|  |  |
| --- | --- |
| **组号:** | **6** |

****

信息科学与工程学院课程实验报告

《面向对象程序设计》

|  |  |
| --- | --- |
| 姓名： | 郑彤 |
| 学号： | 201711990140 |
| 班级： | 计师本1701 |
| 教师： | 张庆科 |
| 时间： | 2018.11.22 |

**面向对象程序设计实验报告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 郑彤 | 班级 | 计师本1701 | 学号 | 201711990140 | 组号 | 6 |
| 时间 | 2018.11.22 | 地点 |  | 周次 |  | 页码 |  |
| 源码 | □ 无源码 □ 文档源码 □ 托管源码 | | | | | | |
| 报  告  内  容  报  告  内  容  报  告  内  容 | 实验报告要求：请围绕实验目的、实验内容、实验过程及步骤(可添加文字、矢量图)、实验结论与分析进行撰写。凡涉及源代码内容可给出完整源码并附上源码Github托管网址（请务必按照条目书写）。  **第四章 类中数据的共享和保护**  4.1.1 静态数据成员  设置静态成员这种机制的目的是将和某些类紧密相关的全局变量和函数写到类里面，看上去像一个整体，易于维护和理解。  静态成员 ——身在曹营心在汉  静态数据成员：在说明前面加了static关键字的成员（包括静态数据成员和静态成员函数）  ·静态数据成员：被static关键字修饰，本质上为类的全局成员  ·静态成员函数：管理类内私有的静态数据成员，对类外提供访问接口  ·声明：静态成员数据必须在类内通过static进行声明  static 类型名 静态数据成员名;  ·初始化：静态数据成员的初始化必须在类外面进行，默认值为0  类型名 类名::静态数据成员名=初值;  静态数据成员本质：静态数据成员本质上为封装在类内的全局变量，因此这些静态数据相对普通全局变量更安全。  本质推理：采用sizeof运算符不会计算静态成员变量占用空间，因此可以看出类内的静态成员变量并不是真正意义上类的成员，而是封装在类内的全局变量，以便看上去像一个整体，易于维护和理解。  如何访问到静态数据成员？——通过类或类的对象进行访问  公有的静态数据成员可以通过类或对象进行直接访问，形式为：  类名::公有静态成员变量名;  对象名.公有静态成员变量名;  私有的静态数据成员只能通过类的静态成员函数进行间接访问  4.1.2 静态成员函数  为什么要设计静态成员函数——如果静态数据成员设置为私有的，那么类和对象无法直接访问这个私有成员，必须通过公有的函数来访问。如果普通函数是作用于对象的，不是类的，故普通成员函数无法满足这个需求，C++针对该情况重新设计一个特殊的公有的成员函数，称为静态成员函数，这个函数同样是属于整个类的，不属于任何对象。  本质剖析：类内的成员函数不是真正意义上类的成员函数  ，而是封装在类内的全局函数，通过该函数完成对全局变量的管理。这样就和类的函数类的成员构成形式上的一致性。它仍然不属于任何对象。  静态成员函数的声明与定义：静态成员函数与类内的普通成员函数声明形式不同，需要通过static关键字修饰函数，但是静态成员函数的定义形式与普通函数完全相同。  ·静态成员函数原型声明格式（static修饰）：  Static 返回类型 函数名（形参）；  ·静态成员函数定义格式（无须static修饰）：  Static 返回类型 函数名（形参）{ }；  注意：静态成员函数属于类不属于对象，它只能访问类的静态数据成员，不能对类的普通成员变量或成员函数进行访问。  静态成员函数访问的方法：   1. 类名::成员名   Date::PrintTotal();   1. 对象名::成员名   Date r;  r.PrintTotal ();   1. 指针->成员名   Date \*p=&r;  p->PrintToal ();   1. 引用.成员名   Date &ref=r;  Int n=ref.PrintTotal ();  4.2.1 常数据成员  数据保护——const  保护的类型：类的数据成员、类的成员函数、类的对象  4.2.2 常成员函数——只读函数  如果一个成员函数对类中数据成员只作访问而不作直接或间接的修改，则最好将此成员函数说明为常成员函数，以明确表示它对数据成员的保护性  注意事项：常成员函数不能调用普通函数；普通成员函数可以调用常成员函数  4.2.3 常对象  常对象在整个生存期内都不能被改  ·常数据成员：类内的某一数据成员函数不能被修改  定义形式 const 类型名 常数据成员名；  初始化形式 构造函数首部： 常数据成员名（常量值或表达式）  ·常成员函数：类的成员函数只能访问类内的其他成员而不允许修改  声明格式 类型名 函数名（形参） const;  定义格式 类型名 函数名（形参） const  {  }  ·常对象：整个对象不能被修改  说明格 类名 const 对象名；或 const 类名 对象名；  01 *//example5\_01.cpp:֔ϳӉԱՄٹԬӫζٹ*  02  03 #include<iostream>  04 **using namespace** std;  05  06 **class** A  07 {  08 **public**:  09 **A**()  10 {  11 cout << "ԴݨA" << endl;  12 }  13  14 ~**A**()  15 {  16 cout << "ζٹA" << endl;  17 }  18 };  19  20 **class** B  21 {  22 **public**:  23 **B**()  24 {  25 cout << "ԴݨB" << endl;  26  27 }  28  29 ~**B**()  30 {  31 cout << "ζٹB" << endl;  32 }  33  34 **private**:  35 A a; *//֔ϳӉԱìB мڬKA*  36 };  37  38 **int** **main**()  39 {  40 B obj;  41  42 **system**("pause");  43 **return** 0;  44 }  E:\QQ\消息记录\2358080893\Image\C2C\M_@Y}ME9%NE]8Z~5SS`DYIY.png  01 *//example5\_02.cpp:֔ϳӉԱٹԬگʽҎʽՄԫ՝*  02  03 #include<iostream>  04 **using namespace** std;  05  06 **class** A  07 {  08 **public**:  09 **A**(**int** a)  10 {  11 x = a;  12 cout << "A:x=" << a << endl;  13  14 }  15  16 **private**:  17 **int** x;  18  19 };  20  21 **class** B  22 {  23 **public**:  24 **B**(**int** a, **int** b, **int** c):**a2**(b),**a1**(a) *//עҢúӵʼۯPѭ֐ʇӉԱ֔ϳûa1,a2ìָһʇ ûA*  25 {  26 y = c;  27 cout << "B:y=" << c << endl;  28 }  29  30 **private**:  31 A a1, a2;  32 **int** y;  33  34 };  35  36 **int** **main**()  37 {  38 B **obj**(10, 20, 30);  39 **system**("pause");  40 **return** 0;  41 }  E:\QQ\消息记录\2358080893\Image\C2C\E7D9@(LD1B()G0TR%{R~VRA.png  01 *//example5\_03\_1.cpp:Ӄ ՄҀ5ʵϖ뗓*  02  03 #include<iostream>  04 #include<ctime>  05 #include<cstdlib>  06 **using namespace** std;  07  08 **class** Dice *//뗓*  09 {  10 **public**:  11 **int** **run**()  12 {  13 **return** (**rand**() % 6 + 1);  14 }  15  16 };  17  18 **class** Gambler *//քͽ*  19 {  20 **public**:  21 **void** **play**(Dice a, Dice b, Dice c)  22 {  23 **int** num1, num2, num3;  24 num1 = a.**run**();  25 num2 = b.**run**();  26 num3 = c.**run**();  27 **if** (num1 == num2&&num1 == num3)  28 cout << num1 << " " << num2 << " " << num3 << "Ѫד" << endl;  29 **else if** (num1 + num2 + num3 > 10)  30 cout << num1 << " " << num2 << " " << num3 << "Գ" << endl;  31 **else**  32 cout << num1 << " " << num2 << " " << num3 << "С" << endl;  33 }  34  35 };  36  37 **int** **main**()  38 {  39 Dice d1, d2, d3;  40 Gambler Jack;  41 **srand**(**time**(NULL));  42 Jack.**play**(d1, d2, d3);  43 **system**("pause");  44 **return** 0;  45 }  E:\QQ\消息记录\2358080893\Image\C2C\394%{X}GMF(ETH~PCC()7OT.png  01 *//example5\_03\_2.cpp:Ӄ Մשڏ٘ϵʵϖ뗓*  02  03 #include<iostream>  04 #include<ctime>  05 #include<cstdlib>  06 **using namespace** std;  07  08 **class** Dice *//뗓*  09 {  10 **public**:  11 **int** **run**()  12 {  13 **return** (**rand**() % 6 + 1);  14 }  15 };  16  17 **class** Gambler *//քͽ*  18 {  19 **public**:  20 **void** **play**()  21 {  22 **int** num1, num2, num3;  23 num1 = a.**run**();  24 num2 = b.**run**();  25 num3 = c.**run**();  26 **if** (num1==num2&&num1==num3)  27 cout << num1 << " " << num2 << " " << num3 << "Ѫד" << endl;  28 **else if** (num1+num2+num3>10)  29 cout << num1 << " " << num2 << " " << num3 << "Գ" << endl;  30 **else**  31 cout << num1 << " " << num2 << " " << num3 << "С" << endl;  32 }  33  34 **private**:  35 Dice a, b, c;  36 };  37  38 **int** **main**()  39 {  40 Gambler Jack;  41 **srand**(**time**(NULL));  42 Jack.**play**();  43 **system**("pause");  44 **return** 0;  45 }  E:\QQ\消息记录\2358080893\Image\C2C\D28`ZMOO@)C(U$182XL@%{3.png | | | | | | |

：可根据内容自行拓展页面，作业内容尾部尽量不要留有空白