

**计算机学院（软件学院）实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **学生姓名** | 徐可可 | **学号** | 171530425 | **实验成绩** |  |
| **专业** | 软件工程 | **班级** | 软工四班 | **实验日期** | 2019.9.18 2019.9.25 |
| **课程名称** | 《 C#高级程序设计》 | | | **任课教师** | 彭伟国 |
| **实验名称** | 面向对象编程基础 | | | **实验序号** | 3 |
| **实验地点** | S407 | **实验台号** | 25 | **指导教师** | 彭伟国 |
| **一、实验目的及要求**   1. 掌握类的定义和对象的创建。 2. 理解并掌握构造函数和析构函数的定义和使用。 3. 理解方法的4种参数传递的区别。 4. 掌握重载方法的使用方法。 5. 会运用String类的属性和方法操作字符串。 6. 能够使用Math类、Random类、DataTime类中的属性和方法进行程序设计。 | | | | | |
| **二、实验内容（或实验原理、实验拓扑）**   1. 写一个Student类和一个Teacher类,他们都有一个打招呼的方法,不同的是Studetn打招呼是说"大家好,我叫XX,我今年XX岁了,我的爱好是XXX",Teacher的打招呼的方法是说"大家好,我叫XX,我今年XX岁了,我已经工作XX年了"? 2. 设计一个程序使用person类，其类中包含如下信息：姓名：name；血型：blood；创建一个person对象p如下：person p=new person(“江涛”,“AB”)；然后，使用printname方法将姓名显示出来。使用printblood方法将血型显示出来。 3. 创建一个Circle类，两个方法分别求圆的面积和周长。并在主函数中调用。 4. 编写Test类，包含average方法，实现求数组中所有元素的平均值，Main中调用方法，显示出平均值。 5. 从日期字符串（“2008-08-08”）中分析出年、月、日；2008年08月08日。让用户输入一个日期格式如:2008-01-02,你输出你输入的日期为2008年1月2日 6. 接收用户输入的字符串，将其中的字符以与输入相反的顺序输出。“abc"→"cba” 7. 接收用户输入的一句英文，将其中的单词以反序输出。“hello c sharp"→"sharp c hello” 8. 从Email中提取出用户名和域名：abc@163.com。 9. 生成一个由大小写及数字组成的验证码，随机生成一个四个字符长度的验证码，输出一下，让用户对比输入，正确输出【输入正确】不正确输出【输入有误】 | | | | | |
| **三、实验设备与环境**  **I7 7700HQ+16G内存+38TB分布式硬盘的笔记本电脑、Windows 10 1903政府版、Visual Studio 2019 企业版、.NET Framework 4.7.2** | | | | | |
| **四、实验设计方案（包括实验步骤、设计思想、算法描述或开发流程等）** | | | | | |

|  |
| --- |
| **五、实验结果（包括设计效果、测试数据、运行结果等）**  **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9** |
| **六、实验小结（包括收获、心得体会、注意事项、存在问题及解决办法、建议等）**   1. 字段 字段主要为类的内部做数据交互使用，字段一般为private。 字段可以赋值，可取值。 当字段需要为外部提供数据时，请将字段封装为属性，而不是使用公有字段，这是面向对象所提倡的。 2. 属性–描述对象的静态特征 属性其实是外界访问私有字段的入口，属性本身不保存任何数据。 给属性赋值，其实就是给属性的私有字段赋值；读取属性，其实是获取属性指向的私有字段或其他值。 属性一般是向外部提供数据，主要用来描述对象的静态特征，所以属性一般是public的。 属性可以根据需要设置为只读、只写，提高数据安全性。只读属性没有set方法。 属性内部可以添加需要的逻辑，从而避免非法数据。 3. 自动属性 不能直接操作私有字段 不能设置只读只写 不能添加验证逻辑 4. 方法的定义 概念：对象的动态特征就是方法（静态特征是属性），方法表示此对象可以做什么。 类型：实例方法，静态方法，（构造方法，多态时会用到抽象方法和虚方法） 5. 方法注意事项 访问修饰符：默认private 方法名：一般时动词或动宾短语。首字母大写。不能以数字开头。 参数列表：根据需要添加 成员变量==字段，相对与局部变量 6. 面向对象小结    1. 封装 封装：将一个或若干个小对象包装在一起，然后以一个新的对象呈现 作用：封装使得一个对象的内部特征变得隐蔽，从而更安全。 类封装：类是最小的程序单元，是对属性、方法的封装，类中的public成员都是对外提供的数据访问接口。 模块封装：类可以被封装成模块，组成更大的程序组件。 从OOP理解：外界调用的时候可以不用关心内部实现细节；充分体现面向对象中的“高内聚、低耦合”特性。 对象的重用性：依此封装，处处使用。    2. 变量 变量作用：指代内存地址，便于记忆。 变量的作用范围：       * 局部变量：一个方法内部交换数据，方法执行完毕，如果没有返回值，则变量很快被清理       * 成员变量：多个方法内部交换数据，通过对象的属性外对提供数据，变量的清理不确定 7. 方法重载    1. 方法重载：编译器将根据方法参数的个数和类型自动匹配类的对应方法。    2. 方法重载的好处：减少类的对外接口（只显示一个方法），降低类的复杂度（低耦合）； 便于用户使用（相同功能的方法名称一样）和识别    3. 方法重载的条件：       * 方法的名称必须一样       * 方法的参数个数或参数类型不一样    4. 方法重载的无关性：方法重载与返回值无关。 8. 静态方法与重载 关键字static：可以修饰类、方法、成员变量，修时候称之为静态类、静态方法、静态字段。 静态成员在程序运行时被调入内存中，并且在系统未关闭之前不会被GC吸收。 类的成员使用非常频繁时，可以考虑使用static修饰，但不要使用过多。 静态成员不能直接调用实例成员（静态方法不能直接调用实例方法）。 静态方法也可以重载。 9. 构造方法 如果两个重载的构造方法中有代码重复，可以用“:this()”调用自身其他构造方法。 10. 对象初始化器 对象在创建时也可以通过对象初始化器完成“属性的初始化”。   Student objStudent3 = new Student()  {  StudentId = 10005,  Birthday = Convert.ToDateTime("1987-10-11"),  StudentName = "小红"  };  对象初始化器和构造方法的异同点 　　相同点：都可以完成属性的初始化。 　　不同点：1构造函数具有强制性，对象初始化器没有强制性 　　　　　　2对象初始化器只能完成属性初始化，构造函数可以完成任何需要的初始化工作，如对象创建时读取文件等 　　　　　　3对象初始化器是在创建对象的时候使用，构造函数写在类里面 　　　　　　4对象初始化器只能在.NET3.0以后版本使用，构造函数在.NET1.0版本就有 对象创建（new）的两种方式：构造方法和初始化器。   1. 对象销毁 对象的生命周期：对象在内存不断额“生生死死”，具有生命周期。 对象在内存中的状态：正在引用 程序正在使用的对象 游离状态 没有引用的对象，已经使用完毕但依然占据空间 垃圾回收机制（GC）：.NET虚拟机特有机制，自动运行，并检查对象的状态。 发现对象不再引用时，会将其释放所占有的空间（销毁）。 GC的使用大大减轻了程序员手动释放对象的操作，且程序运行稳定。 2. 析构函数 C++中新购函数的使用，当使用delete关键字删除对象时，析构函数会自动调用。   ~类名(){}  托管运行环境下对象的销毁由虚拟机负责，析构函数没多大用途。   1. 对象的数据类型 基本数据类型：int、double、bool等 　　引用数据类型：数组、对象、string（string分类上属于引用数据类型，但数据的处理和基本数据类型一样） 　　两个对象，对象1.Name=“王华”; 对象2=对象1; 对象2.Name=“李红”; 此时对象1.Name也是“李红”。说明===>>>对象是引用类型。 2. 两种数据类型特点 　　基本数据类型（值类型）：值类型变量传递时，将自己复制一份赋给了新变量。 　　引用数据类型（对象类型）：引用类型变量传递时，将自己的内存地址赋给了新变量。 3. ref关键字 　　值类型在传递的过程中都是以副本的方式进行传递，使用ref关键字可以将值类型变量按照引用类型方式传递。 　　实际开发中不建议经常使用ref关键字！ 4. out关键字 　　使用out关键字可以让方法有多个返回值 　　out关键字其实也是使用引用类型传递！实践开发中不建议经常使用out关键字！ 　　可以使用“字典集合”方式返回多个参数！ 5. 类和对象小结 　　1封装的概念和作用：根据需要使用private和public修饰符类修饰类的成员（字段、属性、方法等），这样能避免私有的字段和方法被外界使用，我们把这种使用特性称为封装。 　　　　封装是一个广义概念，一个属性是对字段的封装，一个方法是对一系列具体实现方式的封装，一个类是对属性和方法的封装，一个模块组件是对类的封装。 　　2面向对象编程好处 　　　　　　封装使得开发人员可以更加关注自己的需求：类的方法实现特定的功能，其他的类可以完全不用关心它是如何实现的，只需要调用即可；团队开发中，类的编写者也可以完全不用关心别人是如何使用的。 　　　　　　一个类定义以后，可多次使用，避免重复使用。 　　3类视图 　　　　通过视图方式展示类的组织结构。私有成员带一个“锁”标志。 　　　　对类右键“查看类图”。 　　4认识祖先类Object 　　　　.NET平台规定：所有类都必须自动继承自object类。 　　5装箱与拆箱 　　　　装箱：将值类型转化为对象的过程 object a=20； 　　　　拆箱：将对象类型的变量转化为值类型的过程 int b=(int)a; 　　　　频繁的拆装箱操作会影响系统性能。 6. protected关键字 protected声明的字段可以被基类或基类的派生类所引用。 请注意要么在基类中被引用（相当于private），要么是在派生类中被派生类声明的对象所引用。 在派生类中不能被基类声明的对象所引用。 |
| **七、附录（包括作品、流程图、源程序及命令清单等）**  本实验报告同步更新于：<https://blog.csdn.net/COCO56/article/details/101345819>   1. Student.cs   using System;  namespace \_3\_1  {  class Student  {  public string Name;  public double Age;  public string Hobby;  public void hello()  {  Console.WriteLine("我叫{0}，我今年{1}岁了，我的爱好是{2}。", this.Name, this.Age, Hobby);  }  }  }  Teacher.cs  using System;  namespace \_3\_1  {  class Teacher  {  public string Name;  public double Age;  public double WorkingAge;  public void hello()  {  Console.WriteLine("我叫{0}，我今年{1}岁了，我的工龄是{2}。", this.Name, this.Age, WorkingAge);  }  }  }  Program.cs  using System;  namespace \_3\_1  {  class Program  {  static void Main(string[] args)  {  Student stu = new Student()  {  Name = "Coco",  Age = 22,  Hobby = "coding"  };  stu.hello();  Teacher t = new Teacher();  t.Name = "Mr. Smith";  t.Age = 42;  t.WorkingAge = 15;  t.hello();  Console.WriteLine("\nPress any key to quit.");  Console.ReadKey();  }  }  }   1. Person.cs   using System;  namespace \_3\_2  {  class Person  {  public string Name;  public string Blood;  public Person(string name, string blood)  {  Name = name;  Blood = blood;  }  public void printName()  {  Console.WriteLine("我叫{0}。", this.Name);  }  public void printBlood()  {  Console.WriteLine("我的血型是{0}。", this.Blood);  }  }  }  Program.cs  using System;  namespace \_3\_2  {  class Program  {  static void Main(string[] args)  {  Person p = new Person("江涛", "AB");  p.printName();  p.printBlood();  Console.WriteLine("\nPress any key to quit.");  Console.ReadKey();  }  }  }   1. Circle.cs   using System;  namespace \_3\_3  {  class Circle  {  public double R;  public Circle(double r)  {  R = r;  }  public void c()  {  Console.WriteLine("周长{0}。", 2\*Math.PI\* R);  }  public void s()  {  Console.WriteLine("面积{0}。", Math.PI \* R \* R);  }  }  }  Program.cs  using System;  namespace \_3\_3  {  class Program  {  static void Main(string[] args)  {  Circle c = new Circle(1);  c.c();  c.s();  Console.WriteLine("\nPress any key to quit.");  Console.ReadKey();  }  }  }   1. Test.cs   using System;  namespace \_3\_4  {  class Test  {  public double[] Nums;  public Test(double[] numbers)  {  Nums = numbers;  }  public void average()  {  double s=0;  foreach (var item in Nums)  {  s += item;  }  Console.WriteLine("均值为：{0}。", s/Nums.Length);  }  }  }  Program.cs  using System;  namespace \_3\_4  {  class Program  {  static void Main(string[] args)  {  double[] n = { 1, 1, 2 };  Test t=new Test(n);  t.average();  Console.WriteLine("\nPress any key to quit.");  Console.ReadKey();  }  }  }   1. Program.cs   using System;  namespace \_3\_5  {  class Program  {  static void Main(string[] args)  {  string str = "2008-08-08";  DateTime date = DateTime.Parse(str);  Console.WriteLine(date.ToLongDateString());  Console.WriteLine("\nPress any key to quit.");  Console.ReadKey();  }  }  }   1. Program.cs   using System;  using System.Linq;  using System.Text;  namespace \_3\_6  {  class Program  {  static void Main(string[] args)  {  string a = "abc";  var query = a.Reverse();  StringBuilder sbStr = new StringBuilder();  foreach (var item in query)  {  sbStr.Append(item);  }  Console.WriteLine(sbStr.ToString());  string s = "abc";  char[] ss = s.ToCharArray();  StringBuilder sb = new StringBuilder();  for (int i = ss.Length - 1; i >= 0; i--)  {  sb.Append(ss[i]);  }  Console.WriteLine(sb.ToString());  Console.WriteLine("\nPress any key to quit.");  Console.ReadKey();  }  }  }   1. Program.cs   using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace \_3\_7  {  class Program  {  static void Main(string[] args)  {  string[] arr = "hello c sharp".Split(' ');  string result = string.Empty;  for (int i = arr.Count() - 1; i >= 0; i--)  {  result += arr[i] + " ";  }  Console.WriteLine("反转后的英文句子是：\n方法一：" + result);  Console.Write("方法二：");  string[] arr2 = "hello c sharp2".Split(' ');  Array.Reverse(arr2);  foreach (var item in arr2)  {  Console.Write(item + " ");  }  Console.WriteLine();  Console.WriteLine("\nPress any key to quit.");  Console.ReadLine();  }  }  }   1. Program.cs   using System;  namespace \_3\_8  {  class Program  {  static void Main(string[] args)  {  //练习：从email中提取用户名和域名 abc@163.com  string str = "abc@163.com";  int index = str.IndexOf('@');  string strUser = str.Substring(0, index);  string strDNS = str.Substring(index + 1);  Console.WriteLine(strDNS);  Console.WriteLine(strUser);  Console.WriteLine("\nPress any key to quit.");  Console.ReadKey();  }  }  }   1. Program.cs   using System;  namespace \_3\_9  {  class Program  {  static void Main(string[] args)  {  Random random = new Random();  //for (int i = 0; i < 10; i++)  // Console.WriteLine("The latest verification code is: " + getVerificationCode(random));  inputCode(getVerificationCode(random));  Console.WriteLine("\nPress any key to quit.");  Console.ReadKey();  }  public static void inputCode(string code)  {  Console.WriteLine("The latest verification code is: " + code);  Console.WriteLine("Please input the latest verification code.");  while (true)  {  string s = Console.ReadLine();  if(s==code)  {  Console.WriteLine("Congratulation, you passed the validation.");  break;  }  Console.WriteLine("Sorry, your input is wrong, please try again.");  }  }  public static string getVerificationCode(Random random)  {  string str = "";  for (int i = 0; i < 4; i++)  {  int key = random.Next(3);  switch (key)  {  case 0:  int code1 = random.Next(10);  str += code1;  break;  case 1:  char code2 = (char)(random.Next(26) + 65);  str += code2;  break;  case 2:  char code3 = (char)(random.Next(26) + 97);  str += code3;  break;  }  }  return str;  }  }  } |