pj2 文档

顾开孚

2024 年 12 月 24 日

文档思考题

1 将 Player 结构体置于链表的结点中，在使用 malloc 开辟内存时会讲 Player 的内存一并连续开辟，若使用指向 Player 的指针，malloc 在开辟内存时开辟的是指针的大小，而不存在 Player 结构体的内存，且由于不存在 Player 结构体的内存，其指针没有意义。

2 简化重复代码；加强模块化，便于调试和维护；规定变量的作用域，优化内存。

3 将 NPC 的移动当作用户输入，在每一次存在用户输入改变界面时刷新界面。

4 结构化设计 UI 可以清晰明确不同 UI 在何种条件下显示，方便加入新的 UI 界面。但是当 UI 的顺序不固定，需要多次改变 UI 输出顺序时，使用非结构化设计可以方便代码编写。

说明文档

1 文件结构说明

本 pj 源文件除 main.c 外由四部分组成，其中 data.c 中存放了程序中与变量相关的函数，node.c 中存放了程序中与链表相关的函数，save.c 中存放了程序中与存档功能相关的函数，screen.c 中存放了程序中与 ui 界面相关的函数，并有同名头文件进行声明。另有 struct.h 中存放了程序中使用的所有结构体。

2 代码内容说明

2.1 struct

player_info 在本结构体中，x 和 y 代表了玩家在地图中的坐标，manual 是玩家本步会消耗的体力值，manual_spend 是玩家总计消耗的体力值，treasure_num 是地图上的宝藏总数，treasure_found 是玩家已找到的宝藏数，can_exit 是玩家是否能退出游戏的指示器，当 can_exit 为 0 时无法退出，can_exit 为 1 时退出游戏。

map_info 在本结构体中，map_height 和 map_width 记录了地图的高和宽，map_num 是数性地图表示，在游戏中不发生改变，用于进行地图字符化，和玩家移动后原位置状况判断，map_char 用于存放字符性地图表示，会随着玩家移动、宝藏获取等进行改变，并输出到屏幕上。

move_info 在本结构体中，input 用于接收用户的输入，ifcorrect 用于记录用户输入的合法性，column_move 和 row_move 分别记录用户输入后行列方向上的移动情况，moveline 用于在游戏结束后输出路径。

game_info 在本结构体中，mode 用于记录游戏以编程模式或调试模式运行，last_game 记录上一次游戏关卡，passed_game 记录已经通关的关卡（以二进制第一二三位分别代表一二三关），year、month、day、hour、min、sec 用于记录存档时间。

node 在本结构体中，*prev 和 *next 用于记录前后链表的地址，为撤销和恢复功能提供支持，player、map、move 结构体用于记录游戏过程。

2.2 data

initialization 本函数用作整个 pj 地图内容的初始化，其接受的变量 int i 作为关卡指示，node *head 是数据链表头节点的指针，用于存放初始化后的相关数据。mapimport 用于从文件中导入地图数据，然后使用 for 循环和 switch 语句将数性地图转换成字符性地图，最后将字符性地图玩家位置替换成 Y，清屏，退出函数。

iftreasure 本函数用作游戏中失去宝藏的判断，其接受的变量 node *last 是目前数据的地址。当玩家的上下左右存在 T（即宝藏）时，将 T 替换为空格占位，并调用 treasurechange 函数改变数据。当 player.treasure_found 与 player.treasure_num 相等，即宝藏全部被找到时，将 player.can_exit 赋为 1，退出游戏。

manualcal 本函数用作游戏中单步体力消耗计算，其接受的变量 node *last 是目前数据的地址。当玩家输入为 I，即原地不动时，跳过计算，若玩家当前位置数性地图数据为 2，即陷阱时，单步体力消耗会加一，反之将单步体力消耗重置为 1，然后将单步体力消耗加到总体力消耗上，退出函数。

mapimport 本函数用作游戏中将地图数据导入，其接受的变量 int i 标志关卡，node *head 是数据链表头节点的指针，存放导入的数据。使用 stdio.h 中的 FILE 类型，打开文件后通过 fscanf 导入地图的数据，后关闭文件，退出函数。

mapupdate 本函数用作游戏中地图的更新，其接受的变量 node *last 是目前数据的地址。若本步没有移动，则直接退出函数；若玩家原所在位置数性地图数据为 2，即陷阱，则将对应位置的字符性地图数据改为 D，然后将玩家位置数据改变，将玩家现在位置字符性地图数据改为 Y，退出函数。

movement 本函数用作游戏中移动逻辑实现，其接受的变量 node **last 指向目前数据的地址，由于会对 last 的地址做出改变，需要使用二级指针。Line:105 switch 进行移动逻辑判断，在非主动退出情况下嵌套 switch 进行输入是否合法的判断。然后在非退出游戏的情况下，依次调用 manualcal，mapupdate，iftreasure，退出函数。

treasurechange 本函数用作游戏中玩家获取宝箱数的更新，其接受的变量 node *last 是目前数据的地址，调用时将玩家的 treasure_found 加一，退出函数。

2.3 node

movenode 本函数用作游戏中节点移动的作用，其接受的变量 node **last 指向目前数据的地址。通过调用 nodecrete 建立新节点，给变量赋相应的值，调用 passinfo 将上一节点中部分数据转递至当前节点，退出函数。

nodecreate 本函数用作游戏中新节点的建立。通过 stdlib.h 中的 malloc 函数动态开辟指定长度的内存，返回开辟内存头字节的地址，退出函数。

passinfo 本函数用作游戏中两个节点数据的传递，其接受的变量 node *prev 和 node *now 是数据的传递方和接收方。其中传递了数性地图数据、字符性地图数据、地图高宽等数据，退出函数。

2.4 save

savefile 本函数用作游戏中存档保存，其接受的变量 node *last 和 game_info *game 接收当前数据地址和游戏数据地址。将各种数据输出至 save.txt 中，其中调用了 timesave 函数。

saveimport 本函数用作游戏中存档读取，其接受的变量 node *head 和 game_info *game 接收导入的数据。

timesave 本函数用作游戏中存档保存时保存当前时间，其接受的变量 FILE *file 是输出文件的地址，使用 time.h 的函数获取当前时间，并进行了处理后输出到文件中。

2.5 screen

beginscreen 本函数用作打印开始屏幕，其接受的变量 int i 和 game_info *game 是关卡指示和游戏数据地址。先清屏后通过 if 判断输出内容。

modescreen 本函数用作打印模式选择屏幕，其接受的变量 int i 屏幕指示。先清屏后通过 switch 判断输出内容。

movescreen 本函数用作屏幕移动实现,其接受的变量 int screen,game_info *game 和 player_info *player 用于指示屏幕选择，游戏数据和玩家数据。通过 switch 判断调用 beginscreen，modescreen，savescreen。Line:89 使用 ctype.h 中提供的 toupper 函数将玩家输入转化为大写，然后通过 if 判断确定移动方向，返回 screen_num 指示选择项。

savescreen 本函数用作打印读取存档屏幕，其接受的变量 int i，game_info *game 和 player_info *player 用于指示屏幕选择，游戏数据和玩家数据。先清屏后打印相应内容，Line:107 使用格式化输出%02d 确保分秒以两位输出。

printmap 本函数用作游戏中打印地图，其接受的变量 map_info *info 是地图数据地址。先清屏后使用两层循环打印字符性地图数据。

2.6 main

main.c 文件中仅引用了自定义头文件，标准库文件均在自定义头文件中已引用，故不再重复引用。

Line:9~Line:22 进行变量声明和赋初值。

Line:24 使用非格式化标签实现程序结束后的跳转。

Line:25~Line:29　进行存档导入。

Line:32~Line:34　打印开始屏幕，选择关卡或退出游戏。

Line:36~Line:51　若选择上次游玩，输出读取存档屏幕，若选择不读取或非上次游玩，打印模式选择屏幕，进行初始化；若选择读取，则使用存档数据进行游戏。

Line:53~Line:70　进行实时模式，打印地图，进行移动操作，调用对应函数。

Line:71~Line:96　进行编程模式，打印地图，进行移动操作，调用对应函数。使用外置文件 inputline.txt 缓存用户输入，接收到“\r”，即回车符后使用 fflush 将缓存区内容全部输出至文件中，然后使用 fseek 将光标移动至文件首，逐字符分析输入。

Line:97　编程模式输入错误跳出标签。

Line:98~Line:111　输出游戏结果。

Line:112~Line:128　记录通关情况。

Line:130　存档。

Line:132~Line:138　释放内存。

Line:139~Line:140　回到开始。

3　编译说明

由于本 pj 采用多文件编写，编译时需要将多个 c 文件链接编译，以下是两个编译方法。

3.1　makefile

在 cmd、powershell 或 vscode 终端等，将目录移至.\24302010003_pj2\src 中，运行“make”指令进行编译，“make clean”指令删除编译生成的.o 文件, 产生的 main.exe 文件位于.\24302010003_pj2\build 中。

3.2　手动编译

在 cmd、powershell 或 vscode 终端等，将目录移至.\24302010003_pj2\src 中，运行“gcc main.c data.c save.c node.c screen.c -o ..\\build”，产生的 main.exe 文件位于.\24302010003_pj2\build 中。

4 使用说明

1 在编译和进行游戏时，请勿重命名源文件、头文件、地图文件等，勿改变文件结构和文件位置。

2 请以 GBK 编码进行编译和游戏，否则可能导致地图文件无法正确加载，屏幕乱码等问题。

运行截图

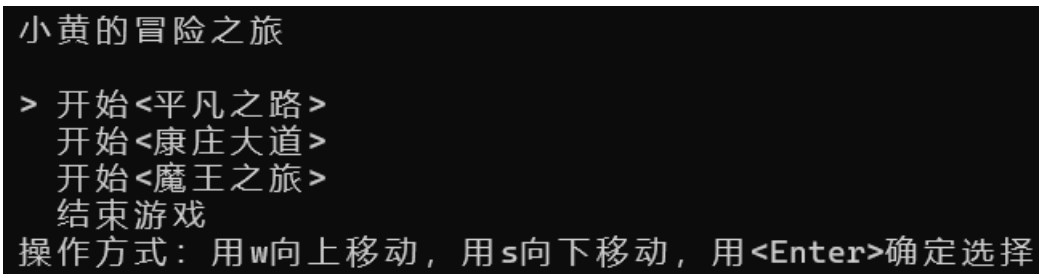

图 1: 开始界面

图 2: 模式选择界面

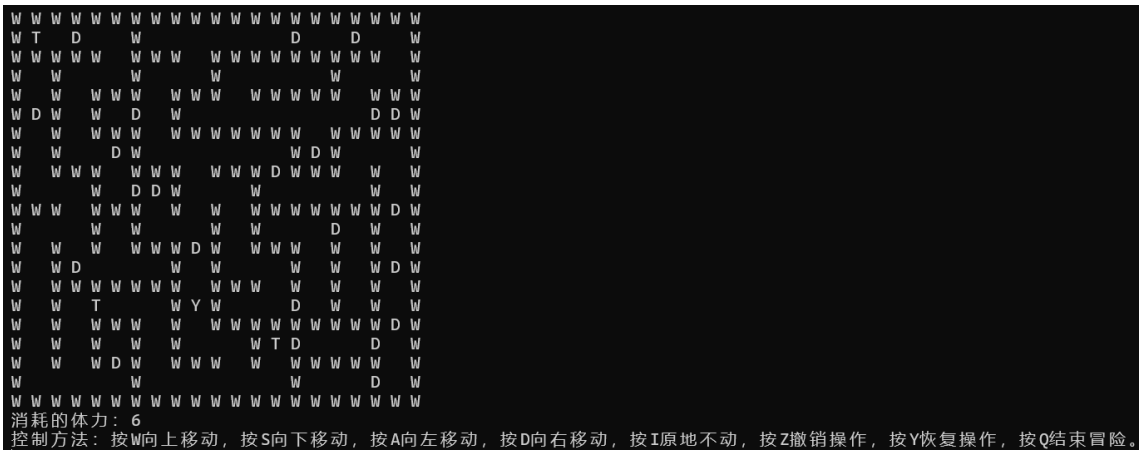

图 3: 游戏界面

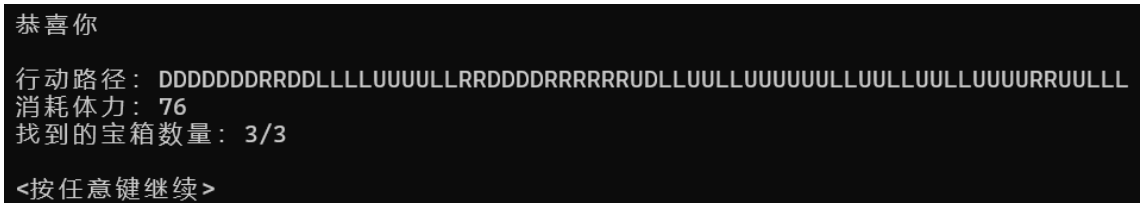

图 4: 结算界面

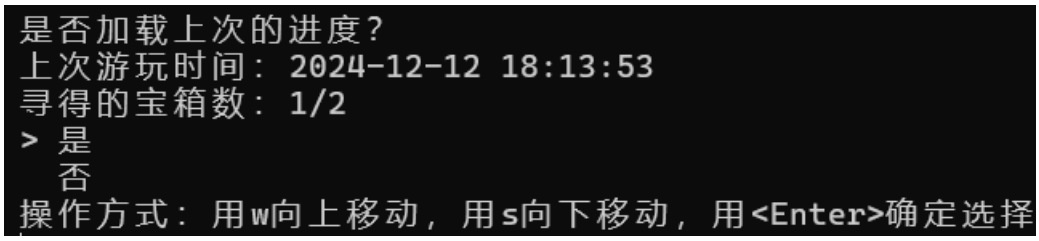

图 5: 存档界面