國立交通大學 National Chiao Tung University

109-2 1177 資料結構與物件導向程式設計

Homework 1 – Dungeon

Paper Report

109550073 資工 陳宥安

Table of Contents

1. Code Outline

1.1. Dungeon Map 3
1.2. Libraries
1.3. Class Room / linkedRoom
1.4. Class NPC / Peter / boss 4
1.5. Class Dungeon
1.6. Main Function 5

2. Basic Functions

2.1. Movement
2.2. Showing Status 8
2.3. Action Menu
2.4. Pick up Items
2.5. Fighting System 9
2.6. NPC 12
2.7. Game Logic 13
2.8. Record System 14
2.9. Inheritance and Virtual Function 15

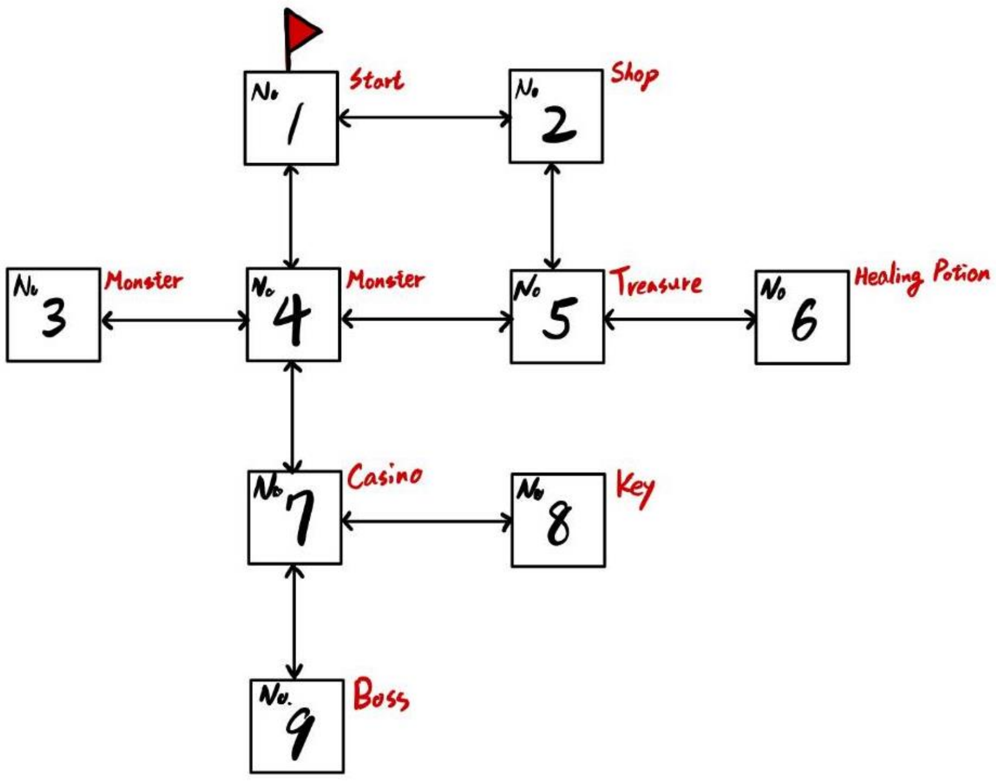
3. Optional Enhancements

3.1. Money System 16
3.2. Casino 17
3.3. Treasure Box and Key 19

4. Conclusion 20

1. Outline

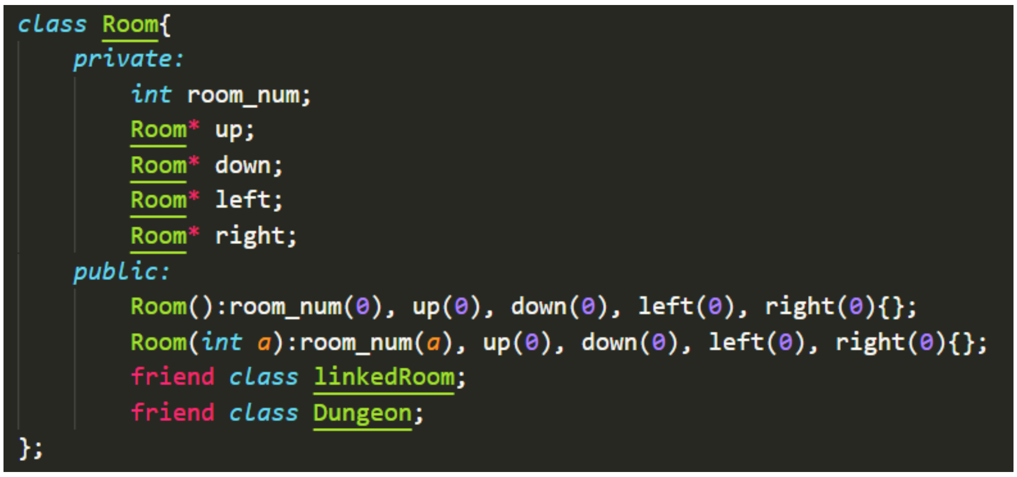
1.1. Dungeon Map



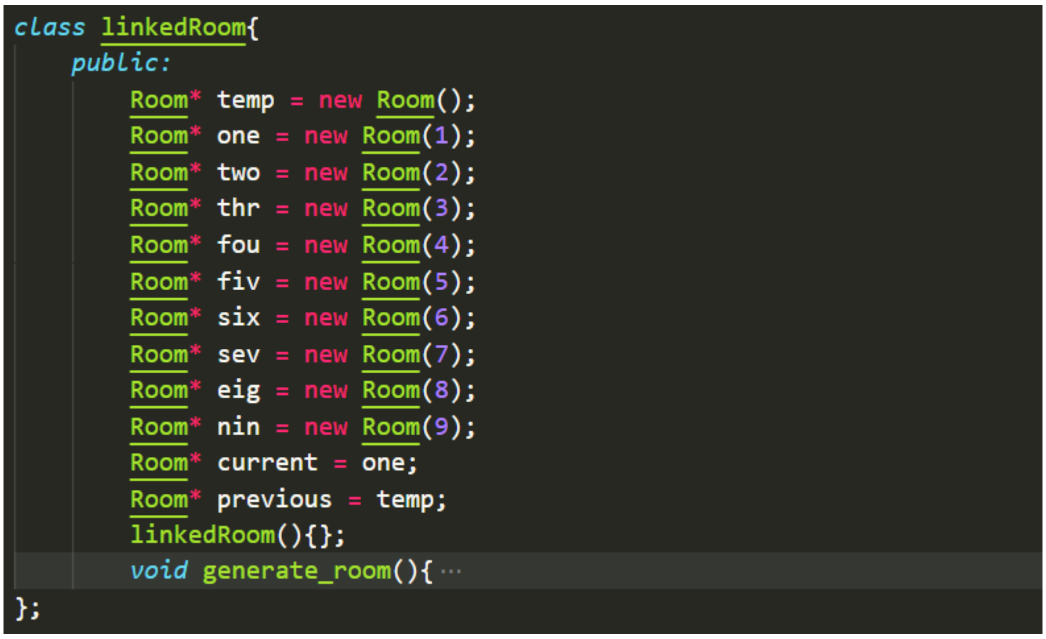
1.2. Libraries

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <cstdlib>
#include <ctime>
#include <fstream>
```

1.3. Class Room / linkedRoom

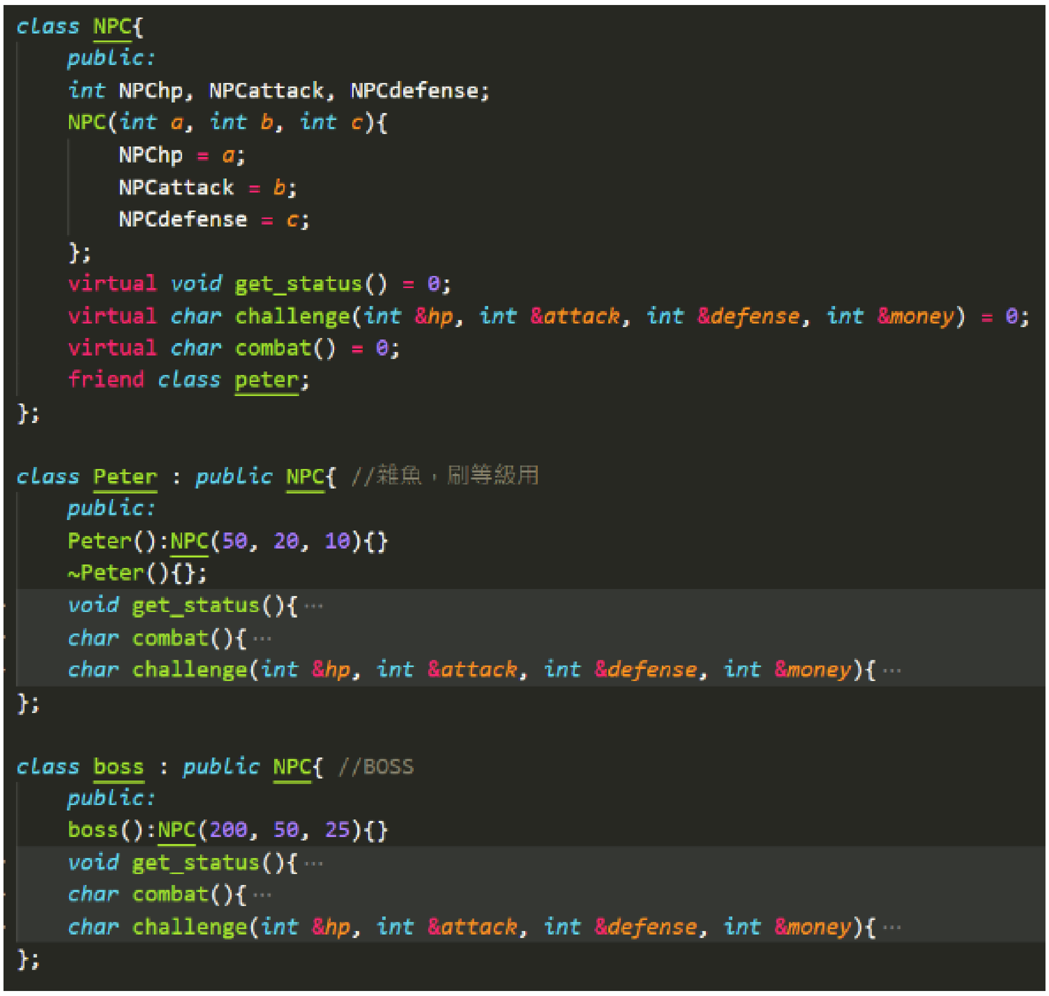


```
class Room{
    private:
        int room_num;
        Room* up;
        Room* down;
        Room* left;
        Room* right;
    public:
        Room():room_num(0), up(0), down(0), left(0), right(0){};
        Room(int a):room_num(a), up(0), down(0), left(0), right(0){};
        friend class linkedRoom;
        friend class Dungeon;
};
```



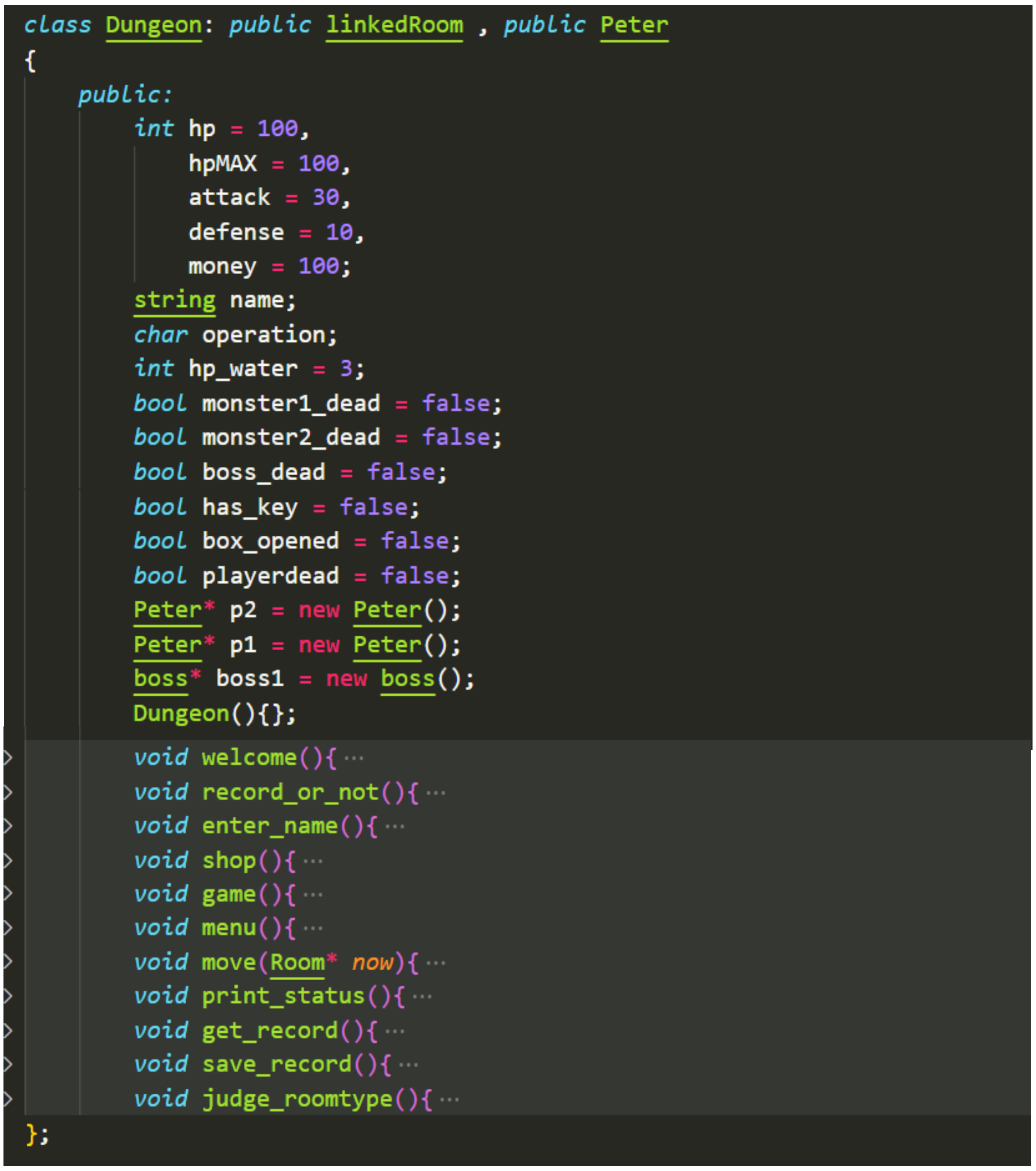
```
class linkedRoom{
    public:
        Room* temp = new Room();
        Room* one = new Room(1);
        Room* two = new Room(2);
        Room* thr = new Room(3);
        Room* fou = new Room(4);
        Room* fiv = new Room(5);
        Room* six = new Room(6);
        Room* sev = new Room(7);
        Room* eig = new Room(8);
        Room* nin = new Room(9);
        Room* current = one;
        Room* previous = temp;
        linkedRoom(){};
        void generate_room(){…
};
```

1.4. Class NPC / Peter / boss



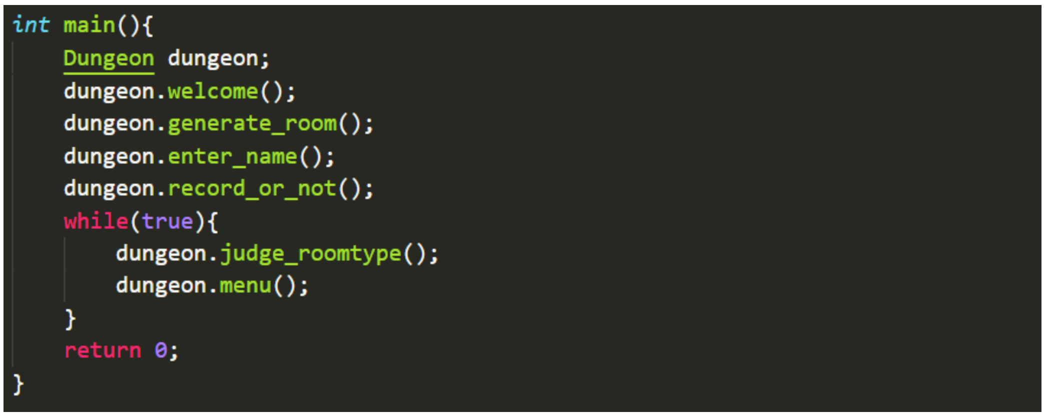
```
class NPC{
    public:
    int NPChp, NPCattack, NPCdefense;
    NPC(int a, int b, int c){
        NPChp = a;
        NPCattack = b;
        NPCdefense = c;
    };
    virtual void get_status() = 0;
    virtual char challenge(int &hp, int &attack, int &defense, int &money) = 0;
    virtual char combat() = 0;
    friend class peter;
};

class Peter : public NPC{ //雜魚，刷等級用
    public:
    Peter():NPC(50, 20, 10){}
    ~Peter(){};
    void get_status(){…
    char combat(){…
    char challenge(int &hp, int &attack, int &defense, int &money){…
};

class boss : public NPC{ //BOSS
    public:
    boss():NPC(200, 50, 25){}
    void get_status(){…
    char combat(){…
    char challenge(int &hp, int &attack, int &defense, int &money){…
};
```

1.5. Class Dungeon



```
class Dungeon: public linkedRoom , public Peter
{
    public:
        int hp = 100,
            hpMAX = 100,
            attack = 30,
            defense = 10,
            money = 100;
        string name;
        char operation;
        int hp_water = 3;
        bool monster1_dead = false;
        bool monster2_dead = false;
        bool boss_dead = false;
        bool has_key = false;
        bool box_opened = false;
        bool playerdead = false;
        Peter* p2 = new Peter();
        Peter* p1 = new Peter();
        boss* boss1 = new boss();
        Dungeon(){};
        void welcome(){ ···
        void record_or_not(){ ···
        void enter_name(){ ···
        void shop(){ ···
        void game(){ ···
        void menu(){ ···
        void move(Room* now){ ···
        void print_status(){ ···
        void get_record(){ ···
        void save_record(){ ···
        void judge_roomtype(){ ···
};
```

1.6. Main Function



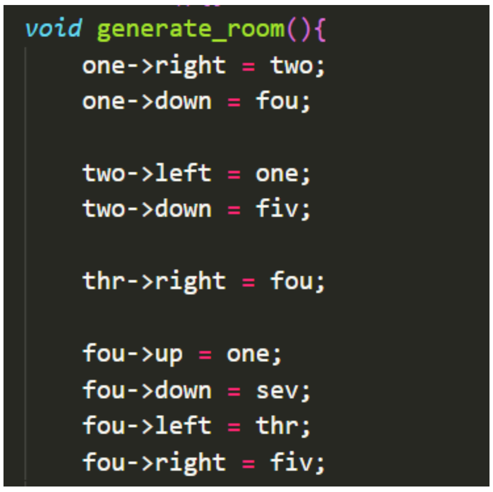
```
int main(){
    Dungeon dungeon;
    dungeon.welcome();
    dungeon.generate_room();
    dungeon.enter_name();
    dungeon.record_or_not();
    while(true){
        dungeon.judge_roomtype();
        dungeon.menu();
    }
    return 0;
}
```

2. Basic Functions

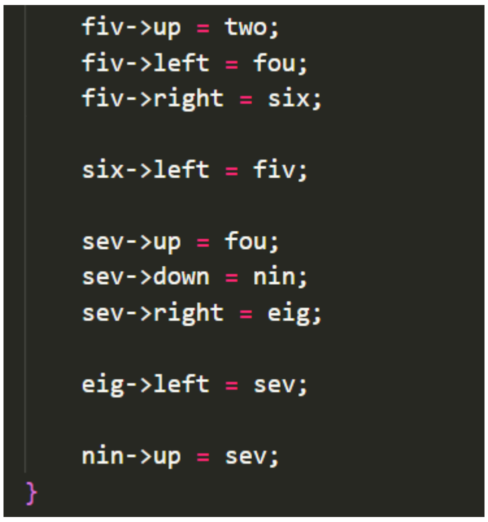
2.1. Dungeon Generate and Movement

我的 dungeon 宣告及數量是由 Class Dungeon / linkedRoom 來完成的，

在 Class Room 裡我先宣告好每一個 dungeon 都會有它的 up, down, right, left 的 pointer 及它的號碼(詳情可見 1.3)，然後再由 Class linkedRoom 裡的 void generate()來把各個生成的 dungeon 串聯在一起:

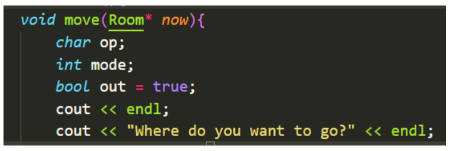


```cpp
void generate_room(){
    one->right = two;
    one->down = fou;

    two->left = one;
    two->down = fiv;

    thr->right = fou;

    fou->up = one;
    fou->down = sev;
    fou->left = thr;
    fou->right = fiv;
```

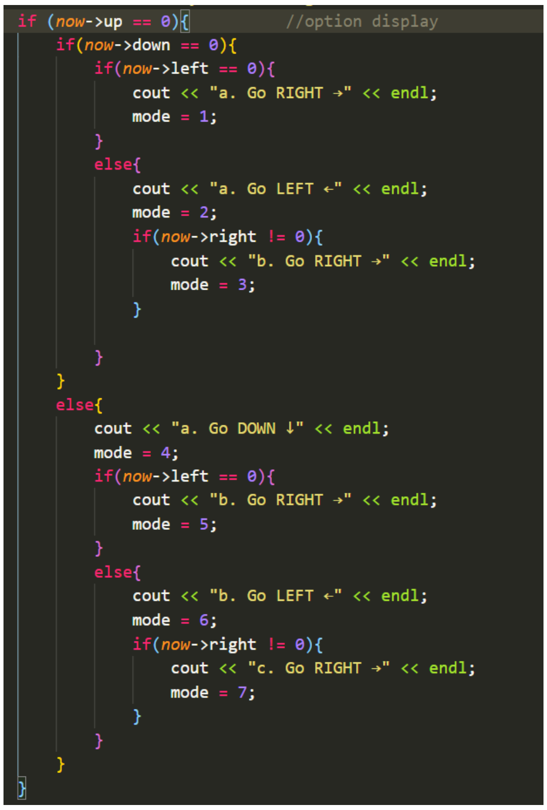


```cpp
    fiv->up = two;
    fiv->left = fou;
    fiv->right = six;

    six->left = fiv;

    sev->up = fou;
    sev->down = nin;
    sev->right = eig;

    eig->left = sev;

    nin->up = sev;
}
```

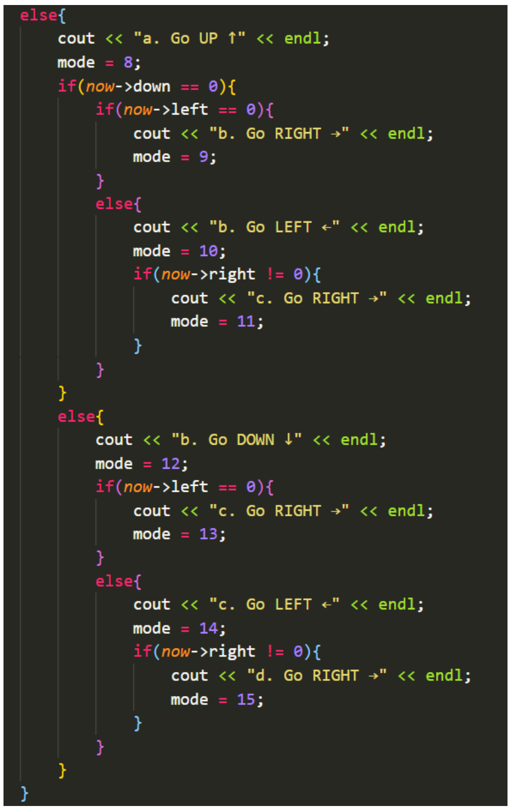
而至於玩家在 dungeon 中的移動，我是藉由 Class Dungeon 裡的方程式 void move(Room* now)來完成，方程一開始先判別現在所在的 dungeon 上下左右有沒有路可走，如果有才顯示出選項，最後再根據使用者輸入選項來讓玩家在地窖內移動:



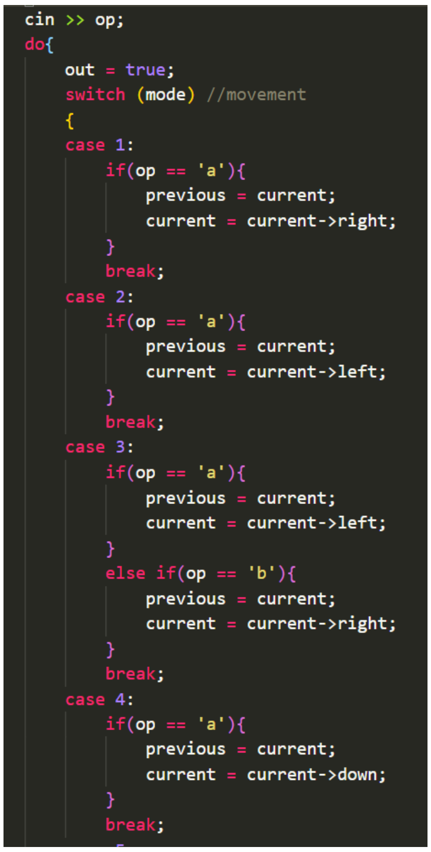
```cpp
void move(Room* now){
    char op;
    int mode;
    bool out = true;
    cout << "Where do you want to go?" << endl;
```



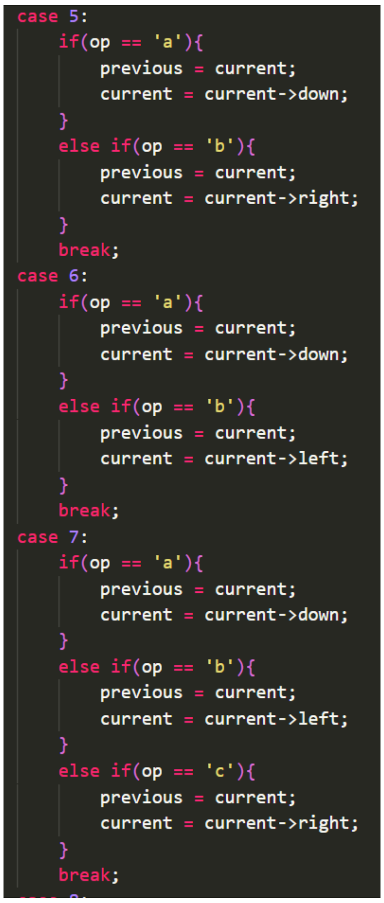
```cpp
if (now->up == 0){          //option display
    if(now->down == 0){
        if(now->left == 0){
            cout << "a. Go RIGHT →" << endl;
            mode = 1;
        }
        else{
            cout << "a. Go LEFT ←" << endl;
            mode = 2;
            if(now->right != 0){
                cout << "b. Go RIGHT →" << endl;
                mode = 3;
            }

        }
    }
    else{
        cout << "a. Go DOWN ↓" << endl;
        mode = 4;
        if(now->left == 0){
            cout << "b. Go RIGHT →" << endl;
            mode = 5;
        }
        else{
            cout << "b. Go LEFT ←" << endl;
            mode = 6;
            if(now->right != 0){
                cout << "c. Go RIGHT →" << endl;
                mode = 7;
            }
        }
    }
}
```



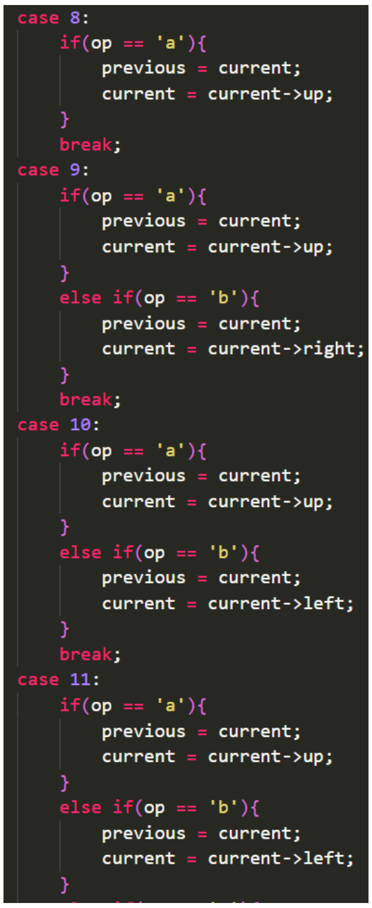
```cpp
else{
    cout << "a. Go UP ↑" << endl;
    mode = 8;
    if(now->down == 0){
        if(now->left == 0){
            cout << "b. Go RIGHT →" << endl;
            mode = 9;
        }
        else{
            cout << "b. Go LEFT ←" << endl;
            mode = 10;
            if(now->right != 0){
                cout << "c. Go RIGHT →" << endl;
                mode = 11;
            }
        }
    }
    else{
        cout << "b. Go DOWN ↓" << endl;
        mode = 12;
        if(now->left == 0){
            cout << "c. Go RIGHT →" << endl;
            mode = 13;
        }
        else{
            cout << "c. Go LEFT ←" << endl;
            mode = 14;
            if(now->right != 0){
                cout << "d. Go RIGHT →" << endl;
                mode = 15;
            }
        }
    }
}
```



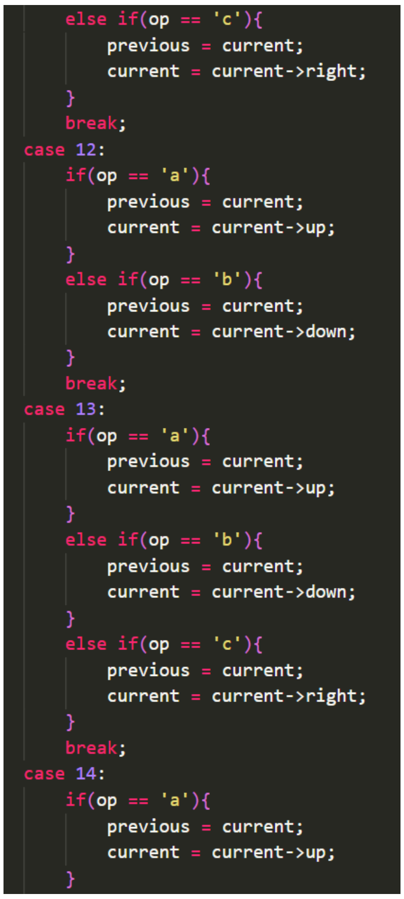
```
cin >> op;
do{
    out = true;
    switch (mode) //movement
    {
    case 1:
        if(op == 'a'){
            previous = current;
            current = current->right;
        }
        break;
    case 2:
        if(op == 'a'){
            previous = current;
            current = current->left;
        }
        break;
    case 3:
        if(op == 'a'){
            previous = current;
            current = current->left;
        }
        else if(op == 'b'){
            previous = current;
            current = current->right;
        }
        break;
    case 4:
        if(op == 'a'){
            previous = current;
            current = current->down;
        }
        break;
```



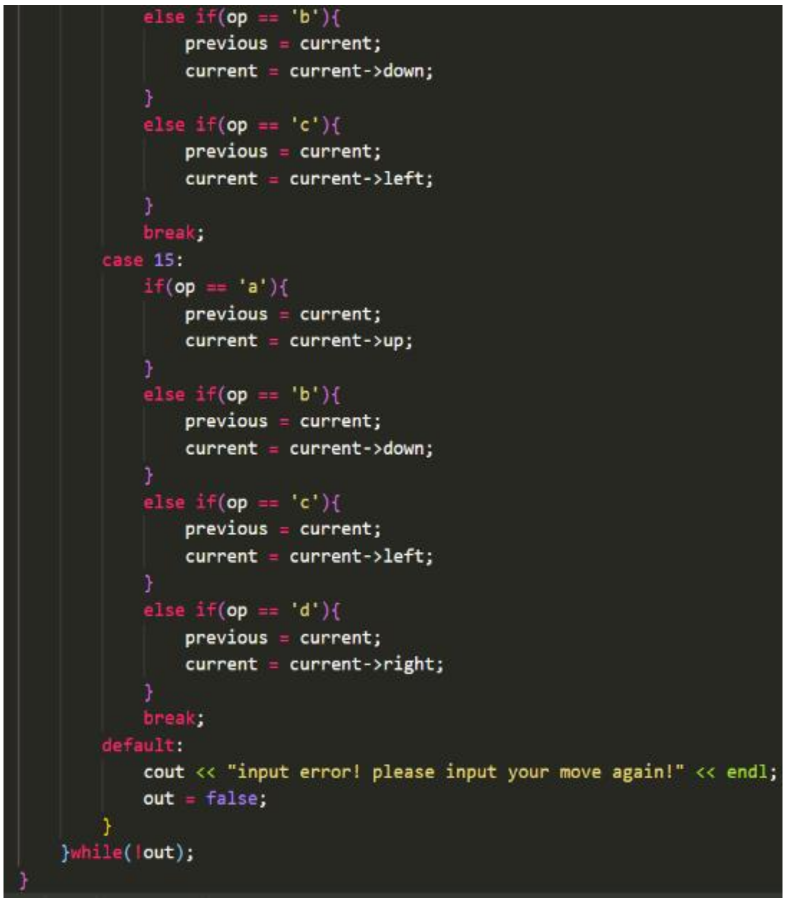
```
case 5:
    if(op == 'a'){
        previous = current;
        current = current->down;
    }
    else if(op == 'b'){
        previous = current;
        current = current->right;
    }
    break;
case 6:
    if(op == 'a'){
        previous = current;
        current = current->down;
    }
    else if(op == 'b'){
        previous = current;
        current = current->left;
    }
    break;
case 7:
    if(op == 'a'){
        previous = current;
        current = current->down;
    }
    else if(op == 'b'){
        previous = current;
        current = current->left;
    }
    else if(op == 'c'){
        previous = current;
        current = current->right;
    }
    break;
```



```
case 8:
    if(op == 'a'){
        previous = current;
        current = current->up;
    }
    break;
case 9:
    if(op == 'a'){
        previous = current;
        current = current->up;
    }
    else if(op == 'b'){
        previous = current;
        current = current->right;
    }
    break;
case 10:
    if(op == 'a'){
        previous = current;
        current = current->up;
    }
    else if(op == 'b'){
        previous = current;
        current = current->left;
    }
    break;
case 11:
    if(op == 'a'){
        previous = current;
        current = current->up;
    }
    else if(op == 'b'){
        previous = current;
        current = current->left;
    }
```



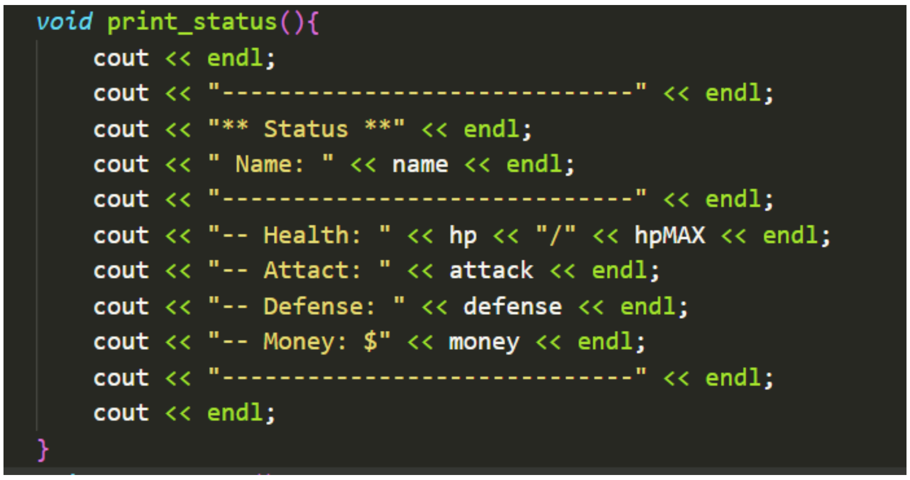
```
    else if(op == 'c'){
        previous = current;
        current = current->right;
    }
    break;
case 12:
    if(op == 'a'){
        previous = current;
        current = current->up;
    }
    else if(op == 'b'){
        previous = current;
        current = current->down;
    }
    break;
case 13:
    if(op == 'a'){
        previous = current;
        current = current->up;
    }
    else if(op == 'b'){
        previous = current;
        current = current->down;
    }
    else if(op == 'c'){
        previous = current;
        current = current->right;
    }
    break;
case 14:
    if(op == 'a'){
        previous = current;
        current = current->up;
    }
```



```
            else if(op == 'b'){
                previous = current;
                current = current->down;
            }
            else if(op == 'c'){
                previous = current;
                current = current->left;
            }
            break;
        case 15:
            if(op == 'a'){
                previous = current;
                current = current->up;
            }
            else if(op == 'b'){
                previous = current;
                current = current->down;
            }
            else if(op == 'c'){
                previous = current;
                current = current->left;
            }
            else if(op == 'd'){
                previous = current;
                current = current->right;
            }
            break;
        default:
            cout << "input error! please input your move again!" << endl;
            out = false;
        }
    }while(!out);
}
```

2.2. Showing Status

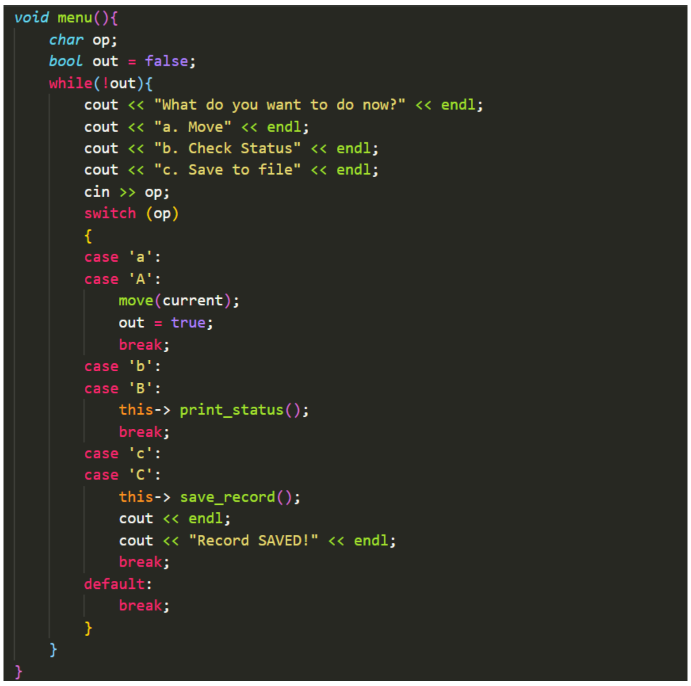
遊戲中有選項可以顯示玩家當下的資訊，包含血量、攻擊力、防禦力、金錢，而顯示是由 Class Dungeon 中的 void print_status()來完成:



```
void print_status(){
    cout << endl;
    cout << "------------------------------" << endl;
    cout << "** Status **" << endl;
    cout << " Name: " << name << endl;
    cout << "------------------------------" << endl;
    cout << "-- Health: " << hp << "/" << hpMAX << endl;
    cout << "-- Attact: " << attack << endl;
    cout << "-- Defense: " << defense << endl;
    cout << "-- Money: $" << money << endl;
    cout << "------------------------------" << endl;
    cout << endl;
}
```

2.3. Action Menu

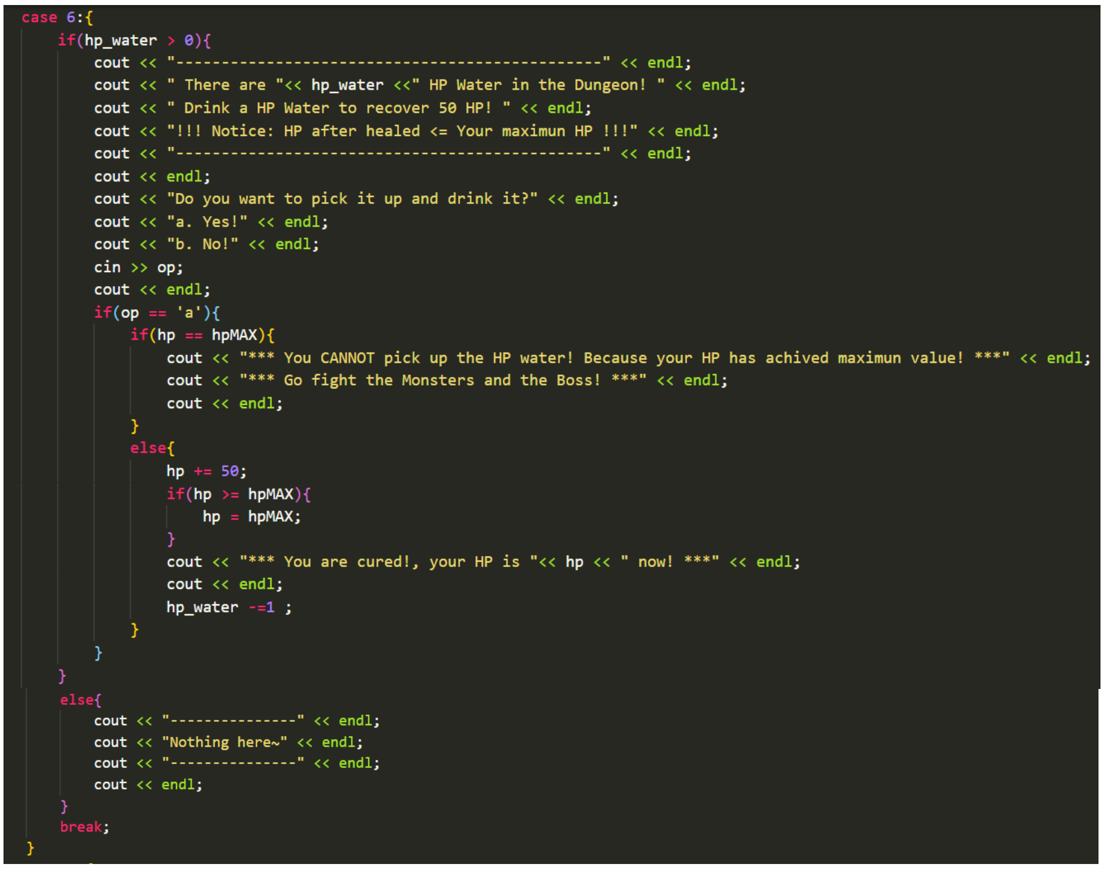
玩家透過選單可以完成基本操作，包含移動、查看自身狀態和存檔，這是由 Class Dungeon 中的 void menu()來完成的:



```
void menu(){
    char op;
    bool out = false;
    while(!out){
        cout << "What do you want to do now?" << endl;
        cout << "a. Move" << endl;
        cout << "b. Check Status" << endl;
        cout << "c. Save to file" << endl;
        cin >> op;
        switch (op)
        {
        case 'a':
        case 'A':
            move(current);
            out = true;
            break;
        case 'b':
        case 'B':
            this-> print_status();
            break;
        case 'c':
        case 'C':
            this-> save_record();
            cout << endl;
            cout << "Record SAVED!" << endl;
            break;
        default:
            break;
        }
    }
}
```

2.4. Pick up Items

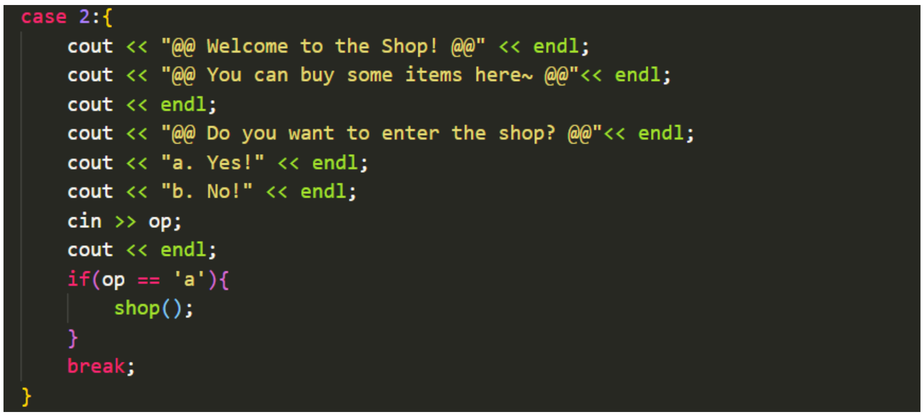
在 6 號 dungeon 中有 3 瓶回血藥水給玩家拿取，每喝一瓶可以恢復 50 點生命值，但回覆的上限為玩家的最大生命值，沒辦法回復到爆表。當三瓶回血藥水都被喝完後，6 號 dungeon 就變成一個空的 dungeon。實作的程式碼位於 Class Dungeon 中的 void judge_roomtype()裡:



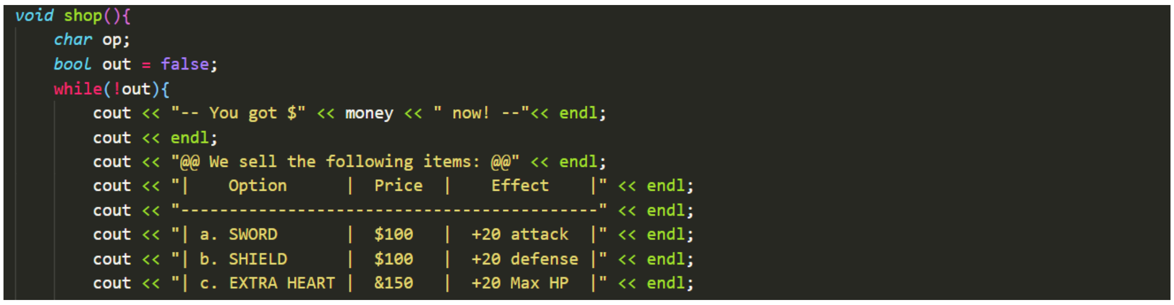
```
case 6:{
    if(hp_water > 0){
        cout << "-------------------------------------------------" << endl;
        cout << " There are "<< hp_water <<" HP Water in the Dungeon! " << endl;
        cout << " Drink a HP Water to recover 50 HP! " << endl;
        cout << "!!! Notice: HP after healed <= Your maximun HP !!!" << endl;
        cout << "-------------------------------------------------" << endl;
        cout << endl;
        cout << "Do you want to pick it up and drink it?" << endl;
        cout << "a. Yes!" << endl;
        cout << "b. No!" << endl;
        cin >> op;
        cout << endl;
        if(op == 'a'){
            if(hp == hpMAX){
                cout << "*** You CANNOT pick up the HP water! Because your HP has achived maximun value! ***" << endl;
                cout << "*** Go fight the Monsters and the Boss! ***" << endl;
                cout << endl;
            }
            else{
                hp += 50;
                if(hp >= hpMAX){
                    hp = hpMAX;
                }
                cout << "*** You are cured!, your HP is "<< hp << " now! ***" << endl;
                cout << endl;
                hp_water -=1 ;
            }
        }
    }
    else{
        cout << "---------------" << endl;
        cout << "Nothing here~" << endl;
        cout << "---------------" << endl;
        cout << endl;
    }
    break;
}
```

2.5. NPC

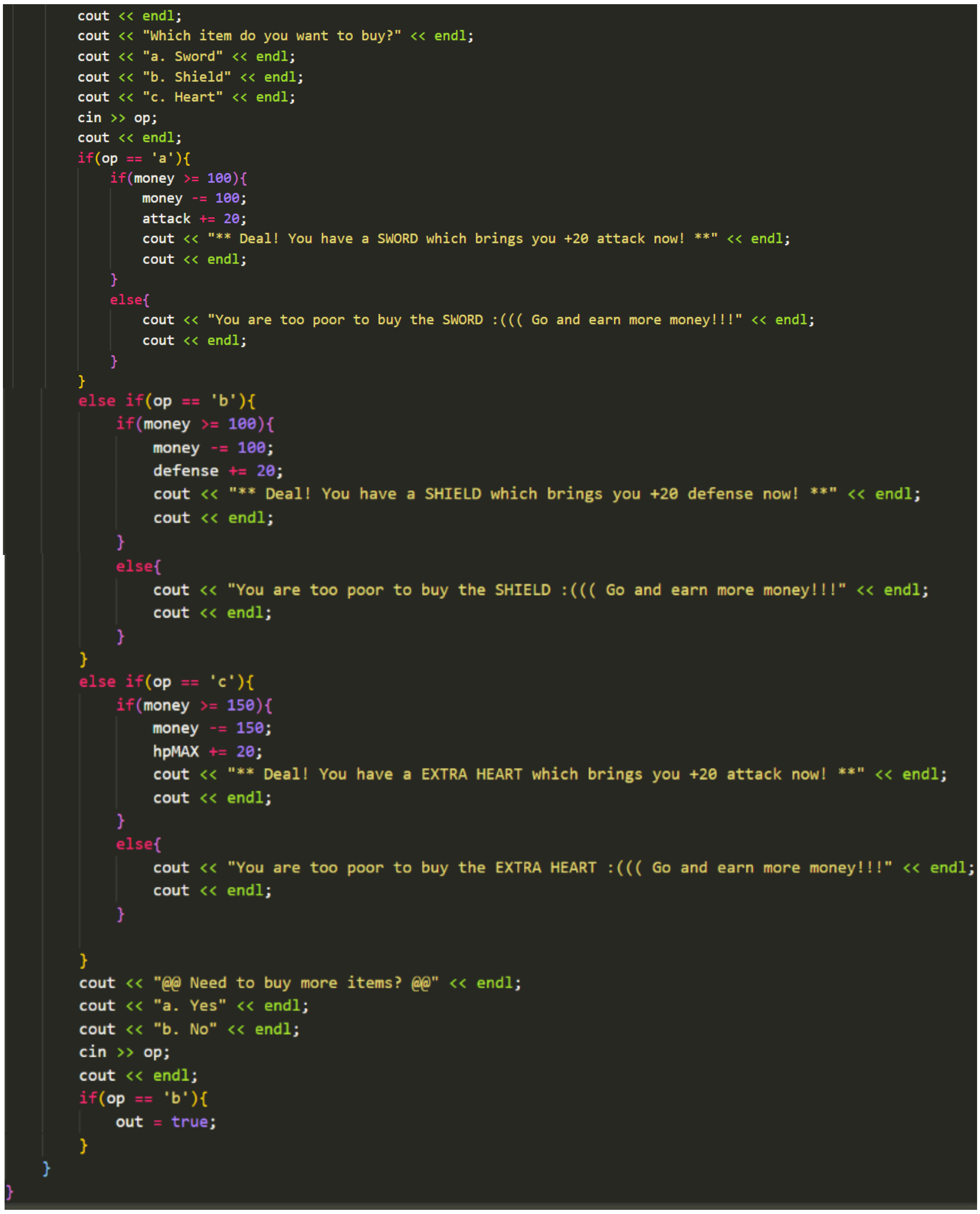
我的dungeon中有三種NPC角色，分別是商人、小怪獸和大Boss。有關商人的部分，它的商店有賣劍、盾牌及額外生命，玩家可以用自己的錢去跟商人買東西，而這部分由 Class Dungeon 中的 void judge_roomtype()和 void shop()來實現:



```
case 2:{
    cout << "@@ Welcome to the Shop! @@" << endl;
    cout << "@@ You can buy some items here~ @@"<< endl;
    cout << endl;
    cout << "@@ Do you want to enter the shop? @@"<< endl;
    cout << "a. Yes!" << endl;
    cout << "b. No!" << endl;
    cin >> op;
    cout << endl;
    if(op == 'a'){
        shop();
    }
    break;
}
```



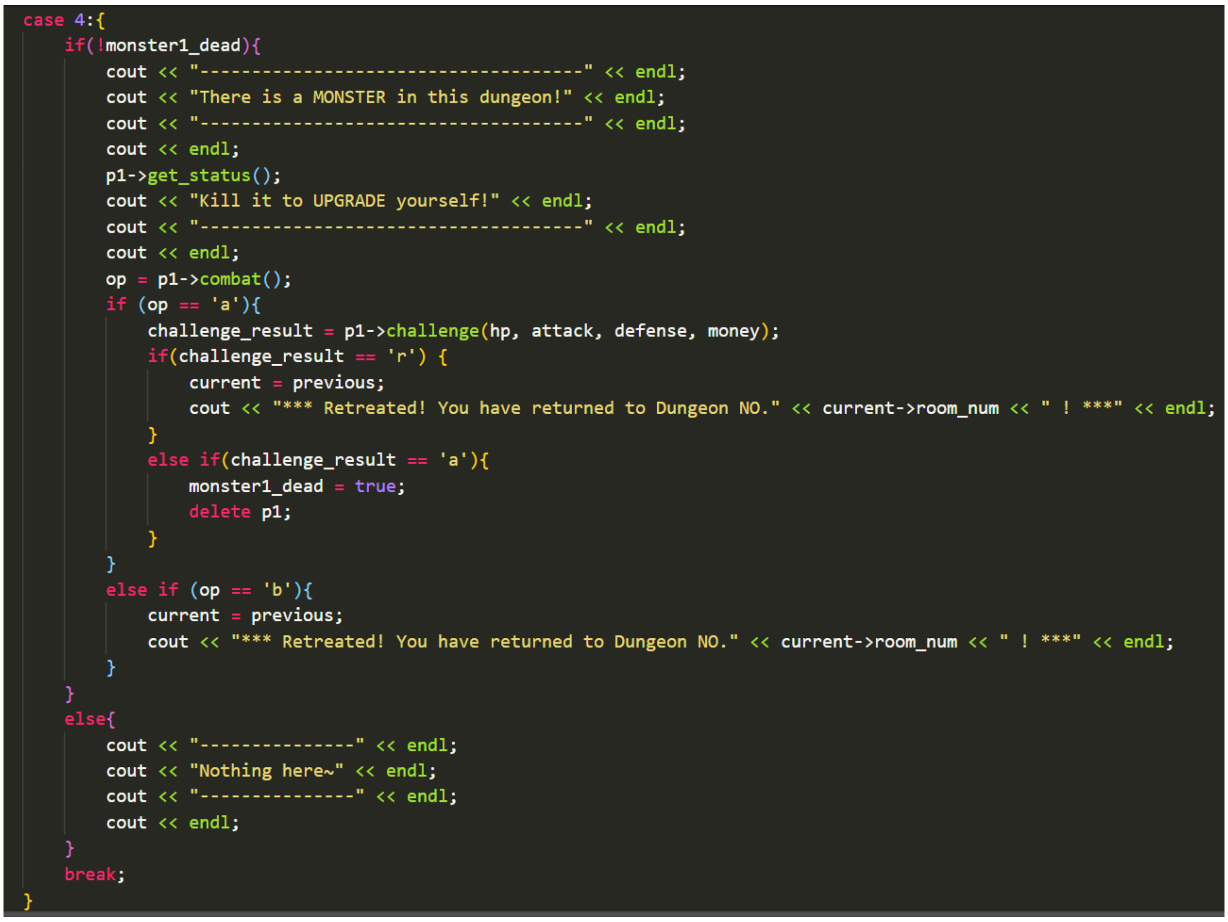
```
void shop(){
    char op;
    bool out = false;
    while(!out){
        cout << "-- You got $" << money << " now! --"<< endl;
        cout << endl;
        cout << "@@ We sell the following items: @@" << endl;
        cout << "|    Option      |  Price  |    Effect    |" << endl;
        cout << "-------------------------------------------" << endl;
        cout << "| a. SWORD       |  $100   |  +20 attack  |" << endl;
        cout << "| b. SHIELD      |  $100   |  +20 defense |" << endl;
        cout << "| c. EXTRA HEART |  &150   |  +20 Max HP  |" << endl;
```



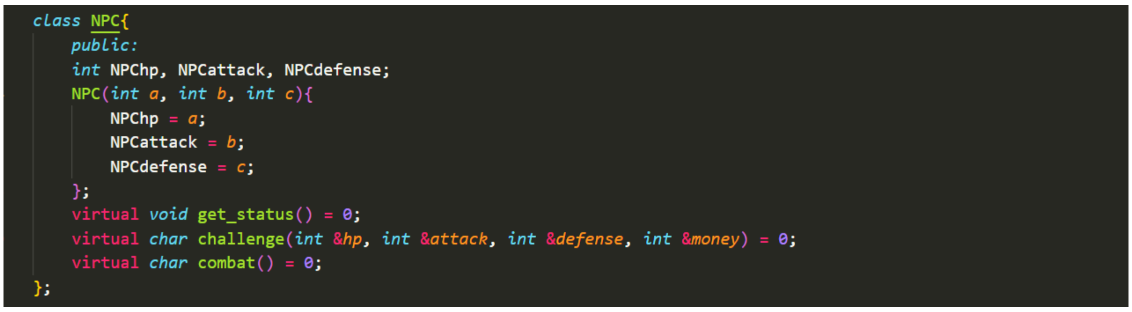
```
                cout << endl;
                cout << "Which item do you want to buy?" << endl;
                cout << "a. Sword" << endl;
                cout << "b. Shield" << endl;
                cout << "c. Heart" << endl;
                cin >> op;
                cout << endl;
                if(op == 'a'){
                    if(money >= 100){
                        money -= 100;
                        attack += 20;
                        cout << "** Deal! You have a SWORD which brings you +20 attack now! **" << endl;
                        cout << endl;
                    }
                    else{
                        cout << "You are too poor to buy the SWORD :((( Go and earn more money!!!" << endl;
                        cout << endl;
                    }
                }
                else if(op == 'b'){
                    if(money >= 100){
                        money -= 100;
                        defense += 20;
                        cout << "** Deal! You have a SHIELD which brings you +20 defense now! **" << endl;
                        cout << endl;
                    }
                    else{
                        cout << "You are too poor to buy the SHIELD :((( Go and earn more money!!!" << endl;
                        cout << endl;
                    }
                }
                else if(op == 'c'){
                    if(money >= 150){
                        money -= 150;
                        hpMAX += 20;
                        cout << "** Deal! You have a EXTRA HEART which brings you +20 attack now! **" << endl;
                        cout << endl;
                    }
                    else{
                        cout << "You are too poor to buy the EXTRA HEART :((( Go and earn more money!!!" << endl;
                        cout << endl;
                    }

                }
                cout << "@@ Need to buy more items? @@" << endl;
                cout << "a. Yes" << endl;
                cout << "b. No" << endl;
                cin >> op;
                cout << endl;
                if(op == 'b'){
                    out = true;
                }
    }
}
```

接下來是小怪獸，遊戲中一共有兩隻小怪獸，分別在兩個不同的 dungeon 中，他們的血量、攻擊力和防禦力都是一樣的，玩家遇到小怪獸時可以選擇跟它戰鬥又或是撤退，在沒有把該 dungeon 中的小怪獸打敗前，玩家是無法通過該 dungeon 的，如果打敗了小怪獸的話，小怪獸就會消失，該 dungeon 內就什麼也不剩了，玩家也就此進入該 dungeon，而小怪獸的功能由 Class Dungeon 的 void judge_roomtype(), Class NPC 和 Class Peter 相輔相成，以

下以在 4 號 dungeon 的小怪獸 monster1 為例:

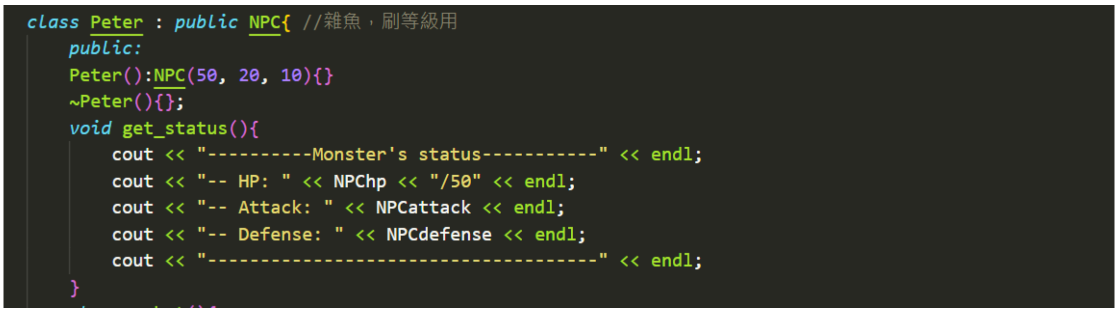


```
case 4:{
    if(!monster1_dead){
        cout << "-------------------------------------" << endl;
        cout << "There is a MONSTER in this dungeon!" << endl;
        cout << "-------------------------------------" << endl;
        cout << endl;
        p1->get_status();
        cout << "Kill it to UPGRADE yourself!" << endl;
        cout << "-------------------------------------" << endl;
        cout << endl;
        op = p1->combat();
        if (op == 'a'){
            challenge_result = p1->challenge(hp, attack, defense, money);
            if(challenge_result == 'r') {
                current = previous;
                cout << "*** Retreated! You have returned to Dungeon NO." << current->room_num << " ! ***" << endl;
            }
            else if(challenge_result == 'a'){
                monster1_dead = true;
                delete p1;
            }
        }
        else if (op == 'b'){
            current = previous;
            cout << "*** Retreated! You have returned to Dungeon NO." << current->room_num << " ! ***" << endl;
        }
    }
    else{
        cout << "---------------" << endl;
        cout << "Nothing here~" << endl;
        cout << "---------------" << endl;
        cout << endl;
    }
    break;
}
```

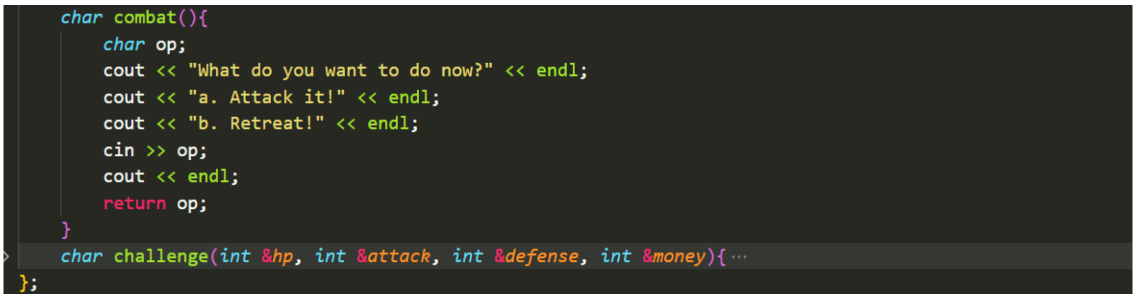


```
class NPC{
    public:
    int NPChp, NPCattack, NPCdefense;
    NPC(int a, int b, int c){
        NPChp = a;
        NPCattack = b;
        NPCdefense = c;
    };
    virtual void get_status() = 0;
    virtual char challenge(int &hp, int &attack, int &defense, int &money) = 0;
    virtual char combat() = 0;
};
```

在 Class Peter 中，void get_status()會顯示出怪獸的狀態，char combat 適用於判斷玩家還想不想跟怪獸戰鬥，如果選擇戰鬥則進入戰鬥系統(詳見 2.6)，如果選擇撤退就退回上一個 dungeon，程式碼如下:

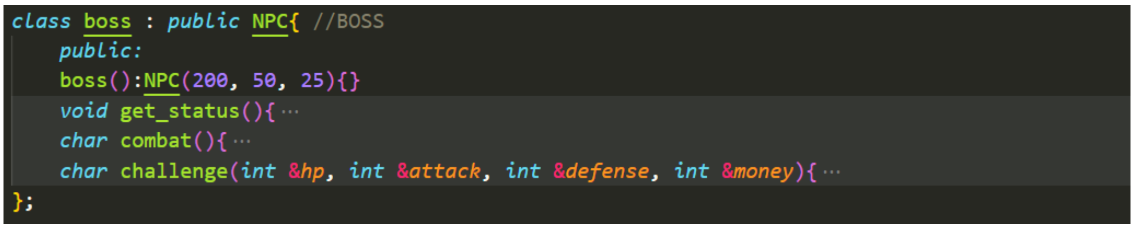


```
class Peter : public NPC{ //雜魚，刷等級用
    public:
    Peter():NPC(50, 20, 10){}
    ~Peter(){};
    void get_status(){
        cout << "----------Monster's status-----------" << endl;
        cout << "-- HP: " << NPChp << "/50" << endl;
        cout << "-- Attack: " << NPCattack << endl;
        cout << "-- Defense: " << NPCdefense << endl;
        cout << "-------------------------------------" << endl;
    }
```



```cpp
    char combat(){
        char op;
        cout << "What do you want to do now?" << endl;
        cout << "a. Attack it!" << endl;
        cout << "b. Retreat!" << endl;
        cin >> op;
        cout << endl;
        return op;
    }
    char challenge(int &hp, int &attack, int &defense, int &money){···
};
```

最後是大 Boss 的部分，基本上所有攻擊和撤退邏輯是和是一樣的，所以這裡我只貼上程式碼的架構:



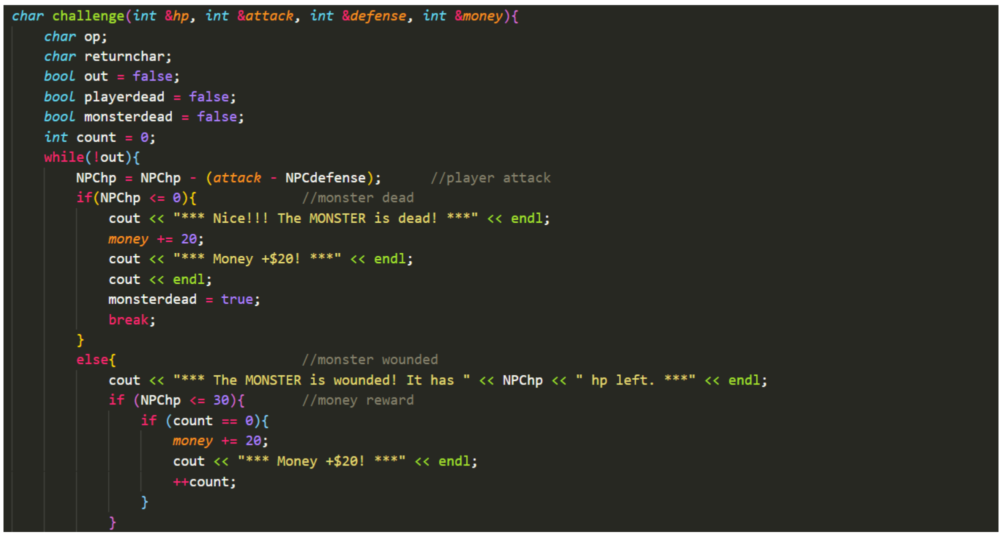
```cpp
class boss : public NPC{ //BOSS
    public:
    boss():NPC(200, 50, 25){}
    void get_status(){···
    char combat(){···
    char challenge(int &hp, int &attack, int &defense, int &money){···
};
```

2.6. Fighting System

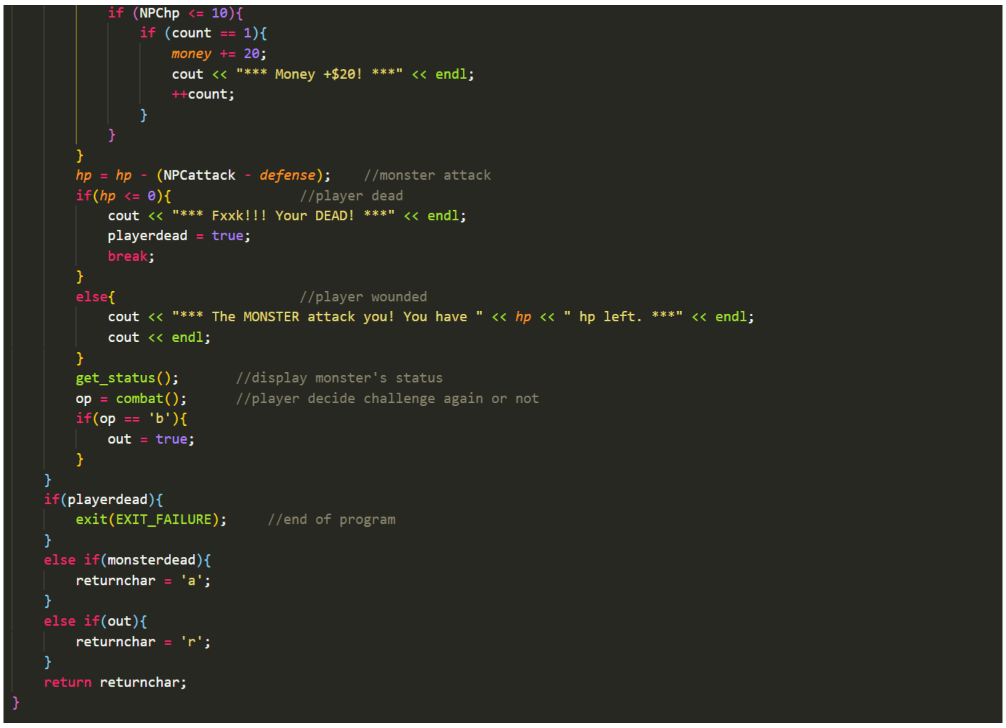
在玩家進入戰鬥系統，選擇攻擊後，NPC 即遭到玩家攻擊，而玩家也會遭到 NPC 的攻擊，兩者扣血的計算邏輯皆為:

$$\boxed{\text{戰鬥後血量} = \text{原有血量} - (\text{敵方攻擊力} - \text{自身防禦力})}$$

在兩方都攻擊完後，遊戲顯示出 NPC 的狀態及玩家的血量，這時玩家可以選擇要不要繼續戰鬥，若選擇繼續則繼續戰鬥，則戰鬥會持續到 NPC 死亡、玩家死亡或玩家選擇撤退為止，若選擇撤退，則玩家退回上一個所在的 dungeon。在戰鬥後，NPC 的血量會一直被保留，直到 NPC 或玩家死亡為止，在程式碼時實作的部份，我們在 Class NPC 中做 Virtual char challenge()的宣告，然後分別在代表小怪獸的 Class Peter 和 Class boss 中實作 char challenge()。由於 Class Peter 和 Class boss 內的 char challenge()邏輯是一樣的，所以這裡只貼上 Class Peter 的 char challenge():



```cpp
char challenge(int &hp, int &attack, int &defense, int &money){
    char op;
    char returnchar;
    bool out = false;
    bool playerdead = false;
    bool monsterdead = false;
    int count = 0;
    while(!out){
        NPChp = NPChp - (attack - NPCdefense);      //player attack
        if(NPChp <= 0){                 //monster dead
            cout << "*** Nice!!! The MONSTER is dead! ***" << endl;
            money += 20;
            cout << "*** Money +$20! ***" << endl;
            cout << endl;
            monsterdead = true;
            break;
        }
        else{                           //monster wounded
            cout << "*** The MONSTER is wounded! It has " << NPChp << " hp left. ***" << endl;
            if (NPChp <= 30){           //money reward
                if (count == 0){
                    money += 20;
                    cout << "*** Money +$20! ***" << endl;
                    ++count;
                }
            }
```



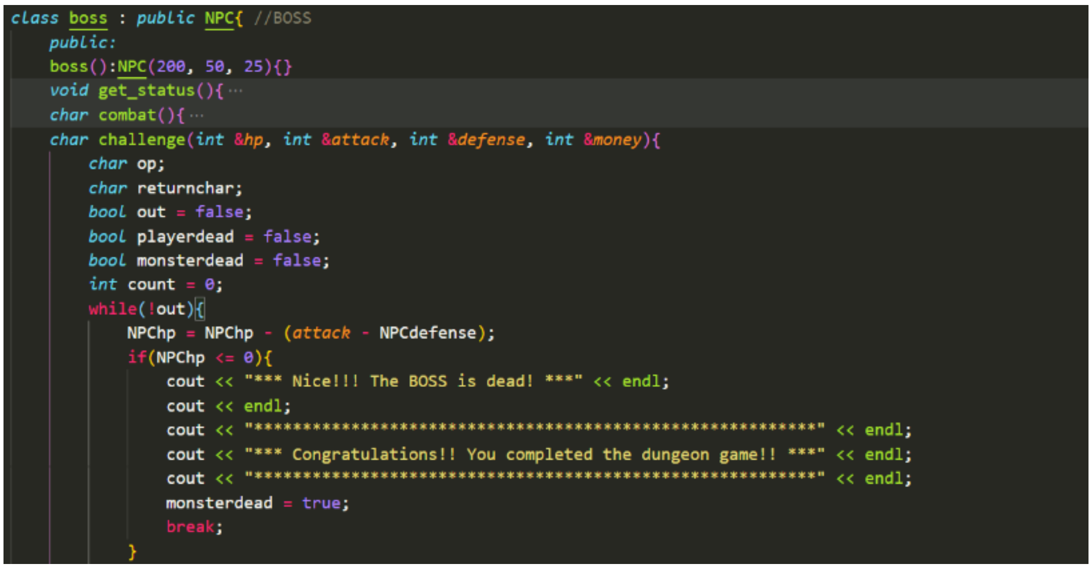
```
                if (NPChp <= 10){
                    if (count == 1){
                        money += 20;
                        cout << "*** Money +$20! ***" << endl;
                        ++count;
                    }
                }
            }
            hp = hp - (NPCattack - defense);    //monster attack
            if(hp <= 0){                //player dead
                cout << "*** Fxxk!!! Your DEAD! ***" << endl;
                playerdead = true;
                break;
            }
            else{                       //player wounded
                cout << "*** The MONSTER attack you! You have " << hp << " hp left. ***" << endl;
                cout << endl;
            }
            get_status();       //display monster's status
            op = combat();      //player decide challenge again or not
            if(op == 'b'){
                out = true;
            }
        }
        if(playerdead){
            exit(EXIT_FAILURE);     //end of program
        }
        else if(monsterdead){
            returnchar = 'a';
        }
        else if(out){
            returnchar = 'r';
        }
        return returnchar;
}
```

2.7. Game Logic

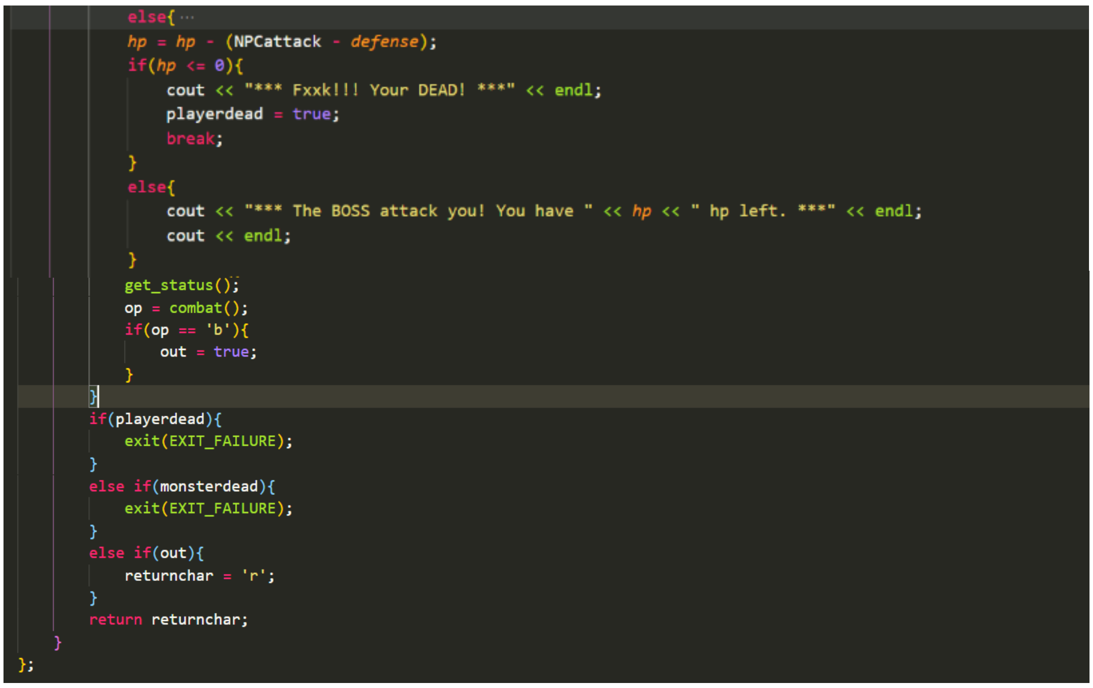
在遊戲中，扣除手動刻意中止程式的情況，能結束或中斷遊戲的情況只有下列三個:

a. 玩家打敗大 Boss
b. 玩家被大 Boss 或小怪獸打死
c. 在賭場(詳見 3.2)賭到輸光生命值

如果玩家打敗 Boss，代表玩家勝利，遊戲會出現恭喜訊息並中止遊戲。如果玩家被打到生命值<0 的話，遊戲會顯示你死了，並中止遊戲。這對應在 Class boss 和 Class Peter 中的 char challenge()，由於兩個 Class 中關於玩家沒血死亡的邏輯幾乎相同，但只有 Class boss 中有打敗 boss 而中止遊戲的情況，所以我們這裡舉 Class boss 中的 char challenge()為例:

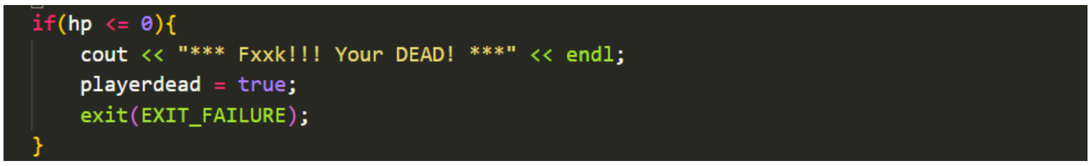


```
class boss : public NPC{ //BOSS
    public:
    boss():NPC(200, 50, 25){}
    void get_status(){…
    char combat(){…
    char challenge(int &hp, int &attack, int &defense, int &money){
        char op;
        char returnchar;
        bool out = false;
        bool playerdead = false;
        bool monsterdead = false;
        int count = 0;
        while(!out){
            NPChp = NPChp - (attack - NPCdefense);
            if(NPChp <= 0){
                cout << "*** Nice!!! The BOSS is dead! ***" << endl;
                cout << endl;
                cout << "************************************************************" << endl;
                cout << "*** Congratulations!! You completed the dungeon game!! ***" << endl;
                cout << "************************************************************" << endl;
                monsterdead = true;
                break;
            }
```



```
            else{…
            hp = hp - (NPCattack - defense);
            if(hp <= 0){
                cout << "*** Fxxk!!! Your DEAD! ***" << endl;
                playerdead = true;
                break;
            }
            else{
                cout << "*** The BOSS attack you! You have " << hp << " hp left. ***" << endl;
                cout << endl;
            }
            get_status();
            op = combat();
            if(op == 'b'){
                out = true;
            }
        }
        if(playerdead){
            exit(EXIT_FAILURE);
        }
        else if(monsterdead){
            exit(EXIT_FAILURE);
        }
        else if(out){
            returnchar = 'r';
        }
        return returnchar;
    }
};
```

最後則是在賭場(詳見 3.2)賭到死亡的情況，對應的程式碼在 Class Dungeon 中的 void game()內:



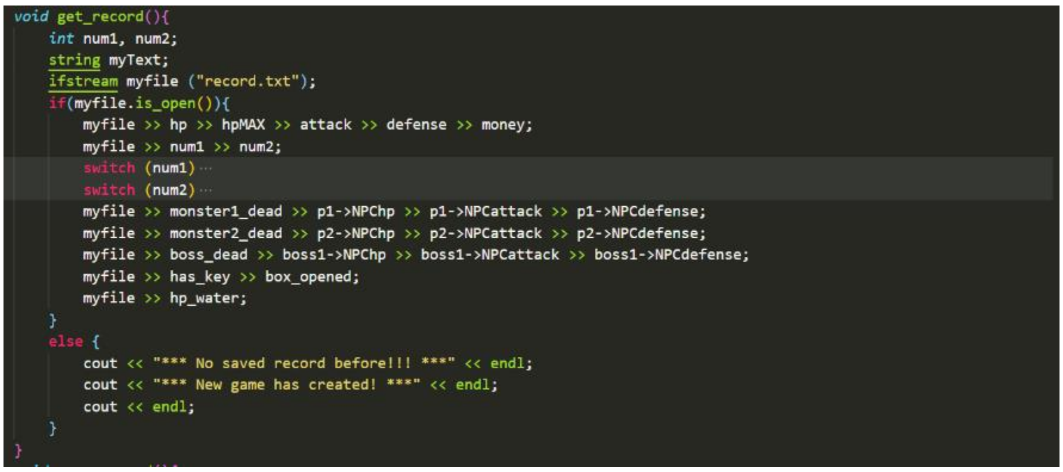
```
if(hp <= 0){
    cout << "*** Fxxk!!! Your DEAD! ***" << endl;
    playerdead = true;
    exit(EXIT_FAILURE);
}
```

2.8. Record System

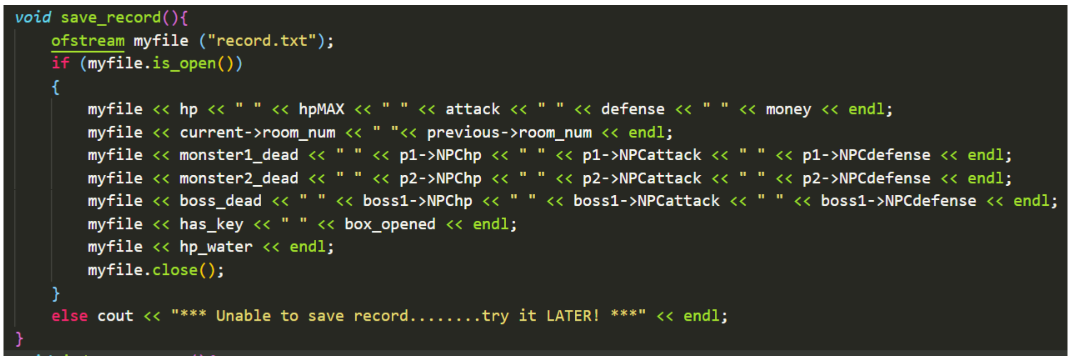
遊戲可以在剛開始時決定要不要載入先前儲存的遊戲紀錄，若沒有之前的遊戲紀錄的話，視窗會顯示無紀錄，並開始新遊戲。若有紀錄的話則會直接跳到記錄內相應的 dungeon 並繼續遊戲。玩家也可以在每進入到一個 dungeon 內時選擇存檔遊戲紀錄，程式會記錄當下的遊戲參數並輸出成文字檔(.txt)。紀錄會記錄的參數有以下幾個:

a. 玩家血量、最大血量、攻擊力、防禦力和金錢
b. 玩家所在 / 前一個所在的 dungeon 號碼
c. 小怪獸 1 / 2 狀態(bool variable)、血量、攻擊力和防禦力
d. 大 Boss 狀態(bool variable)、血量、攻擊力和防禦力
e. 玩家有無鑰匙(詳見 3.3) (bool variable)
f. 寶箱(詳見 3.3)有無被打開(bool variable)
g. 回血藥水剩幾瓶

而存檔及載入紀錄分別由 Class Dungeon 中的 void get_record() 和 void save_record()執行:

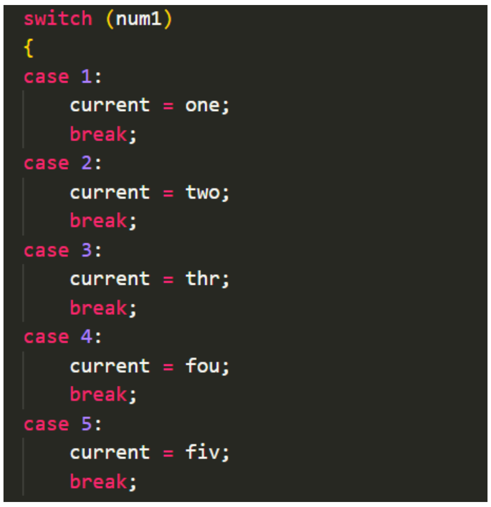


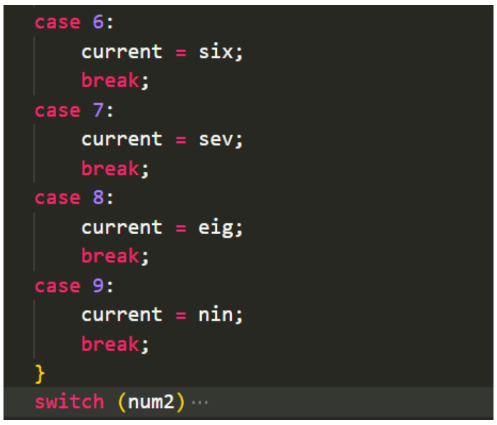
```cpp
void get_record(){
    int num1, num2;
    string myText;
    ifstream myfile ("record.txt");
    if(myfile.is_open()){
        myfile >> hp >> hpMAX >> attack >> defense >> money;
        myfile >> num1 >> num2;
        switch (num1)···
        switch (num2)···
        myfile >> monster1_dead >> p1->NPChp >> p1->NPCattack >> p1->NPCdefense;
        myfile >> monster2_dead >> p2->NPChp >> p2->NPCattack >> p2->NPCdefense;
        myfile >> boss_dead >> boss1->NPChp >> boss1->NPCattack >> boss1->NPCdefense;
        myfile >> has_key >> box_opened;
        myfile >> hp_water;
    }
    else {
        cout << "*** No saved record before!!! ***" << endl;
        cout << "*** New game has created! ***" << endl;
        cout << endl;
    }
}
```



```cpp
void save_record(){
    ofstream myfile ("record.txt");
    if (myfile.is_open())
    {
        myfile << hp << " " << hpMAX << " " << attack << " " << defense << " " << money << endl;
        myfile << current->room_num << " "<< previous->room_num << endl;
        myfile << monster1_dead << " " << p1->NPChp << " " << p1->NPCattack << " " << p1->NPCdefense << endl;
        myfile << monster2_dead << " " << p2->NPChp << " " << p2->NPCattack << " " << p2->NPCdefense << endl;
        myfile << boss_dead << " " << boss1->NPChp << " " << boss1->NPCattack << " " << boss1->NPCdefense << endl;
        myfile << has_key << " " << box_opened << endl;
        myfile << hp_water << endl;
        myfile.close();
    }
    else cout << "*** Unable to save record........try it LATER! ***" << endl;
}
```

其中，void get_record()中的 switch 是用來定位玩家位置用的，num1 定位現在位置，num2 定位上一個所在位置。由於邏輯相似，這裡僅貼上判斷 num1 的程式碼:





```cpp
switch (num1)
{
case 1:
    current = one;
    break;
case 2:
    current = two;
    break;
case 3:
    current = thr;
    break;
case 4:
    current = fou;
    break;
case 5:
    current = fiv;
    break;
case 6:
    current = six;
    break;
case 7:
    current = sev;
    break;
case 8:
    current = eig;
    break;
case 9:
    current = nin;
    break;
}
switch (num2)···
```

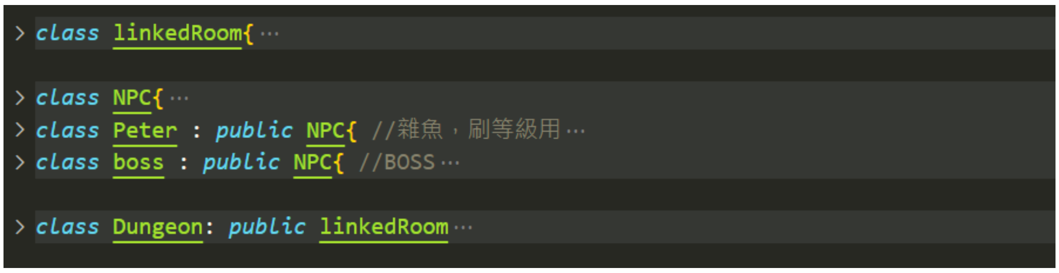
2.9. Inheritance and Virtual Function

本次作業中有要求要使用到 Inheritance 及 Virtual Function 的技巧，我們分開來看:

a. Inheritance

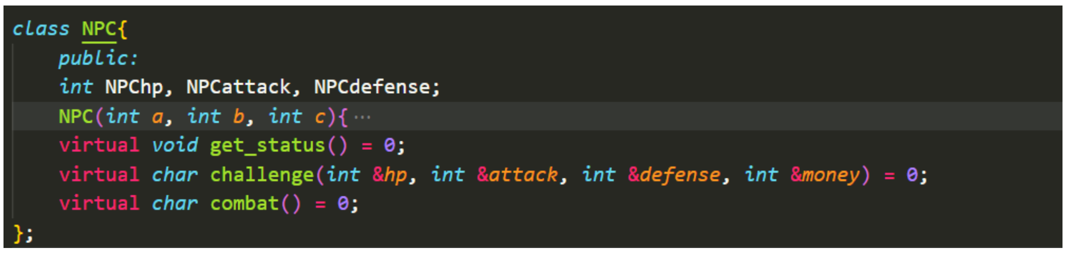
我的程式中有兩處繼承的部分，第一個是 Class Dungeon，繼承了 Class linkedRoom，用於在 Class Dungeon 中移動在 Class linkedRoom 裡宣告的各個 dungeon。第二個是 Class

Peter 和 Class boss，兩個都繼承了 Class NPC，用以使用在 Class NPC 就宣告好了的 hp, attack 和 defense:

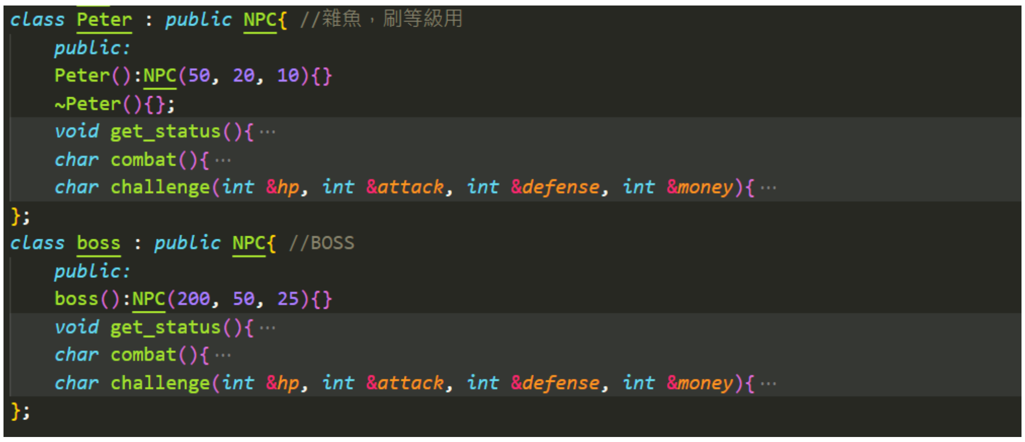

```
> class linkedRoom{…

> class NPC{…
> class Peter : public NPC{ //雜魚，刷等級用…
> class boss : public NPC{ //BOSS…

> class Dungeon: public linkedRoom…
```

b. **Virtual Function**

在 Class NPC 中，我宣告了三個 Virtual Function，分別是 virtual void get_status(), virtual char combat() 和 virtual char challenge():


```
class NPC{
    public:
    int NPChp, NPCattack, NPCdefense;
    NPC(int a, int b, int c){…
    virtual void get_status() = 0;
    virtual char challenge(int &hp, int &attack, int &defense, int &money) = 0;
    virtual char combat() = 0;
};
```

這三個 Virtual Function 分別在 Class Peter 和 Class boss 中得到實作:


```
class Peter : public NPC{ //雜魚，刷等級用
    public:
    Peter():NPC(50, 20, 10){}
    ~Peter(){};
    void get_status(){…
    char combat(){…
    char challenge(int &hp, int &attack, int &defense, int &money){…
};
class boss : public NPC{ //BOSS
    public:
    boss():NPC(200, 50, 25){}
    void get_status(){…
    char combat(){…
    char challenge(int &hp, int &attack, int &defense, int &money){…
};
```

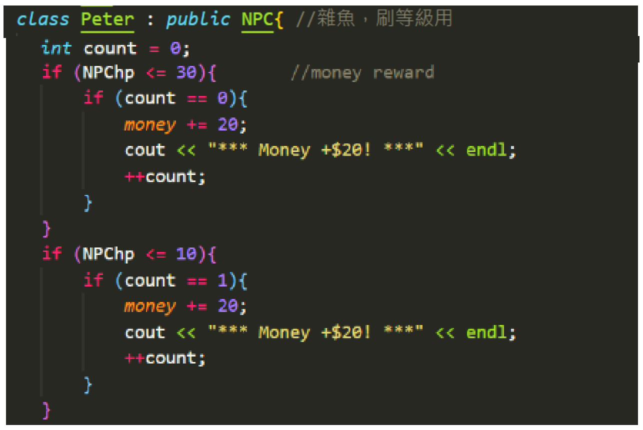
3. Optional Enhancements

3.1. Money System

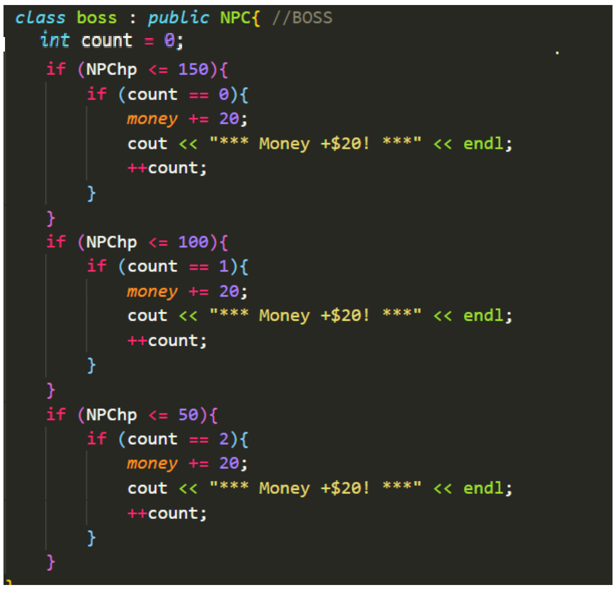
遊戲中有設置金錢系統，玩家在遊戲開始時會預先有 100 元，如果想要賺取金錢，可以透過打小怪獸或大 Boss 來賺，賺錢的機制如下:

a. 小怪獸 -> 每消耗小怪獸 20 滴血即賺取 20 元
b. 大 Boss -> 每消耗大 Boss 50 滴血即賺取 20 元

對應的 code 如下:



```
class Peter : public NPC{ //雜魚，刷等級用
    int count = 0;
    if (NPChp <= 30){        //money reward
        if (count == 0){
            money += 20;
            cout << "*** Money +$20! ***" << endl;
            ++count;
        }
    }
    if (NPChp <= 10){
        if (count == 1){
            money += 20;
            cout << "*** Money +$20! ***" << endl;
            ++count;
        }
    }
```

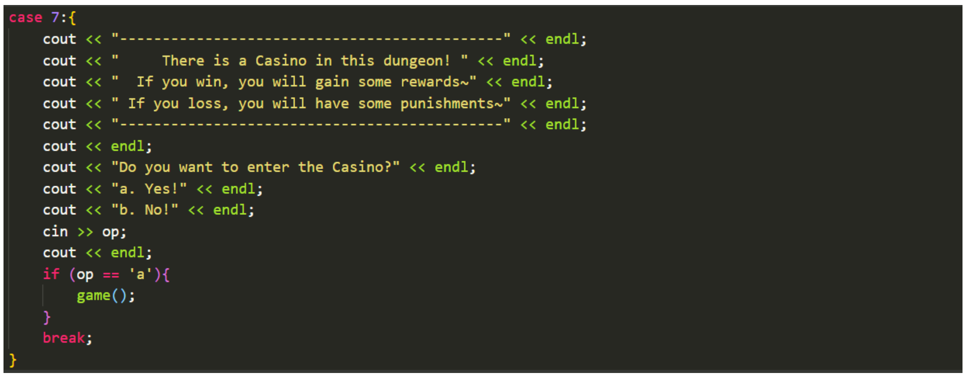


```
class boss : public NPC{ //BOSS
    int count = 0;
    if (NPChp <= 150){
        if (count == 0){
            money += 20;
            cout << "*** Money +$20! ***" << endl;
            ++count;
        }
    }
    if (NPChp <= 100){
        if (count == 1){
            money += 20;
            cout << "*** Money +$20! ***" << endl;
            ++count;
        }
    }
    if (NPChp <= 50){
        if (count == 2){
            money += 20;
            cout << "*** Money +$20! ***" << endl;
            ++count;
        }
    }
```

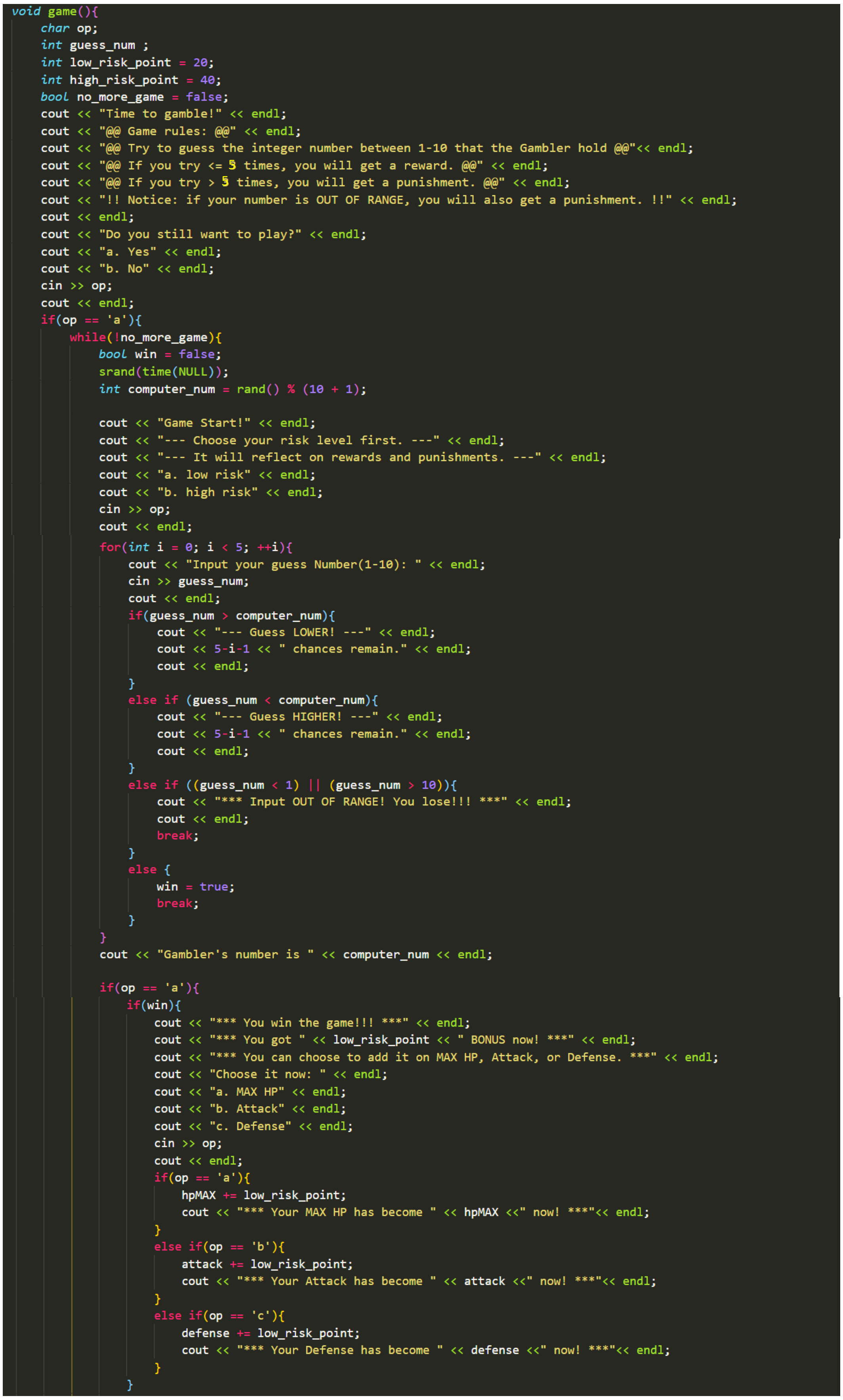
賺了錢後可以去找遊戲中的商店來買裝備(詳見 2.5)，提升玩家的各個攻擊力，防禦力和最大生命。

3.2. Casino

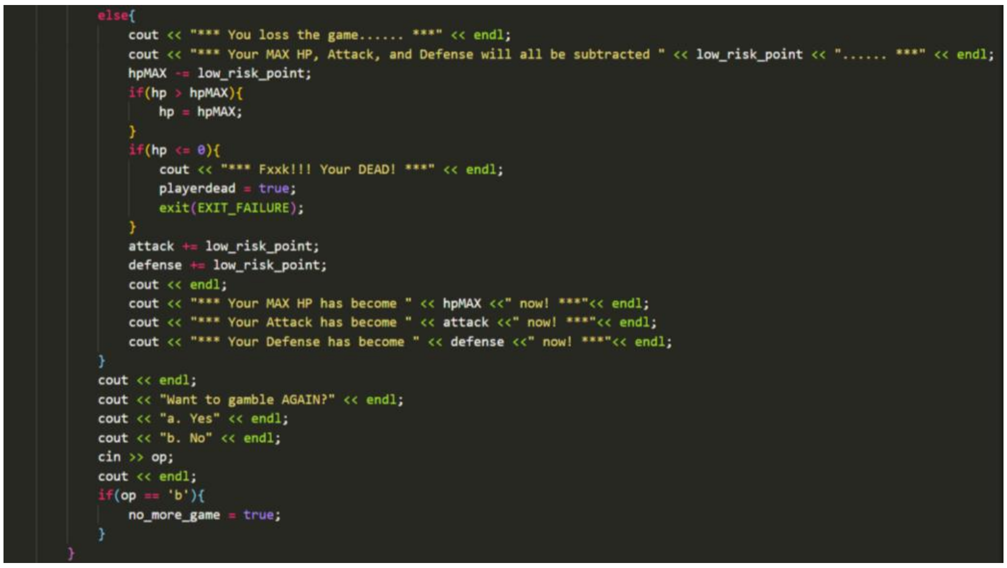
在 7 號 dungeon 中有賭場，贏了有獎勵，輸了有懲罰。玩家可以自由選擇要不要進入賭場，若選擇進入則開始遊戲，不進入則待在該 dungeon 內，不會被迫退到上一個 dungeon 中。開始遊戲後，玩家先選擇他們要參加低風險又或是高風險局，低風險局雖然獎勵較少，但相對的輸了時的懲罰也比較輕微，高風險則是獎勵好但懲罰較重，選擇完後開始遊戲，玩家要在 5 次機會內猜中電腦隨機在 1 到 10 之間生成的數字，5 次內猜中則玩家勝利，可以選擇在攻擊力、防禦力或最大生命加點數，若未猜中，則攻擊力、防禦力和最大生命都會扣點數，低風險局為 20 點，高風險局為 40 點。值得注意的點是，玩家雖然有可能一直贏而獲得很多點數，但也有可能一直輸，甚至把生命值都給輸光了。實作 code 則位於 Class Dungeon 中的 void judge_roomtype()和 void game():



```
case 7:{
    cout << "---------------------------------------------" << endl;
    cout << "     There is a Casino in this dungeon! " << endl;
    cout << "  If you win, you will gain some rewards~" << endl;
    cout << " If you loss, you will have some punishments~" << endl;
    cout << "---------------------------------------------" << endl;
    cout << endl;
    cout << "Do you want to enter the Casino?" << endl;
    cout << "a. Yes!" << endl;
    cout << "b. No!" << endl;
    cin >> op;
    cout << endl;
    if (op == 'a'){
        game();
    }
    break;
}
```



```
void game(){
    char op;
    int guess_num ;
    int low_risk_point = 20;
    int high_risk_point = 40;
    bool no_more_game = false;
    cout << "Time to gamble!" << endl;
    cout << "@@ Game rules: @@" << endl;
    cout << "@@ Try to guess the integer number between 1-10 that the Gambler hold @@"<< endl;
    cout << "@@ If you try <= 5 times, you will get a reward. @@" << endl;
    cout << "@@ If you try > 5 times, you will get a punishment. @@" << endl;
    cout << "!! Notice: if your number is OUT OF RANGE, you will also get a punishment. !!" << endl;
    cout << endl;
    cout << "Do you still want to play?" << endl;
    cout << "a. Yes" << endl;
    cout << "b. No" << endl;
    cin >> op;
    cout << endl;
    if(op == 'a'){
        while(!no_more_game){
            bool win = false;
            srand(time(NULL));
            int computer_num = rand() % (10 + 1);

            cout << "Game Start!" << endl;
            cout << "--- Choose your risk level first. ---" << endl;
            cout << "--- It will reflect on rewards and punishments. ---" << endl;
            cout << "a. low risk" << endl;
            cout << "b. high risk" << endl;
            cin >> op;
            cout << endl;
            for(int i = 0; i < 5; ++i){
                cout << "Input your guess Number(1-10): " << endl;
                cin >> guess_num;
                cout << endl;
                if(guess_num > computer_num){
                    cout << "--- Guess LOWER! ---" << endl;
                    cout << 5-i-1 << " chances remain." << endl;
                    cout << endl;
                }
                else if (guess_num < computer_num){
                    cout << "--- Guess HIGHER! ---" << endl;
                    cout << 5-i-1 << " chances remain." << endl;
                    cout << endl;
                }
                else if ((guess_num < 1) || (guess_num > 10)){
                    cout << "*** Input OUT OF RANGE! You lose!!! ***" << endl;
                    cout << endl;
                    break;
                }
                else {
                    win = true;
                    break;
                }
            }
            cout << "Gambler's number is " << computer_num << endl;

            if(op == 'a'){
                if(win){
                    cout << "*** You win the game!!! ***" << endl;
                    cout << "*** You got " << low_risk_point << " BONUS now! ***" << endl;
                    cout << "*** You can choose to add it on MAX HP, Attack, or Defense. ***" << endl;
                    cout << "Choose it now: " << endl;
                    cout << "a. MAX HP" << endl;
                    cout << "b. Attack" << endl;
                    cout << "c. Defense" << endl;
                    cin >> op;
                    cout << endl;
                    if(op == 'a'){
                        hpMAX += low_risk_point;
                        cout << "*** Your MAX HP has become " << hpMAX <<" now! ***"<< endl;
                    }
                    else if(op == 'b'){
                        attack += low_risk_point;
                        cout << "*** Your Attack has become " << attack <<" now! ***"<< endl;
                    }
                    else if(op == 'c'){
                        defense += low_risk_point;
                        cout << "*** Your Defense has become " << defense <<" now! ***"<< endl;
                    }
                }
```

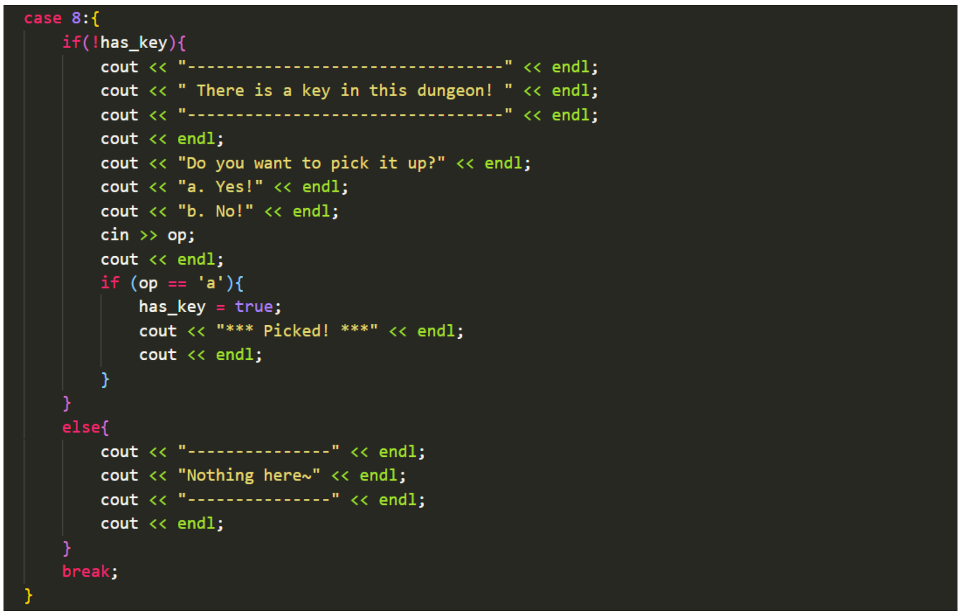


```
        else{
            cout << "*** You loss the game...... ***" << endl;
            cout << "*** Your MAX HP, Attack, and Defense will all be subtracted " << low_risk_point << "...... ***" << endl;
            hpMAX -= low_risk_point;
            if(hp > hpMAX){
                hp = hpMAX;
            }
            if(hp <= 0){
                cout << "*** Fxxk!!! Your DEAD! ***" << endl;
                playerdead = true;
                exit(EXIT_FAILURE);
            }
            attack += low_risk_point;
            defense += low_risk_point;
            cout << endl;
            cout << "*** Your MAX HP has become " << hpMAX <<" now! ***"<< endl;
            cout << "*** Your Attack has become " << attack <<" now! ***"<< endl;
            cout << "*** Your Defense has become " << defense <<" now! ***"<< endl;
        }
        cout << endl;
        cout << "Want to gamble AGAIN?" << endl;
        cout << "a. Yes" << endl;
        cout << "b. No" << endl;
        cin >> op;
        cout << endl;
        if(op == 'b'){
            no_more_game = true;
        }
    }
```

因為太占版面而且邏輯類似，所以上方沒有貼上選擇高風險時對應的 code。

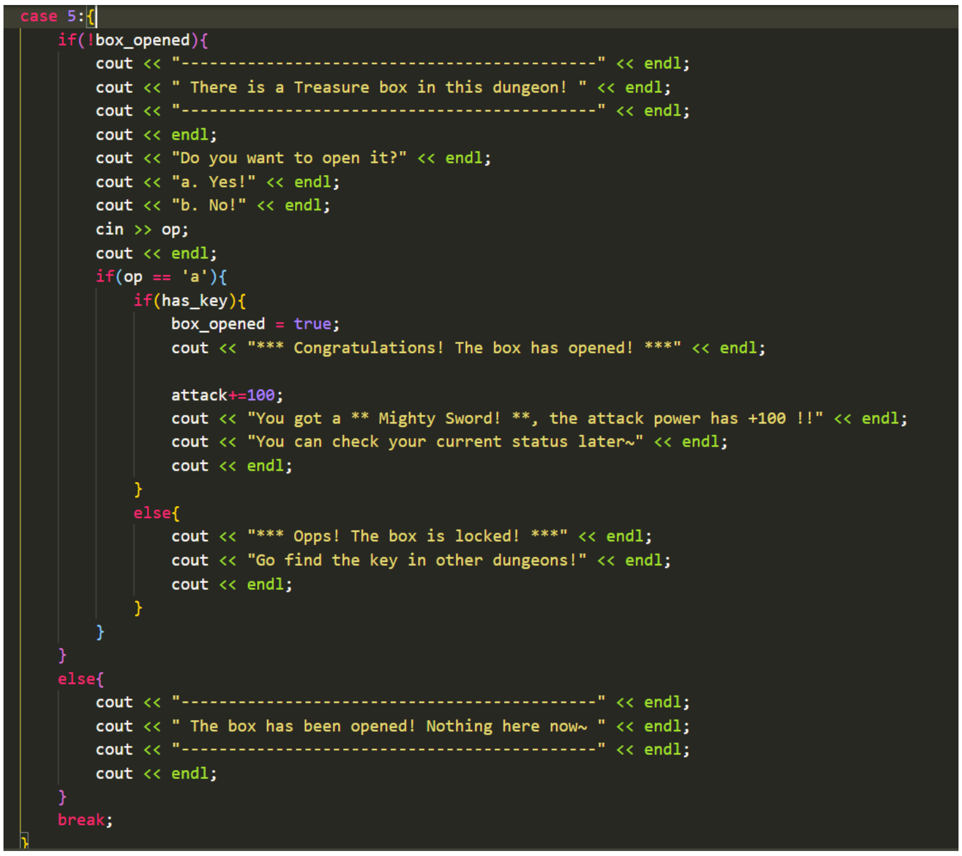
3.3. Treasure Box and Key

在遊戲中，8 號 dungeon 有鑰匙，當玩家進入 8 號 dungeon 時可以自由選擇要不要撿起鑰匙，而鑰匙只有一把，當玩家撿起後，此 dungeon 就沒有剩下任何東西了，而相對的程式碼是在 Class dungeon 的 void judge_roomtype()裡，其中 case 判斷的是使用者在移動後的 dungeon 號碼:



```
case 8:{
    if(!has_key){
        cout << "--------------------------------" << endl;
        cout << " There is a key in this dungeon! " << endl;
        cout << "--------------------------------" << endl;
        cout << endl;
        cout << "Do you want to pick it up?" << endl;
        cout << "a. Yes!" << endl;
        cout << "b. No!" << endl;
        cin >> op;
        cout << endl;
        if (op == 'a'){
            has_key = true;
            cout << "*** Picked! ***" << endl;
            cout << endl;
        }
    }
    else{
        cout << "--------------" << endl;
        cout << "Nothing here~" << endl;
        cout << "--------------" << endl;
        cout << endl;
    }
    break;
}
```

在 8 號 dungeon 中撿到的鑰匙可以用來打開寶箱，寶箱內有超強寶劍可以讓玩家的攻擊力增加 100 點，而寶箱只有一個，沒有鑰匙打不開寶箱，開完後寶箱就消失，此 dungeon 中就沒有東西了，這部分也是由 Class Dungeon 的 void judge_roomtype()來處理:



```cpp
case 5:{
    if(!box_opened){
        cout << "-------------------------------------------" << endl;
        cout << " There is a Treasure box in this dungeon! " << endl;
        cout << "-------------------------------------------" << endl;
        cout << endl;
        cout << "Do you want to open it?" << endl;
        cout << "a. Yes!" << endl;
        cout << "b. No!" << endl;
        cin >> op;
        cout << endl;
        if(op == 'a'){
            if(has_key){
                box_opened = true;
                cout << "*** Congratulations! The box has opened! ***" << endl;

                attack+=100;
                cout << "You got a ** Mighty Sword! **, the attack power has +100 !!" << endl;
                cout << "You can check your current status later~" << endl;
                cout << endl;
            }
            else{
                cout << "*** Opps! The box is locked! ***" << endl;
                cout << "Go find the key in other dungeons!" << endl;
                cout << endl;
            }
        }
    }
    else{
        cout << "-------------------------------------------" << endl;
        cout << " The box has been opened! Nothing here now~ " << endl;
        cout << "-------------------------------------------" << endl;
        cout << endl;
    }
    break;
}
```

4. Discussion and Conclusion

本次作業我決定不用模板而是自己想練練自己寫程式的東西，順便驗證一下自己有沒有融會貫通上課講的東西，最後辛苦的把它完成了。如果說要加強的地方應該是程式的簡潔程度吧，感覺很多部分還可以再簡化，但我現在是很土法煉鋼的把他們全部打出來，邏輯對了就 ok 的狀態，未來要改善一下這個。另外下次也可以嘗試做圖形化的介面，例如顯示地圖、角色在走路之類的，來提升使用者體驗。總而言之，這次作業就這樣啦，掰掰。