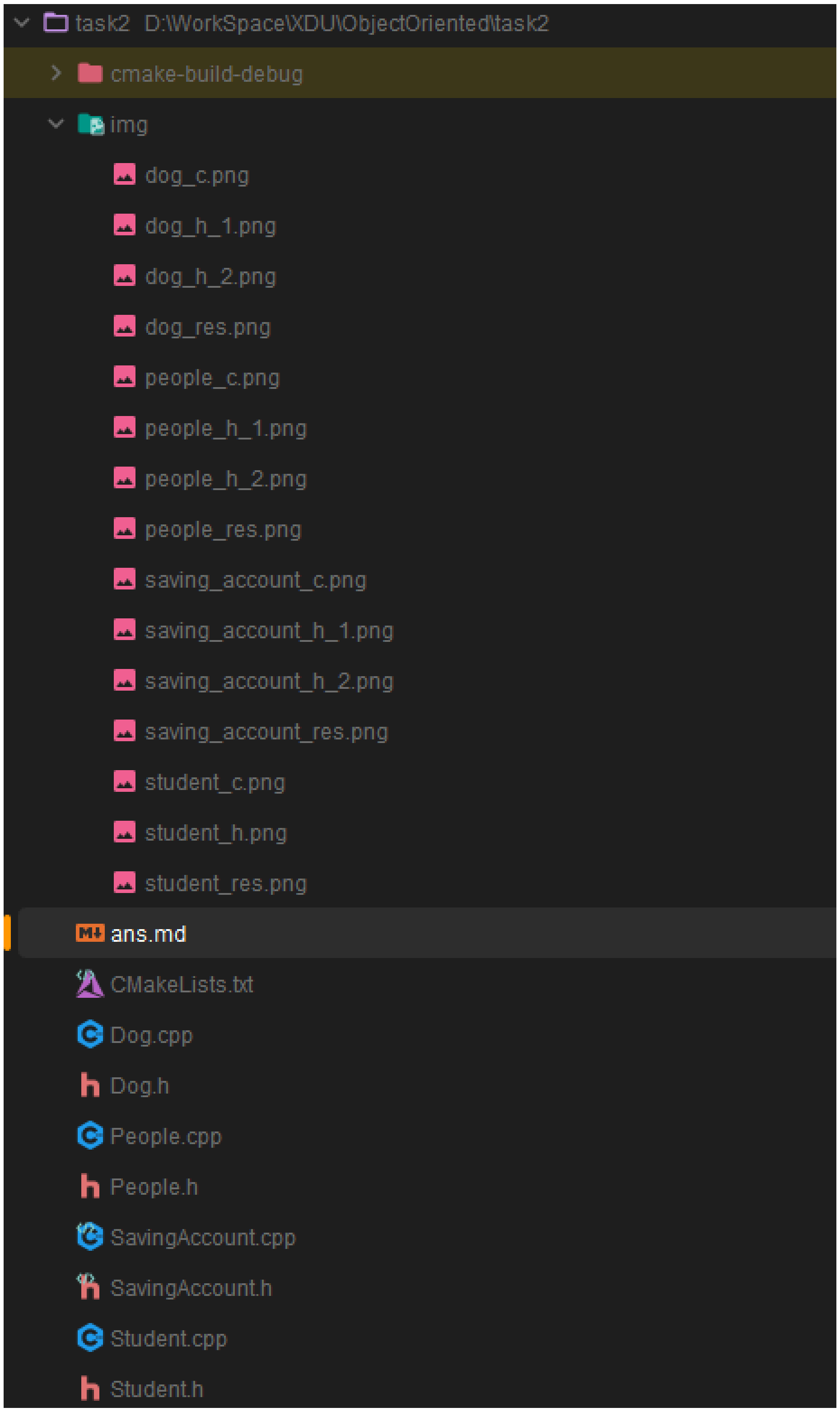
OOP作业2

文件目录

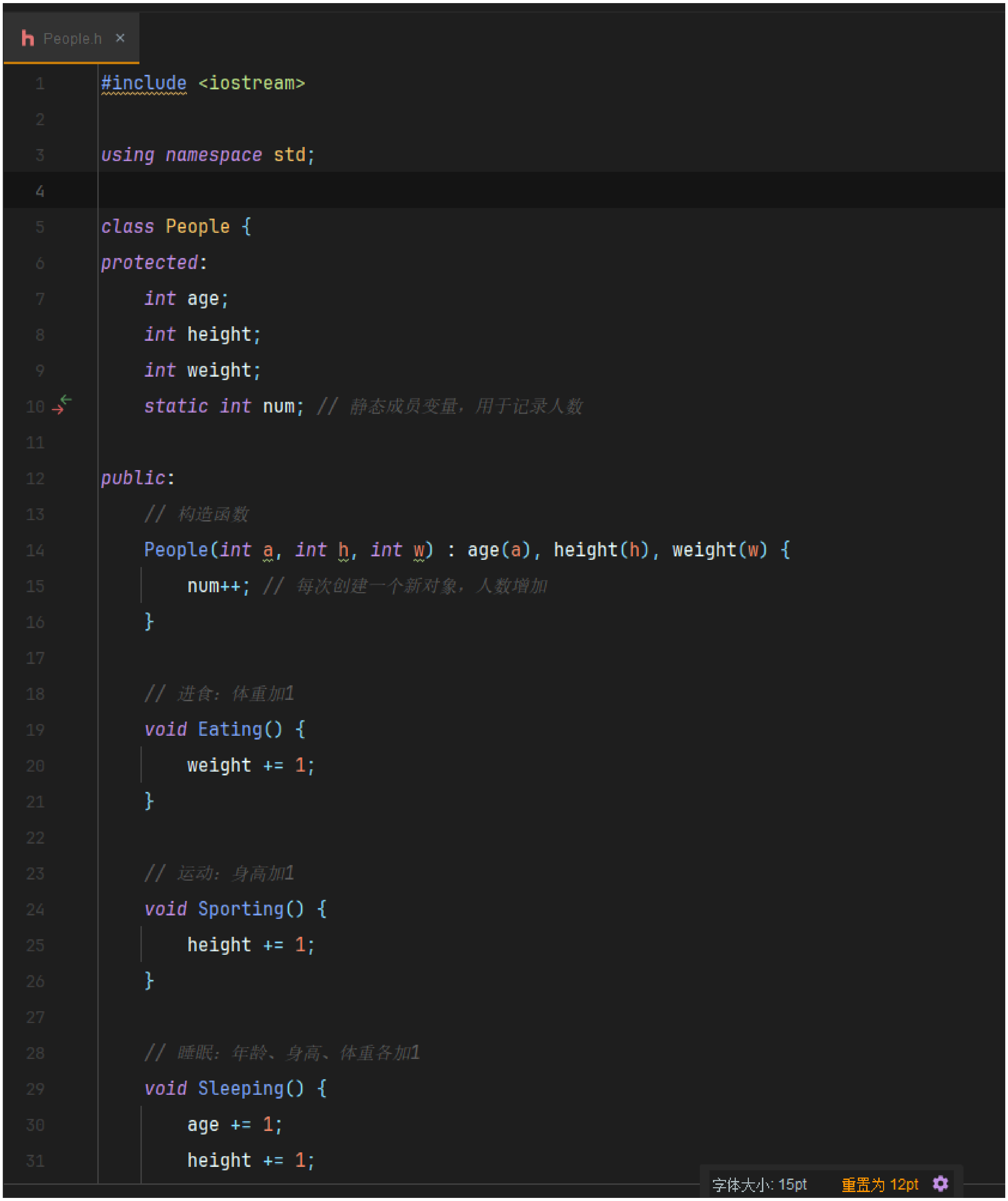


```
task2  D:\WorkSpace\XDU\ObjectOriented\task2
  cmake-build-debug
  img
    dog_c.png
    dog_h_1.png
    dog_h_2.png
    dog_res.png
    people_c.png
    people_h_1.png
    people_h_2.png
    people_res.png
    saving_account_c.png
    saving_account_h_1.png
    saving_account_h_2.png
    saving_account_res.png
    student_c.png
    student_h.png
    student_res.png
  ans.md
  CMakeLists.txt
  Dog.cpp
  Dog.h
  People.cpp
  People.h
  SavingAccount.cpp
  SavingAccount.h
  Student.cpp
  Student.h
```

具体实现

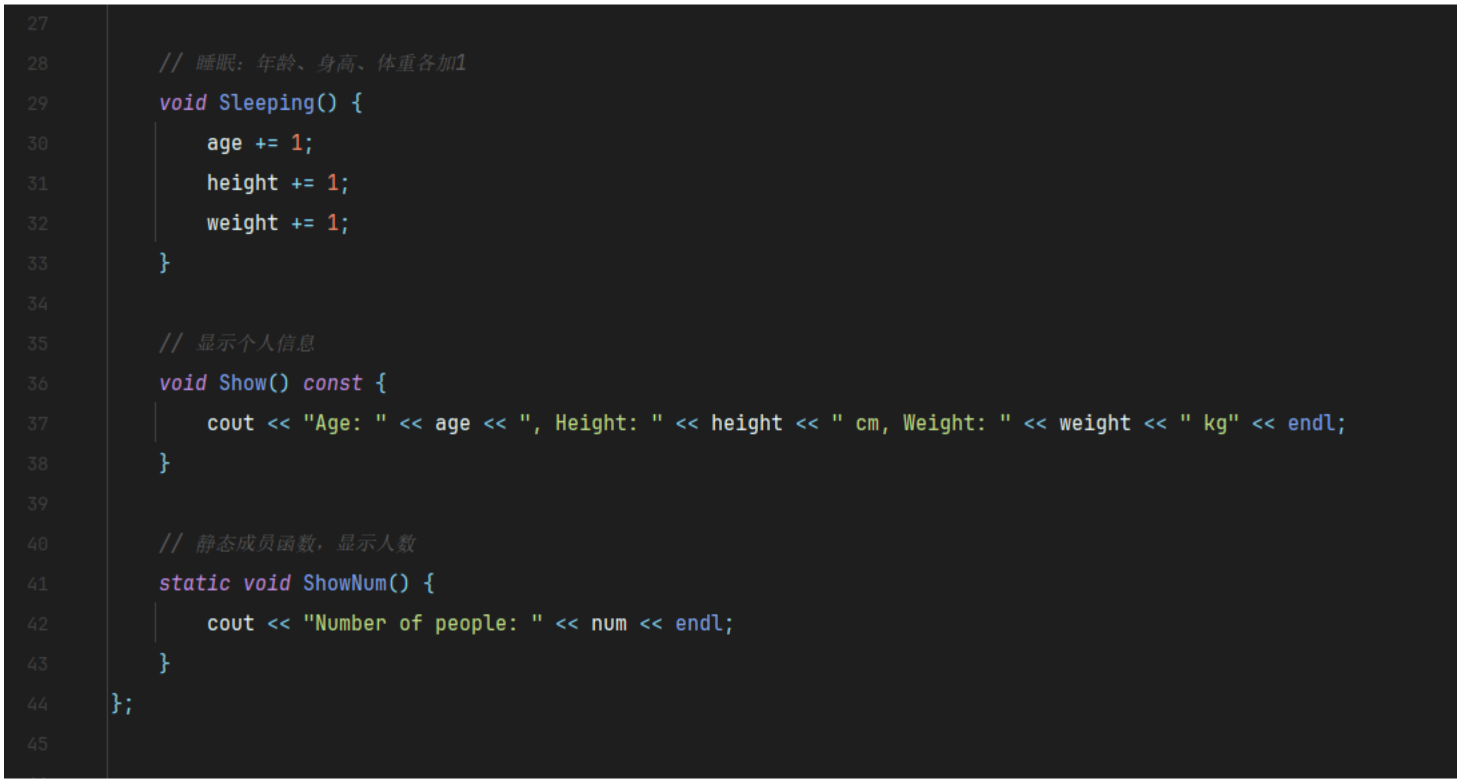
1. People类实现

头文件(包含具体实现)



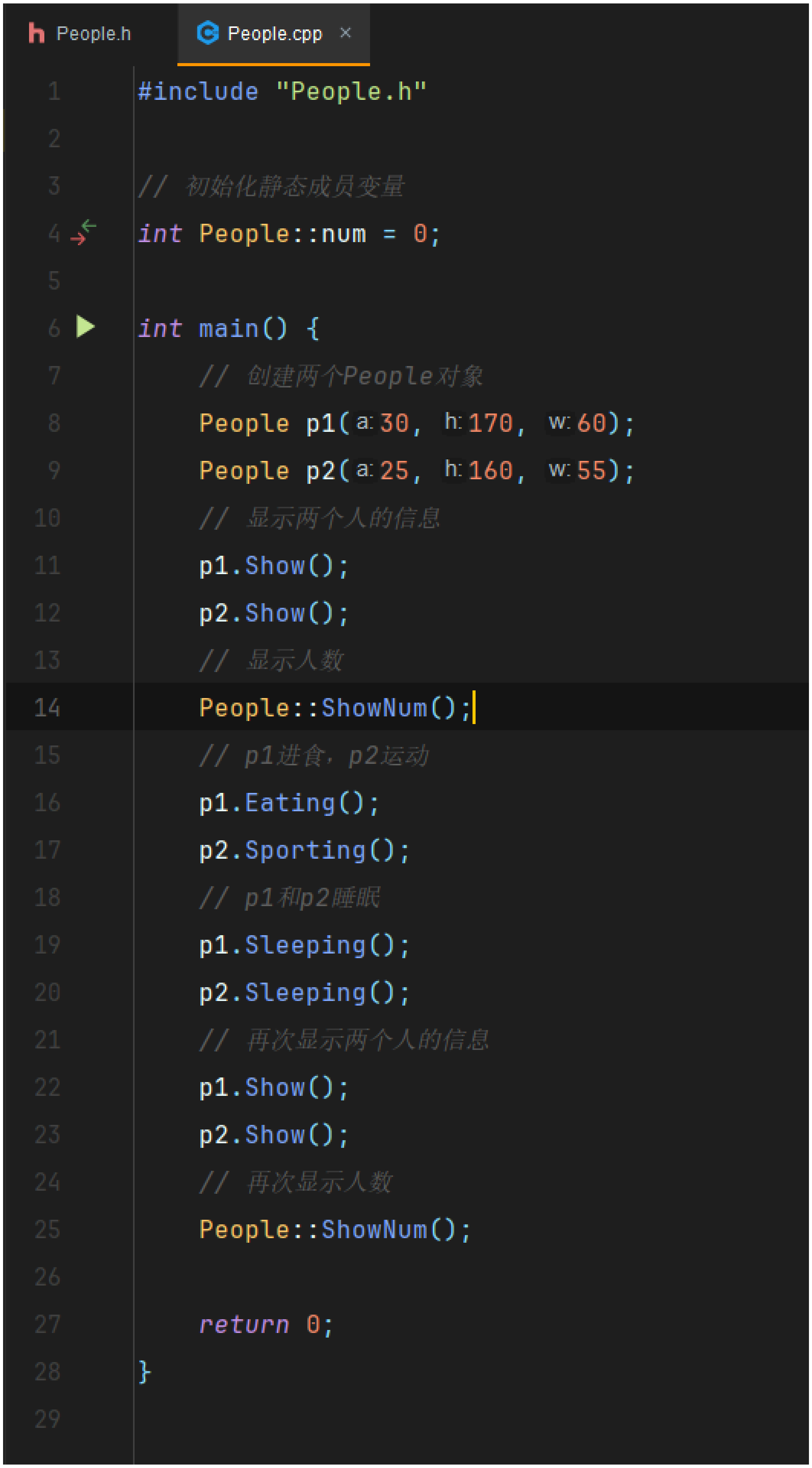
People.h ×

```cpp
#include <iostream>

using namespace std;

class People {
protected:
    int age;
    int height;
    int weight;
    static int num; // 静态成员变量，用于记录人数

public:
    // 构造函数
    People(int a, int h, int w) : age(a), height(h), weight(w) {
        num++; // 每次创建一个新对象，人数增加
    }

    // 进食：体重加1
    void Eating() {
        weight += 1;
    }

    // 运动：身高加1
    void Sporting() {
        height += 1;
    }

    // 睡眠：年龄、身高、体重各加1
    void Sleeping() {
        age += 1;
        height += 1;
```



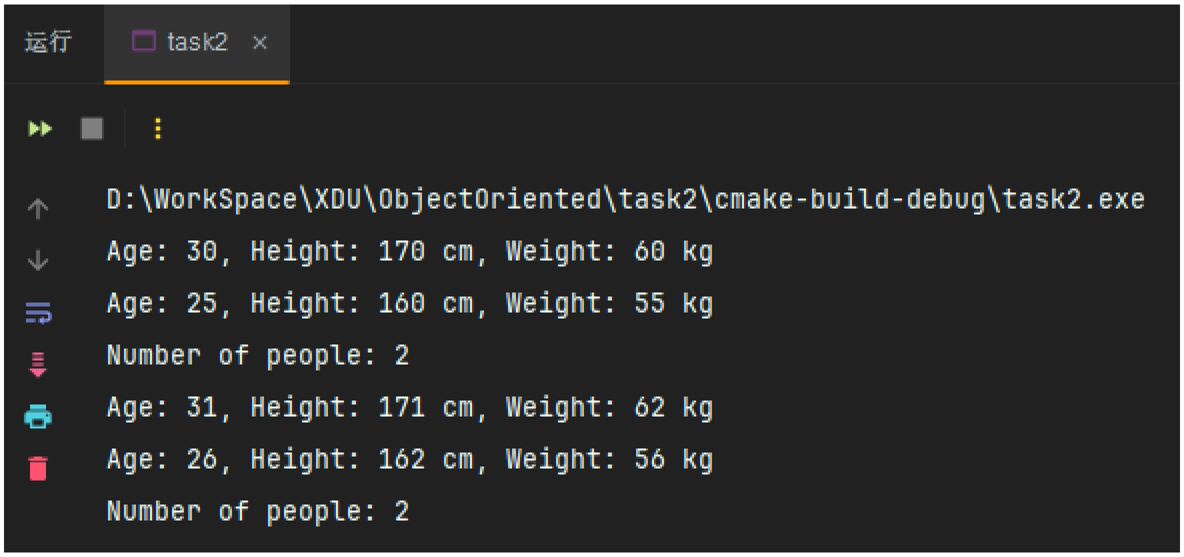
```cpp

    // 睡眠：年龄、身高、体重各加1
    void Sleeping() {
        age += 1;
        height += 1;
        weight += 1;
    }

    // 显示个人信息
    void Show() const {
        cout << "Age: " << age << ", Height: " << height << " cm, Weight: " << weight << " kg" << endl;
    }

    // 静态成员函数，显示人数
    static void ShowNum() {
        cout << "Number of people: " << num << endl;
    }
};

```

编译源文件(包含主函数测试)



```cpp
#include "People.h"

// 初始化静态成员变量
int People::num = 0;

int main() {
    // 创建两个People对象
    People p1( a: 30,  h: 170,  w: 60);
    People p2( a: 25,  h: 160,  w: 55);
    // 显示两个人的信息
    p1.Show();
    p2.Show();
    // 显示人数
    People::ShowNum();
    // p1进食，p2运动
    p1.Eating();
    p2.Sporting();
    // p1和p2睡眠
    p1.Sleeping();
    p2.Sleeping();
    // 再次显示两个人的信息
    p1.Show();
    p2.Show();
    // 再次显示人数
    People::ShowNum();

    return 0;
}
```

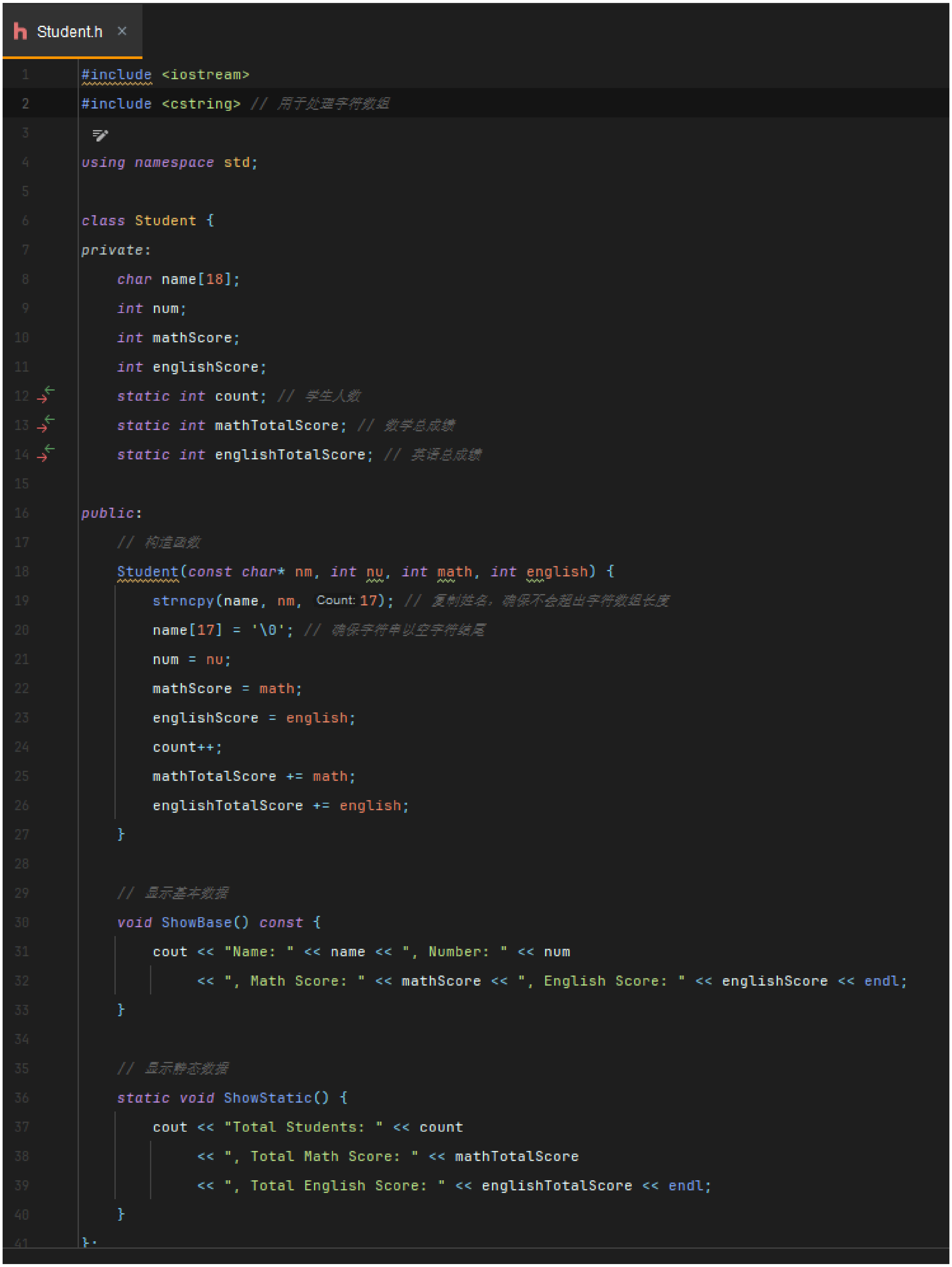
运行结果



```
D:\WorkSpace\XDU\ObjectOriented\task2\cmake-build-debug\task2.exe
Age: 30, Height: 170 cm, Weight: 60 kg
Age: 25, Height: 160 cm, Weight: 55 kg
Number of people: 2
Age: 31, Height: 171 cm, Weight: 62 kg
Age: 26, Height: 162 cm, Weight: 56 kg
Number of people: 2
```

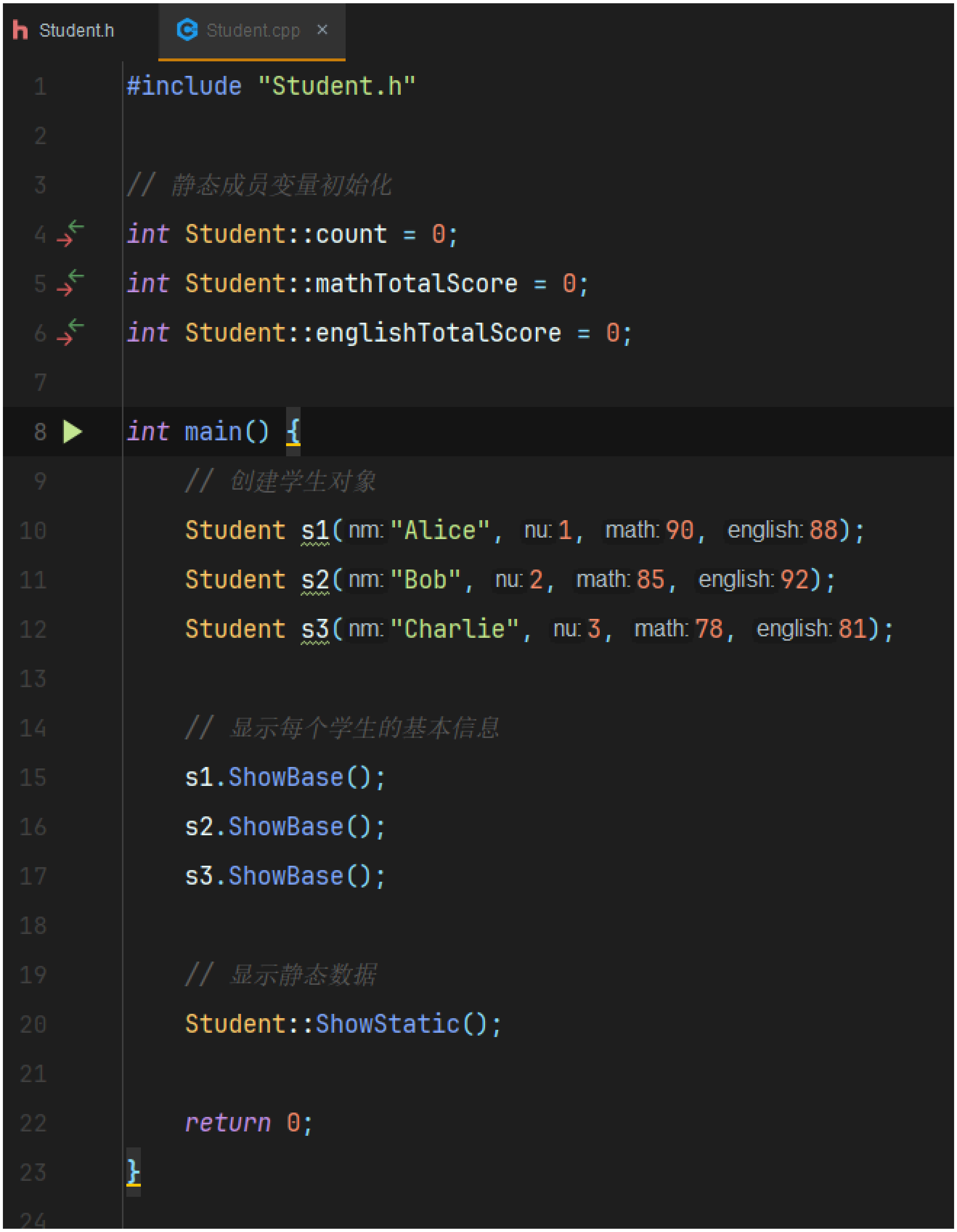
2. Student类实现

头文件(包含具体实现)


Student.h

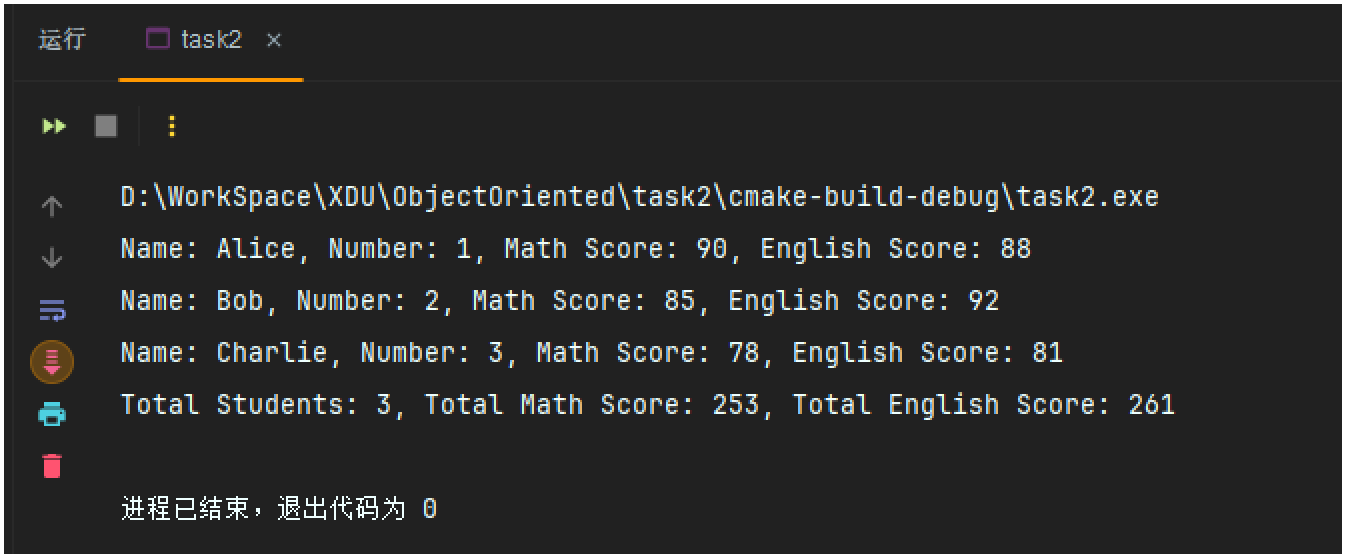
```cpp
#include <iostream>
#include <cstring> // 用于处理字符数组

using namespace std;

class Student {
private:
    char name[18];
    int num;
    int mathScore;
    int englishScore;
    static int count; // 学生人数
    static int mathTotalScore; // 数学总成绩
    static int englishTotalScore; // 英语总成绩

public:
    // 构造函数
    Student(const char* nm, int nu, int math, int english) {
        strncpy(name, nm,  Count: 17); // 复制姓名，确保不会超出字符数组长度
        name[17] = '\0'; // 确保字符串以空字符结尾
        num = nu;
        mathScore = math;
        englishScore = english;
        count++;
        mathTotalScore += math;
        englishTotalScore += english;
    }

    // 显示基本数据
    void ShowBase() const {
        cout << "Name: " << name << ", Number: " << num
             << ", Math Score: " << mathScore << ", English Score: " << englishScore << endl;
    }

    // 显示静态数据
    static void ShowStatic() {
        cout << "Total Students: " << count
             << ", Total Math Score: " << mathTotalScore
             << ", Total English Score: " << englishTotalScore << endl;
    }
};
```

编译源文件(包含主函数测试)



```cpp
#include "Student.h"

// 静态成员变量初始化
int Student::count = 0;
int Student::mathTotalScore = 0;
int Student::englishTotalScore = 0;

int main() {
    // 创建学生对象
    Student s1( nm: "Alice", nu: 1, math: 90, english: 88);
    Student s2( nm: "Bob", nu: 2, math: 85, english: 92);
    Student s3( nm: "Charlie", nu: 3, math: 78, english: 81);

    // 显示每个学生的基本信息
    s1.ShowBase();
    s2.ShowBase();
    s3.ShowBase();

    // 显示静态数据
    Student::ShowStatic();

    return 0;
}
```

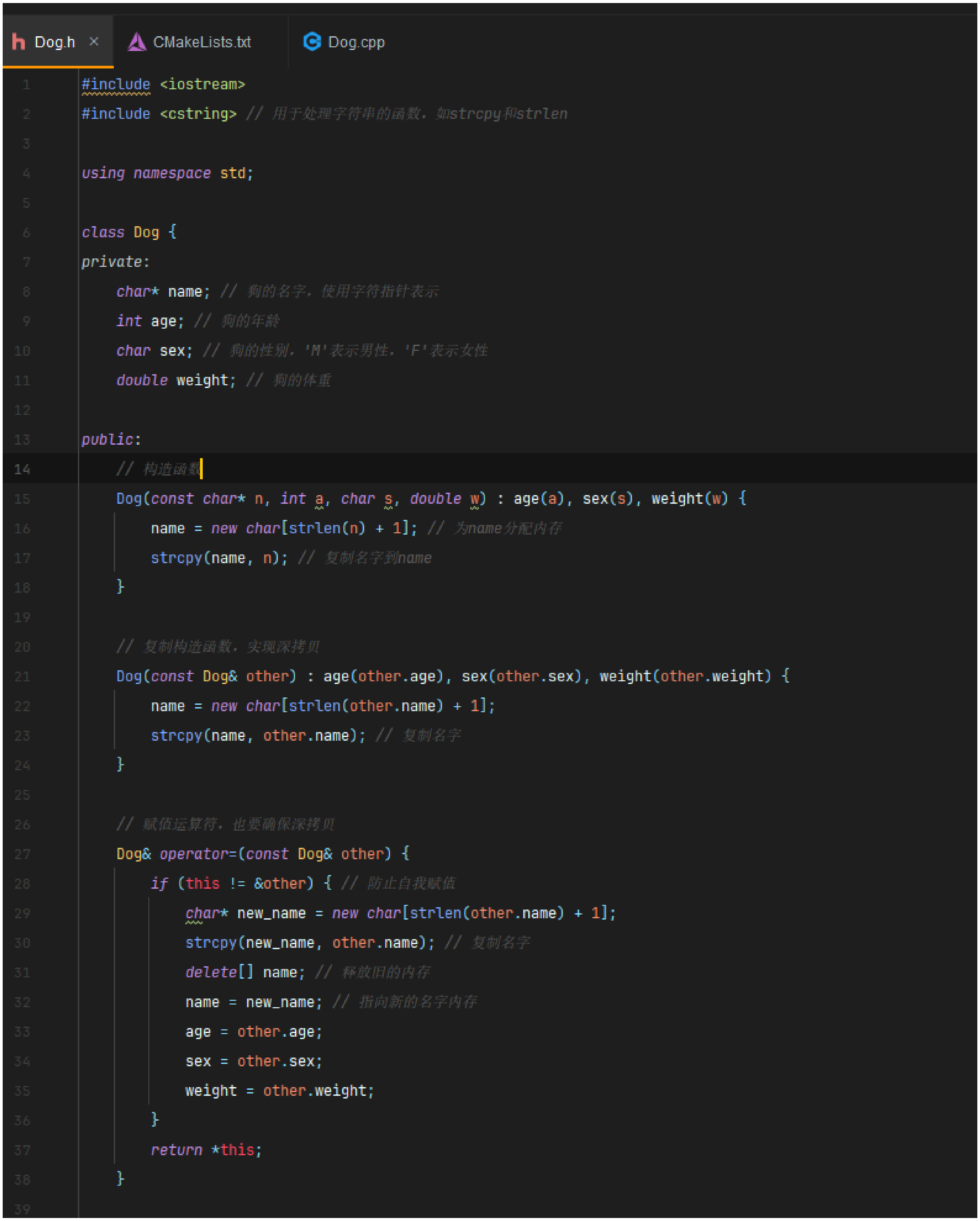
运行结果



```
D:\WorkSpace\XDU\ObjectOriented\task2\cmake-build-debug\task2.exe
Name: Alice, Number: 1, Math Score: 90, English Score: 88
Name: Bob, Number: 2, Math Score: 85, English Score: 92
Name: Charlie, Number: 3, Math Score: 78, English Score: 81
Total Students: 3, Total Math Score: 253, Total English Score: 261

进程已结束，退出代码为 0
```

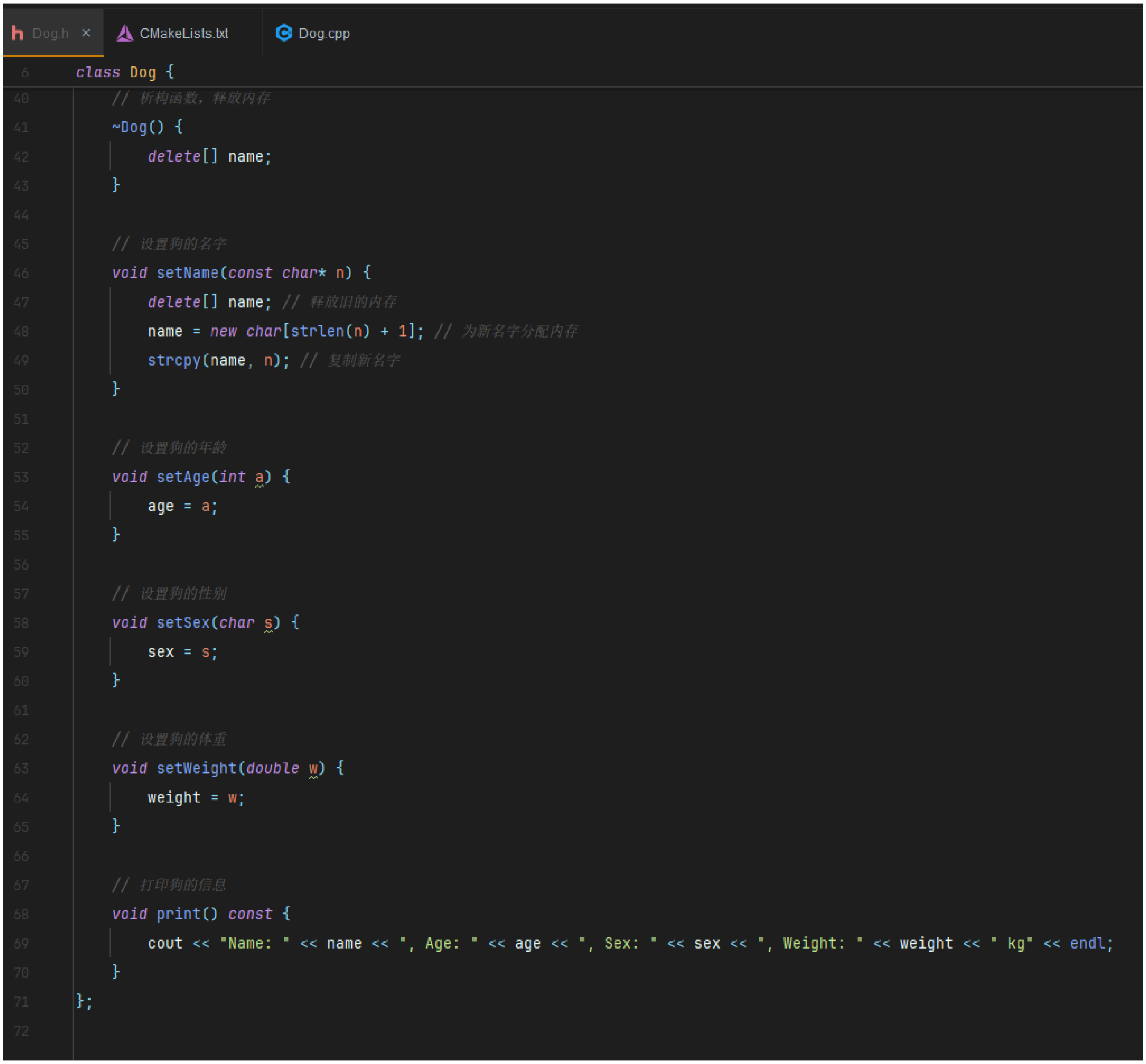
3. Dog类实现

头文件(包含具体实现)



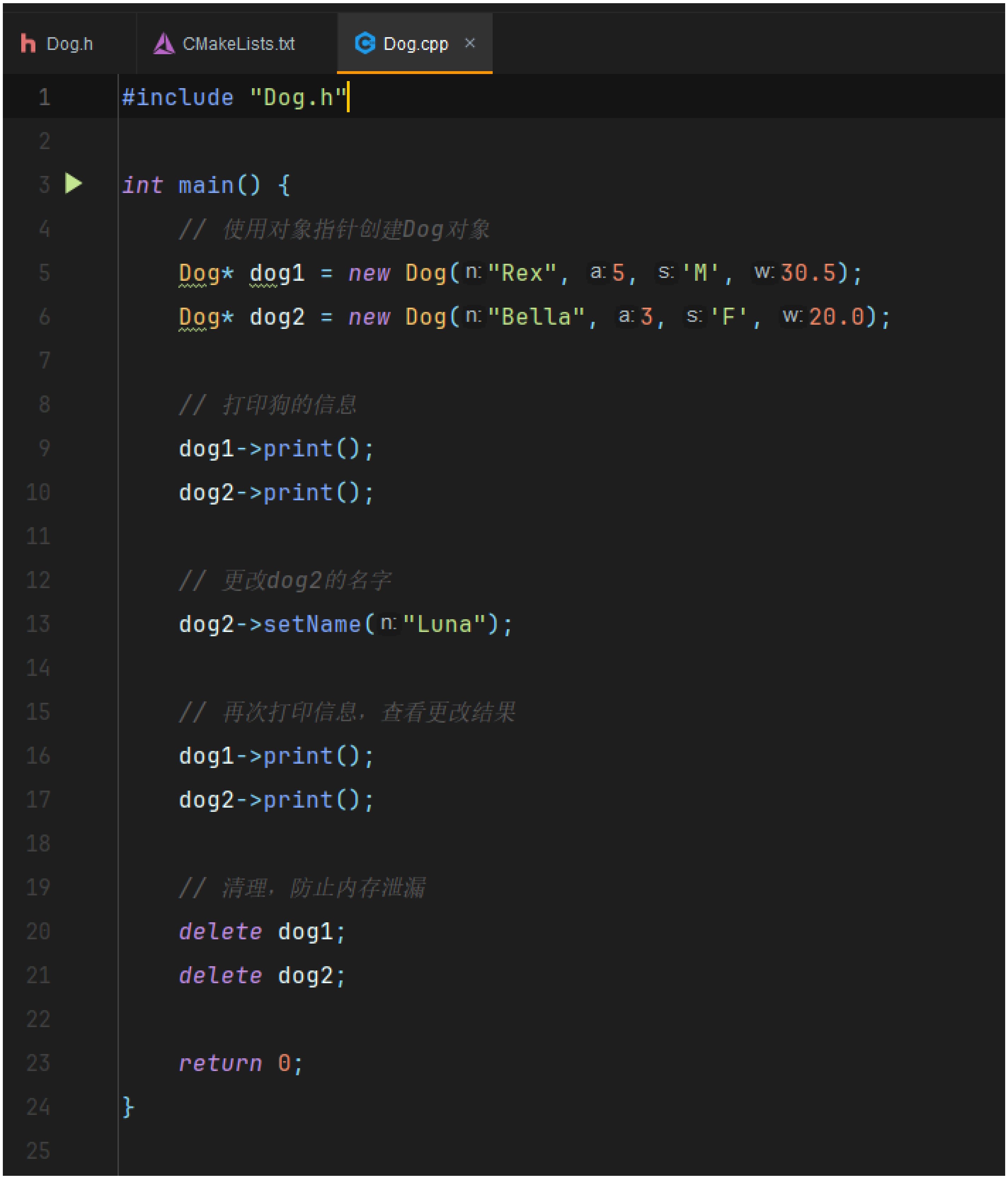
Dog.h × CMakeLists.txt Dog.cpp

```cpp
#include <iostream>
#include <cstring> // 用于处理字符串的函数，如strcpy和strlen

using namespace std;

class Dog {
private:
    char* name; // 狗的名字，使用字符指针表示
    int age; // 狗的年龄
    char sex; // 狗的性别，'M'表示男性，'F'表示女性
    double weight; // 狗的体重

public:
    // 构造函数
    Dog(const char* n, int a, char s, double w) : age(a), sex(s), weight(w) {
        name = new char[strlen(n) + 1]; // 为name分配内存
        strcpy(name, n); // 复制名字到name
    }

    // 复制构造函数，实现深拷贝
    Dog(const Dog& other) : age(other.age), sex(other.sex), weight(other.weight) {
        name = new char[strlen(other.name) + 1];
        strcpy(name, other.name); // 复制名字
    }

    // 赋值运算符，也要确保深拷贝
    Dog& operator=(const Dog& other) {
        if (this != &other) { // 防止自我赋值
            char* new_name = new char[strlen(other.name) + 1];
            strcpy(new_name, other.name); // 复制名字
            delete[] name; // 释放旧的内存
            name = new_name; // 指向新的名字内存
            age = other.age;
            sex = other.sex;
            weight = other.weight;
        }
        return *this;
    }

```


Dog.h × CMakeLists.txt Dog.cpp

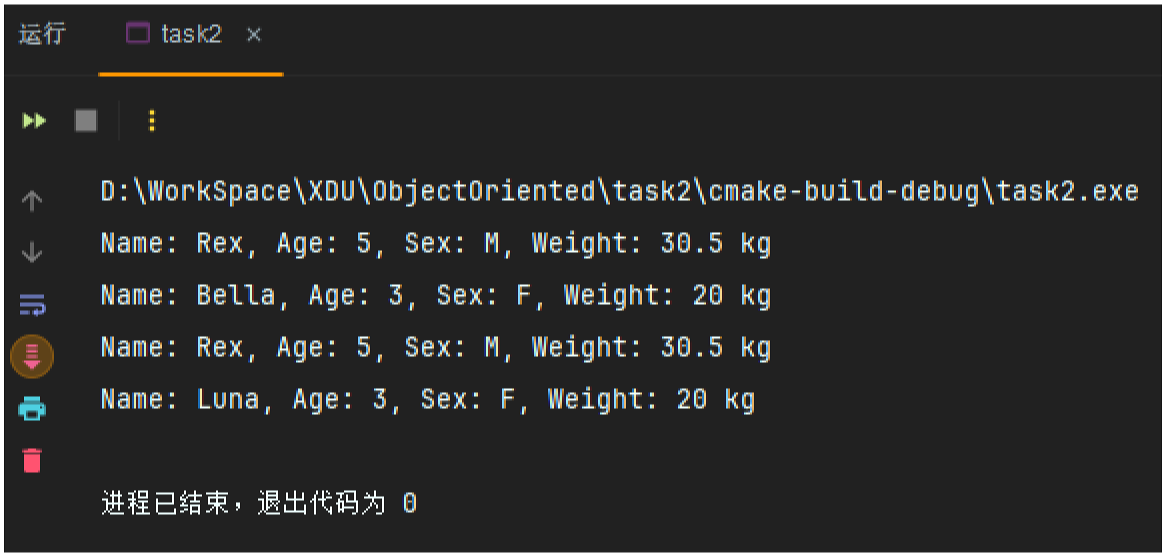
```cpp
class Dog {
    // 析构函数，释放内存
    ~Dog() {
        delete[] name;
    }

    // 设置狗的名字
    void setName(const char* n) {
        delete[] name; // 释放旧的内存
        name = new char[strlen(n) + 1]; // 为新名字分配内存
        strcpy(name, n); // 复制新名字
    }

    // 设置狗的年龄
    void setAge(int a) {
        age = a;
    }

    // 设置狗的性别
    void setSex(char s) {
        sex = s;
    }

    // 设置狗的体重
    void setWeight(double w) {
        weight = w;
    }

    // 打印狗的信息
    void print() const {
        cout << "Name: " << name << ", Age: " << age << ", Sex: " << sex << ", Weight: " << weight << " kg" << endl;
    }
};

```

编译源文件(包含主函数测试)


Dog.h CMakeLists.txt Dog.cpp

```cpp
#include "Dog.h"

int main() {
    // 使用对象指针创建Dog对象
    Dog* dog1 = new Dog(n: "Rex", a: 5, s: 'M', w: 30.5);
    Dog* dog2 = new Dog(n: "Bella", a: 3, s: 'F', w: 20.0);

    // 打印狗的信息
    dog1->print();
    dog2->print();

    // 更改dog2的名字
    dog2->setName(n: "Luna");

    // 再次打印信息，查看更改结果
    dog1->print();
    dog2->print();

    // 清理，防止内存泄漏
    delete dog1;
    delete dog2;

    return 0;
}

```

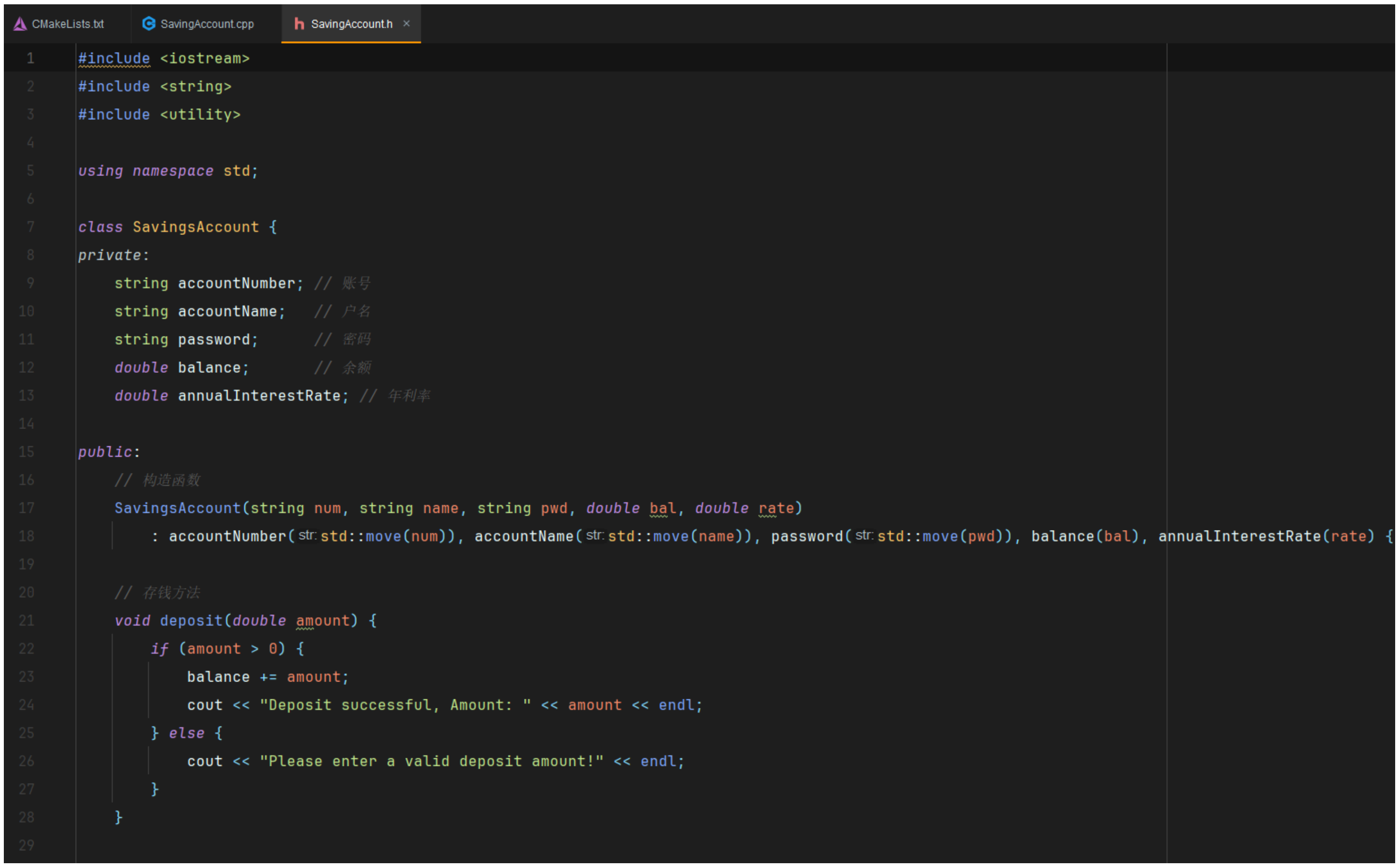
运行结果



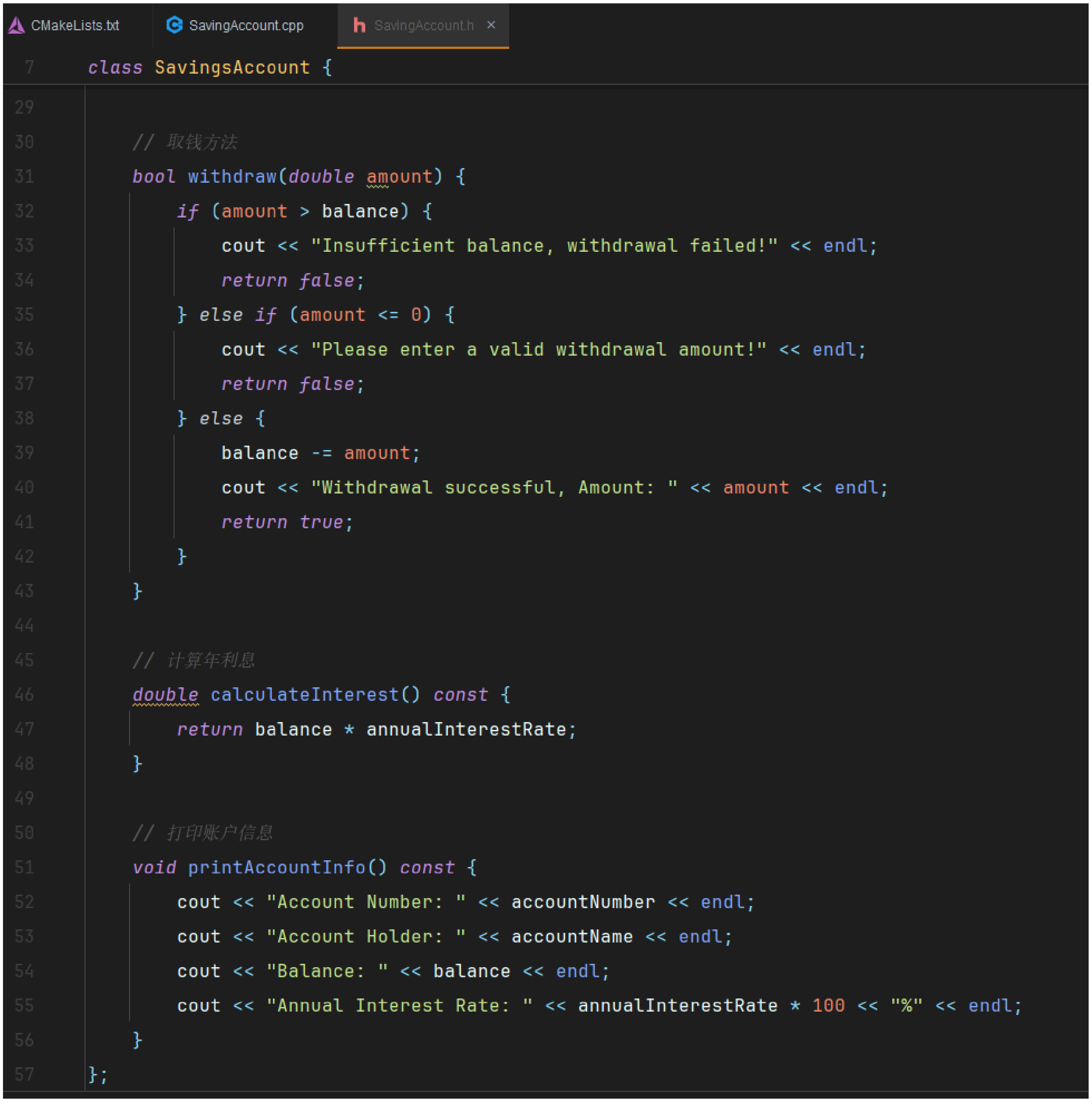
```
D:\WorkSpace\XDU\ObjectOriented\task2\cmake-build-debug\task2.exe
Name: Rex, Age: 5, Sex: M, Weight: 30.5 kg
Name: Bella, Age: 3, Sex: F, Weight: 20 kg
Name: Rex, Age: 5, Sex: M, Weight: 30.5 kg
Name: Luna, Age: 3, Sex: F, Weight: 20 kg

进程已结束，退出代码为 0
```

4. SavingAccount类实现

头文件(包含具体实现)

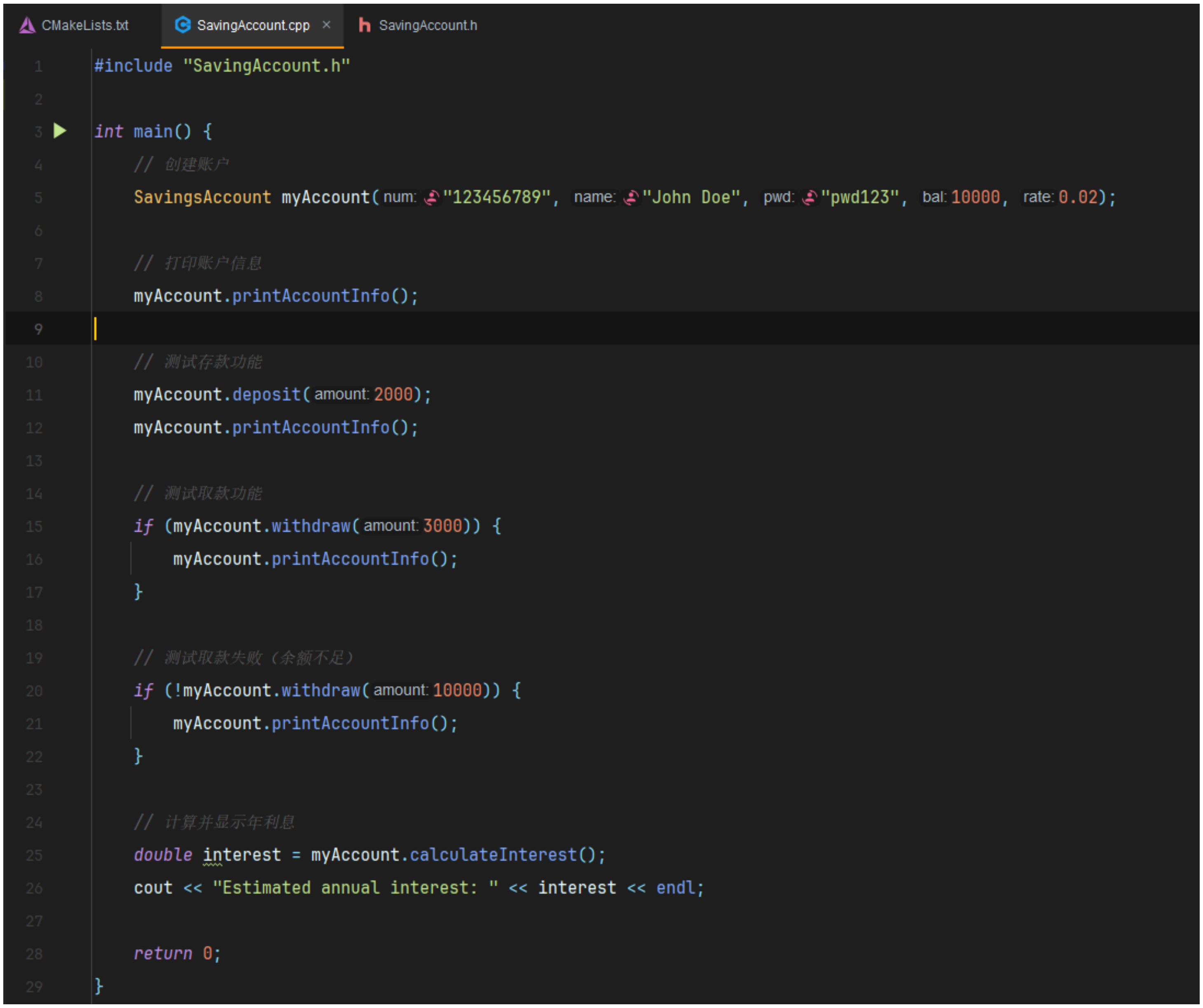


```cpp
#include <iostream>
#include <string>
#include <utility>

using namespace std;

class SavingsAccount {
private:
    string accountNumber; // 账号
    string accountName;   // 户名
    string password;      // 密码
    double balance;       // 余额
    double annualInterestRate; // 年利率

public:
    // 构造函数
    SavingsAccount(string num, string name, string pwd, double bal, double rate)
        : accountNumber(std::move(num)), accountName(std::move(name)), password(std::move(pwd)), balance(bal), annualInterestRate(rate) {

    // 存钱方法
    void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            cout << "Deposit successful, Amount: " << amount << endl;
        } else {
            cout << "Please enter a valid deposit amount!" << endl;
        }
    }

```


CMakeLists.txt SavingAccount.cpp SavingAccount.h

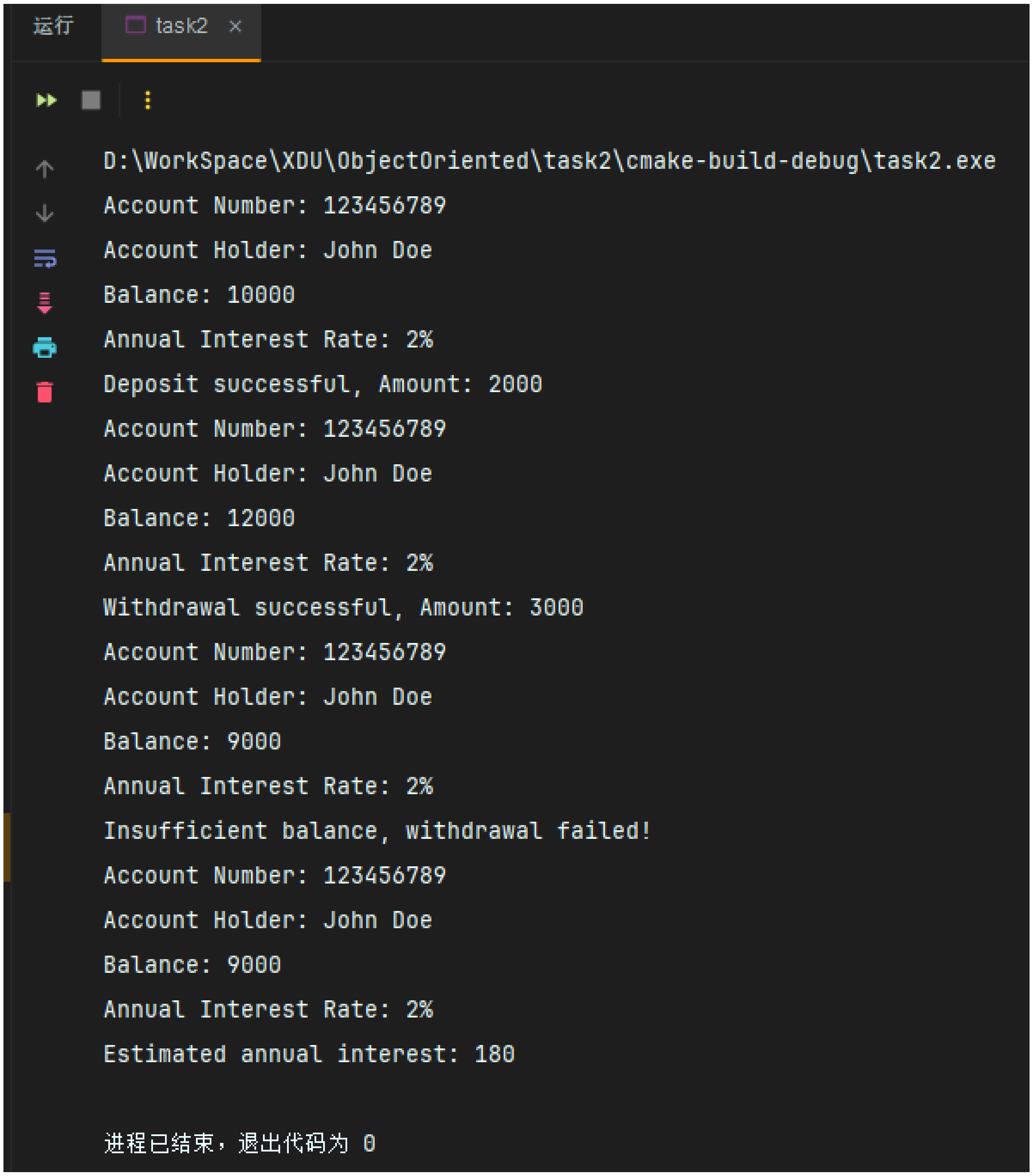
```
class SavingsAccount {

    // 取钱方法
    bool withdraw(double amount) {
        if (amount > balance) {
            cout << "Insufficient balance, withdrawal failed!" << endl;
            return false;
        } else if (amount <= 0) {
            cout << "Please enter a valid withdrawal amount!" << endl;
            return false;
        } else {
            balance -= amount;
            cout << "Withdrawal successful, Amount: " << amount << endl;
            return true;
        }
    }

    // 计算年利息
    double calculateInterest() const {
        return balance * annualInterestRate;
    }

    // 打印账户信息
    void printAccountInfo() const {
        cout << "Account Number: " << accountNumber << endl;
        cout << "Account Holder: " << accountName << endl;
        cout << "Balance: " << balance << endl;
        cout << "Annual Interest Rate: " << annualInterestRate * 100 << "%" << endl;
    }
};
```

编译源文件(包含主函数测试)



```cpp
#include "SavingAccount.h"

int main() {
    // 创建账户
    SavingsAccount myAccount( num: "123456789",  name: "John Doe",  pwd: "pwd123",  bal: 10000,  rate: 0.02);

    // 打印账户信息
    myAccount.printAccountInfo();

    // 测试存款功能
    myAccount.deposit( amount: 2000);
    myAccount.printAccountInfo();

    // 测试取款功能
    if (myAccount.withdraw( amount: 3000)) {
        myAccount.printAccountInfo();
    }

    // 测试取款失败（余额不足）
    if (!myAccount.withdraw( amount: 10000)) {
        myAccount.printAccountInfo();
    }

    // 计算并显示年利息
    double interest = myAccount.calculateInterest();
    cout << "Estimated annual interest: " << interest << endl;

    return 0;
}
```

运行结果



```
D:\WorkSpace\XDU\ObjectOriented\task2\cmake-build-debug\task2.exe
Account Number: 123456789
Account Holder: John Doe
Balance: 10000
Annual Interest Rate: 2%
Deposit successful, Amount: 2000
Account Number: 123456789
Account Holder: John Doe
Balance: 12000
Annual Interest Rate: 2%
Withdrawal successful, Amount: 3000
Account Number: 123456789
Account Holder: John Doe
Balance: 9000
Annual Interest Rate: 2%
Insufficient balance, withdrawal failed!
Account Number: 123456789
Account Holder: John Doe
Balance: 9000
Annual Interest Rate: 2%
Estimated annual interest: 180

进程已结束，退出代码为 0
```