**战士游戏项目报告**

**引言**

本项目实现了一个简单的战士对战游戏，玩家可以选择不同的行动，如攻击、防御或使用烧血机制来增强攻击力。

**功能描述**

本游戏包含两个主要类：Weapon和Warrior。每个战士都有自己的武器，并可以进行攻击、防御和烧血攻击。

**类定义**

**Weapon 类**

* **属性:**
  + attackMin: 最小攻击力
  + attackMax: 最大攻击力
  + wear: 磨损度
* **方法:**
  + 构造函数: 初始化武器属性
  + attack(): 返回武器的实际攻击力，考虑磨损度
  + display(): 显示武器信息

**Warrior 类**

* **属性:**
  + name: 战士名字
  + attackValue: 战士基础攻击力
  + defenseValue: 战士防御力
  + health: 战士生命值
  + defending: 是否处于防御状态
  + weapon: 武器对象
  + burnDamage: 烧血增加的伤害
* **方法:**
  + 构造函数: 初始化战士属性
  + useWeapon(Warrior &opponent): 使用武器攻击对手
  + defend(): 设置防御状态
  + takeDamage(int damage): 计算并承受伤害
  + resetDefense(): 重置防御状态
  + isAlive(): 检查战士是否还活着
  + display(): 显示战士信息
  + play(Warrior &opponent): 玩家选择行动

**主要功能介绍**

1. **普通攻击**：玩家选择攻击后，计算武器攻击力并对对手造成伤害。
2. **防御**：玩家选择防御后，减少所受伤害。
3. **烧血攻击**：玩家选择烧血攻击后，输入要消耗的血量，增加的伤害为所烧血量的2倍。

**完整代码如下：**

#include <iostream>  // 引入输入输出流库

#include <string>    // 引入字符串库

#include <cstdlib>   // 引入C标准库，包含rand()等函数

#include <ctime>     // 引入时间库，包含time()等函数

// 定义武器类

class Weapon {

private:

    int attackMin;  // 最小攻击力

    int attackMax;  // 最大攻击力

    int wear;       // 磨损度

public:

    // 构造函数，初始化武器的最小攻击力、最大攻击力和磨损度

    Weapon(int min, int max, int wear) : attackMin(min), attackMax(max), wear(wear) {}

    // 拷贝构造函数，用于创建一个新的武器，其属性与传入的武器相同

    Weapon(const Weapon &other) : attackMin(other.attackMin), attackMax(other.attackMax), wear(other.wear) {}

    // 析构函数

    ~Weapon() {}

    // 攻击函数，返回攻击力，攻击力为最小攻击力和最大攻击力之间的随机值，受磨损度影响

    int attack() {

        int attackValue = attackMin + (std::rand() % (attackMax - attackMin + 1));

        int weartemp = wear;

        wear += 5;  // 每次攻击后磨损度增加5

        return attackValue \* (100 - weartemp) / 100;

    }

    // 显示武器信息

    void display() const {

        std::cout << "Weapon[Min: " << attackMin << ", Max: " << attackMax << ", Wear: " << wear << "]" << std::endl;

    }

};

// 定义战士类

class Warrior {

private:

    std::string name;  // 名字

    int attackValue;   // 攻击力

    int defenseValue;  // 防御力

    int health;        // 生命值

    bool defending;    // 是否正在防御

    Weapon weapon;     // 武器

    int burnDamage;    // 烧血增加的伤害

public:

    // 构造函数，初始化战士的名字、攻击力、防御力、生命值和武器

    Warrior(std::string name, int attack, int defense, int health, Weapon weapon)

        : name(name), attackValue(attack), defenseValue(defense), health(health), defending(false), weapon(weapon), burnDamage(0) {}

    // 拷贝构造函数，用于创建一个新的战士，其属性与传入的战士相同

    Warrior(const Warrior &other)

        : name(other.name), attackValue(other.attackValue), defenseValue(other.defenseValue),

          health(other.health), defending(other.defending), weapon(other.weapon), burnDamage(other.burnDamage) {}

    // 析构函数

    ~Warrior() {}

    // 使用武器攻击对手

    void useWeapon(Warrior &opponent) {

        int attackPoints = weapon.attack() \* attackValue / 100; // 计算攻击力

        attackPoints += burnDamage; // 增加烧血造成的额外伤害

        burnDamage = 0; // 重置烧血增加的伤害

        opponent.takeDamage(attackPoints); // 对手受到伤害

    }

    // 防御

    void defend() {

        defending = true;

    }

    // 受到伤害

    void takeDamage(int damage) {

        if (defending) {

            damage = damage \* (100 - defenseValue) / 100;

        }

        health -= damage;

        if (health < 0) health = 0;

    }

    // 重置防御状态

    void resetDefense() {

        defending = false;

    }

    // 判断战士是否还活着

    bool isAlive() const {

        return health > 0;

    }

    // 显示战士信息

    void display() const {

        std::cout << "Warrior[Name: " << name << ", Attack: " << attackValue << ", Defense: " << defenseValue

                  << ", Health: " << health << ", Defending: " << (defending ? "Yes" : "No") << "]" << std::endl;

        weapon.display();

    }

    // 玩家行动，可以选择攻击或防御

    void play(Warrior &opponent) {

        int choice;

        std::cout << name << ", choose your action (1 - Attack, 2 - Defend, 3 - Burn Blood): ";

        std::cin >> choice;

        switch (choice) {

            case 1:

                useWeapon(opponent);

                break;

            case 2:

                defend();

                break;

            case 3:

                int bloodBurn;

                std::cout << "Enter the amount of health to burn: ";

                std::cin >> bloodBurn;

                if (bloodBurn >= health) {

                    bloodBurn = health - 1; // 确保不会烧死自己

                }

                health -= bloodBurn;

                burnDamage = bloodBurn \* 2; // 烧血增加的伤害为所烧血量的2倍

                std::cout << name << " burned " << bloodBurn << " health points for increased damage!" << std::endl;

                useWeapon(opponent);

                break;

            default:

                std::cout << "Invalid choice. Defaulting to attack." << std::endl;

                useWeapon(opponent);

                break;

        }

    }

};

int main() {

    std::srand(std::time(0));  // 初始化随机数种子

    Weapon sword(10, 20, 5);  // 创建剑

    Weapon axe(15, 25, 7);    // 创建斧头

    Warrior warrior1("Warrior1", 60, 40, 100, sword);  // 创建战士1

    Warrior warrior2("Warrior2", 50, 50, 100, axe);    // 创建战士2

    // 当两个战士都还活着时，进行战斗

    while (warrior1.isAlive() && warrior2.isAlive()) {

        warrior1.display();

        warrior2.display();

        warrior1.play(warrior2);

        warrior1.resetDefense();

        if (warrior2.isAlive()) {

            warrior2.play(warrior1);

            warrior2.resetDefense();

        }

        std::cout << std::endl;

    }

    // 判断谁赢了

    if (warrior1.isAlive()) {

        std::cout << "Warrior1 wins!" << std::endl;

    } else if (warrior2.isAlive()) {

        std::cout << "Warrior2 wins!" << std::endl;

    }

}