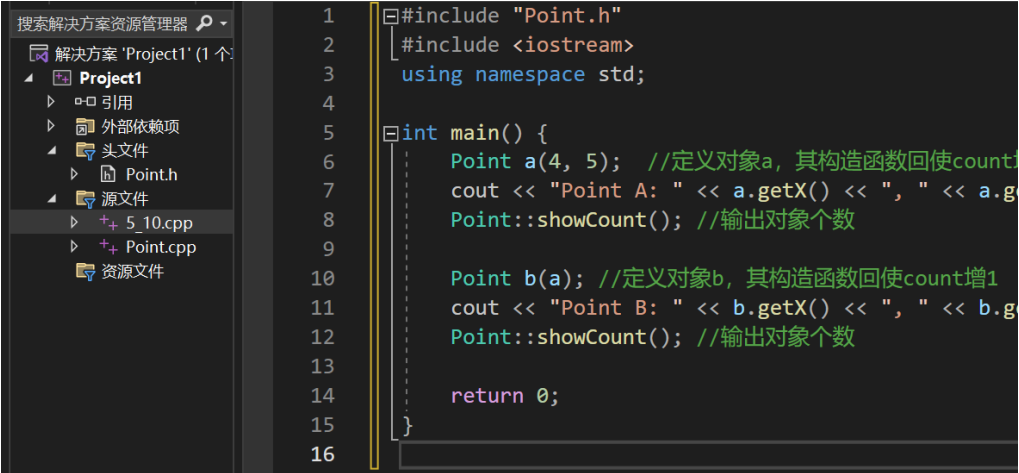
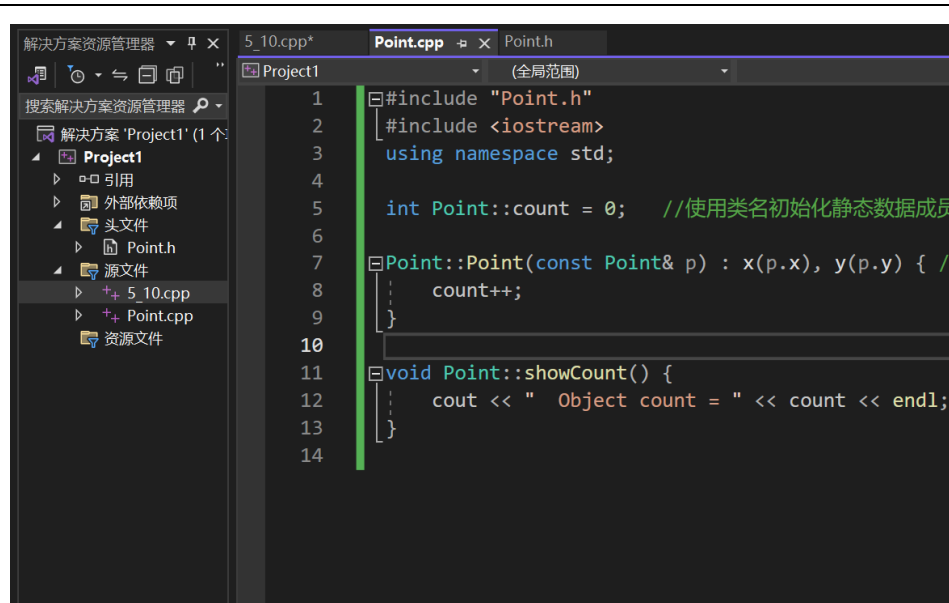
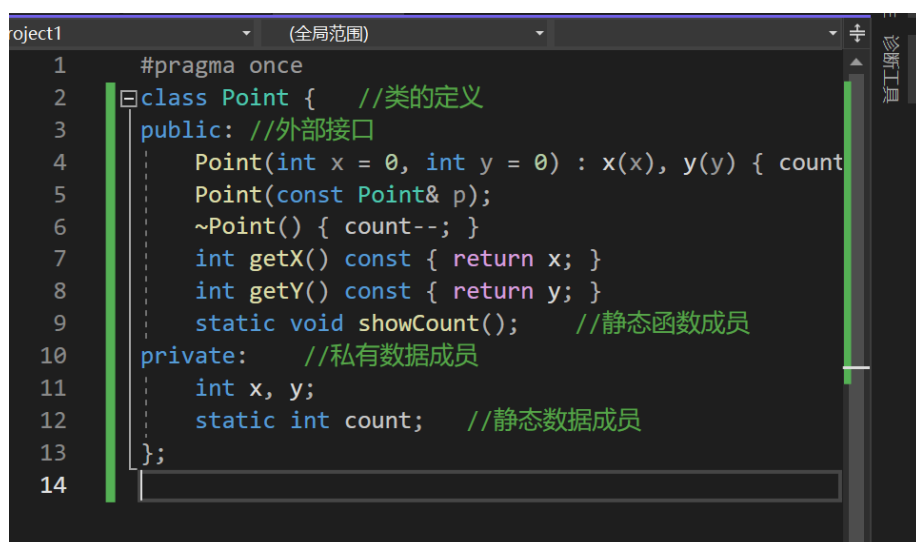


计算机学院 高级程序设计 课程实验报告

实验题目：多文件项目，预编译		学号：202300130150
日期：2024. 3. 28	班级： 4	姓名： 王成意
<p>实验目的：</p> <ol style="list-style-type: none">1. 掌握多文件项目构建2. 熟悉编译预处理指令		
<p>实验步骤与内容：</p> <p>-----方式一、新建一个项目，将所有文件加入-----</p> <p>(1) 建立一个项目文件（名称自定）。在项目中添加例 5-10 中的三个文件。</p> <p>(a) 新建一个项目文件：“新建” / “项目”，选 “Empty project”，输入项目名，“确定”。建立一个 .dev 文件。</p> <p>(b) 在左侧项目管理窗口，右击项目名称，选 “添加”，添加如下三个文件：</p> <div>Point.h Point.cpp 5_10.cpp</div> <p>项目中默认添加的未命名 n 文件可删除。</p> 		

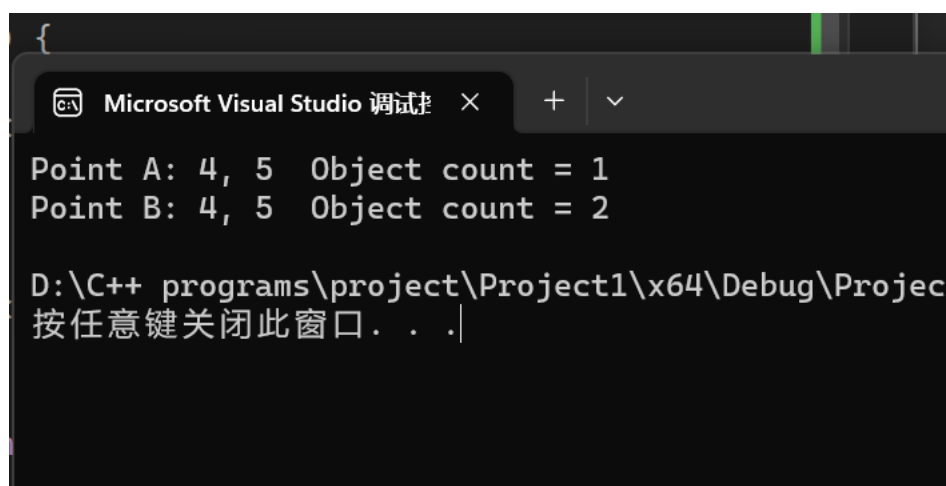


```
1 #include "Point.h"
2 #include <iostream>
3 using namespace std;
4
5 int Point::count = 0; //使用类名初始化静态数据成员
6
7 Point::Point(const Point& p) : x(p.x), y(p.y) { //
8     count++;
9 }
10
11 void Point::showCount() {
12     cout << " Object count = " << count << endl;
13 }
14
```



```
1 #pragma once
2 class Point { //类的定义
3 public: //外部接口
4     Point(int x = 0, int y = 0) : x(x), y(y) { count
5     Point(const Point& p);
6     ~Point() { count--; }
7     int getX() const { return x; }
8     int getY() const { return y; }
9     static void showCount(); //静态函数成员
10 private: //私有数据成员
11     int x, y;
12     static int count; //静态数据成员
13 };
14
```

(c) 编译、运行该项目。



```
{
Microsoft Visual Studio 调试
Point A: 4, 5 Object count = 1
Point B: 4, 5 Object count = 2

D:\C++ programs\project\Project1\x64\Debug\Project1.exe
按任意键关闭此窗口. . .
```

-----方式二、将类实现单独编译，与使用者分开。-----

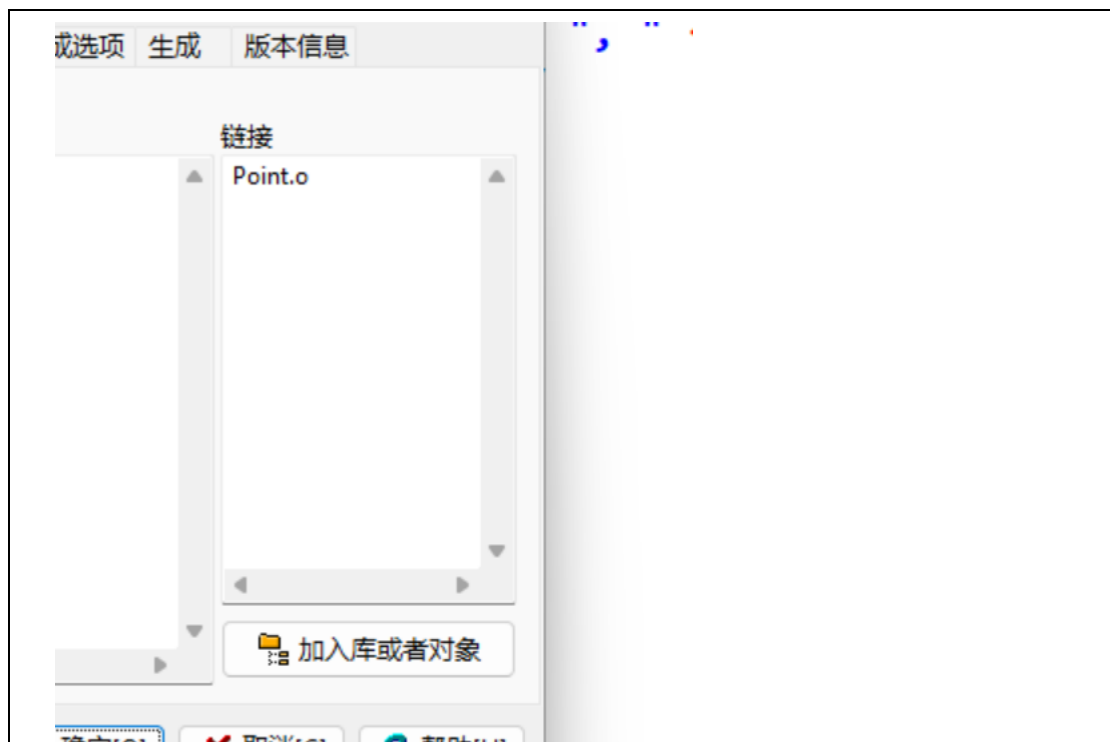
```
1 #include "Point.h"
2 #include <iostream>
3 using namespace std;
4 int Point::count = 0; // 使用类名初始化静态成员变量
5 Point::Point(const Point &p) : x(p.x), y(p.y) {
6     count++;
7 }
8 void Point::showCount() {
9     cout << " Object count = " << count << endl;
10 }
11
```

名称	修改日期	类型	大小
5-10.cpp	2024/3/28 10:23	C++ Source File	1 KB
Makefile.win	2024/3/28 10:23	WIN 文件	2 KB
Point.cpp	2024/3/28 10:23	C++ Source File	1 KB
Point.h	2024/3/28 10:22	C Header 源文件	1 KB
Point.o	2024/3/28 10:22	O 文件	3 KB
项目2.a	2024/3/28 10:22	A 文件	3 KB
项目2.dev	2024/3/28 10:18	Dev-C++ Project File	1 KB
项目2.layout	2024/3/28 10:22	LAYOUT 文件	1 KB
项目4.dev	2024/3/28 10:24	Dev-C++ Project File	2 KB
项目4.layout	2024/3/28 10:24	LAYOUT 文件	1 KB
5-10.o	2024/3/28 10:23	O 文件	5 KB
项目4.exe	2024/3/28 10:23	应用程序	1,879 KB

编译出 point.o 文件如上图。

实现第二个项目：

```
1 class Point { // 类的定义
2 public: // 外部接口
3     Point(int x = 0, int y = 0) : x(x), y(y) {}
4     Point(const Point &p);
5     ~Point() { count--; }
6     int getX() const { return x; }
7     int getY() const { return y; }
8     static void showCount(); // 静态函数
9 private: // 私有数据成员
10     int x, y;
11     static int count; // 静态数据成员
12 };
13
```



加入 point.o 文件
编译运行成功。

```
main() {  
    Point a(4, 5); // 定义对象a, 其构造函数回使coun  
    cout << "Point A: " << a.getX() << " " << a.ge  
    Point  
    Point  
    cout  
    Point  
    retur
```

```
Point A: 4, 5  Object count = 1  
Point B: 4, 5  Object count = 2  
-----  
Process exited after 0.09634 seconds with return value 0  
请按任意键继续. . .
```

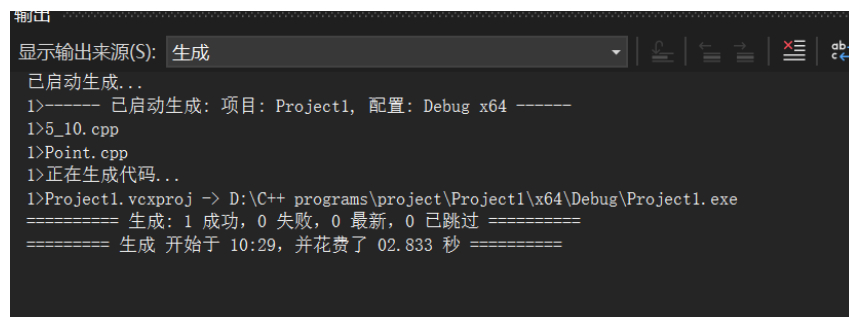
2. 在上述多文件项目环境中完成下列实验（多重定义）

（1）在 Point.h 文件中加入以下函数的定义语句，尝试编译运行，对编译结果进行截图。

```
已启动生成...
1>----- 已启动生成: 项目: Project1, 配置: Debug x64 -----
1>5_10.cpp
1>Point.cpp
1>正在生成代码...
1>LINK : 没有找到 D:\C++ programs\project\Project1\x64\Debug\Project1.exe 或上一个增量链接没有生成它; 正在执行完全链接
1>Point.obj : error LNK2005: "int __cdecl test(void)" (?test@VAHXZ) 已经在 5_10.obj 中定义
1>D:\C++ programs\project\Project1\x64\Debug\Project1.exe : fatal error LNK1169: 找到一个或多个多重定义的符号
1>已完成生成项目 "Project1.vcxproj" 的操作 - 失败。
===== 生成: 0 成功, 1 失败, 0 最新, 0 已跳过 =====
===== 生成 开始于 10:28, 并花费了 02.442 秒 =====
|
```

2) 将上述函数替换为以下函数, 再次尝试编译运行, 对结果进行截图。

结果: 编译成功。



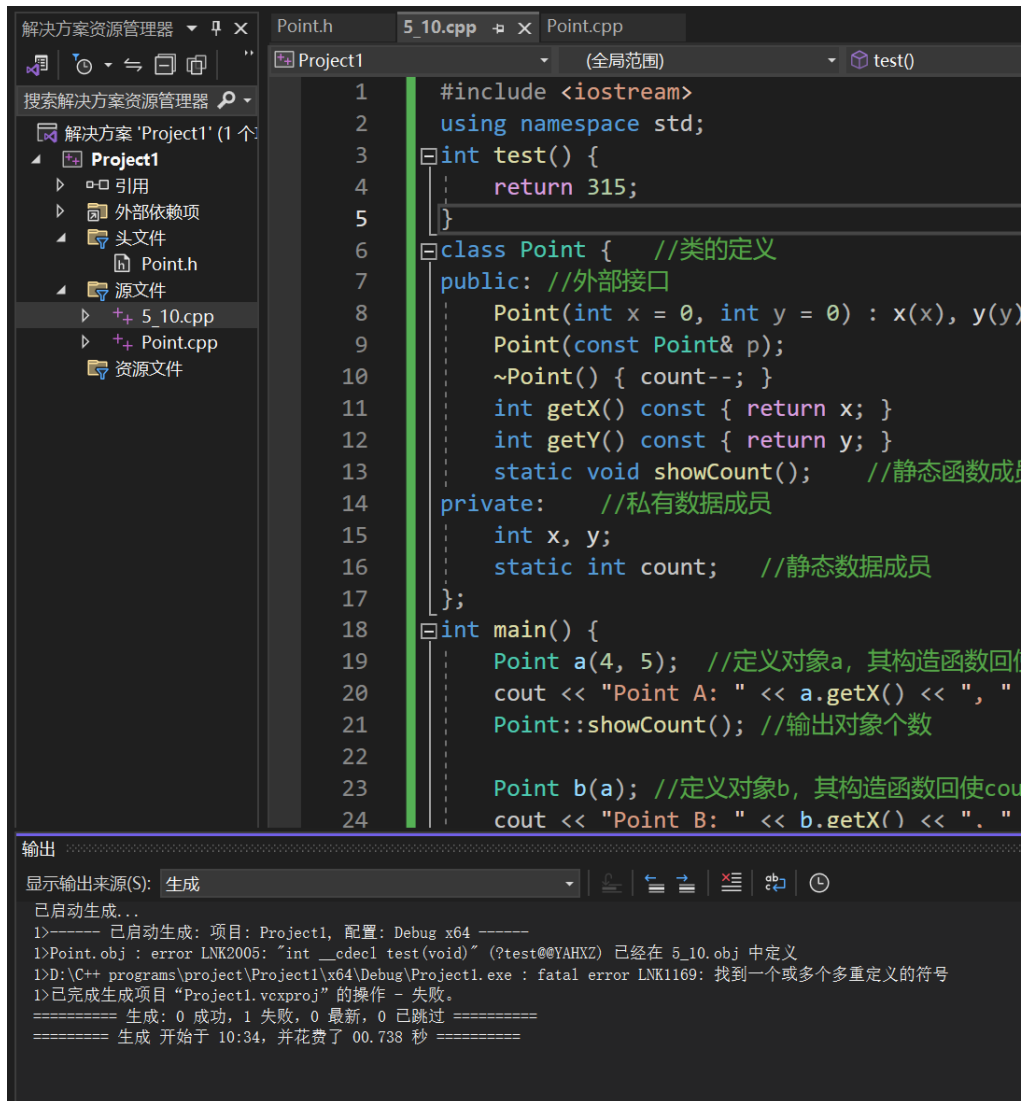
输出

显示输出来源(S): 生成

```
已启动生成...
1>----- 已启动生成: 项目: Project1, 配置: Debug x64 -----
1>5_10.cpp
1>Point.cpp
1>正在生成代码...
1>Project1.vcxproj -> D:\C++ programs\project\Project1\x64\Debug\Project1.exe
===== 生成: 1 成功, 0 失败, 0 最新, 0 已跳过 =====
===== 生成 开始于 10:29, 并花费了 02.833 秒 =====
```

(3) 结合课上讲的#include 的作用是将相应的文件内容嵌入到该文件中这一现实, 尝试解释上述两者的不同。可以尝试用 Point.h 里面的内容替换掉#include "Point.h" 语句, 然后重新测试上述两个题目, 应该更能感受inline 函数和普通函数的不同, 以及同一个文件被不同的文件 include 所产生的影响。

替换 point.cpp 后完成 (1), 编译错误:



```

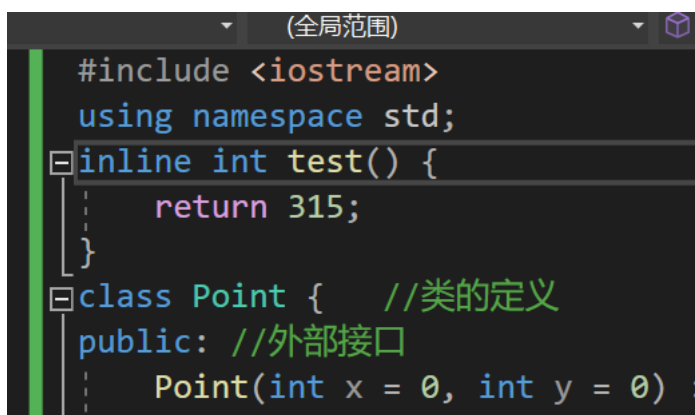
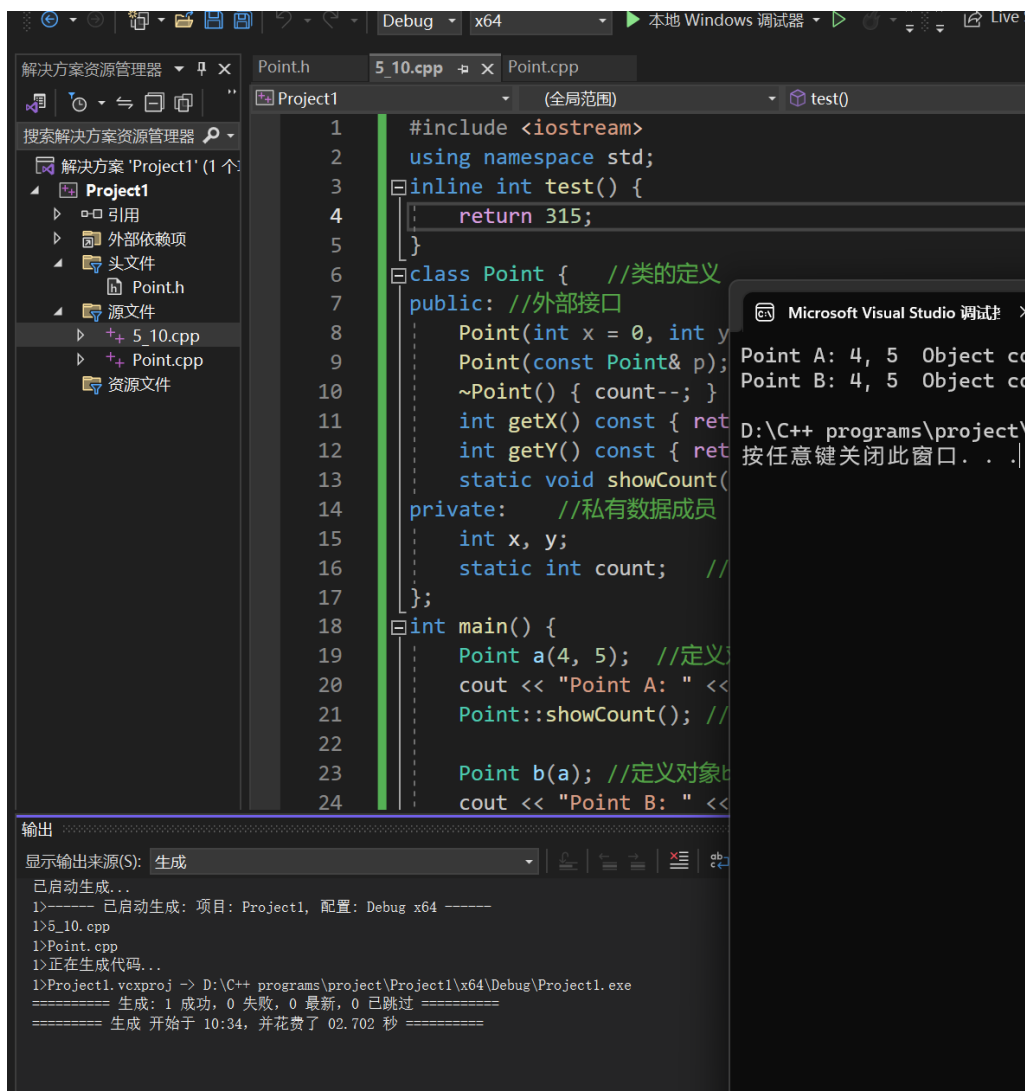
#include <iostream>
using namespace std;
int test() {
    return 315;
}
class Point {    //类的定义
public: //外部接口
    Point(int x = 0, int y = 0) : x(x), y(y) {}
    Point(const Point& p);
    ~Point() { count--; }
    int getX() const { return x; }
    int getY() const { return y; }
    static void showCount();    //静态函数
private:    //私有数据成员
    int x, y;
    static int count;    //静态数据成员
};
int Point::count = 0;    //使用类名初始化静态数据成员

Point::Point(const Point& p) : x(p.x), y(p.y) {
    count++;
}

void Point::showCount() {
    cout << "    Object count = " << count << endl;
}

```

执行 (2)，成功编译：



(4) 如果在 Point.cpp 中加入以下函数的定义语句，如何在 5_10.cpp 中调用该函数？
加 extern 关键字。


```
Point.h  Point.cpp
1  #include <iostream>
2  using namespace std;
3  extern int test() {
4      return 315;
5  }
6  class Point {    //类的定义
7  public: //外部接口
```

```
5_10.cpp  Point.h  Point.cpp
Project1  (全局范围)  test()
13  static int count; //静态数据成员
14  };
15  extern int test();
16  int main() {
17      Point a(4, 5); //定义对象a, 其构造函数
18      cout << "Point A: " << a.getX() << endl;
19      Point::showCount(); //输出对象个数
20
21      Point b(a); //定义对象b, 其构造函数
22      cout << "Point B: " << b.getX() << endl;
23      Point::showCount(); //输出对象个数
24      cout << test();
25      return 0;
26  }
27
```

Microsoft Visual S
Point A: 4, 5
Point B: 4, 5
315
D:\C++ program
按任意键关闭此

(1) 分别在 Point.cpp 和 5_10.cpp 中定义一个全局变量 `int val = 315;` 并在 main 函数中打印该值, 然后进行编译, 观察编译运行结果。

```
1>----- 已启动生成: 项目: Project1, 配置: Debug x64 -----
1>5_10.cpp
1>Point.cpp
1>正在生成代码...
1>LINK : 没有找到 D:\C++ programs\project\Project1\x64\Debug\Project1.exe 或上一个增量链接没有生成它; 正在执行完全链接
1>Point.obj : error LNK2005: "int val" (?val@03HA) 已经在 5_10.obj 中定义
1>D:\C++ programs\project\Project1\x64\Debug\Project1.exe : fatal error LNK1169: 找到一个或多个多重定义的符号
1>已完成生成项目 "Project1.vcxproj" 的操作 - 失败。
===== 生成: 0 成功, 1 失败, 0 最新, 0 已跳过 =====
===== 生成 开始于 10:43, 并花费了 03.037 秒 =====
|
```

(2) 在 Point.cpp 中定义全局变量 `int val = 315;`, 之后在 5_10.cpp 中通过 `extern int val;` 定义该变量为外部变量。在 main 函数中打印该值, 然后进行编译, 观察编译并解释运行结果。

```
1 #include <iostream>
2 #include "Point.h"
3 using namespace std;
4 extern int val;
5 int main() {
6     Point a(4, 5); //定义对象
7     cout << "Point A: " <<
8     Point::showCount(); //输出
9 }
```

Microsoft Visual Studio 调试

Point A: 4, 5 Object count = 1
Point B: 4, 5 Object count = 2
315
D:\C++ programs\project\Project1
按任意键关闭此窗口. . .

```
Point.h
Point.cpp

1 #include <iostream>
2 #include "Point.h"
3 using namespace std;
4 int Point::count = 0; //使用类名初始化
5
6 Point::Point(const Point& p) : x(p.x)
7 {
8     count++;
9 }
10 int val = 315;
11 void Point::showCount() {
```

结果: val 由 Point.cpp 的赋值 extern 挂载到了 5_10.cpp 中, 输出 315.

(4) 将以下实验三中的抛骰子的代码通过多文件项目进行实现。该项目至少包含三个文件: dice.h 对 Dice 类及 GameStatus 进行了定义。Dice.cpp 对 Dice 的各个函数进行了实现, main.cpp 函数定义了 main 函数实现了对类的实例化和相应函数的调用。请将每个文件截图。运行后的结果也需要截图。

结果&& main.cpp:

```
main.cpp dice.cpp dice.h

1 #include "dice.h"
2 #include <iostream>
3 using namespace std;
4 int main()
5 {
6     unsigned seed;
7     cout << "Please enter an unsigned integer: ";
8     cin >> seed; //输入随机数种子
9     Dice dice(seed);
10    dice.play();
11    return 0;
12 }
```

Microsoft Visual Studio 调试

Please enter an unsigned integer: 2
player rolled 2 + 2 = 4
player rolled 4 + 2 = 6
player rolled 6 + 5 = 11
player wins
D:\C++ programs\project\Dice\main.cpp
按任意键关闭此窗口. . .

Dice.cpp:

解决方案资源管理器

- 解决方案 'Dice' (1 个项目)
 - Dice
 - 引用
 - 外部依赖项
 - 头文件
 - dice.h
 - 源文件
 - dice.cpp
 - main.cpp
 - 资源文件

```

1  #include "dice.h"
2  #include <iostream>
3  using namespace std;
4  enum GameStatus { WIN, LOSE, PLAYING };
5  Dice::Dice(int seed) : seed(seed)
6  {
7      srand(seed);
8  }
9  int Dice::rollDice()
10 {
11     int die1 = 1 + rand() % 6;
12     int die2 = 1 + rand() % 6;
13     int sum = die1 + die2;
14     cout << "player rolled " << die1 <<
15     return sum;
16 }
17 void Dice::play() {
18     GameStatus status = PLAYING;
19     while (status == PLAYING) { //只要状态为PLAYING
20         int sum = rollDice(); //第一轮投骰子
21         switch (sum) {
22             case 7: //如果和数为7或11则为胜利
23             case 11:
24                 status = WIN;
25                 break;
26             case 2: //和数为2、3或12则为负,游戏结束
27             case 3:
28             case 12:
29                 status = LOSE;
30                 break;
31             default: /*其它情况,游戏尚无结果,继续游戏,
32                      PLAYING,记下点数,为下一轮做准备*/
33                 status = PLAYING;
34                 break;
35         }
36     }
37 }

```

Dice.h:

(全局范围)

```

1  #pragma once
2  class Dice {
3      int seed;
4  public:
5      Dice(int seed);
6      int rollDice();
7      void play();
8  };
9

```

结论分析与体会：

非常好的实验，使我重新认识了多文件项目构建和 Vs 2022 的伟大。
同时我还学到了掌握多文件项目构建，熟悉编译预处理指令。