编号：

设计报告文档成绩	
设计作品分数成绩	
课程设计成绩	

程序设计课程设计文档

题　　目： 和平岛（游戏）

姓　　名： 曾盛林

学　　号： 202224110203

专业班级： 计算机一班

2023 年 8 月 26 日

一、题目意义和设计思想

1、题目意义

■ 警示教育

《和平岛》（游戏）以两个海岛之间相互杀戮和冲突，双方两败俱伤的场景，向人们宣示：对抗中没有胜利者，战争只会给人类带来灾难，良知、鲜血和生命呼唤人类和平。游戏蕴含现实意义，契合学院老师对设计作品的目标要求。

■ 休闲娱乐，丰富人们精神生活

■ 提升自身业务素养

⑴ 检验巩固自己《高级语言程序设计》的学习成效，查漏补缺，对所学知识拓展、深化和系统化。

⑵ 培养和增强计算思维，锻炼自己在不同领域运用编程解决实际问题的能力。

⑶ 掌握 C 程序开发的全过程，熟悉 C 语言的语法规则，面向过程的编程思想。

⑷ 养成严谨的编程习惯和简洁优雅的程序风格。

⑸ 掌握文献查阅方法，提高学习能力。

2、设计思想

面向过程思想：

函数概况：主要通过 gameTwoPeople（游戏运行的统筹总函数）、

excute（玩家操作主函数）、attack（攻击主函数）、placeArm（出兵主函数）、placeBridge（架桥、拆桥主函数）及其附属函数实现了游戏的平稳运行。(五者结构关系详见：三、实现的主要功能和系统结构)

各核心函数及其附属函数：

(1) gameTwoPeople:游戏运行的总函数，用于统筹所有游戏执行函数。

① creatBackground:初始化岛屿、桥梁、玩家。

②decideChoice:执行掷色子模块，返还胜利玩家编号给gameTwoPeople。

(2) excute：对接 attack、placeArm、placeBridge 三大操作函数、作弊函数和“屯”模块，实现玩家游戏操作。

printIsland:展示游戏当前战况，重置图层。

(3) attack:接收胜利玩家编号。统筹攻击者、受袭击者检索、攻击执行以及核打击函数，实现攻击模块。

① sortAttacker:接收胜利玩家编号。根据中间岛所属权状态，分情况判断近战、远程兵种是否可用于攻击，将结果通过信息验证机制返还给 attack。

② sortVictim:接收受攻击岛屿编号、嘲讽状态。根据嘲讽状态

和球哥状态检索可受袭击者，检索结果通过信息验证机制返还给attack。

③ attackExcute:接收岛屿编号、位置编号、伤害数值。对相应位置单位执行伤害结算，判定单位是否死亡并执行相应的初始化程序。

④ nuclearStrike:接收胜利玩家编号。内置岛屿判断和交互选择，对编号岛屿上每个有单位的位置调用 attackExcute 执行秒杀伤害。

⑷ placeArm:统筹出兵岛屿检索和岛屿具体位置检索，将兵种数据附属于岛屿。核弹可单独调用 nuclearStrike 执行核打击。实现出兵模块。

① sortSoldierIsland:接收胜利玩家编号。进行普通兵种可放置岛屿检索并通过信息验证机制返还给 placeArm。

② sortHouseIsland:接收胜利玩家编号。进行房子可放置岛屿检索并通过信息验证机制返还给 placeArm。

③ sortPosition:接收岛屿编号。挨个检索并返回兵种在选定岛屿的放置位置给 placeArm。

④ houseSoldier：接收胜利玩家编号。调用 sortSoldierIsland 检索小兵可放置位置，执行自动出兵（可放弃）。

⑸ placeBridge:统筹架桥、拆桥检索函数，实现基础操作模块。

① sortBuildBridge:接收胜利玩家编号。检索可架的桥梁并通过信息验证机制返还给 placeBridge。

② sortBreakBride:接收胜利玩家编号。检索可拆的桥梁通过信息验证机制返还给 placeBridge。

(6) cheatCode:接收胜利玩家编号。输入对应指令，快速放置房子和核弹，加快游戏进度。

主函数设计：

```
int main() {
    void gameTwoPeople();                                          // 双人对战
    void peopleAi();                                               // 人机对战
    void openInstructions(const char* fileName, int x, int y);  // 输出文档

    Button startGame, close;
    int veri;
    IMAGE img;

    initgraph(750, 490, 1);
    while (1) {                        // 菜单选择
        loadimage(&img, "./picture/菜单背景.jpg", 750, 490);
        putimage(0, 0, &img);
        HWND hnd = GetHWnd();
        SetWindowText(hnd, "菜单");
        creatButton(&startGame, 110, 80, 150, 50, "和平岛");
        creatButton(&close, 480, 80, 150, 50, "关闭游戏");

        while (1) {                    // 判断点击
            veri = 0;
            if (peekmessage(&msg, EX_MOUSE)) {
                if (click(startGame)) {     // 进入和平岛
                    while (1) {
                        PlaySoundA(NULL, NULL, 0);          // 关闭音乐
                        initgraph(750, 490, 1);
                        HWND hnd = GetHWnd();
                        SetWindowText(hnd, "和平岛");
                        loadimage(&img, "./picture/和平岛背景.jpg", 750, 490);
                        putimage(0, 0, &img);
                        creatButton(&choice[0], 110, 80, 150, 50, "进入游戏");
                        creatButton(&choice[1], 480, 80, 150, 50, "游戏说明");
                        creatButton(&choice[2], 110, 300, 150, 50, "返回菜单");
                        creatButton(&choice[3], 480, 300, 150, 50, "关闭游戏");
                        while (1) {         // 判断点击
```

```
                    if (peekmessage(&msg, EX_MOUSE)) {
                        if (click(choice[0])) {                         // 进入游戏
                            gameTwoPeople();
                            break;
                        }
                        else if (click(choice[1])) {                    // 游戏说明
                            openInstructions("./游戏玩法介绍.txt", 0, 0);
                            initgraph(640, 480, 2);
                            break;
                        }
                        else if (click(choice[2])) {                    // 返回菜单
                            veri = 1;
                            break;
                        }
                        else if (click(choice[3])) {                    // 关闭游戏
                            exit(0);
                        }
                    }
                }
                if (veri)
                    break;
            }
            if (veri)
                break;
        }
        else if (click(close)) {        // 关闭游戏
            exit(0);
        }
    }
}
closegraph();
return 0;
}
```

主函数设计了两级菜单，首级菜单提供游戏入口，二级菜单中提供了游戏玩法介绍和双人模式入口，两者为以后的项目功能扩充和游戏更多模式扩展提供空间。同时，main.cpp 文件中预留 peopleAi 空函数，用于未来的人机对战开发。

二、采用的主要技术、遇到的难点和解决方法

1、主要技术

(1) C 语言面向过程，用于程序流程实现。

(2) C 文件流与输入输出流：使用 FILE 指针打开玩法介绍、彩蛋

的文本文档，输入输出实现游戏交互。

⑶ 宏定义：使用宏定义设置岛屿初始兵种单位上限，便于后期调整初始兵种上限，提高了程序的可维护性。

⑷ 图形化：调用 easyx 库进行图形化输出，提升玩家游戏体验。

⑸ 头文件：项目整体超过 1500 行、二十多个函数。编写 help.h 头文件总体定义了结构体，声明各.cpp 文件之间的主要调用函数和变量，采用#ifndef+#define+#endif 避免了重复编译，解决了不同.cpp 文件之间的声明重复而琐碎的问题。

2、遇到的难点和解决方法

⑴ 编译时遇到"multiple definition of ‘attack(int)’ ”等错误。

解决方法：阅读报错并查阅相关资料后，发现是因为编译时遇到了对同一函数的重定义，出现这种错误往往是因为存在.cpp 相互#include。对程序进行检查后，发现是因为我的游戏底层搭建.cpp 引用了 attack.cpp 文件，而 C 语言编译器会编译所有.cpp 文件，这造成了 attack.cpp 下的函数重复定义。通过查询 C 语言语法相关资料，我了解到头文件的运行逻辑，重新创建了 help.h 头文件集中处理函数和变量的声明，并使用了条件编译#ifndef+#define+#endif 规避重复编译，解决了函数重复定义的问题。

⑵ 在判断岛屿是否为空岛时，房屋需要被检索到，但房屋本身

又不占用岛屿单位空间。

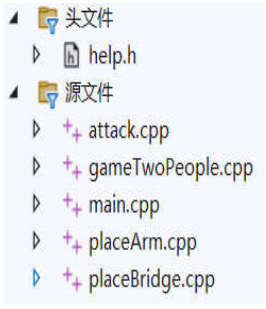
解决方法：在房屋存在时，额外进行岛屿单位上限+1，当房屋销毁时上限-1，这样等价于房屋没有占用岛屿单位。实现了房屋占位检索并不会影响其他兵种的正常岛屿单位。

(3) 放背景音乐时出现了无报错但无音乐的情况。

解决方法：我首先详细排查了相关库的链接和播放器状态，确认不是环境的问题。接着我四处搜寻资料，在一个论坛上面了解到可能是网易云音频的问题。网易云的音频自带封面，干扰了 PlaySoundA 的数据读取。我重新在 QQ 音乐上下载歌曲并进行格式转换后成功播放背景音乐。

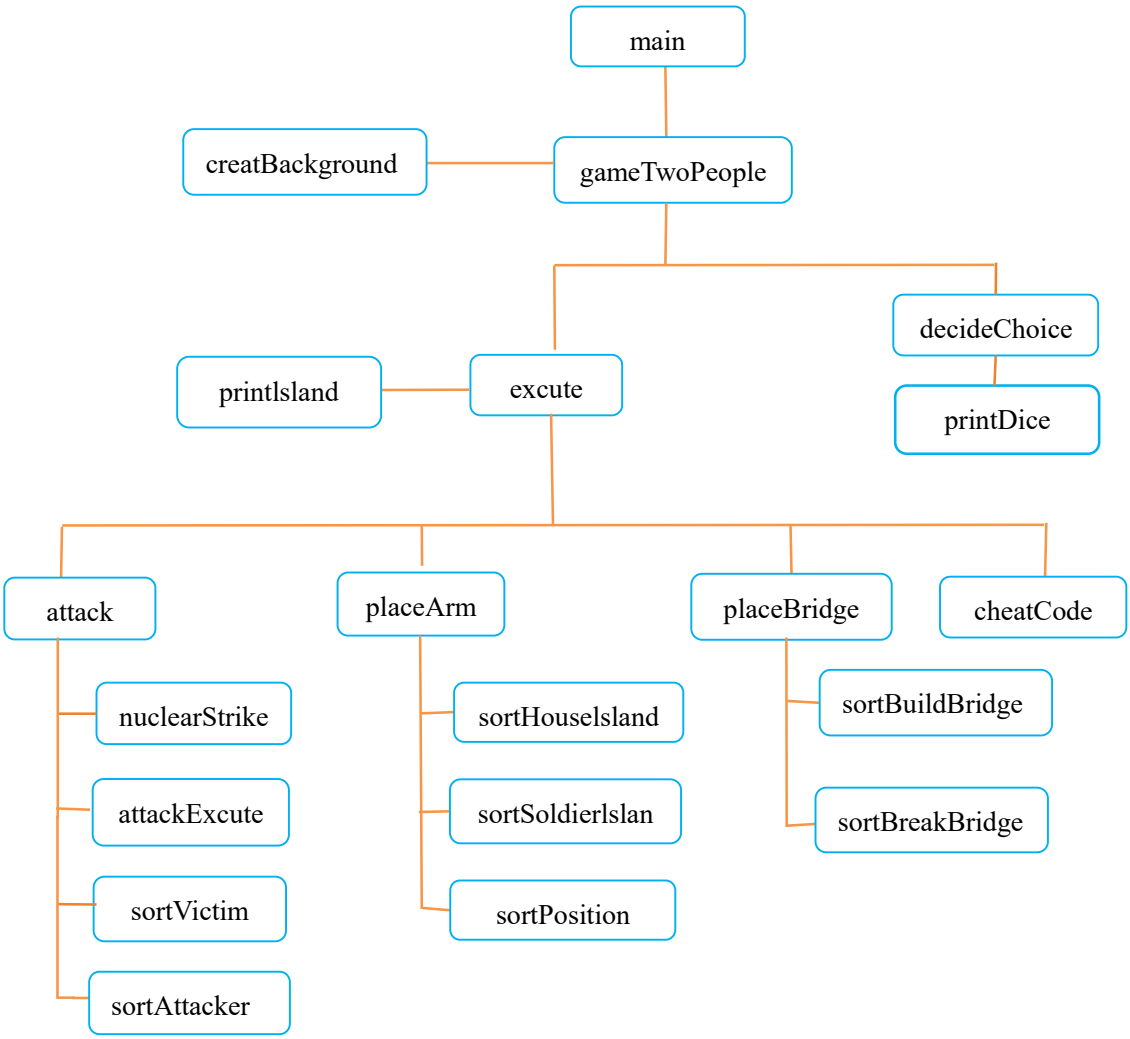
三、实现的主要功能和系统结构

1、源码架构



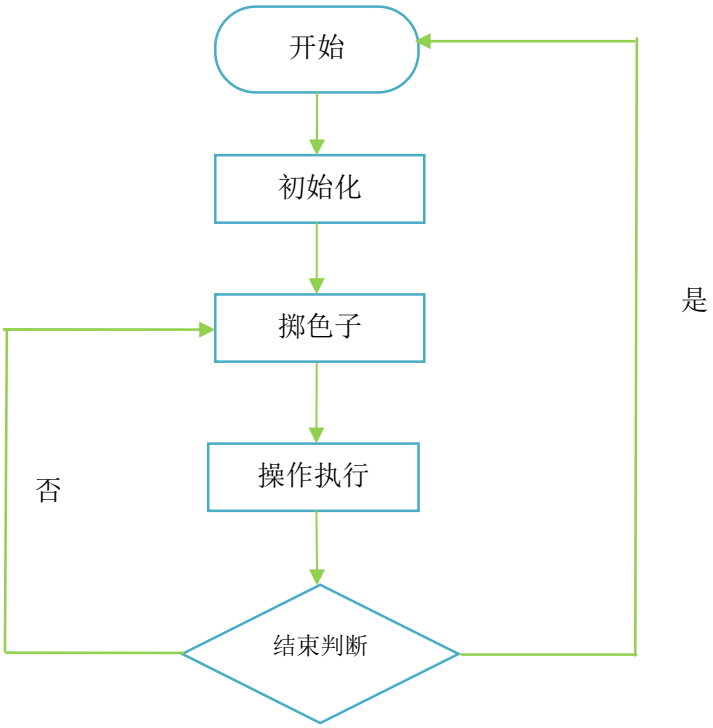
考虑到程序后期的开发和维护，源码采用分文件编写的方式。help.h 声明游戏所需的头文件、共同调用的函数和变量，定义岛屿等核心结构体。其余三大操作函数 attack、placeArm、placeBridge 及其相关函数分别构成对应的.cpp 文件。

2、主要函数调取关系



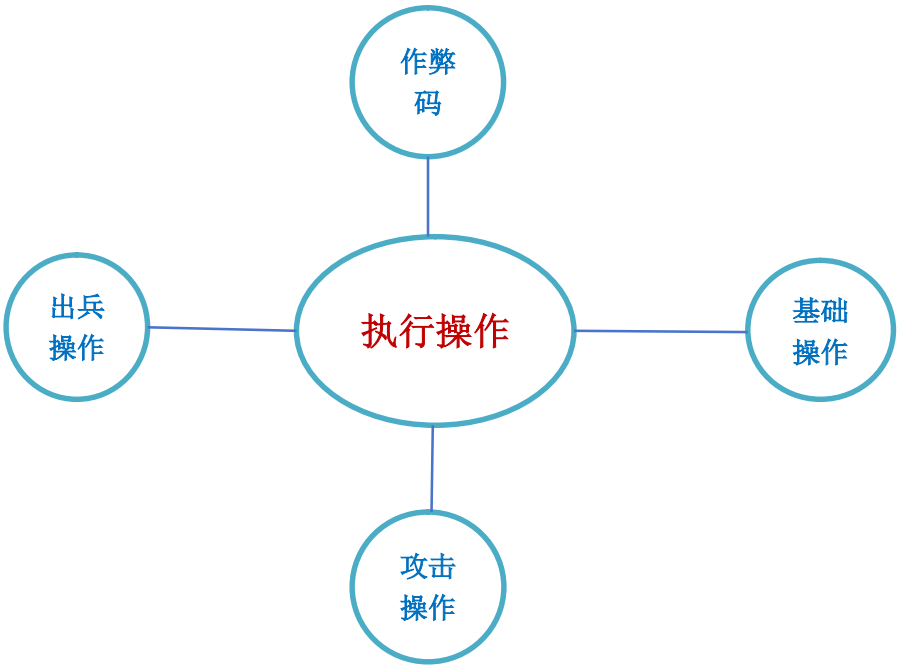
3、关键函数与游戏流程对照

(1) **gameTwoPeople**：统筹游戏所有函数，实现游戏基本框架：



其中 crearBackground、decideChoice 和 excute 分别为初始化、掷色子和操作执行的函数实现。

⑵ **excute:**统一调度三大操作函数和“屯”模块，扩充作弊码功能。



其中 attack、placeArm、placeBridge、cheatCode 分别为攻击操作、

出兵操作、基础操作和作弊码。

4、图形化

核心调用 easyx 库函数。同时，项目自定义了 createButton、clear、printIsland 等函数对按钮制造、操作页面初始化、岛屿战况输出等图形操作进行了集成封装。总体实现了较为优良的图形页面效果。

四、核心算法描述和相关技术说明

1、兵种设计

项目最初采用各类兵种对应的结构体，运用数组储存具体兵种单位，通过父类指针与岛屿挂钩（如图 1）。

图 1

```
typedef struct island_position{ //岛上单个位置属性
    all_pointer character;      //存放这个岛上的兵种
    int veri;                   //判断这个位置是什么兵
    char name[5];               //存放的兵的名字
}island_position;

typedef union all_pointer{
    caster* caster;
    cannon* cannon;
    house* house;
    fatty* fatty;
    minion* minion;
    stronger* stronger;
}all_pointer;

typedef struct island_all{      //岛的属性
    island_position main[11];   //3个主要位置
    int limit;                  //判断岛上最多放兵数量
    int num;                    //判断岛上已有几人
    int owner;                  //判断所属权
    int troop[5];               //判断兵种数：小，胖，
}island_all;

island_all island[3];           //设置三个岛
```

图 2

```
//创建 Caster 结构体实例的函数
Caster* create_Caster() {
    Caster* caster = (Caster*)malloc(sizeof(caster));
    strcpy(caster->name, "法师");
    caster->HP = 1;
    caster->attack = 3;
    caster->way = 0;
    return caster;
}

// 添加节点到链表尾部的函数（通用函数）
node* add_node(node** head, void* data) {
    node* new_node = (node*)malloc(sizeof(node));
    new_node->data = data;
    new_node->next = NULL;

    if (*head == NULL) {
        *head = new_node;
        return new_node;
    }
    node* current = *head;
    while (current->next != NULL) {
        current = current->next;
    }
    current->next = new_node;
    return new_node;
}
```

此处的父类指针用结构体+int 的方式实现，首先一个 struct 包含各兵种指针，在程序运行时用 veri（int 型）判断具体兵种。该方法时间复杂度低，代码可读性高。

但上述方法用数组储存兵种，空间复杂度太高，并有小概率出现数组溢出，程序不稳定。故而采用链式结构储存兵种，然后使用 void 型指针实现父类指针将岛屿单位与具体士兵挂钩（如图 2）。该方法大幅降低了空间复杂度，但节点的创建删除相对提高了时间复杂度。

为了从根本上优化程序流程，我放弃了通常的兵种信息储存。在岛屿单位结构体中增加 HP 成员变量，岛屿上直接进行伤害结算（如下图）。这样减掉了具体士兵结构体，提高了整套程序的时间效率和空间效率，大幅精简了代码，提高了程序质量，形成了最终版本。

```
typedef struct IslandPosition { // 岛上单个位置属性
    int HP;                     // 存放这个岛上的兵的血量
    int veri;                   // 判断这个位置是什么兵：小，胖，球，法，轻，房
} IslandPosition;

typedef struct IslandAll {      // 岛的属性
    IslandPosition main[10];    // 3个主要位置
    int limit;                  // 判断岛上最多放兵数量
    int num;                    // 判断岛上已有几人
    int owner;                  // 判断所属权
    int troop[5];               // 判断兵种数：小，胖，球，法，轻
} IslandAll;
```

2、信息反馈

该项目通过在检索函数中加入数组形参，记录正确信息反馈给上级函数。上级函数处理反馈信息，运用 createButton 函数产生对应的操作按钮，并通过 while 循环+peekmessage 检测点击实现页面按钮交互。

3、动态页面

游戏通过clear和printIsland两个函数实现对玩家的操作反应进行实时响应。clear通过背景图覆盖操作区实现感官上的动态页面，响应速度优良，用于一轮操作中途的动态页面实现；printIsland则对所有图层进行清空再重新绘制，可以避免clear造成的图层堆叠降低性能，但响应速度较慢，用于一轮操作后的总处理。

五、总结和体会

1、技术亮点

⑴ **打破思维惯性。**通常兵种类游戏采用面向对象的编程思想，通过类针对各兵种进行处理。然而在综合考量游戏体量、C语言程序特性后，我打破了兵种设计的思维惯性，将兵种直接依附于岛屿位置，精简了代码，同时大幅降低了程序的时间复杂度和空间复杂度。

⑵ **重视编程的规范性。**本项目代码书写规范，我遵从驼峰命名法的原则命名，结构体名每个单词首字母均大写，函数和变量名第一个单词全部小写，其他单词首字母大写；在代码架构上，我将所有结构体存放在了help.h 文件中，并在相应的cpp文件中实现三大操作模块的相关函数定义。编程的规范性使得程序的健壮性大大提升，为程序日后的拓展和维护带来了很大的便利。

⑶ **图形化编程。**我们以往的编程通常局限于字符模式。为了增强游戏体验，“和平岛”引入图形化编程，大幅提升了玩家的视觉感

受。同时，游戏通过详细的交互信息、Q 版的页面和人物设计给人以良好的游戏体验。

2、选题依据和对作品的自我评价

⑴ 以踮起脚尖摘桃子的高度确定目标。我确定的目标是：在保证质量的前提下，作品有 1000 余行语句的规模（实际有 1500 多行）；程序比较复杂，有一定难度；技术上有创新。为达到这个目标，我和多数同学一样，选择了游戏程序设计。理由是，游戏需要的程序语句多，比一般管理软件复杂；有创新空间，能满足学院老师对这次设计在作品规模、难度和创造性上的要求。同时，编程的本质就是模拟游戏，游戏程序设计方法和技术可移植到各种类型软件。

⑵ 以“工匠精神”雕刻《和平岛》。《和平岛》整体上精心构思，每一步精雕细刻。遇到难点，读教材，查文献，悟原理，上机测试找原因，凭自己独立思考找到解决方案，实现技术创新。全程时间，因为和历时 30 多天的学校机器人实验室入室考核（我在考核之列）重叠而无法计算。课程设计各阶段时间占比，纸上设计约占课程设计总时间的 20%，查资料思考问题占 30%，上机输入调试占 50%。

⑶《和平岛》是我的处女作和满意作品。《和平岛》架构设计合理，程序编写规范标准，简洁流畅。经反复测试，其稳定性、可延展性等各项功能、性能均有很好的表现，是一件设计有难度，程序规模较大，功能、性能优越的满意作品。

3、体会

程序设计是脑力、耐力和创造力的合成艺术，成果是“艺术品”。

程序设计时，要精心构思，认真探索，积极改进创新。本次程序设计中，我摆脱了兵种设计的思维惯性，将兵种直接依附于岛屿位置，大幅提高了程序的时间效率和空间效率。同时精简了代码，形成整套程序简洁、高效的风格，确保了软件整体质量。

分层实施时，要标准化编程，增强代码可读性，提高编程效率，减少不必要的时间浪费。要有铁杵磨成针的耐心，反复琢磨，逐步优化。要有细致入微的洞察力，快速锁定 bug 的逻辑源码。同时提前设想极端情况，提高测试效率。