**C++实验报告二：“.h文件的使用”**

**刘振宇 自18G2 201830312021**

**设计程序的逻辑思路：**

在本次上机实验中，通过一个简单的程序，黄老师给我们讲解了.h文件的使用方法和在实际的开发中的应用。真实的一个大型的软件项目,可能有几千个,上万个 add 而不只是add1,add2这么简单, 这样就可能有N个类似 add1();add2(); 这样的声明, 这个时候就需要我们想办法把这样的 add1(); add2(); 另行管理, 而不是把他放在.cpp文件中, 于是.h 文件出现了。这次的上机让我了解了.h文件的使用方法

**程序：**

main.cpp下的程序内容如下：

#include<iostream>

#include"add.h"

usingnamespacestd;

intmain()

{

intx,y,sum;

cout<<"Enter two numbers:"<<endl;

cin>>x>>y;

sum=add(x,y);

cout<<x<<"+"<<y<<"="<<sum<<endl;

return0;

}

add.cpp下的程序如下

#include"add.h"

intadd(inta,intb)

{

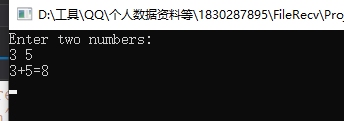
returna+b;

}

add.h下的程序如下

intadd(inta,intb);

**运行结果如下：**



**思考与收获：**

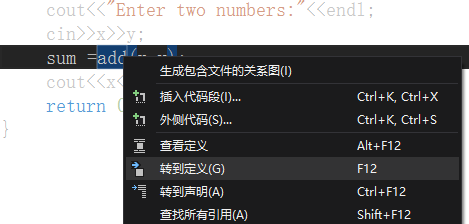
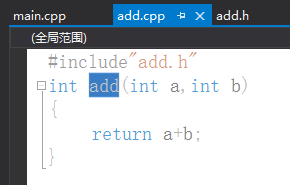
这一节课下来我大致思考并总结了一些.h文件作用:

.h文件中一般可以放入同名.cpp文件中定义的变量、数组还有函数的声明，不过需要.cpp外部使用的声明，才能让计算机知道去哪里找这个同名.h文件。这样的做法无论是在开发编写程序阶段还是从后面编译检查纠错时都非常的简明扼要。因为你编写的这些.h文件包含一些文件所需的一些相同的变量的声明、常量、结构、类型定义、函数，可以大大增加你所编写的程序的可读性和逻辑性。

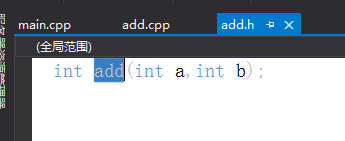
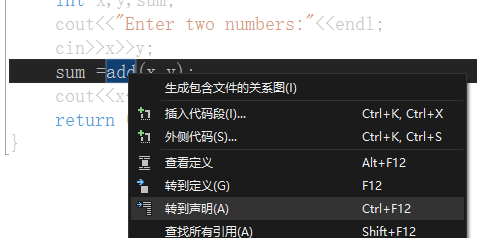
应用了.h文件后，程序的可读性是在一定程度上增加了，但是对于那些本身程序长度就特别长、内容特别多的工程、不管是不是你自己编写的程序，在几百行程序中想要迅速准确的找到某一个函数或者变量的定义，那都是不现实的，所以为了方便我们可以迅速查找定义的变量或者声明的函数，在任何一个被调用的函数上右键，下拉菜单里面“转到定义”和“转到声明”，

点击“转到定义”，光标就会立即显示在程序定义变量的位置上，点击“转到声明”，光标就会立即显示到程序声明函数的位置上。

具体操作如下图所示：

如图从main.cpp下跳转到定义add函数的位置



同上，这就转到声明函数add的位置，非常的方便简捷。

**作业：王者荣耀五连绝世（要用到内联函数）**

**由于程序太过冗长，我把它附在同一文件夹下，运行即可查看。**

**思路：**

（内联函数）因为要不断地拿人头重复地调用某个函数，所以我感觉不如在英雄的击杀数哪里设置一个内联函数，每杀一个人计数+1，从first blood 到penta kill，杀到手软。

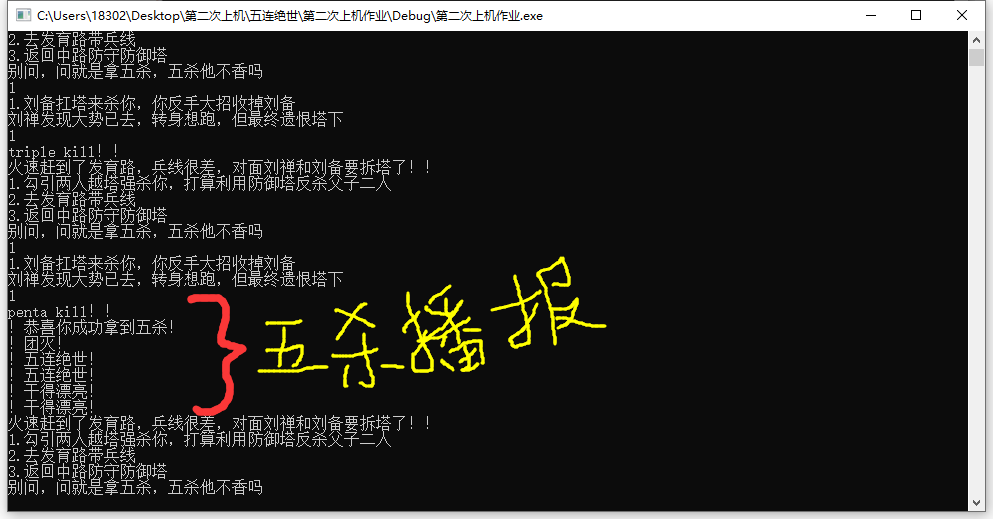
想到程序运行界面是文字信息，不如把拿五杀直接写成一个纯文字游戏，给玩家提供多样化地选择，增加玩家的游戏体验性，就像提前写好的剧本，玩家可以亲身体验拍个电影。

**开始界面如图所示：**



你只要根据我提示的选项就能一步步拿到五杀，简直不要太容易。（激动不？比手机上简单多了**[= =]**）

**最后的系统五杀播报：**



**问题：**

当时在差不多编写完这个程序时，试运行期间发现有个别定义变量的名称打错了，导致系统报错，无法正常运行；

写的这个程序对面只有四个人，可以重复拿相同的英雄的人头来达到五杀，而且五杀之后还能无限次地拿五杀（真的是拿五杀拿到手软）。

**Extern** 关键字：

extern可以置于变量或者函数前，以标示变量或者函数的定义在别的文件中，提示编译器遇到此变量和函数时在其他模块中寻找其定义。此外extern也可用来进行链接指定。

  也就是说extern有两个作用，第一个,当它与"C"一起连用时，如: extern "C" void fun(int a, int b);则告诉编译器在编译fun这个函数名时按着C的规则去翻译相应的函数名而不是C++的，C++的规则在翻译这个函数名时会把fun这个名字变得面目全非。

    第二，当extern不与"C"在一起修饰变量或函数时，如在头文件中: extern int g\_Int; 它的作用就是**声明函数或全局变量的作用范围**的关键字，其声明的函数和变量可以在本模块或其他模块中使用，**记住它是一个声明不是定义**!也就是说B模块(编译单元)要是引用模块(编译单元)A中定义的全局变量或函数时，它只要包含A模块的头文件即可,在编译阶段，模块B虽然找不到该函数或变量，但它不会报错，它会在连接时从模块A生成的目标代码中找到此函数。

问题：**extern 函数声明**  
　　常常见extern放在函数的前面成为函数声明的一部分，那么，C语言的关键字extern在函数的声明中起什么作用？  
　　答案与分析：  
　　如果函数的声明中带有关键字extern，仅仅是暗示这个函数可能在别的源文件里定义，没有其它作用。即下述两个函数声明没有明显的区别：  
extern int f(); 和int f();  
　　当然，这样的用处还是有的，就是在程序中取代include “\*.h”来声明函数，在一些复杂的项目中，我比较习惯在所有的函数声明前添加extern修饰。关于这样做的原因和利弊可见下面的这个例子：“用extern修饰的全局变量”

    (1) 在test1.h中有下列声明:  
    #ifndef TEST1H  
    #define TEST1H  
    extern char g\_str[]; // 声明全局变量g\_str  
    void fun1();  
    #endif  
    (2) 在test1.cpp中  
    #include "test1.h"  
        char g\_str[] = "123456"; // 定义全局变量g\_str  
        void fun1() { cout << g\_str << endl; }  
    (3) 以上是test1模块， 它的编译和连接都可以通过,如果我们还有test2模块也想使用g\_str,只需要在原文件中引用就可以了  
    #include "test1.h"

     void fun2()    { cout << g\_str << endl;    }