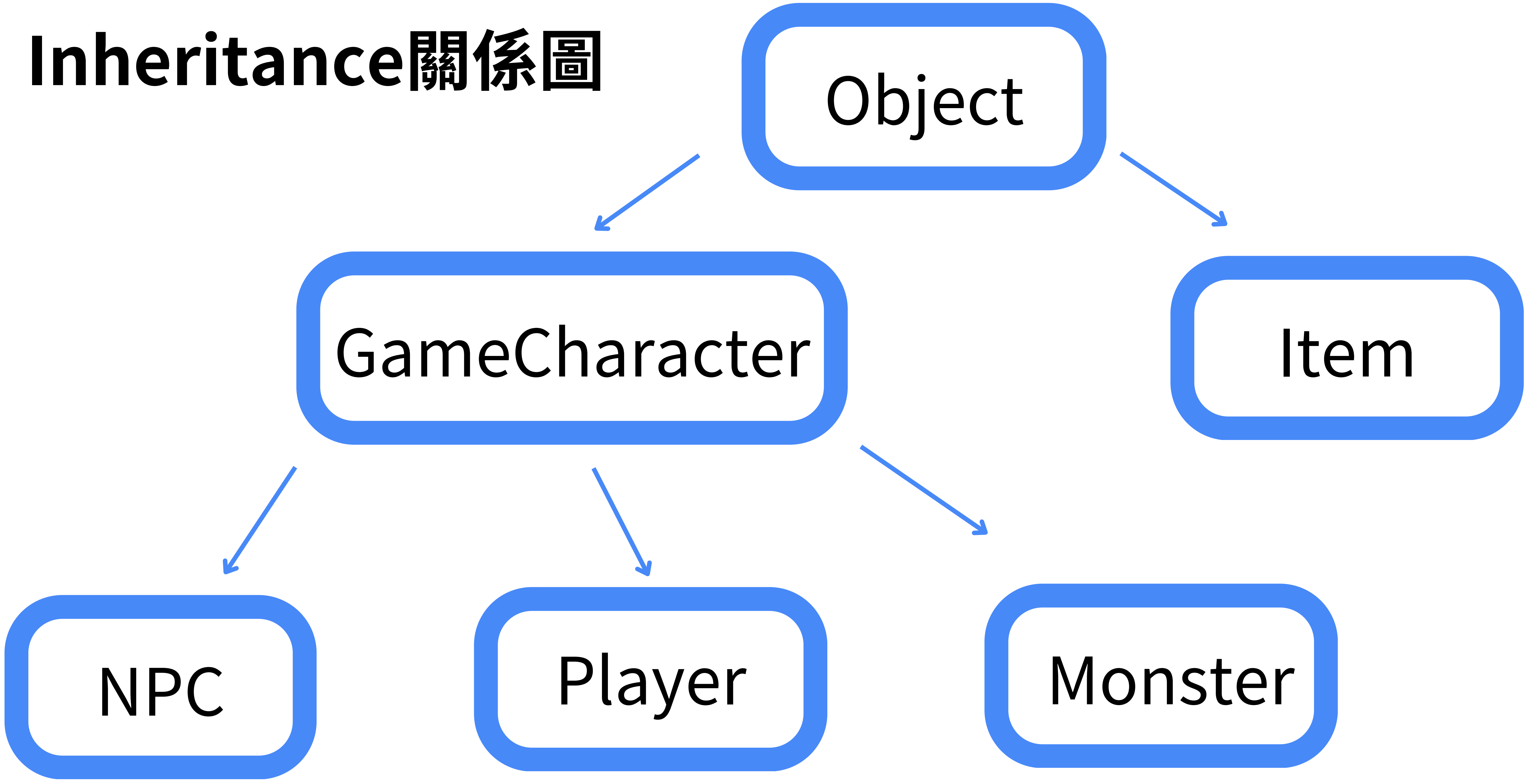
# OOP RPG-Dungeon

學號: 112550198 姓名: 簡嫚萱

START MENU

## Class Inheritance關係圖



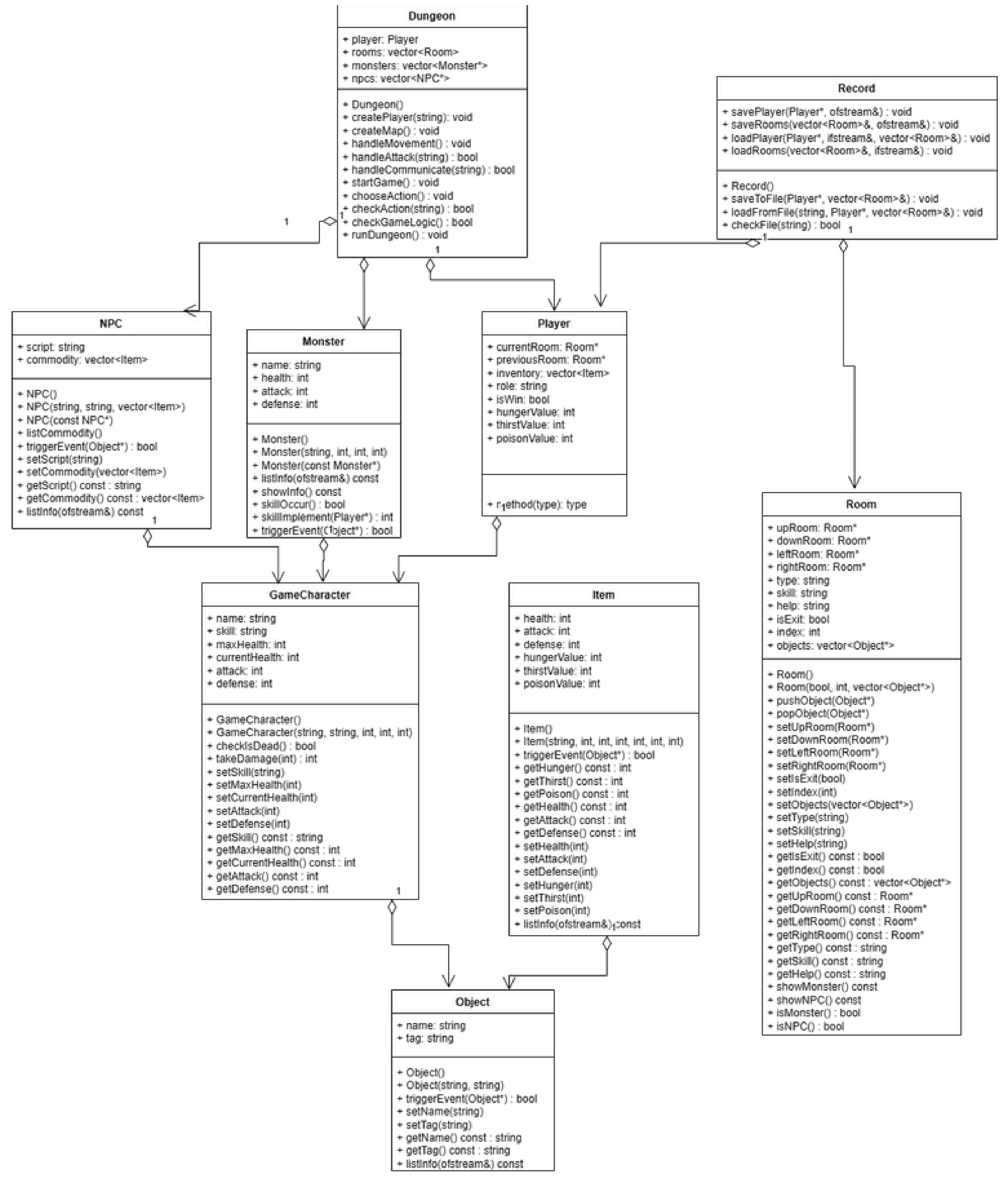
## 其他Class/設置

Dungeon

Room

Record

## UML Design


Dungeon
+ player: Player
+ rooms: vector<Room>
+ monsters: vector<Monster*>
+ npcs: vector<NPC*>
+ Dungeon()
+ createPlayer(string): void
+ createMap() : void
+ handleMovement() : void
+ handleAttack(string) : bool
+ handleCommunicate(string) : bool
+ startGame() : void
+ chooseAction() : void
+ checkAction(string) : bool
+ checkGameLogic() : bool
+ runDungeon() : void
Record
+ savePlayer(Player*, ofstream&) : void
+ saveRooms(vector<Room>&, ofstream&) : void
+ loadPlayer(Player*, ifstream&, vector<Room>&) : void
+ loadRooms(vector<Room>&, ifstream&) : void
+ Record()
+ saveToFile(Player*, vector<Room>&) : void
+ loadFromFile(string, Player*, vector<Room>&) : void
+ checkFile(string) : bool
NPC
+ script: string
+ commodity: vector<Item>
+ NPC()
+ NPC(string, string, vector<Item>)
+ NPC(const NPC*)
+ listCommodity()
+ triggerEvent(Object*) : bool
+ setScript(string)
+ setCommodity(vector<Item>)
+ getScript() const : string
+ getCommodity() const : vector<Item>
+ listInfo(ofstream&) const
Monster
+ name: string
+ health: int
+ attack: int
+ defense: int
+ Monster()
+ Monster(string, int, int, int)
+ Monster(const Monster*)
+ listInfo(ofstream&) const
+ showInfo() const
+ skillOccur() : bool
+ skillImplement(Player*) : int
+ triggerEvent(Object*) : bool
Player
+ currentRoom: Room*
+ previousRoom: Room*
+ inventory: vector<Item>
+ role: string
+ isWin: bool
+ hungerValue: int
+ thirstValue: int
+ poisonValue: int
+ method(type): type
Room
+ upRoom: Room*
+ downRoom: Room*
+ leftRoom: Room*
+ rightRoom: Room*
+ type: string
+ skill: string
+ help: string
+ isExit: bool
+ index: int
+ objects: vector<Object*>
+ Room()
+ Room(bool, int, vector<Object*>)
+ pushObject(Object*)
+ popObject(Object*)
+ setUpRoom(Room*)
+ setDownRoom(Room*)
+ setLeftRoom(Room*)
+ setRightRoom(Room*)
+ setIsExit(bool)
+ setIndex(int)
+ setObjects(vector<Object*>)
+ setType(string)
+ setSkill(string)
+ setHelp(string)
+ getIsExit() const : bool
+ getIndex() const : bool
+ getObjects() const : vector<Object*>
+ getUpRoom() const : Room*
+ getDownRoom() const : Room*
+ getLeftRoom() const : Room*
+ getRightRoom() const : Room*
+ getType() const : string
+ getSkill() const : string
+ getHelp() const : string
+ showMonster() const
+ showNPC() const
+ isMonster() : bool
+ isNPC() : bool
GameCharacter
+ name: string
+ skill: string
+ maxHealth: int
+ currentHealth: int
+ attack: int
+ defense: int
+ GameCharacter()
+ GameCharacter(string, string, int, int, int)
+ checkIsDead() : bool
+ takeDamage(int) : int
+ setSkill(string)
+ setMaxHealth(int)
+ setCurrentHealth(int)
+ setAttack(int)
+ setDefense(int)
+ getSkill() const : string
+ getMaxHealth() const : int
+ getCurrentHealth() const : int
+ getAttack() const : int
+ getDefense() const : int
Item
+ health: int
+ attack: int
+ defense: int
+ hungerValue: int
+ thirstValue: int
+ poisonValue: int
+ Item()
+ Item(string, int, int, int, int, int, int)
+ triggerEvent(Object*) : bool
+ getHunger() const : int
+ getThirst() const : int
+ getPoison() const : int
+ getHealth() const : int
+ getAttack() const : int
+ getDefense() const : int
+ setHealth(int)
+ setAttack(int)
+ setDefense(int)
+ setHunger(int)
+ setThirst(int)
+ setPoison(int)
+ listInfo(ofstream&) const
Object
+ name: string
+ tag: string
+ Object()
+ Object(string, string)
+ triggerEvent(Object*) : bool
+ setName(string)
+ setTag(string)
+ getName() const : string
+ getTag() const : string
+ listInfo(ofstream&) const

## Object

name: name of Object(包含其繼承class的name)

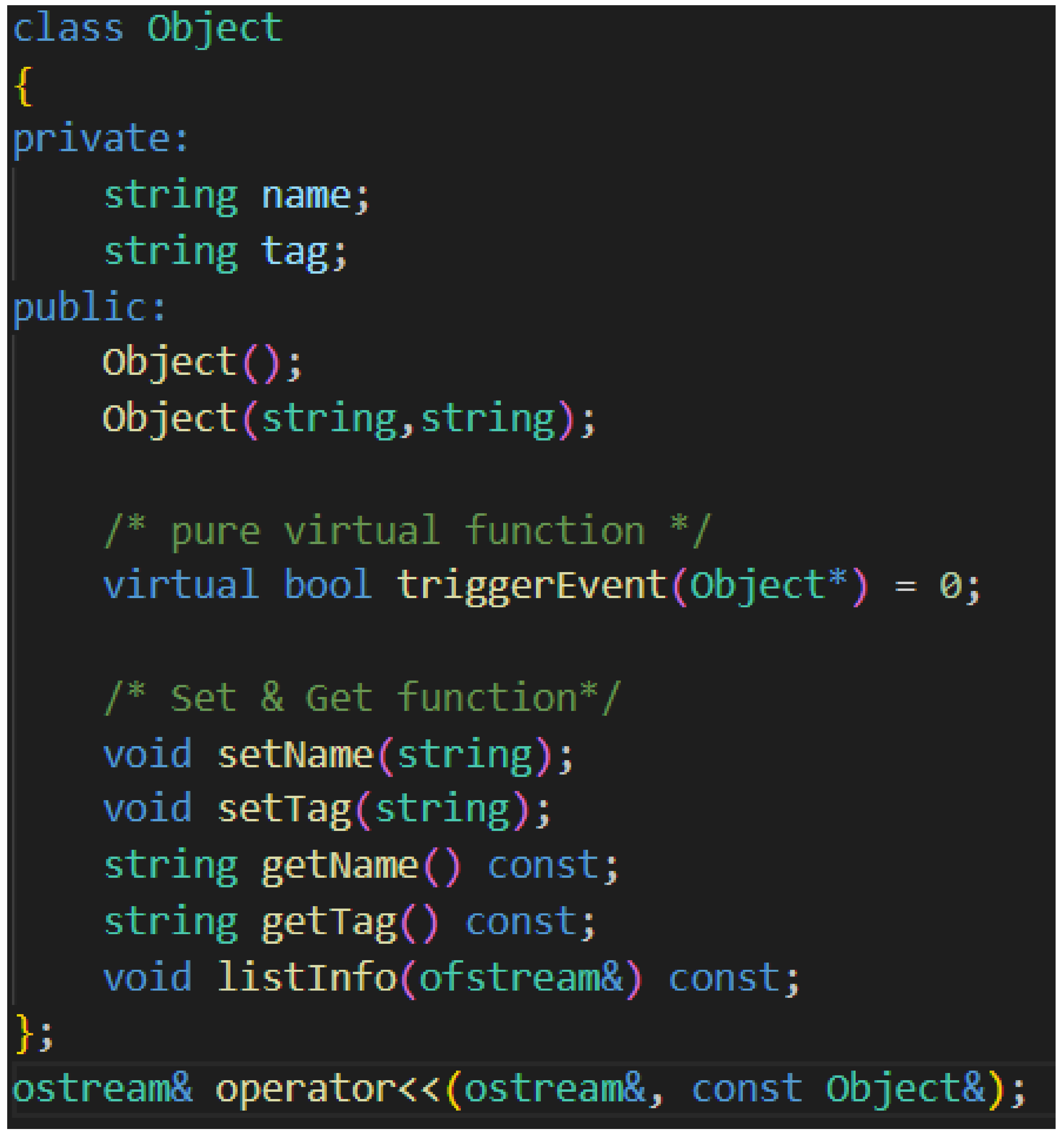
tag: Object的身分, 如Monster型態的tag 為Monster

## 特殊函式介紹

virtual bool triggerEvent(Object*):
用於引發/執行事件的函式, 實際操作在繼承了Object
的其他class中, bool 的return值代表是否成功執行事件

void listInfo(ofstream&) const:
於Record處詳細解釋

p.s operator overload輸出形式是tag+空格+name



```
class Object
{
private:
    string name;
    string tag;
public:
    Object();
    Object(string,string);

    /* pure virtual function */
    virtual bool triggerEvent(Object*) = 0;

    /* Set & Get function*/
    void setName(string);
    void setTag(string);
    string getName() const;
    string getTag() const;
    void listInfo(ofstream&) const;
};
ostream& operator<<(ostream&, const Object&);
```

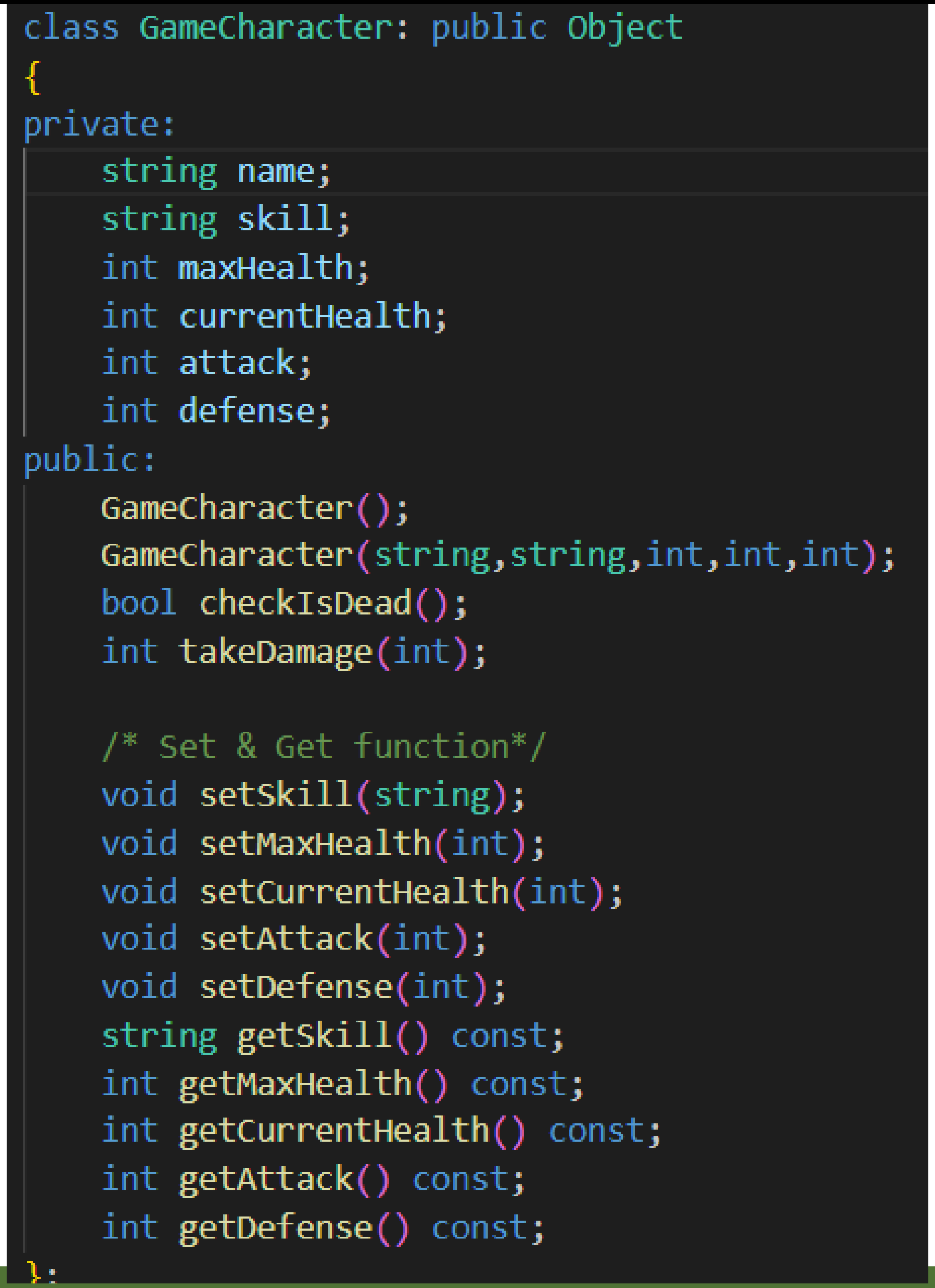
## GameChracter

skill: 角色的技能(實際用在Monster和Player中)

### 特殊函式介紹

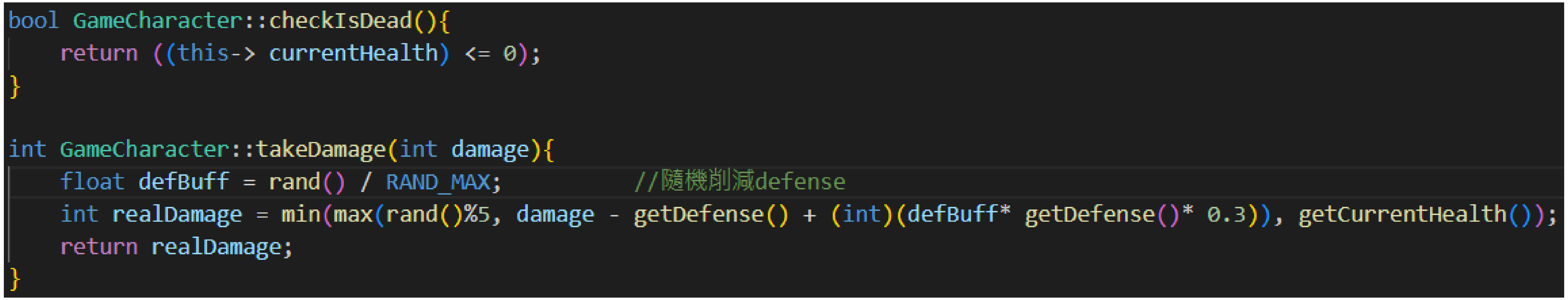
int takeDamage(int):
計算實際受到的傷害, 如attack會被defense/debuff抵銷

bool checkIsDead():
檢查角色(Monster or Player) currentHealth是否歸零



```cpp
class GameCharacter: public Object
{
private:
    string name;
    string skill;
    int maxHealth;
    int currentHealth;
    int attack;
    int defense;
public:
    GameCharacter();
    GameCharacter(string,string,int,int,int);
    bool checkIsDead();
    int takeDamage(int);

    /* Set & Get function*/
    void setSkill(string);
    void setMaxHealth(int);
    void setCurrentHealth(int);
    void setAttack(int);
    void setDefense(int);
    string getSkill() const;
    int getMaxHealth() const;
    int getCurrentHealth() const;
    int getAttack() const;
    int getDefense() const;
};
```

# Detailed Implementation



```cpp
bool GameCharacter::checkIsDead(){
    return ((this-> currentHealth) <= 0);
}

int GameCharacter::takeDamage(int damage){
    float defBuff = rand() / RAND_MAX;          //隨機削減defense
    int realDamage = min(max(rand()%5, damage - getDefense() + (int)(defBuff* getDefense()* 0.3)), getCurrentHealth());
    return realDamage;
}
```

## Room

type: room建立時利用rand()抽的房間類型

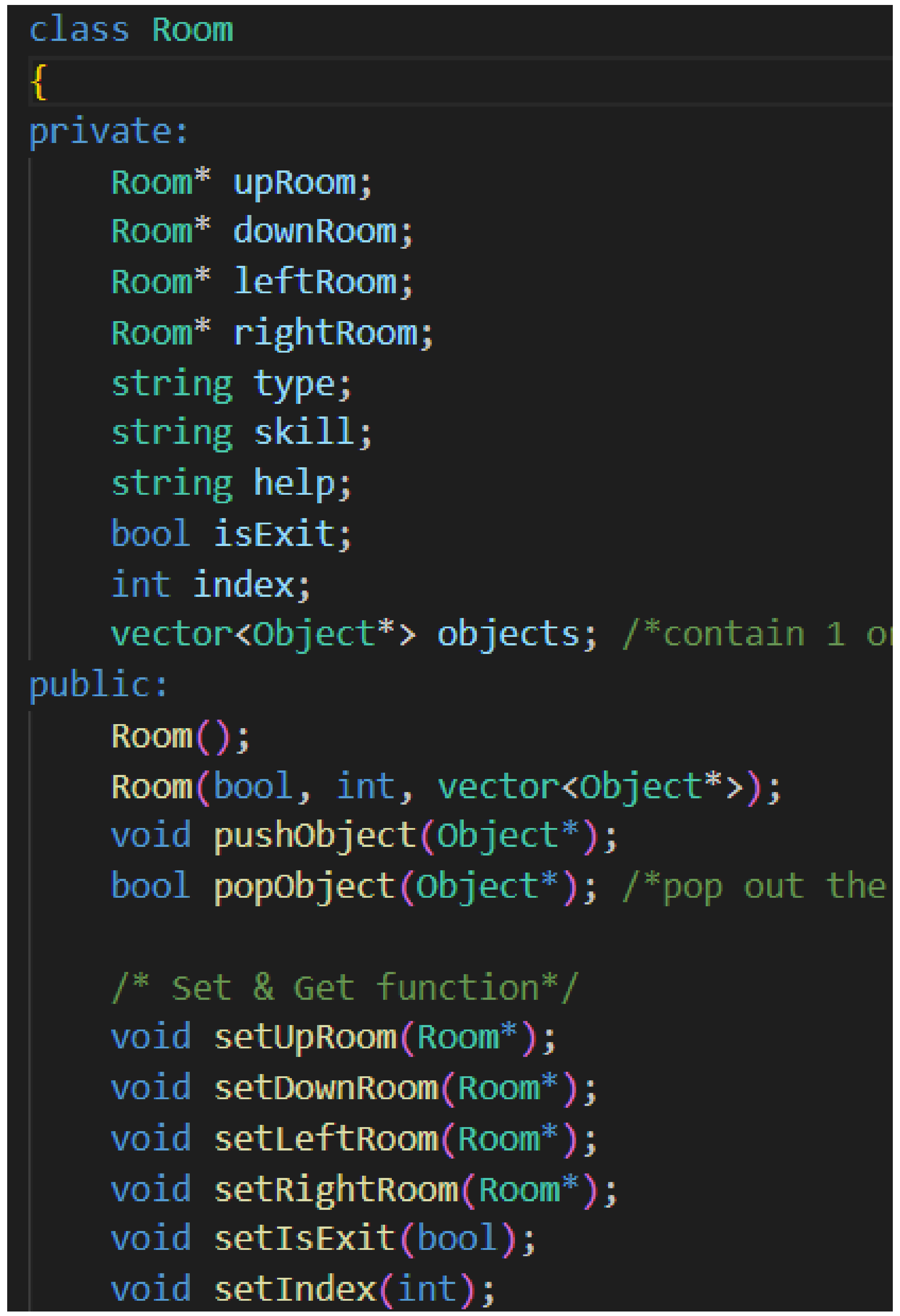
skill: 同上, 依照type(環境)決定skill(環境特性), skill為對player不利的特性

help: 同上, 依照type(環境)決定help(環境特性), help為對player有利的特性

## 特殊函式介紹

void showMonster()/showNPC():
把房間裡的Monster/NPC的資料(依照operator overload)全部輸出, 以讓player知道目前情況

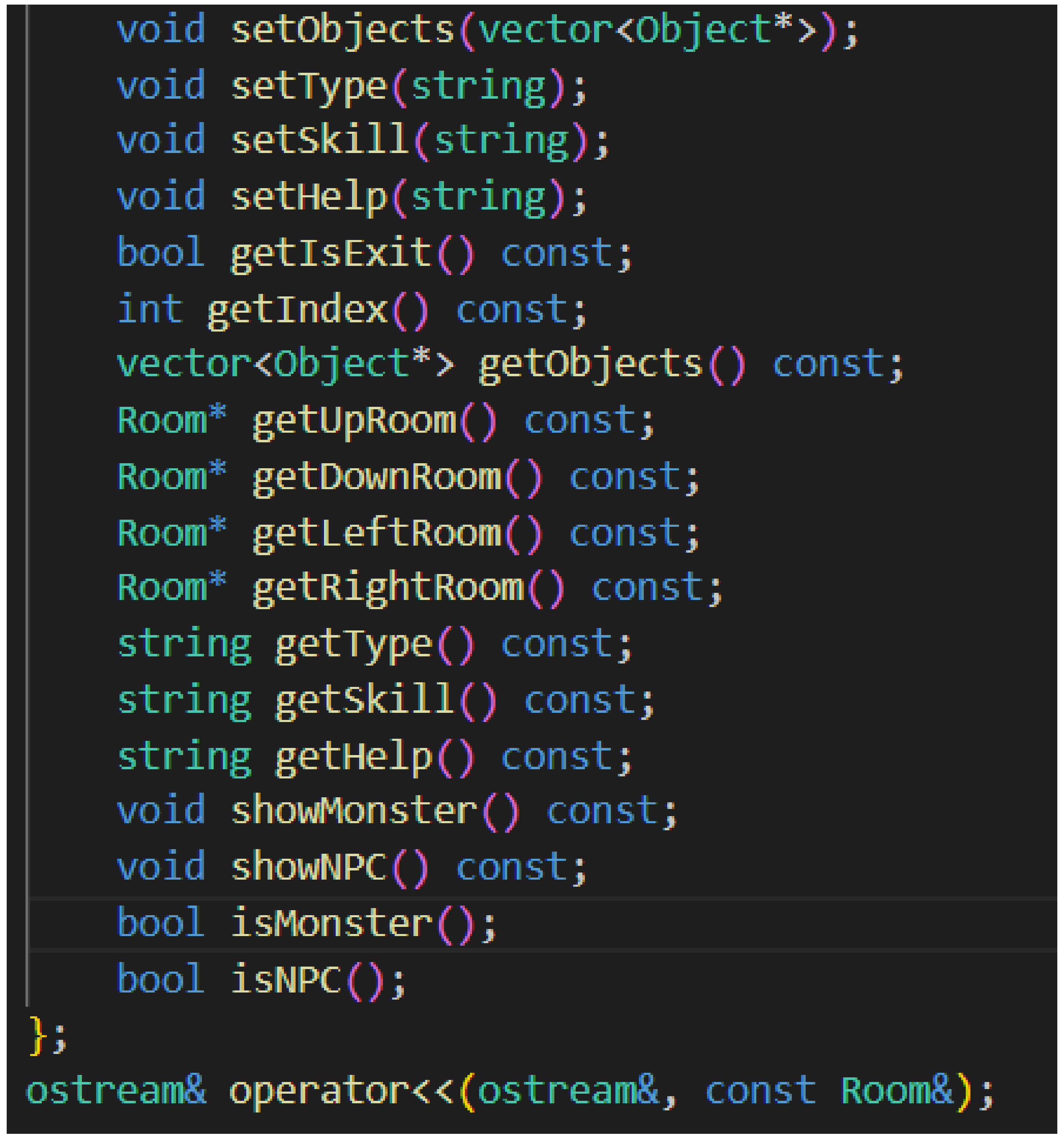
(下頁繼續)



```
class Room
{
private:
    Room* upRoom;
    Room* downRoom;
    Room* leftRoom;
    Room* rightRoom;
    string type;
    string skill;
    string help;
    bool isExit;
    int index;
    vector<Object*> objects; /*contain 1 o
public:
    Room();
    Room(bool, int, vector<Object*>);
    void pushObject(Object*);
    bool popObject(Object*); /*pop out the

    /* Set & Get function*/
    void setUpRoom(Room*);
    void setDownRoom(Room*);
    void setLeftRoom(Room*);
    void setRightRoom(Room*);
    void setIsExit(bool);
    void setIndex(int);
```

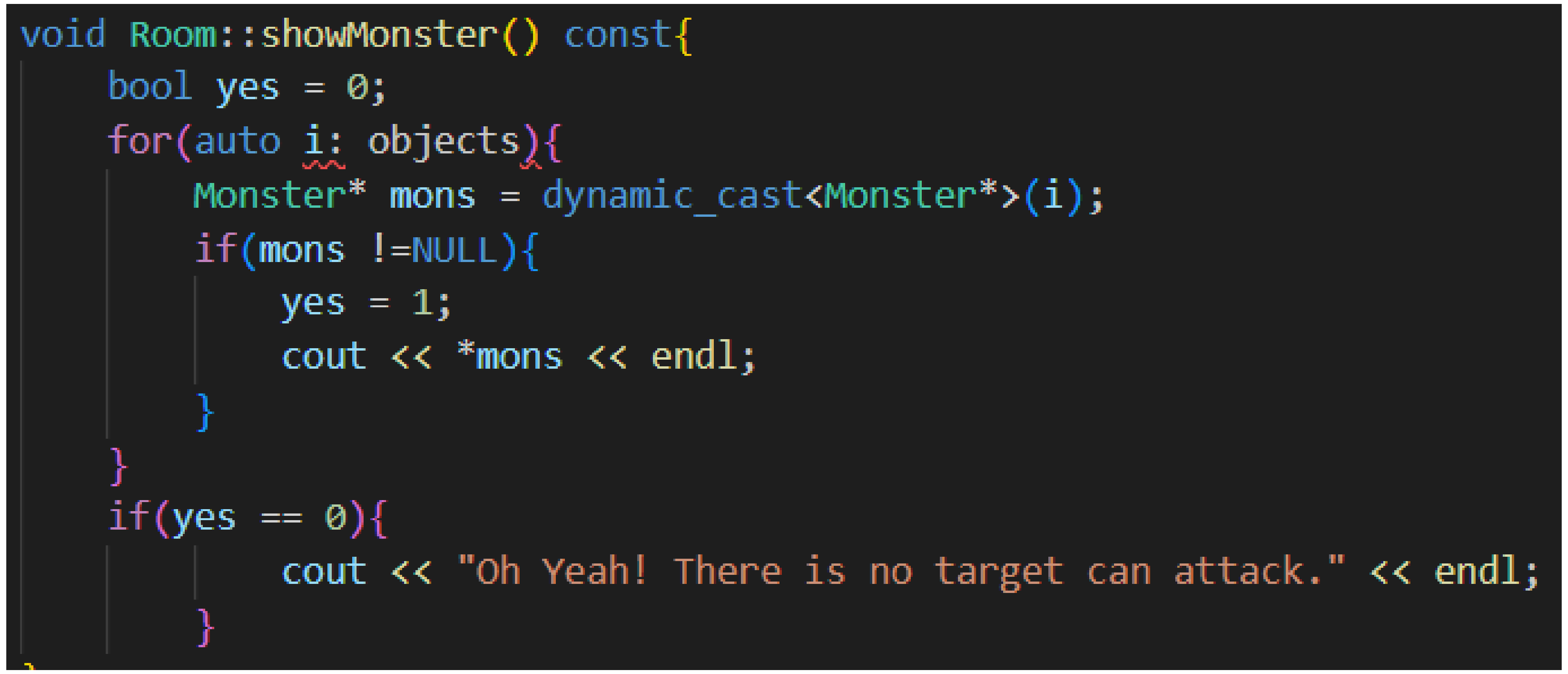
bool isMonster()/isNPC():
用來判斷房間裡是否還有Monster/NPC

p.s. operator overload輸出型式
見detailed implementation

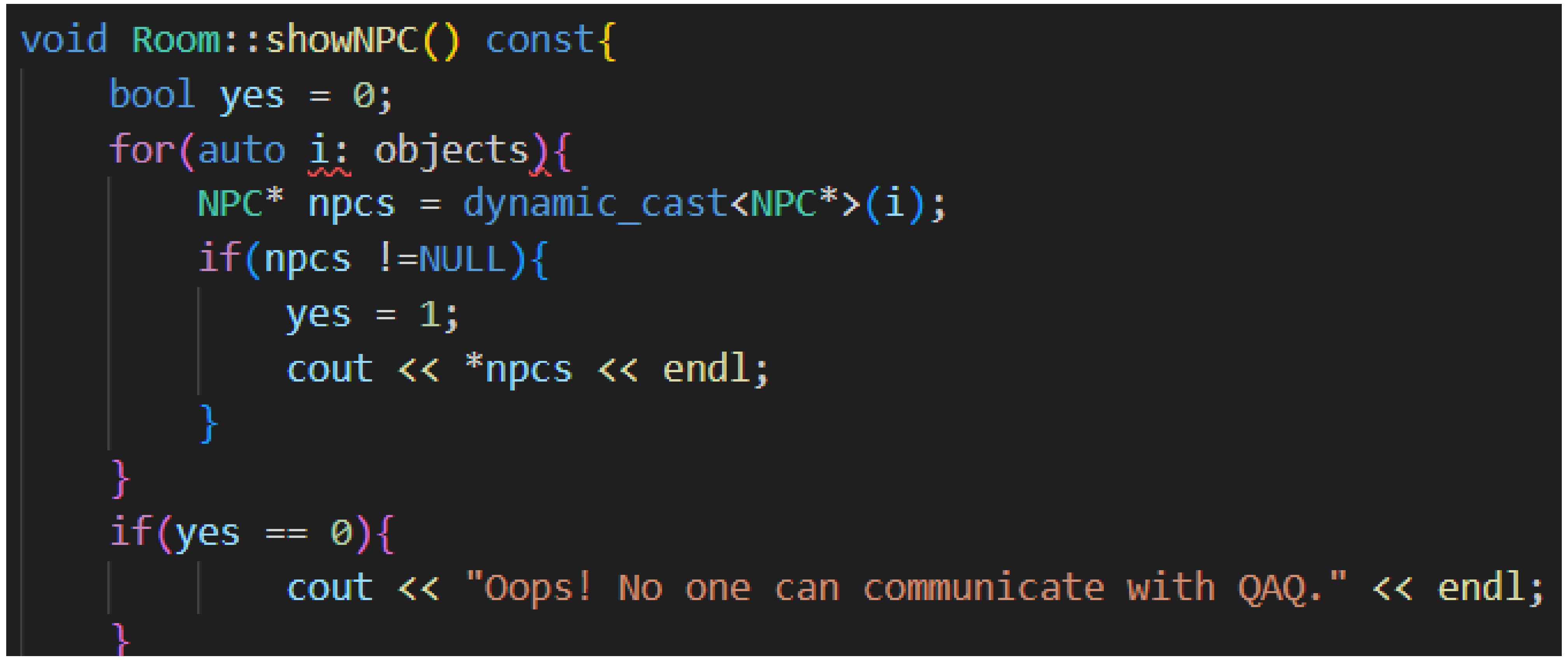


```
    void setObjects(vector<Object*>);
    void setType(string);
    void setSkill(string);
    void setHelp(string);
    bool getIsExit() const;
    int getIndex() const;
    vector<Object*> getObjects() const;
    Room* getUpRoom() const;
    Room* getDownRoom() const;
    Room* getLeftRoom() const;
    Room* getRightRoom() const;
    string getType() const;
    string getSkill() const;
    string getHelp() const;
    void showMonster() const;
    void showNPC() const;
    bool isMonster();
    bool isNPC();
};
ostream& operator<<(ostream&, const Room&);
```

## Detailed Implementation

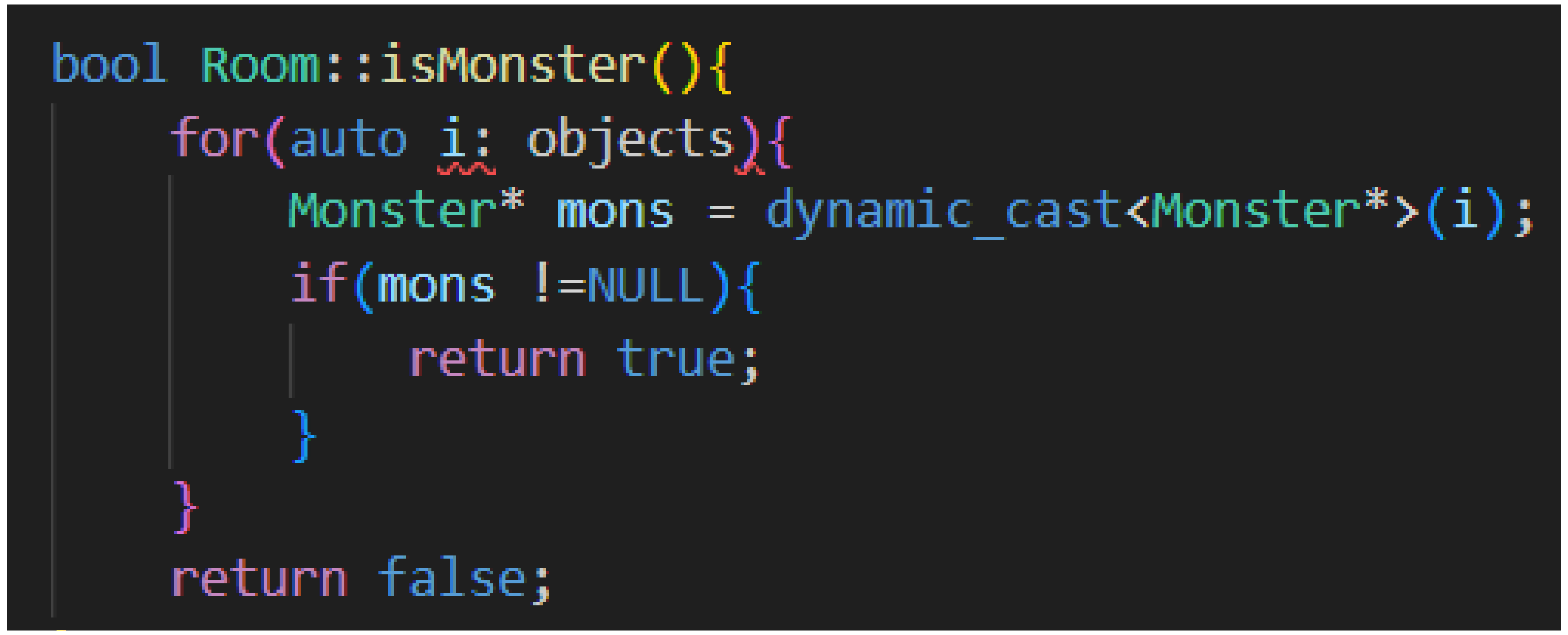


```cpp
void Room::showMonster() const{
    bool yes = 0;
    for(auto i: objects){
        Monster* mons = dynamic_cast<Monster*>(i);
        if(mons !=NULL){
            yes = 1;
            cout << *mons << endl;
        }
    }
    if(yes == 0){
            cout << "Oh Yeah! There is no target can attack." << endl;
        }
```

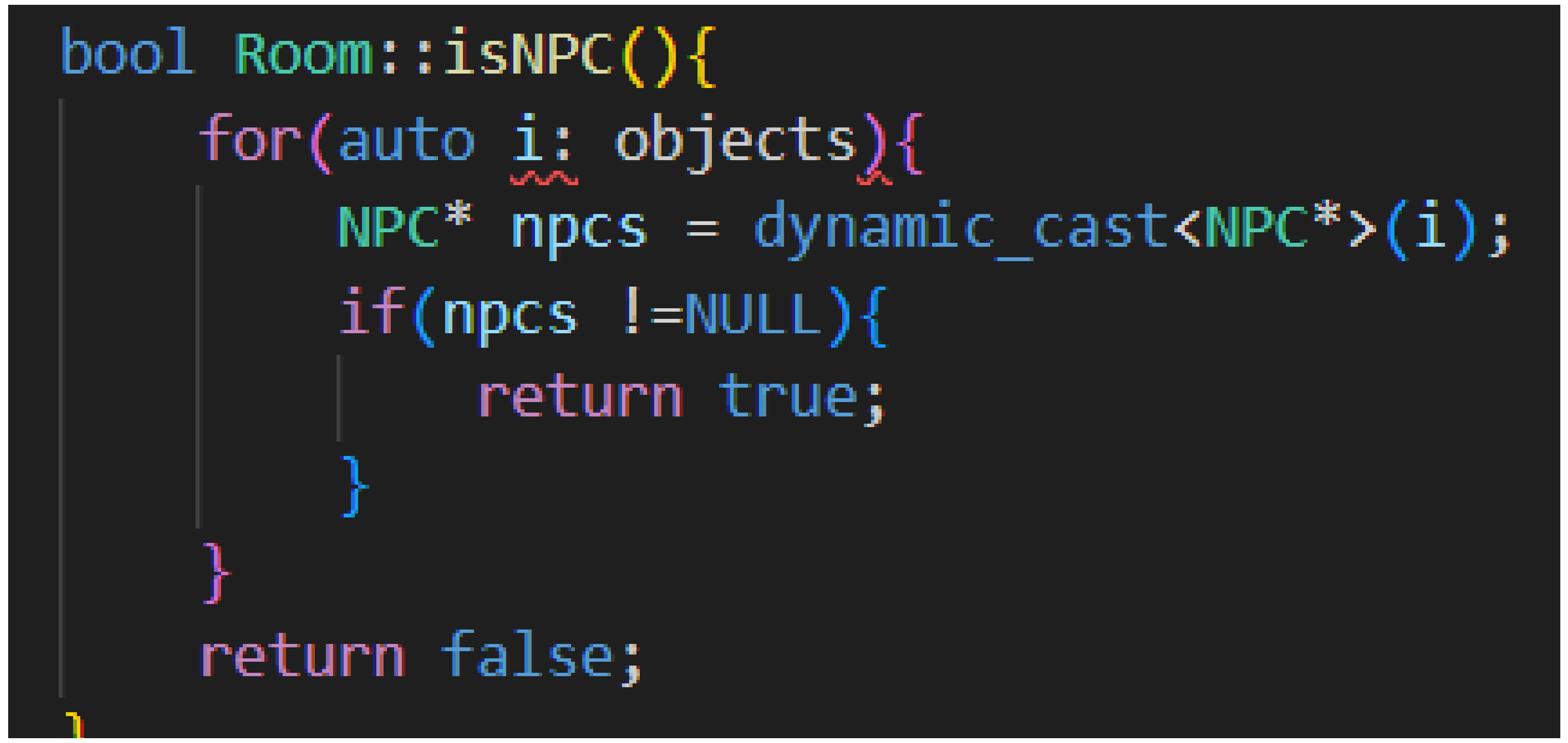


```cpp
void Room::showNPC() const{
    bool yes = 0;
    for(auto i: objects){
        NPC* npcs = dynamic_cast<NPC*>(i);
        if(npcs !=NULL){
            yes = 1;
            cout << *npcs << endl;
        }
    }
    if(yes == 0){
            cout << "Oops! No one can communicate with QAQ." << endl;
    }
```

## Detailed Implementation



```
bool Room::isMonster(){
    for(auto i: objects){
        Monster* mons = dynamic_cast<Monster*>(i);
        if(mons !=NULL){
            return true;
        }
    }
    return false;
```



```
bool Room::isNPC(){
    for(auto i: objects){
        NPC* npcs = dynamic_cast<NPC*>(i);
        if(npcs !=NULL){
            return true;
        }
    }
    return false;
```

## NPC

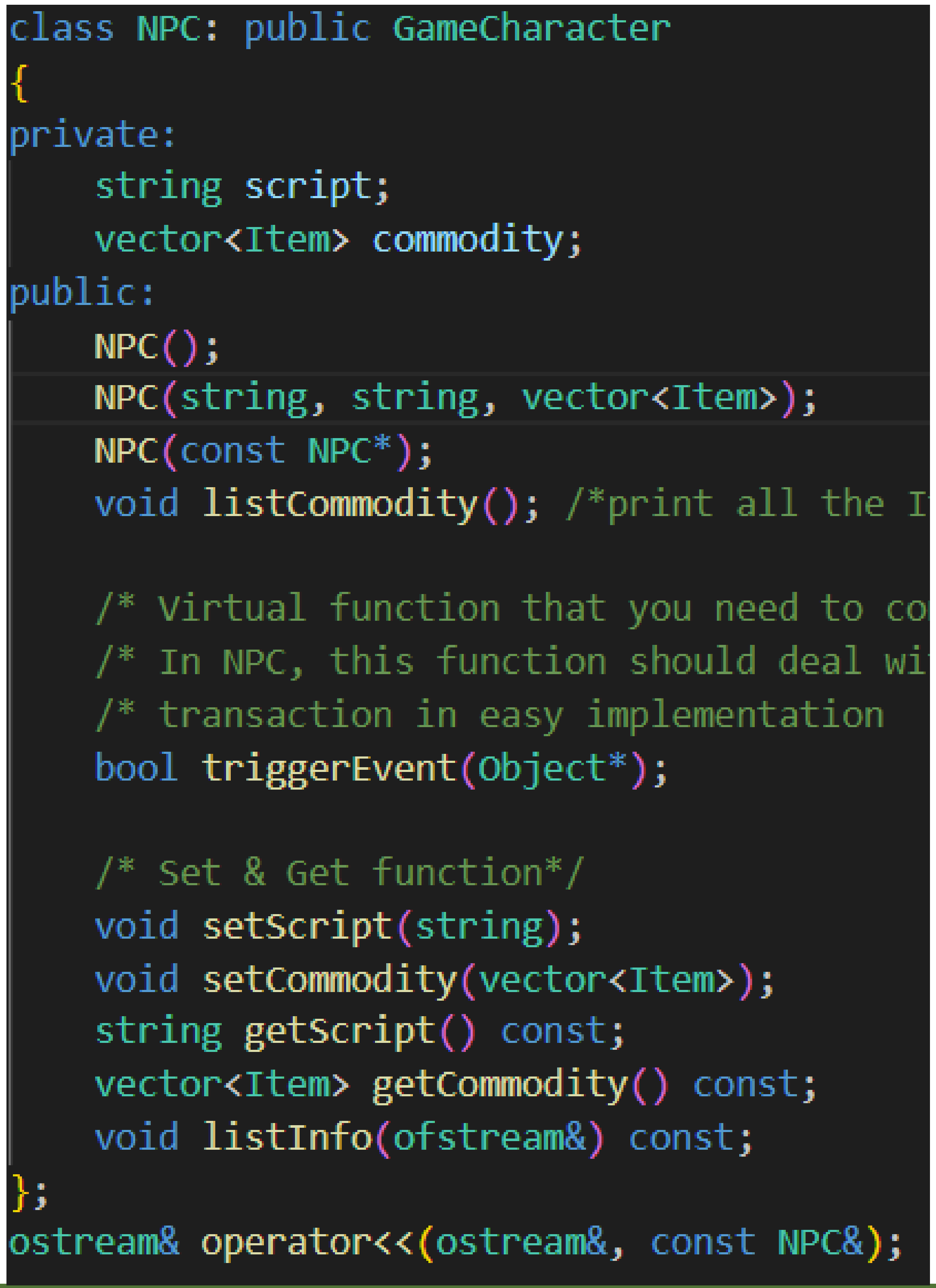
script: NPC出場時說的話
commodity: NPC所持有的物品

## 特殊函式介紹

bool triggerEvent(Object*):
為class Object中virtual函式的override, 用來執行遇到NPC和跟NPC拿東西時的情況, 東西都被拿完的話return true, 反之return false

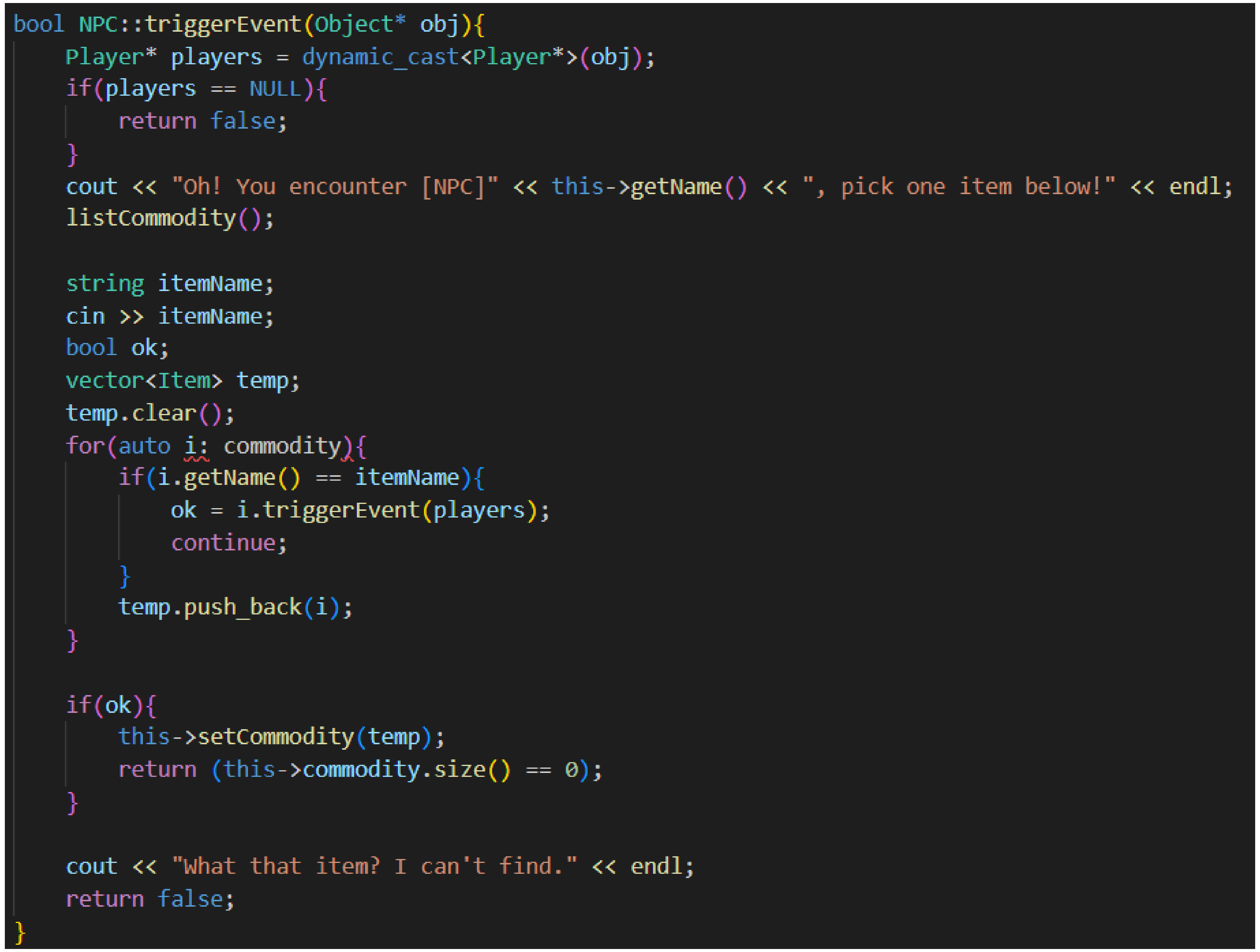
void listInfo(ofstream&) const:
於Record處詳細解釋

p.s operator overload輸出形式是NPC:+空格+name



```
class NPC: public GameCharacter
{
private:
    string script;
    vector<Item> commodity;
public:
    NPC();
    NPC(string, string, vector<Item>);
    NPC(const NPC*);
    void listCommodity(); /*print all the I

    /* Virtual function that you need to co
    /* In NPC, this function should deal wi
    /* transaction in easy implementation
    bool triggerEvent(Object*);

    /* Set & Get function*/
    void setScript(string);
    void setCommodity(vector<Item>);
    string getScript() const;
    vector<Item> getCommodity() const;
    void listInfo(ofstream&) const;
};
ostream& operator<<(ostream&, const NPC&);
```

## Detailed Implementation



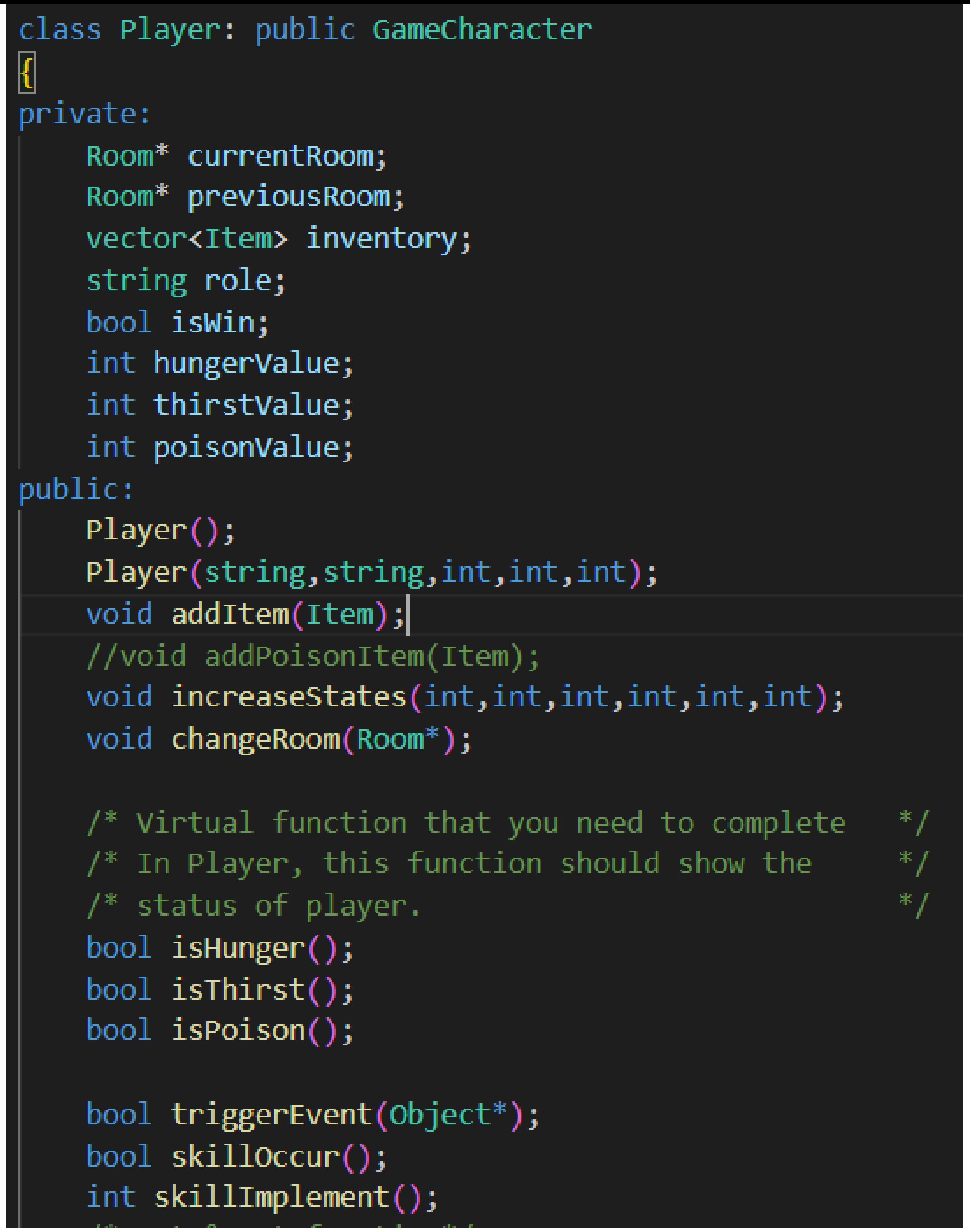
```
bool NPC::triggerEvent(Object* obj){
    Player* players = dynamic_cast<Player*>(obj);
    if(players == NULL){
        return false;
    }
    cout << "Oh! You encounter [NPC]" << this->getName() << ", pick one item below!" << endl;
    listCommodity();

    string itemName;
    cin >> itemName;
    bool ok;
    vector<Item> temp;
    temp.clear();
    for(auto i: commodity){
        if(i.getName() == itemName){
            ok = i.triggerEvent(players);
            continue;
        }
        temp.push_back(i);
    }

    if(ok){
        this->setCommodity(temp);
        return (this->commodity.size() == 0);
    }

    cout << "What that item? I can't find." << endl;
    return false;
}
```

## Player

inventory: player目前持有的item
role: 開局時抽的角色, 會決定技能為何, 詳見
    class Dungeon
isWin: 確認Boss的checkIsDead()來判斷是否獲勝的值
hungerValue, thirstValue, poisonValue:
Hunger system的實作, hungerValue和thirstValue會從100開始扣, 而poisonValue則是中毒時才會>0

## 特殊函式介紹

bool isHunger()/isThirst()/isPoison():
用來判斷是否達臨界值(達臨界值則需要扣Health)
(下頁繼續)



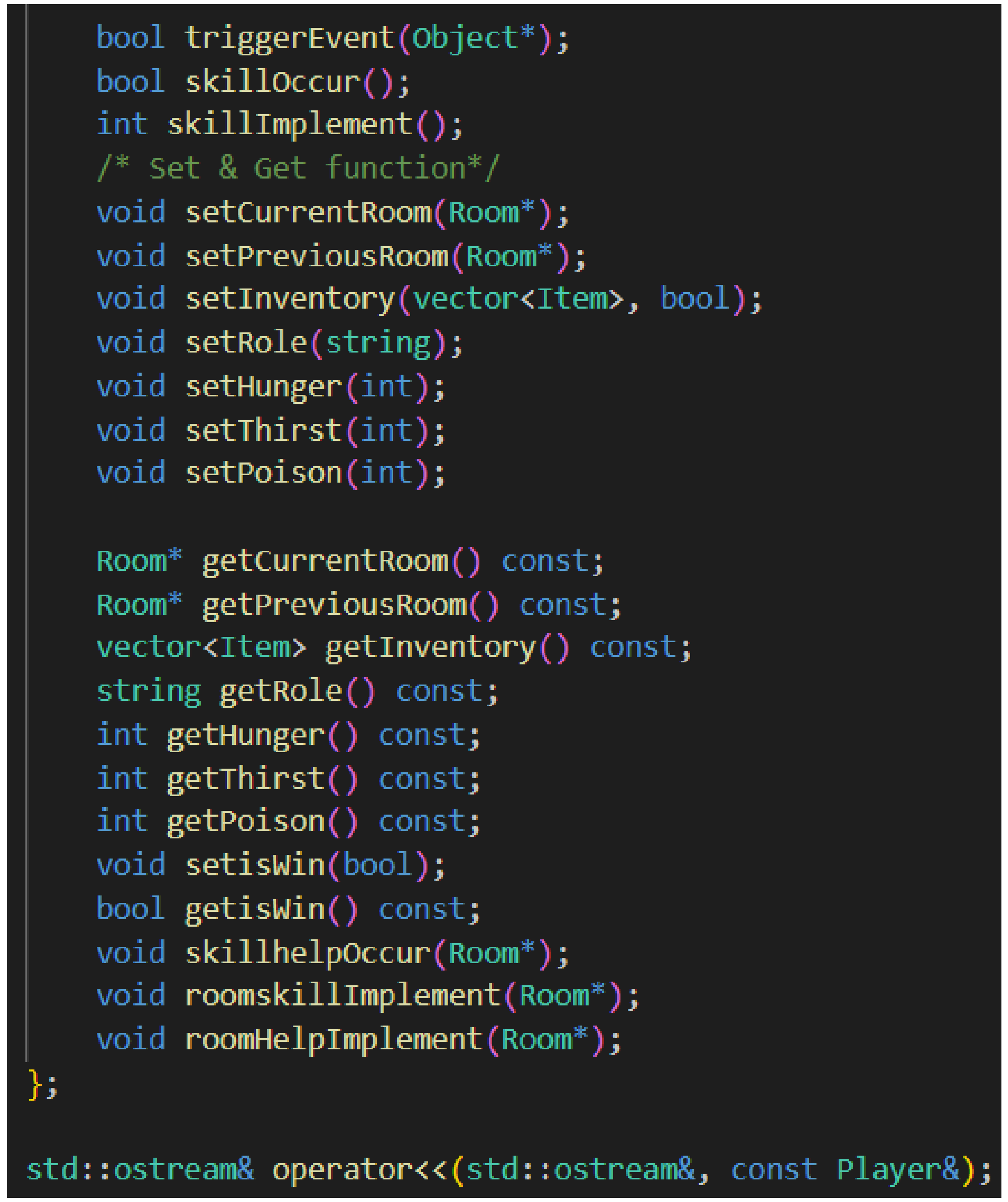
```
class Player: public GameCharacter
{
private:
    Room* currentRoom;
    Room* previousRoom;
    vector<Item> inventory;
    string role;
    bool isWin;
    int hungerValue;
    int thirstValue;
    int poisonValue;
public:
    Player();
    Player(string,string,int,int,int);
    void addItem(Item);
    //void addPoisonItem(Item);
    void increaseStates(int,int,int,int,int,int);
    void changeRoom(Room*);

    /* Virtual function that you need to complete   */
    /* In Player, this function should show the     */
    /* status of player.                            */
    bool isHunger();
    bool isThirst();
    bool isPoison();

    bool triggerEvent(Object*);
    bool skillOccur();
    int skillImplement();
```

bool triggerEvent(Object*):
在我的設計中並沒有讓這個函式做事, 而是用NPC和Monster的triggerEvent來達成事件

bool skillOccur(Object*):
用rand()來隨機決定是否要執行skillImplement()

int skillImplement():
依照抽到的role決定技能並發動並執行
(對Monster or 對自己), return改變的傷害值

void skillhelpOccur():
用rand()來隨機決定是否要執行
roomskillImplement()還是
roomHelpImplement()
(下頁繼續)

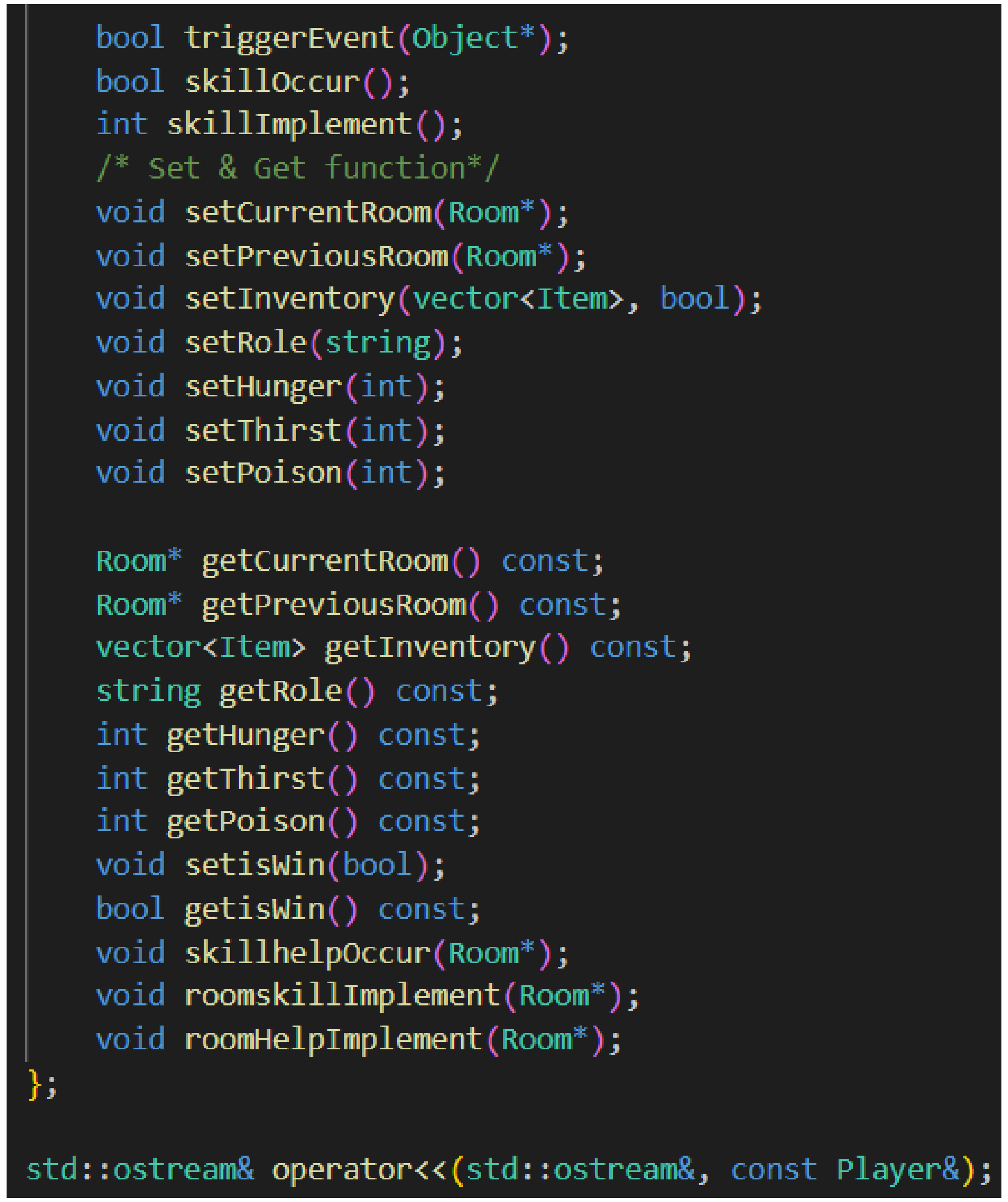


```
    bool triggerEvent(Object*);
    bool skillOccur();
    int skillImplement();
    /* Set & Get function*/
    void setCurrentRoom(Room*);
    void setPreviousRoom(Room*);
    void setInventory(vector<Item>, bool);
    void setRole(string);
    void setHunger(int);
    void setThirst(int);
    void setPoison(int);

    Room* getCurrentRoom() const;
    Room* getPreviousRoom() const;
    vector<Item> getInventory() const;
    string getRole() const;
    int getHunger() const;
    int getThirst() const;
    int getPoison() const;
    void setisWin(bool);
    bool getisWin() const;
    void skillhelpOccur(Room*);
    void roomskillImplement(Room*);
    void roomHelpImplement(Room*);
};

std::ostream& operator<<(std::ostream&, const Player&);
```

void roomskillImplement():
開局時Room同樣會利用rand()抽取環境得到技能,
此為執行環境特性對Player造成的影響(不好的)

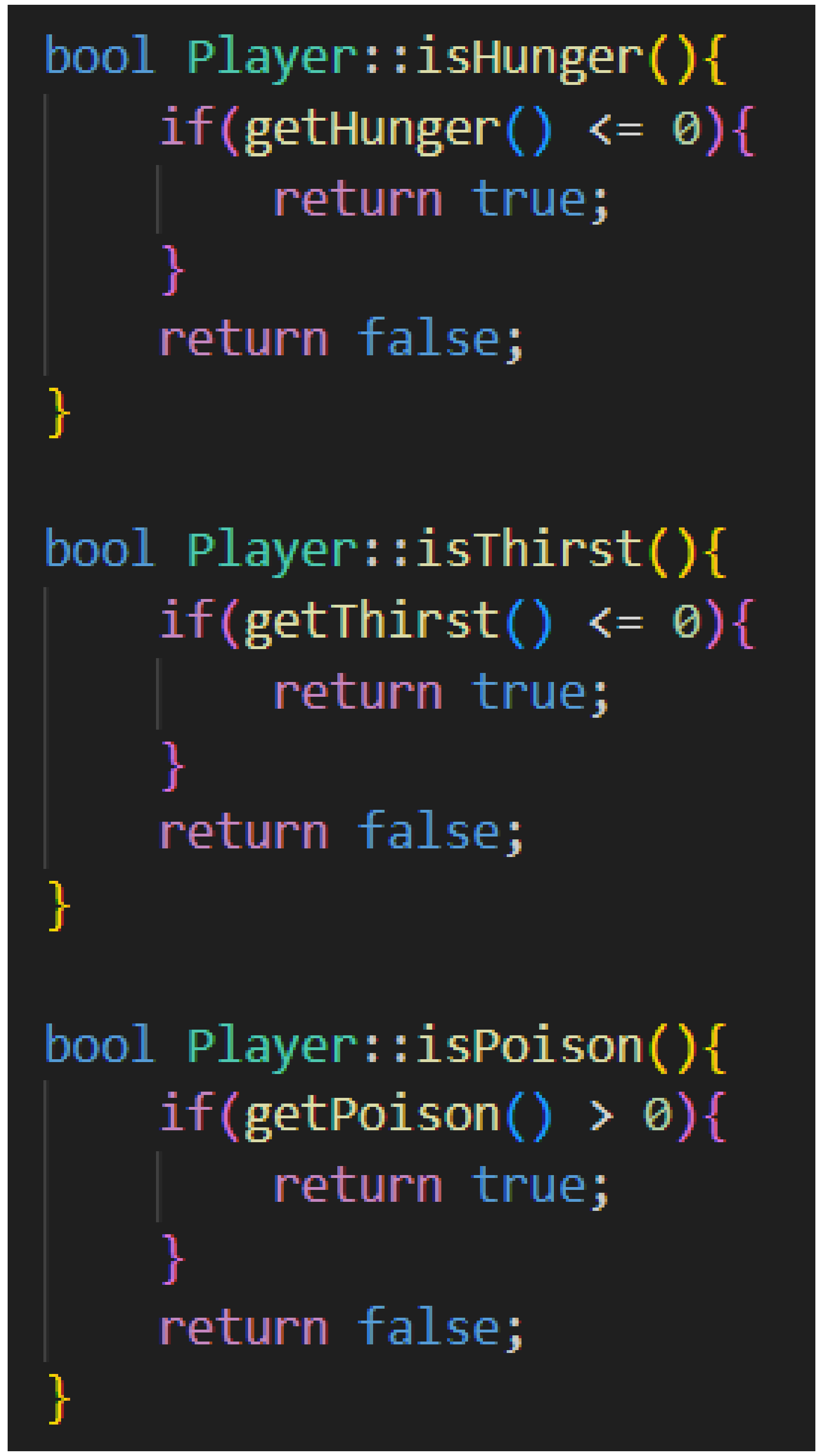
void roomHelpImplement():
開局時Room同樣會利用rand()抽取環境得到技能,
此為執行環境特性對Player造成的影響(好的)

p.s. operator overload輸出型式見detailed implementation

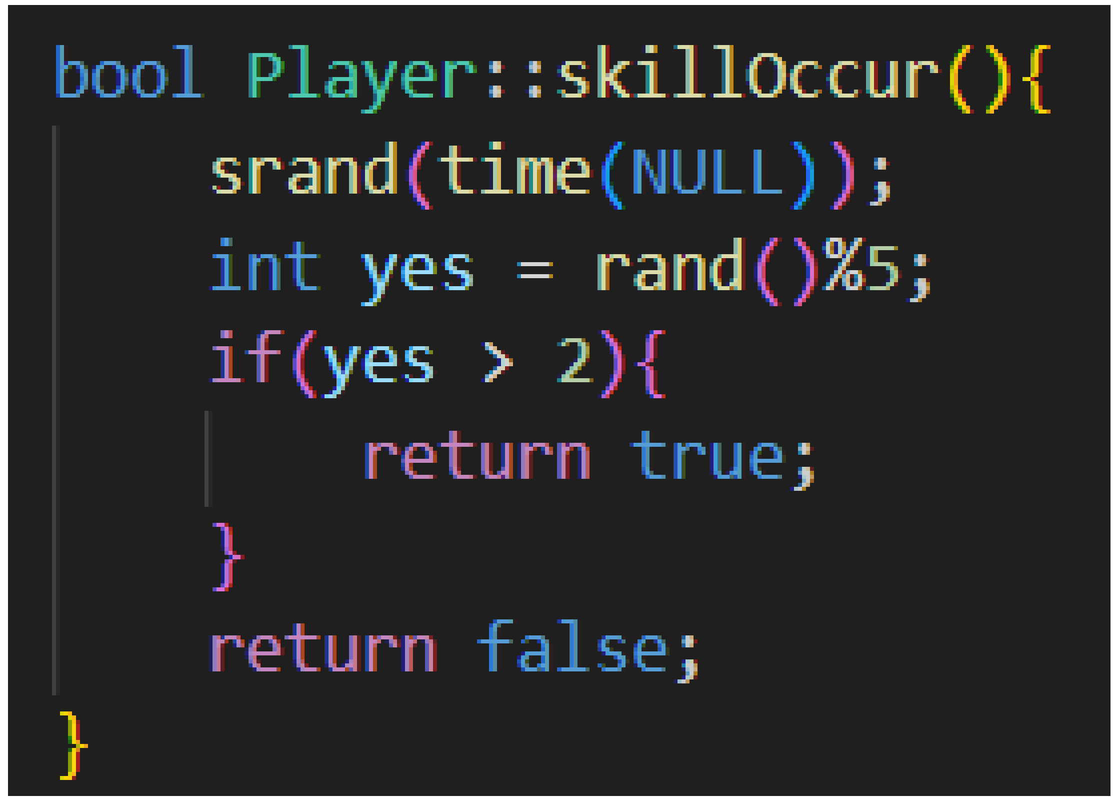


```
    bool triggerEvent(Object*);
    bool skillOccur();
    int skillImplement();
    /* Set & Get function*/
    void setCurrentRoom(Room*);
    void setPreviousRoom(Room*);
    void setInventory(vector<Item>, bool);
    void setRole(string);
    void setHunger(int);
    void setThirst(int);
    void setPoison(int);

    Room* getCurrentRoom() const;
    Room* getPreviousRoom() const;
    vector<Item> getInventory() const;
    string getRole() const;
    int getHunger() const;
    int getThirst() const;
    int getPoison() const;
    void setisWin(bool);
    bool getisWin() const;
    void skillhelpOccur(Room*);
    void roomskillImplement(Room*);
    void roomHelpImplement(Room*);
};

std::ostream& operator<<(std::ostream&, const Player&);
```

## Detailed Implementation



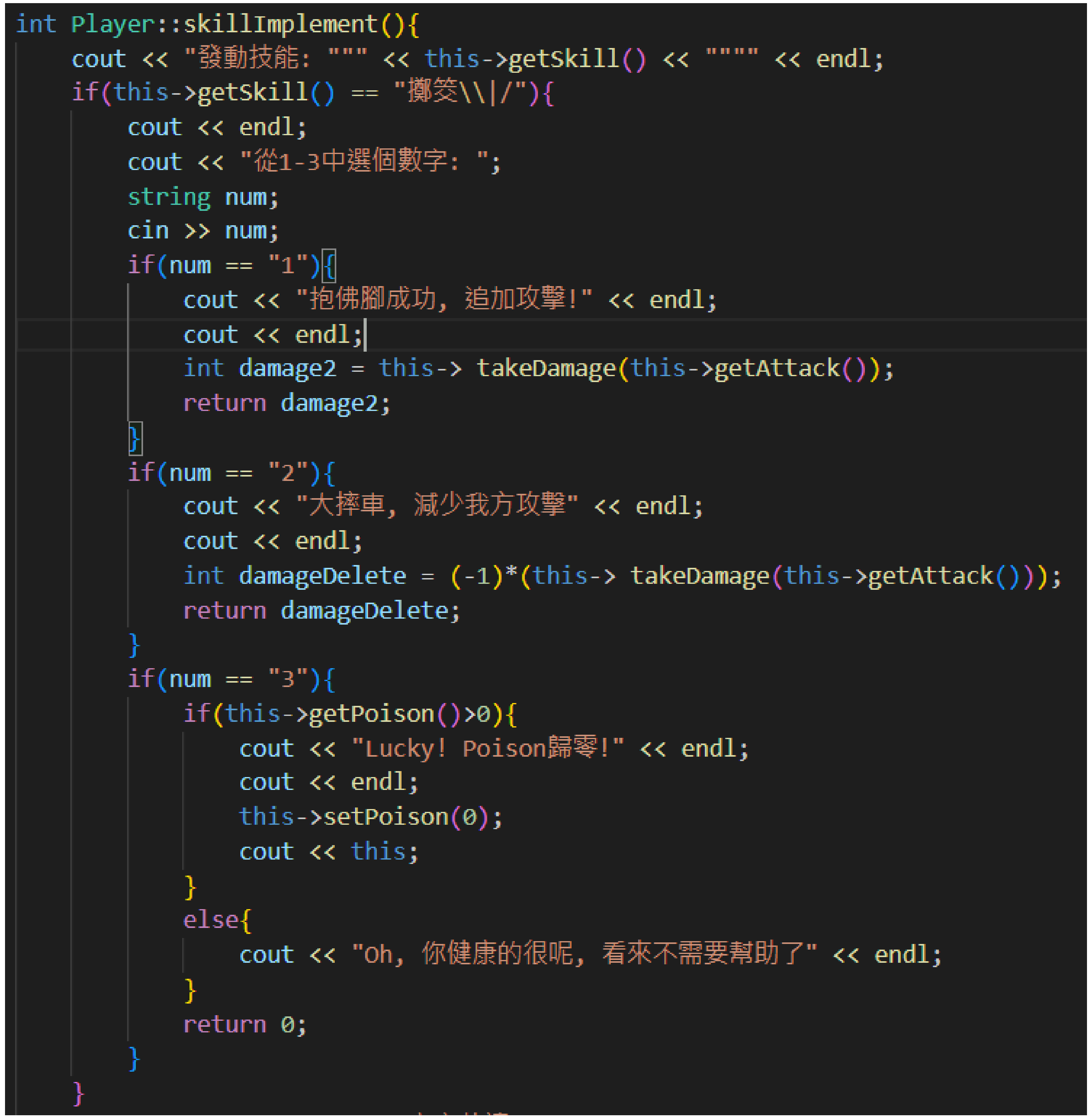
```
bool Player::isHunger(){
    if(getHunger() <= 0){
        return true;
    }
    return false;
}

bool Player::isThirst(){
    if(getThirst() <= 0){
        return true;
    }
    return false;
}

bool Player::isPoison(){
    if(getPoison() > 0){
        return true;
    }
    return false;
}
```



```
bool Player::skillOccur(){
    srand(time(NULL));
    int yes = rand()%5;
    if(yes > 2){
        return true;
    }
    return false;
}
```

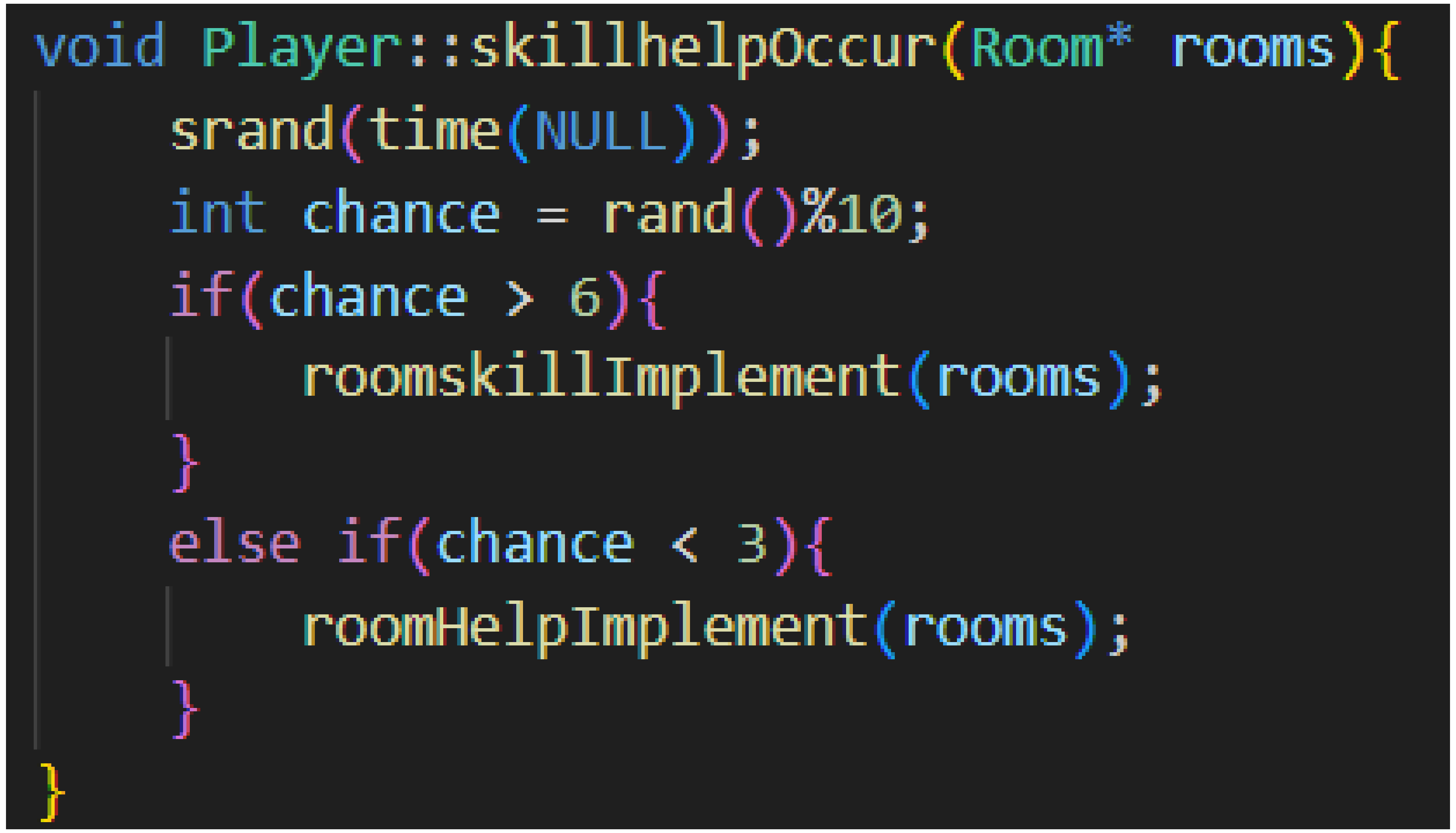
## Detailed Implementation

這裡只列一個技能當例子



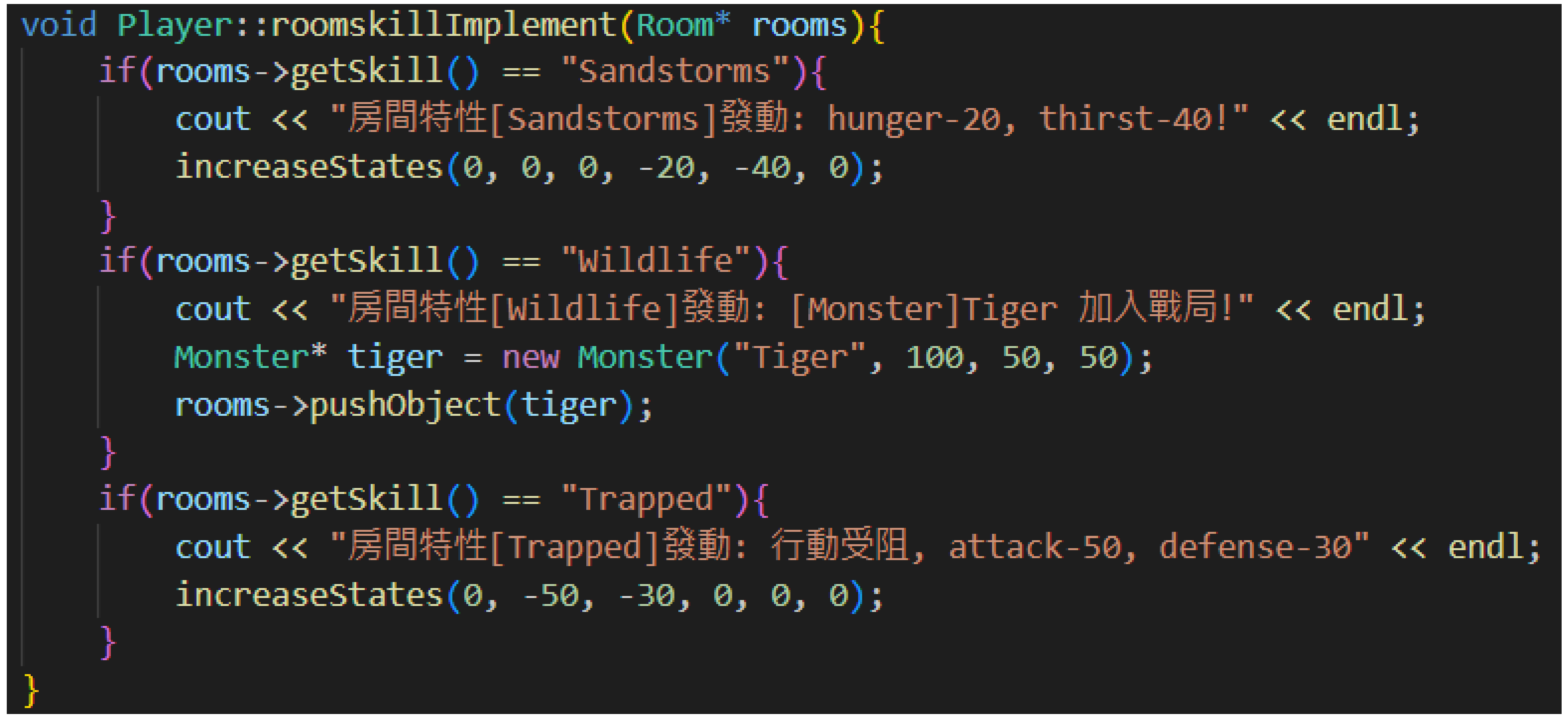
```cpp
int Player::skillImplement(){
    cout << "發動技能: """ << this->getSkill() << """" << endl;
    if(this->getSkill() == "擲筊\\|/"){
        cout << endl;
        cout << "從1-3中選個數字: ";
        string num;
        cin >> num;
        if(num == "1"){
            cout << "抱佛腳成功, 追加攻擊!" << endl;
            cout << endl;
            int damage2 = this-> takeDamage(this->getAttack());
            return damage2;
        }
        if(num == "2"){
            cout << "大摔車, 減少我方攻擊" << endl;
            cout << endl;
            int damageDelete = (-1)*(this-> takeDamage(this->getAttack()));
            return damageDelete;
        }
        if(num == "3"){
            if(this->getPoison()>0){
                cout << "Lucky! Poison歸零!" << endl;
                cout << endl;
                this->setPoison(0);
                cout << this;
            }
            else{
                cout << "Oh, 你健康的很呢, 看來不需要幫助了" << endl;
            }
            return 0;
        }
    }
```

## Detailed Implementation

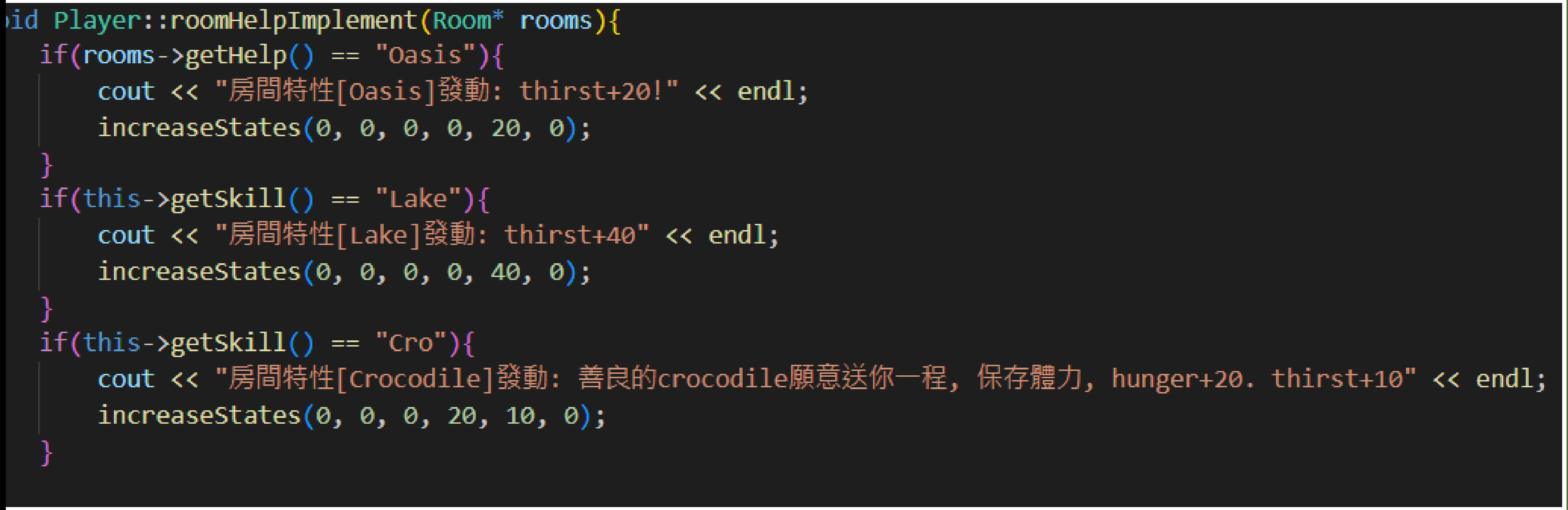


```
void Player::skillhelpOccur(Room* rooms){
    srand(time(NULL));
    int chance = rand()%10;
    if(chance > 6){
        roomskillImplement(rooms);
    }
    else if(chance < 3){
        roomHelpImplement(rooms);
    }
}
```

# Detailed Implementation

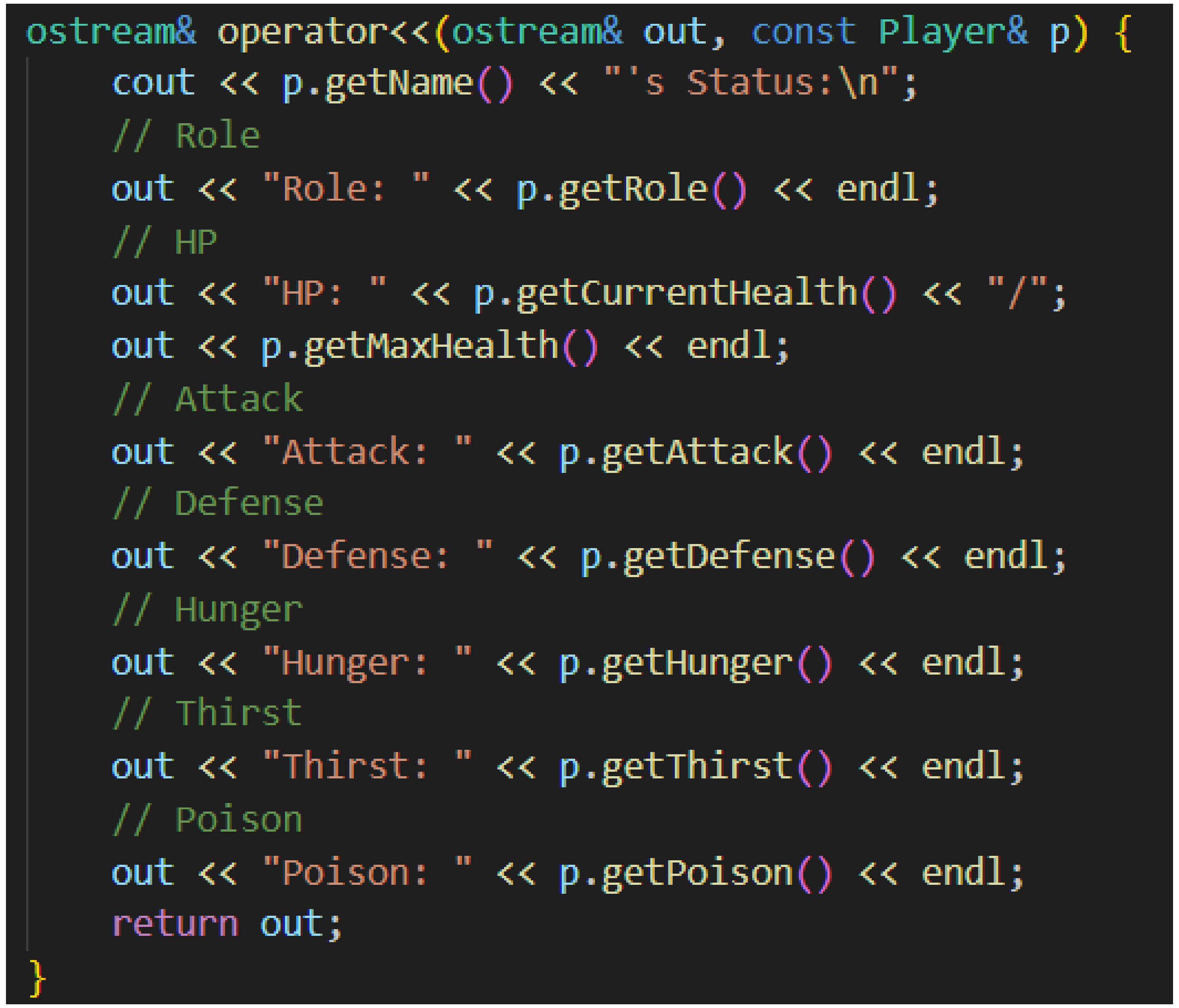


```cpp
void Player::roomskillImplement(Room* rooms){
    if(rooms->getSkill() == "Sandstorms"){
        cout << "房間特性[Sandstorms]發動: hunger-20, thirst-40!" << endl;
        increaseStates(0, 0, 0, -20, -40, 0);
    }
    if(rooms->getSkill() == "Wildlife"){
        cout << "房間特性[Wildlife]發動: [Monster]Tiger 加入戰局!" << endl;
        Monster* tiger = new Monster("Tiger", 100, 50, 50);
        rooms->pushObject(tiger);
    }
    if(rooms->getSkill() == "Trapped"){
        cout << "房間特性[Trapped]發動: 行動受阻, attack-50, defense-30" << endl;
        increaseStates(0, -50, -30, 0, 0, 0);
    }
}
```



```cpp
oid Player::roomHelpImplement(Room* rooms){
    if(rooms->getHelp() == "Oasis"){
        cout << "房間特性[Oasis]發動: thirst+20!" << endl;
        increaseStates(0, 0, 0, 0, 20, 0);
    }
    if(this->getSkill() == "Lake"){
        cout << "房間特性[Lake]發動: thirst+40" << endl;
        increaseStates(0, 0, 0, 0, 40, 0);
    }
    if(this->getSkill() == "Cro"){
        cout << "房間特性[Crocodile]發動: 善良的crocodile願意送你一程, 保存體力, hunger+20. thirst+10" << endl;
        increaseStates(0, 0, 0, 20, 10, 0);
    }
```

## Detailed Implementation



```
ostream& operator<<(ostream& out, const Player& p) {
    cout << p.getName() << "'s Status:\n";
    // Role
    out << "Role: " << p.getRole() << endl;
    // HP
    out << "HP: " << p.getCurrentHealth() << "/";
    out << p.getMaxHealth() << endl;
    // Attack
    out << "Attack: " << p.getAttack() << endl;
    // Defense
    out << "Defense: " << p.getDefense() << endl;
    // Hunger
    out << "Hunger: " << p.getHunger() << endl;
    // Thirst
    out << "Thirst: " << p.getThirst() << endl;
    // Poison
    out << "Poison: " << p.getPoison() << endl;
    return out;
}
```

## Monster

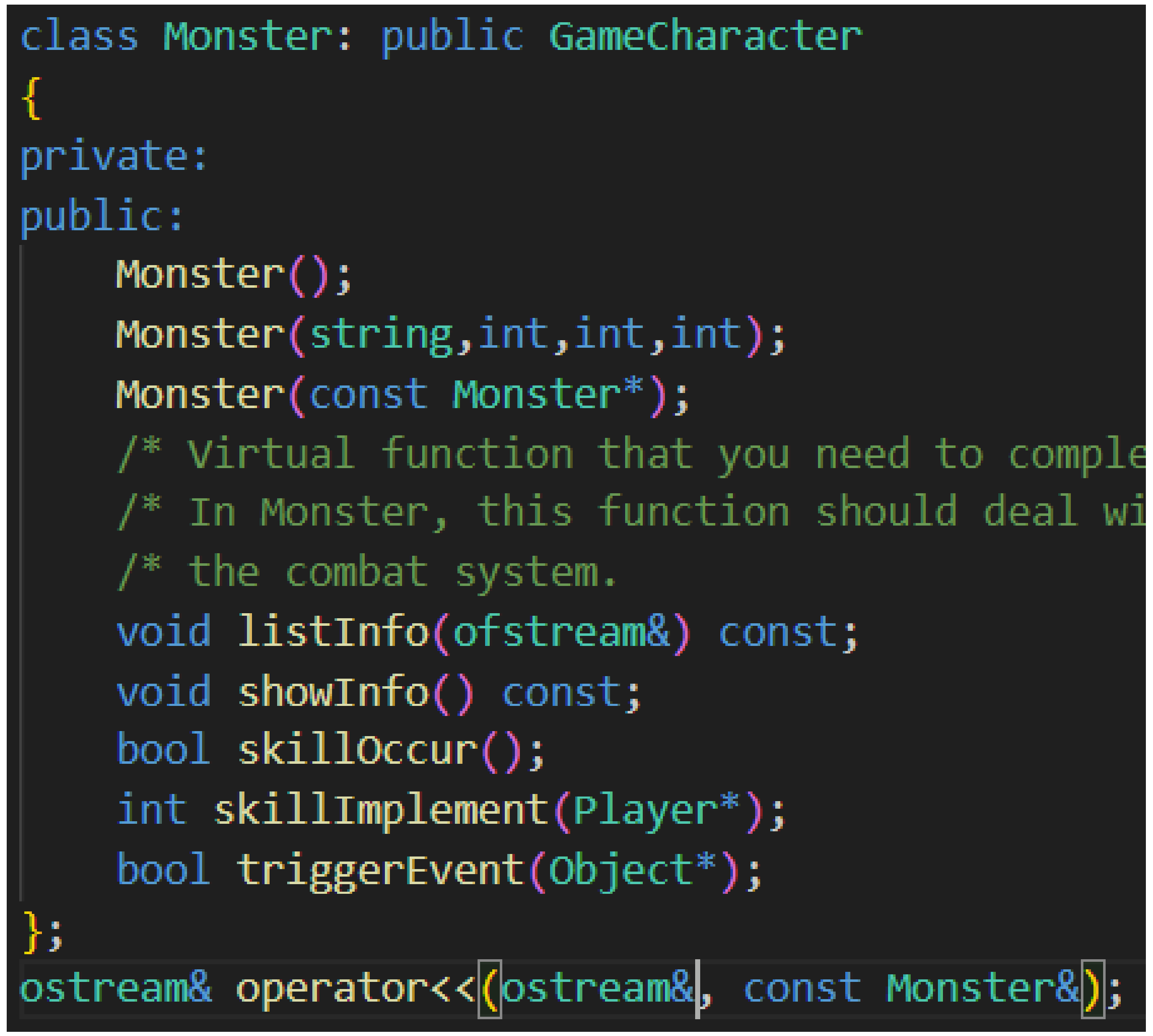
### 特殊函式介紹

void listInfo(ofstream&) const:
於Record處詳細解釋

bool skillOccur():
用rand()來隨機決定是否要執行skillImplement()

int skillImplement():
跟player類似, 不過是依照本身名字決定技能並發動並執行(對player), return改變的傷害值

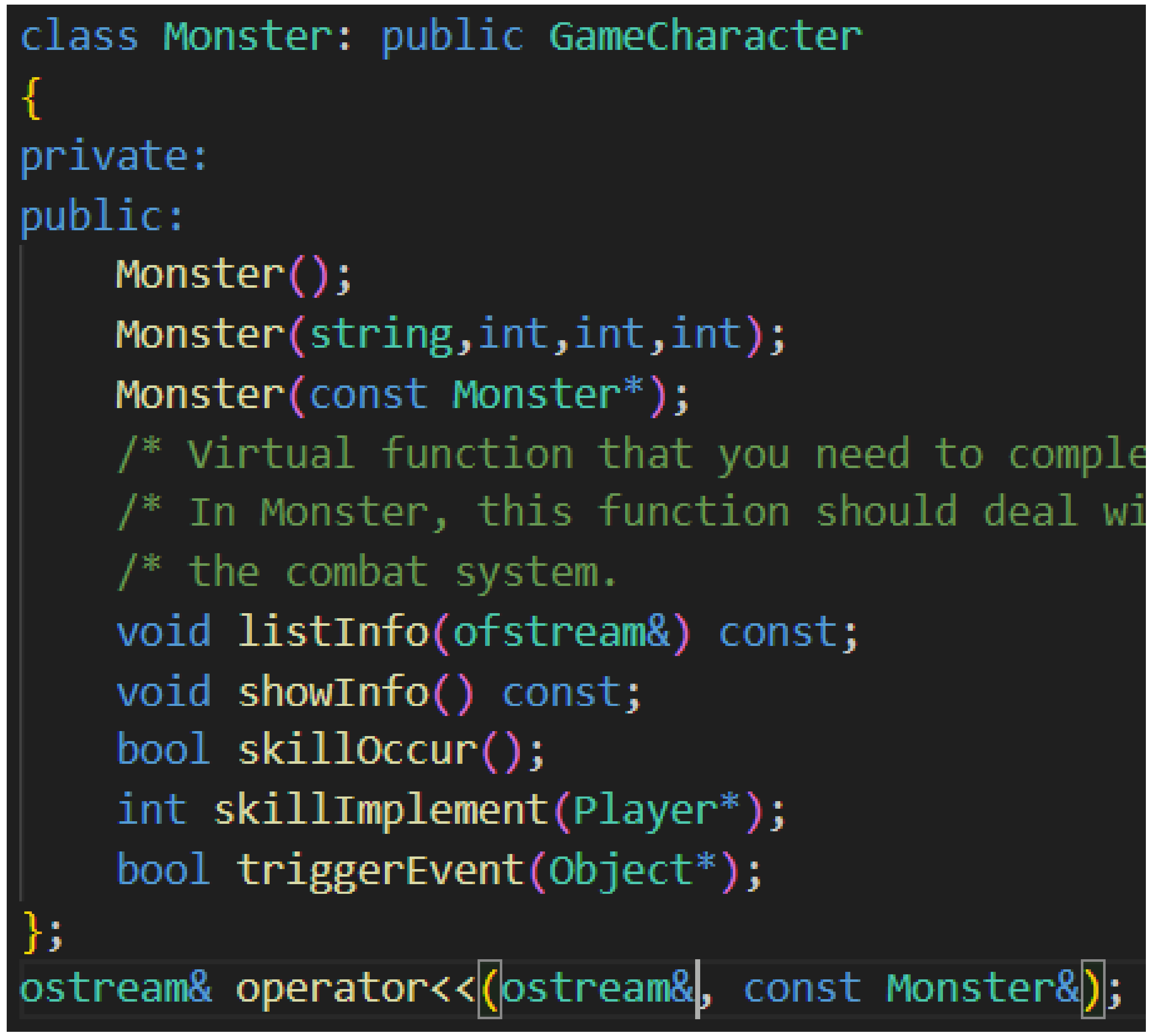
(下頁繼續)



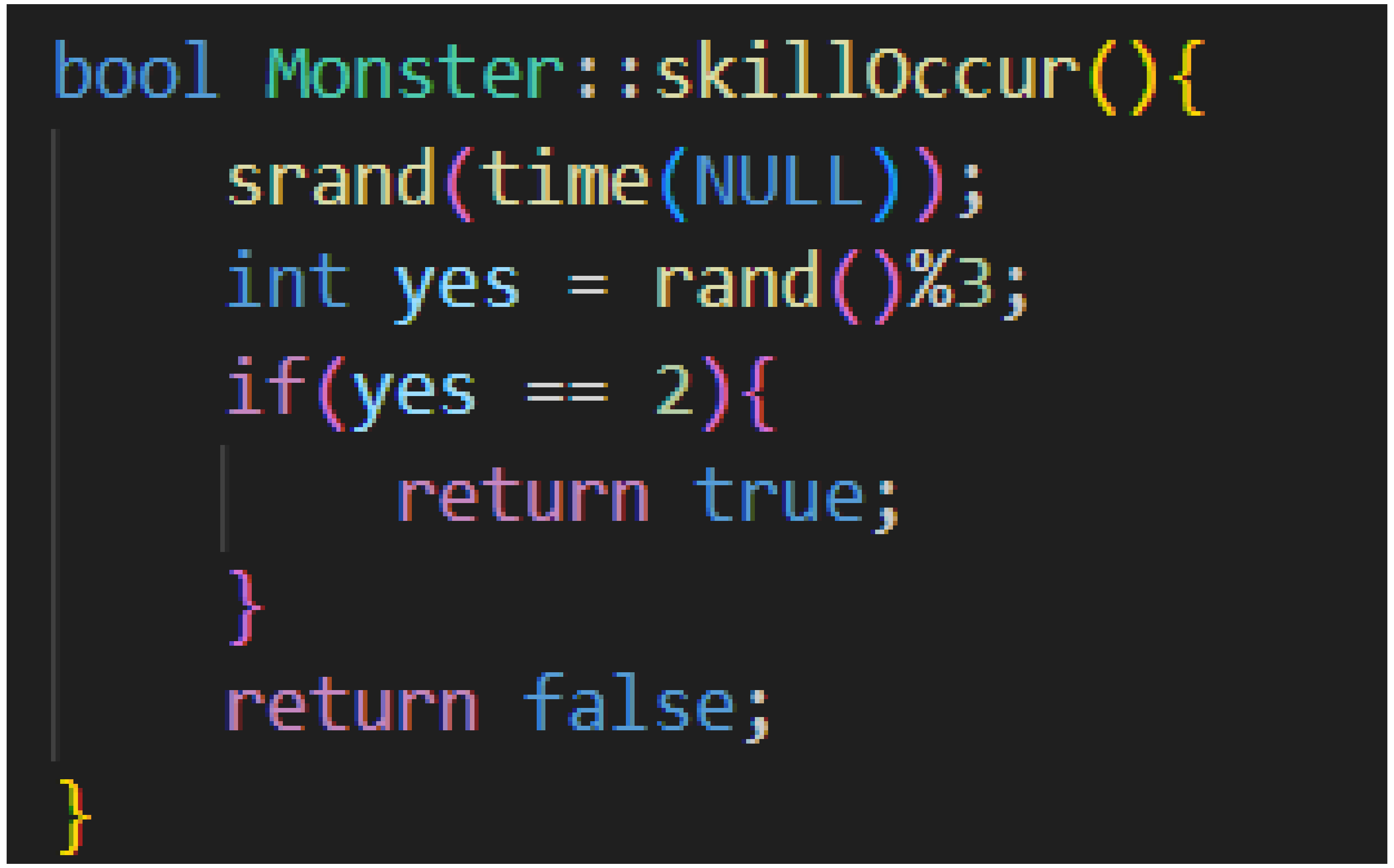
```
class Monster: public GameCharacter
{
private:
public:
    Monster();
    Monster(string,int,int,int);
    Monster(const Monster*);
    /* Virtual function that you need to comple
    /* In Monster, this function should deal wi
    /* the combat system.
    void listInfo(ofstream&) const;
    void showInfo() const;
    bool skillOccur();
    int skillImplement(Player*);
    bool triggerEvent(Object*);
};
ostream& operator<<(ostream&, const Monster&);
```

bool triggerEvent(Object*):
為class Object中virtual函式的override, 用來執行進攻/撤退, 處理Monster攻擊Player和Player攻擊Monster的情況(包括技能發動等)

p.s. operator overload輸出型式見detailed implementation


```
class Monster: public GameCharacter
{
private:
public:
    Monster();
    Monster(string,int,int,int);
    Monster(const Monster*);
    /* Virtual function that you need to comple
    /* In Monster, this function should deal wi
    /* the combat system.
    void listInfo(ofstream&) const;
    void showInfo() const;
    bool skillOccur();
    int skillImplement(Player*);
    bool triggerEvent(Object*);
};
ostream& operator<<(ostream&, const Monster&);
```

## Detailed Implementation

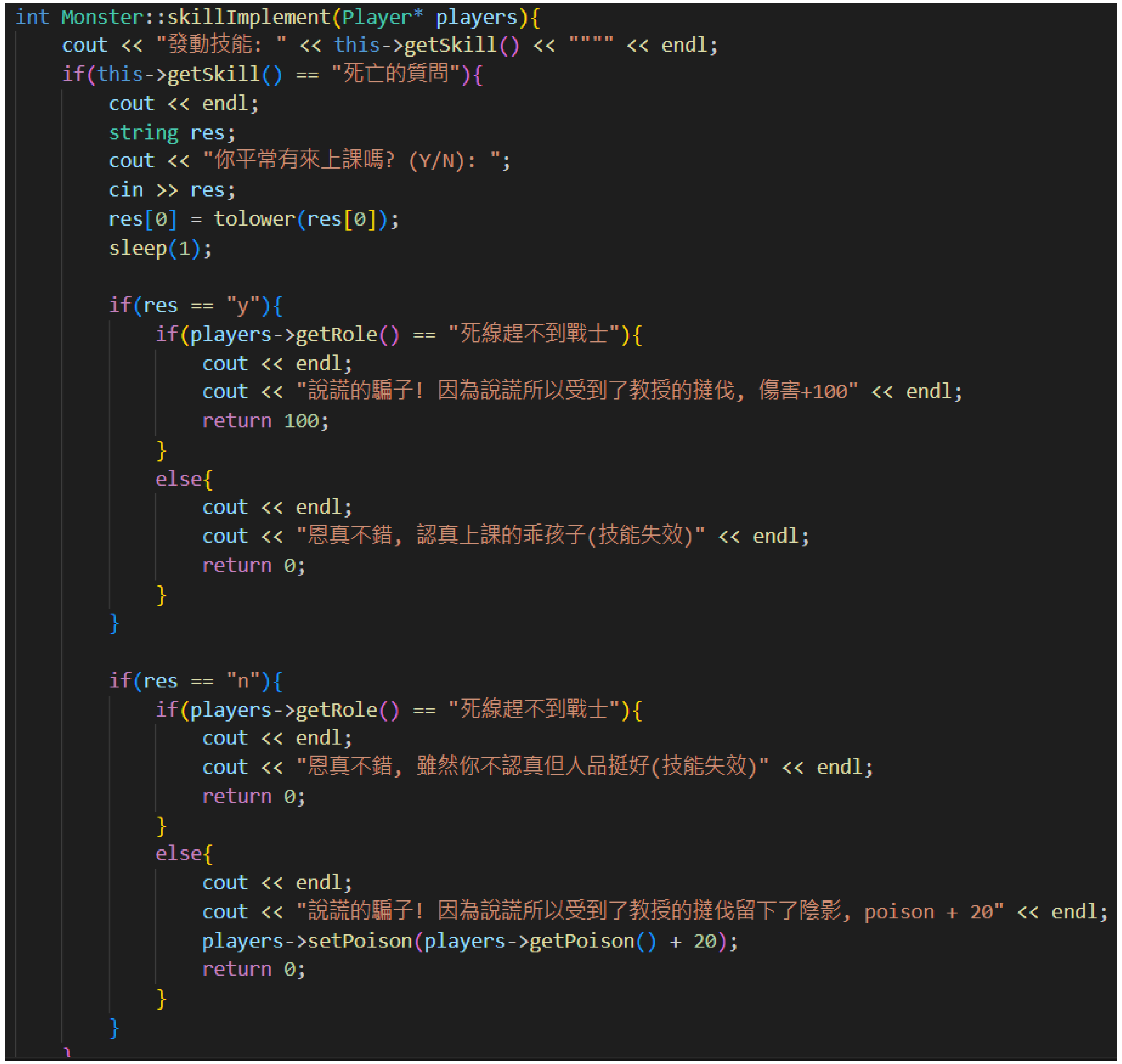


```
bool Monster::skillOccur(){
    srand(time(NULL));
    int yes = rand()%3;
    if(yes == 2){
        return true;
    }
    return false;
}
```

# Detailed Implementation

int skillImplement(Player*);

這裡只列一個技能當例子

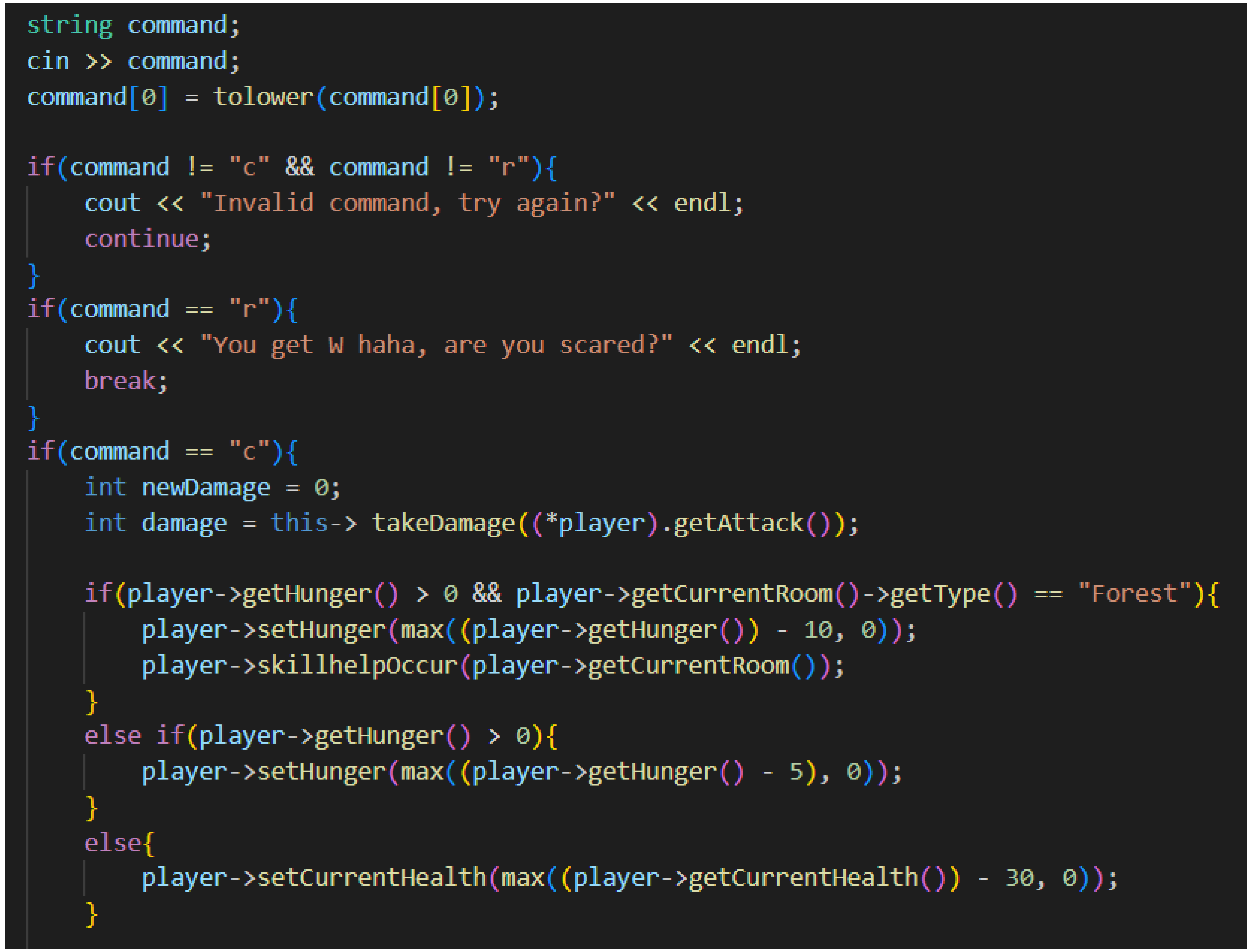


```cpp
int Monster::skillImplement(Player* players){
    cout << "發動技能: " << this->getSkill() << """" << endl;
    if(this->getSkill() == "死亡的質問"){
        cout << endl;
        string res;
        cout << "你平常有來上課嗎? (Y/N): ";
        cin >> res;
        res[0] = tolower(res[0]);
        sleep(1);

        if(res == "y"){
            if(players->getRole() == "死線趕不到戰士"){
                cout << endl;
                cout << "說謊的騙子! 因為說謊所以受到了教授的撻伐, 傷害+100" << endl;
                return 100;
            }
            else{
                cout << endl;
                cout << "恩真不錯, 認真上課的乖孩子(技能失效)" << endl;
                return 0;
            }
        }

        if(res == "n"){
            if(players->getRole() == "死線趕不到戰士"){
                cout << endl;
                cout << "恩真不錯, 雖然你不認真但人品挺好(技能失效)" << endl;
                return 0;
            }
            else{
                cout << endl;
                cout << "說謊的騙子! 因為說謊所以受到了教授的撻伐留下了陰影, poison + 20" << endl;
                players->setPoison(players->getPoison() + 20);
                return 0;
            }
        }
```

# Detailed Implementation

bool triggerEvent(Object*);
(下頁還有)

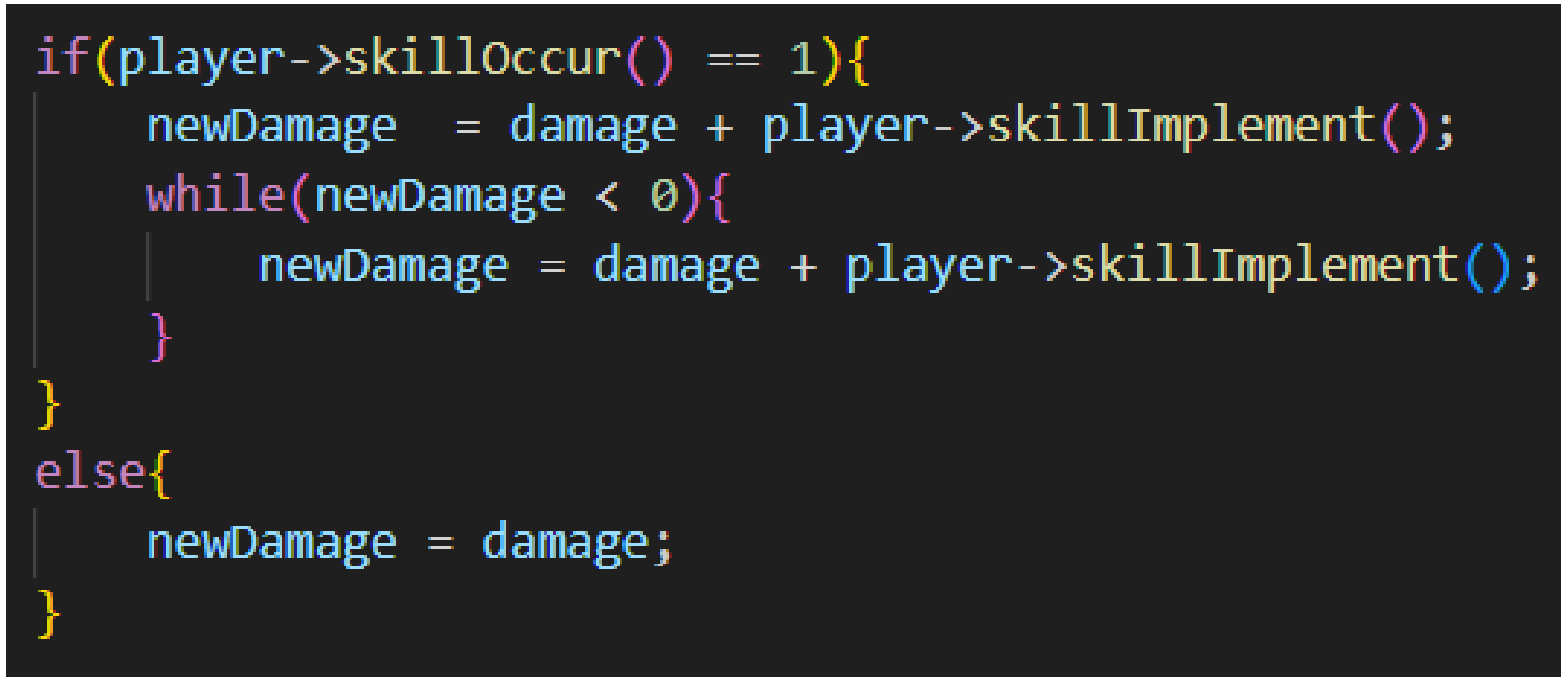
這裡只列一個技能當例子



```cpp
string command;
cin >> command;
command[0] = tolower(command[0]);

if(command != "c" && command != "r"){
    cout << "Invalid command, try again?" << endl;
    continue;
}
if(command == "r"){
    cout << "You get W haha, are you scared?" << endl;
    break;
}
if(command == "c"){
    int newDamage = 0;
    int damage = this-> takeDamage((*player).getAttack());

    if(player->getHunger() > 0 && player->getCurrentRoom()->getType() == "Forest"){
        player->setHunger(max((player->getHunger()) - 10, 0));
        player->skillhelpOccur(player->getCurrentRoom());
    }
    else if(player->getHunger() > 0){
        player->setHunger(max((player->getHunger() - 5), 0));
    }
    else{
        player->setCurrentHealth(max((player->getCurrentHealth()) - 30, 0));
    }
```

## Detailed Implementation

bool triggerEvent(Object*);
(下頁還有)

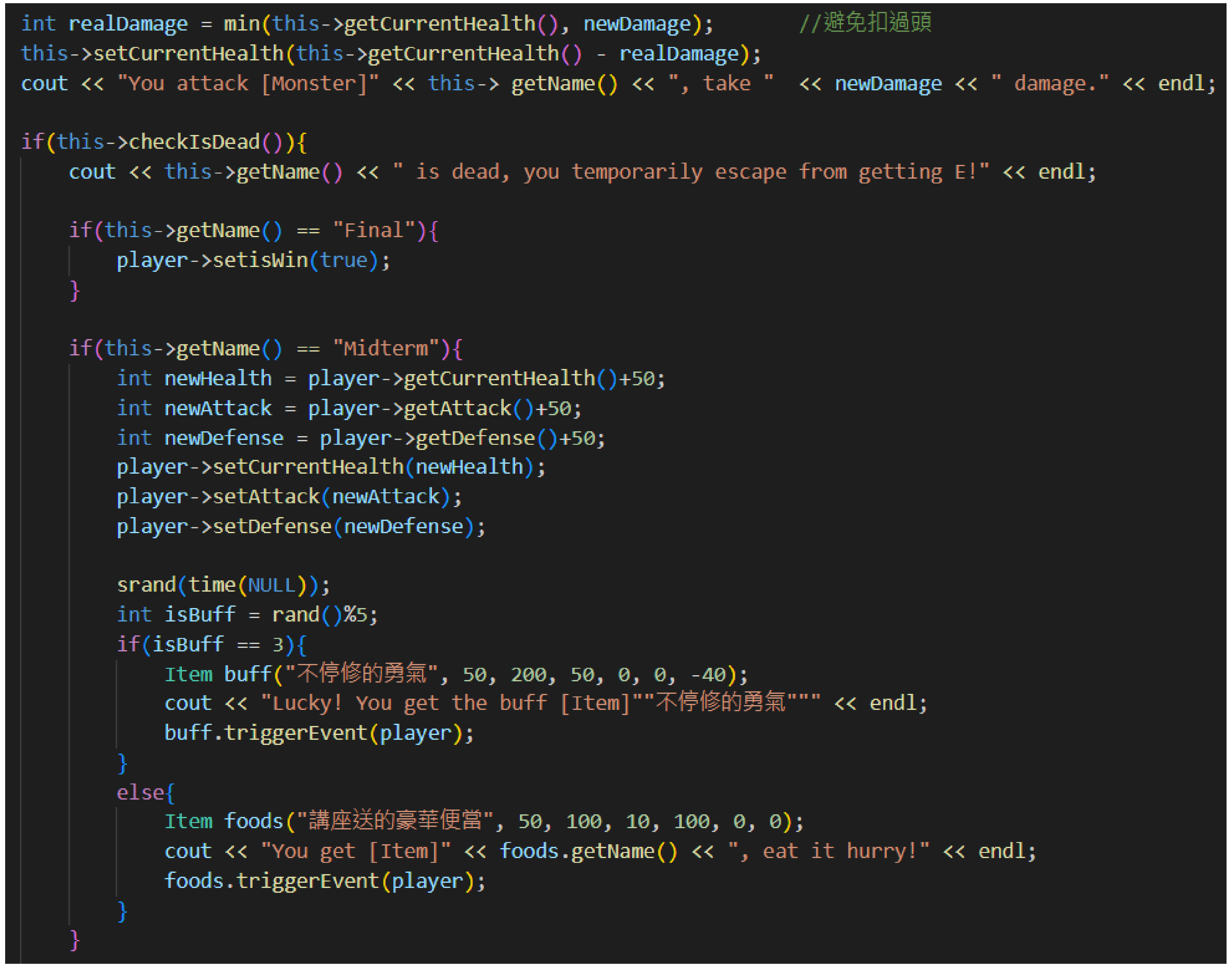


```
if(player->skillOccur() == 1){
    newDamage  = damage + player->skillImplement();
    while(newDamage < 0){
        newDamage = damage + player->skillImplement();
    }
}
else{
    newDamage = damage;
}
```

## Detailed Implementation

bool triggerEvent(Object*);
(Player攻擊Monster的情況)
(下頁還有)

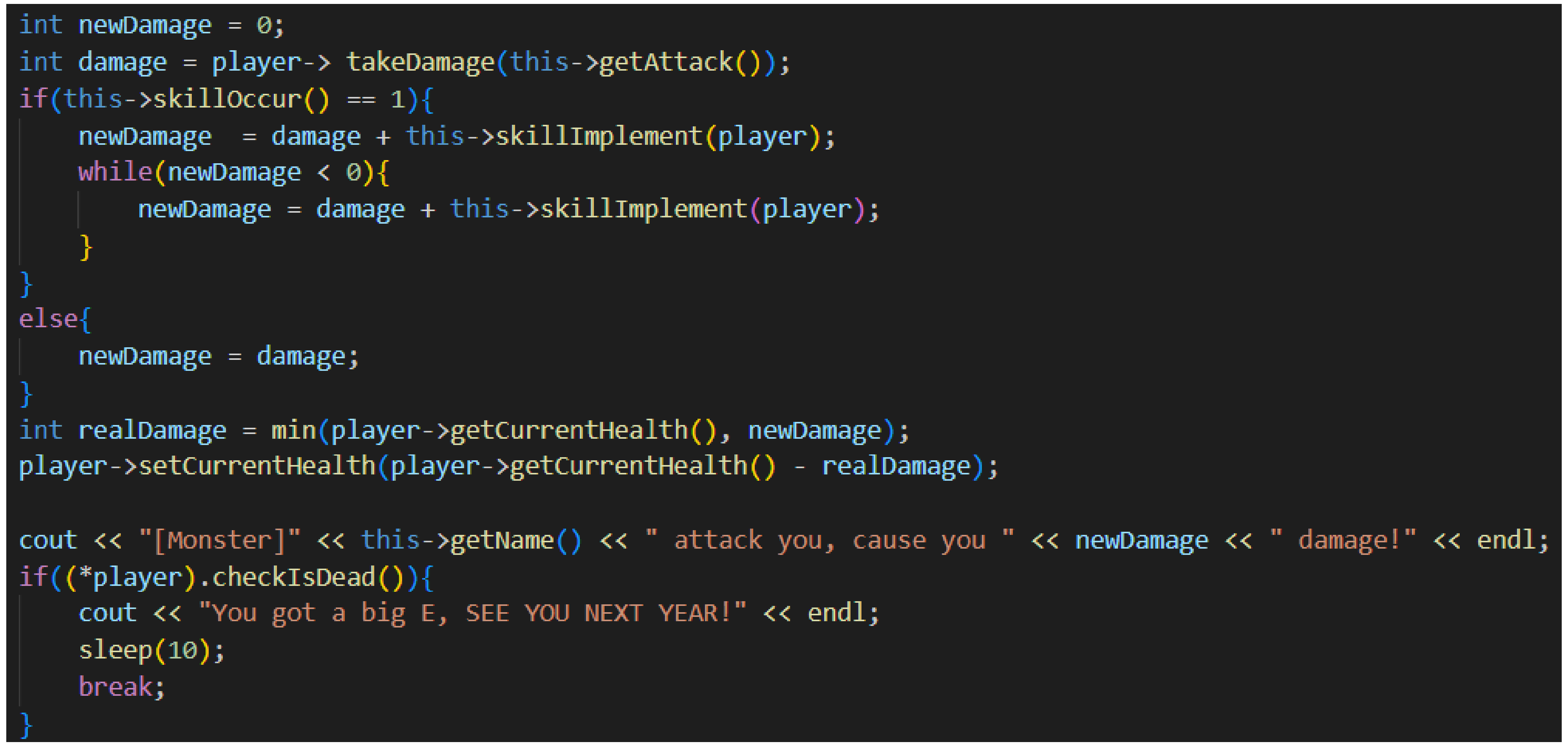
這裡只列一個技能當例子



```cpp
int realDamage = min(this->getCurrentHealth(), newDamage);       //避免扣過頭
this->setCurrentHealth(this->getCurrentHealth() - realDamage);
cout << "You attack [Monster]" << this-> getName() << ", take "  << newDamage << " damage." << endl;

if(this->checkIsDead()){
    cout << this->getName() << " is dead, you temporarily escape from getting E!" << endl;

    if(this->getName() == "Final"){
        player->setisWin(true);
    }

    if(this->getName() == "Midterm"){
        int newHealth = player->getCurrentHealth()+50;
        int newAttack = player->getAttack()+50;
        int newDefense = player->getDefense()+50;
        player->setCurrentHealth(newHealth);
        player->setAttack(newAttack);
        player->setDefense(newDefense);

        srand(time(NULL));
        int isBuff = rand()%5;
        if(isBuff == 3){
            Item buff("不停修的勇氣", 50, 200, 50, 0, 0, -40);
            cout << "Lucky! You get the buff [Item]""不停修的勇氣""" << endl;
            buff.triggerEvent(player);
        }
        else{
            Item foods("講座送的豪華便當", 50, 100, 10, 100, 0, 0);
            cout << "You get [Item]" << foods.getName() << ", eat it hurry!" << endl;
            foods.triggerEvent(player);
        }
    }
```

## Detailed Implementation

bool triggerEvent(Object*);
(Monster攻擊Player的情況)



```cpp
int newDamage = 0;
int damage = player-> takeDamage(this->getAttack());
if(this->skillOccur() == 1){
    newDamage  = damage + this->skillImplement(player);
    while(newDamage < 0){
        newDamage = damage + this->skillImplement(player);
    }
}
else{
    newDamage = damage;
}
int realDamage = min(player->getCurrentHealth(), newDamage);
player->setCurrentHealth(player->getCurrentHealth() - realDamage);

cout << "[Monster]" << this->getName() << " attack you, cause you " << newDamage << " damage!" << endl;
if((*player).checkIsDead()){
    cout << "You got a big E, SEE YOU NEXT YEAR!" << endl;
    sleep(10);
    break;
}
```

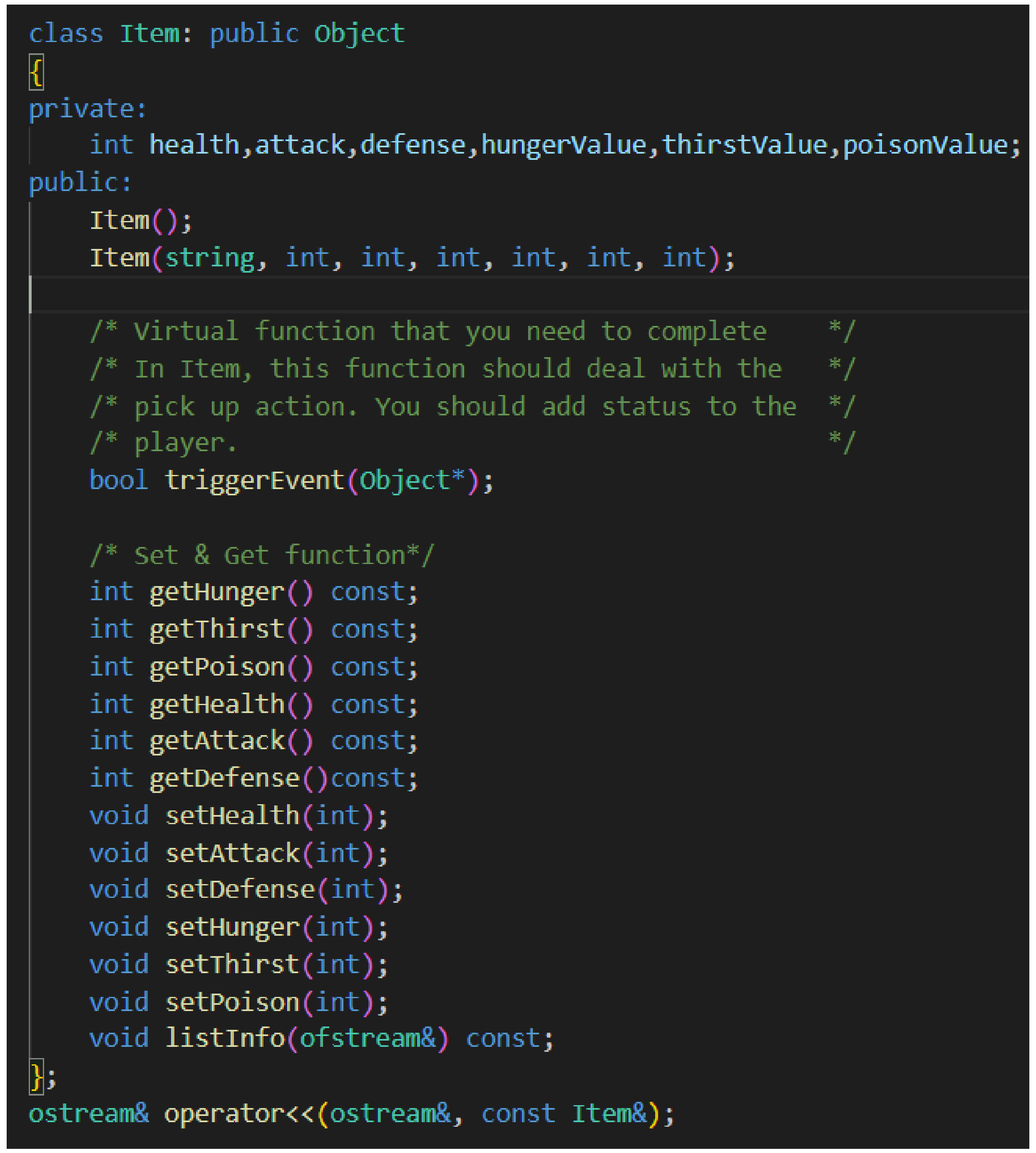
這裡只列一個技能當例子

## Item

health/attack/defense/
hungerValue/thirstValue/poisonValue:
當player拿到item時, 會增加/減少的值

### 特殊函式介紹

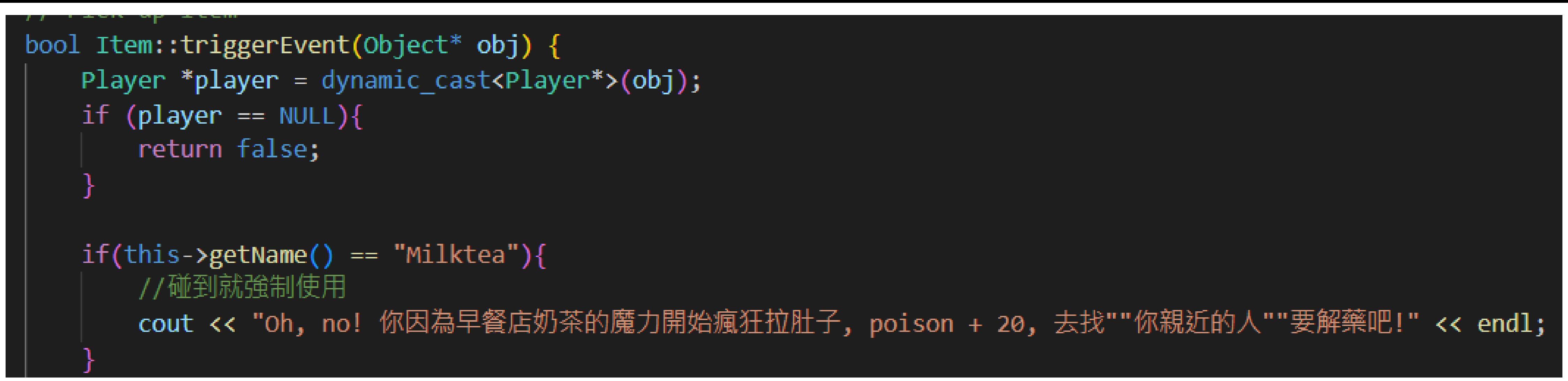
bool triggerEvent(Object*):
為class Object中virtual函式的override, 在NPC的triggerEvent(Object*)中被引用, 用來執行跟NPC拿東西時的情況, 東西都被拿完的話return true, 反之return false
p.s. operator overload輸出型式
見detailed implementation



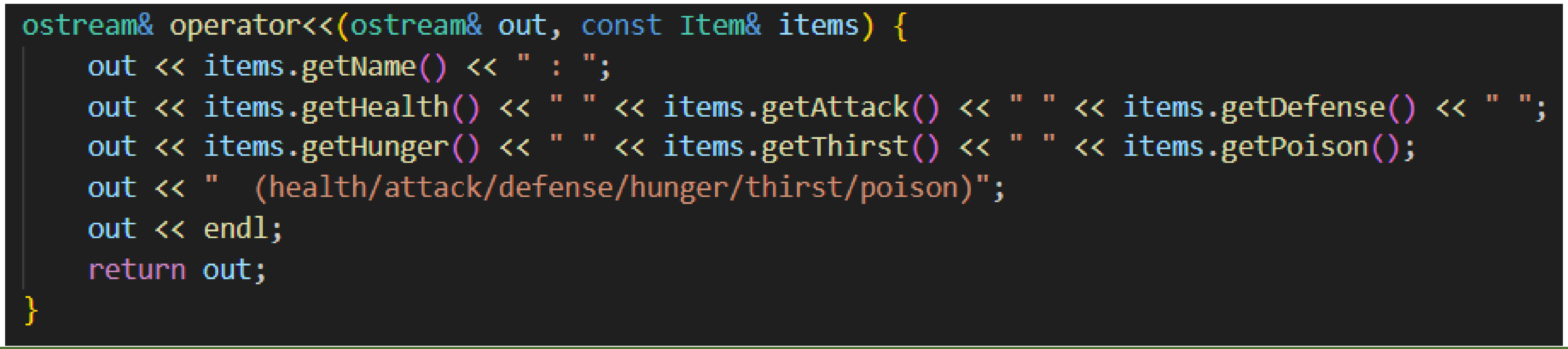
```cpp
class Item: public Object
{
private:
    int health,attack,defense,hungerValue,thirstValue,poisonValue;
public:
    Item();
    Item(string, int, int, int, int, int, int);

    /* Virtual function that you need to complete    */
    /* In Item, this function should deal with the   */
    /* pick up action. You should add status to the  */
    /* player.                                       */
    bool triggerEvent(Object*);

    /* Set & Get function*/
    int getHunger() const;
    int getThirst() const;
    int getPoison() const;
    int getHealth() const;
    int getAttack() const;
    int getDefense()const;
    void setHealth(int);
    void setAttack(int);
    void setDefense(int);
    void setHunger(int);
    void setThirst(int);
    void setPoison(int);
    void listInfo(ofstream&) const;
};
ostream& operator<<(ostream&, const Item&);
```

# Detailed Implementation

這裡只列一個名字當例子



```cpp
bool Item::triggerEvent(Object* obj) {
    Player *player = dynamic_cast<Player*>(obj);
    if (player == NULL){
        return false;
    }

    if(this->getName() == "Milktea"){
        //碰到就強制使用
        cout << "Oh, no! 你因為早餐店奶茶的魔力開始瘋狂拉肚子, poison + 20, 去找""你親近的人""要解藥吧!" << endl;
    }
```



```cpp
ostream& operator<<(ostream& out, const Item& items) {
    out << items.getName() << " : ";
    out << items.getHealth() << " " << items.getAttack() << " " << items.getDefense() << " ";
    out << items.getHunger() << " " << items.getThirst() << " " << items.getPoison();
    out << "  (health/attack/defense/hunger/thirst/poison)";
    out << endl;
    return out;
}
```

## Record

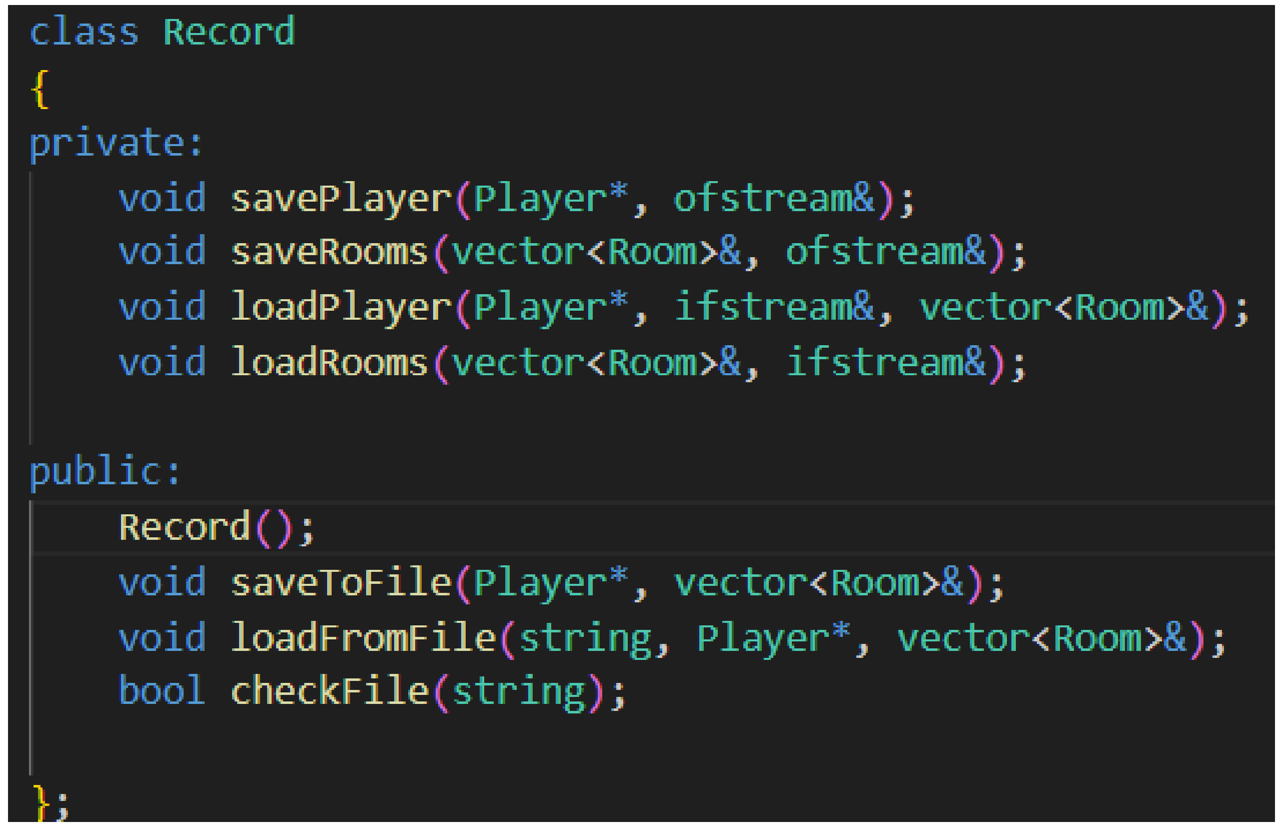
### 特殊函式介紹

void listInfo(ofstream&) const:
前面一直出現, 其實是用來把資料存入到ofsteam& out中，會出現在savePlayer(Player*, ofsteam&)和saveRooms(vector<Room>&, ofsteam&)中

void savePlayer(Player*, ofsteam&):
用來把資料存入到ofsteam& out中

void saveRooms(vector<Room>&, ofsteam&):
用來把資料存入到ofsteam& out中

(下頁繼續)



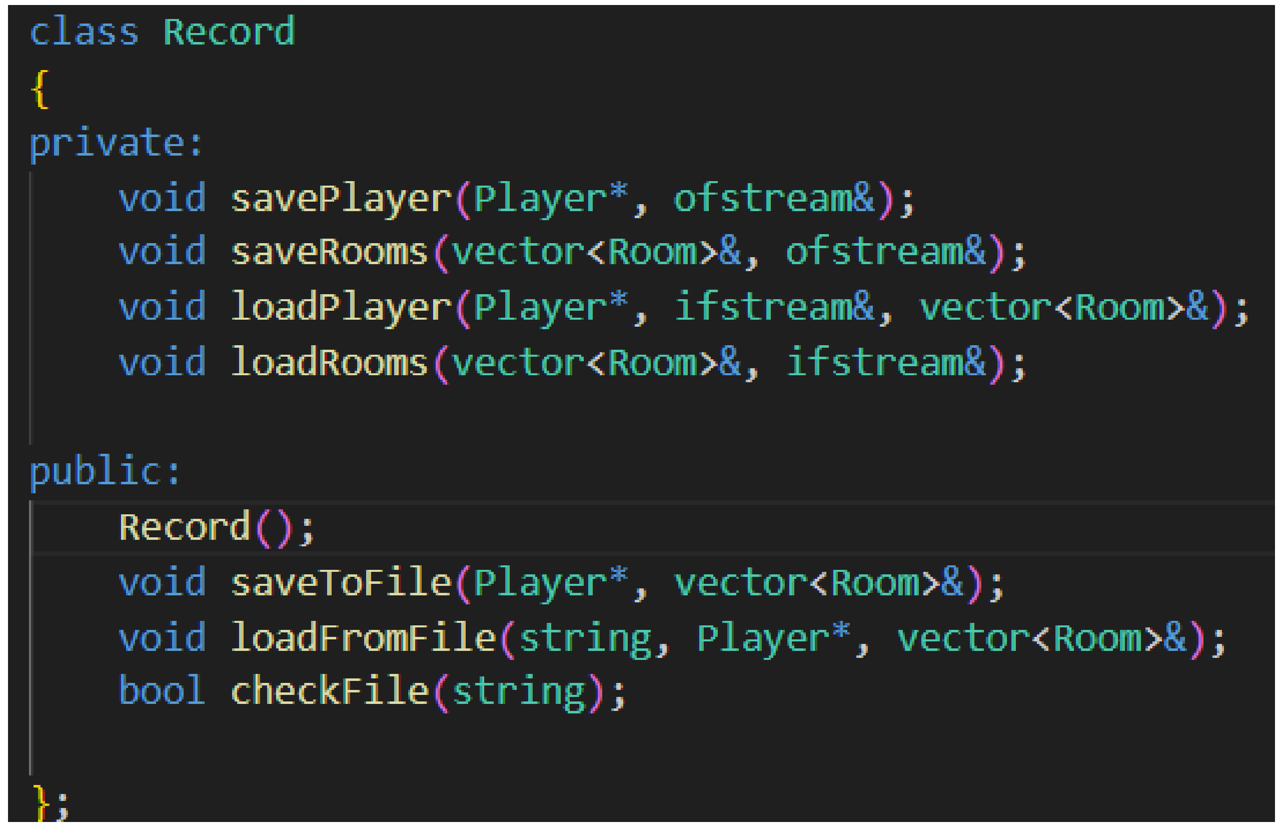
```
class Record
{
private:
    void savePlayer(Player*, ofstream&);
    void saveRooms(vector<Room>&, ofstream&);
    void loadPlayer(Player*, ifstream&, vector<Room>&);
    void loadRooms(vector<Room>&, ifstream&);

public:
    Record();
    void saveToFile(Player*, vector<Room>&);
    void loadFromFile(string, Player*, vector<Room>&);
    bool checkFile(string);

};
```

void loadPlayer(Player*, ifsteam&, vector<Room>&):
利用ifstream& in, 將原來被out到txt檔的資料讀入並再次生成Player相關的物件並組合

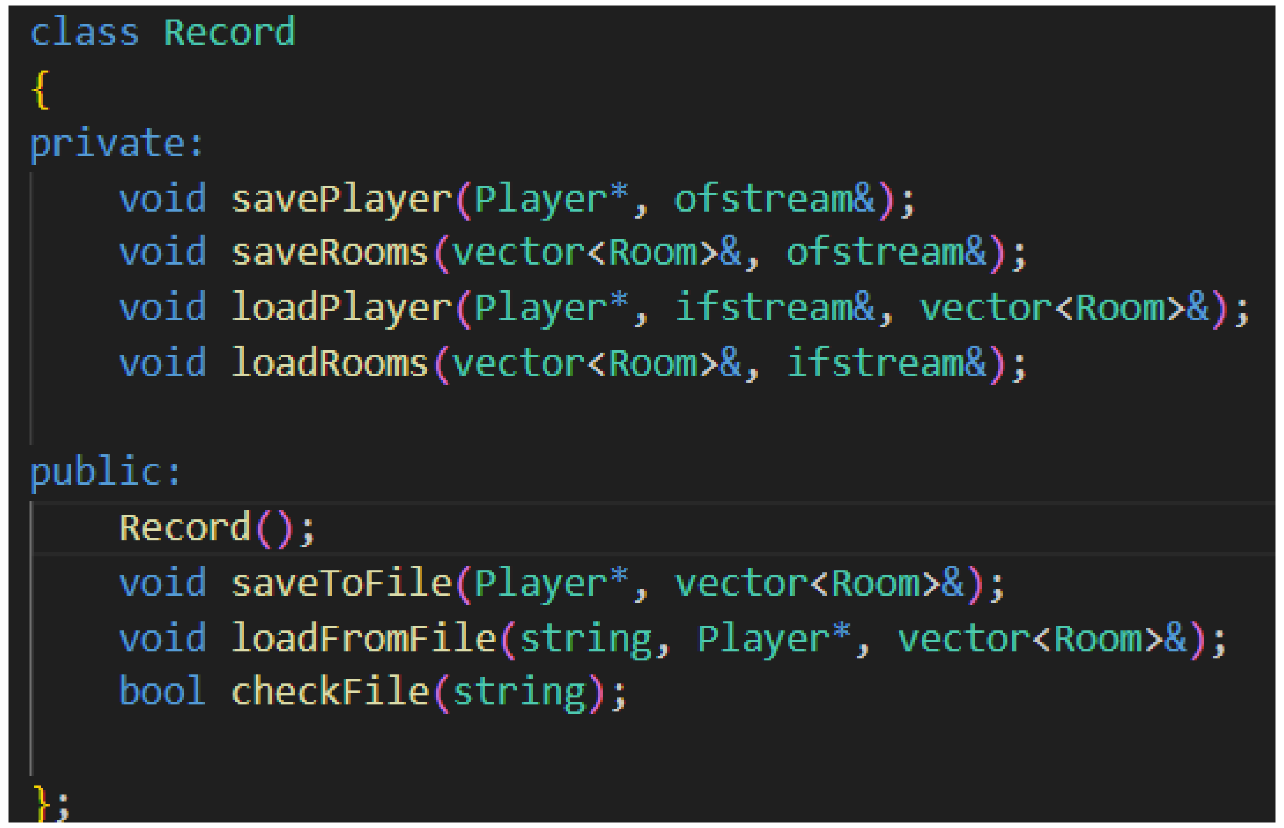
void loadRooms(vector<Room>&, ifsteam&):
利用ifstream& in, 將原來被out到txt檔的資料讀入並再次生成Room相關的物件並組合

void saveToFile(Player*, vector<Room>&):
生成一個檔案, 並把存到ofsteam& out中的資料放到檔案中

void loadFromFile(string, Player*, vector<Room>&):
找到資料, 將檔案中的資料讀取, 並結合loadPlayer跟loadRooms來重新生成保存的資料物件
(下頁繼續)

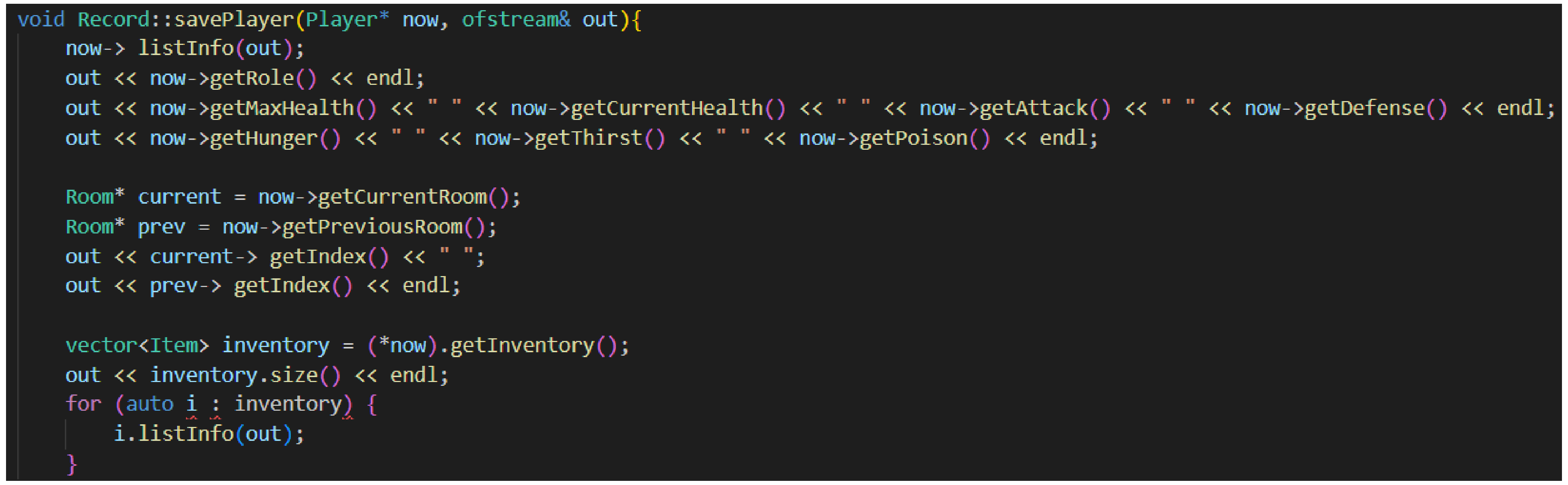

```
class Record
{
private:
    void savePlayer(Player*, ofstream&);
    void saveRooms(vector<Room>&, ofstream&);
    void loadPlayer(Player*, ifstream&, vector<Room>&);
    void loadRooms(vector<Room>&, ifstream&);

public:
    Record();
    void saveToFile(Player*, vector<Room>&);
    void loadFromFile(string, Player*, vector<Room>&);
    bool checkFile(string);

};
```

bool checkFile(string):
測試是否有讀取資料 成功

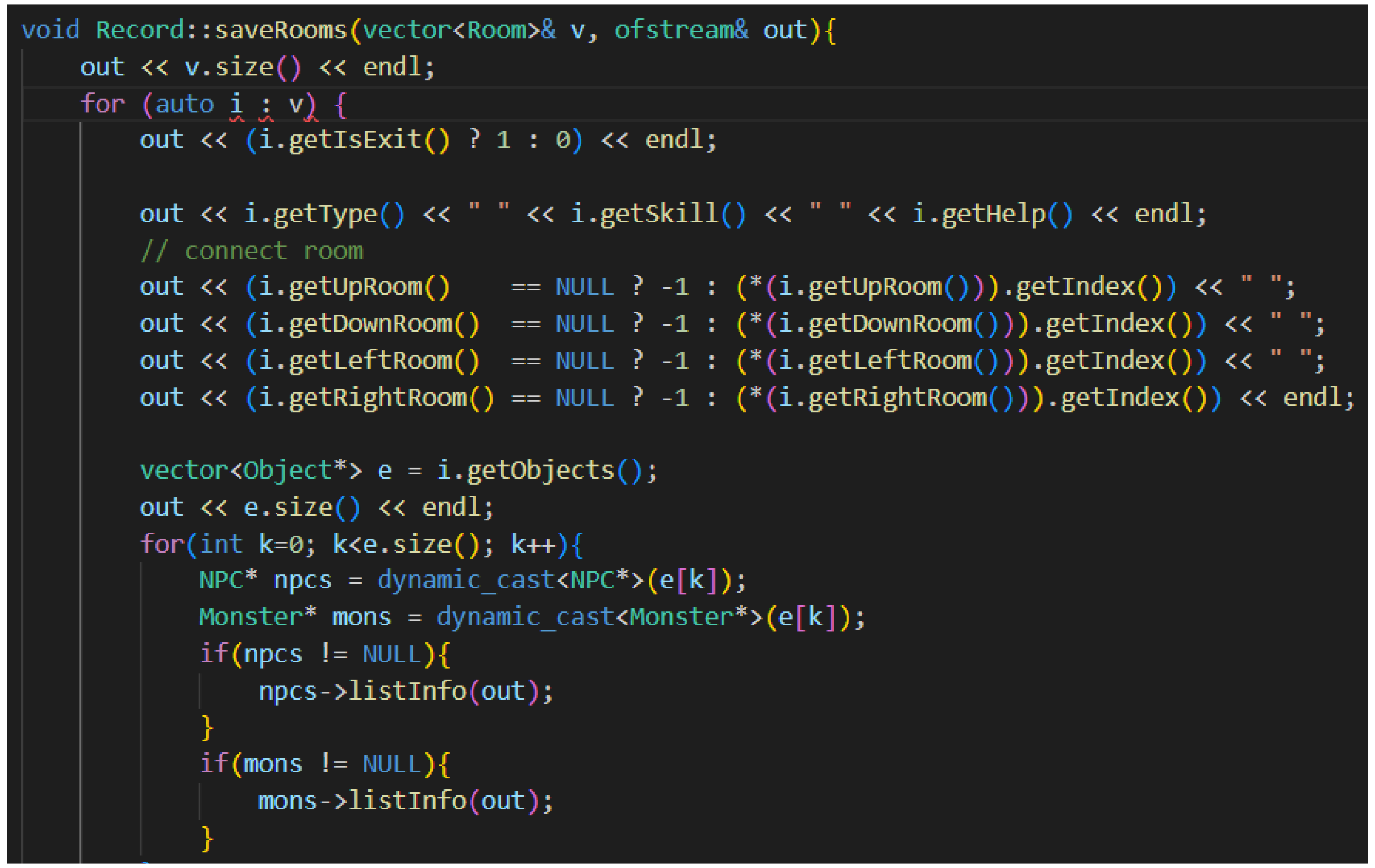


```
class Record
{
private:
    void savePlayer(Player*, ofstream&);
    void saveRooms(vector<Room>&, ofstream&);
    void loadPlayer(Player*, ifstream&, vector<Room>&);
    void loadRooms(vector<Room>&, ifstream&);

public:
    Record();
    void saveToFile(Player*, vector<Room>&);
    void loadFromFile(string, Player*, vector<Room>&);
    bool checkFile(string);

};
```

## Detailed Implementation



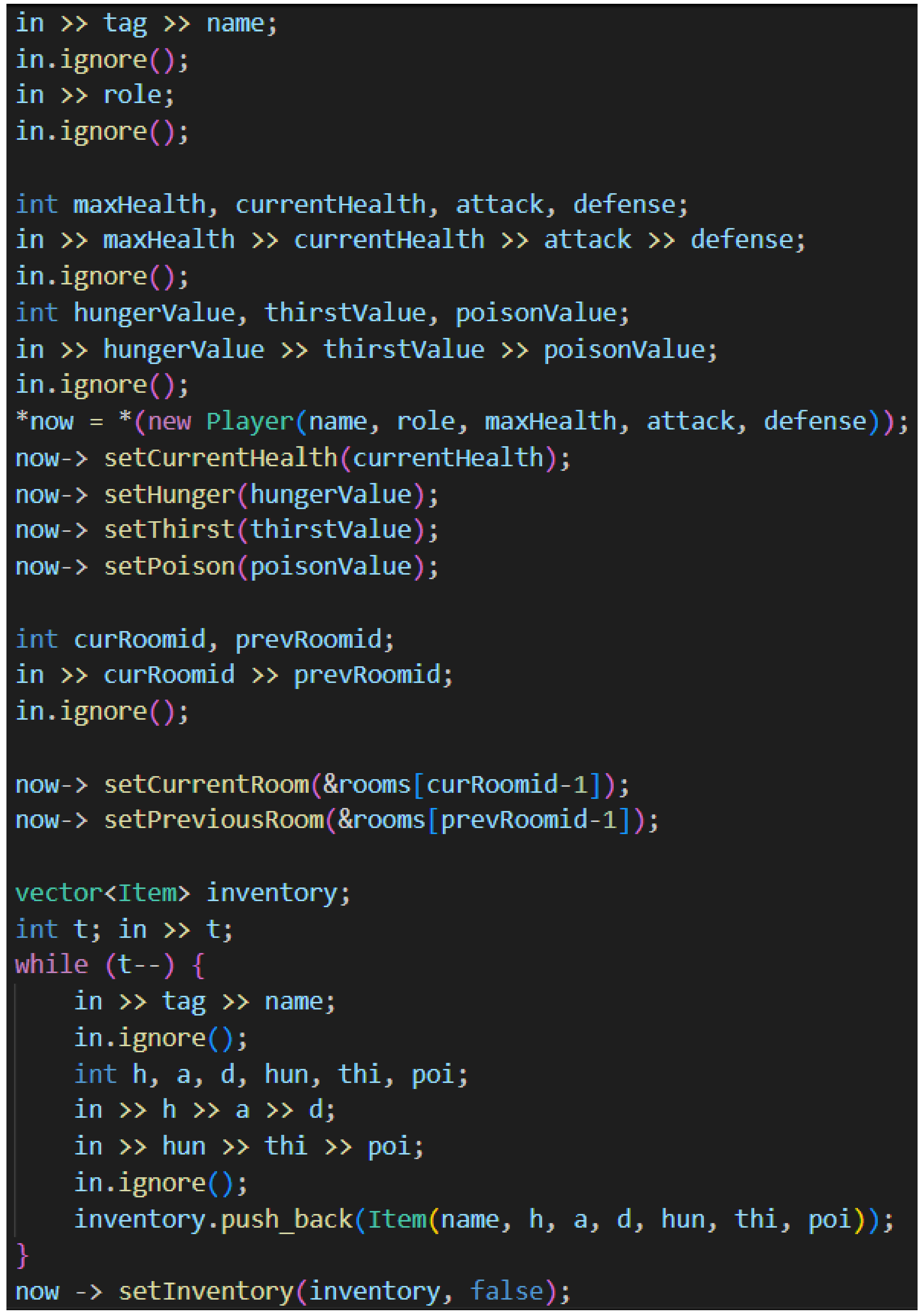
```cpp
void Record::savePlayer(Player* now, ofstream& out){
    now-> listInfo(out);
    out << now->getRole() << endl;
    out << now->getMaxHealth() << " " << now->getCurrentHealth() << " " << now->getAttack() << " " << now->getDefense() << endl;
    out << now->getHunger() << " " << now->getThirst() << " " << now->getPoison() << endl;

    Room* current = now->getCurrentRoom();
    Room* prev = now->getPreviousRoom();
    out << current-> getIndex() << " ";
    out << prev-> getIndex() << endl;

    vector<Item> inventory = (*now).getInventory();
    out << inventory.size() << endl;
    for (auto i : inventory) {
        i.listInfo(out);
    }
```



```cpp
void Record::saveRooms(vector<Room>& v, ofstream& out){
    out << v.size() << endl;
    for (auto i : v) {
        out << (i.getIsExit() ? 1 : 0) << endl;

        out << i.getType() << " " << i.getSkill() << " " << i.getHelp() << endl;
        // connect room
        out << (i.getUpRoom()    == NULL ? -1 : (*(i.getUpRoom())).getIndex()) << " ";
        out << (i.getDownRoom()  == NULL ? -1 : (*(i.getDownRoom())).getIndex()) << " ";
        out << (i.getLeftRoom()  == NULL ? -1 : (*(i.getLeftRoom())).getIndex()) << " ";
        out << (i.getRightRoom() == NULL ? -1 : (*(i.getRightRoom())).getIndex()) << endl;

        vector<Object*> e = i.getObjects();
        out << e.size() << endl;
        for(int k=0; k<e.size(); k++){
            NPC* npcs = dynamic_cast<NPC*>(e[k]);
            Monster* mons = dynamic_cast<Monster*>(e[k]);
            if(npcs != NULL){
                npcs->listInfo(out);
            }
            if(mons != NULL){
                mons->listInfo(out);
            }
```

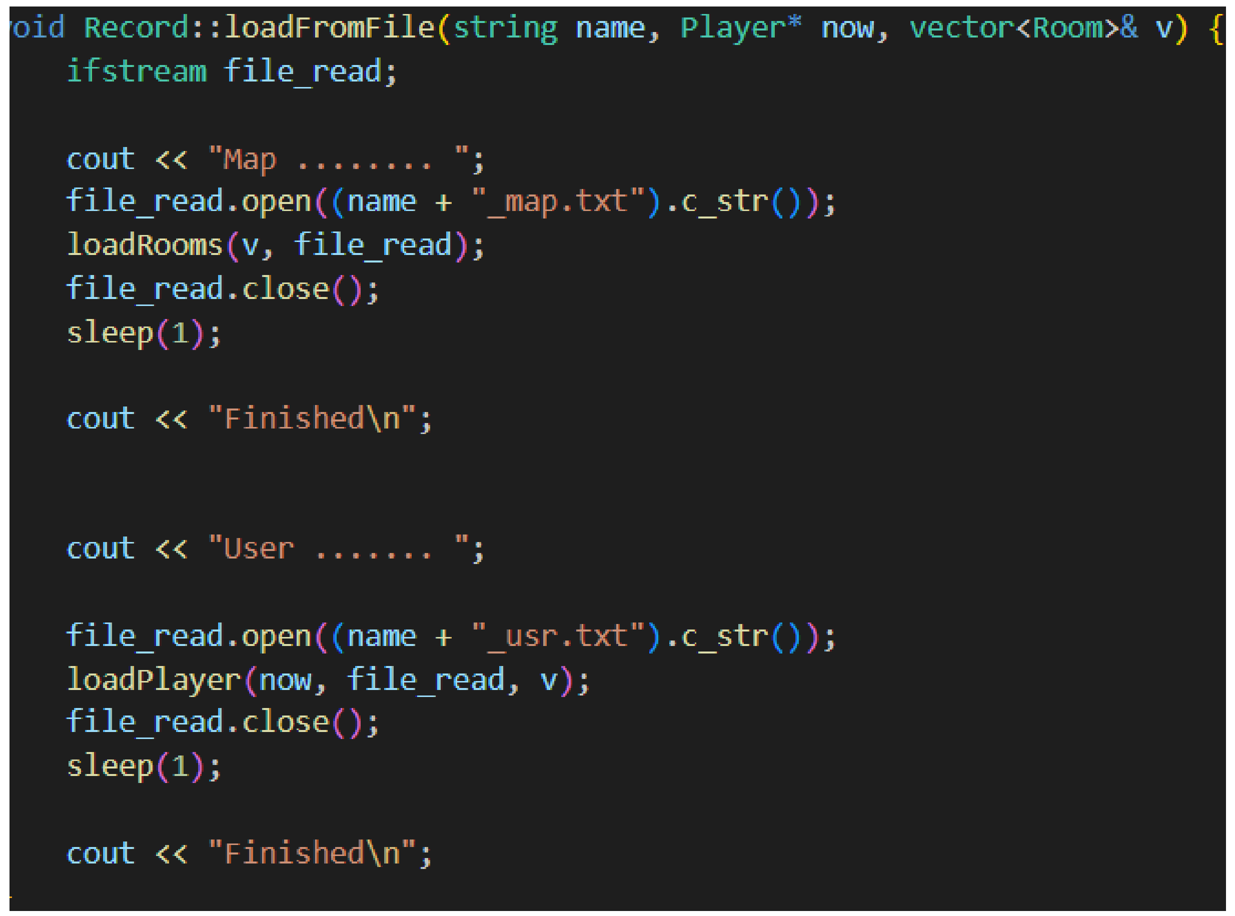
## Detailed Implementation

void loadPlayer
(Player*, ifsteam&, vector<Room>&)
loadRooms方法同loadPlayer

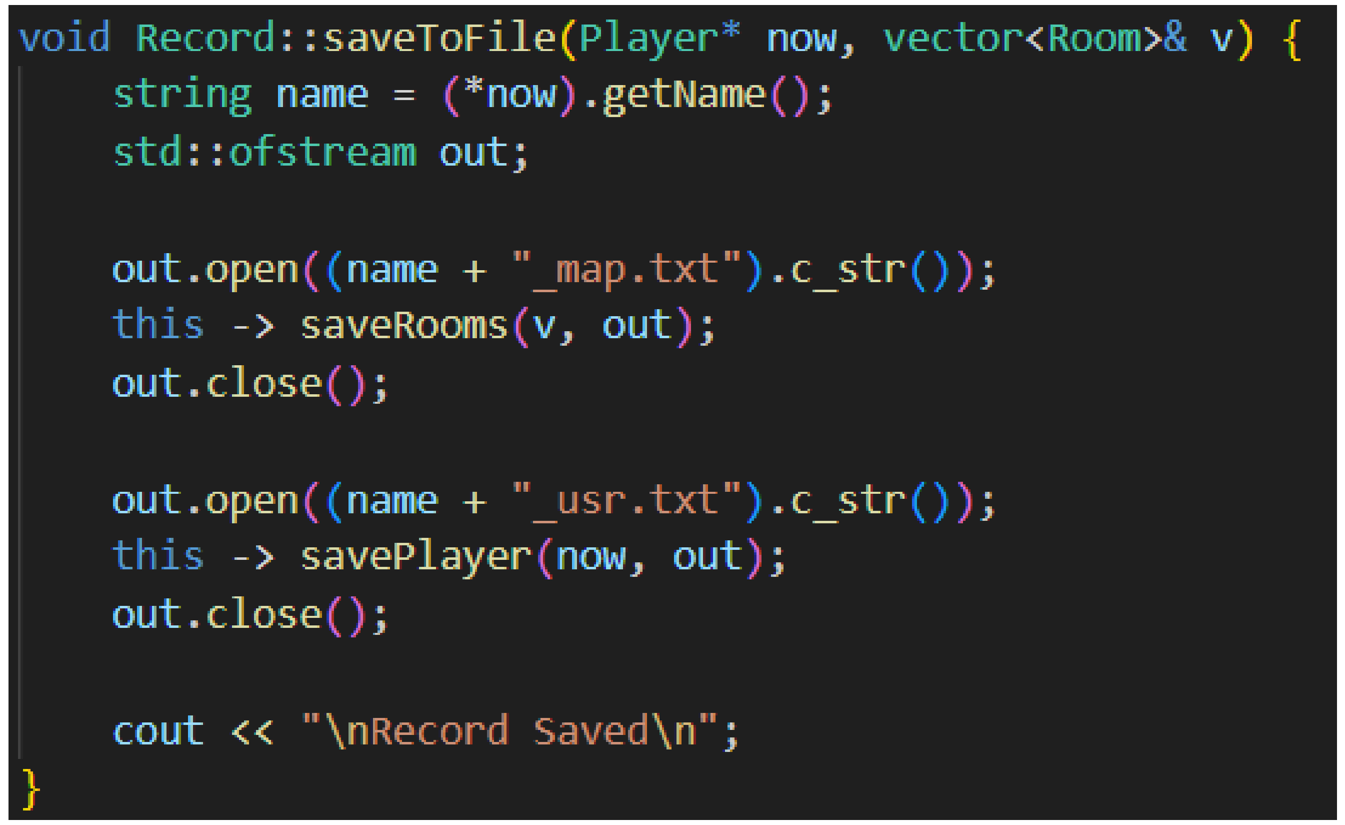


```
in >> tag >> name;
in.ignore();
in >> role;
in.ignore();

int maxHealth, currentHealth, attack, defense;
in >> maxHealth >> currentHealth >> attack >> defense;
in.ignore();
int hungerValue, thirstValue, poisonValue;
in >> hungerValue >> thirstValue >> poisonValue;
in.ignore();
*now = *(new Player(name, role, maxHealth, attack, defense));
now-> setCurrentHealth(currentHealth);
now-> setHunger(hungerValue);
now-> setThirst(thirstValue);
now-> setPoison(poisonValue);

int curRoomid, prevRoomid;
in >> curRoomid >> prevRoomid;
in.ignore();

now-> setCurrentRoom(&rooms[curRoomid-1]);
now-> setPreviousRoom(&rooms[prevRoomid-1]);

vector<Item> inventory;
int t; in >> t;
while (t--) {
    in >> tag >> name;
    in.ignore();
    int h, a, d, hun, thi, poi;
    in >> h >> a >> d;
    in >> hun >> thi >> poi;
    in.ignore();
    inventory.push_back(Item(name, h, a, d, hun, thi, poi));
}
now -> setInventory(inventory, false);
```

## Detailed Implementation

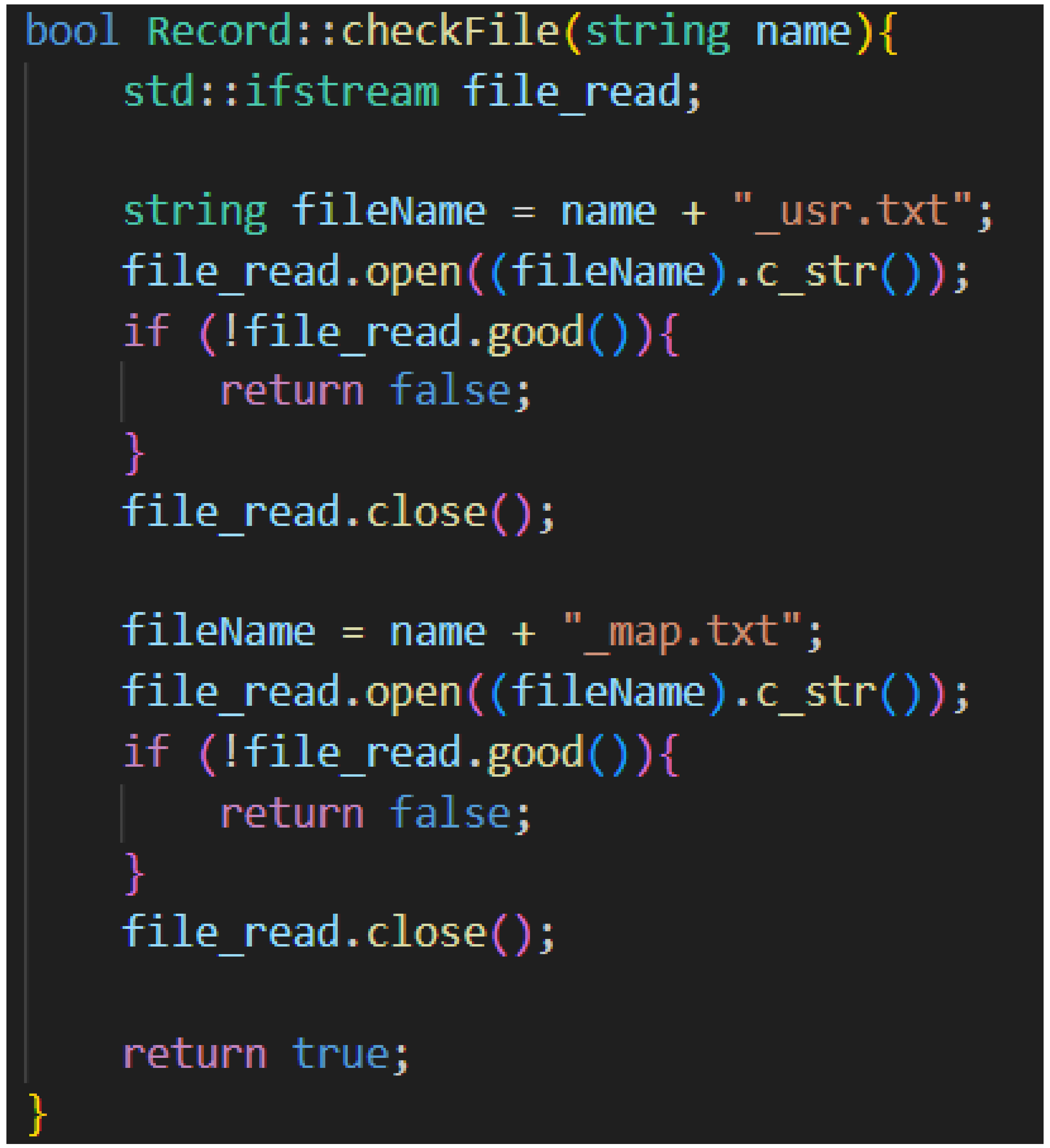


```
oid Record::loadFromFile(string name, Player* now, vector<Room>& v) {
    ifstream file_read;

    cout << "Map ........ ";
    file_read.open((name + "_map.txt").c_str());
    loadRooms(v, file_read);
    file_read.close();
    sleep(1);

    cout << "Finished\n";


    cout << "User ....... ";

    file_read.open((name + "_usr.txt").c_str());
    loadPlayer(now, file_read, v);
    file_read.close();
    sleep(1);

    cout << "Finished\n";
```



```
void Record::saveToFile(Player* now, vector<Room>& v) {
    string name = (*now).getName();
    std::ofstream out;

    out.open((name + "_map.txt").c_str());
    this -> saveRooms(v, out);
    out.close();

    out.open((name + "_usr.txt").c_str());
    this -> savePlayer(now, out);
    out.close();

    cout << "\nRecord Saved\n";
}
```

## Detailed Implementation



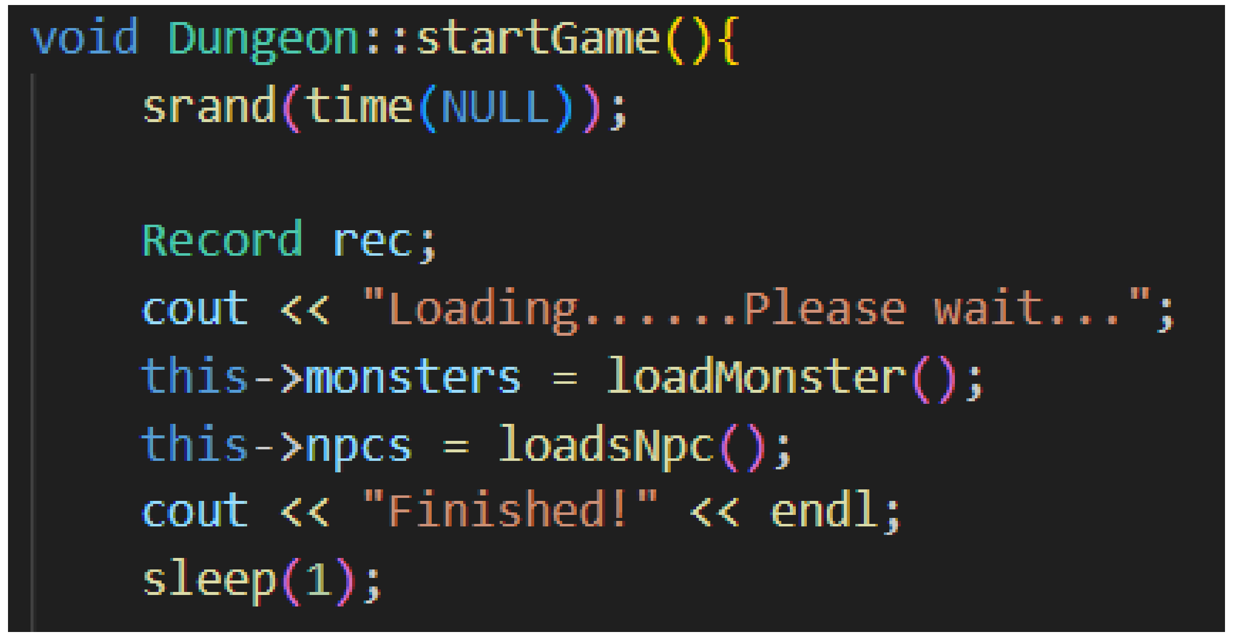
```
bool Record::checkFile(string name){
    std::ifstream file_read;

    string fileName = name + "_usr.txt";
    file_read.open((fileName).c_str());
    if (!file_read.good()){
        return false;
    }
    file_read.close();

    fileName = name + "_map.txt";
    file_read.open((fileName).c_str());
    if (!file_read.good()){
        return false;
    }
    file_read.close();

    return true;
}
```

## Result of dungeon (Class dungeon)

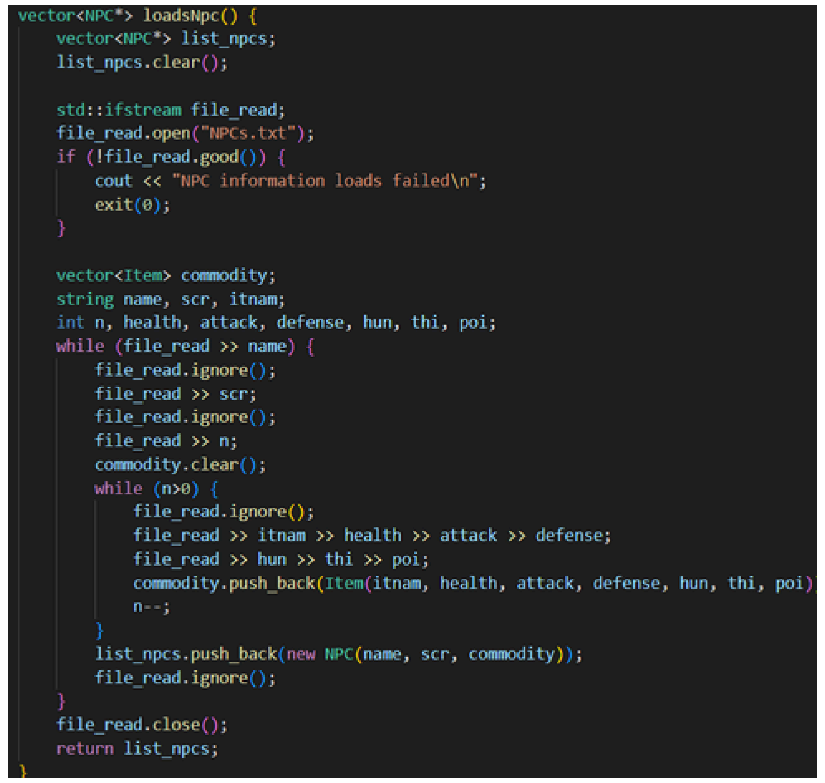
- Enter your name
  Create character
- Decide action
  and implement
- Repeat
  again and again
- Beat Boss
  win!

# Create GameCharacter

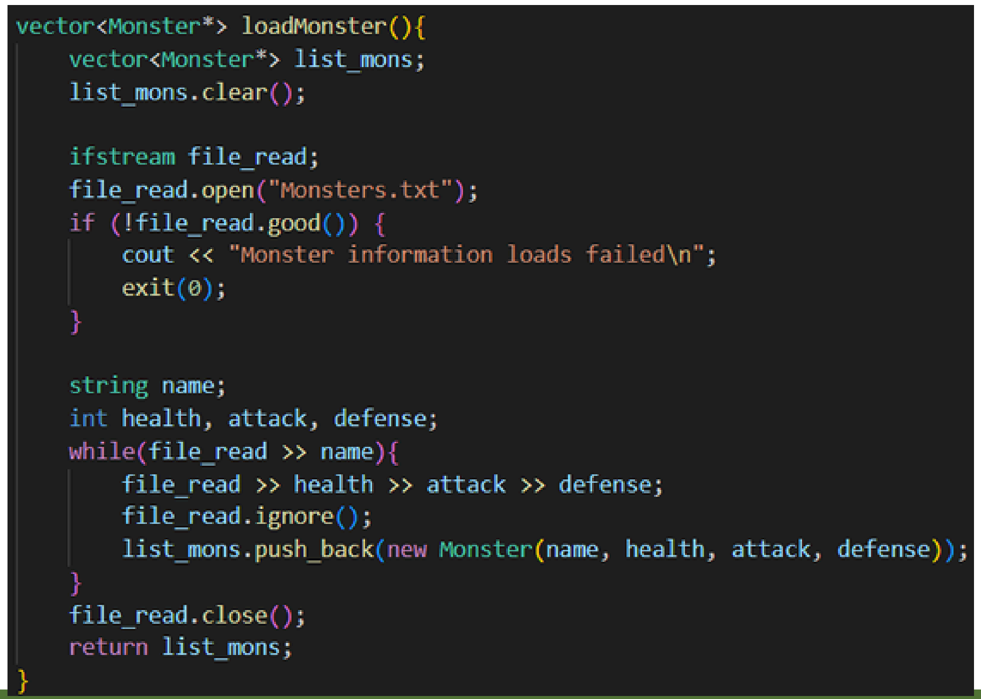
1. 讀取"NPCs.txt"和"Monsters.txt"中的資料並create出game character



```cpp
void Dungeon::startGame(){
    srand(time(NULL));

    Record rec;
    cout << "Loading......Please wait...";
    this->monsters = loadMonster();
    this->npcs = loadsNpc();
    cout << "Finished!" << endl;
    sleep(1);
```



```cpp
vector<NPC*> loadsNpc() {
    vector<NPC*> list_npcs;
    list_npcs.clear();

    std::ifstream file_read;
    file_read.open("NPCs.txt");
    if (!file_read.good()) {
        cout << "NPC information loads failed\n";
        exit(0);
    }

    vector<Item> commodity;
    string name, scr, itnam;
    int n, health, attack, defense, hun, thi, poi;
    while (file_read >> name) {
        file_read.ignore();
        file_read >> scr;
        file_read.ignore();
        file_read >> n;
        commodity.clear();
        while (n>0) {
            file_read.ignore();
            file_read >> itnam >> health >> attack >> defense;
            file_read >> hun >> thi >> poi;
            commodity.push_back(Item(itnam, health, attack, defense, hun, thi, poi)
            n--;
        }
        list_npcs.push_back(new NPC(name, scr, commodity));
        file_read.ignore();
    }
    file_read.close();
    return list_npcs;
}
```

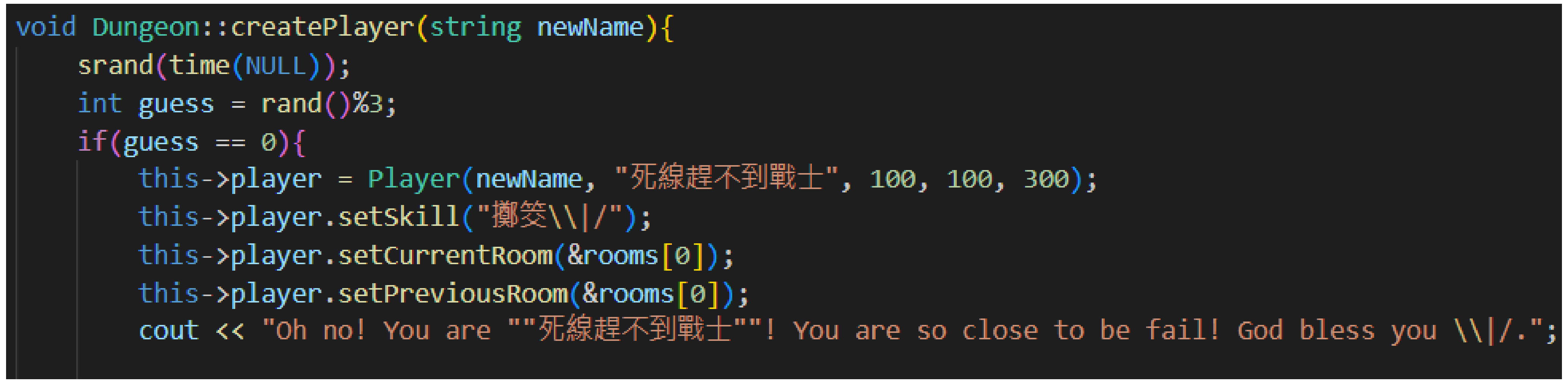


```cpp
vector<Monster*> loadMonster(){
    vector<Monster*> list_mons;
    list_mons.clear();

    ifstream file_read;
    file_read.open("Monsters.txt");
    if (!file_read.good()) {
        cout << "Monster information loads failed\n";
        exit(0);
    }

    string name;
    int health, attack, defense;
    while(file_read >> name){
        file_read >> health >> attack >> defense;
        file_read.ignore();
        list_mons.push_back(new Monster(name, health, attack, defense));
    }
    file_read.close();
    return list_mons;
}
```

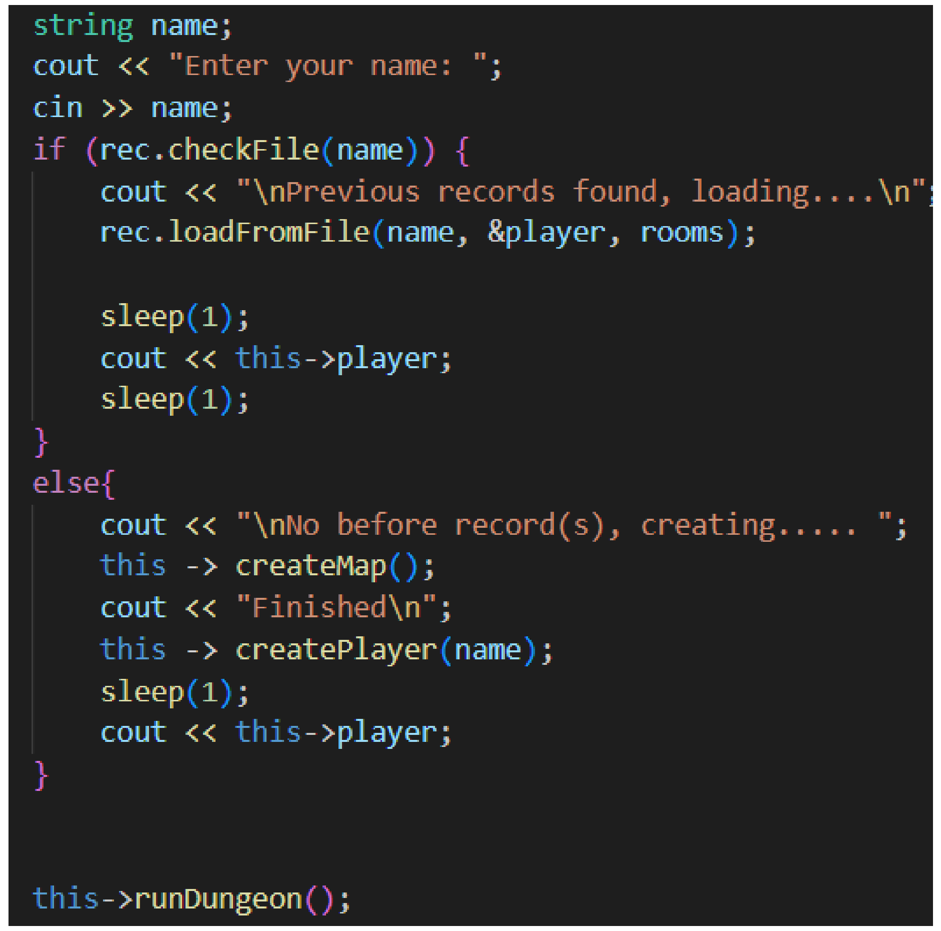
# Create GameCharacter

這裡只列一個rand()的可能值當例子

2. 輸入名字, 若有找到previous data則讀入保存的資料, 沒有的話則create一個新的並抽取role和獲取skill



```
void Dungeon::createPlayer(string newName){
    srand(time(NULL));
    int guess = rand()%3;
    if(guess == 0){
        this->player = Player(newName, "死線趕不到戰士", 100, 100, 300);
        this->player.setSkill("擲筊\\|/");
        this->player.setCurrentRoom(&rooms[0]);
        this->player.setPreviousRoom(&rooms[0]);
        cout << "Oh no! You are ""死線趕不到戰士""! You are so close to be fail! God bless you \\|/.";
```



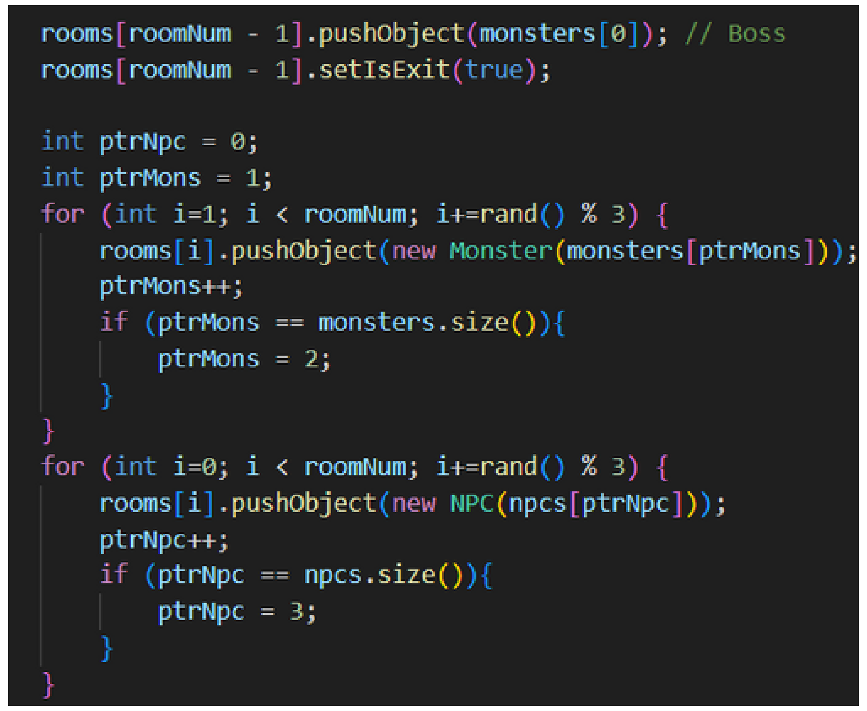
```
string name;
cout << "Enter your name: ";
cin >> name;
if (rec.checkFile(name)) {
    cout << "\nPrevious records found, loading....\n";
    rec.loadFromFile(name, &player, rooms);

    sleep(1);
    cout << this->player;
    sleep(1);
}
else{
    cout << "\nNo before record(s), creating..... ";
    this -> createMap();
    cout << "Finished\n";
    this -> createPlayer(name);
    sleep(1);
    cout << this->player;
}


this->runDungeon();
```

# Create GameCharacter

3. 讀取"map.txt", 將利用index值將Room連接

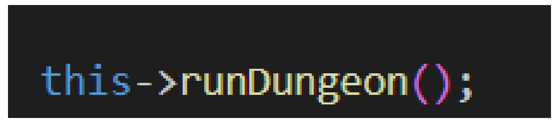
右圖為讀入Room數量後創建, 並讀取index值設定Room相對關係

下圖是把Monster隨機分配放入Room中

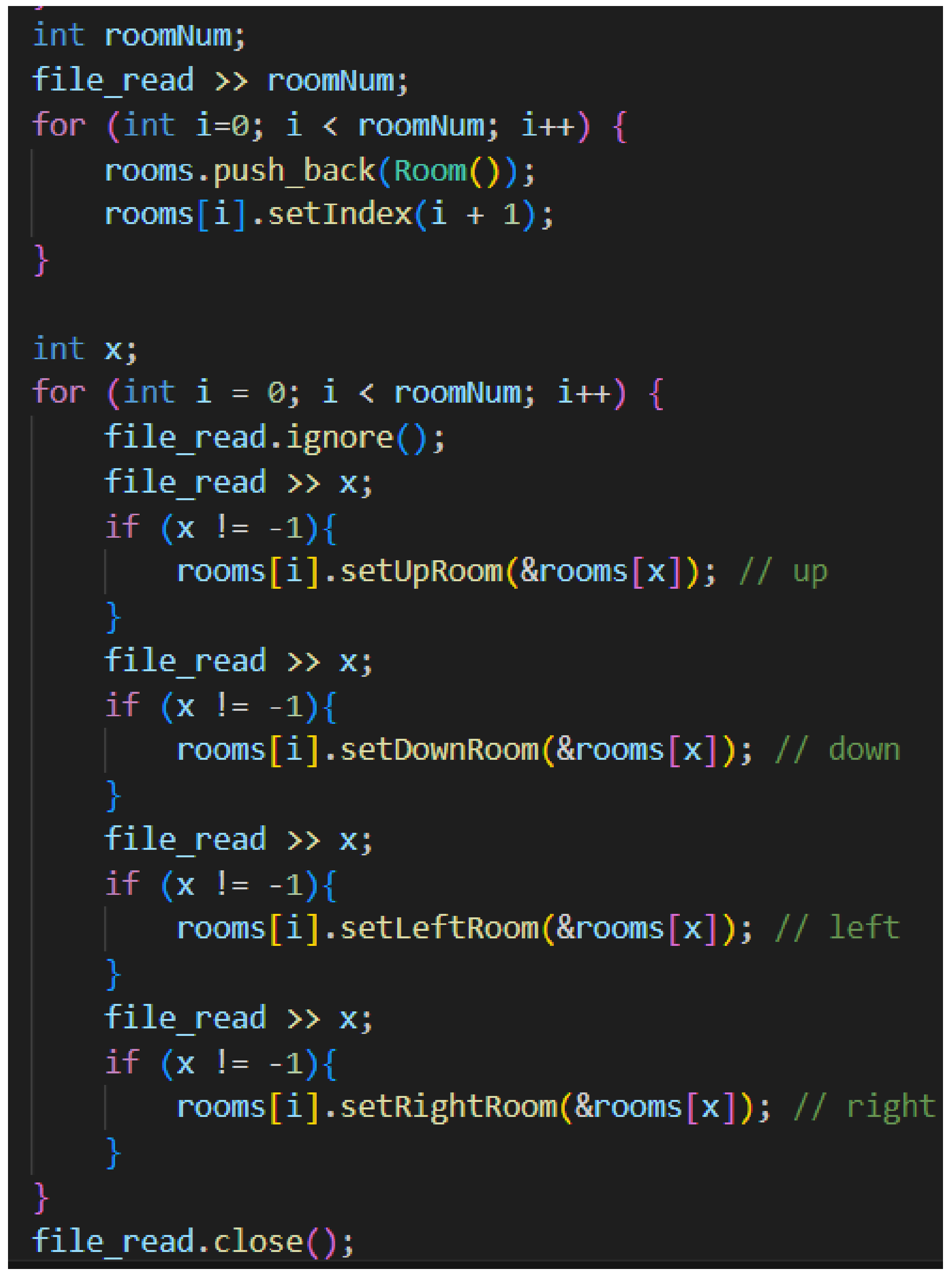


```cpp
rooms[roomNum - 1].pushObject(monsters[0]); // Boss
rooms[roomNum - 1].setIsExit(true);

int ptrNpc = 0;
int ptrMons = 1;
for (int i=1; i < roomNum; i+=rand() % 3) {
    rooms[i].pushObject(new Monster(monsters[ptrMons]));
    ptrMons++;
    if (ptrMons == monsters.size()){
        ptrMons = 2;
    }
}
for (int i=0; i < roomNum; i+=rand() % 3) {
    rooms[i].pushObject(new NPC(npcs[ptrNpc]));
    ptrNpc++;
    if (ptrNpc == npcs.size()){
        ptrNpc = 3;
    }
}
```

接著進入runDungeon函式!



```cpp
this->runDungeon();
```

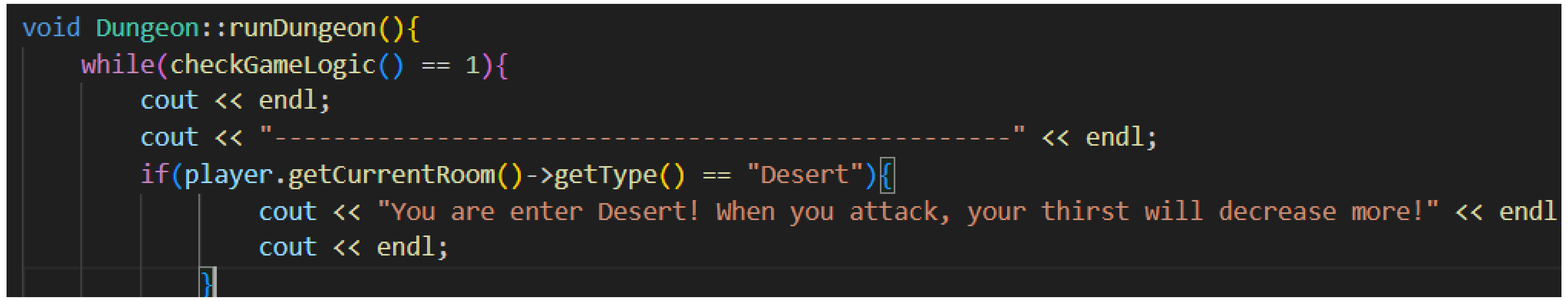


```cpp
int roomNum;
file_read >> roomNum;
for (int i=0; i < roomNum; i++) {
    rooms.push_back(Room());
    rooms[i].setIndex(i + 1);
}

int x;
for (int i = 0; i < roomNum; i++) {
    file_read.ignore();
    file_read >> x;
    if (x != -1){
        rooms[i].setUpRoom(&rooms[x]); // up
    }
    file_read >> x;
    if (x != -1){
        rooms[i].setDownRoom(&rooms[x]); // down
    }
    file_read >> x;
    if (x != -1){
        rooms[i].setLeftRoom(&rooms[x]); // left
    }
    file_read >> x;
    if (x != -1){
        rooms[i].setRightRoom(&rooms[x]); // right
    }
}
file_read.close();
```

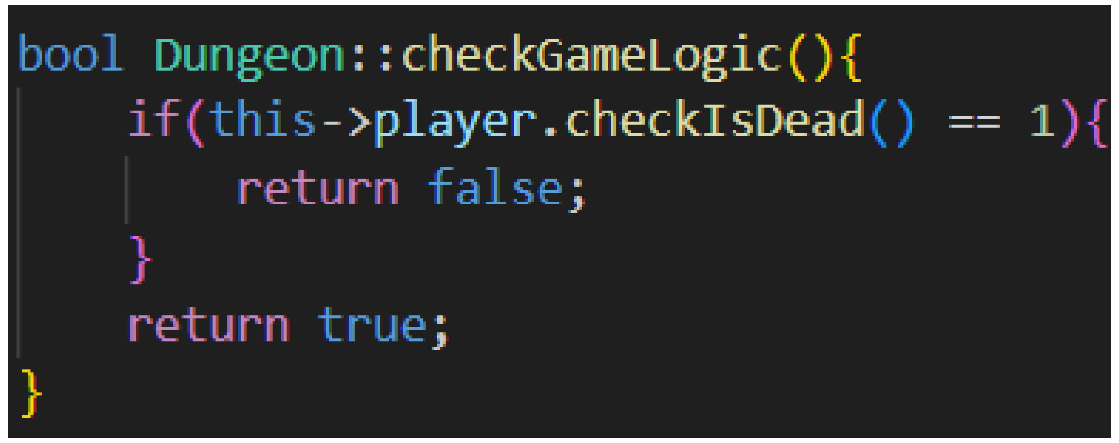
# Run Dungeon--Check Game Logic

這裡只列一個Room的type當例子

1. 檢查遊戲邏輯是否符合(是不是Health還沒歸零), 要是符合則
   依照Player所在的Room type輸出不同對話
   以下是type = "Desert"的情況



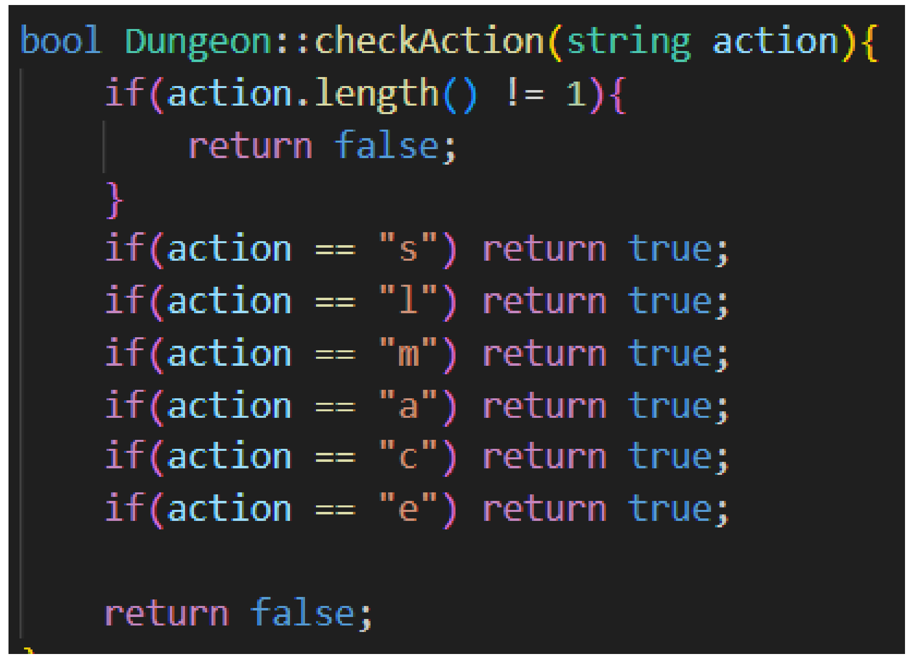
```cpp
void Dungeon::runDungeon(){
    while(checkGameLogic() == 1){
        cout << endl;
        cout << "----------------------------------------------------" << endl;
        if(player.getCurrentRoom()->getType() == "Desert"){
            cout << "You are enter Desert! When you attack, your thirst will decrease more!" << endl
            cout << endl;
        }
```



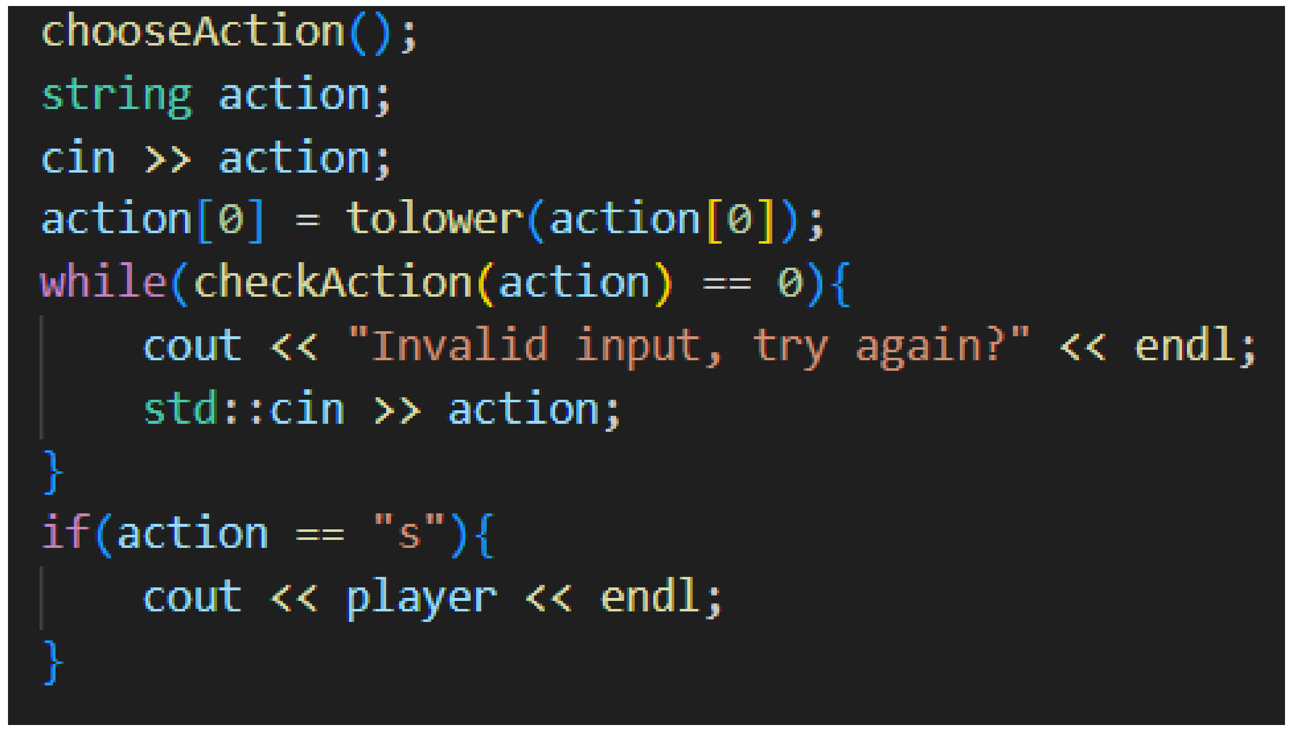
```cpp
bool Dungeon::checkGameLogic(){
    if(this->player.checkIsDead() == 1){
        return false;
    }
    return true;
}
```

# Choose Action--"s"

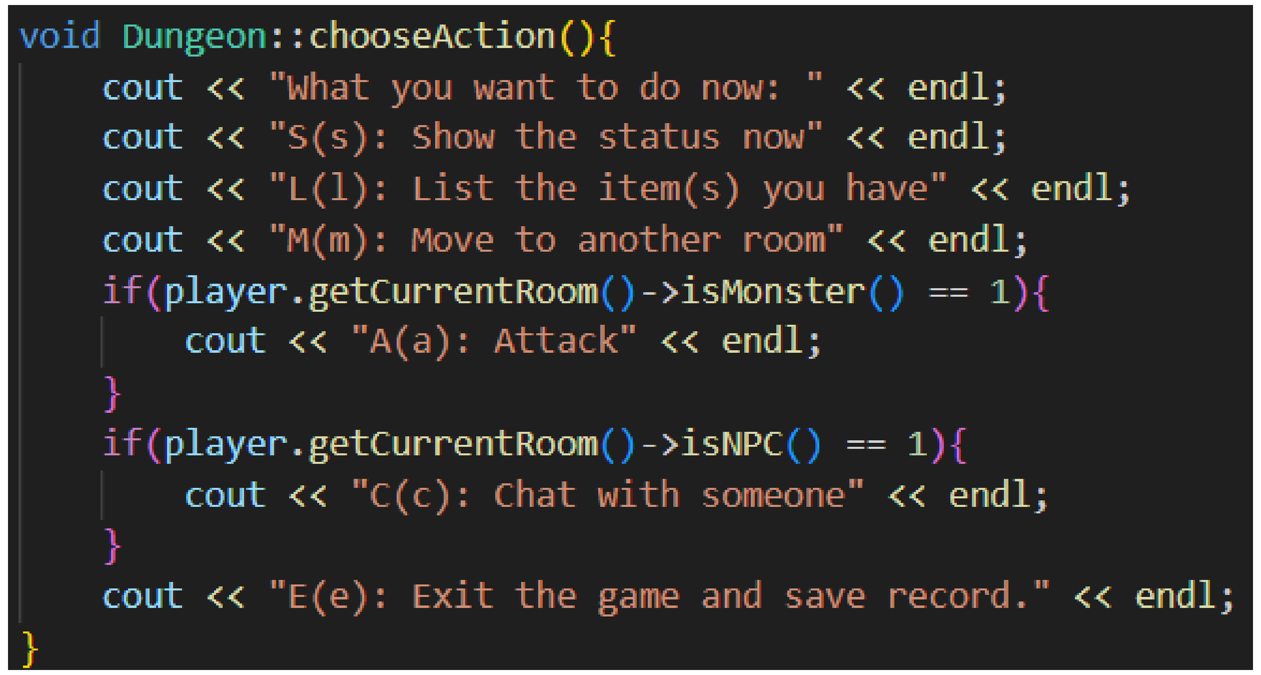
2. 依照chooseAction()指示輸入想執行的action,
   再利用checkAction()確認輸出的action是否有效,
   若無效則會要求再輸入一次, 若輸入成功,
"s": 輸出player目前status
(見Player operator overload)



```cpp
bool Dungeon::checkAction(string action){
    if(action.length() != 1){
        return false;
    }
    if(action == "s") return true;
    if(action == "l") return true;
    if(action == "m") return true;
    if(action == "a") return true;
    if(action == "c") return true;
    if(action == "e") return true;

    return false;
```



```cpp
chooseAction();
string action;
cin >> action;
action[0] = tolower(action[0]);
while(checkAction(action) == 0){
    cout << "Invalid input, try again?" << endl;
    std::cin >> action;
}
if(action == "s"){
    cout << player << endl;
}
```

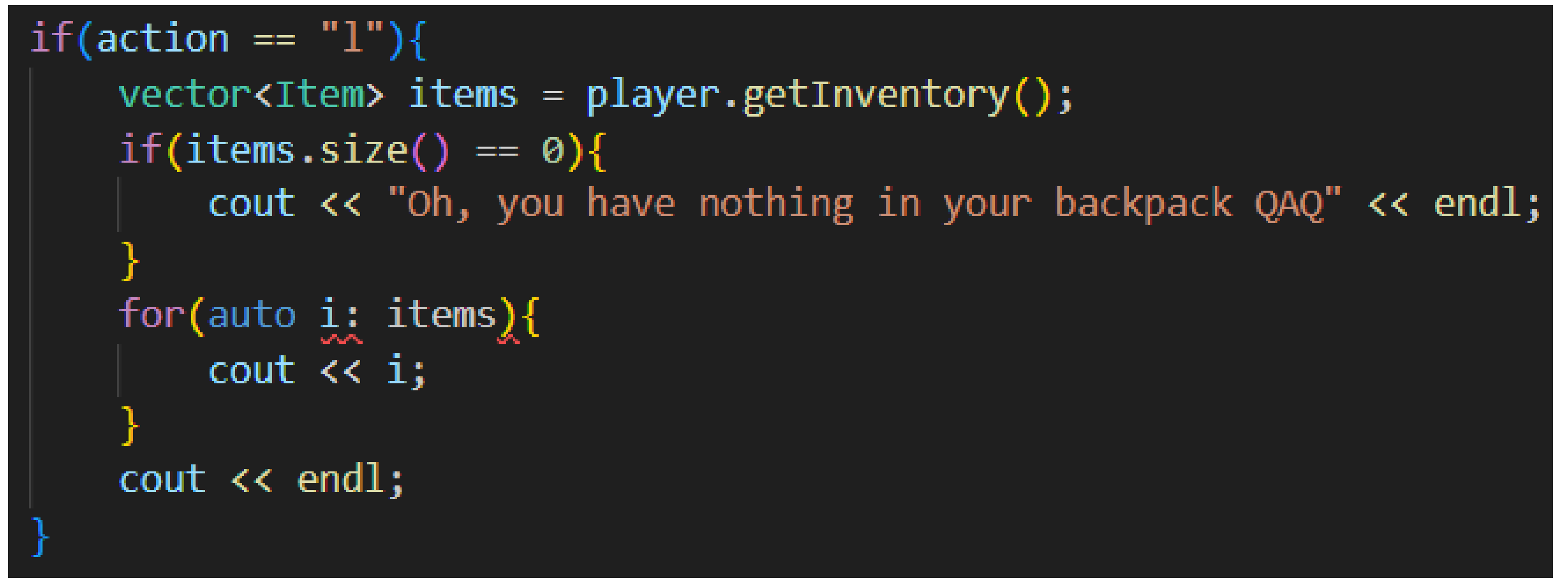


```cpp
void Dungeon::chooseAction(){
    cout << "What you want to do now: " << endl;
    cout << "S(s): Show the status now" << endl;
    cout << "L(l): List the item(s) you have" << endl;
    cout << "M(m): Move to another room" << endl;
    if(player.getCurrentRoom()->isMonster() == 1){
        cout << "A(a): Attack" << endl;
    }
    if(player.getCurrentRoom()->isNPC() == 1){
        cout << "C(c): Chat with someone" << endl;
    }
    cout << "E(e): Exit the game and save record." << endl;
}
```

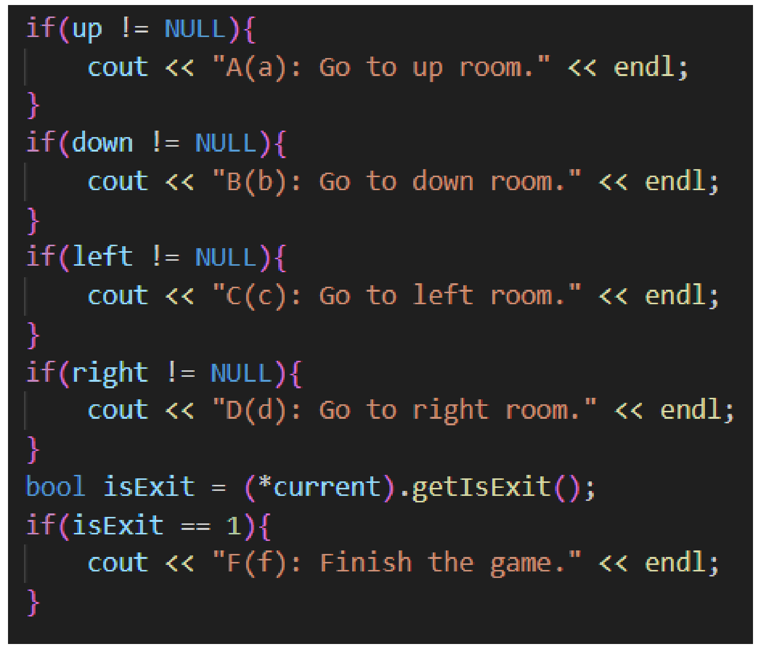
## Choose Action--"l"and"m"

"l": 把Player手中持有的所有Item的name和attack, health等數值全部輸出(見Item operator overload)

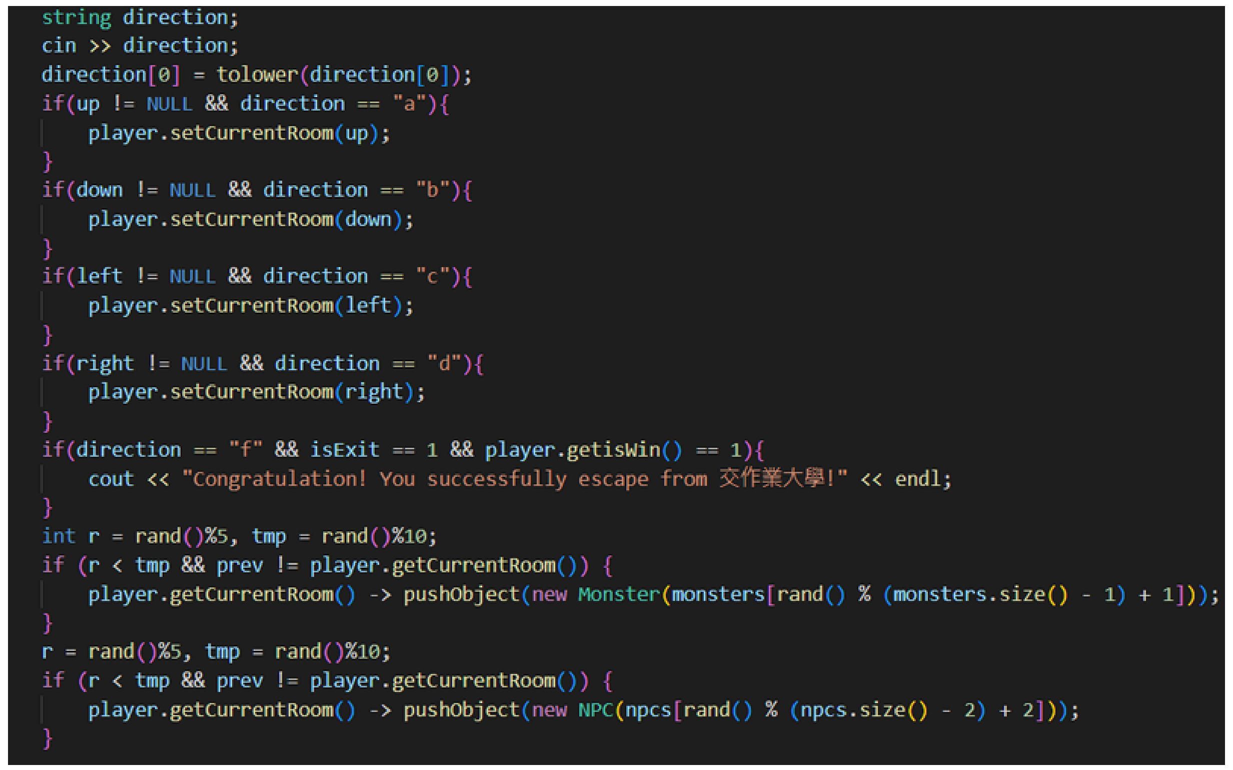
"m": 會引到函式handleMovement(), 輸出目前可以前進的方向(或Finish)對話並依照選擇設定currentRoom和Room中的NPC和Monster



```cpp
if(action == "l"){
    vector<Item> items = player.getInventory();
    if(items.size() == 0){
        cout << "Oh, you have nothing in your backpack QAQ" << endl;
    }
    for(auto i: items){
        cout << i;
    }
    cout << endl;
}
```



```cpp
if(up != NULL){
    cout << "A(a): Go to up room." << endl;
}
if(down != NULL){
    cout << "B(b): Go to down room." << endl;
}
if(left != NULL){
    cout << "C(c): Go to left room." << endl;
}
if(right != NULL){
    cout << "D(d): Go to right room." << endl;
}
bool isExit = (*current).getIsExit();
if(isExit == 1){
    cout << "F(f): Finish the game." << endl;
}
```

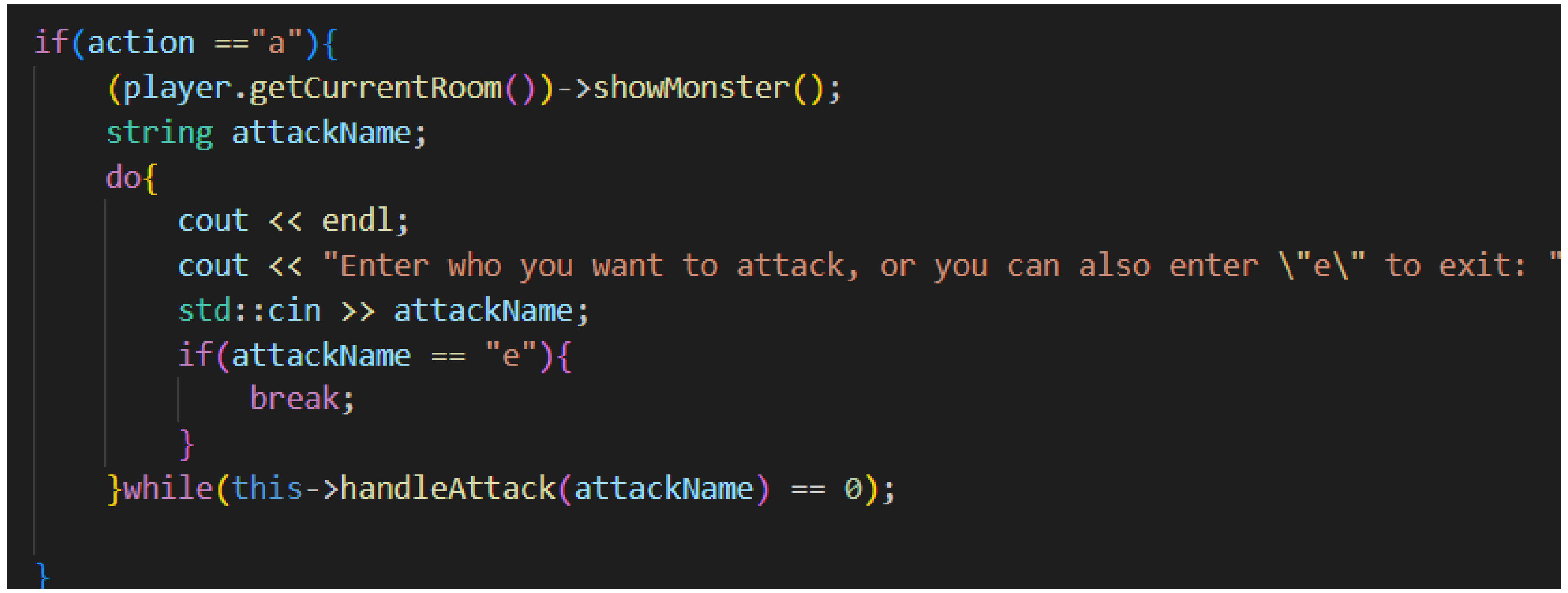


```cpp
string direction;
cin >> direction;
direction[0] = tolower(direction[0]);
if(up != NULL && direction == "a"){
    player.setCurrentRoom(up);
}
if(down != NULL && direction == "b"){
    player.setCurrentRoom(down);
}
if(left != NULL && direction == "c"){
    player.setCurrentRoom(left);
}
if(right != NULL && direction == "d"){
    player.setCurrentRoom(right);
}
if(direction == "f" && isExit == 1 && player.getisWin() == 1){
    cout << "Congratulation! You successfully escape from 交作業大學!" << endl;
}
int r = rand()%5, tmp = rand()%10;
if (r < tmp && prev != player.getCurrentRoom()) {
    player.getCurrentRoom() -> pushObject(new Monster(monsters[rand() % (monsters.size() - 1) + 1]));
}
r = rand()%5, tmp = rand()%10;
if (r < tmp && prev != player.getCurrentRoom()) {
    player.getCurrentRoom() -> pushObject(new NPC(npcs[rand() % (npcs.size() - 2) + 2]));
}
```

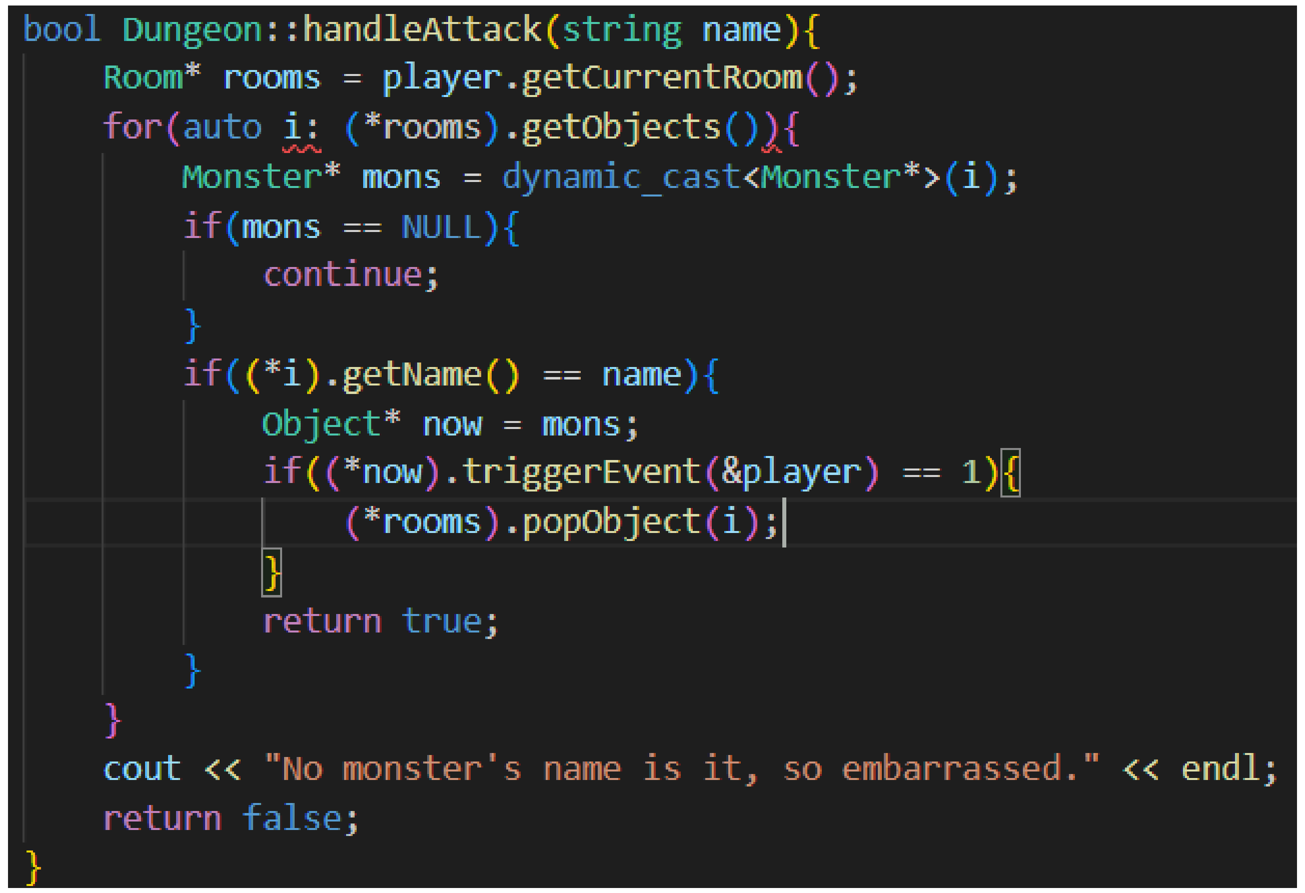
# Choose Action--"a"

"a": 在handleAttack() return false時(Monster還沒死), 對Monster攻擊

(下頁詳談Monster的triggerEvent)



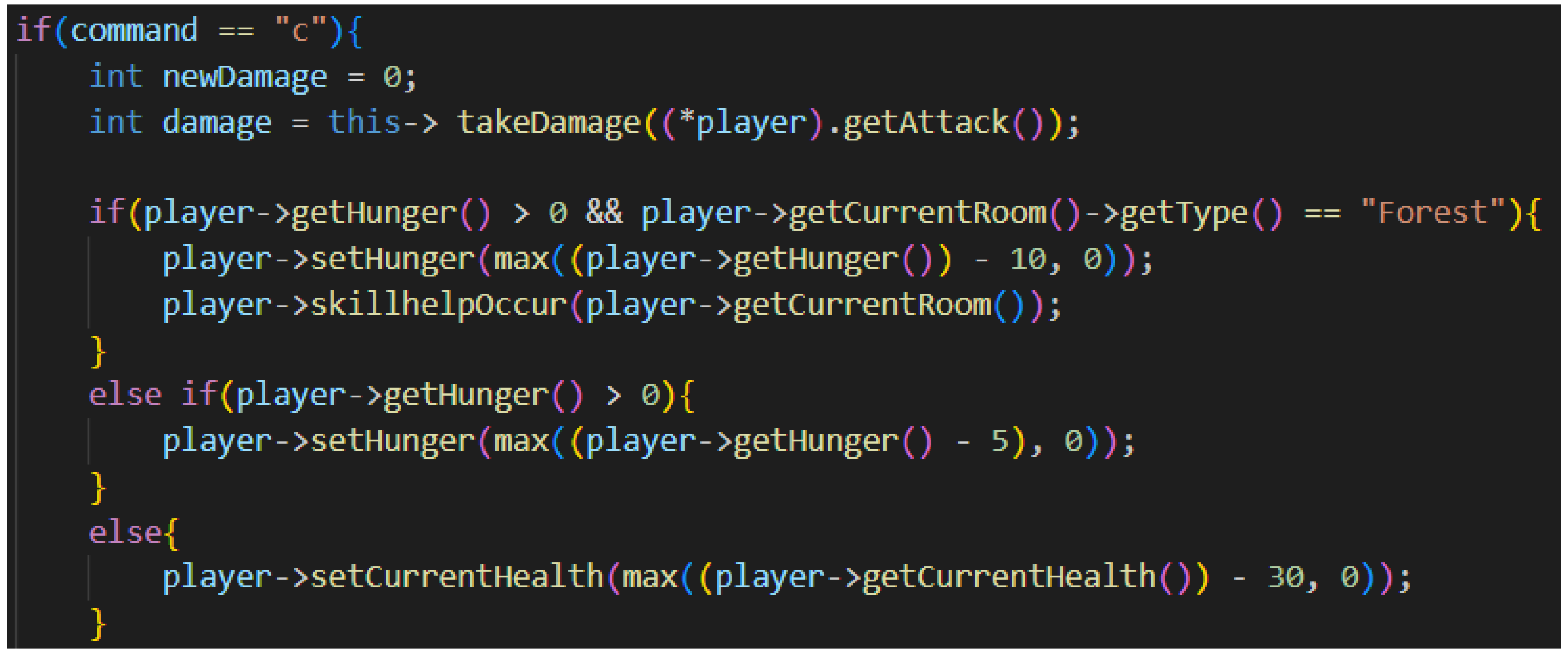
```
if(action =="a"){
    (player.getCurrentRoom())->showMonster();
    string attackName;
    do{
        cout << endl;
        cout << "Enter who you want to attack, or you can also enter \"e\" to exit: "
        std::cin >> attackName;
        if(attackName == "e"){
            break;
        }
    }while(this->handleAttack(attackName) == 0);

}
```



```
bool Dungeon::handleAttack(string name){
    Room* rooms = player.getCurrentRoom();
    for(auto i: (*rooms).getObjects()){
        Monster* mons = dynamic_cast<Monster*>(i);
        if(mons == NULL){
            continue;
        }
        if((*i).getName() == name){
            Object* now = mons;
            if((*now).triggerEvent(&player) == 1){
                (*rooms).popObject(i);
            }
            return true;
        }
    }
    cout << "No monster's name is it, so embarrassed." << endl;
    return false;
}
```

## Choose Action--"a"

關於Monster::triggerEvent:
如同在class Monster中看到的, 在按"a"後若按"r"則代表撤退, 那麼就會再次回到選擇action的頁面
但如果按"c"就會開始一連串的攻防

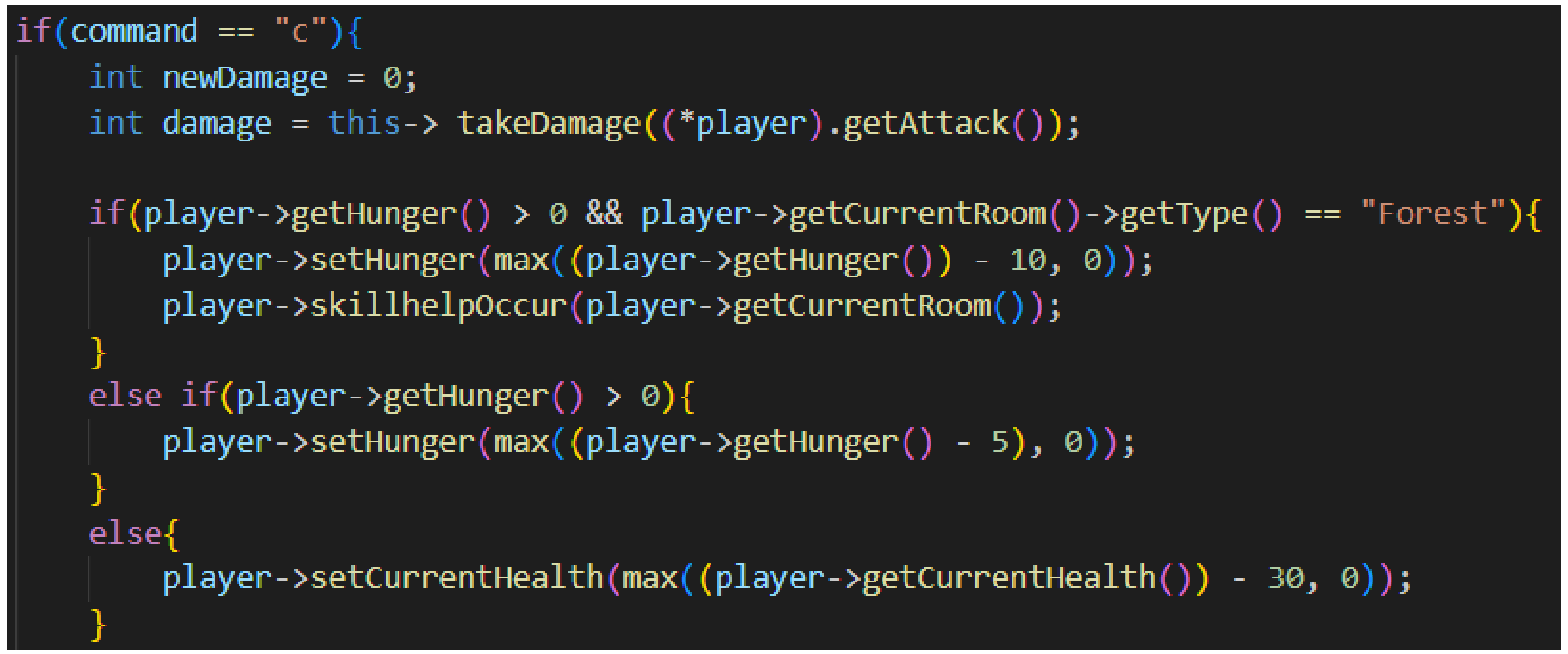


```
if(command == "c"){
    int newDamage = 0;
    int damage = this-> takeDamage((*player).getAttack());

    if(player->getHunger() > 0 && player->getCurrentRoom()->getType() == "Forest"){
        player->setHunger(max((player->getHunger()) - 10, 0));
        player->skillhelpOccur(player->getCurrentRoom());
    }
    else if(player->getHunger() > 0){
        player->setHunger(max((player->getHunger() - 5), 0));
    }
    else{
        player->setCurrentHealth(max((player->getCurrentHealth()) - 30, 0));
    }
```

Hunger System設定:
Hunger:
攻擊Monster一次, 則hunger降低
若是Room type為"Forest"的話則會降低更多
Thirst:
攻擊Monster一次, 則thirst降低
若是Room type為"Desert"的話則會降低更多
Poison:
攻擊Monster一次, 則health會再被扣poson的值
若是Room type為"Swamp"的話則會使poison增加

## Choose Action--"a"

如同在class Player中說的, skillhelpOccur()會隨機決定是否執行和執行哪一個

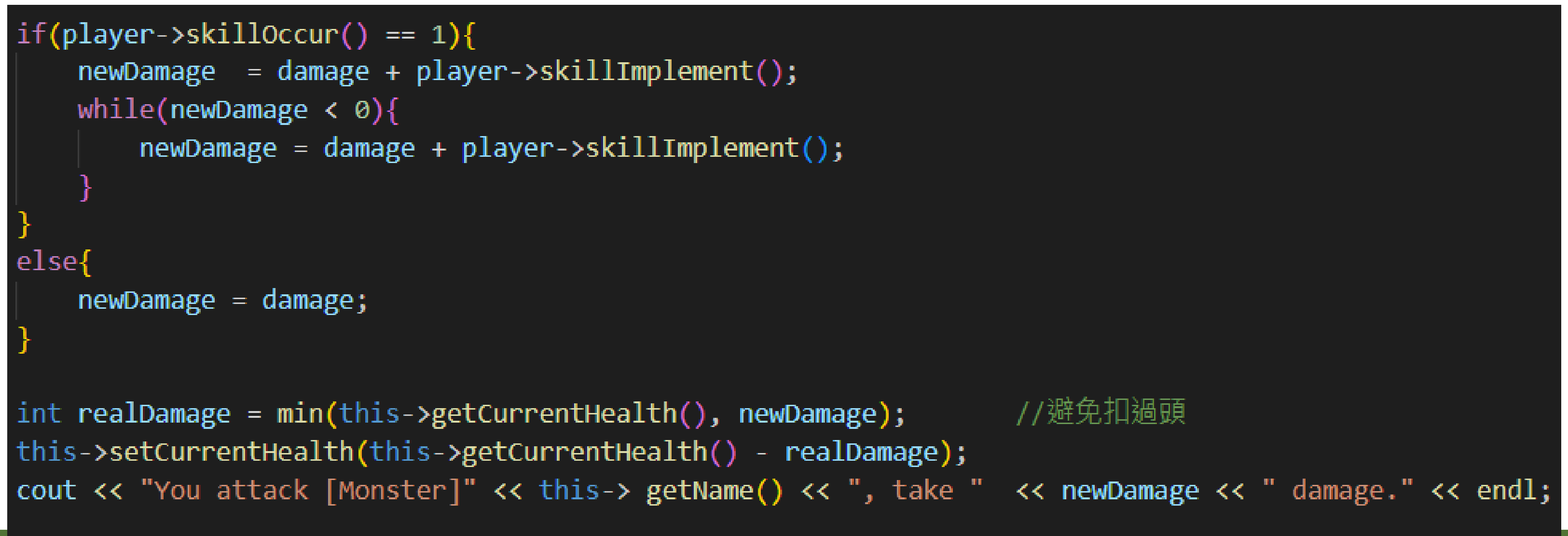


```
if(command == "c"){
    int newDamage = 0;
    int damage = this-> takeDamage((*player).getAttack());

    if(player->getHunger() > 0 && player->getCurrentRoom()->getType() == "Forest"){
        player->setHunger(max((player->getHunger()) - 10, 0));
        player->skillhelpOccur(player->getCurrentRoom());
    }
    else if(player->getHunger() > 0){
        player->setHunger(max((player->getHunger() - 5), 0));
    }
    else{
        player->setCurrentHealth(max((player->getCurrentHealth()) - 30, 0));
    }
```

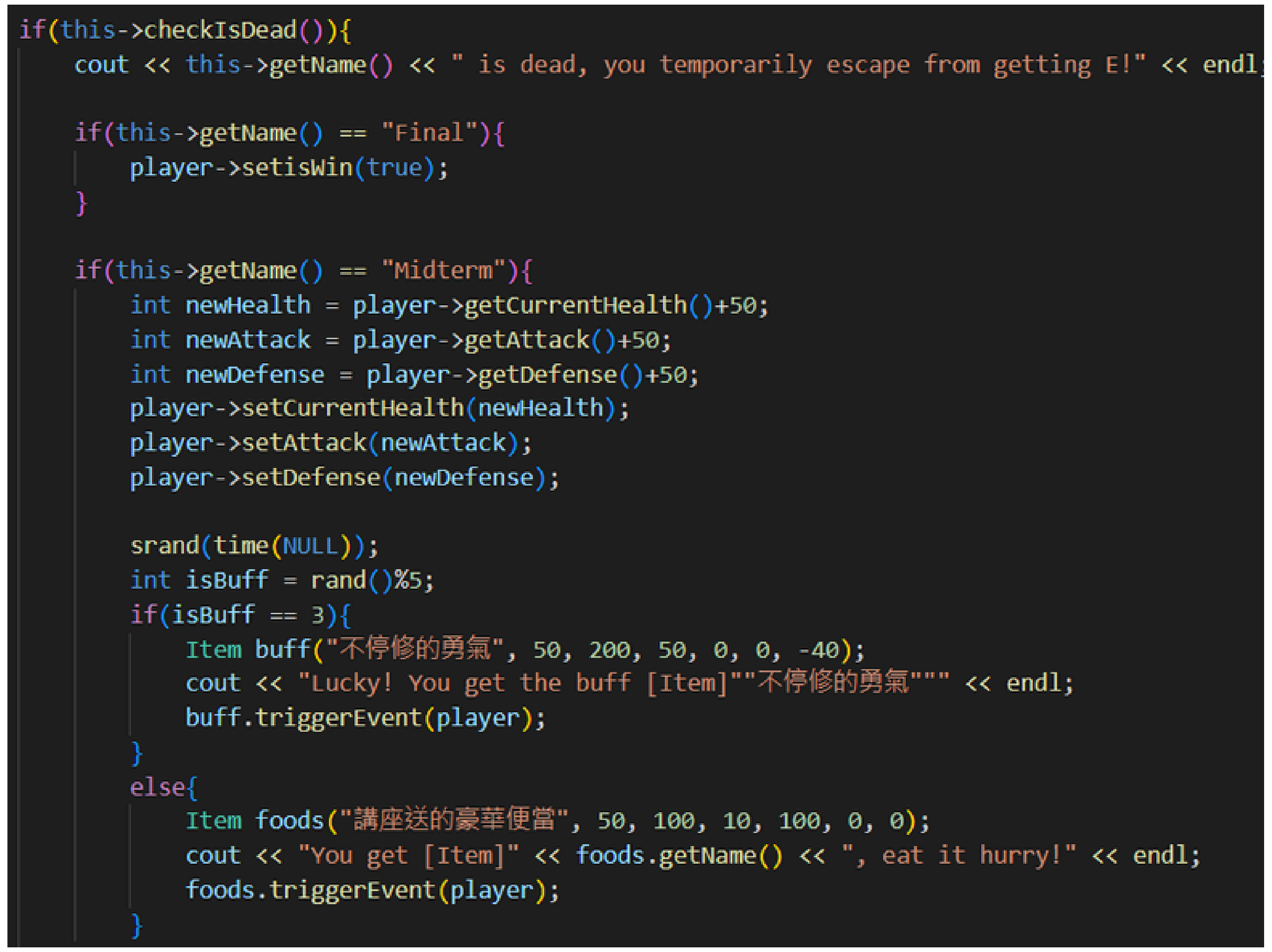
Room System設定:
Forest:
hunger會降的更多, 並且skill"Wildlife"會添加新的Monster到Room中
但help"Lake"能夠補充thirst值
Desert:
thirst會降的更多, 並且skill"Sandstorm"會降低hunger和thirst
但help"Oasis"能夠補充thirst值
Swamp:
poison會增加導致傷害變大, 並且skill "Trapped"會讓attack和defense降低, 不過help"Crocodile"可以讓hunger和thirst增加

# Choose Action--"a"

同樣的, Player用skillOccur來隨機決定是否發動技能
(詳見class Player, 又Monster打Player也是一樣的程式)
若Monster被打倒, 則依照Monster類型得到不同的獎勵,
若是是Boss被打倒則把isWin設為1, 若是Player被打倒,
triggerEvent就會return false
(右邊以Monster "Midterm"為例)



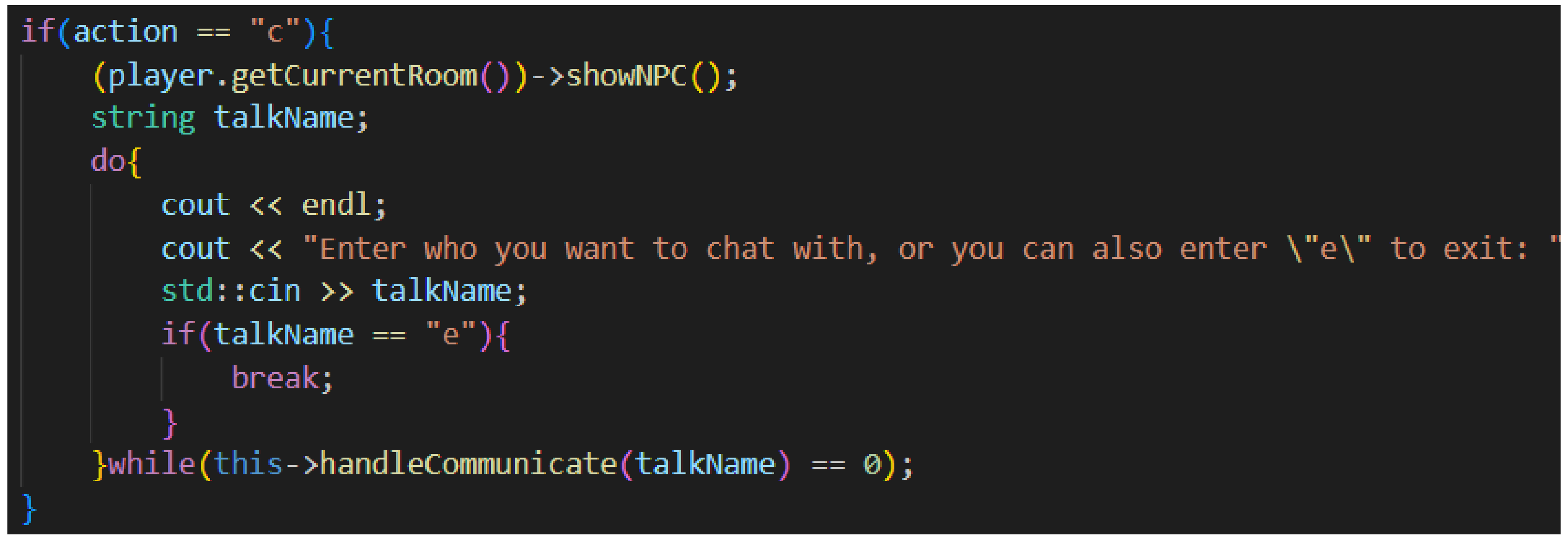
```
if(player->skillOccur() == 1){
    newDamage  = damage + player->skillImplement();
    while(newDamage < 0){
        newDamage = damage + player->skillImplement();
    }
}
else{
    newDamage = damage;
}

int realDamage = min(this->getCurrentHealth(), newDamage);       //避免扣過頭
this->setCurrentHealth(this->getCurrentHealth() - realDamage);
cout << "You attack [Monster]" << this-> getName() << ", take "  << newDamage << " damage." << endl;
```



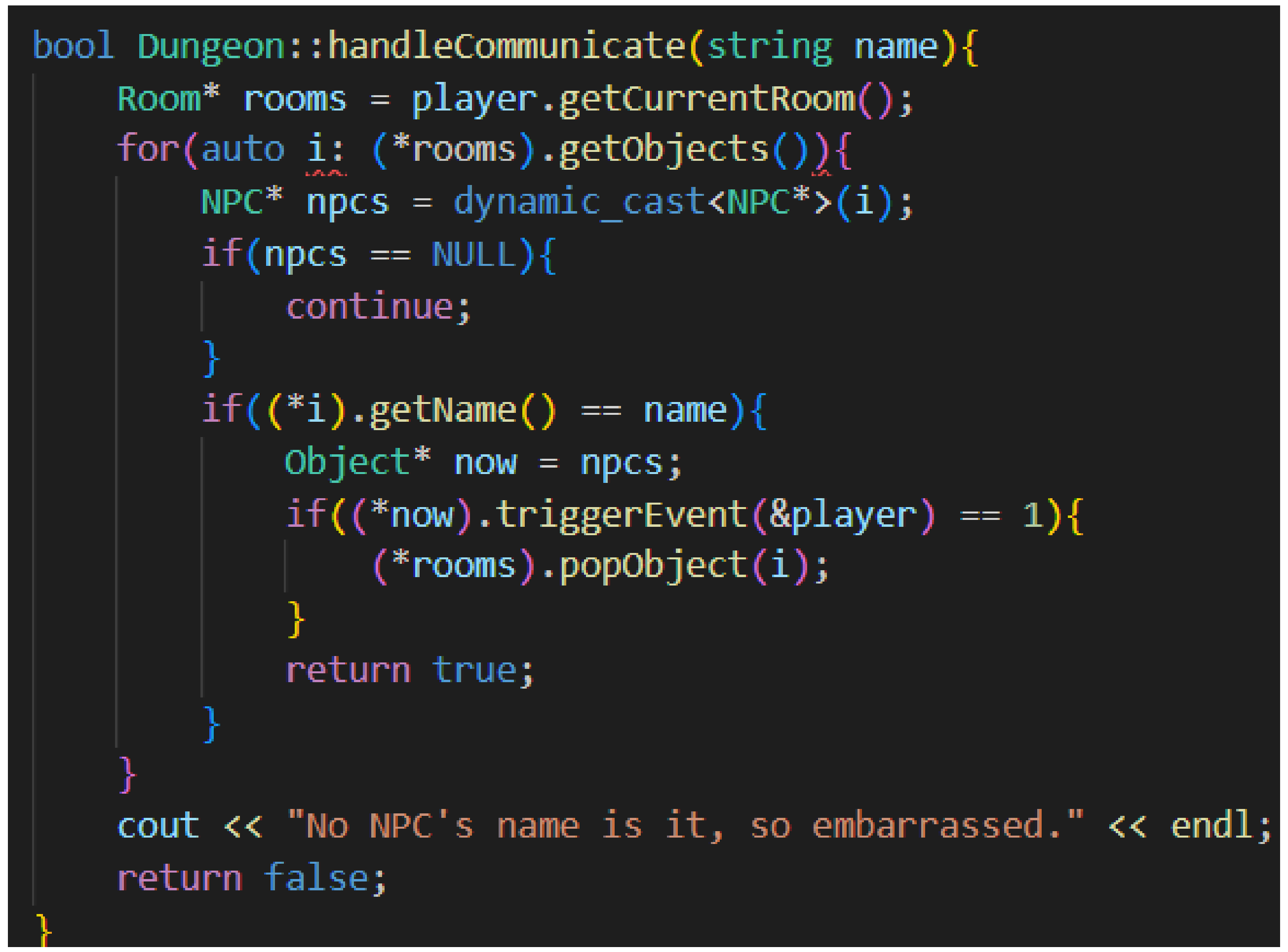
```
if(this->checkIsDead()){
    cout << this->getName() << " is dead, you temporarily escape from getting E!" << endl

    if(this->getName() == "Final"){
        player->setisWin(true);
    }

    if(this->getName() == "Midterm"){
        int newHealth = player->getCurrentHealth()+50;
        int newAttack = player->getAttack()+50;
        int newDefense = player->getDefense()+50;
        player->setCurrentHealth(newHealth);
        player->setAttack(newAttack);
        player->setDefense(newDefense);

        srand(time(NULL));
        int isBuff = rand()%5;
        if(isBuff == 3){
            Item buff("不停修的勇氣", 50, 200, 50, 0, 0, -40);
            cout << "Lucky! You get the buff [Item]""不停修的勇氣""" << endl;
            buff.triggerEvent(player);
        }
        else{
            Item foods("講座送的豪華便當", 50, 100, 10, 100, 0, 0);
            cout << "You get [Item]" << foods.getName() << ", eat it hurry!" << endl;
            foods.triggerEvent(player);
        }
```

## Choose Action--"c"

"c": 跟NPC講話並拿取Item, 會先用showNPC()把Room中所有NPC都輸出讓Player去選擇, 再用NPC的triggerEvent來讓Item在Player上發揮作用(功能詳見class NPC和class Item)



```
if(action == "c"){
    (player.getCurrentRoom())->showNPC();
    string talkName;
    do{
        cout << endl;
        cout << "Enter who you want to chat with, or you can also enter \"e\" to exit: "
        std::cin >> talkName;
        if(talkName == "e"){
            break;
        }
    }while(this->handleCommunicate(talkName) == 0);
}
```

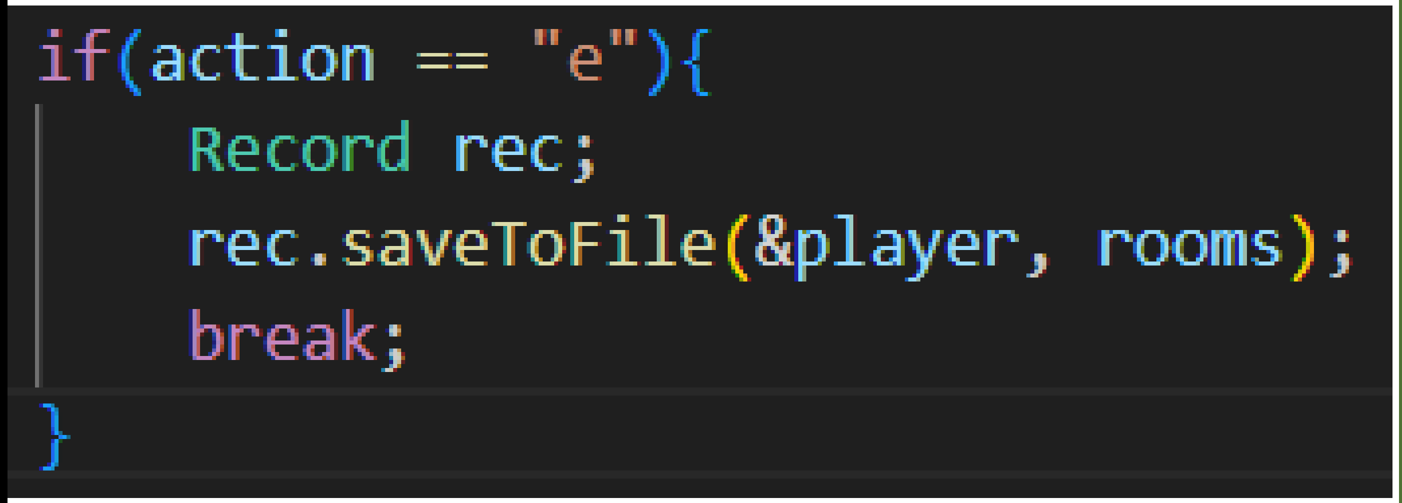


```
bool Dungeon::handleCommunicate(string name){
    Room* rooms = player.getCurrentRoom();
    for(auto i: (*rooms).getObjects()){
        NPC* npcs = dynamic_cast<NPC*>(i);
        if(npcs == NULL){
            continue;
        }
        if((*i).getName() == name){
            Object* now = npcs;
            if((*now).triggerEvent(&player) == 1){
                (*rooms).popObject(i);
            }
            return true;
        }
    }
    cout << "No NPC's name is it, so embarrassed." << endl;
    return false;
}
```

## Choose Action--"e"

"e": 跳出並存檔, 不過如果是在按了其他action後按了 "e", 那就只是跳回choose action的地方而已

(saveToFile詳見class Record)



```
if(action == "e"){
    Record rec;
    rec.saveToFile(&player, rooms);
    break;
}
```

## Discussion: What I do to make it better

### Monster/Player角色技能設置

除了在開局時隨機抽取player角色, 獲取不同skill外, 我也讓不同Monster依照名字設定自己的skill, 增加遊戲有趣性和不可預測性

### Record系統設置

實際玩遊戲時的存檔功能, 能夠讓player在有事情還不能玩完時儲存目前的遊戲資料, 並在下次遊戲開始時透過輸入相同名字來叫出之前的檔案

### 打倒Monster隨機獲取Buff

打倒怪物時, 一樣用隨機方法來對應Monster獲取不同的Buff(Item)

### Item poison值保密機制

為了避免poison值被看到而導致player不願意拿, 我設定在拿到Item以前, player是看不到poison值的(只會看到一個“?”)

## Discussion: What I do to make it better

### Monster數量不定且隨機

為了增加刺激性(每個Room只有一個太少了),
所以就設置讓for迴圈在跑的時候除了最後一
間一定是Boss以外，其他間的數量和種類都隨
機

### NPC/Monster/Map資料可透過改變檔案改變

NPC/Monster的資料是從外面叫"NPCs.txt"和"Monsters.txt"來
讀取的, 所以可以隨時改變其資料, 而Map則是從"Map.txt"來讀取

## Conclusion

雖然這個HW真的很累、很麻煩，但不得不說我真的從裡面複習到了很多OOP的觀念，從 virtual override到inheritance各種特性, 再加上rand()隨機概念, 真的一次練到很多東西, 也有覺得越來越有趣!

# 謝謝！

Game over!

喜歡 不喜歡