《可视化程序设计》实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 | 实验1、流程控制语句的使用 | | |
| 姓名 | 胡松涛 | 学号 | 1501511769 |
| 实验时间 | 2016年9月19 | 成绩 |  |
| **实验目的：**  1.学会使用Visual Studio 2012 编写C#程序  2.训练使用流程控制语句；  **实验环境：**  安装有Windows 7操作系统的计算机一台  Visual Studio 2012  **实验内容：**  **实验 1-1：**控制台菜单（Menu\_Hst，如果输入的题号不是菜单中所列数字序号，要明确提示是负数、超范围或非数字）  **实验1-2：**九九乘法表（MultiplicationTable\_Hst，锯齿状显示；2\*1=2）  **实验1-3：**输入年份输出生肖（ShengXiao\_Hst，鼠、牛、虎、兔、龙、蛇、马、羊、猴鸡、狗、猪；年份范围：1900-2100年）  **实验1-4：**客车售票系统（Ticket\_Hst，不能有0行0列，for中不能出现9和4）  **实验1-5：**字符串加密（StrEncrypt\_Hst，即将一个字符串中的所有字符按同一密钥变成另外一个字符串。如有a1，密钥为3时，应为d4。注意越界的情况，如有xZ9，当密钥为3时，应为aC2。分别输入字符串和密钥，显示加密后的字符串）  **实验过程及结果记录：**   1. **实验1.1**   （一）控制台菜单实现代码：  static void Menu\_Hst() //控制台菜单  {  while (true)  {  Console.Title = "程序清单";  Console.ForegroundColor = ConsoleColor.DarkRed; //改字体颜色  Console.WriteLine("\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  Console.WriteLine("\t\t1.九九乘法表\n");  Console.WriteLine("\t\t2.输入年份输出生肖\n");  Console.WriteLine("\t\t3.客车售票系统\n");  Console.WriteLine("\t\t4.字符串加密\n");  Console.WriteLine("\t\t0.退出程序\n");  Console.WriteLine("\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Console.WriteLine("\t\t请选择功能：");  try //检测选择功能时的输入异常  {  int select\_Hst; //记录选择的功能  select\_Hst = int.Parse(Console.ReadLine());  if (select\_Hst < 0 || select\_Hst > 4) throw new IndexOutOfRangeException();  switch (select\_Hst)  {  case 1:  Console.Clear();  MultiplicationTable\_Hst(); //九九乘法表  break;  case 2:  Console.Clear();  ShengXiao\_Hst(); //生肖查询  break;  case 3:  Console.Clear();  Ticket\_Hst(); //购票系统  break;  case 4:  Console.Clear();  SrtEncrypt\_Hst(); //密码加密  break;  case 0:  return;  }  }  catch (Exception ex\_Hst) //捕获抛出的异常  {  Console.WriteLine(ex\_Hst);  Console.ReadLine();  }  finally  {  Console.Clear();  }  }  }  （二）程序实际运行效果：  正常输入：  1  输入超出范围序号：  1-2  误输入字符串：  1-3  **二、实验1-2**  （一）九九乘法表实现代码：  static void MultiplicationTable\_Hst() //九九乘法表  {  int[][] elements\_Hst= new int [9][]; //建立不规则数组来用于遍历  Console.WriteLine("\t\t\t九九乘法表");  for (int i\_Hst = 0; i\_Hst < elements\_Hst.Length; i\_Hst++)  {  elements\_Hst[i\_Hst] = new int[i\_Hst + 1]; //对不规则数组进行实例化  for (int j\_Hst = 0; j\_Hst < elements\_Hst.Length; j\_Hst++)  {  if (i\_Hst >= j\_Hst) //判断输出条件，使乘法表能够以阶梯形输出  {  Console.Write((j\_Hst + 1) + "x" + (i\_Hst + 1) + "=" + ((i\_Hst + 1) \* (j\_Hst + 1)) + "\t");  }  }  Console.WriteLine(); //用于输出换行  }  Console.ReadLine();  }  （二）程序实际运行效果：  2  （三）出现错误和解决：  错误：  2-2  输出的表和正确的表排列不同  解决：  互换循环中i（i\_Hst）和j（i\_Hst）的位置  **三、实验1-3**  （一）输入年份输出生肖实现代码：  static void ShengXiao\_Hst()  {  char[] ShengXiao\_Hst = { '鼠', '牛', '虎', '兔', '龙', '蛇', '马', '羊', '猴', '鸡', '狗', '猪' }; //定义字符型数组存放生肖  int year\_Hst,Index\_Hst; //声明年份储存和生肖数组索引  while (true)  {  Console.WriteLine("\t\t1900年-2100年生肖查询");  Console.Write("\t请输入年份（输入-1返回上一级菜单）：");  try  {  year\_Hst = int.Parse(Console.ReadLine());  if ((year\_Hst > 2100 || year\_Hst < 1900)&&year\_Hst !=-1) throw new IndexOutOfRangeException(); //判断输入的年份是否在规定范围内  if (year\_Hst == -1)  {  break;  }  Index\_Hst = (year\_Hst - 1900) % 12; //判断生肖，确定索引  Console.WriteLine("\t"+year\_Hst + "年的生肖为：" + ShengXiao\_Hst[Index\_Hst]);  Console.Write("\t确定");  Console.ReadLine();  Console.Clear();  }  catch(Exception ex\_Hst)  {  Console.WriteLine(ex\_Hst);  Console.ReadLine();  Console.Clear();  }  }  }  （二）程序实际运行效果：  3-1  3-2  3-3  **四、实验1-4**  （一）客车售票系统实现代码：  static void Ticket\_Hst() //客车售票系统  {  String[,] seat\_Hst = new String[9, 4];  for (int i\_Hst = 0; i\_Hst < seat\_Hst.GetLength(0); i\_Hst++) //初始化所有数组元素为“有票”  for (int j\_Hst = 0; j\_Hst < seat\_Hst.GetLength(1); j\_Hst++)  {  seat\_Hst[i\_Hst, j\_Hst] = "【有票】";  }  while(true)  {  Console.WriteLine(" 1 2 3 4");  for (int i\_Hst = 0; i\_Hst < seat\_Hst.GetLength(0); i\_Hst++)  {  for (int j\_Hst = 0; j\_Hst < seat\_Hst.GetLength(1); j\_Hst++)  {  if (seat\_Hst[i\_Hst, j\_Hst] == "【已售】")  {  Console.ForegroundColor = ConsoleColor.Yellow; //改变字体颜色以突出此座位已被预订  }  Console.Write(seat\_Hst[i\_Hst ,j\_Hst]+" "); //输出数组元素内的文本内容  Console.ForegroundColor = ConsoleColor.DarkRed; //恢复字体颜色  }  Console.WriteLine(i\_Hst+1);  }  try  {  Console.WriteLine("请输入您要购买的座位（如：(2,3)）（输入q退出）");  String location\_Hst = Console.ReadLine();  if (location\_Hst == "q")  { break; }  String[] temp = location\_Hst.Split(','); //建立临时数组存放分割的字符  int row\_Hst = int.Parse(temp[0]); //行定位  int line\_Hst = int.Parse(temp[1]); //列定位  if (row\_Hst < 1 || row\_Hst > 9 || line\_Hst < 1 || line\_Hst > 4) throw new IndexOutOfRangeException(); //判断是否越界  if (seat\_Hst[row\_Hst - 1, line\_Hst - 1] == "【已售】") //判断座位是否已经被卖出  {  Console.WriteLine("此座位已被预定，请选择其他座位");  Console.ReadLine();  Console.Clear();  continue;    }  else  {  seat\_Hst[row\_Hst - 1, line\_Hst - 1] = "【已售】";  }  Console.WriteLine("确认");  Console.ReadLine();  Console.Clear();  }  catch (Exception ex\_Hst)  {  Console.WriteLine(ex\_Hst);  Console.ReadLine();  Console.Clear();  }  }    }  （二）程序实际运行效果：  正确的输入座位：  4-1  选择已被选择的座位：  4-2  选择规定座位以外的座位：  4-3  输入任意字符串：  4-4  **五、实验1-5：**  （一）字符串加密实现代码：  static void SrtEncrypt\_Hst()  {  while (true)  {  try  {  int code\_Hst = 0;  Console.WriteLine("请输入您要加密的内容（输入-1退出）");  String sentence\_Hst;  sentence\_Hst = Console.ReadLine();  if (sentence\_Hst == "-1")  { break; }  for (int i\_Hst = 0; i\_Hst < sentence\_Hst.Length; i\_Hst++) //检查输入的内容是否合法  {  if (!((sentence\_Hst[i\_Hst] >= '0' && sentence\_Hst[i\_Hst] <= '9') || (sentence\_Hst[i\_Hst] <= 'z' && sentence\_Hst[i\_Hst] >= 'a') || (sentence\_Hst[i\_Hst] <= 'Z' && sentence\_Hst[i\_Hst] >= 'A'))) throw new IndexOutOfRangeException();  }  Console.WriteLine("请输入加密密钥（大于零的整数）");  code\_Hst = int.Parse(Console.ReadLine());  if (code\_Hst < 0) throw new IndexOutOfRangeException(); //检查输入的密钥是否大于零  int[] temp = new int[sentence\_Hst.Length]; //建立临时数组以储存改变后的已加密字符串  for (int i\_Hst = 0; i\_Hst < sentence\_Hst.Length; i\_Hst++)  {  if (sentence\_Hst[i\_Hst] >= '0' && sentence\_Hst[i\_Hst] <= '9') //判断单个字符元素是数字还是字母  {  int key;  key = (code\_Hst) % 10; //详细加密算法  temp[i\_Hst] = sentence\_Hst[i\_Hst] + key;  if (temp[i\_Hst] > '9')  {  temp[i\_Hst] = '0' + (key - ('9' - sentence\_Hst[i\_Hst])) - 1;  }  }  if (sentence\_Hst[i\_Hst] >= 'a' && sentence\_Hst[i\_Hst] <= 'z') //判断单个字符元素是数字还是大小写字母  {  int key;  key = (code\_Hst) % 26; //详细加密算法  temp[i\_Hst] = sentence\_Hst[i\_Hst] + key;  if (temp[i\_Hst] > 'z')  {  temp[i\_Hst] = 'a' + (key - ('z' - sentence\_Hst[i\_Hst])) - 1;  }  }  if (sentence\_Hst[i\_Hst] >= 'A' && sentence\_Hst[i\_Hst] <= 'Z') //判断单个字符元素是数字还是大小写字母  {  int key;  key = (code\_Hst) % 26; //详细加密算法  temp[i\_Hst] = sentence\_Hst[i\_Hst] + key;  if (temp[i\_Hst] > 'Z')  {  temp[i\_Hst] = 'A' + (key - ('Z' - sentence\_Hst[i\_Hst])) - 1;  }  }  }  Console.WriteLine("加密后的字符串为");  for (int i\_Hst = 0; i\_Hst < sentence\_Hst.Length; i\_Hst++) //输出加密后的字符串  {  Console.Write((char)temp[i\_Hst]);  }  Console.ReadLine();  Console.Clear();  }  catch (Exception ex\_Hst)  {  Console.WriteLine(ex\_Hst);  Console.ReadLine();  Console.Clear();  }  }    }  （二）程序实际运行效果：  正确的输入：  5-1  输入超出范围的字符串：  5-2  输入错误的密钥：  5-3 | | | |
| **实验总结：**  经过此次的实验，我熟练掌握了C#语言中流程控制语句的使用，还熟悉了try...catch...语句在C#语言中的使用，了解了一些常见的类中的常见方法，为后期的可视化编程提供了一定的基础。 | | | |