

**实 验 报 告**

**（ 2023 / 2024 学年 第 二 学期）**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | 面向对象程序设计及C++ | | | | | |
| 实验名称 | 实验一： 类和对象的定义及使用 | | | | | |
| 实验时间 | 2024 | 年 |  | 月 |  | 日 |
| 指导单位 | 计算机学院、软件学院、网络空间安全学院 | | | | | |
| 指导教师 |  | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 学生姓名 |  | 班级学号 |  |
| 学院(系) |  | 专 业 |  |

| **实 验 报 告** | | | |
| --- | --- | --- | --- |
| **实验名称** | **类和对象的定义及使用** | | |
| **实验类型** | **验证** | **实验学时** | **2** |
| 1. **实验目的和要求**   （1）掌握类与对象的定义与使用方法，理解面向对象方法中通过对象间传递消息的工作机制。  （2）正确掌握类的不同属性成员的使用方法。  （3）掌握构造函数与析构函数的概念，理解构造函数与析构函数的执行过程。  （4）掌握友元函数和友元类的定义和使用。  （5）基本掌握指针和引用作为函数参数的应用。 | | | |
| 二、**实验环境(实验设备)**  硬件: 微型计算机  软件: Windows 操作系统、Microsoft Visual Studio 2010 | | | |
| **三、实验原理及内容**  **实验题目1** 定义一个借书证类BookCard，在该类定义中包括如下内容。  （1）私有数据成员：  string id; //借书证学生的学号  string stuName; //借书证学生的姓名  int number; //所借书的数量  （2）公有成员函数：  构造函数 //用来初始化3 个数据成员，是否带默认参数值参考结果来分析  void display() //显示借书证的3 个数据成员的信息  bool borrow() //已借书数量不足10 则将数量加1，数量达到10 则直接返回false  主函数及f()函数代码如下。请结合输出结果完成程序。  void f(BookCard &bk)  {  if (!bk.borrow())  {  bk.display();  cout<<"you have borrowed 10 books,can not borrow any more!"<<endl;  }  else  bk.display();  }  int main()  {  BookCard bk1("B20190620","东平",10),bk2;  f(bk1);  f(bk2);  return 0;  }  程序的运行结果为：  B20190620 东平 10  you have borrowed 10 books,can not borrow any more!  B19010250 雪峰 4  **参考实验教材中相应的实验指导，完成源程序代码如下：（**中文五号宋体，英文五号Consolas字体，单倍行距)  #include<iostream>  #include <cstring>  using namespace std;  class bookCard  {  private:  const string id;  const string stuName;  int number;  public:  bookCard(): id("null"), stuName("null"), number(11){}  bookCard(string id, string stuName, int number): id(id), stuName(stuName), number(number){}  void display()  {  cout << id << " " << stuName << " " << number << endl;  }  bool borrow()  {  if (number > 10)  {  return false;  }  else  {  number++;  }  }  };  void f(bookCard &bk)  {  if (!bk.borrow())  {  cout << "you have borrowed 10 books and you can not borrow any more!" << endl;  }  else  bk.display();  }  int main()  {  bookCard bk1("B20190620", "东平", 10), bk2, bk3("B19010250", "雪峰", 4);  f(bk1);  f(bk2);  f(bk3);  return 0;  }  **实验题目2** 定义一个时间类Time，有三个私有成员变量Hour、Minute、Second，定义构造函数、析构函数以及用于改变、获取、输出时间信息的公有函数，主函数中定义时间对象，并通过调用各种成员函数完成时间的设定、改变、获取、输出等功能。  ① 按要求完成类的定义与实现。  ② 修改数据成员的访问方式，观察编译结果。  ③ 在Time类中定义一个成员函数，用于实现时间增加一秒的功能，主函数中通过对象调用该函数，并输出增加一秒后的时间信息。  ④ 定义一个普通函数。  void f(Time t)  { t. PrintTime( );  }  在Time类中增加拷贝构造函数的定义，主函数中调用该函数，运用调试工具跟踪，分析整个程序调用构造函数（包括拷贝构造函数）和析构函数的次数；再将f函数的形式参数分别修改为引用参数和指针参数（此时函数代码修改为{t-> PrintTime( );},主函数中调用，再分析此时调用构造函数和析构函数的次数。 | | | |
| **参考实验教材中相应的实验指导完成程序，并回答相关问题。完成后的源程序代码如下：（**中文五号宋体，英文五号Consolas字体，单倍行距)  #define \_CRT\_SECURE\_NO\_WARNINGS  #include<iostream>  #include <cstring>  using namespace std;  class Time  {  private:  int hour;  int minute;  int second;  public:  Time(int hour = 0, int minute = 0, int second = 0): hour(hour), minute(minute), second(second)  {  cout << "Constructing..." << endl;  ;  }  Time(const Time& ob)  {  cout << "Copy constructing..." << endl;  ;  }  ~Time()  {  cout << "Destructing..." << endl;  ;  }  void changeTime(int hour = 0, int minute = 0, int second = 0)  {  this->hour = hour;  this->minute = minute;  this->second = second;  }  int getHour()  {  return hour;  }  int getMinute()  {  return minute;  }  int getSecond()  {  return second;  }  void printTime()  {  cout<<hour << ":" << minute << ":" << second << endl;  }  void increaseOneSecond()  {  if (second == 59)  {  if (minute == 59)  {  if (hour == 23)  {  hour = 0;  minute = 0;  second = 0;  }  else  {  hour++;  minute = 0;  second = 0;  }  }  else  {  minute++;  second = 0;  }  }  else  {  second++;  }  }  };  void f(Time t)  {  t.printTime();  }  int main()  {  Time time1, time2(23), time3(23, 59), time4(23, 59, 59);  time1.printTime();  time2.printTime();  time3.printTime();  time4.printTime();  time1.changeTime();  time2.changeTime(23);  time3.changeTime(23, 59);  time4.changeTime(23, 59, 59);  time1.printTime();  time2.printTime();  time3.printTime();  time4.printTime();  time1.increaseOneSecond();  time2.increaseOneSecond();  time3.increaseOneSecond();  time4.increaseOneSecond();  cout << time1.getHour() << ":" << time1.getMinute() << ":" << time1.getSecond() << endl;  cout << time2.getHour() << ":" << time2.getMinute() << ":" << time2.getSecond() << endl;  cout << time3.getHour() << ":" << time3.getMinute() << ":" << time3.getSecond() << endl;  cout << time4.getHour() << ":" << time4.getMinute() << ":" << time4.getSecond() << endl;  return 0;  }  **程序的运行结果是：**  **0:0:0**  **23:0:0**  **23:59:0**  **23:59:59**  **0:0:0**  **23:0:0**  **23:59:0**  **23:59:59**  **构造函数与析构函数的调用方式及执行顺序是：**  构造函数在对象创建的时候自动调用，用于初始化对象的成员变量，在同一个类中，成员对象的构造函数按照它们在类中定义的声明顺序执行，而不是它们在构造函数初始化列表中的顺序；析构函数在对象销毁时自动调用，在同一个类中，成员对象的析构函数按照它们构造的逆序执行。  ③取消类中成员函数IncreaceOneSecond( )的注释标志，将该函数补充完整，注意时间在增加一秒情况下的进位关系。  **该函数的代码如下：（**中文五号宋体，英文五号Consolas字体，单倍行距)  **#define \_CRT\_SECURE\_NO\_WARNINGS**  **#include<iostream>**  **#include <cstring>**  **using namespace std;**  **class Time**  **{**  **private:**  **int hour;**  **int minute;**  **int second;**  **public:**  **Time(int hour = 0, int minute = 0, int second = 0): hour(hour), minute(minute), second(second)**  **{**  **cout << "Constructing..." << endl;**  **;**  **}**  **Time(const Time& ob)**  **{**  **cout << "Copy constructing..." << endl;**  **;**  **}**  **~Time()**  **{**  **cout << "Destructing..." << endl;**  **;**  **}**  **void changeTime(int hour = 0, int minute = 0, int second = 0)**  **{**  **this->hour = hour;**  **this->minute = minute;**  **this->second = second;**  **}**  **int getHour()**  **{**  **return hour;**  **}**  **int getMinute()**  **{**  **return minute;**  **}**  **int getSecond()**  **{**  **return second;**  **}**  **void printTime()**  **{**  **cout<<hour << ":" << minute << ":" << second << endl;**  **}**  **void increaseOneSecond()**  **{**  **if (second == 59)**  **{**  **if (minute == 59)**  **{**  **if (hour == 23)**  **{**  **hour = 0;**  **minute = 0;**  **second = 0;**  **}**  **else**  **{**  **hour++;**  **minute = 0;**  **second = 0;**  **}**  **}**  **else**  **{**  **minute++;**  **second = 0;**  **}**  **}**  **else**  **{**  **second++;**  **}**  **}**  **};**  **void f(Time t)**  **{**  **t.printTime();**  **}**  **int main()**  **{**  **Time time1, time2(23), time3(23, 59), time4(23, 59, 59);**  **time1.printTime();**  **time2.printTime();**  **time3.printTime();**  **time4.printTime();**  **time1.changeTime();**  **time2.changeTime(23);**  **time3.changeTime(23, 59);**  **time4.changeTime(23, 59, 59);**  **time1.printTime();**  **time2.printTime();**  **time3.printTime();**  **time4.printTime();**  **time1.increaseOneSecond();**  **time2.increaseOneSecond();**  **time3.increaseOneSecond();**  **time4.increaseOneSecond();**  **cout << time1.getHour() << ":" << time1.getMinute() << ":" << time1.getSecond() << endl;**  **cout << time2.getHour() << ":" << time2.getMinute() << ":" << time2.getSecond() << endl;**  **cout << time3.getHour() << ":" << time3.getMinute() << ":" << time3.getSecond() << endl;**  **cout << time4.getHour() << ":" << time4.getMinute() << ":" << time4.getSecond() << endl;**  **return 0;**  **}**  **④** 主函数中定义一个Time类对象并调用一次f函数，观察结果填写下表：   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **f函数的原型** | **主函数中调用f的语句** | **构造函数**  **调用次数** | **拷贝构造函数调用次数** | **析构函数调用次数** | | void f(Time t); | **f(obj);** | **1** | **1** | **1** | | void f(Time &t); | **f(obj);** | **1** | **0** | **1** | | void f(Time \*t); | **f(&obj);** | **1** | **0** | **1** |   通过以上结果，关于对象作形式参数、对象引用作形式参数、对象指针作形式参数时构造函数、析构函数的调用次数及顺序，你得到什么结论？   1. **对象作为形式参数** (void f(Time t);)：    * **构造函数调用次数**：1次（为创建主函数中的对象）    * **拷贝构造函数调用次数**：1次（将主函数中的对象作为实参传递给函数f时，需要通过拷贝构造函数创建一个新的局部对象）    * **析构函数调用次数**：2次（一次是局部对象t在f函数结束时被销毁，另一次是主函数中的对象在其作用域结束时被销毁） 2. **对象引用作为形式参数** (void f(Time &t);)：    * **构造函数调用次数**：1次（创建主函数中的对象）    * **拷贝构造函数调用次数**：0次（函数f接收的是对象的引用，不需要创建新的对象）    * **析构函数调用次数**：1次（只有主函数中的对象在其作用域结束时被销毁，因为没有新的对象被创建） 3. **对象指针作为形式参数** (void f(Time \*t);)：    * **构造函数调用次数**：1次（创建主函数中的对象）    * **拷贝构造函数调用次数**：0次（通过传递对象地址，不涉及对象的复制）    * **析构函数调用次数**：1次（只有主函数中的对象在结束时被销毁，因为没有新的对象被创建）   综上所述，使用对象引用或对象指针作为函数参数可以避免不必要的对象拷贝，从而提高程序的效率。对象作为形式参数时会涉及到额外的拷贝构造和析构操作，这可能会影响程序的性能，特别是当对象较大或者拷贝代价较高时。因此，在需要传递复杂的数据结构或大型对象时，推荐使用引用或指针。  **实验题目3** 定义一个Girl类和一个Boy类，这两个类中都有表示姓名、年龄的私有成员变量，都要定义构造函数、析构函数、输出成员变量信息的公有成员函数。  ①根据要求定义相应的类。  ②将Girl类作为Boy类的友元类，在Girl类的成员函数VisitBoy(Boy & )中访问Boy类的私有成员，观察程序运行结果。  ③在Boy类的某成员函数VisitGirl(Girl & )中试图访问Girl类的私有成员，观察编译器给出的错误信息，理解原因。  ④主函数中正确定义两个类的对象，调用各自的成员函数实现相应功能。  ⑤再将Boy类作为Girl类的友元类，在Boy类的某成员函数VisitGirl(Girl & )中访问Girl类的私有成员，观察编译器给出的信息。  ⑥删除两个类中的函数VisitGirl(Girl & ) ，VisitBoy(Boy & )，定义一个顶层函数VisitBoyGirl(Boy &, Girl &)，作为以上两个类的友元，通过调用该函数输出男孩和女孩的信息。  **实验解答：**  ①定义相应的类，主函数中定义相应的类成员，调用各类的输出函数显示信息。  **源程序代码如下：**  #include<iostream>  #include <cstring>  using namespace std;  class Boy;  class Girl  {  private:  string name;  int age;  public:  Girl(string name = "null", int age = 0): name(name), age(age){}  ~Girl() {}  void printInfo()  {  cout << name << " " << age << endl;  }  };  class Boy  {  private:  string name;  int age;  public:  Boy(string name = "null", int age = 0) : name(name), age(age) {}  ~Boy() {}  void printInfo()  {  cout << name << " " << age << endl;  }  };  int main()  {  Girl Girl;  Boy Boy;  Girl.printInfo();  Boy.printInfo();  Girl.visitBoy(Boy);  return 0;  }  **程序的运行结果是：**  **null 0**  **null 0**  ②将Girl类作为Boy类的友元类， 写出Girl类的成员函数VisitBoy(Boy & )的实现代码。  void Girl::visitBoy(Boy& b)  {  cout << b.name << " " << b.age << endl;  }  **程序的运行结果是：**  **null 0**  **null 0**  **null 0**  ③在Boy类的某成员函数VisitGirl(Girl & )中试图访问Girl类的私有成员，记录编译器给出的错误信息，与②对比，你能得出友元的什么特性？   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 严重性 | 代码 | 说明 | 项目 | 文件 | 行 | 禁止显示状态 | 详细信息 | | 错误 | C2248 | “Girl::name”: 无法访问 private 成员(在“Girl”类中声明) | Draft | C:\Users\Robin\OneDrive\文档\VisualStudio\Draft\Draft\Draft\Dratf.cpp | 48 |  |  | | 严重性 | 代码 | 说明 | 项目 | 文件 | 行 | 禁止显示状态 | 详细信息 | | 错误 | C2248 | “Girl::age”: 无法访问 private 成员(在“Girl”类中声明) | Draft | C:\Users\Robin\OneDrive\文档\VisualStudio\Draft\Draft\Draft\Dratf.cpp | 48 |  |  |   友元只能访问自己类中的私有成员  ④在上面代码的基础上，在Girl类的定义中，增加一行代码：friend Boy; 在主函数中通过**Boy类对象. VisitGirl(Girl类对象)** 的形式输出Girl类对象的信息。编译的结果是什么？写出这一步你的主函数代码，要求分别用友元函数**Girl类对象. VisitBoy(Boy类对象);**和**Boy类对象. VisitGirl(Girl类对象) ;**和输出两个类对象的信息。  #include<iostream>  #include <cstring>  using namespace std;  class Boy;  class Girl  {  private:  string name;  int age;  public:  Girl(string name = "null", int age = 0): name(name), age(age){}  ~Girl() {}  void printInfo()  {  cout << name << " " << age << endl;  }  void visitBoy(Boy& b);  friend Boy;  };  class Boy  {  private:  string name;  int age;  public:  Boy(string name = "null", int age = 0) : name(name), age(age) {}  ~Boy() {}  void printInfo()  {  cout << name << " " << age << endl;  }  friend class Girl;  void visitGirl(Girl& g);  };  void Girl::visitBoy(Boy& b)  {  cout << b.name << " " << b.age << endl;  }  void Boy::visitGirl(Girl& g)  {  cout << g.name << " " << g.age << endl;  }  int main()  {  Girl Girl;  Boy Boy;  Girl.printInfo();  Boy.printInfo();  Girl.visitBoy(Boy);  Boy.visitGirl(Girl);  return 0;  }  ⑤定义一个顶层函数void VisitBoyGirl(Boy &, Girl &),作为以上两个类的友元函数，主函数中通过调用该函数输出男孩和女孩的信息。写出该友元函数的完整代码，以及主函数的代码。  #include<iostream>  #include <cstring>  using namespace std;  class Boy;  class Girl;  void visitBoyGirl(Boy& b, Girl& g);  class Girl  {  private:  string name;  int age;  public:  Girl(string name = "null", int age = 0): name(name), age(age){}  ~Girl() {}  void printInfo()  {  cout << name << " " << age << endl;  }  void visitBoy(Boy& b);  friend class Boy;  friend void visitBoyGirl(Boy& b, Girl& g);  };  class Boy  {  private:  string name;  int age;  public:  Boy(string name = "null", int age = 0) : name(name), age(age) {}  ~Boy() {}  void printInfo()  {  cout << name << " " << age << endl;  }  friend class Girl;  void visitGirl(Girl& g);  friend void visitBoyGirl(Boy& b, Girl& g);  };  void Girl::visitBoy(Boy& b)  {  cout << b.name << " " << b.age << endl;  }  void Boy::visitGirl(Girl& g)  {  cout << g.name << " " << g.age << endl;  }  void visitBoyGirl(Boy& b, Girl& g)  {  cout << g.name << " " << g.age << endl;  cout << b.name << " " << b.age << endl;  }  int main()  {  Girl Girl;  Boy Boy;  Girl.printInfo();  Boy.printInfo();  Girl.visitBoy(Boy);  Boy.visitGirl(Girl);  visitBoyGirl(Boy, Girl);  return 0;  }  **四、实验小结**（包括问题和解决方法、心得体会、意见与建议等）  **（**中文五号宋体，英文五号Consolas字体，单倍行距)  (一)实验中遇到的主要问题及解决方法  1.在题目2中不改变main()函数中的对象的定义方式，若取消构造函数中参数的默认值，编译程序错误提示信息及出错原因是：  参数会缺少初始值。  2.在题目2中如果删除类中自定义的构造函数，仅使用系统默认构造函数，再编译，程序错误提示信息及出错原因是：  默认构造函数只适用于无参的构造调用：当你删除了所有自定义的构造函数，C++编译器将为类 Time 自动生成一个默认构造函数。这个自动生成的默认构造函数不带任何参数，也不会初始化类的任何成员变量。  3.在题目2中如果将main()函数中的输出语句改为：cout<<对象名.Hour<<":"<<对象名.Minute<<":"<<对象名.Second<<endl; 重新编译，会出现什么错误提示？在这种情况下，如果将成员变量的访问属性修改为public再编译，结果如何？  正在尝试从类的外部访问私有成员变量，这是C++不允许的操作。  4.其它问题及解决办法  无。  （二）实验心得  通过本次面向对象程序设计及C++课程的实验一，我对类和对象的概念有了更深刻的理解。实验过程中，我不仅学习到了如何定义类和对象，还掌握了构造函数、析构函数的使用，以及如何通过成员函数来实现对象间的交互。 在实验一中，我首先定义了一个BookCard类，用于模拟借书证的管理。在设计过程中，我特别注意了私有成员的使用，以确保数据的封装性。通过构造函数和析构函数，我学习到了对象的生命周期管理。此外，borrow和display函数的实现，让我对类的方法实现有了更深的认识。 在实现Time类时，我遇到了一些挑战，尤其是在处理时间增加一秒时的进位逻辑。通过不断调试和思考，我最终理解了如何正确处理小时、分钟和秒之间的进位关系。这个过程锻炼了我的逻辑思维能力，也加深了我对C++编程的理解。 此外，通过实验中的函数调用和参数传递，我了解到了对象、引用和指针作为参数时，构造函数和析构函数的调用机制。我认识到了引用和指针在避免对象拷贝、提高程序效率方面的重要性。 在实验的最后部分，我学习了友元类和友元函数的概念。通过尝试不同的访问权限，我体会到了友元在实现类间特定数据共享时的便利性，同时也理解了其在安全性方面的限制。 总结来说，这次实验不仅加深了我对C++面向对象编程的理解，而且提高了我的程序设计能力和解决问题的能力。我认识到了编程不仅仅是写代码，更重要的是如何设计出结构清晰、逻辑严谨的程序。在未来的学习中，我将继续探索面向对象设计的最佳实践，以提升我的编程技能。  （三）意见与建议（没有可省略）   1. **实验指导书的完善**：实验指导书已相当详尽，但建议在某些复杂的程序设计部分，如时间类中增加一秒的逻辑处理，可以提供更多的示例代码或详细步骤说明，帮助学生更好地理解。 2. **增加实践案例**：希望在实验中增加更多与实际应用相关的案例，如数据库管理、网络编程等，以增强实验的实用性和趣味性。 3. **增强互动性**：建议在实验课中增加学生之间的互动环节，如小组讨论、代码互评等，以促进学生之间的交流和学习。 | | | |

**实 验 报 告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1. **支撑毕业要求指标点**   **信息安全：**  1.2-M掌握计算机软硬件相关工程基础知识，能将其用于分析信息安全领域的相关工程问题。  3.1-H掌握信息安全领域所涉及的软硬件系统，从数字电路、计算机系统、到各类系统软件的基本理论与设计结构。 | | | | | | |
| **六、指导教师评语** | | | | | | |
| **评 分 细 则** | **评分项** | **优秀** | **良好** | **中等** | **合格** | **不合格** |
| **遵守实验室规章制度** |  |  |  |  |  |
| **学习态度** |  |  |  |  |  |
| **算法思想准备情况** |  |  |  |  |  |
| **程序设计能力** |  |  |  |  |  |
| **解决问题能力** |  |  |  |  |  |
| **算法设计合理性** |  |  |  |  |  |
| **算法效能评价** |  |  |  |  |  |
| **报告书写认真程度** |  |  |  |  |  |
| **内容详实程度** |  |  |  |  |  |
| **文字表达熟练程度** |  |  |  |  |  |
| **其它评价意见** |  | | | | |
| **本次实验能力达成评价（总成绩）** |  | **批阅人** |  | **日期** |  |