C 语言程序设计 — 大作业

五子棋游戏

(Gomoku)

姓名：_____ 学号：_____

班级：_____ 日期：_____

满分：100 分

提交：.c 源文件 + .exe 可执行文件 + 运行截图

作业要求：

- 独立完成全部代码编写，禁止抄袭。
- 代码要求结构清晰、注释完整、风格规范。
- 需编译为 Windows 平台可执行文件 (.exe)。
- 提交时包含源程序、可执行文件和运行截图。

项目概述

项目目标

使用 C 语言开发一个完整的五子棋（Gomoku）双人对战游戏，运行于控制台界面。要求实现完整的游戏逻辑、良好的用户交互体验，并最终打包为 Windows 可执行文件（.exe）。

功能需求

- **标准棋盘**：15×15 标准五子棋棋盘。
- **双人对战**：黑棋 (O) 与白棋 (X) 轮流落子。
- **落子交互**：通过输入行列坐标进行落子。
- **胜利检测**：横、竖、正斜、反斜四个方向五子连珠检测。
- **悔棋功能**：支持撤回上一步落子。
- **保存/载入**：将游戏状态保存到文件，支持继续未完成的游戏。
- **平局判断**：棋盘满时判定为平局。
- **输入校验**：对越界、重复落子等非法输入给予提示。
- **坐标辅助**：棋盘四周显示行列数字，方便定位。

系统设计

总体架构

程序采用模块化设计，主要分为以下模块：

- **初始化模块**：游戏状态初始化、棋盘清零。
- **显示模块**：绘制棋盘、状态信息、菜单。
- **游戏逻辑模块**：落子、胜负判定、平局判定。
- **悔棋模块**：历史记录管理、撤回操作。
- **存档模块**：文件读写、状态序列化。
- **输入处理模块**：玩家输入解析、特殊命令处理。

数据结构设计

棋盘表示：棋盘使用二维字符数组表示：

```
char board[BOARD_SIZE][BOARD_SIZE];  /* 15x15 字符数组 */
/* EMPTY='.', BLACK='O', WHITE='X' */
```

坐标与历史记录：

```
typedef struct {
    int row;
    int col;
} Position;

Position history[MAX_STEPS];  /* 落子历史（最多225步） */
```

游戏状态结构体：

```
typedef struct {
    char board[BOARD_SIZE][BOARD_SIZE];
    int  step_count;
    char current_player;  /* BLACK 或 WHITE */
    char winner;          /* 赢家/EMPTY */
    int  game_over;
    Position history[MAX_STEPS];  /* 悔棋用 */
} Game;
```

函数模块划分

函数名	功能说明
init_game	初始化游戏所有状态
init_board	初始化棋盘为全空

函数名	功能说明
print_board	绘制棋盘界面（含坐标辅助）
print_status	显示当前游戏状态
print_menu	显示主菜单
make_move	在指定位置落子
is_valid_move	检查落子合法性
check_win	检查是否有人获胜
check_direction	检查指定方向连子数
check_draw	检查是否平局
switch_player	切换当前玩家
undo_move	悔棋操作
save_game	保存游戏到文件
load_game	从文件加载游戏
get_player_input	获取玩家输入并解析

核心算法详解

胜负判定算法

五子棋的胜负判定是在每次落子后，检查该落子点在四个方向上是否有连续五个同色棋子。

算法步骤：

1. 以当前落子位置为起点。
2. 在水平、垂直、正斜（/）、反斜（\）四个方向分别检测。
3. 每个方向沿正反两个方向延伸，统计连续同色棋子数。
4. 任一方向计数 ≥ 5 即为获胜。

```c
int check_win(const Game *game, int row, int col)
{
    /* 四个方向：水平、垂直、正斜(\)、反斜(/) */
    int dr[] = {0, 1, 1, 1};
    int dc[] = {1, 0, 1, -1};

    for (int i = 0; i < 4; i++) {
        if (check_direction(game, row, col, dr[i], dc[i]) >= 5) {
            return 1;
        }
    }
    return 0;
}
```

`check_direction` 函数沿指定方向正反延伸统计：

```c
int check_direction(const Game *game, int row, int col, int dr,
    int dc)
{
    int count = 1;  /* 包含起始位置 */
    char player = game->board[row][col];

    /* 正方向延伸 */
    int r = row + dr, c = col + dc;
    while (合法范围内 && game->board[r][c] == player) {
        count++; r += dr; c += dc;
    }

    /* 反方向延伸 */
    r = row - dr; c = col - dc;
    while (合法范围内 && game->board[r][c] == player) {
        count++; r -= dr; c -= dc;
    }

    return count;
}
```

悔棋实现

悔棋通过维护落子历史数组实现：

1. 每次落子将坐标存入 `history[step_count]`，`step_count` 自增。
2. 悔棋时 `step_count` 减 1，取出上一步坐标，将棋盘对应位置恢复为空。
3. 切换回上一个玩家。

存档实现

使用 `fwrite` / `fread` 直接序列化整个 `Game` 结构体到文件：

```c
int save_game(const Game *game) {
    FILE *fp = fopen("gomoku_save.dat", "wb");
    fwrite(game, sizeof(Game), 1, fp);
    fclose(fp);
}

int load_game(Game *game) {
    FILE *fp = fopen("gomoku_save.dat", "rb");
    fread(game, sizeof(Game), 1, fp);
    fclose(fp);
}
```

编译与打包为 exe

Linux / macOS 环境编译

```
# 编译
gcc -o gomoku gomoku.c -Wall -O2 -lm

# 运行
./gomoku

# 清理
make clean
```

Windows 环境编译（MinGW）

```
# 使用 MinGW-w64 编译
gcc -o gomoku.exe gomoku.c -Wall -O2 -lm

# 运行
gomoku.exe
```

Linux 下交叉编译 Windows exe

需要安装 MinGW-w64 交叉编译工具链：

```
# 安装 MinGW-w64
sudo apt install mingw-w64

# 交叉编译
x86_64-w64-mingw32-gcc gomoku.c -o gomoku_win64.exe -Wall -O2 -lm

# 生成 gomoku_win64.exe，可在 Windows 上直接运行
```

使用 Makefile

项目提供了完整的 Makefile，支持以下命令：

```
make           # 编译 Linux 版本
make run       # 编译并运行
make clean     # 清理编译产物和存档
make exe-win   # 交叉编译 Windows exe（需 mingw-w64）
make exe-mingw# MinGW 原生编译 exe
```

打包发布步骤

1. 使用交叉编译生成 gomoku_win64.exe。

2. 将 exe 放在一个单独的文件夹中。
3. 可选：使用 UPX 压缩 exe 体积。
4. 将文件夹打包为 zip 发布。

游戏操作说明

启动与菜单

- 启动后显示主菜单，可选择新游戏、载入存档、游戏说明。
- 选择「开始新游戏」或载入存档后进入游戏主界面。

落子操作

请输入坐标（行 列）：7 7

表示在第 7 行第 7 列落子。棋盘行号和列号范围均为 1–15。

特殊命令

- 0 0 — 悔棋：撤回上一步落子。
- -1 -1 — 保存：将当前游戏状态保存到文件。
- -2 -2 — 退出：退出游戏（会二次确认）。

界面示例

```
    1  2  3  4  5  6  7  8  9 10 11 12 13 14 15

 1  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 1

 2  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 2

 3  .  .  .  .  .  .  O  .  .  .  .  .  .  .  . 3

 4  .  .  .  .  .  X  .  .  .  .  .  .  .  .  . 4

    1  2  3  4  5  6  7  8  9 10 11 12 13 14 15

当前玩家:  黑棋 (O) (请落子)
已走步数: 2
```

评分标准

评分项	分值	得分标准
1. 棋盘显示与交互界面	15 分	棋盘绘制正确，坐标辅助完整，界面美观
2. 落子功能与输入校验	15 分	正确落子，坐标合法性检查，重复落子提示
3. 胜负判定算法	20 分	四个方向五子连珠检测完全正确
4. 悔棋功能	10 分	正确撤回上一步，恢复棋盘状态和当前玩家
5. 保存/加载功能	10 分	文件读写正确，能完整恢复游戏状态
6. 平局判断	5 分	棋盘满时正确判定平局
7. 代码结构与风格	10 分	模块化设计，命名规范，注释充分
8. 程序鲁棒性	10 分	异常输入处理，不会崩溃，用户提示友好
9. 打包为 exe	5 分	成功编译为 Windows 平台 exe 可执行文件
总分	**100 分**	

各评分项详细说明

1. 棋盘显示与交互界面 (15 分)

5 分 15×15 棋盘正确显示，行列号辅助标注完整。

5 分 黑棋和白棋使用不同符号清晰区分。

5 分 界面布局合理，游戏状态信息（当前玩家、步数）实时更新。

2. 落子功能与输入校验 (15 分)

5 分 能正确接收坐标输入并在指定位置落子。

5 分 对越界坐标进行检测并提示。

5 分 对已有棋子的位置进行检测并提示禁止落子。

3. 胜负判定算法 (20 分)

5 分 水平方向五子连珠检测正确。

5 分 垂直方向五子连珠检测正确。

5 分 正斜方向（/）五子连珠检测正确。

5 分 反斜方向（\）五子连珠检测正确。

4. 悔棋功能 (10 分)

5 分 能正确撤回上一步落子。

5 分 撤回后当前玩家正确切换，无棋可悔时有提示。

5. 保存/加载功能 (10 分)

5 分 保存功能正确，能生成存档文件。

5 分 加载功能正确，能恢复完整的游戏状态继续游戏。

6. 平局判断 (5 分)

5 分 棋盘上所有位置均被占满时正确判定为平局。

7. 代码结构与风格 (10 分)

4 分 函数模块划分合理，代码复用性好。

3 分 变量命名规范，缩进一致，代码可读性强。

3 分 关键函数有充分的注释说明。

8. 程序鲁棒性 (10 分)

4 分 非法输入（非数字、特殊字符）不导致程序崩溃。

3 分 所有操作都有明确的用户提示。

3 分 退出时有确认操作，防止误退出。

9. 打包为 exe (5 分)

3 分 成功生成.exe 可执行文件。

2 分 exe 可在 Windows 上正常运行，功能正常。

附录：总评分表

序号	评分项	满分	得分
1	棋盘显示与交互界面	15	
2	落子功能与输入校验	15	
3	胜负判定算法	20	
4	悔棋功能	10	
5	保存/加载功能	10	
6	平局判断	5	
7	代码结构与风格	10	
8	程序鲁棒性	10	
9	打包为 exe	5	
总分		**100**	

成绩等级划分：

- **90-100 分**：功能完整，代码优秀，无 Bug，界面友好。
- **75-89 分**：功能基本完整，有小缺陷但核心功能正常，代码规范。
- **60-74 分**：主要功能能运行，部分功能有缺陷。
- **60 分以下**：无法运行、核心功能严重缺失或抄袭。

评阅教师： _____

评阅日期： _____