**实验报告二**

1. **程序设计思路**

当程序规模变大后，一个程序用多个文件组织，便于组织生产，提高程序的编写效率，c++中常用的做法是独立出一个头文件来，内联函数提高了程序中函数调用的效率。在本次实验中，我们在头文件中建立了一个名为“gongji”的类，相当于建立了一个攻击动作的模板，这样以后无论发动什么样的攻击只要改变类中参数即可，不需要再重新编写攻击函数。本程序还建立了一个欢迎界面的模块，可以被随时调用，这样以后就不用重复编写欢迎界面了。

1. **程序代码**
2. 主函数

#include <iostream>

#include"gongji.h"

#include"show.h"

using namespace std;

int main()

{

show();

cout<<"》 普通攻击：按键1 （敌方英雄受到攻击后减少20血量"<<endl;

cout<<"》 技能一： 按键2 （敌方英雄受到攻击后减少25血量"<<endl;

cout<<"》 技能二： 按键3 （敌方英雄受到攻击后减少30血量"<<endl;

cout<<"》 技能三： 按键4 （敌方英雄受到攻击后减少35血量"<<endl;

cout<<"》 敌方英雄血量："<<endl;

cout<<"》 100"<<endl;

gongji c;

}

1. 欢迎界面函数

#include"gongji.h"

void show()

{cout<<"》 亲爱的召唤师：你好，我你的助手妲己！！"<<endl;

cout<<"》 欢迎来到王者峡谷！！"<<endl;

cout<<"》 敌军还有五秒钟到达战场！！"<<endl;

cout<<"》 让我们一起战斗吧！！"<<endl;

}

1. 欢迎界面头函数

void show();

1. 攻击函数

#include <iostream>

using namespace std;

class gongji

{

public:

gongji ()

{ int a2; //a2：攻击

static int a1=100;//a1：血量

int a3=0; //a3：击杀数

while(a3<5)

{

{while(a1>0)

{

cin>>a2;

switch(a2)

{

case 1 :a1=a1-20;break;

case 2 :a1=a1-25;break;

case 3 :a1=a1-30;break;

case 4 :a1=a1-35;break;

default : cout<<"无效键"<<endl;break;

}

cout<<"敌方英雄血量："<<endl;

cout<<a1<<endl;

}

a3=a3+1;

a1=100;

cout<<"#####敌方英雄被击败！！#####"<<endl;

}

}

cout<<" >>>>>>>>>>五连绝世！！<<<<<<<<<<"<<endl;

cout<<" >>>>>>>>>>五连绝世！！<<<<<<<<<<"<<endl;

while(a3=5)

{ cin>>a2;

switch(a2)

{

case 1 :cout<<"\*\*\*\*\*\*\*\*\*\*敌方已被团灭\*\*\*\*\*\*\*\*\*\*"<<endl;break;

case 2 :cout<<"\*\*\*\*\*\*\*\*\*\*敌方已被团灭\*\*\*\*\*\*\*\*\*\*"<<endl;break;

case 3 :cout<<"\*\*\*\*\*\*\*\*\*\*敌方已被团灭\*\*\*\*\*\*\*\*\*\*"<<endl;break;

case 4 :cout<<"\*\*\*\*\*\*\*\*\*\*敌方已被团灭\*\*\*\*\*\*\*\*\*\*"<<endl;break;

default : cout<<"~~~~~无效键!!!!!"<<endl;break;

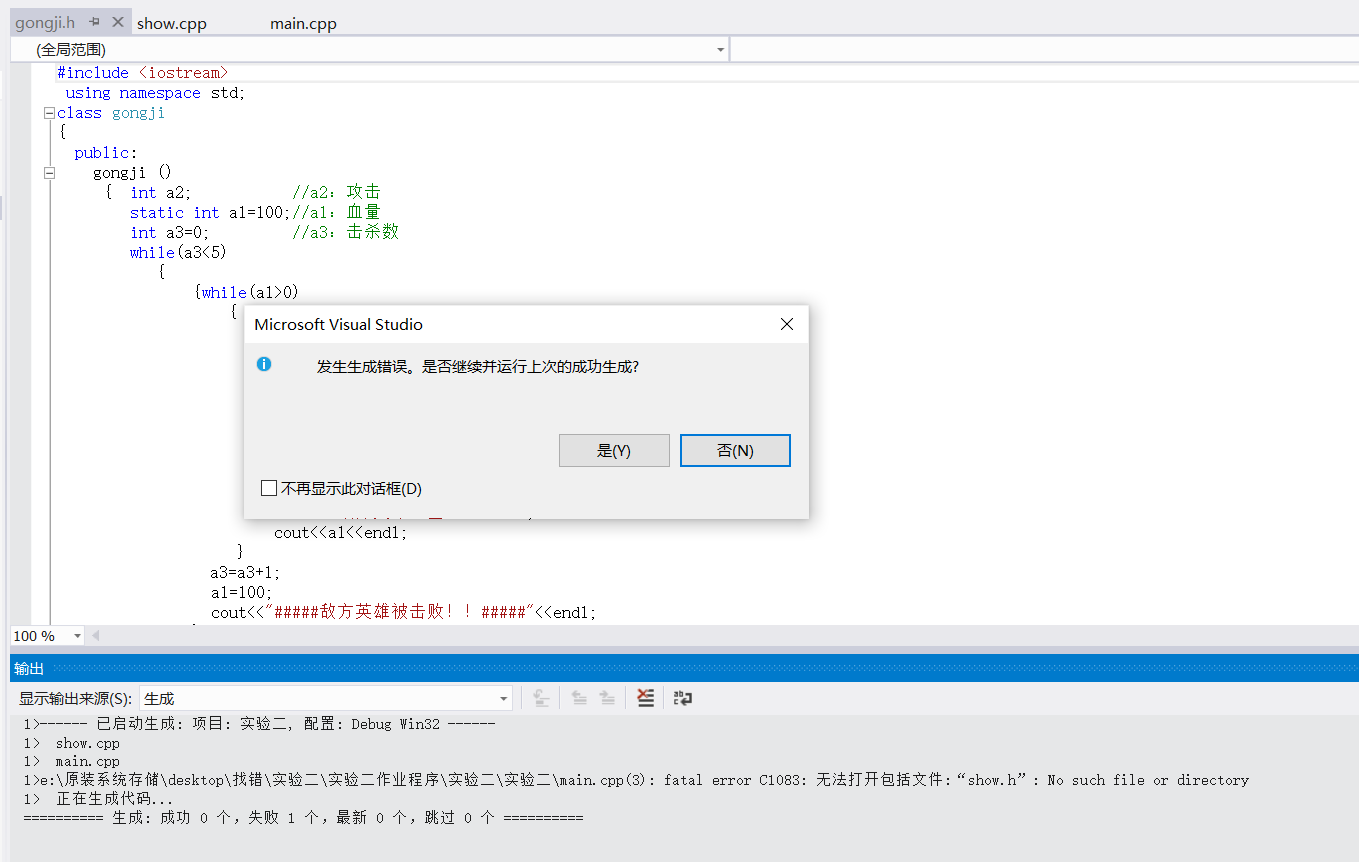
}

}

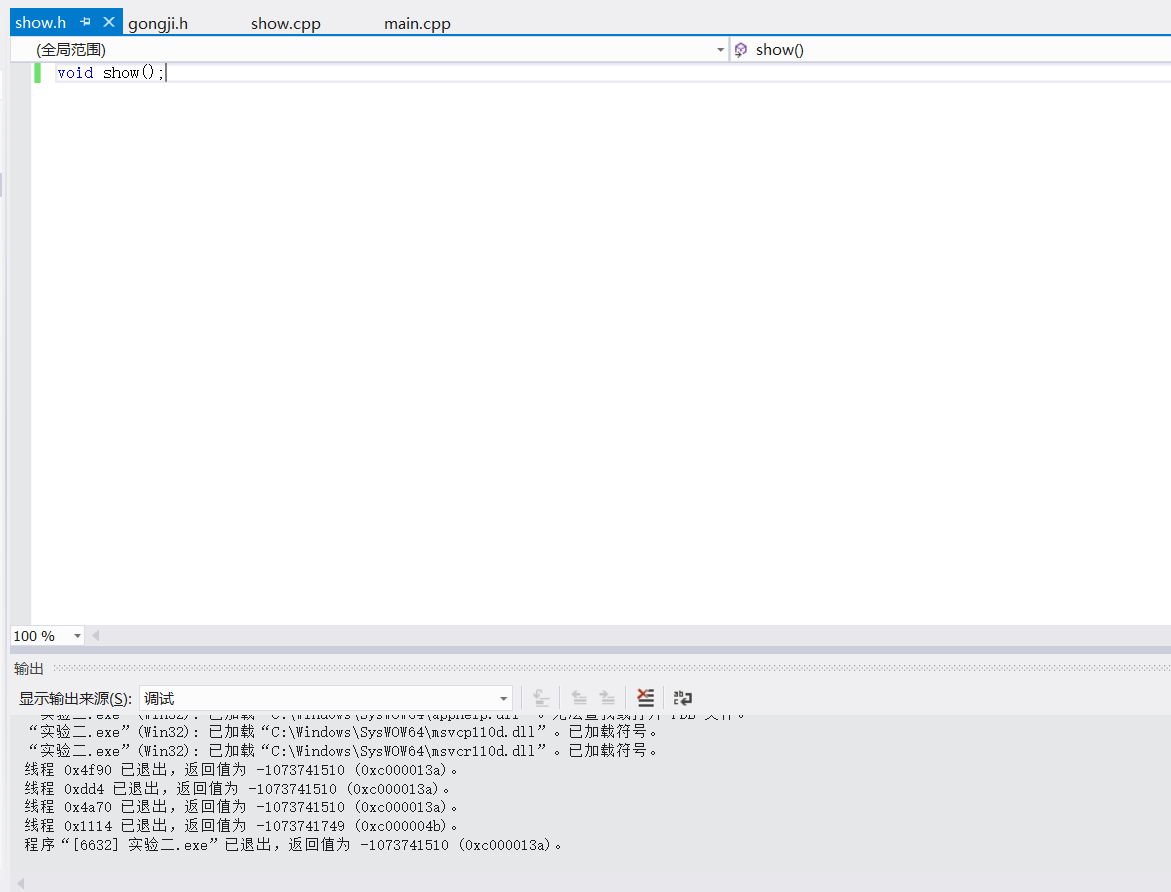
}

};

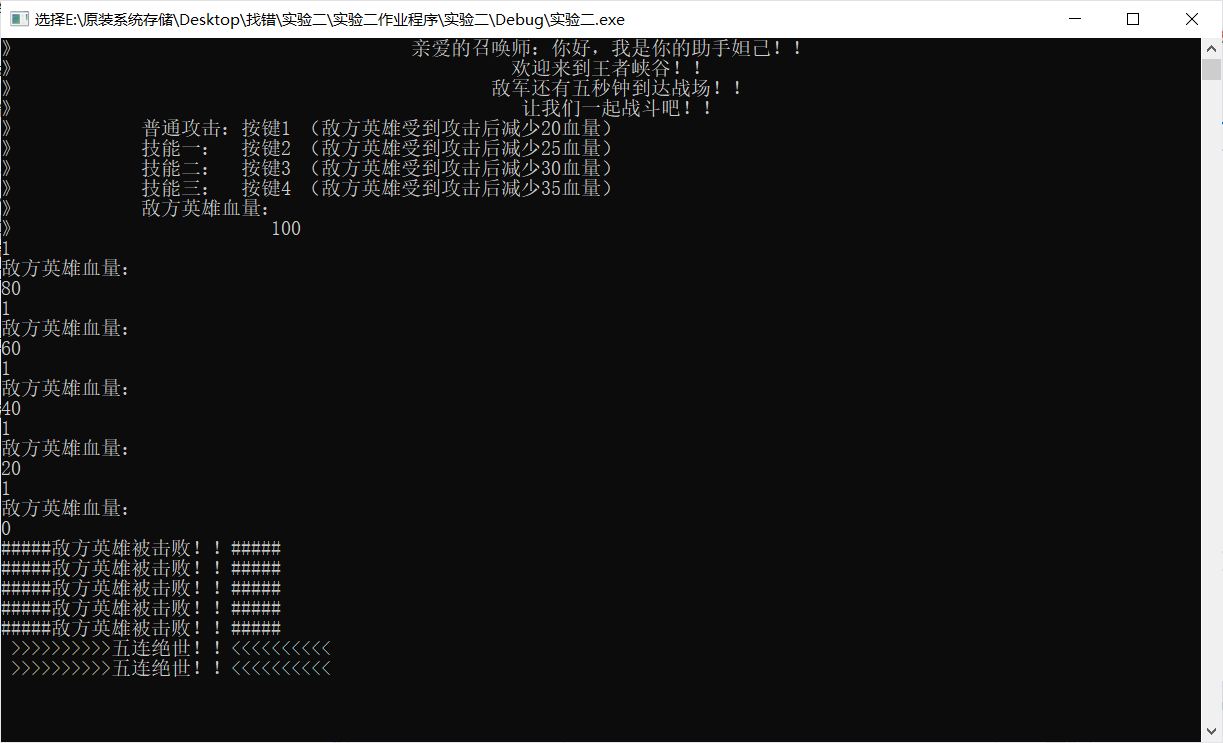
1. **程序错误及解决方法**
2. 在本次调试过程中出现了程序无法正常编译的现象，如下图：



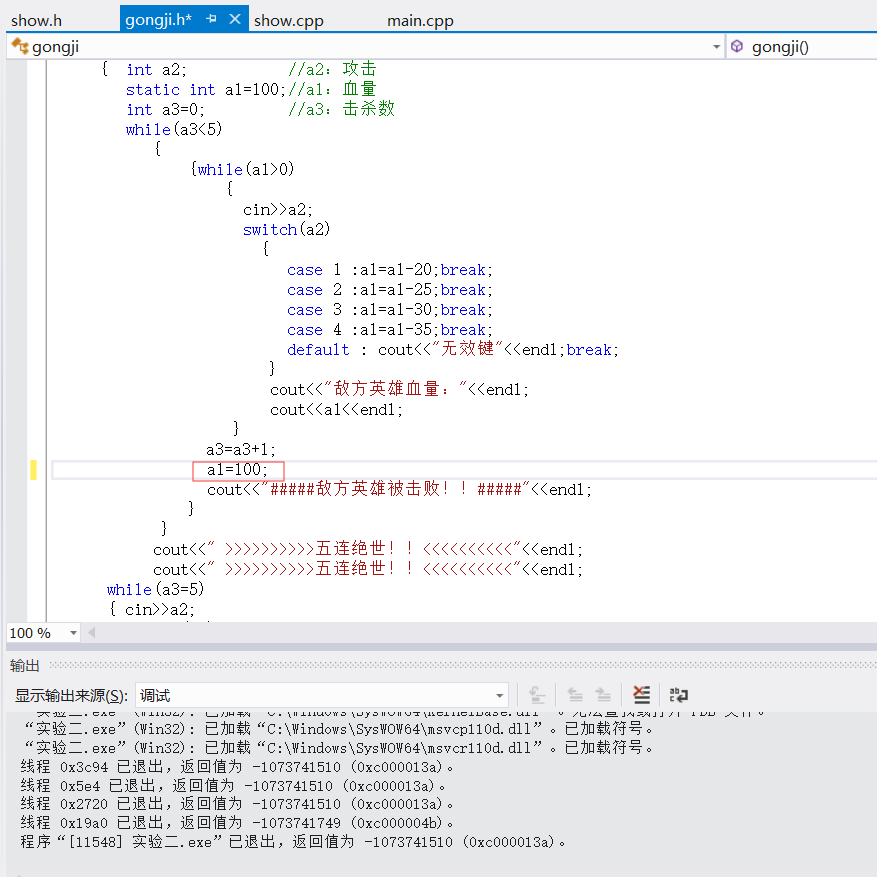
错误更改：根据提示，程序无法调用“show”函数，经过检查发现未建立“show”函数的头文件，更改如下：



2、在程序运行后发现，击败一个英雄就会出现“五连绝世”的现象，现象如下：



错误更改：经检查发现程序在每次完成击杀动作后没有对血量进行初始化，下面我们对血量进行初始化：



1. **程序运行结果**

