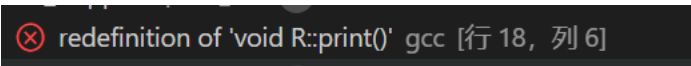



计算机学院 高级语言程序设计 课程实验报告

实验题目：数据共享与保护（二），多文件项目， 预编译；对象数组		学号：202200400053
日期：2024-03-28	班级：2202	姓名：王宇涵
Email：1941497679@qq.com		
实验目的： 1. 数据共享与保护： 练习 const: 常对象、常成员、常引用 2. 掌握多文件项目构建 3. 熟悉编译预处理指令 4. 练习对象数组		
实验软件和硬件环境： 软件环境：VSCODE + DEV-C++ 硬件环境：Legion Y7000P		
实验步骤与内容： (1) 常对象： 练习课本例 5-7.cpp, 讨论去掉 <code>void R::print() const{</code> 语句行中的 <code>const</code> 可以吗？为什么？ 在 R 类中增加一个成员函数 <code>change()</code> 如下，尝试用 a 对象和 b 对象分别调用它可以吗？为什么？ <pre>change(int x1,int x2){ r1=x1; r2=x2; }</pre>  <p>答：不可以，因为如果去掉了 <code>const</code>，则与已经实现的不带 <code>const</code> 的 <code>print</code> 函数冲突，函数重定义编译错误，则调用常对象的函数无法识别调用哪一个 <code>print</code> 函数</p> (2) 常成员： 练习课本例 5-8.cpp, 讨论将 <code>A::A(int i):a(i){}</code> 语句改为 <code>A::A(int i):{a=i;}</code> 可以吗？为什么？其中的 <code>static const int b</code> ；能在类内初始化吗？  <p>答：不能。因为常对象 a 只能通过列表初始化，如果写成 <code>a=i</code> 则是赋值的形式，不符合常对象无法被修改的定义。Static const int b 经过测试，可以在类内初始化。</p> 3) 常引用： 练习课本例 5-9.cpp, 尝试将友元函数 <code>dist(const Point &p1, const Point &p2)</code> 的两处声明和类外定义的形参都取消 <code>const</code> 可以吗？为什么？ 尝试将 <code>main()</code> 函数中 <code>const Point myp1(1, 1), myp2(4, 5);</code> 语句中的 <code>const</code> 取消，程序能正常运行吗？为什么？		

解释引用 2.cpp 中的问题。

💡 将 "Point &" 类型的引用绑定到 "const Point" 类型的初始值设定项时，限定符被丢弃 C/C++(433) [行 25, 列 18]

答：不能取消 const，因为测试案例中 myp1 和 myp2 均为 const 类型，若形参不声明为 const，则会编译错误

若把 main() 函数中的 const 取消，程序可以正常运行，因为形参为 const 类型，代表不能修改传入的变量的值，但是可以传入变量。

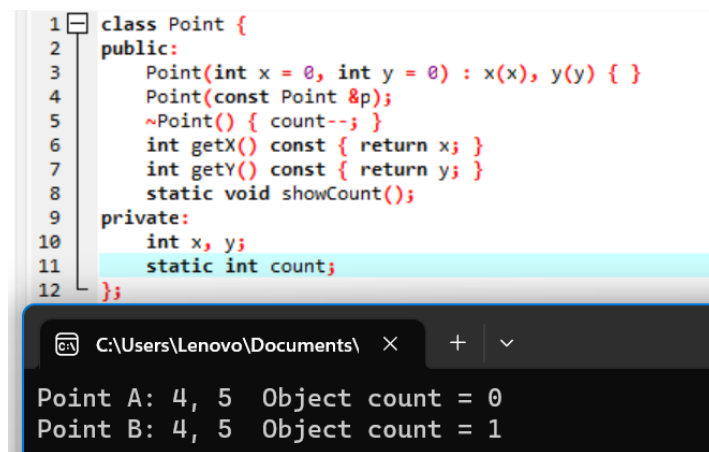
解释引用 2.cpp：用 a2 可以是因为 a2 和 a1 是同类型对象，可以直接访问私有变量。用 b 不可以因为 b 和 a1 是不同类对象，不可以直接访问私有变量

4) 多文件项目练习

运行第 5 章实验 5，实验任务（2）lab5_2 项目。写出分析和体会。

参考附件：DEV-C++ 中多文件项目操作.txt;

参考附件：

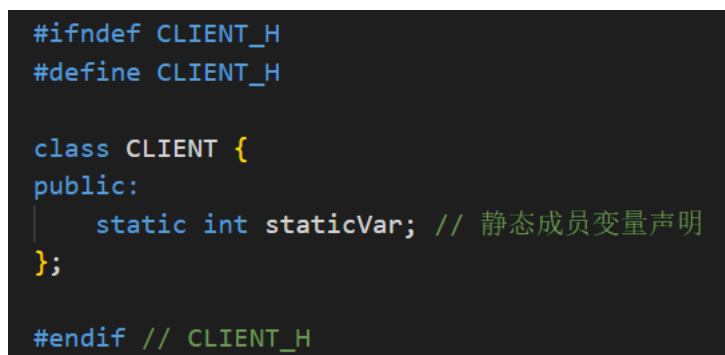


```
1 class Point {
2 public:
3     Point(int x = 0, int y = 0) : x(x), y(y) { }
4     Point(const Point &p);
5     ~Point() { count--; }
6     int getX() const { return x; }
7     int getY() const { return y; }
8     static void showCount();
9 private:
10    int x, y;
11    static int count;
12 };

Point A: 4, 5  Object count = 0
Point B: 4, 5  Object count = 1
```

Lab5_2:

Client.h

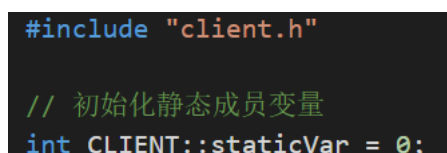


```
#ifndef CLIENT_H
#define CLIENT_H

class CLIENT {
public:
    static int staticVar; // 静态成员变量声明
};

#endif // CLIENT_H
```

Client.cpp



```
#include "client.h"

// 初始化静态成员变量
int CLIENT::staticVar = 0;
```

Main.cpp

```
#include <iostream>
#include "client.h"
#include "client.cpp"

int main() {
    std::cout << "Initial staticVar value: " << CLIENT::staticVar << std::endl;

    // 修改静态成员变量的值
    CLIENT::staticVar = 5;

    std::cout << "Modified staticVar value: " << CLIENT::staticVar << std::endl;

    return 0;
}
```

输出结果

```
Initial staticVar value: 0
Modified staticVar value: 5
```

结论分析 :h 头文件也可以不加在项目中, 因为.h 头文件是被#include 组合进其它文件编译, 本身并不作为独立的编译单元, 因此我们最好将.h 头文件和函数的定义分开实现, 不要都放在同一头文件里.

5)编译预处理:

(1) 分析 c5.zip 文件中的多文件编译出现的问题, 如何解决?

(2) 参照所给的《DEV-C++中多文件项目操作》, 针对 c5_1.zip 中的多文件, 练习类声明、成员函数、主调程序分离。

讨论: 为什么一个项目多个文件中都有#include <iostream>, 而不担心重复?

探索所用环境的 C++标准 include 及 library 所在位置。探索在 vs code 环境下多文件项目编译。

答(1) 修改 head.h 文件为

```
#ifndef HEAD_H
#define HEAD_H
class Point
{
};
#endif
```

即可保证仅编译一次

(2)练习输出结果

```
num:101
name:zhangsan
sex:m
```

讨论答案: 在 C++中, #include <iostream> 是一个预处理指令, 用于将标准输入输出流库引入到程

序中。当多个文件中都包含了 `#include <iostream>`，编译器会对这些文件进行预处理，但实际上只会包含一次 `<iostream>` 的内容。这是因为预处理器会在编译过程中对重复包含的标准库进行防护，以避免重复定义和编译错误。

探索所用环境的 C++ 标准 include 及 library 所在位置。

```
{
  "configurations": [
    {
      "name": "Win32",
      "includePath": [
        "${workspaceFolder}/**"
      ],
      "defines": [
        "_DEBUG",
        "UNICODE",
        "_UNICODE"
      ],
      "compilerPath": "D:\\fix\\mingw64\\bin\\gcc.exe",
      "cStandard": "c11",
      "cppStandard": "gnu++14",
      "intelliSenseMode": "windows-gcc-x64"
    }
  ],
  "version": 4
}
```

则 include 包含当前工作区中的所有文件和文件夹，编译路径为 `D:\\fix\\mingw64\\bin\\gcc.exe`

6) 综合项目:

练习课本 5.7 中，个人银行账户管理程序新加的内容，即例题 5_11。（涉及 `static`, `const`, 多文件）

输出结果:

```
1      #21325302 is created
1      #58320212 is created
5      #21325302      5000      5000
25     #58320212      10000     10000
45     #21325302      5500      10500
60     #58320212      -4000     6000
90     #21325302      27.64     10527.6
90     #58320212      21.78     6021.78
#21325302      Balance: 10527.6
#58320212      Balance: 6021.78
Total: 16549.4
```

Static：定义了所有账户的总金额 total，通过静态函数成员 `getTotal` 调用

Const：定义常成员变量和常函数，后者不能改变成员变量的值

多文件编译输出正确结果

7) 第 6 章实验 6，实验任务（1）lab6_1。运用面向对象的思想，该程序还可以怎样设计？

Lab6_1:

```

int main() {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cin >> a[i][j];
        }
    }

    for (int i = 0; i < 3; i++) {
        for (int j = i + 1; j < 3; j++) {
            swap(a[i][j], a[j][i]);
        }
    }

    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << a[i][j] << " ";
        }
        cout << endl;
    }
}

```

设计方法：将矩阵包装成矩阵类，含有成员函数 transpose() 转置

```

class Matrix {
private:
    int data[3][3];
public:
    // 构造函数
    Matrix(int initData[3][3]) {
        for (int i = 0; i < 3; ++i) {
            for (int j = 0; j < 3; ++j) {
                data[i][j] = initData[i][j];
            }
        }
    }

    // 矩阵转置函数
    void transpose() {
        int temp;
        for (int i = 0; i < 3; ++i) {
            for (int j = i + 1; j < 3; ++j) {
                // 交换元素
                temp = data[i][j];
                data[i][j] = data[j][i];
                data[j][i] = temp;
            }
        }
    }
}

```

结论分析与体会：

本次实验我通过编写代码和阅读代码，深入理解并掌握了数据共享与保护中 `const`：常对象、常成员、常引用的概念，掌握多文件项目构建的设计思想，加深了对编译预处理指令作用的认识，通过练习对象数组也意识到封装的重要性。

就实验过程中遇到的问题及解决处理方法，自拟 1—3 道问答题：

1. 如何解决头文件重复包含的问题？

答：在头文件中加入预编译指令 `ifndef .. #define .. #ifdef` 可以保证仅编译一次

2. 如何解决 `#include <iostream>` 等 c++ 库头文件重复包含的问题？

答：编译器会自动进行优化，因此不需要关注。

3. 如何初始化类内静态成员变量？

答：优先类外初始化。