##### 面向对象程序设计方法

1. **题目分析**

### 1.功能描述

几乎所有的RPG游戏（一种源自《龙与地下城》的游戏类型）在进入游戏时都会让用户自己来创建自己喜欢的角色。本次上机要求编写一个简化的创建游戏角色的程序。

### 2.游戏角色应有的属性

本题目要求的游戏角色应有以下属性：名字、性别、种族、职业、力量、敏捷、体力、智力、智慧、生命值和魔法值。

名字：不超过50个字符。

性别：可以选择男性和女性。

种族：一共可选五个种族，人类、精灵、兽人、矮人和元素。

职业：可选六种职业，狂战士、圣骑士、刺客、猎手、祭司和巫师。

其余属性均为整数。

本题目要求首先用户输入角色姓名，然后由用户选择角色性别，然后由用户选择种族，然后选择职业，然后自动分配力量、敏捷、体力、智力和智慧属性，并计算生命值和魔法值。

生命值=体力\*20。

魔法值=（智力+智慧）\*10。

### 3.职业限制

很多职业会限制某些种族选择，例如兽人不能就职圣骑士等等，种族和职业的限制表如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 种族/职业 | 狂战士 | 圣骑士 | 刺客 | 猎手 | 祭司 | 巫师 |
| 人类 | 允许 | 允许 | 允许 | 允许 | 允许 | 允许 |
| 精灵 | 不允许 | 不允许 | 允许 | 允许 | 允许 | 允许 |
| 兽人 | 允许 | 不允许 | 不允许 | 允许 | 允许 | 不允许 |
| 矮人 | 允许 | 允许 | 不允许 | 不允许 | 允许 | 不允许 |
| 元素 | 不允许 | 不允许 | 不允许 | 不允许 | 允许 | 允许 |

所以在要求用户选择职业时，输出信息里面只能有用户所选择种族可以就职的职业。

### 4.初始属性

本题目要求力量、敏捷、体力、智力和智慧要求是随机值（利用随机数函数来取得随机数），但是五项属性的总和应该是100，并且应该和职业相关。例如狂战士的体力和力量就要比较高，而巫师需要较高的智力，而祭司则需要较高的智慧。各职业初始属性的大致比例应遵从下表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 职业/属性 | 力量 | 敏捷 | 体力 | 智力 | 智慧 |
| 狂战士 | 40 | 20 | 30 | 5 | 5 |
| 圣骑士 | 25 | 15 | 30 | 20 | 10 |
| 刺客 | 20 | 35 | 20 | 15 | 10 |
| 猎手 | 15 | 40 | 15 | 10 | 20 |
| 祭司 | 15 | 20 | 15 | 35 | 15 |
| 巫师 | 10 | 20 | 10 | 20 | 40 |

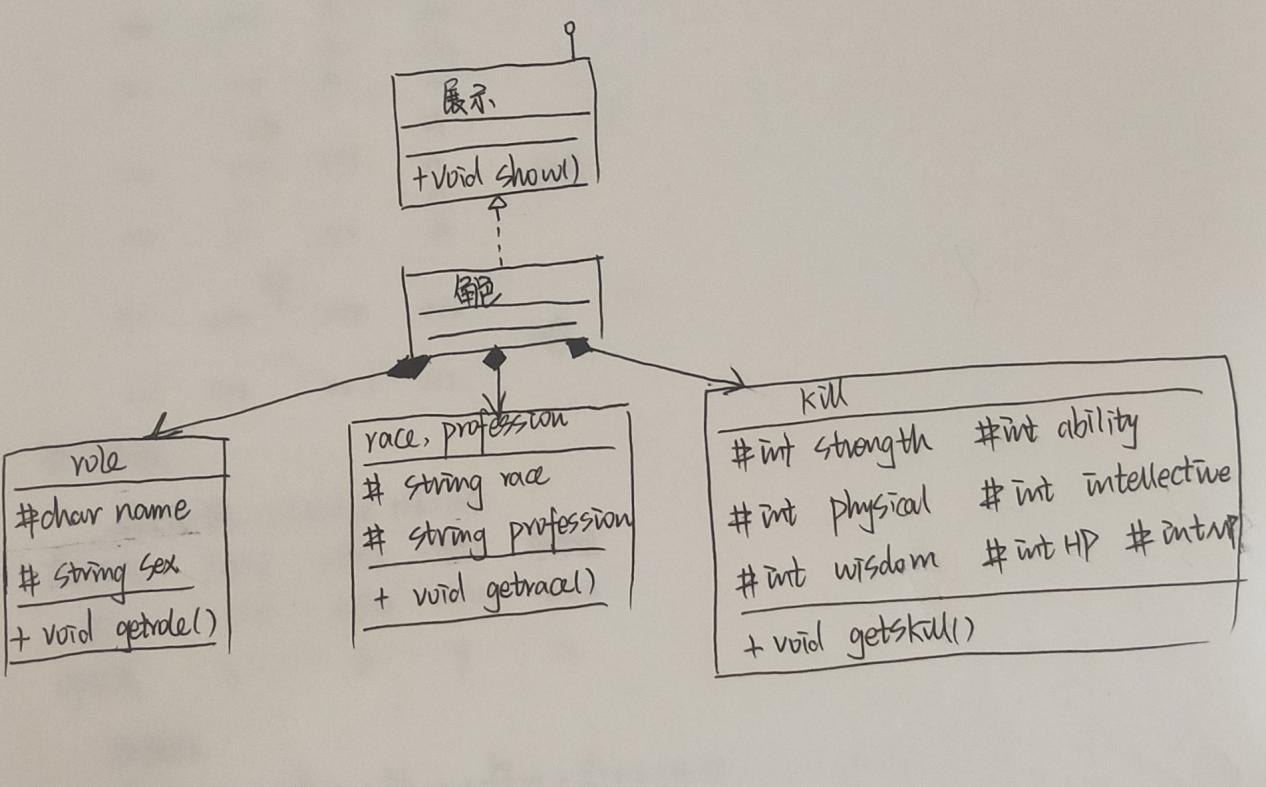
例如，前面示意图中的祭司的初始属性，大致满足该比例，但是应该是随机的。

然后利用属性值计算生命值和魔法值。

### 5.显示信息

最后向用户显示该角色的所有信息，然后询问用户是否满意，如用户不满意则重新创建，若用户满意则程序结束，并将用户创建角色的相关信息写入文件保存。

1. **类图设计**

****

遵循的原则

1. 单一职责原则

将其要实现的属性分为3个大类，每一个类都有其所要实现的功能。

1. 角色类

成员变量：char name,string sex

成员方法：void getrole()

主要功能：生成角色的名字和姓名

1. 种族、职业类

成员变量：string race，string profession

成员方法：void getrace()

主要功能：从五个种族，人类、精灵、兽人、矮人和元素选出自己游戏人物的所属种族，在种族的选择下会有很多职业限制，因此，种族、职业类会在用户选择种族时，输出信息里面只能有用户所选择种族可以就职的职业。

1. 属性类

成员变量：int strength，int ability，int physical，int intellective，int wisdom，int HP，int MP

成员方法：void getskill()

主要功能：选择种族、职业，然后自动分配力量、敏捷、体力、智力和智慧属性，并计算生命值和魔法值。

生命值=体力\*20。

魔法值=（智力+智慧）\*10。

1. 开闭原则

在设计类的过程中，没有采用继承等关系，而是采用了3个独立大类，实现展示属性界面这一接口，这一点正是充分的应用了开闭原则，是展示界面可以在不修改其他大类的情况下进行扩展。

1. 里氏代换原则

一个总的属性生成基类，可以根据角色的具体要求和考量，将其分为3大子类，在程序中尽量使用基类类型来对对象进行定义，而在运行时在确定其子类类型，用子类对象替换父类的对象。

遵循的模式

1. 工厂模式

在创建角色的工厂父类，负责实现展示角色公共接口，而创建角色的三个子类中，每一个类则负责生产具体的属性值，这样做的目的是将属性的实例化操作延迟到工厂子类中完成，即通过工厂子类来确定究竟实例化那个属性值。

1. 单例模式

每创建一个角色的姓名、性别、种族，都会在文件类中传入对应地参数，使其选择的职业与对应地属性值是一定的，确保最终展示这个大类只有只有一个实例，而且自行实例化并向整个系统提供这个实例，这个大的游戏角色展示类，称为单例类，它提供全局访问的方法。

1. **程序实现**

include<iostream> //输入输出流头文件

#include<stdlib.h> //标准库头文件

#include<fstream>//操作文件读入写出头文件

#include<ctime>//随机数头文件

#include<string>//使用字符串头文件

using namespace std; //使用命名空间std

int profession\_choice;//选择职业序号数字

class role//角色基类

{

protected:

char name[50]; //名字

string sex; //性别

public:

void getrole(); //得到角色基本信息函数

friend class output; //友元类，用于输出角色信息

friend class File;//友元类，将角色信息保存在文档中

};

void role::getrole()// 得到角色基本信息

{

cout << "输入您游戏角色的姓名:" << endl;

cin >> name;//输入姓名

int a; //选择角色性别序号

cout << "选择您游戏角色的性别(0 男性，1 女性):" << endl;

cin >> a;

if (a == 0)

sex = "男性";

else if (a == 1)

sex = "女性";

else

{

cout << "输入错误，请重新输入" << endl;

getrole();

}

}

class roce //role派生类，记录角色的种族和职业

{

protected:

string race; //种族

string profession; //职业

public:

void getrace(); //得到种族和职业函数

friend class output; //友元类，用于输出角色信息

friend class File;//友元类，将角色信息保存在文档中

};

void roce::getrace()//得到种族和职业函数

{

int a;

cout << "选择您游戏角色的种族(0 人类，1 精灵，2 兽人，3 矮人，4 元素):" << endl;

cin >> a; //选择角色种族序号

if (a == 0)

{

race = "人类";

cout << "选择您的职业(0 狂战士，1 圣骑士，2 刺客，3 猎手，4 祭祀，5 巫师):" << endl;

}

else if (a == 1)

{

race = "精灵";

cout << "选择您的职业(2 刺客，3 猎手，4 祭祀，5 巫师):" << endl;

}

else if (a == 2)

{

race = "兽人";

cout << "选择您的职业(0 狂战士，3 猎手，4 祭祀):" << endl;

}

else if (a == 3)

{

race = "矮人";

cout << "选择您的职业(0 狂战士，1 圣骑士，4 祭祀):" << endl;

}

else if (a == 4)

{

race = "元素";

cout << "选择您的职业(4 祭祀，5 巫师):" << endl;

}

else

{//输入数字有误时重新调用函数

cout << "输入有误请输入正确数字" << endl;

getrace();

}

cin >> profession\_choice; //选择职业序号数字

//根据数字为职业赋值

switch (profession\_choice)

{

case 0:profession = "狂战士"; break;

case 1:profession = "圣骑士"; break;

case 2:profession = "刺客"; break;

case 3:profession = "猎手"; break;

case 4:profession = "祭司"; break;

case 5:profession = "巫师"; break;

default:getrace();

}

}

class skill //roce派生类skill记录角色的属性

{

protected:

int strength; //力量

int agility; //敏捷

int physical;//体力

int intellective; //智力

int wisdom; //智慧

int HP;//生命值

int MP; //魔法值

public:

void getskill(); //获得属性函数

void getrandom(int a, int b, int c, int d, int e); //生成角色属性随机数函数

friend class output; //友元类，用于输出角色信息

friend class File;//友元类，将角色信息保存在文档中

};

void skill::getrandom(int a, int b, int c, int d, int e) //随机生成每项属性的值

{

int sum; //5个属性的总和

srand((unsigned)time(NULL));//产生随机种子

do

{

strength = a + (rand() % 5); //力量

agility = b + (rand() % 5); //敏捷

physical = c + (rand() % 5); // 体力

intellective = d + (rand() % 5); //智力

wisdom = e + (rand() % 5); //智慧

sum = strength + agility + physical + intellective + wisdom;

} while (sum < 100);

HP = physical \* 20; // 生命值

MP = (wisdom + intellective) \* 10; //魔法值

}

void skill::getskill() //根据选择的职业，向 getrandom传各个职业的大致比例而生成随机数

{

if (profession\_choice == 0)

getrandom(40, 20, 30, 5, 5);//狂战士

if (profession\_choice == 1)

getrandom(25, 15, 30, 20, 10);//圣骑士

if (profession\_choice == 2)

getrandom(20, 35, 20, 15, 10);//刺客

if (profession\_choice == 3)

getrandom(15, 40, 15, 10, 20);//猎手

if (profession\_choice == 4)

getrandom(15, 20, 15, 35, 15);//祭司

if (profession\_choice == 5)

getrandom(10, 20, 10, 20, 40);//巫师

}

class output //输出角色属性

{

public:

void show(role &, roce &, skill &);//访问友元类

};

void output::show(role &t1, roce &t2, skill &t3)

{

cout << "==================================== " << endl;

cout << "姓名：" << t1.name << endl;

cout << "==================================== " << endl;

cout << "性别：" << t1.sex << endl;

cout << "==================================== " << endl;

cout << "种族：" << t2.race << endl;

cout << "==================================== " << endl;

cout << " 职业：" << t2.profession << endl;

cout << "====================================" << endl;

cout << "力量：" << t3.strength << endl;

cout << " ====================================" << endl;

cout << "敏捷： " << t3.agility << endl;

cout << "==================================== " << endl;

cout << "体力：" << t3.physical << endl;

cout << "==================================== " << endl;

cout << "智力： " << t3.intellective << endl;

cout << "==================================== " << endl;

cout << " 智慧：" << t3.wisdom << endl;

cout << "==================================== " << endl;

cout << "生命值：" << t3.HP << endl;

cout << " ====================================" << endl;

cout << "魔法值：" << t3.MP << endl;

cout << "==================================== " << endl;

}

//将角色信息保存在文档中

class File

{

public:

void file(role &, roce &, skill &);

};

void File::file(role &t1, roce &t2, skill &t3)

{

ofstream outfile;

outfile.open("data.txt", ios::out);

cout << "" << endl;

outfile << "姓名:" << t1.name << endl;

cout << "" << endl;

outfile << "性别:" << t1.sex << endl;

cout << "" << endl;

outfile << " 种族:" << t2.race << endl;

cout << "" << endl;

outfile << " 职业:" << t2.profession << endl;

cout << "" << endl;

outfile << " 力量:" << t3.strength << endl;

cout << "" << endl;

outfile << " 敏捷:" << t3.agility << endl;

cout << "" << endl;

outfile << "体力:" << t3.physical << endl;

cout << "" << endl;

outfile << "智力:" << t3.intellective << endl;

cout << "" << endl;

outfile << "智慧: " << t3.wisdom << endl;

cout << "" << endl;

outfile << "生命值: " << t3.HP << endl;

cout << "" << endl;

outfile << "法力值: " << t3.MP << endl;

cout << "" << endl;

}

//主函数

int main()

{

role player;//角色基类对象

roce player\_race;//角色派生类对象

skill player\_ski; //属性类对象

output player\_show; //输出类

File test;

int i;

do

{

player.getrole(); //调用获得名字，性别信息函数

player\_race.getrace(); //调用得到 种族，职业信息函数

player\_ski.getskill();//调用获得角色属性函数

player\_show.show(player, player\_race, player\_ski);//输出信息

cout << endl;

cout << "是否继续生成角色游戏？" << endl;

cout << "0.退出游戏 1.继续生成" << endl;

cin >> i;

} while (i);

test.file(player, player\_race, player\_ski); //将角色信息保存在文档中

return 0;

}

**4.调试及测试结果**

4.1创建角色姓名以及性别

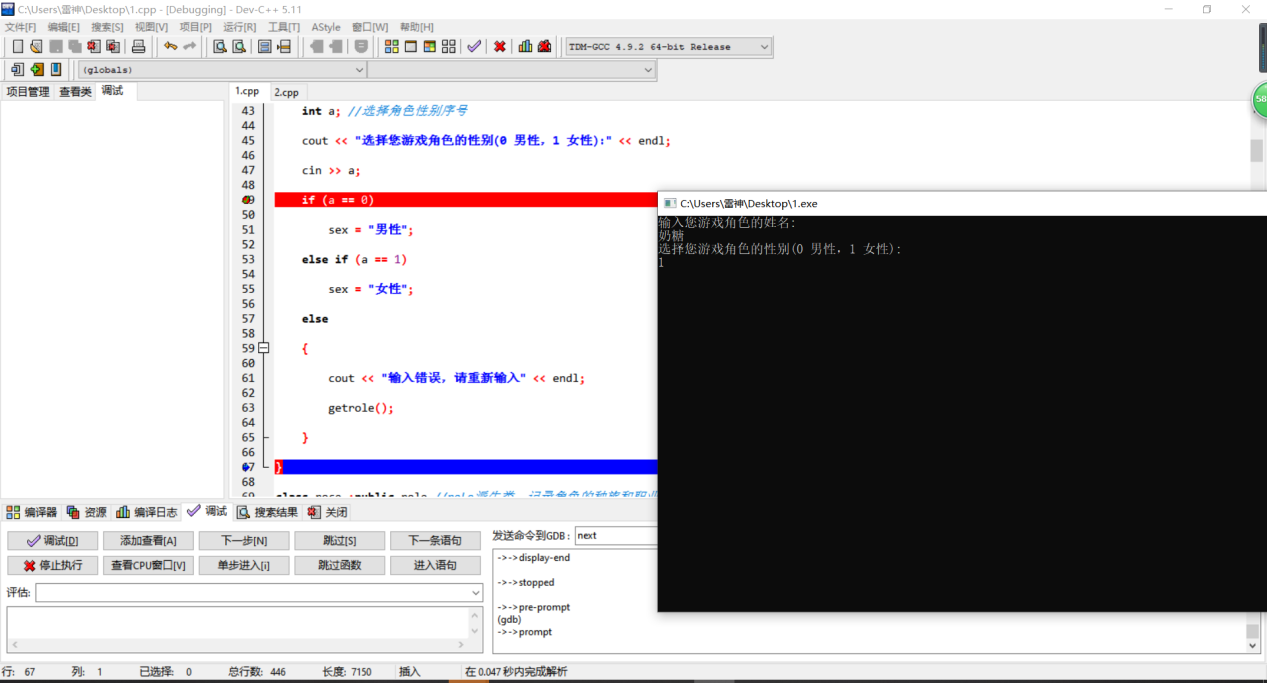
4.1.1调试函数

void role::getrole()

4.1.2调试变量

Name=奶糖；a=1

4.1.3调试截图



4.2role派生类，记录角色的种族和职业

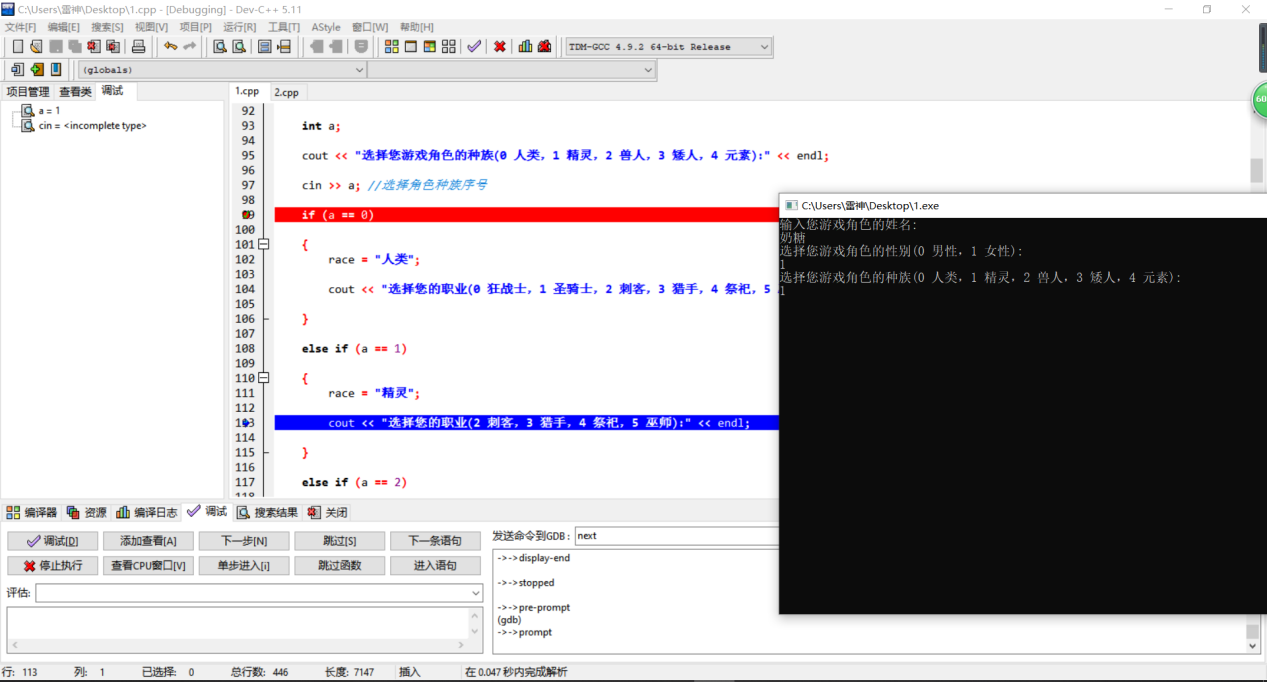
4.2.1调试函数

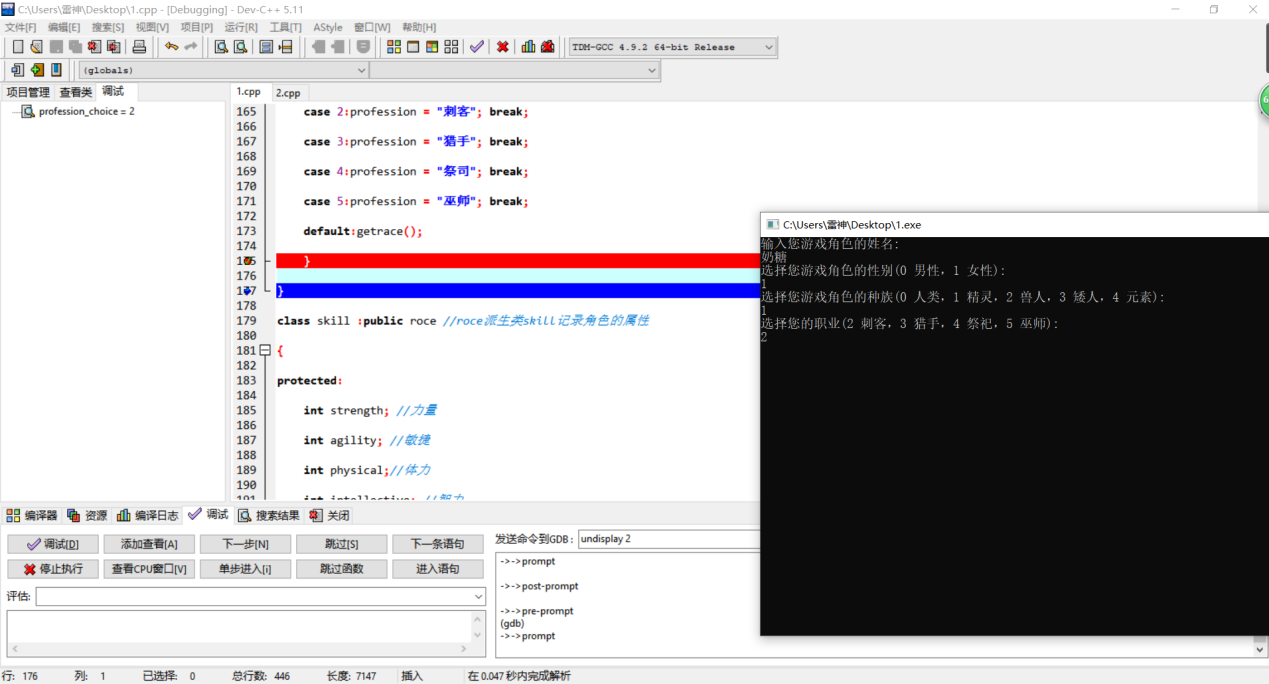
void roce::getrace()

4.2.2调试变量

a = 1;；race=精灵；profession\_choice =3

4.2.3调试截图





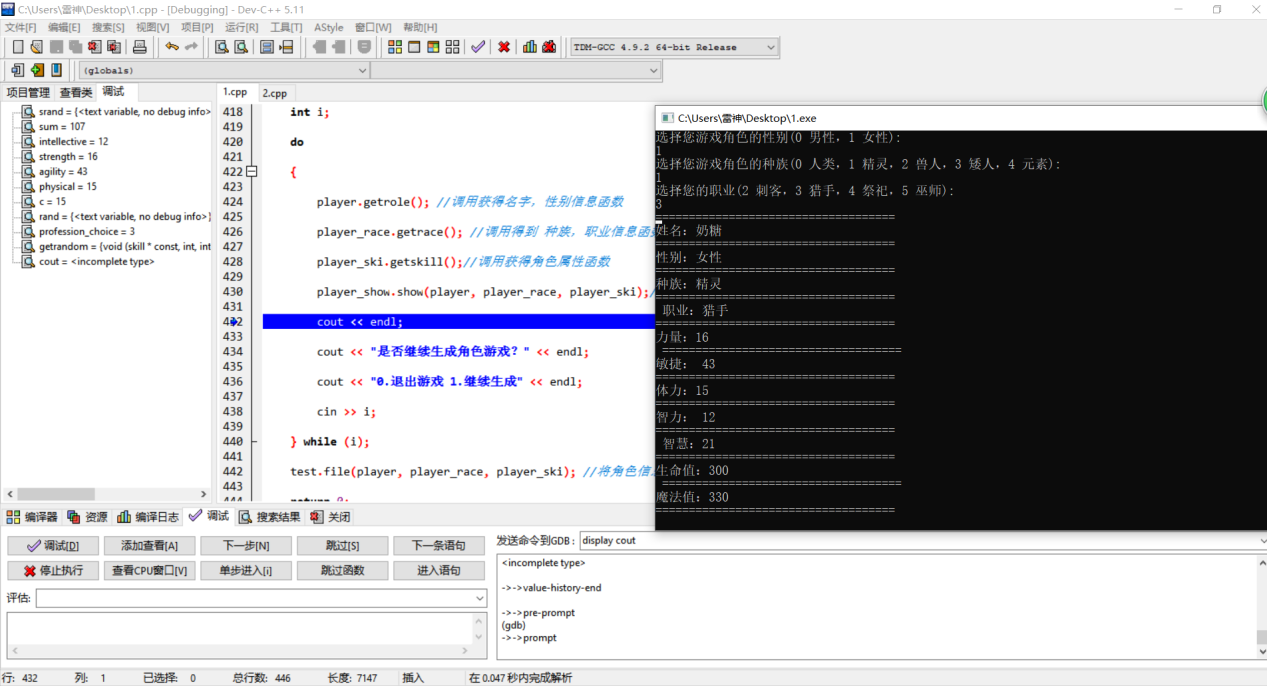
4.3roce派生类skill记录角色的属性

4.3.1调试函数

void skill::getrandom(int a, int b, int c, int d, int e)

4.3.2调试变量

|  |  |  |  |
| --- | --- | --- | --- |
| 变量 | strength | agility | intellective |
| 值 | 16 | 43 | 12 |
| 变量 | physical | wisdom | sum |
| 值 | 15 | 21 | 107 |
| 变量 | HP | MP |  |
| 值 | 300 | 330 |  |



**5.经验归纳**

1.掌握面向对象程序设计的方法。在类与对象的概念这一方面，加深了理解，首先建一个基类Base用来保存和实现,保存玩家输入的姓名和性别的选择,同时需要声明两个友缘类Output，File可以访问该类中的数据。

在基类Base基础上建一个派生类Race,用来记录玩家选择的种族,和职业,同时规定什么种族可以选择什么职业，人类可以选择全部职业；精灵不能选择狂战士，圣骑士；兽人不能选择圣骑士，刺客；矮人不能选择刺客，猎手；元素只能选择祭司，同时需要声明两个友缘类Output，File可以访问该类中的数据。

在派生类Race基础上派生一个Attribute类用来记录角色随机生产的属性，首选规定各个职业每个属性的最小值，在这个最小值的基础上，随机生成差距在3左右的属性，再根据随机生产的属性计算出该角色的血量值，法力值，同时需要声明两个友元类Output，File可以访问该类中的数据。

2.理解并应用了面向对象设计七大原则以及常见的设计模式以及类图的描述，在知识的理解方法更加的深入了。