四川大学期末考试试题（闭卷）

（2024—2025 学年第 2 学期）　　A 卷

课程号：304516030　课序号：01-07　课程名称：面向对象程序设计（C++篇）
任课教师：田星/陈良银/周佩/李琳/胡大裟　　成绩：
适用专业年级：　　学生人数：　　印题份数：　　学号：　　姓名：

考 生 承 诺

我已认真阅读并知晓《四川大学考场规则》和《四川大学本科学生考试违纪作弊处分规定（修订）》，郑重承诺：

1. 已按要求将考试禁止携带的文具用品或与考试有关的物品放置在指定地点；
2. 不带手机进入考场；
3. 考试期间遵守以上两项规定，若有违规行为，同意按照有关条款接受处理。

考生签名：

请按照题目编号在答题纸上书写答案，写在试卷上不得分。

按照要求与编号填写缺失代码。

一、背景

游戏在一个大洞穴中，由一个个房间组成，房间呈网格状排列(NxN)，每个房间都有四条隧道通向上下左右的房间，游戏随机为玩家指定一个空房间开始游戏。

二、游戏的 main 函数如下，

```
1. int main(int argc, char** argv) {
2.     int size = 5;  // 洞穴的大小 5 x 5 的房间
3.     Game g(  (1)  );  // 创建游戏对象
4.     g.run();  // 运行游戏
5.     return 0;
6. }
```

2.1. 游戏 Game 对象声明如下，它包含了一个两层的 vector 容器的 Room 对象指针存储游戏洞穴的房间。

```
1. class Game {
2.  (2)
3.         std::vector<std::vector<Room*>> rooms;  // 洞穴的房间
4.          (5)  int size_;  // 洞穴的大小
5.  (3)
6.         Game(int );  //游戏初始化
7.         ~Game();
8.         void run();  //游戏运行
9.  (4)
```

```
10.         void show_map();  //在字符界面显示地图
11.         void print_info();  //打印游戏信息
12.         void check_end();  //检查游戏是否结束
13. };
```

根据 C++的封装特性完善代码 **(1) (2) (3) (4)**（每题 2 分，共 8 分）。

游戏地图的尺寸在游戏开始后不会改变，请完成 **(5)** （每题 2 分，共 2 分）。

2.2. Game 对象的关键函数实现如下：

```
1.  void Game::run() {
2.      print_info();
3.      show_map();
4.      check_end();
5.  }
6.  void Game::show_map() { //在字符界面显示地图
7.    for (int i = 0; i < size_ * 2 + 1; i++) {
8.        for (int j = 0; j < size_ * 2 + 1; j++) {
9.            if (i%2 == 0 && j%2 == 0) {
10.               std::cout << '+';
11.           } else if (i%2 == 0 && j%2 != 0) {
12.               std::cout << "-";
13.           } else if (i%2 != 0 && j%2 == 0) {
14.               std::cout << "|";
15.           } else if (i%2 != 0 && j%2 != 0) {
16.               int x = (i-1)/2;
17.               int y = (j-1)/2;
18.               rooms[x][y]->show();
19.           }
20.       } std::cout << std::endl;
21.   }
22. }
23. Game::Game(int size) { //游戏初始化
24.     for(int i = 0; i < size; i++) {
25.         std::vector<Room*> tmp;
26.         for(int j = 0; j < size; j++) {
27.             tmp.push_back(new Room()); // 初始化每个房间
28.         } rooms.push_back(tmp); // 添加一行房间
29.     } size_ = size;
30. }
```

Game::Game(int)构造函数的代码有错误，请指出错误并改正 **(6)** 。 （本题 5 分）

Game::~Game()析构函数需要清除 vector 中所有的 Room 对象，请写出析构函数的代码。

(7) 。 （本题 8 分）

三、定义游戏的房间，由房间组成 NxN 洞穴地图。

3.1. 房间 Room 对象声明如下：

```
1. class Room {
2.   public:
3.     Room(); //缺省构造函数
4.     Room(const Room& other); //拷贝构造函数
5.     void show();
6. };
```

空房间在字符界面上显示空格，Room::show()函数实现如下：

```
1. void Room::show() {
2.     std::cout << " ";  //空房间显示空格
3. }
```

根据前述 Game::show_map()和 Room::show()的代码，画出游戏的房间。 **(8)** （至少画出两个相邻房间即可，本题 5 分）

四、现在有一个玩家进入了洞穴，他被随机传送到一个房间中，通过键盘操作可以在房间中移动。

4.1. 玩家 Player 对象声明与实现如下：

```
1. class Player
2. {
3. private:
4. public:
5.   Player() {}
6.   ~Player() {}
7. };
```

Player 对象没有具体的功能。但是 Player 对象的出现修改了 Game 和 Room 的定义。

4.2. Game 对象包含了一个 Player 对象，它是一个对象指针，在游戏开始时为玩家随机分配一个房间。同时 Game 对象控制玩家的输入与移动。Game 对象声明修改如下。

```
1. class Game {
2.     private:
3.         std::vector<std::vector<Room*>> rooms;  // 洞穴的房间
4.         Player* player_;  // 玩家
```

```
        const int size_;  // 洞穴的大小
        std::string direction_;  // 玩家的移动方向

        void random_rooms();
        void player_input(int& dx, int& dy); // 玩家输入
        void move_player();  // 移动玩家
        void print_info();
        void show_map();
        void check_end();
    public:
        Game(int size); //游戏初始化
        ~Game();

        //functions
        void run();
};
```

4.3. Game 对象的构造函数修改如下：

```
Game::Game(int size):size_(size) { //游戏初始化
    srand(time(NULL));  // 随机数种子
    for(int i = 0; i < size; i++) {
        std::vector<Room*> tmp;
        for(int j = 0; j < size; j++) {
            tmp.push_back(new Room()); // 初始化每个房间
        } rooms.push_back(tmp); // 添加一行房间
    }
    player_ = (9) // 初始化玩家
    random_rooms(); // 随机分配一个房间给玩家
}
```

Game::Game()构造函数创建一个 Player 对象，请完善代码 __(9)__。（本题 4 分）

4.4. 在游戏结束时，Game 对象需要清除 Player 对象，请实现 Game 对象的析构函数，注意游戏中只有一个玩家对象。 __(10)__ （本题 5 分）

4.5. Game::player_input()方法实现如下：

```
void Game::player_input(int& dx, int& dy) { // 玩家输入
    std::cout << "请输入你的移动方向: " ; // 提示输入
    std::cout << "w: 向上移动" ; // 向上移动
    std::cout << "s: 向下移动" ; // 向下移动
    std::cout << "a: 向左移动" ; // 向左移动
```

```
    std::cout << "d: 向右移动" << std::endl; // 向右移动
    std::getline(std::cin, direction_); // 获取玩家输入的方向
    if (direction_ == "w") { // 向上移动
        dy = 0; //
        dx -= 1; // 向上行移动
    } else if (direction_ == "s") { // 向下移动
        dy = 0; //
        dx += 1; // 向下行移动
    } else if (direction_ == "a") { // 向左移动
        dy -= 1; // 向左列移动
        dx = 0; //
    } else if (direction_ == "d") { // 向右移动
        dy += 1; // 向右列移动
        dx = 0; //
    } else { // 输入错误
  std::cout << "输入错误,请重新输入" << std::endl; // 提示输入错误
        return;
    }
}
```

4.6. Game::move_player()方法实现如下：

```
void Game::move_player() {
    int room_x = 0 , room_y = 0;
    player_input(room_x, room_y); // 获取玩家输入的方向
    for(int i = 0; i < size_; i++) { // 遍历所有房间
        for(int j = 0; j < size_; j++) {
           if (rooms[i][j]->is_occupied()) {//找到玩家所在的房间
                room_x += i; // 计算玩家所在的房间的坐标
                room_y += j;
                if (room_x < 0 || room_x >= size_ || room_y < 0 |
| room_y >= size_) { // 超出边界
                    return; // 返回
                }
              rooms[i][j]->leave(player_); // 离开房间
              rooms[room_x][room_y]->enter(player_); //进入房间
              break;
            }
        }
    }
}
```

上述代码遍历所有房间找到玩家所在房间，然后玩家离开该房间，根据玩家输入，再进入到旁边的房间。

4.7. Room 对象中包含了一个玩家对象指针，进入房间、离开房间和判断玩家所在的方法，同时 Room::show()方法在玩家存在时显示 *，在空房间里显示空格。Room 对象的声明修改如下：

```
1.  class Room {
2.      public:
3.          Room(); //缺省构造函数
4.          ~Room() {} //析构函数
5.
6.          void show(); //显示房间的信息
7.          void enter(Player* player); //玩家进入房间
8.          void leave(Player* player); //玩家离开房间
9.          bool is_occupied(); //判断房间是否有玩家
10.     private:
11.         Player* player_; //房间中的玩家
12. };
```

Room 对象初始化时都是没有玩家的空房间，请实现 Room 对象的构造函数。 (11)
（本题 6 分）

Room 对象只是参考 Player 对象，因此 Room 对象的析构函数不用删除 Player 对象。

五、现在游戏加入 2 个新的元素，一个是房间中出现了陷阱，玩家进入房间掉入陷阱并死亡，Game Over。另一个是房间中出现了黄金，玩家遍历地图中所有房间试图找到所有的黄金。

如此，游戏中的房间分别有空房间、玩家房间，陷阱房间和黄金房间，于是 Room 对象增加了静态方法用于分别创建不同的房间，Room::roomWithPlayer()创建游戏开始时玩家的随机房间，Room::roomWithPit()创建陷阱房间，Room::roomWithGold()创建黄金房间，Room::roomWithEmpty()创建空房间。

5.1. Game::random_rooms()调用这些静态方法以 50%概率创建特定房间，同时控制着游戏中陷阱房间与黄金房间的数量。

```
1.  void Game::random_rooms() {
2.      for(int i = 0; i < size_; i++) {
3.          std::vector<Room*> tmp;
4.          for(int j = 0; j < size_; j++) {
5.              if(rand() % 10 < 5 && player_ == nullptr) {
6.                  player_ = new Player(); // 创建玩家
7.                  tmp.push_back(Room::roomWithPlayer(player_));
8.              } else if(rand() % 10 < 5 && pits_ > 0) {
9.                  tmp.push_back(Room::roomWithPit()); //
10.                 pits_--; // 陷阱数量减一
```

```
            } else if(rand() % 10 < 5 && golds_ > 0) {
                tmp.push_back(Room::roomWithGold());
                 golds_ --;
            } else {
                tmp.push_back(Room::roomWithEmpty());
            }
        } rooms.push_back(tmp); // 添加一行房间
    }
}
```

根据 Game::random_rooms()实现代码，完成 Room 对象的静态方法声明。__(12)__（本题 5 分，Room 对象的其它声明代码可以不写）

玩家为了避免掉入陷阱，游戏加入了“感知”功能，在玩家进入房间之前，可以先感知一下房间的情况，再决定是否进入房间。

5.2. Game::run()的实现代码如下：

```
void Game::run() {
    while (true) { // 游戏循环
        print_info();
        percept_rooms();  // 感知房间
        show_map();
        move_player();
        check_end();
    }
}
```

Game::percept_rooms()方法根据玩家所在房间感知周围的房间。

```
void Game::percept_rooms() { // 感知房间

    for (int i = 0; i < size_; i++) { // 遍历房间

        for (int j = 0; j < size_; j++) { // 遍历房间
            if (rooms[i][j]->is_occupied()) {
                if (i - 1 >= 0) {
                    std::cout << "Up Room:";
                    rooms[i - 1][j]->percept();
                } if (i + 1 < size_) {
                    std::cout << "Down Room:";
                    rooms[i + 1][j]->percept();
                } if (j - 1 >= 0) {
```

```
                    std::cout << "Left Room:";
                    rooms[i][j - 1]->percept();
                } if (j + 1 < size_) {
                    std::cout << "Right Room:";
                    rooms[i][j + 1]->percept();
                }
            }
        }
    }
}
```

Room::percept()可以感知房间中是陷阱还是黄金，或者是空房间。陷阱房间会发出飕飕的阴风，黄金房间会发出金光闪闪，空房间则感觉阴森森湿漉漉的。

```
void Room::percept() { //感知房间
    if (event_!= nullptr) {
        event_->percept();
    } else { // 如果空房间
        std::cout << "你感觉湿漉漉的！" << std::endl;
    }
}
```

Room 对象包含了一个感知事件对象。

```
class Room {
    public:
        Room(); //缺省构造函数
        Room(const Room& other); //拷贝构造函数
        ~Room();
        // …… 省略其它声明
    private:
        Player* player_;  // 玩家对象指针
        Event* event_;  // 感知对象指针
};
```

Event 是一个抽象基类，有一个 percept()的纯虚函数，请写出 Event 类的声明。 **(13)** **（本题 5 分）**

空房间，黄金房间，陷阱房间分别继承 Event 类，分别实现 percept()方法，输出不同房间的提示信息。请写出黄金房间的声明与实现。 **(14)** **（本题 10 分）**

Room 对象提供了不同的静态方法创建房间，请实现黄金房间的创建方法

Room::roomWithGold()。 __(15)__ **（本题 5 分）**

六、如果玩家进入陷阱房间，玩家死亡，游戏重启，玩家重新出现在一个随机的房间中。

6.1. Event 类添加了一个死亡方法 Event::die()，由派生类房间实现，空房间和黄金房间不会导致玩家死亡，陷阱房间会导致玩家死亡。请重构 Event 类的声明。 __(16)__ **（本题 6 分）**

然后 Pit 类继承 Event 类。请实现陷阱 Pit 类中实现 Pit::die()方法。 __(17)__ **（本题 10 分）**

6.2. 现在玩家进入房间，会触发 Event::die()方法，抛出异常。

请在 Room::enter()方法中检查 Event::die()方法的结果，抛出 std::runtime_error("You have died!")。 __(18)__ **（本题 10 分）**

请在 Game::run()中捕获异常，提示玩家游戏重启。

```
1.  void Game::run() {
2.      while (true) { // 游戏循环
3.          __(19)__
4.              print_info();
5.              percept_rooms();  // 感知房间
6.              show_map();
7.              move_player();
8.              check_end();
9.          __(20)__
10.             __(21)__
11.             for (auto &row : rooms) { // 遍历房间
12.                 for (auto &room : row) {
```

```
                    delete room; // 删除房间
                }
            }
            rooms.clear();      // 清空房间
            random_rooms(); // 重新生成房间
        }
    }
}
```

请完成 **(19) (20) (21)** **（本题每空 2 分，共 6 分）**