实验报告

一、 操作系统与 IDE

版本……Windows 11 家庭中文版

版本号……24H2

操作系统版本……26100.4061

IDE……Visual Studio 2022

二、 项目背景

在一个迷宫式的地图中，玩家控制一个小人，任务是将所有的箱子推到目标位置。地图由空地、墙壁、箱子和目标位置构成。玩家可以通过移动与推箱子，调整其位置，使得所有箱子都被推到目标位置。

三、 项目目标实现

1、 欢迎与菜单界面

```
//游戏开始选择函数
void choose() {
    printf("################################################################################\n");
    printf("##                                 欢迎!                                      ##\n");
    printf("##                              推箱子游戏                                    ##\n");
    printf("################################################################################\n\n");
    printf("是否开始游戏（输入“y”表示“是”，输入“n”表示“否”）:");
    char choice[2];
        fgets(choice, 2, stdin);
        char c;
    while ((c = getchar()) != '\n' && c != EOF);//清理缓冲区\n，避免main函数中直接读取"\n"导致直接退出循环
    if (choice[0] == 'n') {
    printf("游戏结束。\n");
        exit(1);
    }
    else if (choice[0] == 'y') {
        printf("**********************************\n");
        printf("*******     游戏开始!      ******\n");
        printf("*******    #:     墙壁     ******\n");
        printf("*******    $:     箱子     ******\n");
        printf("*******    .:     目标点   ******\n");
        printf("*******    @:     玩家     ******\n");
        printf("*******   空格:   空地     ******\n");
        printf("**********************************\n");

        return;
    }
    else {
        printf("输入错误，请重新输入：");
    }

}
```

选择开始游戏，则

```
##############################################################################
##                                欢迎！                                    ##
##                              推箱子游戏                                  ##
##############################################################################

是否开始游戏（输入“y”表示“是”，输入“n”表示“否”):y
********************************
*******     游戏开始！      ******
*******    #:     墙壁     ******
*******    $:     箱子     ******
*******    .:     目标点   ******
*******    @:     玩家     ******
*******   空格:   空地     ******
********************************
```

选择不开始游戏，则

```
##############################################################################
##                                欢迎！                                    ##
##                              推箱子游戏                                  ##
##############################################################################

是否开始游戏（输入“y”表示“是”，输入“n”表示“否”):n
游戏结束。
```

2、 绘制地图

以读取文档的方式读取地图数组，代码如下：

```
FILE* file;//地图文件
FILE* file1;//体力文件
FILE* fp;//地图存档
FILE* fp1;//体力存档

void game(int map_number) {

    int count = 0;//体力值初始化
    // 选择文件
    if (map_number == 1) {
        file = fopen("map1.txt", "r");
    }
    if (map_number == 2) {
        file = fopen("map2.txt", "r");
        printf("限制体力为：60\n");
    }
    if (map_number == 3) {
        file = fopen("record.txt", "r");
        file1 = fopen("count.txt", "r");
        fscanf(file1, "%d", &count);
    }

    int rows, cols;
    fscanf(file, "%d %d", &rows, &cols);


    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            fscanf(file, "%d", &map[i][j]);
        }
    }
    fclose(file);
```

绘制地图函数

```
for (int i = 0;i < rows;i++) {
    for (int j = 0;j < cols;j++) {
        switch (map[i][j]) {
        case 0: printf(" "); break;
        case 1: printf("#"); break;
        case 2: printf("$"); break;
        case 3: printf("."); break;
        case 4: printf("@"); break;
        case 5: printf("*"); break;
        }
    }
    printf("\n");
}
```

“map1.txt”文件内容及打印出的地图：

```
8 8
1 1 1 1 1 1 1 1
1 3 0 0 0 0 1 1
1 0 1 0 0 0 0 1
1 0 1 0 0 1 1 1
1 0 1 1 0 0 1 1
1 0 1 0 2 0 0 1
1 1 4 0 0 1 0 1
1 1 1 1 1 1 1 1
```

```
#########
#.     ##
# #     #
# #  ###
# ##  ##
# # $  #
##@  # #
#########
```

“map2.txt”文件内容及打印出的地图：

```
8 16
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 3 0 0 1 1 0 0 0 0 1 0 0 0 1
1 1 0 0 0 0 0 0 0 1 1 0 3 1 0 1
1 0 0 0 1 0 1 2 0 2 0 0 0 0 0 1
1 4 1 0 0 0 0 0 0 0 0 1 0 1 1 1
1 1 0 0 2 0 0 1 0 0 0 0 0 0 0 1
1 0 1 0 0 0 1 3 0 0 0 0 1 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
################
# .  ##    #   #
##       ## .# #
#   # #$ $     #
#@#        # ###
##  $  #       #
# #   #.    #  #
################
```

3、 加载关卡

```
printf("————————————————————————————————————————————\n");
printf("请输入你选择的关卡（输入1或2）：\n");
printf("读取存档文件（输入3）：\n");
printf("————————————————————————————————————————————\n");
char game_level[2];
fgets(game_level, 2, stdin);
char c;
while ((c = getchar()) != '\n' && c != EOF);//清理缓冲区\n，避免MAP1函数中直接读取"\n"导致第一个操作无法正确执行
if (game_level[0] == '1') {
    game(1);//选择1，则执行关卡1，map1.txt
}
if (game_level[0] == '2') {
    game(2);//选择2，则执行关卡2，map2.txt
}
if (game_level[0] == '3') {
    game(3);//选择3，则读取游戏存档文件record.txt,恢复存档状态
}
```

设置 3 个关卡，分别为关卡 1，关卡 2，以及游戏存档后读档，通过输入 1/2/3 来选择。选择后，执行函数 game()，其中包括加载地图及玩家按键操作。

4、 实现玩家的移动逻辑（上下左右）

首先，查找玩家位置，确定移动的起始点

```
//找玩家@(4)在哪里
int i, j;
for (i = 0;i < rows;i++) {
    for (j = 0;j < cols;j++) {
        if (map[i][j] == 4) {
            break;
        }
    }
    if (map[i][j] == 4) {
        break;
    }
}
//i和j表示人的坐标
```

通过 switch 语句确定玩家移动逻辑，定义 route 数组，记录玩家移动

上 W：

```
if (map[i - 1][j] == 0) { //人上方是空地
    map[i][j] -= 4;
    map[i - 1][j] += 4;
    count++;
}
if (map[i - 1][j] == 2 && map[i - 2][j] == 0) { //人上方是箱子，箱子上方是空地
    map[i][j] -= 4;
    map[i - 1][j] -= 2;
    map[i - 1][j] += 4;
    map[i - 2][j] += 2;
    count++;
}
if (map[i - 1][j] == 1 || (map[i - 1][j] == 2 && map[i - 2][j] == 1)) { //人上方是墙或人上方是箱子，箱子上方是墙
    printf("\n");
    printf("! ! ! ! ! ! ! ! ! ! ! ! ! ! ! \n");
    printf("!    前面是墙，不能前进!    ! \n");
    printf("! ! ! ! ! ! ! ! ! ! ! ! ! ! ! \n");
    printf("\n");
}
if (map[i - 1][j] == 2 && map[i - 2][j] == 3) { //人上方是箱子，箱子上方是目标点
    map[i][j] -= 4;
    map[i - 1][j] -= 2;
    map[i - 1][j] += 4;
    map[i - 2][j] -= 3;
    map[i - 2][j] += 5;
    count++;
}
route[count - 1] = 'W';
break;
```

下 S：

```
if (map[i + 1][j] == 0) { //人下方是空地
    map[i][j] -= 4;
    map[i + 1][j] += 4;
    count++;
}
if (map[i + 1][j] == 2 && map[i + 2][j] == 0) { //人下方是箱子，箱子下方是空地
    map[i][j] -= 4;
    map[i + 1][j] -= 2;
    map[i + 1][j] += 4;
    map[i + 2][j] += 2;
    count++;
}
if (map[i + 1][j] == 1 || (map[i + 1][j] == 2 && map[i + 2][j] == 1)) { //人下方是墙或人下方是箱子，箱子下方是墙
    printf("\n");
    printf("! ! ! ! ! ! ! ! ! ! ! ! ! ! ! \n");
    printf("!    前面是墙，不能前进!    ! \n");
    printf("! ! ! ! ! ! ! ! ! ! ! ! ! ! ! \n");
    printf("\n");
}
if (map[i + 1][j] == 2 && map[i + 2][j] == 3) { //人下方是箱子，箱子下方是目标点
    map[i][j] -= 4;
    map[i + 1][j] -= 2;
    map[i + 1][j] += 4;
    map[i + 2][j] -= 3;
    map[i + 2][j] += 5;
    count++;
}
route[count - 1] = 'S';
break;
```

左 A：

```
if (map[i][j - 1] == 0) { //人左边是空地
    map[i][j] -= 4;
    map[i][j - 1] += 4;
    count++;
}
if (map[i][j - 1] == 2 && map[i][j - 2] == 0) { //人左边是箱子，箱子左边是空地
    map[i][j] -= 4;
    map[i][j - 1] -= 2;
    map[i][j - 1] += 4;
    map[i][j - 2] += 2;
    count++;
}
if (map[i][j - 1] == 1 || (map[i][j - 1] == 2 && map[i][j - 2] == 1)) { //人左边是墙或人左边是箱子，箱子左边是墙
    printf("\n");
    printf("! ! ! ! ! ! ! ! ! ! ! ! ! ! ! \n");
    printf("!    前面是墙，不能前进!    ! \n");
    printf("! ! ! ! ! ! ! ! ! ! ! ! ! ! ! \n");
    printf("\n");
}
if (map[i][j - 1] == 2 && map[i][j - 2] == 3) { //人左边是箱子，箱子左边是目标点
    map[i][j] -= 4;
    map[i][j - 1] -= 2;
    map[i][j - 1] += 4;
    map[i][j - 2] -= 3;
    map[i][j - 2] += 5;
    count++;
}
route[count - 1] = 'A';
break;
```

右 D：

```
if (map[i][j + 1] == 0) { //人右边是空地
    map[i][j] -= 4;
    map[i][j + 1] += 4;
    count++;
}
if (map[i][j + 1] == 2 && map[i][j + 2] == 0) { //人右边是箱子，箱子右边是空地
    map[i][j] -= 4;
    map[i][j + 1] -= 2;
    map[i][j + 1] += 4;
    map[i][j + 2] += 2;
    count++;
}
if (map[i][j + 1] == 1 || (map[i][j + 1] == 2 && map[i][j + 2] == 1)) { //人右边是墙或人右边是箱子，箱子右边是墙
    printf("\n");
    printf("! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! \n");
    printf("!    前面是墙，不能前进！    ! \n");
    printf("! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! \n");
    printf("\n");
}
if (map[i][j + 1] == 2 && map[i][j + 2] == 3) { //人左边是箱子，箱子左边是目标点
    map[i][j] -= 4;
    map[i][j + 1] -= 2;
    map[i][j + 1] += 4;
    map[i][j + 2] -= 3;
    map[i][j + 2] += 5;
    count++;
}
route[count - 1] = 'D';
break;
```

设置需要退出游戏，则输入 K：

```
case ('k')://退出游戏
case ('K'):
    return 0;
```

输入指令错误提示：

```
default:
    printf("=======================================\n");
    printf("===  输入错误指令，请重新输入！！！  ===\n");
    printf("=======================================\n");
    break;
```

5、 判断游戏是否完成以及结束界面

```
//查找是否所有箱子都到达目标点
int find = 0;
for (i = 0;i < rows;i++) {
    for (j = 0;j < cols;j++) {
        if (map[i][j] != 3) {//地图中已无目标点
            find++;
        }
    }
}
if (find == rows * cols) {
    printf("**********************************\n");
    printf("***    恭喜你，关卡挑战成功！    ***\n");
    printf("***        消耗体力为：%d        ***\n", count);
    printf("路径为：");
    for (int i = 0;i < count;i++) {
        printf("%c ", route[i]);
    }
    printf("\n");
    printf("**********************************\n");
    break;

}
```

通过查找地图中是否有剩余目标点判断游戏是否成功。由于玩家将箱子($)推至目标点后，设置显示为“*”，遍历地图数组，如果没有目标点"."(3)，则判定为游戏成功，并显示游戏成功界面。

若玩家体力耗尽（即体力 count>60），则显示“体力耗尽游戏失败！”界面。

```
//体力限制
if (count > 60) {
    printf("########################");
    printf("# 体力耗尽，游戏失败！#");
    printf("########################");
    return 0;
}
```

6、 撤销功能

利用“栈”先进后出，后进先出的特点，将玩家执行每一步操作前的状态存在栈中，保存的状态为玩家执行当前按键操作时，该方向上包含玩家位置在内的、依次三个位置。

定义过程

```
struct position {
    int i;
    int j;
    int value;
};//三个信息i,j,map[i][j]

struct role {
    struct position pos[3];//
};
//定义栈
struct role stack[100];
int top = -1;//栈顶标记-1

//入栈操作
void push(struct position array[]) {
    top++;
    stack[top].pos[0] = array[0];
    stack[top].pos[1] = array[1];
    stack[top].pos[2] = array[2];
}
```

上 W：

```
pos[0].i = i;
pos[0].j = j;
pos[0].value = map[i][j];

pos[1].i = i - 1;
pos[1].j = j;
pos[1].value = map[i - 1][j];

pos[2].i = i - 2;
pos[2].j = j;
pos[2].value = map[i - 2][j];

push(pos);
```

下 S：

```
pos[0].i = i;
pos[0].j = j;
pos[0].value = map[i][j];

pos[1].i = i;
pos[1].j = j - 1;
pos[1].value = map[i][j - 1];

pos[2].i = i;
pos[2].j = j - 2;
pos[2].value = map[i][j - 2];

push(pos);
```

左 A：

```
pos[0].i = i;
pos[0].j = j;
pos[0].value = map[i][j];

pos[1].i = i + 1;
pos[1].j = j;
pos[1].value = map[i + 1][j];

pos[2].i = i + 2;
pos[2].j = j;
pos[2].value = map[i + 2][j];

push(pos);
```

右 D：

```
pos[0].i = i;
pos[0].j = j;
pos[0].value = map[i][j];

pos[1].i = i;
pos[1].j = j + 1;
pos[1].value = map[i][j + 1];

pos[2].i = i;
pos[2].j = j + 2;
pos[2].value = map[i][j + 2];

push(pos);
```

按键操作“z”时，将地图状态出栈，实现撤回步骤。

同时，由于步骤撤回，体力消耗减少 1。

```
case ('z'):
case ('Z'):
    if (top != -1) {
        map[stack[top].pos[0].i][stack[top].pos[0].j] = stack[top].pos[0].value;
        map[stack[top].pos[1].i][stack[top].pos[1].j] = stack[top].pos[1].value;
        map[stack[top].pos[2].i][stack[top].pos[2].j] = stack[top].pos[2].value;
        top--;
        count--;
    }
    break;
```

7、 游戏读档

设置按键操作时，按“P”可以将当前地图状态保存至“record.txt”文件，同时记录体力为“count.txt”文件，便于之后调用。

```
case('p')://游戏存档
case('P'):
    //记录地图
    fp = fopen("record.txt", "w+");
    fprintf(fp, "%d %d\n", rows, cols);

    for (int i = 0;i < rows;i++) {
        for (int j = 0;j < cols;j++) {
            fprintf(fp, "%d ", map[i][j]);
        }
        fprintf(fp, "\n");
    }
    fclose(fp);
    //记录体力
    fp1 = fopen("count.txt", "w+");

    fprintf(fp1, "%d", count);

    fclose(fp1);
    break;
```