**算法实现：**判断传入参数句子是否符合“头指针不为空，下一指针为空”的条件，符合即为单子句，否则不是单子句。

**时空效率分析：**时间复杂度和空间复杂度均为O(1)。

1. **int evaluateClause(HeadNode\*, int v[]);**

**功能：**评估子句的真假状态，真返回1，假返回0。

**算法实现：**根据结果记录数组v里面的值，如果参数句子cnf里的某一个文字真值与v中的对应，则该句子为真，否则为假。

**时空效率分析：**需遍历该句子的所有文字，时间复杂度为O(m)，空间复杂度为O(1)，m是句子中的文字数量。

1. **void DeleteHeadNode(HeadNode\*,HeadNode\*&);**

**功能：**在已有的十字链表中删除指定的子句，删除成功返回1，失败返回0。**算法实现：**若删除为头，则使头指针指向下一个结点，然后释放内存；若删除为其他位置，找到该结点的直接前驱，使其next指针指向要删除结点的下一结点，然后释放删除结点内存。

**时空效率分析：**需遍历查找指定句子，时间复杂度为O(n)，空间复杂度为O(1)，n是链表中句子的数量。

1. **void DeleteDataNode(int,HeadNode\*&);**

**功能：**在指定的子句中删除指定的文字，删除成功返回1，失败返回0。

**算法实现：**若删除为头，则使头指针指向下一个结点，然后释放内存；若删除为其他位置，找到该结点的直接前驱，使其next指针指向要删除结点的下一结点，然后释放删除结点内存。

**时空效率分析：**需遍历查找指定文字，时间复杂度为O(m)，空间复杂度为O(1)，m是句子中文字的数量。

1. **HeadNode\* ADDSingleClause(HeadNode\*,int);**

**功能：**在已有的十字链表中添加指定的子句，添加成功返回1，失败返回0。

**算法实现：**直接添加在已有链表头部，使该结点next指针指向原链表头，然后原链表头赋值为该节点。

**时空效率分析：**时间复杂度和空间复杂度均为O(1)。

1. **status IsEmptyClause(HeadNode\*);**

**功能：**判断是否含有空子句，是返回1，不是返回0。

**算法实现：**遍历所有句子，若出现某一句子不含任何文字即head为空，则含有空子句，否则不含。

**时空效率分析：**需遍历链表中所有句子，时间复杂度为O(n)，空间复杂度为O(1)，n是子句的数量。

1. **status DPLL1(HeadNode \*LIST,consequence \*result);**

**status DPLL2(HeadNode\* LIST, consequence\* result);**

**功能：**求解SAT问题，给出满足条件时的一个式子,若有解则返回1，无解返回0。

**算法实现：**

1）查找链表中的单子句tp，使结果记录数组v中对应记录为真，即v[tp->data]=1或v[-tp->data]=0。

2）删除所有含文字tp->data的子句，以及删除所有负文字-tp->data。

3）删除完后，判断链表是否为空，若为空，则说明问题已解决，返回1；判断链表中是否含有空子句，若有，则说明问题无解，返回0；

4）重复1）、2）、3)直到链表中不出现单子句。

5）选取变元a策略：记录每一个变元出现的次数（正文字和负文字区别开来），若链表中有正文字存在，则在正文字中找到出现次数最多的变元，该变元为所选变元。若链表中只有负文字，则在负文字中找到出现次数最多的变元，该变元为所选变元。

6）在原有链表中添加只含文字a的单子句，递归DPLL函数，从步骤1）开始继续运算，若结果为1，则问题求解成功，返回1；若结果为0，则在原有链表中添加只含文字-a的单子句，再次递归DPLL函数，并直接返回求解结果。

**时空效率分析：**需遍历每个句子查找单子句，且要遍历每个文字找到出现次数最多的变元，即每一步的时间复杂度为O(n\*m)，最坏情况下，总共需走2^b步，即总时间复杂度为O(n\*m\*2^b)，每一步要记录先前的链表，空间复杂度为O(b)，n是链表子句数量，m是句子中文字的数量，b是总变元的数量。

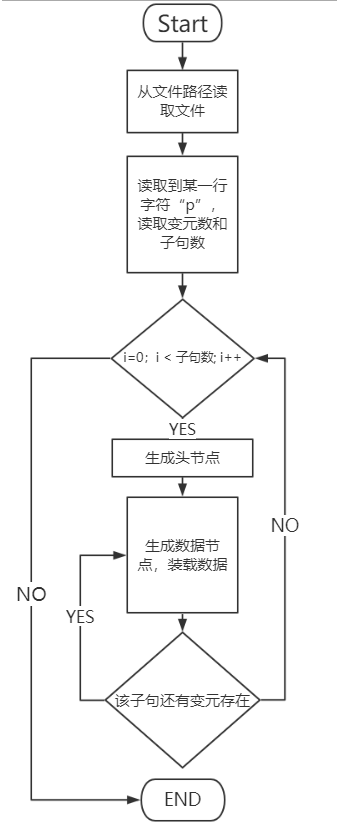
**3.2 主要算法设计**

本系统的主要算法包括对CNF范式数据结构的操作、DPLL算法和数独生成算法三部分。

## 3.2.1 CNF范式数据结构的相关操作

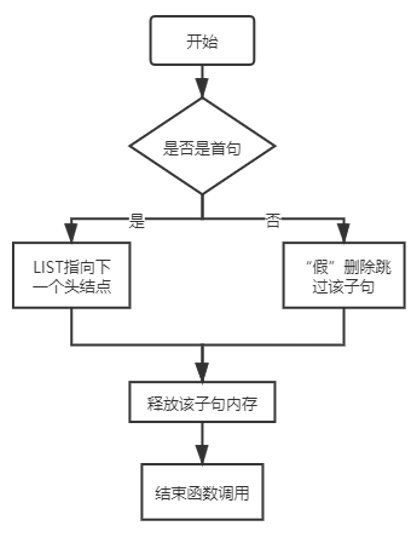
对CNF范式数据结构的操作（如生成结构，删除子句，删除文字，判断单子句等）是DPLL函数的基础。

由于DPLL算法基于递归原理的探测操作，需要将结构复原为操作之前的状态，在这里首先采取一种较为直观的复制函数记录操作前的状态，从而得到需还原的CNF范式。因此可以方便地进行文字和子句的删除以及复原操作。

1. 解析CNF模块，包括读入变元，创建链表等步骤，其程序流程图如图所示。

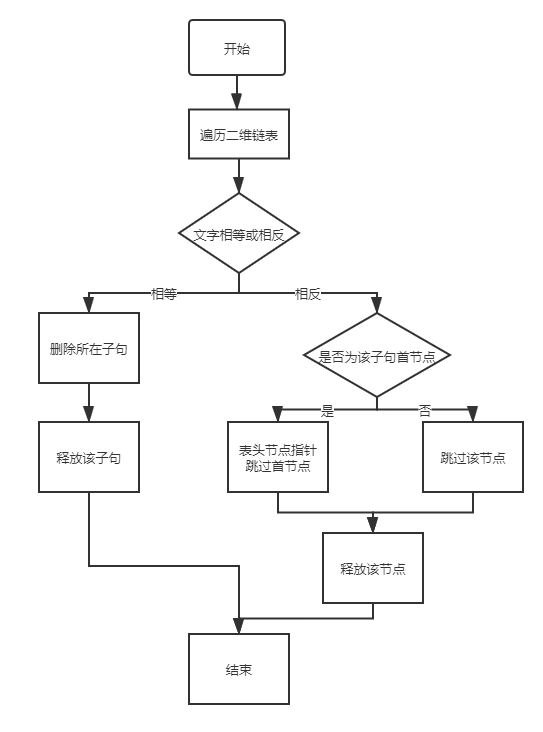
解析并创建CNF范式结构模块功能的实现包括HeadNode\* CreateClause(int &,string &)，CreateClause函数读取文件内每个子句的第一个文字，生成子句链表的头结点链表，并传文字值、文件指针地址及创建的子句链表头结点地址至CreateClause函数中，CreateClause函数再继续读文件并生成子句链表，同时创建变元表内变元正负文字的信息链表，记录并返回每个子句内的文字数量至头结点中。

1. 删除子句程序流程图：

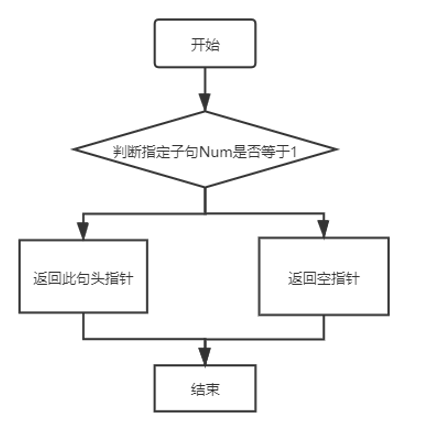


删除子句操作是DPLL函数中CNF范式化简过程的基本操作之一，由于本系统采用“假删除”记录子句和文字是否被删除，故删除子句时，只需将需删除的子句前后结点指针指向改变则表示子句已被删除，删除子句由函数DeleteHeadNode(HeadNode\*,HeadNode\*&)实现。

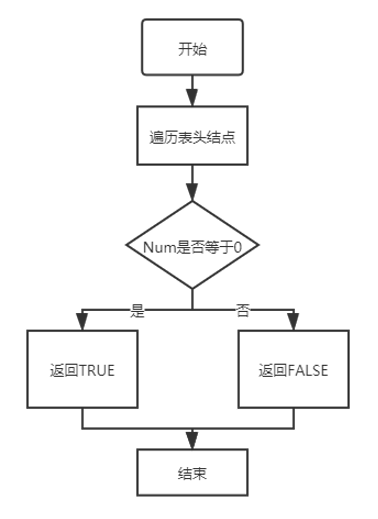
1. 删除文字程序流程图：



删除文字操作也是DPLL函数中CNF范式化简过程的基本操作之一，与删除子句操作相同，删除文字操作的方法同样是用“假删除”将相应文字结点前后指针指向改变。删除文字操作由函数DeleteDataNode(int,HeadNode\*&)实现，其中参数l是变元的数字编号，变元2是子句头指针。

1. 判断单子句程序流程图：

由于本系统中CNF链表结构内的子句链表头结点结构内用num变量存储了子句内剩余的文字数目，故只需遍历CNF链表的头结点链表，寻找number值为1的结点即为单子句链表头结点。系统判断单子句操作由函数HeadNode\* IsSingleClause(HeadNode\*)实现。

1. 判断是否存在空子句程序流程图：

由于本系统中CNF链表结构内的子句链表头结点结构内用num变量存储了子句内剩余的文字数目，故只需遍历CNF链表的头结点链表，寻找number值为1的结点即为单子句链表头结点。系统判断单子句操作由函数status IsEmptyClause(HeadNode\*)实现。

## 3.2.2 DPLL算法

DPLL算法是SAT求解器的核心算法，利用递归的思想，和单子句策略、分裂策略两个策略将CNF公式逐步化简。单子句策略是指挑出CNF范式中的所有单子句，易得要使CNF范式有解其文字真值必为1，进而可以得出公式中所有含有该文字的子句真值均为1，可以删去，同时公式中所有该文字的反文字真值皆为0，可以删去子句中的该文字的反文字；分裂策略是指按某种策略选出一个未赋真值的变元的文字，设其真值为1，对CNF范式进行探测，再循环执行单子句策略和分裂策略，如若过程中某个子句中只剩下设真值为1的文字的反文字，说明此次探测失败，则需返回前一次分裂策略，得该选择文字的真值必为0，再重新探测CNF范式。递归式DPLL函数伪代码框架如下：

**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

**基于最短正字句策略选取变元*v*；//策略对DPLL性能影响很大**

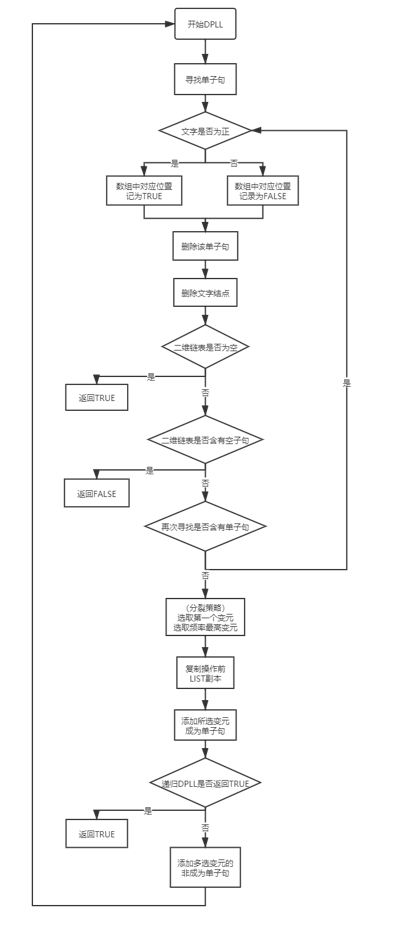
**if DPLL（*S* ∪*v* ）return(TURE);**

**return DPLL(*S* ∪¬*v*);**

**}**

DPLL中，使用的函数有回溯（复制），查找单子句，用单字句规则化简，判断cnf文件真假状态，配合DPLL完成CNF文件的求解。

其DPLL框架流程图如图所示。

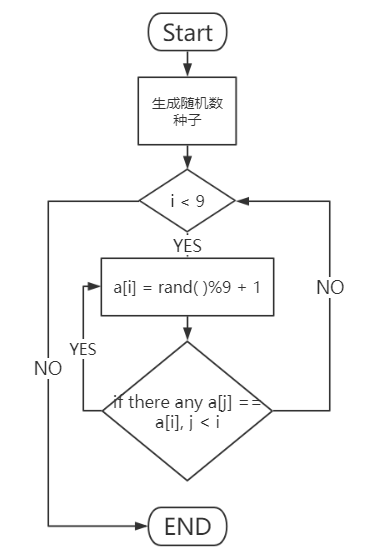


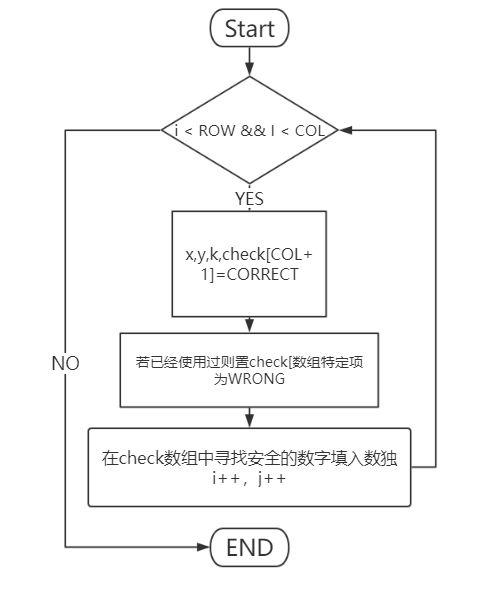
## 3.2.3 数独生成

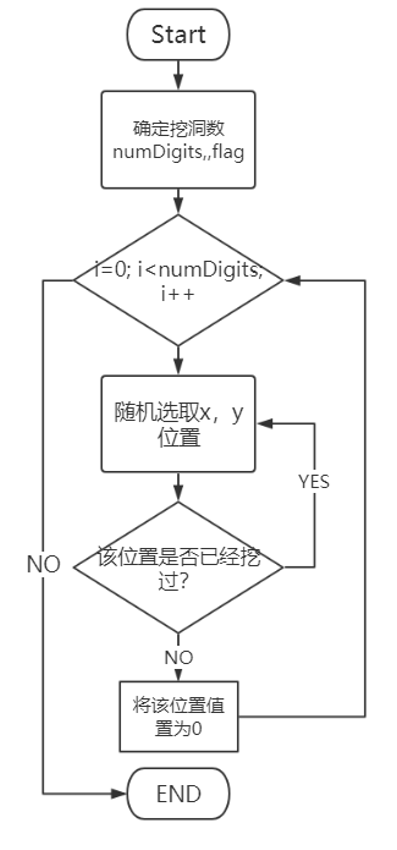
该模块包括三部分的内容：数独部分包括三个核心算法：数独终盘及初盘的生成、转换为CNF文件、挖洞算法获取只有唯一解的数独残局、数独文件的求解以及基于easyx的界面展示。其中数独终盘创建及挖洞算法用于实现系统数独功能的数独游戏功能模块。

1. 数独终盘及初盘的生成；主要包括生成9\*9的数独，第一步，建立随机种子，生成第一排随机数，第二步，按照数独规则生成棋盘剩余数字；第三步，挖洞，随机选择不同的位置将该位置的数置零。第四步，将所有的约束条件以及数独残局信息转换为CNF写入文件；

（a）建立随机种子程序流程图：



（b）生成剩余数字的程序流程图：

（c）挖洞算法的程序流程图：

1. 转换为CNF文件

将数独求解转换为DPLL文件。主要包括：

每个单元的约束条件，如对于（1，1）有 111 112 113 114 115 116 117 118 119 0

两两组合： -111 -112 0

-111 -113 0

…………

行约束：考虑第一行

包含1：111 121 131 141 151 161 171 181 191 0

包含2：112 122 132 142 152 162 172 182 192 0

…………

两两组合： -111 -121 0

-111 -131 0

列约束，考虑第一列

包含1：111 121 131 141 151 161 171 181 191 0

包含2：112 122 132 142 152 162 172 182 192 0

…………

两两组合： -111 -211 0

-111 -231 0

块约束，考虑第一块

包含1：111 121 131 211 221 231 311 321 331 0

包含2：112 122 132 212 222 232 312 322 332 0

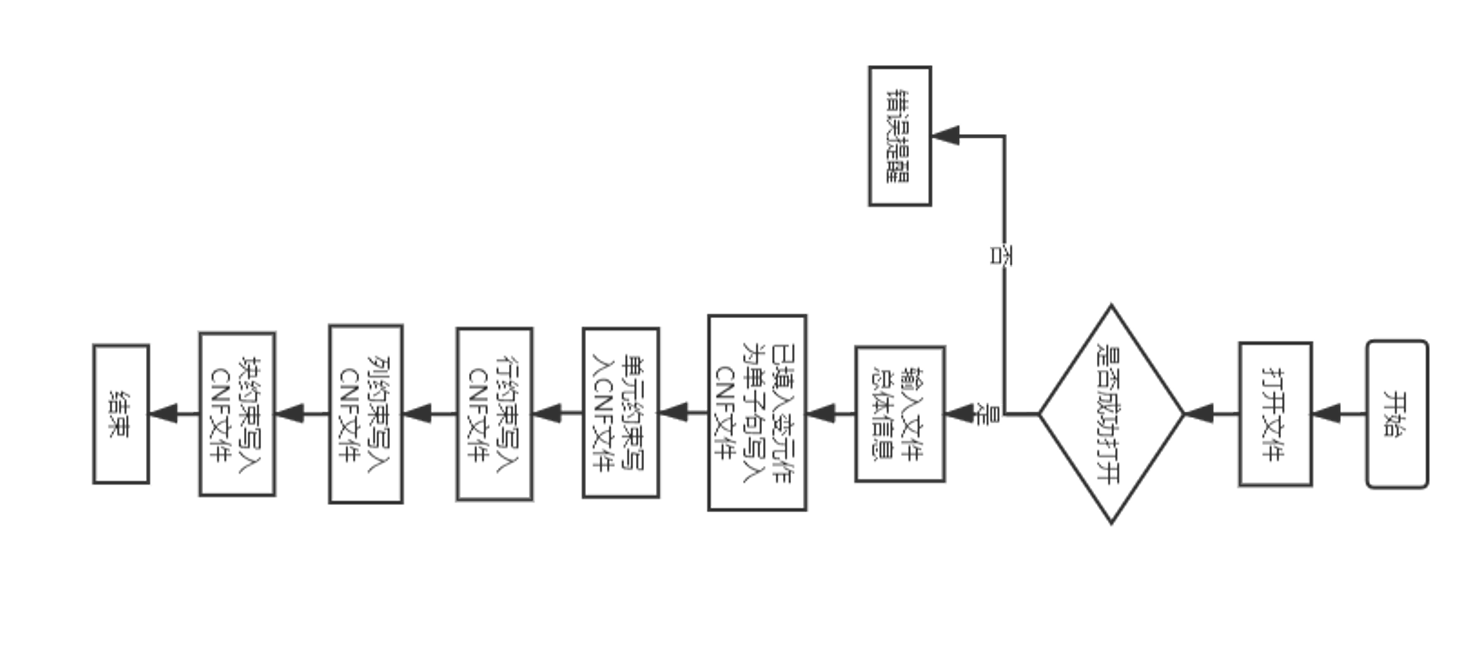
…………

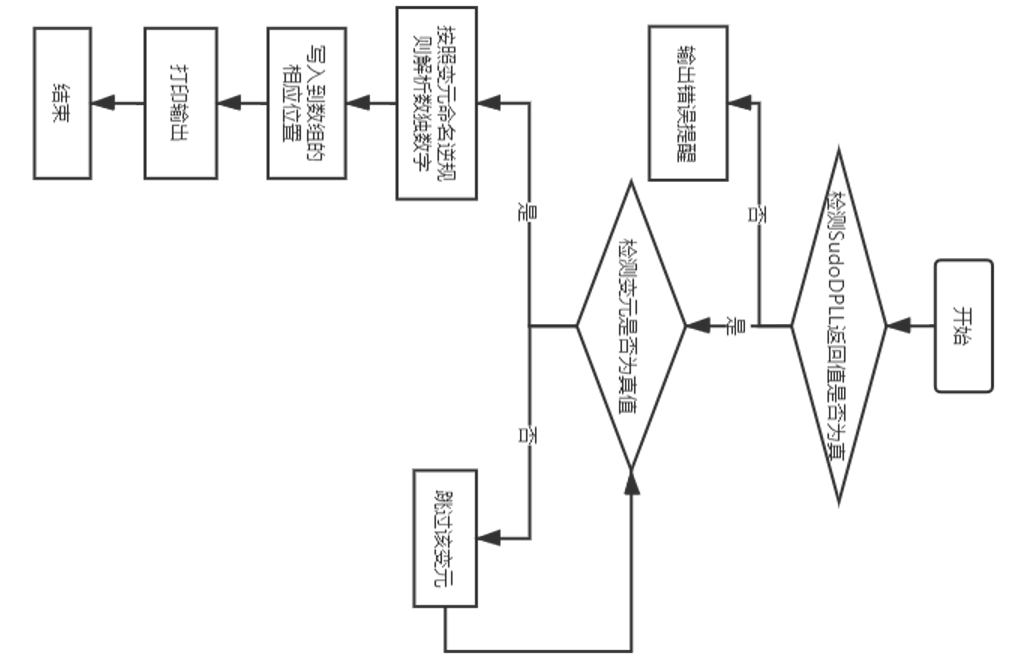
两两组合： -111 -211 0

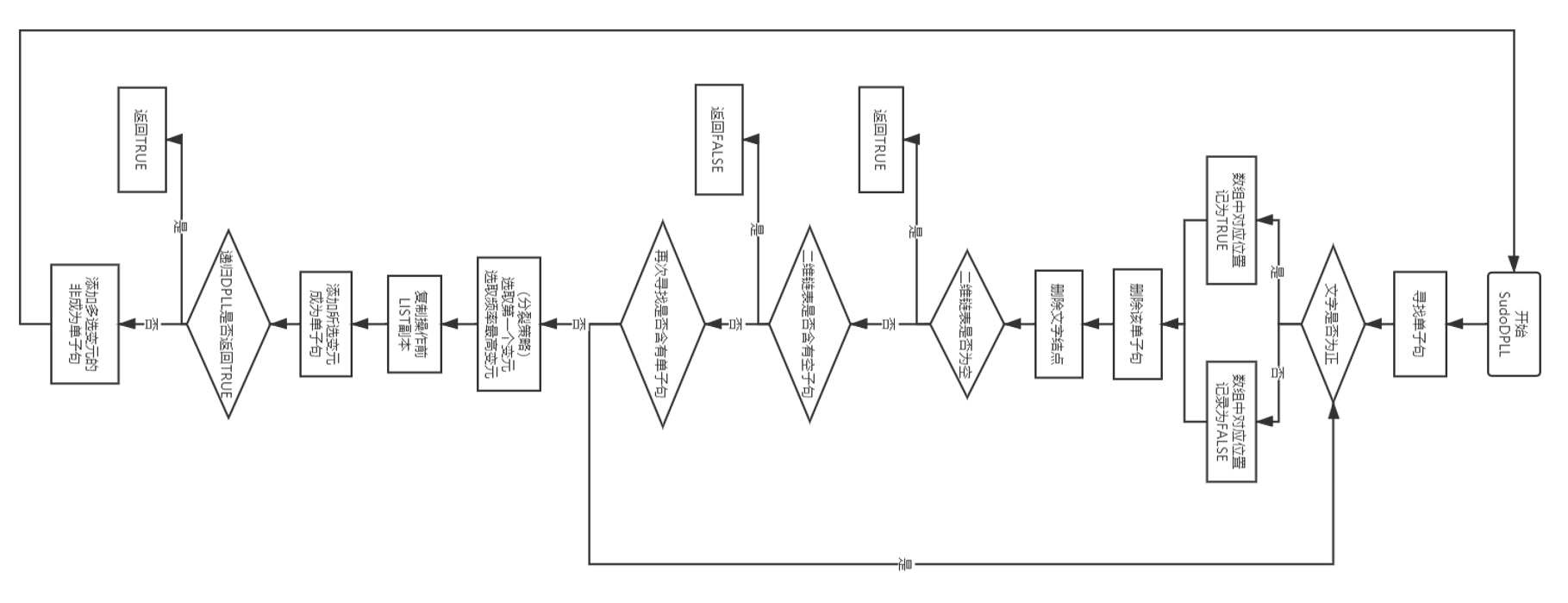
-111 -231 0

单子句约束，如（3,6）包含 5有单子句 365 0

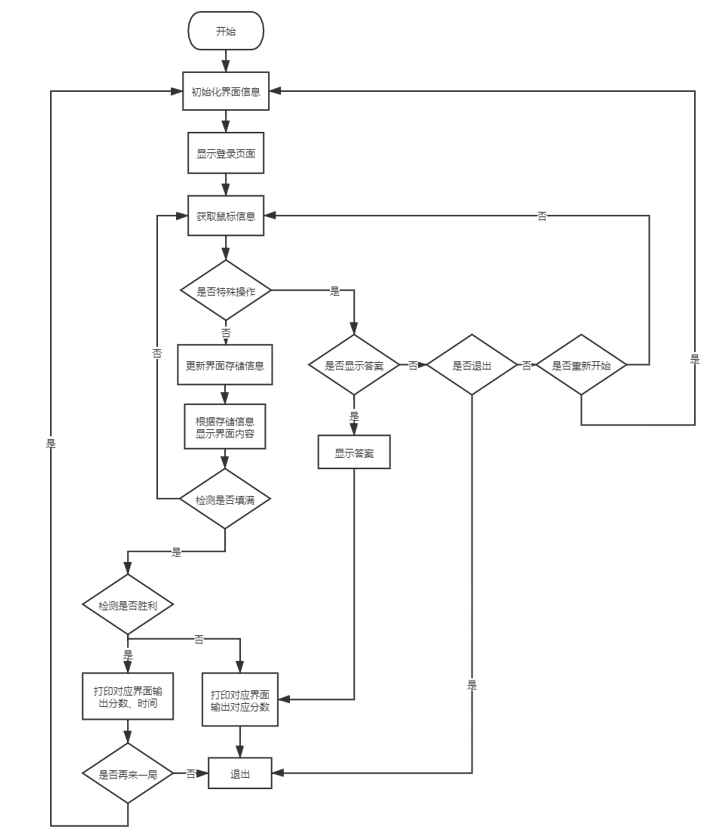
其流程图如图所示：



1. 数独游戏的求解：利用1）中有数独棋盘转换成的CNF文件，并用DPLL算法求解，最后解析数独中变元名，输出棋盘，流程图如下：

其中SudoDPLL函数的算法基本与DPLL函数一样，流程图如下：

1. Easyx的界面实现：

Easyx是一款C/C++的图形库，可以帮助C/C++语言程序员快速上手图形和游戏的编程。他主要涉及四个方面：传入挖洞后可游戏的初盘、显示界面并动态刷新、根据界面操作更新界面存储信息、根据数独游戏规则检测待填数字。程序流程图如下图所示：

**4.系统实现与测试**

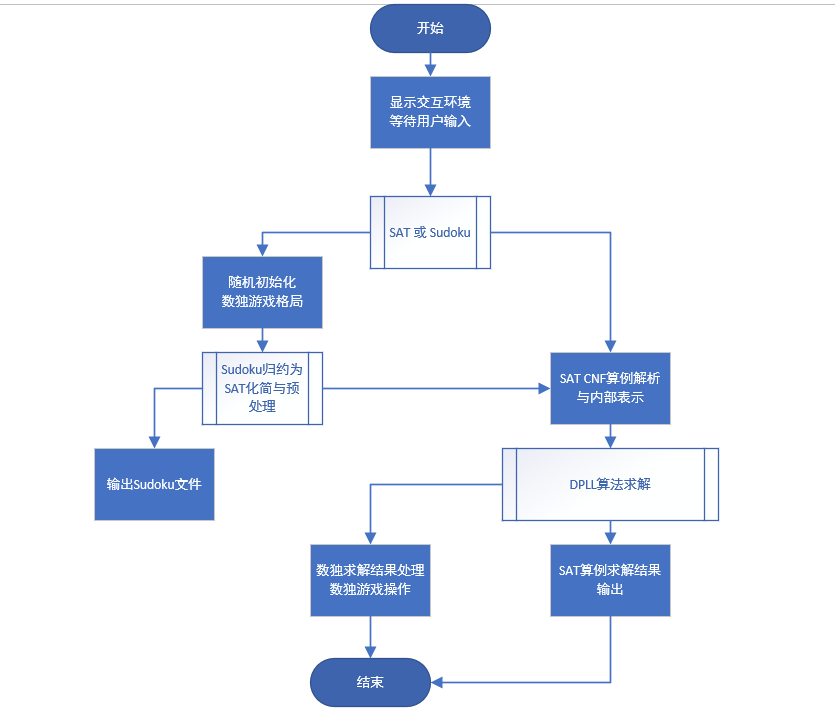
**4.1 系统实现**

1. 系统实现环境

软件环境：visual studio2019；

硬件环境：intel 酷睿i5-7500 内存8G；

1. 用户需求：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，并将DPLL求解应用到数独求解具体问题中；
2. 处理流程，如图所示。



1. 用C语言定义的各种数据结构；
2. CNF文件存储结构，采用二维链表定义，物理结构见图 3.1图；

函数如下所示，完整函数参见附录；

struct consequence {

int value = -1;//存真值 真时为true-1，假时为false-0

};

typedef struct DataNode {//二维链表的文字结点

int data = 0;

DataNode \*next{};

}DataNode;

typedef struct HeadNode {//二维链表的头结点

int Num = 0;

DataNode \*right{};//指向右边第一个文字结点

HeadNode \*down{};//指向下一个子句

}HeadNode;

1. CNF解析模块，使用的函数，完整代码详见附录；

读入文件：

string HFilePath = R"(D:\VS\_project\SAT\_DPLL\)";

string path = HFilePath + filename;

ifstream fis(path);

复制文件：

HeadNode\* Duplication(HeadNode\*);//用于回溯

1. 核心DPLL模块，使用的函数；

status DPLL(HeadNode \*LIST,consequence \*result);

HeadNode\* IsSingleClause(HeadNode\*);

status IsEmptyClause(HeadNode\*);

HeadNode\* ADDSingleClause(HeadNode\*,int);

HeadNode\* Duplication(HeadNode\*);

void DeleteHeadNode(HeadNode\*,HeadNode\*&

void DeleteDataNode(int,HeadNode\*&);

**4.2 系统测试**

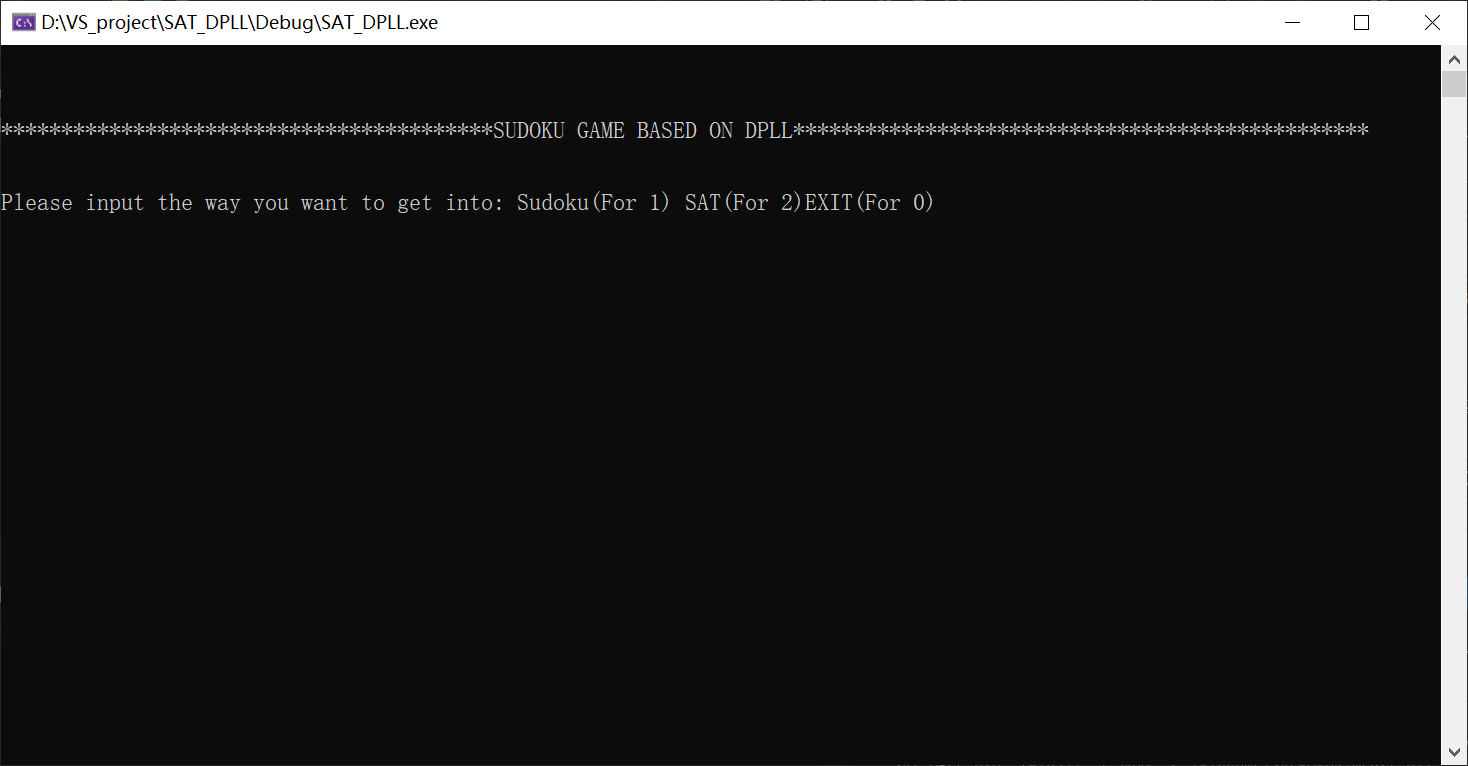
 系统与用户交互界面如图1所示。

图1用户交互界面

1. CNF解析模块；
2. 模块说明；

该模块为本课程设计的核心模块，其是在DPLL框架下展开的；

1. 测试方法；

读取cnf算例文件，解析文件，基于一定的林基表的结构，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。

1. 运行结果；

测试用例1：如图2基准满足算例problem3-100.cnf。

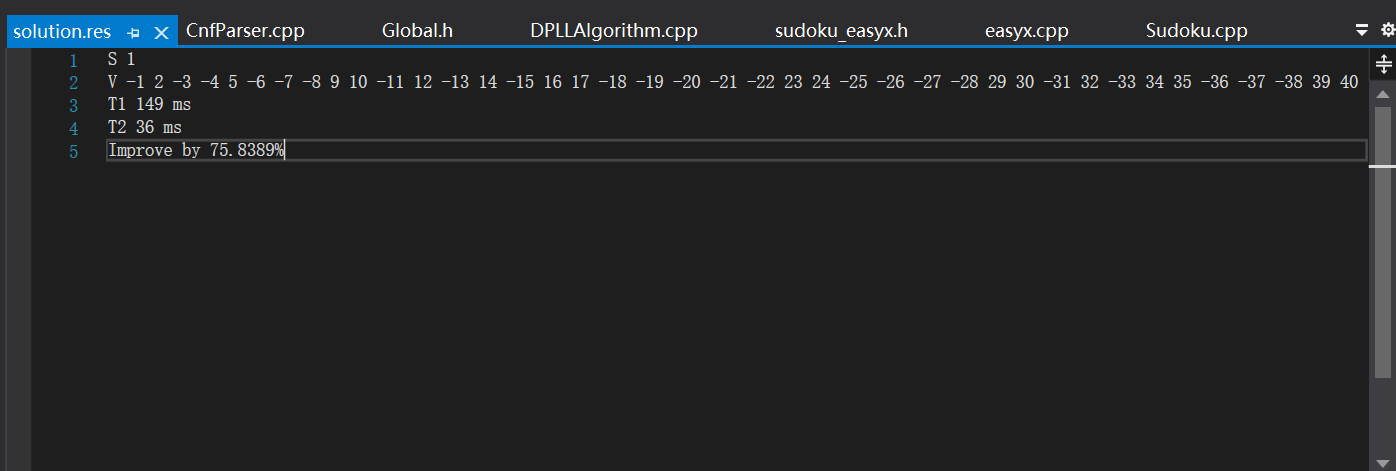


图2基准满足算例problem3-100.cnf

测试用例二：如图3中型基准满足算例sud00082.cnf所示。

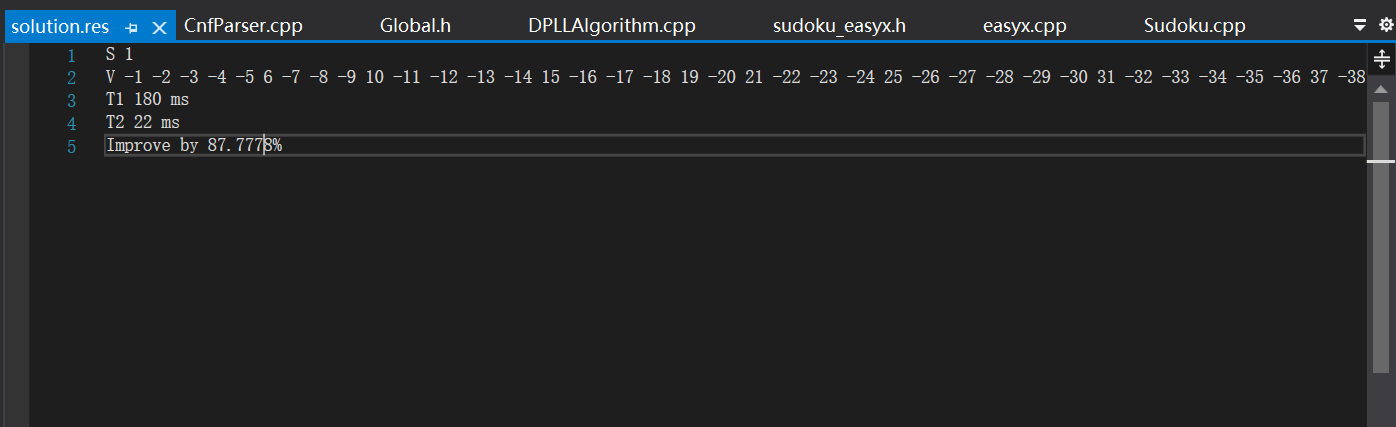
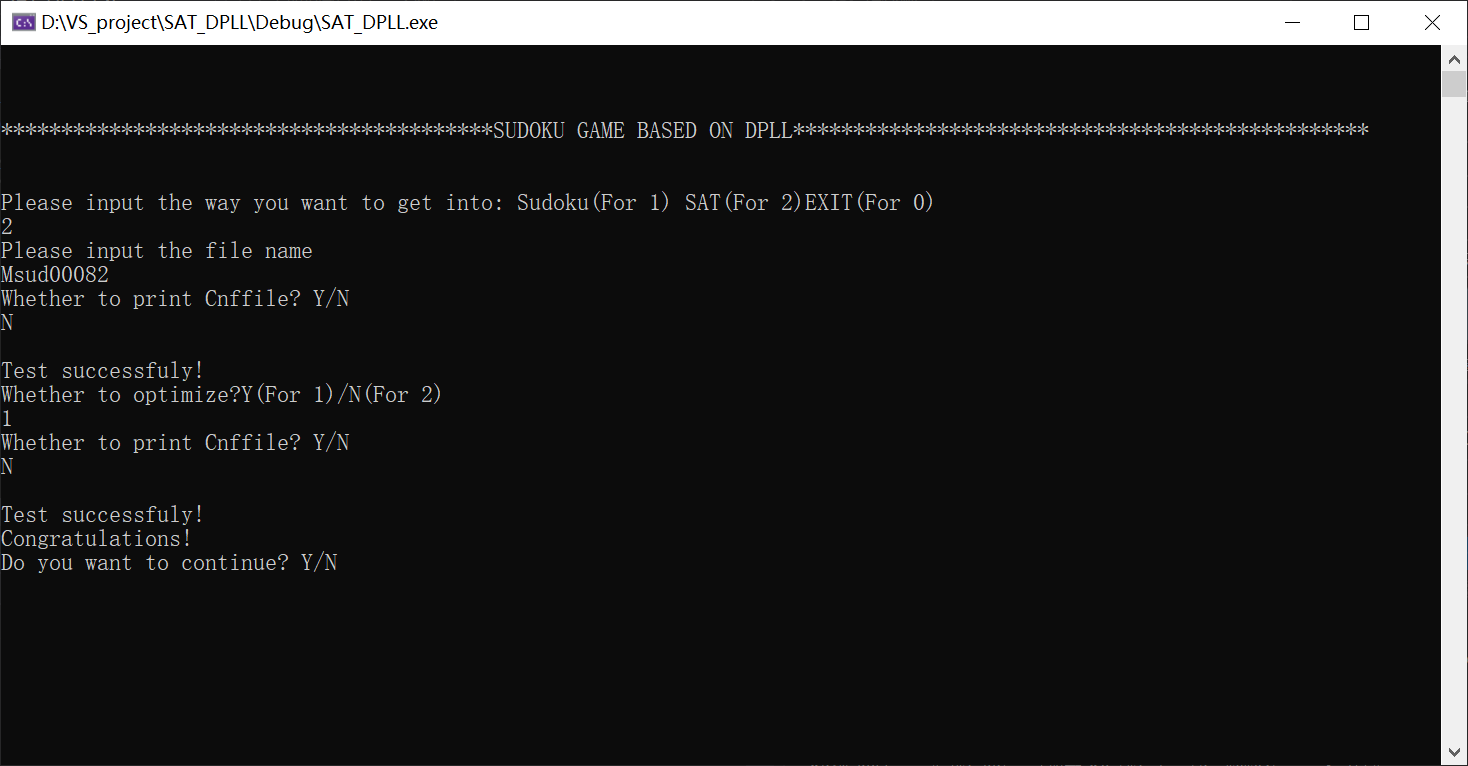
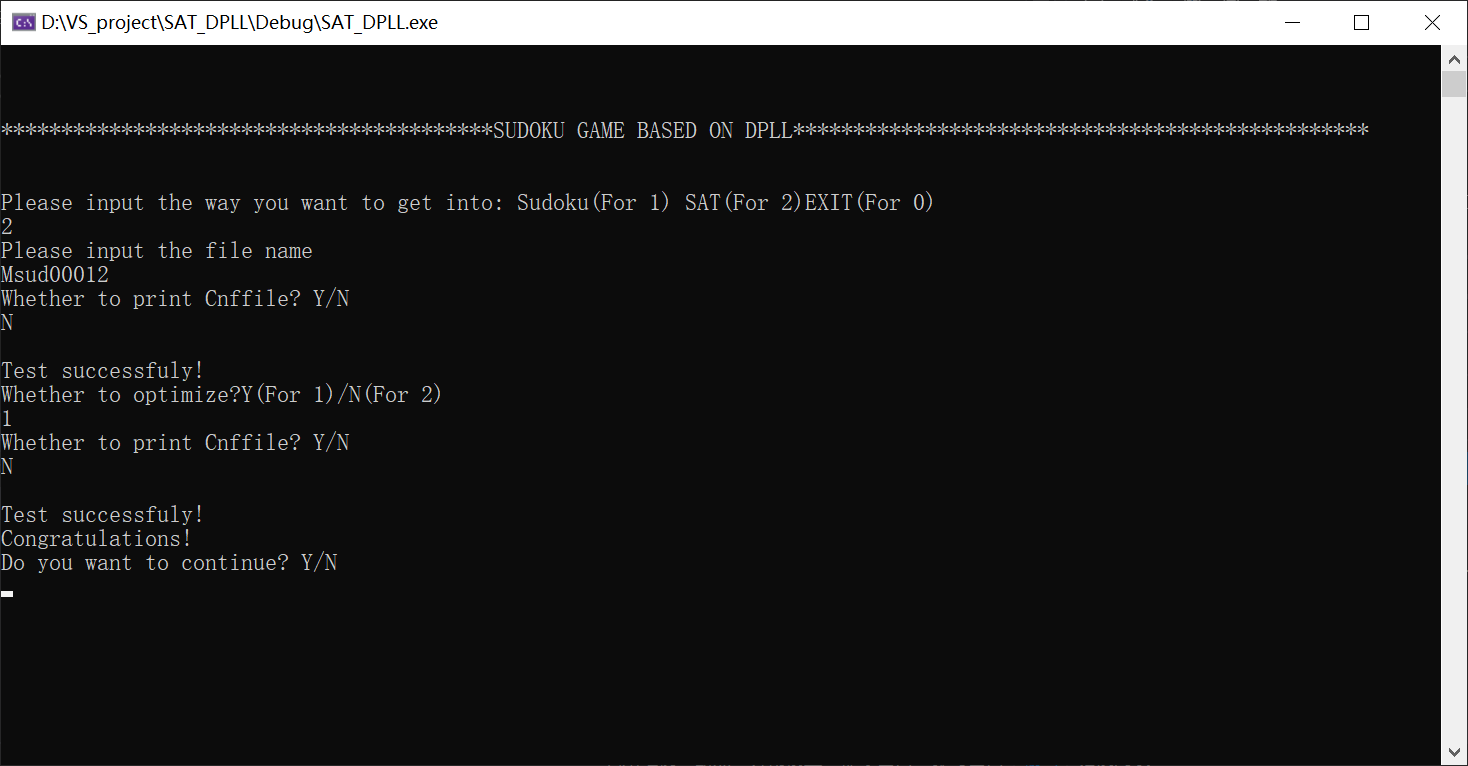
运行截图：

图3小型基准满足算例sud00082.cnf

测试算例三：中型基准满足算例sud00012.cnf

运行截图：

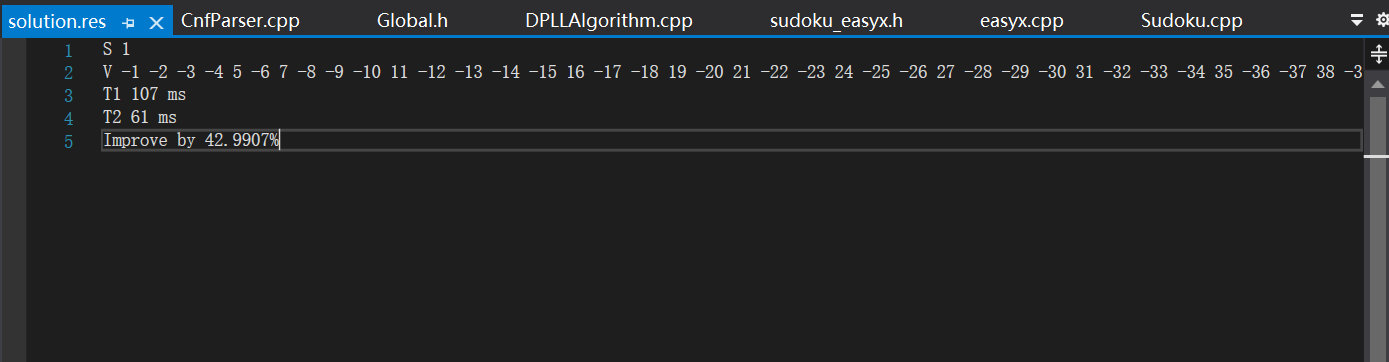


图4中型基准满足算例sud00012.cnf

1. 数独模块,包括数独生成、归约、求解(Sudoku)模块
2. 模块说明：数独终盘的生成；主要包括生成9\*9的数独，第一步，随机生成15个数填入数独终盘，其中每生成一个数，都要根据数独的规则都要判断其填入的合法性；第二步，将所有的约束条件转换为CNF写入文件，并将已经填入的数字作为单子句也写入CNF文件；第三步，调用DPLL算法，求解CNF文件并转换为数独游戏生成数独终盘。
3. 测试方法：先随机生成棋盘，接着用DPLL算法生成数独终盘，最后用挖洞发生成一个数独游戏；
4. 运行结果，截图如图5生成随机数独棋盘所示。

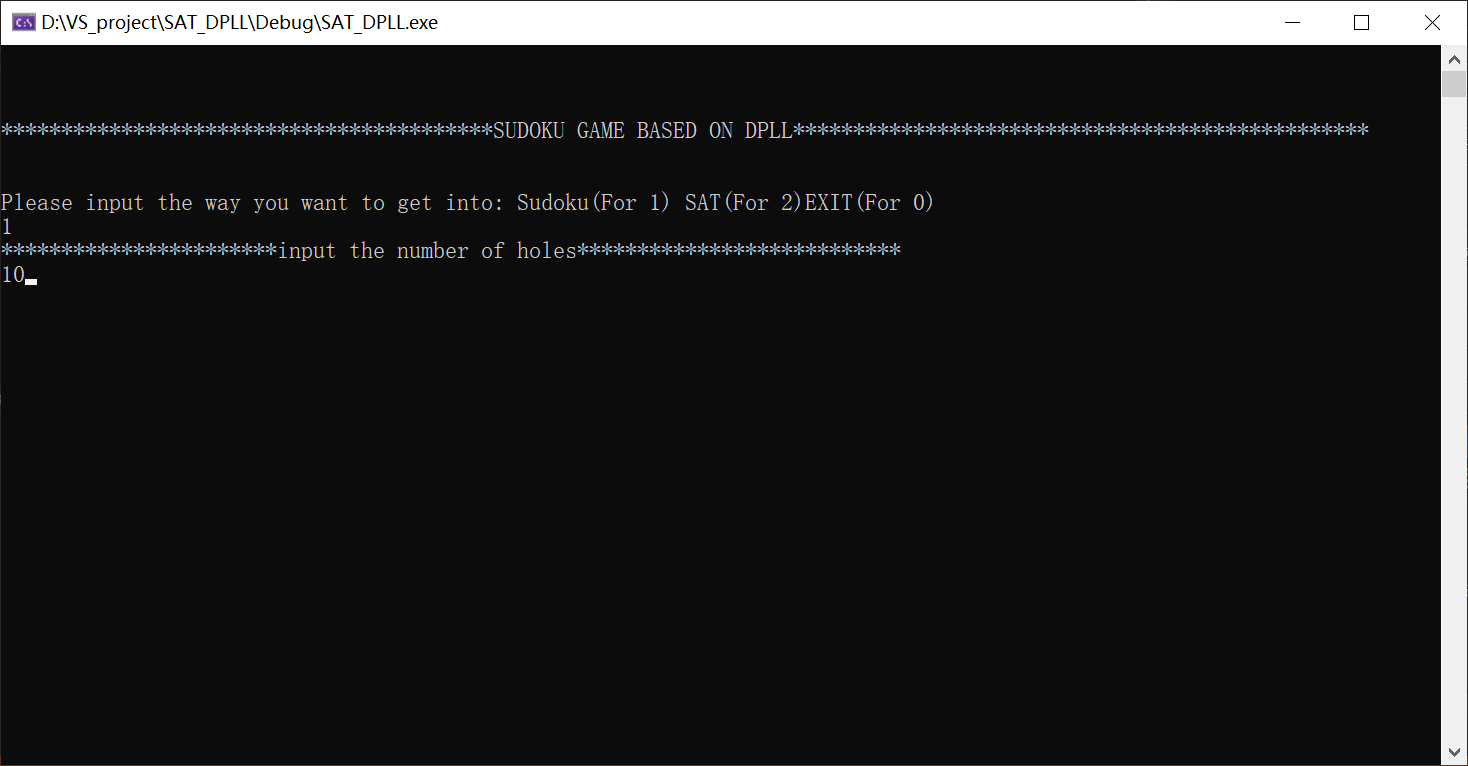


图5-1生成随机数独棋-输入内容



图5-2生成随机数独棋盘-游戏初盘

图 5-3生成数独棋盘-显示答案



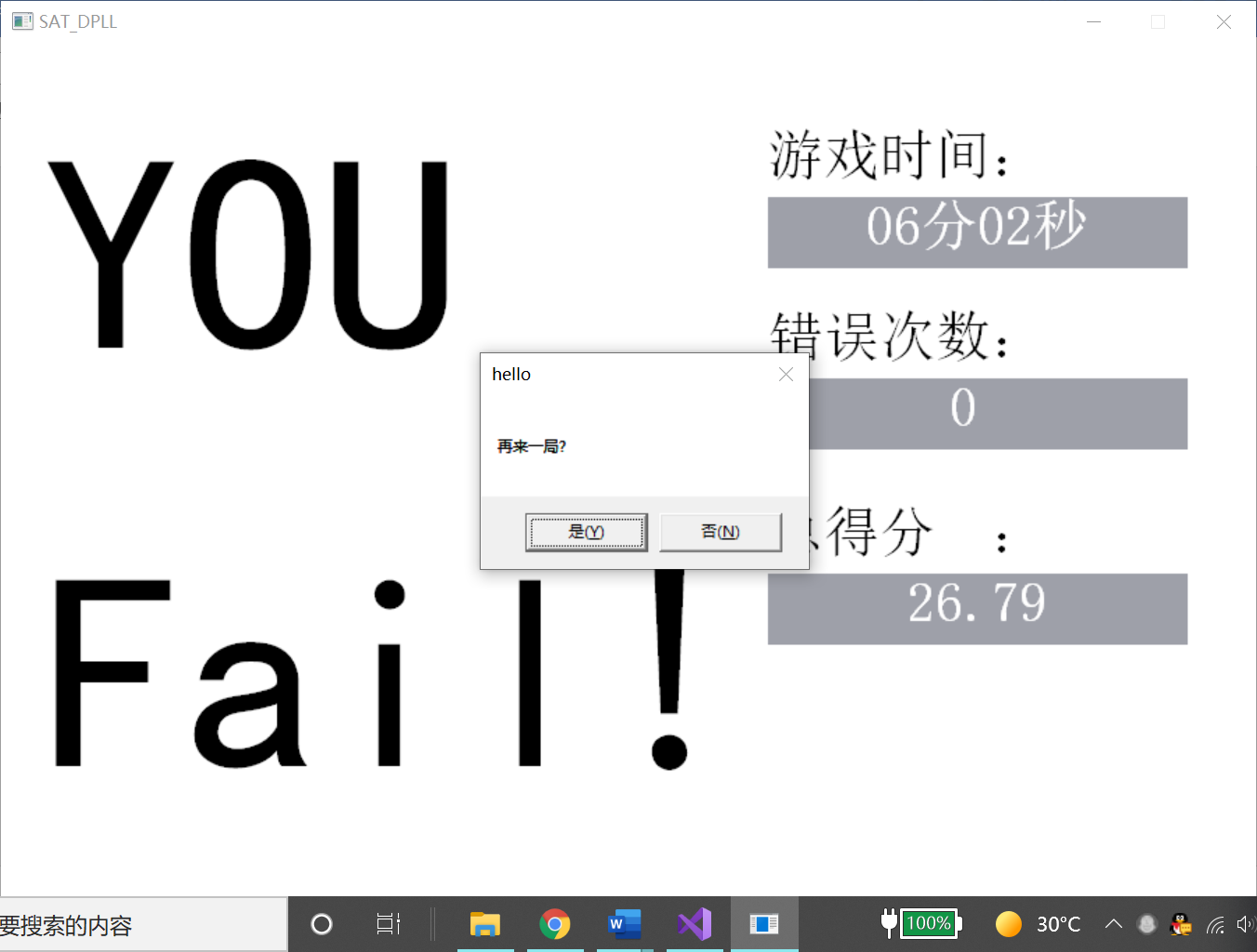
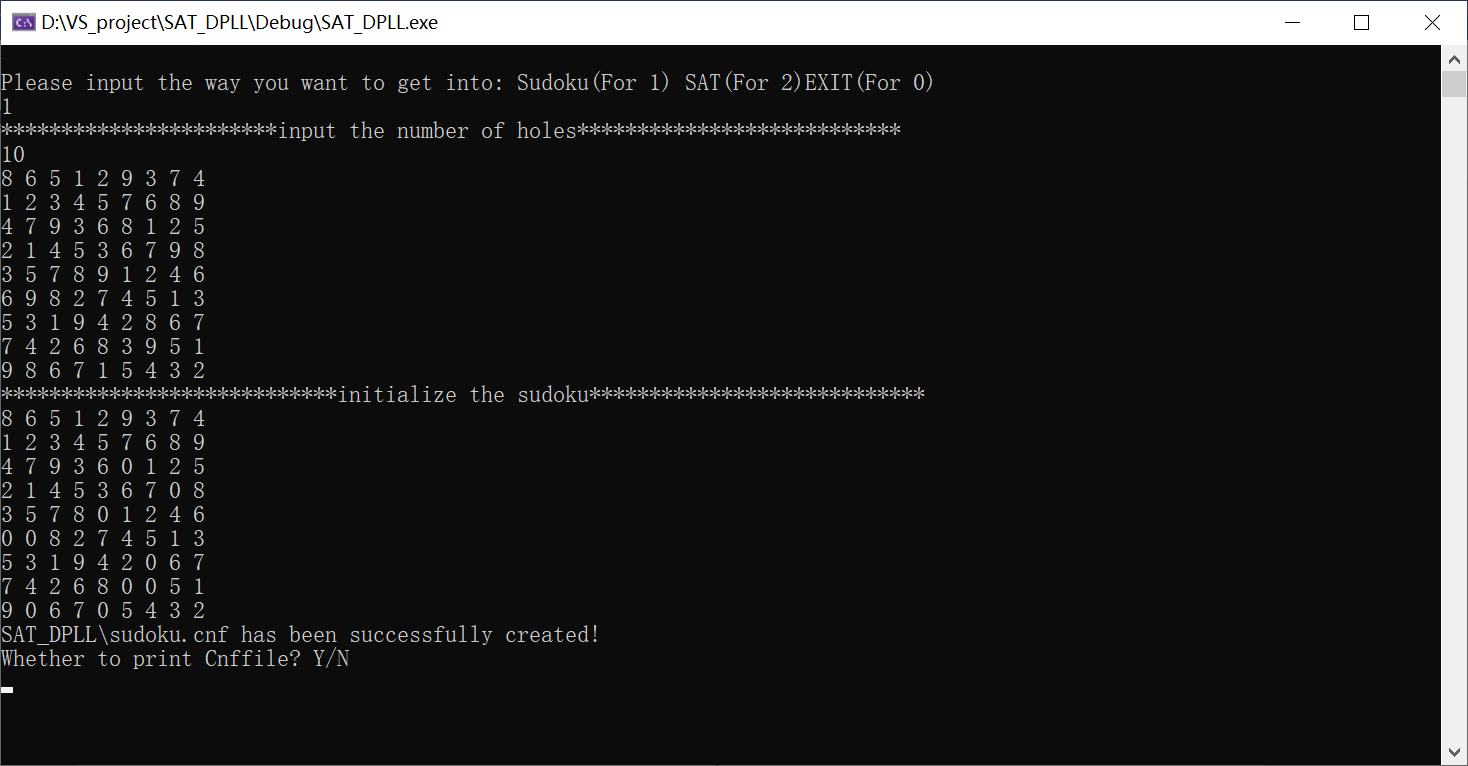
图 5-4生成数独棋盘-退出选项

图 5-5生成数独棋盘-退出界面



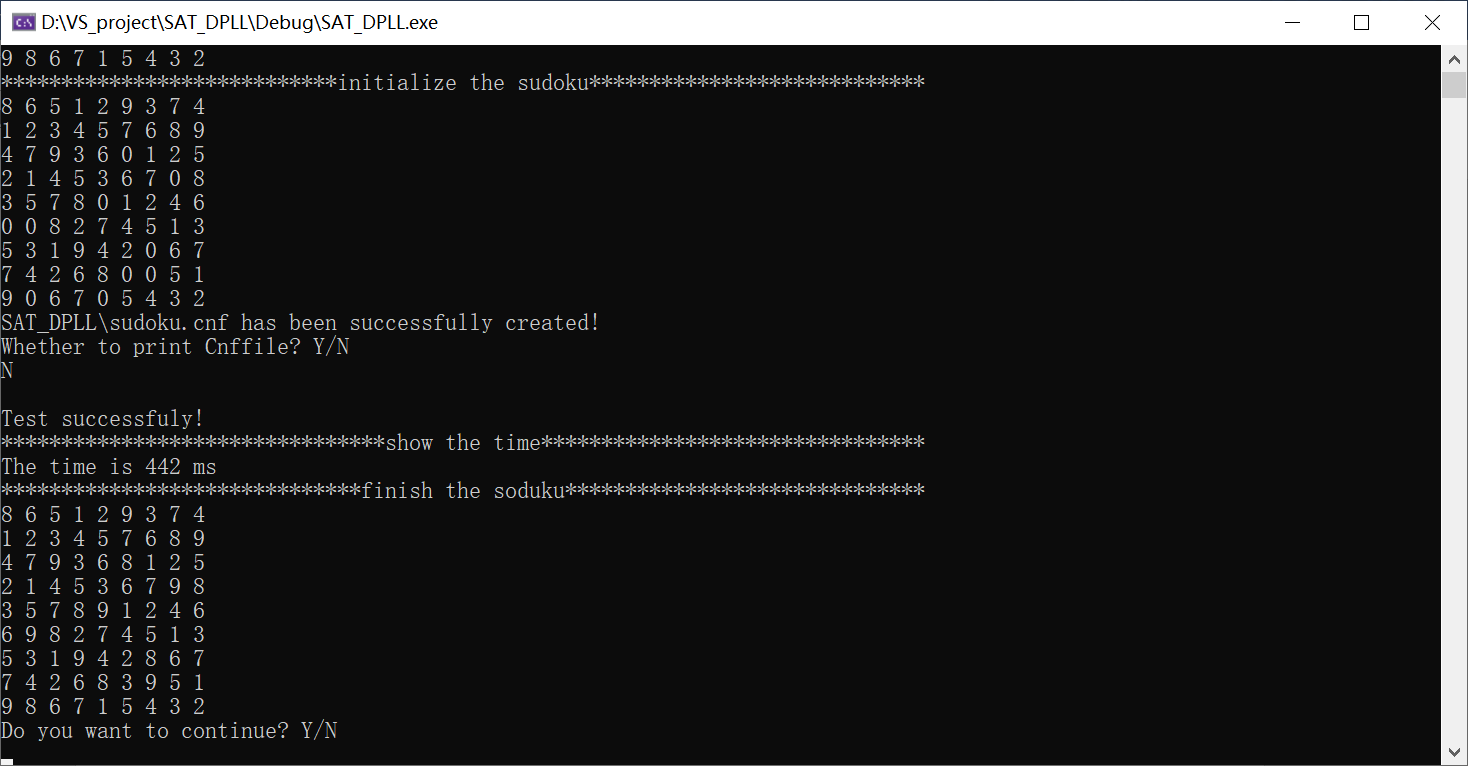
图 5-6生成数独棋盘-弹回控制窗

图 5-7生成数独棋盘-基于DPLL填出正确答案及时间

**4.3 程序优化**

1. 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。
2. 主要优化方向，包括代码指令减少，算法改进；

算法改进主要是决策方式的改变，即将最短正字句改变为在字句中出现频率最高的变元；

优化前后运行时间以及优化率如表6所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试算例 | 运行结果 | 优化前运行时间/ms | 优化后运行时间/ms | 优化率 |
|  |
| sat-20 | SAT | 3 | 2 | 33.33% |  |
| problem2-50 | SAT | 28 | 1 | 96.43% |  |
| problem3-100 | SAT | 316 | 85 | 73.10% |  |
| problem6-50 | SAT | 10 | 4 | 60.00% |  |
| problem8-50 | SAT | 14 | 17 | -21.43% |  |
| problem1-20 | SAT | 2 | 1 | 50.00% |  |
| problem9-100 | SAT | 168 | 26 | 84.52% |  |
| problem11-100 | SAT | 159 | 19 | 88.05% |  |
| tst\_v10\_c100 | UNSAT | 1 | 0 | / |  |
| u-5cnf\_3500\_3500\_30f1.shuffled-30 | UNSAT | 335 | 360 | -7.46% |  |
| u-problem7-50 | UNSAT | 131 | 19 | 85.50% |  |
| problem5-200 | SAT | 1784 | 142 | 92.04% |  |
| sud00012 | SAT | 198 | 138 | 30.30% |  |
| sud00082 | SAT | 421 | 49 | 88.36% |  |
| Mtst\_v200\_c210 | SAT | 65 | 13 | 80% |  |
| 7cnf20\_90000\_90000\_7.shuffled-20 | SAT | 1067 | 388 | 63.64% |  |
| e-par32-3.shuffled-3176 | SAT | 4191 | 519 | 87.62% |  |

表6优化前后运行时间以及优化率

1. 优化分析，在（2）表6中可以观察到，优化之后算例反而出现负优化，在分析算例后发现。对于变现出现频率相等或者说相差不大的算例，以变量出现频率作为决策方式，并不能达到预期结果，这是因为若各个变元出现频率相近，则无论如何选择最终运行时间都很相似，但优化策略额外的对CNF文件中各个变元遍历并统计，此过程也会耗费一定时间，又因为DPLL算法为递归算法，故对算法整体影响较大，导致负优化。因此，优化还是需要针对实际算例进行。

**5 总结与展望**

**5.1 全文总结**

对自己的工作做个总结，主要工作如下：

（1）利用二维链表（头结点纵向连接）实现了将CNF文件转换为可以求解的数据结构；

（2）基于二维链表的数据结构实现了最基础的SAT问题求解，其中包括判断子句是否为空、判断子句是否为单子句、增添单子句、回溯、删除变元、删除子句以及最后完整的DPLL等操作；

（3）将SAT问题求解应用于数独游戏的求解，实现数独游戏的生成，并利用Easyx制作图形界面，数独游戏可玩性强，项目完成度高。

**5.1 工作展望**

关于本次项目，有以下期望：

1. DPLL算法优化率有待提高，正如优化分析中所提到的那样，对于各个变元出现频率相近的样例，盲目的按照“频率优先”选择变元会适得其反，解决方案有两个，一是改变选择策略；二是在DPLL算法前统计各个变元的出现频率，并据此对DPLL的几种策略进行评估和选择，从而选出最优策略。目前认为方案二比较可行。
2. 对于数据结构的选择有待完善，本次使用的链表结构存储各变元并进行相关操作。数据结构的选择的简单导致后续设计相应操作时较为复杂，尤其是回溯，本次采用操作前复制副本的方式实现回溯，造成了大量内存的占用。期望实现在本链表中完成删除、恢复等操作，解决方案为将删除、恢复操作以标记的形式完成，而不是实际意义上的删除。

另外，在今后的学习研究中，我也有以下期望：

1. 学习中尽量多运用数据结构中所学的内容，在遇到一些问题时，不着急编写代码，多构思，多琢磨找出最优的数据结构去存储和管理它们；
2. 多实践，将数据结构带到即将要学的专业课程中去，学以致用，不断巩固提升所学内容；
3. 平时多学习一些算法，及它们的应用，在学习优秀算法的过程中不断提升自己的代码能力，并学会如何将优秀算法思想迁移到实际问题中。

# 6体会

这次数据结构课程设计对我来说，很有挑战性，但也意义重大。此次课程设计是我转入计算机学院以来第一个较大项目，因此，我是怀着学习的态度，和打赢第一场战斗的决心一步一步将其完成、完善的。从刚开始的文件读取，文件保存，到DPLL算法的理解，各个基本操作的实现，再到最终基本完成，以及借助Easyx对其可玩性的完善。这个漫长又充满挑战的过程教会了我很多东西，也极大增强了我的信心。当然这其中少不了王振江学长、袁凌老师的指点，正是他们的帮助让我能打赢这关键的第一仗。

下面我就简单说一下这两个星期的心路历程吧！

读懂算法原理后我就开始着手编写代码，依照着实验要求中首先将每个操作函数逐一实现，在编写各个小函数的过程中也渐渐把思路捋清了，直到DPLL算法实现的时候，在“回溯”这一步上卡了许久，如何实现在原结构上恢复操作前的数据，又如何实现有效的返回呢？思来想去，也想到一些办法，比如在上一部分提到的以做标记来代替删除等操作。确实是一种即为先进的做法，但当时由于着急看到结果，选择了一种现在认为较为愚钝的办法，就是直接复制一个副本来实现有效回溯。突破这个关卡之后，我较为顺利地完成下面的文件读写操作，并在艰难地DEBUG后成功运行处结果。此时已经过去了一星期左右。

接下来，因为剩下的时间并不多，我仅测试了基准算例后便一头扎进数独游戏的实现。这也为最后两天的纠结奠定了基础……

由于数独游戏比较典型，在阅读网络上各种资料以后，我便成功创建随机的未挖孔的数组盘。接下来是如何挖孔的问题，既然要确定生成的数独初盘有唯一解，那么在挖孔的时候就不能随意乱挖，而是要基于一定的算法思想。通过查阅各种资料，我明晰了其中的道理，就是每挖一个孔，便将除这个孔原有数字以外的所有数字均填入并将其转化为CNF文件代入第一部分中求解，若出现解为SAT则不唯一，否则，解唯一。因此，需首先实现残局信息转为CNF文件的操作，于是按照实验要求中的提示，按部就班完成了CNF的转换工作，之后，雏形就基本建立完成。可以实现按照输入数字进行挖洞，并给出最后结果。此时已经只剩下两天，我开始回头验证第一部分的其他算例，然而却意识到了一个很大的问题。

我发现对于中等偏大的算例，我的程序经常运行崩溃，思考了很久，发现是在代码中new的对象没有在适当的地方释放。修修补补后，并没有很大改观，最后意识到问题出现在DPLL算法的回溯中，由于我的回溯是利用复制副本的方式完成的，而由于DPLL的实现依赖于递归算法，在这个过程中，程序的每一层都会创建一个副本，因此在遇到较大的算例时，所需空间也会急剧增大。因此，我打算修改回溯的操作函数，在上文已提到，采用标记代替删除从而解决如何恢复的问题，进而避免了用复制代替恢复的做法。但由于数据结构也会随之改变，代码需要根本上的改动，于是就放弃了这种做法，继续修修补补，尽可能的在运行过程中释放堆区内存。修改之后，通过不断的调试，一步步的打印运行结果，一遍遍的调试，过程真的感觉十分艰难，当身边同学一个都完成时，自己颇为焦虑，好在终于能运行稍大的算例，至此告一段落。

最终提交的前一天，我发现我的同学们都用上了华丽界面，甚至他们的数独真的算是一个游戏，可以很流畅的人机互动，我很是好奇，请教过后，我赌上最后一天自学了Easyx，阅读了很多教程，终于赶在晚上验收前，讲数独界面嵌入到原有代码中，这是我效率最高，但也最痛苦的一天。那天晚上验收很顺利，月亮也很美~

这个课设总算是踉踉跄跄的画上一个较为完美的句号，真的是获益匪浅。当时着手操作时感觉好难好难，今天在这里写报告总结，不会再感到那么的难，不仅仅完成一次学习任务，对程序的设计方法，和刚学完的数据结构，都有了比较深入的认识和理解，同时，正如我在刚开始所说，这是我来计科的第一个项目，这次的成功给了我极大的信心能让我在以后走下去。最后，非常感谢和蔼的袁凌老师，给我们几位转专业同学的照顾和鼓励，非常感谢王振江学长在我痛苦万分的时候跟我一起DEBUG，也非常感谢我的小伙伴们，他们非常优秀，跟他们的交流让我少走很多弯路。数据结构课设至此而终，但我会永远记得，永远怀念~

# 7附录

**7.1 主函数**

#include "Global.h"

int main() {

Start();

int choice = 0;

int value = 0;

double T1 = 0.0;

double T2 = 0.0;

char judge\_continue;

cin>>choice;

while (choice){

if(choice == 1) {

string filename = createSudokuToFile();

sudoku\_easyx();

cout <<filename<< " has been successfully created!" << endl;

int VARNUM;

HeadNode\* LIST = CreateClause(VARNUM,filename);

conse \* SudoResult= new conse [VARNUM]; //conse SudoResult[VARNUM];//记录最终的真假值\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*添加new

clock\_t StartTime,EndTime;

StartTime = clock();

int value = SudoDPLL(LIST,SudoResult,VARNUM);

EndTime = clock();

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*show the time\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout<<"The time is "<<(double)(EndTime-StartTime)/CLOCKS\_PER\_SEC\*1000.0<<" ms\n";

if (value == 1)

{

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*finish the soduku\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

SudokuShow(SudoResult, VARNUM);

delete[] SudoResult;//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*添加delete

cout << "Do you want to continue? Y/N" << endl;

cin >> judge\_continue;

if (judge\_continue == 'Y') {

Start();

cin >> choice;

continue;

}

// else exit(0);

}

else {

cout << "there is no anwser.";

cout << "Do you want to continue? Y/N" << endl;

cin >> judge\_continue;

if (judge\_continue == 'Y') {

Start();

cin >> choice;

continue;

}

// else exit(0);

}

choice = 0;

exit(0);

}

else if (choice == 2) {//SAT

int VARNUM;

string filename;// = "sat-20.cnf";

cout << "Please input the file name" << endl;

HeadNode\* LIST = nullptr;

do {

cin >> filename;

filename = filename + ".cnf";

LIST = CreateClause(VARNUM, filename);

if (LIST == nullptr)cout << "please input the existed filename!" << endl;

} while (LIST==nullptr);

consequence\* result=new consequence[VARNUM];//consequence result[VARNUM];//记录最终的真假值\*\*\*\*\*\*\*\*\*\*添加new

clock\_t StartTime,EndTime;

//cout<<"Result: \n";

//StartTime = clock();

//int value = DPLL(LIST,result);

//EndTime = clock();

//if(value) {

// cout << "S " << TRUE << endl;

// show(result, VARNUM);//输出解

//}

//else {

// cout << "S " << NoAnwser << endl;

// cout<<"V "<<endl;

//}

//cout<<"T "<<(double)(EndTime-StartTime)/CLOCKS\_PER\_SEC\*1000.0<<" ms\n";

//写到文件

string suffix = ".res";

string name = "solution";

string Outputfile = name + suffix;

ofstream fos(Outputfile);

if(!fos.is\_open()) {

cout<<"Can not open a new file.\n";

exit;

}

//StartTime = clock();

int choose\_dpll =0;//决策下行函数

cout << "Whether to optimize?Y(For 1)/N(For 2)" << endl;

cin >> choose\_dpll;

while (choose\_dpll != 1 && choose\_dpll != 2) {

cout << "input the correct number!" << endl;

cin >> choose\_dpll;

}

if (choose\_dpll == 1) {//优化

StartTime = clock();

value = DPLL1(LIST, result);//

EndTime = clock();//计算dpll1时间

T1 = (double)(EndTime - StartTime);

HeadNode\* LIST = CreateClause(VARNUM, filename);

// cout << EndTime<<StartTime<<T1 << endl;

StartTime = clock();//////////////////////////

DPLL2(LIST, result);/////////////////////

EndTime = clock();//计算dpll2时间//////////////////////

T2 = (double)(EndTime - StartTime);/////////////

// cout << EndTime << StartTime << T2 << endl;

// fos << "T2 " << T2 / CLOCKS\_PER\_SEC \* 1000.0 << " ms\n";//

// fos << "T1 " << T1 / CLOCKS\_PER\_SEC \* 1000.0 << " ms\n";

cout << "Congratulations!" << endl;

}

else if(choose\_dpll==2) {

StartTime = clock();

value = DPLL1(LIST, result);

EndTime = clock();

cout << "Congratulations!" << endl;

T2= (double)(EndTime - StartTime);

// cout << EndTime << StartTime << T2 << endl;

}

//结果

if(value) {

fos << "S " << TRUE << endl;

fos<<"V ";

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

if (result[i].value == TRUE)

fos<<i+1<<" ";

else if(result[i].value == FALSE)

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

else

fos<<(i+1)<<" ";//剩下一堆可true可false，就索性输出true

}

fos<<endl;

}

else {

fos << "S " << NoAnwser << endl;

fos<<"V ";

fos<<endl;

}

//时间

if (choose\_dpll == 1) {

if (T1 > T2) {

fos << "T1 " << T1 / CLOCKS\_PER\_SEC \* 1000.0 << " ms\n";

fos << "T2 " << T2 / CLOCKS\_PER\_SEC \* 1000.0 << " ms\n";

if(T1&&T2)

fos << "Improve by " << (T1 - T2) / T1 \* 100.0<<"%";

}

else {

fos << "T1 " << T2 / CLOCKS\_PER\_SEC \* 1000.0 << " ms\n";

fos << "T2 " << T1 / CLOCKS\_PER\_SEC \* 1000.0 << " ms\n";

if (T1 && T2)

fos << "Improve by " << (T2 - T1) / T2 \* 100.0<<"%";

}

}

else {

fos << "T " << T2 / CLOCKS\_PER\_SEC \* 1000.0 << " ms\n";

}

fos.close();

cout << "Do you want to continue? Y/N" << endl;

cin >> judge\_continue;

if (judge\_continue == 'Y') {

Start();

cin >> choice;

continue;

}

delete[] result;//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*添加delete

choice = 0;

exit(0);

}

else { //其他值不理睬

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

cin >> choice;

cout << endl;

// exit;

}

}

}

**7.2 各操作函数**

## 7.2.1 CNF解析及相关操作

//

// Created by PC on 2019/2/17.

#include "Global.h"

HeadNode\* CreateClause(int& VARNUM, string& filename/\*, int unique\_flag\*/) {

//FileOpen

string HFilePath = R"(D:\VS\_project\SAT\_DPLL\)";

string path = HFilePath + filename;

ifstream fis(path);

if (!fis) {

cout << "File can not open." << endl;

return nullptr;

// exit;

}

char ch;

char buf[100];

fis >> ch;

while (ch != 'p') {

fis.getline(buf, 100);

fis >> ch;

}

string cnf;

int VarNum, ClauseNum;

fis >> cnf >> VarNum >> ClauseNum;

fis.get();

/\*and write into 2 dimensional link lists\*/

//loading the initial node

HeadNode\* HEAD = new HeadNode;

HeadNode\* headRear = HEAD;

HeadNode\* END = new HeadNode;

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

//load on the data lists

int temp;

fis >> temp;

//load the first data node

DataNode\* front = new DataNode;

front->data = temp;

headRear->right = front;

headRear->Num++;

//the >2th data loading

fis >> temp;

while (temp != 0) {

DataNode\* rear = new DataNode;

front->next = rear;

rear->data = temp;

front = front->next;

headRear->Num++;

fis >> temp;

}

front->next = nullptr;

fis.get();//换行符

HeadNode\* tp = new HeadNode;

headRear->down = tp;

END = headRear;

headRear = headRear->down;

}

END->down = nullptr;

//output link lists

HeadNode\* Phead = HEAD;

DataNode\* front;

//cout << "验证唯一性步骤(For1)；选择输出步骤(For2)" << endl;

/\*if (unique\_flag == 1) {

while (Phead != nullptr) {

front = Phead->right;

while (front != nullptr) {

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

front = front->next;

}

cout << endl;

Phead = Phead->down;

}

}\*/

// if(unique\_flag == 2) {

// cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CnfParser\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout << "Whether to print Cnffile? Y/N" << endl;//选择是否输出cnf文件：Y输出，N不输出

char judge\_cnfshow;

cin >> judge\_cnfshow;

while (judge\_cnfshow != 'N')

{

if (judge\_cnfshow == 'Y') {

while (Phead != nullptr) {

front = Phead->right;

while (front != nullptr) {

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

front = front->next;

}

cout << endl;

Phead = Phead->down;

}

}

judge\_cnfshow = 'N';

}

cout << "\nTest successfuly!" << endl;

// }

VARNUM = VarNum;

return HEAD;

}

## 7.2.2 DPLL算法及相关操作

//

// Created by PC on 2019/2/18.

//

#include "Global.h"

status IsEmptyClause(HeadNode\* LIST) {

HeadNode\* PHead = LIST;

while (PHead != nullptr) {

if(PHead->Num == 0)

return TRUE;

PHead = PHead->down;

}

return FALSE;

}

HeadNode\* IsSingleClause(HeadNode\* Pfind) {

while (Pfind != nullptr ) {

if(Pfind->Num == 1)

return Pfind;

Pfind = Pfind->down;

}

return nullptr;

}

HeadNode\* Duplication(HeadNode\* LIST) { //此处检验传参正常，开始检查复制有无逻辑错误

HeadNode\* SrcHead = LIST;

HeadNode\* ReHead = new HeadNode;//新链表的头节点

ReHead->Num = SrcHead->Num;//复制第一个头节点

HeadNode\* Phead = ReHead;//Phead创建头节点

DataNode \*ReData = new DataNode;//新链表的数据节点

DataNode \*FirstSrcData = SrcHead->right;//用于创建第一行的第一个数据节点

ReData->data = FirstSrcData->data;//新链表的第一个数据节点的数值

Phead->right = ReData;

for (FirstSrcData = FirstSrcData->next;FirstSrcData != nullptr; FirstSrcData = FirstSrcData->next) {//第一行链表复制完成

DataNode \*NewDataNode = new DataNode;

NewDataNode->data = FirstSrcData->data;

ReData->next = NewDataNode;

ReData = ReData->next;

}

//此下行数节点的复制 >=2th

for(SrcHead = SrcHead->down; SrcHead != nullptr ; SrcHead = SrcHead->down) {

HeadNode\* NewHead = new HeadNode;

DataNode\* NewData = new DataNode;

NewHead->Num = SrcHead->Num;

Phead->down = NewHead;

Phead = Phead->down;

DataNode\* SrcData = SrcHead->right;

NewData->data = SrcData->data;

Phead->right = NewData;//第一个数据节点

for (SrcData = SrcData->next;SrcData != nullptr; SrcData = SrcData->next) {//此行剩下的数据节点

DataNode\* node = new DataNode;

node->data = SrcData->data;

NewData->next = node;

NewData = NewData->next;

}

NewData->next = nullptr;

}

Phead->down = nullptr;

return ReHead;

}

HeadNode\* ADDSingleClause(HeadNode\* LIST,int Var) { //所加的单子句位于链表的头

HeadNode\* AddHead = new HeadNode;

DataNode\* AddData = new DataNode;

AddData->data = Var;

AddData->next = nullptr;

AddHead->right = AddData;

AddHead->Num = 1;

AddHead->down = LIST;

LIST = AddHead;

return LIST;

}

void DeleteDataNode(int temp,HeadNode \*&LIST) {

DataNode\* p\_rear = nullptr;

// HeadNode\* p\_pHeadNode = nullptr;

for (HeadNode\* pHeadNode = LIST; pHeadNode != nullptr ; pHeadNode = pHeadNode->down)

for (DataNode \*rear = pHeadNode->right; rear != nullptr ; rear = rear->next) {

if (rear->data == temp) //相等删除整行

{

// p\_pHeadNode= pHeadNode->down;

DeleteHeadNode(pHeadNode, LIST);

// pHeadNode->down = p\_pHeadNode;

break;

}

else if (abs(rear->data) == abs(temp)) { //仅仅是绝对值相等铲除该节点

if(rear == pHeadNode->right) { //头节点删除

pHeadNode->right = rear->next;

pHeadNode->Num--;

}

else{ //删除普通节点

for (DataNode\* front = pHeadNode->right; front != nullptr; front= front->next)

if(front->next == rear) {

front->next = rear->next;

delete rear;

rear = front;

pHeadNode->Num--;

}

}

}

}

}

void DeleteHeadNode(HeadNode \*Clause,HeadNode \*&LIST) {

if (!Clause) return;

if (Clause == LIST) {

LIST = Clause->down;

FreeHeadNode(Clause);//释放整行内存

}

else {

for (HeadNode \*front = LIST; front != nullptr; front = front->down)

if (front->down == Clause) {

front->down = Clause->down;

FreeHeadNode(Clause);//释放整行内存

}

}

}

void FreeHeadNode(HeadNode\* Clause) {

DataNode\* p1\_DataNode = Clause->right;

DataNode\* p2\_DataNode = Clause->right;

Clause->right = nullptr;

//Clause->down = nullptr;

while (p2\_DataNode != nullptr) {

p1\_DataNode = p2\_DataNode->next;

delete p2\_DataNode;

p2\_DataNode = p1\_DataNode;

}

//delete Clause;

}

void show(struct consequence \*result,int VarNum) {

cout<<"V ";

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

if (result[i].value == TRUE)

cout<<i+1<<" ";

else if(result[i].value == FALSE)

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

else

cout<<(i+1)<<" ";//剩下一堆可true可false，就索性输出true

}

cout<<endl;

}

int Choose\_max(HeadNode\* LIST) {

HeadNode\* p\_head = LIST;

DataNode\* p\_data = LIST->right;

// Count\* Result\_fre = new Count;

Count Result\_fre[CHOOSE\_SOLUTION];

Count\* p\_count = Result\_fre;

int flag\_count = 0;

int max\_count = 0;

int max\_num = 0;

while (p\_head != nullptr) {

p\_data = p\_head->right;

while (p\_data != nullptr) {

p\_count = Result\_fre;

flag\_count = 0;

while (p\_count->num) {

if (p\_count->num == abs(p\_data->data)) {

(p\_count->count)++;

flag\_count = 1;

}

p\_count++;

}

if (flag\_count == 0) {

p\_count->num = abs(p\_data->data);

(p\_count->count)++;

}

p\_data = p\_data->next;

}

p\_head = p\_head->down;

}

p\_count = Result\_fre;

while (p\_count->num) {

if (p\_count->count > max\_count) {

max\_num = p\_count->num;

}

p\_count++;

}

return max\_num;

}

status DPLL2(HeadNode \*LIST,consequence \*result) {

//单子句规则

//int choose\_dpll=0;

//int Var = 0;

HeadNode\* Pfind = LIST;

HeadNode\* SingleClause = IsSingleClause(Pfind);

while (SingleClause != nullptr) {

SingleClause->right->data > 0 ? result[abs(SingleClause->right->data)-1].value = TRUE : result[abs(SingleClause->right->data)-1].value = FALSE;

int temp = SingleClause->right->data;

DeleteHeadNode(SingleClause,LIST);//删除单子句这一行

DeleteDataNode(temp,LIST);//删除相等或相反数的节点

if(!LIST) return TRUE;

else if(IsEmptyClause(LIST)) return FALSE;

Pfind = LIST;

SingleClause = IsSingleClause(Pfind);//回到头节点继续进行检测是否有单子句

}

//分裂策略

// if (choose\_dpll == 1) {

// int Var = LIST->right->data;//选取变元

// }

// else if (choose\_dpll == 2) {

int Var = Choose\_max(LIST);

// }

HeadNode\* replica = Duplication(LIST);//存放LIST的副本replica

HeadNode \*temp1 = ADDSingleClause(LIST,Var);//装载变元成为单子句

if(DPLL2(temp1,result)) return TRUE;

else {

HeadNode \*temp2 = ADDSingleClause(replica,-Var);

if (DPLL1(temp2, result)) // return DPLL(temp2,result);

{

//delete replica;

return TRUE;

}

else {

//delete replica;

return FALSE;

}

}

}

status DPLL1(HeadNode\* LIST, consequence\* result) {//拉一点点

//单子句规则

//int choose\_dpll = 0;

//int Var = 0;

HeadNode\* Pfind = LIST;

HeadNode\* SingleClause = IsSingleClause(Pfind);

while (SingleClause != nullptr) {

SingleClause->right->data > 0 ? result[abs(SingleClause->right->data) - 1].value = TRUE : result[abs(SingleClause->right->data) - 1].value = FALSE;

int temp = SingleClause->right->data;

DeleteHeadNode(SingleClause, LIST);//删除单子句这一行

DeleteDataNode(temp, LIST);//删除相等或相反数的节点

if (!LIST) return TRUE;

else if (IsEmptyClause(LIST)) return FALSE;

Pfind = LIST;

SingleClause = IsSingleClause(Pfind);//回到头节点继续进行检测是否有单子句

}

//分裂策略

// if (choose\_dpll == 1) {

int Var = LIST->right->data;//选取变元

// }

// else if (choose\_dpll == 2) {

//int Var = Choose\_max(LIST);

// }

HeadNode\* replica = Duplication(LIST);//存放LIST的副本replica

HeadNode\* temp1 = ADDSingleClause(LIST, Var);//装载变元成为单子句

if (DPLL1(temp1, result)) return TRUE;

else {

HeadNode\* temp2 = ADDSingleClause(replica, -Var);

if (DPLL1(temp2, result)) // return DPLL(temp2,result);

{

//delete replica;

return TRUE;

}

else {

//delete replica;

return FALSE;

}

}

}

## 7.2.3 数独游戏及相关操作

#include "Global.h"

#define CORRECT 0

#define WRONG -1

static int T = 0;

int Digit(int a[][COL], int i, int j) {//递归填充数独元素

if (i < ROW && j < COL) {

int x,y,k;

int check[COL+1]={CORRECT};//用于排除已经使用过的的数字

for(x = 0 ; x < i ; x++)

check[a[x][j]] = WRONG;//列已使用的数字置为WRONG

for(x = 0 ; x < j ; x++)

check[a[i][x]] = WRONG;//行使用过的数字置为WRONG

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

if(x == i)

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

check[a[x][y]] = WRONG;

else

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

check[a[x][y]] = WRONG;

}

int flag = 0;

for(k = 1; k <= COL && flag == 0 ; k++){//从check数组中查找安全的数字

if(check[k] == CORRECT){

flag = 1;

a[i][j] = k;

if(j == COL-1 && i != ROW-1){

if(Digit(a,i+1,0) == CORRECT) return CORRECT;

else flag = 0;

}

else if(j != COL-1){

if(Digit(a,i,j+1) == CORRECT) return CORRECT;

else flag = 0;

}

}

}

if( flag == 0 ) {

a[i][j] = 0;

return WRONG;

}

}

return CORRECT;

}

void randomFirstRow(int a0[], int n) {//随机生成第一行

int i,j;

srand((unsigned)time(nullptr));

for( i = 0 ; i < n ; i++){

a0[i] = rand()%9 + 1;

j = 0 ;

while(j < i){

if(a0[i] == a0[j]){

a0[i] = rand()%9 + 1;

j = 0;

}

else j++;

}

}

}

void createSudoku(int a[][COL]){ //生成数独

randomFirstRow(a[0],COL);//随机生成第一行

Digit(a,1,0);//递归生成后i行

}

void createStartinggrid(const int a[][COL], int b[][COL], int numDigits) {//随机生成初盘

int i,j,k;

srand((unsigned)time(nullptr));

for( i = 0; i < ROW ; i ++)

for( j = 0; j < COL ; j++)

b[i][j] = a[i][j];

int\*\* c = new int\* [numDigits]; //int c[numDigits][2];\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*添加new

for (int i = 0; i < numDigits; i++) c[i] = new int[2];//int c[numDigits][2];\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*添加new

int m,flag = 0;

//int test\_num = 0;

for( i = 0; i < numDigits ; i++) {

j = rand()%9;

k = rand()%9;

flag = 0;

for(m = 0; m < i ; m++)

if( j == c[m][0] && k == c[m][1])

flag = 1;

//if (flag == 0) {//是否唯一解

// for (test\_num = 1; test\_num <= 9; test\_num++) {

// if (test\_num != a[j][k]) {

// b[j][k] = test\_num;

// string filename\_temp = ToCnf(b, i);

// int VARNUM\_temp;

// HeadNode\* LIST\_temp = CreateClause(VARNUM\_temp, filename\_temp);

// conse\* SudoResult\_temp = new conse[VARNUM\_temp];

// int value = SudoDPLL(LIST\_temp, SudoResult\_temp, VARNUM\_temp);

// if (value == 1)flag = 1;

// }

// }

//}

if(flag == 0){

b[j][k] = 0;

c[i][0] = j;

c[i][1] = k;

}

else

i--;

}

for (int i = 0; i < numDigits; i++) delete[]c[i];//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*添加delete

delete[]c;//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*添加delete

}

void print(const int a[][COL]){//打印数独数组

int i,j;

for( i = 0 ; i < ROW ; i++){

for( j = 0 ; j < COL ; j++)

printf("%d ", a[i][j]);

cout<<endl;

}

}

string ToCnf(int a[][COL],int holes) {

ofstream in ("sudoku.cnf");//定义输入文件

if(!in.is\_open())

cout<<"can't open!\n";

in<<"p"<<" "<<"cnf"<<" "<<729<<" "<<8829+81-holes<<" "<<endl;

//single clause

for (int x = 0; x < ROW; ++x) {

for (int y = 0; y < COL; ++y)

if(a[x][y] != 0)

in<<(x+1)\*100 + (y+1)\*10 + a[x][y]<<" "<<0<<endl;

}

//entry

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

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

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

in << x \* 100 + y \* 10 + z << " ";

in<<0;

in<<endl;

}

}

//row

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

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

for (int x = 1; x <= 8; ++x)

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

in<<0 - (x\*100 + y\*10 + z)<<" "

<<0 - (i\*100 + y\*10 + z)<<" "<<0<<endl;

}

//column

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

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

for (int y = 1; y <= 8; ++y)

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

in<<0-(x\*100 + y\*10 + z)<<" "

<<0-(x\*100 + i\*10 + z)<<" "<<0<<endl;

}

//3\*3 sub-grids

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

for (int i = 0; i <=2 ; ++i)

for (int j = 0; j <=2 ; ++j)

for (int x = 1; x <= 3 ; ++x)

for (int y = 1; y <= 3; ++y)

for (int k = y+1; k <= 3; ++k)

in<<0 - ((3\*i+x)\*100 + (3\*j+y)\*10 + z)<<" "

<<0-((3\*i+x)\*100 + (3\*j+k)\*10 + z)<<" "<<0<<endl;

}

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

for (int i = 0; i <= 2; i++)

for (int j = 0; j <= 2; j++)

for (int x = 1; x <= 3; x++)

for (int y = 1; y <= 3; y++)

for (int k = x + 1; k <= 3; k++)

for (int l = 1; l <= 3; l++)

in << 0 - ((3\*i+x)\*100 + (3\*j+y)\*10 + z) << ' '

<< 0 - ((3\*i+k)\*100 + (3\*j+l)\*10 + z) << ' ' << 0 <<endl;

}

in.close();

return string("SAT\_DPLL\\sudoku.cnf");

}

int sudoku[ROW][COL] = { 0 };

int starting\_grid[ROW][COL] = { 0 };

int holes = 15;

string createSudokuToFile() {

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*input the number of holes\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

// int holes = 15;//挖洞个数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*默认为15

cin >> holes;

createSudoku(sudoku);//生成数独终盘

print(sudoku);

createStartinggrid(sudoku,starting\_grid,holes);//生成初盘

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*initialize the sudoku\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

print(starting\_grid);//输出初盘

// system("pause");//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//转化为cnf文件

string filename = ToCnf(starting\_grid,holes);

return filename;

}

status SudoDPLL(HeadNode \*LIST,conse \*result,int VARNUM) {

//单子句规则

HeadNode\* Pfind = LIST;

HeadNode\* SingleClause = IsSingleClause(Pfind);

while (SingleClause != nullptr) {

result[T].num = SingleClause->right->data;

SingleClause->right->data > 0 ? result[T++].value = TRUE : result[T++].value = FALSE;

int temp = SingleClause->right->data;

DeleteHeadNode(SingleClause,LIST);//删除单子句这一行

DeleteDataNode(temp,LIST);//删除相等或相反数的节点

if(!LIST) return TRUE;

else if(IsEmptyClause(LIST)) return FALSE;

Pfind = LIST;

SingleClause = IsSingleClause(Pfind);//回到头节点继续进行检测是否有单子句

}

//分裂策略

int Var = LIST->right->data;//选取变元

//策略二

HeadNode\* replica = Duplication(LIST);//存放LIST的副本replica

HeadNode \*temp1 = ADDSingleClause(LIST,Var);//装载变元成为单子句

if(SudoDPLL(temp1,result,VARNUM)) return TRUE;

else {

HeadNode \*temp2 = ADDSingleClause(replica,-Var);

return SudoDPLL(temp2,result,VARNUM);

}

}

//int Choose\_max(HeadNode\* LIST) {

// HeadNode\* p\_head = LIST;

// DataNode\* p\_data = LIST->right;

// // Count\* Result\_fre = new Count;

// Count Result\_fre[500];

// Count\* p\_count = Result\_fre;

// int flag\_count = 0;

//

// int max\_count = 0;

// int max\_num = 0;

// while (p\_head != nullptr) {

// p\_data = p\_head->right;

// while (p\_data != nullptr) {

// p\_count = Result\_fre;

// while (p\_count->num) {

// if (p\_count->num ==abs(p\_data->data) ) {

// p\_count->count++;

// flag\_count = 1;

// }

// p\_count++;

// }

// if (flag\_count == 0) {

// p\_count->num = abs(p\_data->data);

// (p\_count->count)++;

// }

// p\_data = p\_data->next;

// }

// p\_head = p\_head->down;

// }

// p\_count = Result\_fre;

// while (p\_count != nullptr) {

// if (p\_count->count > max\_count) {

// max\_num = p\_count->num;

// }

// p\_count = p\_count->next;

// }

// return max\_num;

//}

void Uniquejudge(const int a[][COL], const int b[][COL])

{

int i, j ;

for (i = 0; i < 9; i++) {

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

if (a[i][j] != b[i][j]) {

cout << "this result is not unique!" << endl;

return;

}

}

}

return;

}

void SudokuShow(conse \*result,int VARNUM) {

int res[9][9] = {0};

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

if(result[i].value == TRUE) {

int x = (int)( abs(result[i].num) / 100 ) - 1;

int y = (int)( (abs(result[i].num) - (x+1)\*100) / 10 ) - 1;

res[x][y] = abs(result[i].num) - (x+1)\*100 - (y+1)\*10;

}

}

//输出result数组

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

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

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

}

cout<<endl;

}

Uniquejudge(sudoku, res);

}

## 7.2.4 数独游戏界面制作

// 程序名称：数独游戏

// 开发环境：VS2019 + EasyX\_20200902

#include "Global.h"

#include"sudoku\_easyx.h"

extern int sudoku[ROW][COL];//完整版

extern int starting\_grid[ROW][COL];//挖洞版

extern int holes;

int sudoku\_easyx()

{

void (\*fun[2])() = { select1,select3 };

initgraph(900, 660);

while (1)

{

dataInit();

while (process < 2)

fun[process++]();

t1 = clock();

GameInit(gameMode,sudoku);

while (1)

{

if (peekmessage(&msg, EM\_MOUSE, true))

msg = getmessage();

GameUpdate();

GameDraw();

isWin = WinJudge();//是不是填满了

if (isWin) {

//judge\_win();//是不是和sudoku一样

issame = judge\_win();

Sleep(1000);

break;

}

else if (restart) {

Sleep(1000);

break;

}

if (endGame)

break;

//if (noMistake == 0)

//isWin = WinJudge();

/\*if (is\_same || restart)

{

Sleep(1000);

break;

}\*/

}

if (issame)

end();

else

react();

if (endGame)

break;

}

closegraph();

return 0;

}

void dataInit()

{

process = 0;

restart = 0;

sleepTime = 0;

faults = 0;

endGame = 0;

for (int i = 0; i < MAP\_ROW; i++)

for (int j = 0; j < MAP\_COL; j++)

Map1[i][j] = Map2[i][j] =Map\_HL[i][j] = Map\_Mistake[i][j] = 0;

}

void start1()

{

BeginBatchDraw();

setbkcolor(WHITE); //设置窗口背景颜色

settextcolor(BLACK);

setbkmode(1); //设置文字背景透明

cleardevice();

TCHAR s[5];

\_stprintf\_s(s, \_T("%d/2"), process);

outtextxy(0, 0, s);

LOGFONT f;

gettextstyle(&f); // 获取当前字体设置

f.lfHeight = 100; // 设置字体高度为 100

wcscpy\_s(f.lfFaceName, \_T("楷体")); // 设置字体为“黑体”(高版本 VC 推荐使用 \_tcscpy\_s 函数)

f.lfQuality = ANTIALIASED\_QUALITY; // 设置输出效果为抗锯齿

settextstyle(&f); // 设置字体样式

outtextxy(300, 100, \_T("数 独"));

settextstyle(40, 0, \_T("宋体"));

setlinecolor(BLACK);

setlinestyle(PS\_SOLID, 3);

rectangle(340, 290, 560, 360);

if (currentPos.X > 340 && currentPos.X < 560 && currentPos.Y>290 && currentPos.Y < 360)

{

settextcolor(BLUE);

line(370, 350, 530, 350);

}

else

settextcolor(BLACK);

outtextxy(370, 305, L"标准数独");

/\*rectangle(340, 390, 560, 460);

if (currentPos.X > 340 && currentPos.X < 560 && currentPos.Y>390 && currentPos.Y < 460)

{

settextcolor(BLUE);

line(350, 450, 550, 450);

}

else

settextcolor(BLACK);

outtextxy(350, 405, L"对角线数独");\*/

EndBatchDraw();

}

void select1()

{

int post = 1;

while (post)

{

start1();

ExMessage msg = getmessage(); // 获取鼠标信息

int x = msg.x;

int y = msg.y;

switch (msg.message)

{

case WM\_MOUSEMOVE:

currentPos.X = x;

currentPos.Y = y;

break;

case WM\_LBUTTONUP:

if (currentPos.X > 340 && currentPos.X < 560 && currentPos.Y>290 && currentPos.Y < 360)

{

gameMode = 1;

post = 0;

}

/\*if (currentPos.X > 340 && currentPos.X < 560 && currentPos.Y>390 && currentPos.Y < 460)

{

gameMode = 0;

post = 0;

}\*/

break;

default:

break;

}

}

}

void start2()

{

BeginBatchDraw();

setbkcolor(WHITE);

settextcolor(BLACK);

setbkmode(1); //设置文字背景透明

cleardevice();

TCHAR s[5];

\_stprintf\_s(s, \_T("%d/3"), process);

outtextxy(0, 0, s);

settextstyle(40, 0, \_T("宋体"));

setlinecolor(BLACK);

setlinestyle(PS\_SOLID, 3);

//rectangle(340, 90, 560, 160);

if (currentPos.X > 340 && currentPos.X < 560 && currentPos.Y>90 && currentPos.Y < 160)

{

settextcolor(BLUE);

line(370, 150, 530, 150);

}

else

settextcolor(BLACK);

outtextxy(370, 105, L"容 易");

//rectangle(340, 190, 560, 260);

//if (currentPos.X > 340 && currentPos.X < 560 && currentPos.Y>190 && currentPos.Y < 260)

//{

// settextcolor(BLUE);

// line(370, 250, 530, 250);

//}

//else

// settextcolor(BLACK);

//outtextxy(370, 205, L"一 般");

//rectangle(340, 290, 560, 360);

//if (currentPos.X > 340 && currentPos.X < 560 && currentPos.Y>290 && currentPos.Y < 360)

//{

// settextcolor(BLUE);

// line(370, 350, 530, 350);

//}

//else

// settextcolor(BLACK);

//outtextxy(370, 305, L"专 家");

//rectangle(340, 390, 560, 460);

if (currentPos.X > 340 && currentPos.X < 560 && currentPos.Y>390 && currentPos.Y < 460)

{

settextcolor(BLUE);

line(370, 450, 530, 450);

}

else

settextcolor(BLACK);

outtextxy(370, 405, L"返 回");

EndBatchDraw();

}

void select2()

{

int post = 1;

while (post)

{

start2();

ExMessage msg = getmessage(); // 获取鼠标信息

int x = msg.x;

int y = msg.y;

switch (msg.message)

{

case WM\_MOUSEMOVE:

currentPos.X = x;

currentPos.Y = y;

break;

case WM\_LBUTTONUP:

if (currentPos.X > 340 && currentPos.X < 560 && currentPos.Y>90 && currentPos.Y < 160)

{

post = 0;

gameDifficulty = 0;

}

/\*if (currentpos.x > 340 && currentpos.x < 560 && currentpos.y>190 && currentpos.y < 260)

{

post = 0;

gamedifficulty = 1;

}

if (currentpos.x > 340 && currentpos.x < 560 && currentpos.y>290 && currentpos.y < 360)

{

post = 0;

gamedifficulty = 2;

}\*/

if (currentPos.X > 340 && currentPos.X < 560 && currentPos.Y>390 && currentPos.Y < 460)

{

post = 0;

process -= 2;

}

break;

default:

break;

}

}

}

void start3()

{

BeginBatchDraw();

setbkcolor(WHITE);

settextcolor(BLACK);

setbkmode(1); //设置文字背景透明

cleardevice();

TCHAR s[5];

\_stprintf\_s(s, \_T("%d/2"), process);

outtextxy(0, 0, s);

settextstyle(40, 0, \_T("宋体"));

setlinecolor(BLACK);

setlinestyle(PS\_SOLID, 3);

rectangle(340, 190, 560, 260);

if (currentPos.X > 340 && currentPos.X < 560 && currentPos.Y>190 && currentPos.Y < 260)

{

settextcolor(BLUE);

line(370, 250, 530, 250);

}

else

settextcolor(BLACK);

outtextxy(370, 205, L"开始游戏");

rectangle(340, 390, 560, 460);

if (currentPos.X > 340 && currentPos.X < 560 && currentPos.Y>390 && currentPos.Y < 460)

{

settextcolor(BLUE);

line(370, 450, 530, 450);

}

else

settextcolor(BLACK);

outtextxy(370, 405, L"返 回");

EndBatchDraw();

}

void select3()

{

int post = 1;

while (post)

{

start3();

ExMessage msg = getmessage(); // 获取鼠标信息

int x = msg.x;

int y = msg.y;

switch (msg.message)

{

case WM\_MOUSEMOVE:

currentPos.X = x;

currentPos.Y = y;

break;

case WM\_LBUTTONUP:

if (currentPos.X > 340 && currentPos.X < 560 && currentPos.Y>190 && currentPos.Y < 260)

post = 0;

if (currentPos.X > 340 && currentPos.X < 560 && currentPos.Y>390 && currentPos.Y < 460)

{

post = 0;

process -= 2;

}

break;

default:

break;

}

}

}

void GameDraw()

{

BeginBatchDraw();

setbkcolor(WHITE); //设置背景颜色

setbkmode(1);

cleardevice();

LOGFONT f;

gettextstyle(&f); // 获取当前字体设置

f.lfHeight = 40; // 设置字体高度为 40

wcscpy\_s(f.lfFaceName, \_T("楷体")); // 设置字体为“楷体”(高版本 VC 推荐使用 \_tcscpy\_s 函数)

f.lfQuality = ANTIALIASED\_QUALITY; // 设置输出效果为抗锯齿

settextstyle(&f);

/\*画九宫格的框线\*/

setlinecolor(BLACK);

for (int row = 1; row <= MAP\_ROW + 1; row++) //横线

{

if ((row - 1) % 3 == 0)

setlinestyle(PS\_SOLID, 3);

else

setlinestyle(PS\_DASH, 1);

line(60, row \* 60, 600, row \* 60);

}

for (int col = 1; col <= MAP\_COL + 1; col++) //竖线

{

if ((col - 1) % 3 == 0)

setlinestyle(PS\_SOLID, 3);

else

setlinestyle(PS\_DASH, 1);

line(col \* 60, 60, col \* 60, 600);

}

/\*先高亮后outtextxy数字，否则数字会被覆盖\*/

//高亮提示数字

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

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

if (Map\_HL[i][j])

{

if (Map\_HL[i][j] == 1)

setfillcolor(RGB(85, 242, 85));//相同数字用绿色显示

else

setfillcolor(RGB(174, 174, 174));//同行列宫用浅灰色显示

solidrectangle((j + 1) \* 60 + 2, (i + 1) \* 60 + 2, (j + 2) \* 60 - 2, (i + 2) \* 60 - 2);

}

setfillcolor(RGB(99, 99, 99));

if (flag\_w)

solidrectangle((waitPos.X + 1) \* 60 + 2, (waitPos.Y + 1) \* 60 + 2, (waitPos.X + 2) \* 60 - 2, (waitPos.Y + 2) \* 60 - 2);

//高亮错误数字所在的框格

setfillcolor(RGB(250, 44, 20));

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

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

if (Map\_Mistake[i][j])

solidrectangle((j + 1) \* 60 + 1, (i + 1) \* 60 + 1, (j + 2) \* 60 - 1, (i + 2) \* 60 - 1);

//画宫格数字

TCHAR str[10];

if (showanswer == 0) {

for (int row = 0; row < MAP\_ROW; row++)

{

for (int col = 0; col < MAP\_COL; col++)

{

//数字0就不需要显示了

if (Map1[row][col])

{

if (Map2[row][col] == 1)

settextcolor(BLUE);

else if (Map\_HL[row][col] || Map\_Mistake[row][col])

settextcolor(WHITE);

else

settextcolor(RGB(61, 61, 61));

\_stprintf\_s(str, \_T("%d"), Map1[row][col]);

outtextxy((col + 1) \* 60 + 20, (row + 1) \* 60 + 10, str);

}

}

}

}

else {

for (int row = 0; row < MAP\_ROW; row++)

{

for (int col = 0; col < MAP\_COL; col++)

{

//数字0就不需要显示了

if (Map1[row][col])

{

if (Map2[row][col] == 1)

settextcolor(BLUE);

else if (Map\_HL[row][col] || Map\_Mistake[row][col])

settextcolor(WHITE);

else

settextcolor(RGB(61, 61, 61));

\_stprintf\_s(str, \_T("%d"), Map1[row][col]);

outtextxy((col + 1) \* 60 + 20, (row + 1) \* 60 + 10, str);

}

else {

settextcolor(RED);

\_stprintf\_s(str, \_T("%d"),sudoku[row][col]);

outtextxy((col + 1) \* 60 + 20, (row + 1) \* 60 + 10, str);

}

}

}

}

/\*画数字键盘内外框\*/

setlinestyle(PS\_SOLID, 1);

rectangle(660, 60, 840, 300);

rectangle(650, 50, 850, 310);

////////////////////////////////////////////////////////////////////////

//画数字键盘数字 & 计时器

settextcolor(RGB(9, 74, 247));

setlinestyle(PS\_SOLID, 3);

t2 = clock();

second = (int)(double)(t2 - t1) / CLOCKS\_PER\_SEC;

second -= sleepTime;

minute = second / 60;

second %= 60;

\_stprintf\_s(str, \_T("%02d:%02d"), minute, second);

outtextxy(0, 0, str);

/\*\_stprintf\_s(str, \_T("%d"), noMistake);

outtextxy(400, 0, str);\*/

int n = 1;

outtextxy(720, 250, \_T("del"));

for (int col = 0; col <= 2; col++)

{

for (int row = 10; row <= 12; row++)

{

\_stprintf\_s(str, \_T("%d"), n++);

outtextxy((row + 1) \* 60 + 20, (col + 1) \* 60 + 10, str);

}

}

//用红框标注数字键盘中的当前所选数字

setlinecolor(RED);

setlinestyle(PS\_SOLID, 3);

rectangle((numPos.X + 1) \* 60, (numPos.Y + 1) \* 60, (numPos.X + 2) \* 60, (numPos.Y + 2) \* 60);

setfillcolor(RGB(249, 247, 176));

///////////////////////////////////// 左下角 /////////////////////////////////////

setlinecolor(BLACK);

setlinestyle(PS\_SOLID, 2);

settextstyle(30, 0, \_T("宋体"));

setfillcolor(RGB(158, 160, 169));

if (currentPos.X > 660 && currentPos.X < 840 && currentPos.Y > 350 && currentPos.Y < 400)

{

solidrectangle(660, 350, 840, 400);

settextcolor(WHITE);

}

else

settextcolor(BLACK);

outtextxy(670, 360, \_T(" 暂 停"));

if (currentPos.X > 660 && currentPos.X < 840 && currentPos.Y > 420 && currentPos.Y < 470)

{

solidrectangle(660, 420, 840, 470);

settextcolor(WHITE);

}

else

settextcolor(BLACK);

outtextxy(670, 430, \_T(" 重新开始"));

if (currentPos.X > 660 && currentPos.X < 840 && currentPos.Y > 490 && currentPos.Y < 540)

{

solidrectangle(660, 490, 840, 540);

settextcolor(WHITE);

}

else

settextcolor(BLACK);

outtextxy(670, 500, \_T(" 退 出"));

if (currentPos.X > 660 && currentPos.X < 840 && currentPos.Y > 560 && currentPos.Y < 610)

{

solidrectangle(660, 560, 840, 610);

settextcolor(WHITE);

}

else

settextcolor(BLACK);

outtextxy(670, 570, \_T(" 显示答案"));

setfillcolor(BLACK);

solidcircle(650, 380, 5);

solidcircle(650, 450, 5);

solidcircle(650, 520, 5);

solidcircle(650, 590, 5);

EndBatchDraw();

}

void GameUpdate()

{

if (peekmessage(&msg, EM\_MOUSE, true))

msg = getmessage();

int x = msg.x;

int y = msg.y;

switch (msg.message)

{

case WM\_MOUSEMOVE:

if (x > 60 && x < 600 && y > 60 && y < 600)

{

flag\_w = 1;

waitPos.X = (x - 60) / 60;

waitPos.Y = (y - 60) / 60;

coord.X = waitPos.Y;

coord.Y = waitPos.X;

coord = FindLeftUpperCornor(coord);

if (Map1[waitPos.Y][waitPos.X])

HeighLight(1);

else

HeighLight(0);

}

else

{

//消除红框

flag\_w = 0;

//消除高亮

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

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

Map\_HL[i][j] = 0;

}

currentPos.X = x;

currentPos.Y = y;

break;

case WM\_LBUTTONUP:

if (x > 660 && x < 840 && y>60 && y < 240)

{

numPos.X = (x - 60) / 60;

numPos.Y = (y - 60) / 60;

num = 3 \* numPos.Y + (numPos.X - 10) + 1;

}

else if (x > 720 && y > 240 && x < 780 && y < 300)

{

numPos.X = (x - 60) / 60;

numPos.Y = (y - 60) / 60;

num = 0;

}

else if (x > 60 && x < 600 && y > 60 && y < 600)

{

if (Map2[waitPos.Y][waitPos.X] == 1)

Map1[waitPos.Y][waitPos.X] = num;

else

{

clock\_t t3, t4;

t3 = clock();

MessageBox(GetHWnd(), L"提示数字不可修改", L"注意", 0);

t4 = clock();

sleepTime += (int)(double)(t4 - t3) / CLOCKS\_PER\_SEC;

}

noMistake = FindMistake(gameMode);

faults += noMistake;

}

else if (currentPos.X > 660 && currentPos.X < 840 && currentPos.Y > 350 && currentPos.Y < 400)

{

clock\_t t3, t4;

t3 = clock();

MessageBox(GetHWnd(), L"暂停中...", L"提示", 0);

t4 = clock();

sleepTime += (int)(double)(t4 - t3) / CLOCKS\_PER\_SEC;

}

else if (currentPos.X > 660 && currentPos.X < 840 && currentPos.Y > 420 && currentPos.Y < 470)

{

clock\_t t3, t4;

t3 = clock();

int result = MessageBox(GetHWnd(), \_T("重新开始?"), \_T("hello"), MB\_YESNO);

switch (result)

{

case IDNO:

break;

case IDYES:

restart = 1;

break;

}

t4 = clock();

sleepTime += (int)(double)(t4 - t3) / CLOCKS\_PER\_SEC;

}

else if (currentPos.X > 660 && currentPos.X < 840 && currentPos.Y > 490 && currentPos.Y < 540)

{

int result=MessageBox(GetHWnd(), L"确认退出？", L"提示", MB\_YESNO);

switch (result)

{

case IDNO:

break;

case IDYES:

endGame = 1;

break;

}

}

else if (currentPos.X > 660 && currentPos.X < 840 && currentPos.Y > 560 && currentPos.Y < 610)

{

int result = MessageBox(GetHWnd(), L"显示答案", L"提示", MB\_YESNO);

switch (result)

{

case IDNO:

break;

case IDYES:

{

showanswer = 1;

break;

}

}

}

break;

default:

break;

}

//the need of‘faults'

msg.message = WM\_MBUTTONUP;

}

void HeighLight(int post)

{

int i, j;

//清除上次的高亮数据

for (i = 0; i < 9; i++)

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

Map\_HL[i][j] = 0;

if (waitPos.Y >= 0 && waitPos.Y <= 8 && waitPos.X >= 0 && waitPos.X <= 8)

{

/\*两种高亮模式\*/

if (post)//1.高亮相同的数字

{

for (i = 0; i < 9; i++)

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

if (Map1[i][j] == Map1[waitPos.Y][waitPos.X])

Map\_HL[i][j] = 1;

}

else//2.高亮同行、同列、同宫

{

for (i = 0; i < 9; i++)

Map\_HL[i][waitPos.X] = 2;

for (i = 0; i < 9; i++)

Map\_HL[waitPos.Y][i] = 2;

//2.1根据每一宫的左上角的行列来高亮对应的那一宫

for (i = coord.X; i < coord.X + 3; i++)

for (j = coord.Y; j < coord.Y + 3; j++)

if (i >= 0 && i < 9 && j >= 0 && j < 9)

Map\_HL[i][j] = 2;

if (gameMode == 0)

{

if (waitPos.Y == waitPos.X)

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

Map\_HL[i][i] = 2;

if (waitPos.Y + waitPos.X == 8)

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

Map\_HL[i][8 - i] = 2;

}

}

}

}

COORD FindLeftUpperCornor(COORD c1)

{

COORD c2;

for (int i = 0; i <= 6; i += 3)

{

c2.X = i;

for (int j = 0; j <= 6; j += 3)

{

c2.Y = j;

if (c1.X >= c2.X && c1.X <= c2.X + 2 && c1.Y >= c2.Y && c1.Y <= c2.Y + 2)

return c2;

}

}

//这是被迫加上的

return c2;

}

int FindMistake(int mode)

{

int i, j, r, c, post = 0;

//清除上次的错误数据

for (i = 0; i < 9; i++)

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

Map\_Mistake[i][j] = 0;

//对每列检查错误

for (i = 0; i < 9; i++)

{

int arr[10] = { 0 };

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

arr[Map1[i][j]]++;

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

if (arr[Map1[i][j]] >= 2 && Map1[i][j] != 0)

{

Map\_Mistake[i][j] = 1;

post = 1;

}

}

//对每行检查错误

for (i = 0; i < 9; i++)

{

int arr[10] = { 0 };

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

arr[Map1[j][i]]++;

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

if (arr[Map1[j][i]] >= 2 && Map1[j][i] != 0)

{

post = 1;

Map\_Mistake[j][i] = 1;

}

}

//对每宫检查错误

for (i = 0; i <= 6; i += 3)

{

for (j = 0; j <= 6; j += 3)

{

int arr[10] = { 0 };

for (r = i; r <= i + 2; r++)

for (c = j; c <= j + 2; c++)

arr[(Map1[r][c])]++;

for (r = i; r <= i + 2; r++)

for (c = j; c <= j + 2; c++)

if (arr[Map1[r][c]] >= 2 && Map1[r][c] != 0)

{

Map\_Mistake[r][c] = 1;

post = 1;

}

}

}

//对角线数独还需再判断对角线上数字是否符合条件

if (mode == 0)

{

int arr[10] = { 0 };

i = 0;

while (i < 9)

arr[Map1[i][i++]]++;

i = 0;

while (i < 9)

{

if (arr[Map1[i][i]] >= 2 && Map1[i][i] != 0)

{

Map\_Mistake[i][i] = 1;

post = 1;

}

i++;

}

int arrb[10] = { 0 };

i = 0;

while (i < 9)

arrb[Map1[i][8 - i++]]++;

i = 0;

while (i < 9)

{

if (arrb[Map1[i][8 - i]] >= 2 && Map1[i][8 - i] != 0)

{

Map\_Mistake[i][8 - i] = 1;

post = 1;

}

i++;

}

}

return post;

}

int WinJudge()

{

int post = 1, P = 0;

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

{

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

if (Map1[i][j] == 0)

{

post = 0;

P = 1;

break;

}

if (P)

break;

}

return post;

}

int judge\_win() {

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

{

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

if (Map1[i][j]!=sudoku[i][j])

{

return 0;

}

}

return 1;

}

void GameInit(int post, int a[][9])

{

//srand((unsigned)time(NULL));

//int a[10];

//int b[10] = { 0 };

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

//{

// do {

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

// } while (b[a[i]]);

// b[a[i]]++;

//}

////现在两个数组a和c已经生成了对应关系了(很乱，但一一对应)

////模式一(标准数独)

//if (post)

//{

// int map[9][9] = { {9,8,2,5,3,6,1,7,4},

// {4,3,6,7,1,9,5,8,2},

// {1,5,7,4,8,2,3,6,9},

// {8,4,5,6,2,7,9,3,1},

// {2,6,3,1,9,5,7,4,8},

// {7,1,9,8,4,3,2,5,6},

// {3,2,4,9,5,8,6,1,7},

// {6,9,1,3,7,4,8,2,5},

// {5,7,8,2,6,1,4,9,3} };

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

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

Map1[i][j] = a[i][j];

//}

//else //对角线数独

//{

// int map[9][9] = { {2,3,1,4,5,8,6,9,7},

// {5,4,7,6,9,3,2,1,8},

// {6,8,9,7,2,1,4,5,3},

// {8,9,6,1,4,5,3,7,2},

// {7,5,3,2,8,9,1,6,4},

// {1,2,4,3,6,7,9,8,5},

// {3,7,2,8,1,6,5,4,9},

// {4,6,5,9,7,2,8,3,1},

// {9,1,8,5,3,4,7,2,6} };

// for (int i = 0; i < 9; i++)

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

// Map1[i][j] = a[(map[i][j])];

//}

Dig(gameDifficulty);

}

void Dig(int post)

{

// srand((unsigned)time(NULL));

// int way[3][9] = { { 3,3,3,3,3,4,4,5,5 },

// { 4,4,4,4,4,5,5,6,6 },

// { 5,5,5,5,5,6,6,7,7 } };

// int t[9] = { 0 };

// int tmp[9] = { 0 }, n = 0;

// for (int i = 0; i < 9; i++)

// {

// do {

// n = rand() % 9;

// } while (t[n]);

// t[n]++;

// tmp[i] = way[post][n];

// }

// int m = 0, x, y;

// for (int i = 0; i < 9; i += 3)

// for (int j = 0; j < 9; j += 3)

// {

// int dig[3][3] = { 0 };

// for (int k = 0; k < tmp[m]; k++)

// {

// do {

// x = rand() % 3;

// y = rand() % 3;

// } while (dig[x][y]);

// dig[x][y] = 1;

// Map2[i + x][j + y] = 1;//标记被挖去的位置

// }

// m++;

//

//

// }

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

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

if(starting\_grid[i][j]==0)

Map2[i][j] = 1;

if(Map2[i][j])

Map1[i][j] = 0;

}

}

}

void react() {

setbkcolor(WHITE);

settextcolor(BLACK);

setbkmode(1);

setfillcolor(RGB(158, 160, 169));

cleardevice();

LOGFONT f;

gettextstyle(&f);

f.lfHeight = 200;

wcscpy\_s(f.lfFaceName, \_T("楷体"));

f.lfQuality = ANTIALIASED\_QUALITY;

settextstyle(&f);

outtextxy(30, 50, \_T("YOU"));

outtextxy(30, 350, \_T("Fail!"));

settextstyle(40, 0, \_T("宋体"));

outtextxy(550, 60, \_T("游戏时间："));

solidrectangle(550, 110, 850, 160);

outtextxy(550, 190, \_T("错误次数："));

solidrectangle(550, 240, 850, 290);

outtextxy(550, 330, \_T("总得分 ："));

solidrectangle(550, 380, 850, 430);

TCHAR str[20];

settextcolor(WHITE);

\_stprintf\_s(str, \_T("%02d分%02d秒"), minute, second);

outtextxy(620, 110, str);

\_stprintf\_s(str, \_T("%d"), faults);

outtextxy(680, 240, str);

float a = (1 - (minute + (float)second / 60) / ((1) \* 100)) \* 20;

float b = (float)(80 - faults) / 800 \* 80;

if ((minute \* 60 + second) > (600 \* (1)))

a = 0;

if (faults >= 30)

b = 0;

float scores = a + b;

\_stprintf\_s(str, \_T("%.2f"), scores);

outtextxy(650, 380, str);

Sleep(1000);

int result = MessageBox(GetHWnd(), \_T("再来一局?"), \_T("hello"), MB\_YESNO);

switch (result)

{

case IDNO:

endGame = 1;

break;

case IDYES:

break;

}

}

void end()

{

setbkcolor(WHITE);

settextcolor(BLACK);

setbkmode(1);

setfillcolor(RGB(158, 160, 169));

cleardevice();

LOGFONT f;

gettextstyle(&f);

f.lfHeight = 200;

wcscpy\_s(f.lfFaceName, \_T("楷体"));

f.lfQuality = ANTIALIASED\_QUALITY;

settextstyle(&f);

outtextxy(30, 50, \_T("Good"));

outtextxy(30, 350, \_T("job!"));

settextstyle(40, 0, \_T("宋体"));

outtextxy(550, 60, \_T("完成时间："));

solidrectangle(550, 110, 850, 160);

outtextxy(550, 190, \_T("错误次数："));

solidrectangle(550, 240, 850, 290);

outtextxy(550, 330, \_T("总得分 ："));

solidrectangle(550, 380, 850, 430);

TCHAR str[20];

settextcolor(WHITE);

\_stprintf\_s(str, \_T("%02d分%02d秒"), minute, second);

outtextxy(620, 110, str);

\_stprintf\_s(str, \_T("%d"), faults);

outtextxy(680, 240, str);

float a = (1 - (minute + (float)second / 60) / ((1) \* 10)) \* 20;

float b = (float)(80 - faults) / 80 \* 80;

if ((minute \* 60 + second) > (600 \* (1)))

a = 0;

if (faults >= 30)

b = 0;

float scores = a + b;

\_stprintf\_s(str, \_T("%.2f"), scores);

outtextxy(650, 380, str);

Sleep(1000);

int result = MessageBox(GetHWnd(), \_T("再来一局?"), \_T("hello"), MB\_YESNO);

switch (result)

{

case IDNO:

endGame = 1;

break;

case IDYES:

break;

}

}

**7.3 头文件**

// Created by PC on 2019/2/27.

#ifndef WORKSHOP\_GLOBAL\_H

#define WORKSHOP\_GLOBAL\_H

#include <iostream>

#include <fstream>

#include <ctime>

using namespace std;

#define CHOOSE\_SOLUTION 500

#define TRUE 1

#define FALSE 0

#define ROW 9

#define COL 9

#define NoAnwser -1

typedef int status;

typedef struct DataNode {

int data = 0;

DataNode \*next{};

}DataNode;

typedef struct HeadNode {

int Num = 0;

DataNode \*right{};

HeadNode \*down{};

}HeadNode;

struct consequence {

int value = -1;//存真值 真时为true-1，假时为false-0

};

struct conse {

int num = 0;

int value = -1;//存真值 真时为true-1，假时为false-0

};

struct Count {

int num = 0;

int count = 0;

Count \*next=nullptr;

};

void Start();

HeadNode\* CreateClause(int &,string &);

status DPLL1(HeadNode \*LIST,consequence \*result);

status DPLL2(HeadNode\* LIST, consequence\* result);

HeadNode\* IsSingleClause(HeadNode\*);

status IsEmptyClause(HeadNode\*);

HeadNode\* ADDSingleClause(HeadNode\*,int);

HeadNode\* Duplication(HeadNode\*);

void DeleteHeadNode(HeadNode\*,HeadNode\*&);////////////////////////////////////

void DeleteDataNode(int,HeadNode\*&);/////////////////////////////////////////

void FreeHeadNode(HeadNode\* Clause);

int Choose\_max(HeadNode\* LIST);

void show(struct consequence \*,int);

int Digit(int a[][COL], int i, int j);

void randomFirstRow(int a0[], int n);

void createSudoku(int a[][COL]);

void createStartinggrid(const int a[][COL], int b[][COL], int numDigits);

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

string ToCnf(int a[][COL],int holes);

string createSudokuToFile();

status SudoDPLL(HeadNode \*LIST,conse \*result,int VARNUM);

void SudokuShow(conse \*result,int VARNUM);

void Uniquejudge(const int a[][COL], const int b[][COL]);

int sudoku\_easyx();

/////////////////////////////////////////////////////////////////////////////

#endif //WORKSHOP\_GLOBAL\_H

// 作者：陈昊 <2975280738@qq.com>

// 最后修改：2020-11-1

//

#ifndef SUDOKU\_EASYX\_H

#define SUDOKU\_EASYX\_H

#include <conio.h>

#include <graphics.h>

#include <time.h>

#define MAP\_COL 9

#define MAP\_ROW 9

COORD currentPos = { 0,0 }, numPos = { 10,0 }, coord, waitPos;

clock\_t t1, t2;

ExMessage msg;

int sleepTime = 0;

int flag\_w = 1;

int num = 1;

int process;

int second = 0, minute = 0;

int gameDifficulty = 0, gameMode = 0, restart, faults, endGame;

int isWin = 0, noMistake = 0;

int issame = 0;

int Map1[MAP\_ROW][MAP\_COL], Map2[MAP\_ROW][MAP\_COL];

int Map\_HL[MAP\_ROW][MAP\_COL], Map\_Mistake[MAP\_ROW][MAP\_COL];

int showanswer = 0;

void react();

int judge\_win();

void dataInit();

void select1();

void start1();

void select2();

void start2();

void select3();

void start3();

void end();

void GameDraw();

void GameUpdate();

void HeighLight(int);

COORD FindLeftUpperCornor(COORD);

int FindMistake(int);

int WinJudge();

void GameInit(int,int[][9]);

void Dig(int);

//#pragma once

#endif;//