# 兰顿蚂蚁第五次工作报告

软件1班 202430551743 严嘉鹏

1. **工作总结**

本来这周有很多功能要实现，但卡在了第一个任务（自由模式题库）上，到现在都只完成了一个半成品。

我想实现的自由模式题库功能：用户可以将自己在自由模式下电脑随机生成的有效题目上传至同一个题库，然后用户可以查看题库选择题目尝试作答。

经过一番推敲：我确定的顺序：先提示用户是否存档，再提示用户是否查看题库，若选择查看题库，先整齐地打印所有题目的序号（题目1，题目2，每行最多5个），然后提示用户输入想尝试的题目序号，程序打印棋盘和题目信息，提示用户作答，程序返回正误，作答完毕看完作答结果后有两个选择1、回到题库（重复刚才的步骤，可以让用户继续尝试其他题目）2、退出题库

然后我开始写函数,一开始我用struct集合棋盘和题目信息，创建了problemBank这个指向该结构体类型的vector数组，然后定义两个接口用于存档和查看题库。

void saveProblem(int x, int y, Direction heading, int finalX, int finalY, Direction finalHeading);

void viewProblemBank();

运行后发现题库里始终只有一个题，原来是problemBank是一个非静态的成员变量，这意味着每次程序运行时，题库都是空的，除非用户显式添加题目。当程序结束时，vector被销毁，其中的数据会丢失。

所以又添加了两个函数用来实现把题目存到文件里

// 从文件中读取题目信息

void loadProblemsFromFile();

// 将题目信息保存到文件

void saveProblemsToFile();

由于对文件的操作不熟悉，理清这四个函数的顺序花费了大量时间。4个函数定义如下：

// 将题目信息保存到文件

void freedomMode::saveProblemsToFile() {

std::ofstream file(filePath,std::ios::app);

if (file.is\_open()) {

for (const auto& problem : problemBank) {

file << problem.size << " " << problem.steps << " " << problem.initialX << " "

<< problem.initialY << " " << static\_cast<int>(problem.initialHeading) << " "

<< problem.finalX << " " << problem.finalY << " "

<< static\_cast<int>(problem.finalHeading) << " ";

for (int i = 0; i < problem.size \* problem.size; ++i) {

file << problem.board[i] << " ";

}

file << std::endl;

}

file.close();

}

}

// 从文件中读取题目信息

void freedomMode::loadProblemsFromFile()

{

// 清空题库

for (auto& problem : problemBank) {

delete[] problem.board;

}

problemBank.clear();

std::ifstream file(filePath);

if (file.is\_open()) {

std::string line;

while (std::getline(file, line)) {

std::istringstream iss(line);

ProblemInfo problem;

int initialHeadingInt, finalHeadingInt;

iss >> problem.size >> problem.steps >> problem.initialX >> problem.initialY

>> initialHeadingInt >> problem.finalX >> problem.finalY

>> finalHeadingInt;

problem.initialHeading = static\_cast<Direction>(initialHeadingInt);

problem.finalHeading = static\_cast<Direction>(finalHeadingInt);

problem.board = new bool[problem.size \* problem.size];

for (int i = 0; i < problem.size \* problem.size; ++i) {

iss >> problem.board[i];

}

problemBank.push\_back(problem);

}

file.close();

}

}

// 判断某个题目信息是否已经存在于题库中

bool freedomMode::isProblemExists(const ProblemInfo& newProblem) {

for (const auto& problem : problemBank) {

if (problem.size == newProblem.size &&

problem.steps == newProblem.steps &&

problem.initialX == newProblem.initialX &&

problem.initialY == newProblem.initialY &&

problem.initialHeading == newProblem.initialHeading &&

problem.finalX == newProblem.finalX &&

problem.finalY == newProblem.finalY &&

problem.finalHeading == newProblem.finalHeading) {

bool boardsMatch = true;

for (int i = 0; i < problem.size \* problem.size; ++i) {

if (problem.board[i] != newProblem.board[i]) {

boardsMatch = false;

break;

}

}

if (boardsMatch) {

return true;

}

}

}

return false;

}

// 新增函数用于存档

void freedomMode::saveProblem(int x, int y, Direction heading, int finalX, int finalY, Direction finalHeading) {

ProblemInfo problem;

problem.size = size;

problem.steps = steps;

problem.initialX = x;

problem.initialY = y;

problem.initialHeading = heading;

problem.finalX = finalX;

problem.finalY = finalY;

problem.finalHeading = finalHeading;

problem.board = new bool[size \* size];

copyBoard(initialBoard, problem.board);

if (isProblemExists(problem)) {

std::cout << "题库已存在该题，无需重复保存。" << std::endl;

delete[] problem.board;

return;

}

problemBank.push\_back(problem);

std::cout << "题目已存入题库" << std::endl;

}

// 新增函数用于查看题库

void freedomMode::viewProblemBank() {

std::cout << "自由模式题库：" << std::endl;

for (size\_t i = 0; i < problemBank.size(); ++i) {

std::cout << "题目" << i + 1 << " ";

if ((i + 1) % 5 == 0) {

std::cout << std::endl;

}

}

std::cout << std::endl;

int problemIndex;

std::cout << "请输入你想查看的题目序号 ";

std::cin >> problemIndex;

if (problemIndex < 1 || problemIndex > static\_cast<int>(problemBank.size()))

{

std::cout << "无效的题目序号，请重新输入" << std::endl;

viewProblemBank();

return;

}

ProblemInfo problem = problemBank[problemIndex - 1];

std::cout << "题目信息如下：" << std::endl;

std::cout << "步数: " << problem.steps << std::endl;

std::cout << "最终位置: (" << problem.finalX << ", " << problem.finalY << ")" << std::endl;

std::cout << "最终朝向: ";

switch (problem.finalHeading) {

case UP:

std::cout << "UP";

break;

case DOWN:

std::cout << "DOWN";

break;

case LEFT:

std::cout << "LEFT";

break;

case RIGHT:

std::cout << "RIGHT";

break;

}

std::cout << std::endl;

std::cout << "蚁行图如下：" << std::endl;

freedomMode tempGame(problem.size, problem.steps);

tempGame.copyBoard(problem.board, tempGame.initialBoard);

tempGame.generateFinalBoard(problem.initialX, problem.initialY, problem.initialHeading);

tempGame.printFinalBoard();

bool\* userBoard = new bool[problem.size \* problem.size];

while (true)

{

std::cout << "请用户尝试作答，输入原始棋盘 (0 表示白格, 1 表示黑格):" << std::endl;

for (int i = 0; i < problem.size; ++i) {

for (int j = 0; j < problem.size; ++j) {

std::cin >> userBoard[i \* problem.size + j];

}

}

if (tempGame.checkSolution(userBoard))

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

// 设置文本颜色为绿色

SetConsoleTextAttribute(hConsole, FOREGROUND\_GREEN);

std::cout << "回答正确！" << std::endl;

// 恢复默认颜色

SetConsoleTextAttribute(hConsole, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);

int choice = 0;

std::cout << "输入 1 回到题库，输入 2 退出题库: ";

std::cin >> choice;

if (choice == 1)

{

delete[] userBoard;

viewProblemBank();

return;

}

else if (choice == 2) {

delete[] userBoard;

return;

}

}

else

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

// 设置文本颜色为红色

SetConsoleTextAttribute(hConsole, FOREGROUND\_RED);

std::cout << "回答错误！输入 1 重新作答，输入 2 查看答案: " << std::endl;

// 恢复默认颜色

SetConsoleTextAttribute(hConsole, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);

int subChoice;

std::cin >> subChoice;

if (subChoice == 1) {

continue;

}

else if (subChoice == 2) {

std::cout << "初始棋盘答案如下：" << std::endl;

tempGame.printInitialBoard();

int choice;

std::cout << "输入 1 回到题库，输入 2 退出题库: ";

std::cin >> choice;

if (choice == 1) {

delete[] userBoard;

viewProblemBank();

return;

}

else if (choice == 2) {

delete[] userBoard;

return;

}

}

}

}

}

一运行刚开始一切合理，多测试几次后，发现了题库中会记录重复的题目以及题库里原本就有的题又被重复保存。我尝试调试这四个函数的位置，又清空vector数组，但还是解决不了，还遇到了一个新的问题：不同用户保存题目进入题库，和同一用户多次玩该模式将多个题目保存进入题库，能不能实现效果一致？

到现在也不知道问题出现在哪里，只是得出了几个可能的原因：

1文件函数的放置位置不对，导致出现了多个题库副本

2不同用户保存题目进入题库，和同一用户多次玩该模式将多个题目保存进入题库，这两种情况要分开讨论

本质上还是对这四个函数的逻辑不够清晰，找不到问题出在哪。反思改进：写实训要多安排一些时间，每次开始前，将任务具体到某个点，解决完一个点再考虑下一个点，非必要不同时考虑。

1. **下周计划**

将这4个函数从头到尾理一遍，看一看到底错在哪。能把题库做好就添加该功能，实在不行就制作简化版题库，还不行就放弃。

1. **实验运行结果（尚未找到题库中题目数量>实际存入的题目数量这一问题的解决方案）**

