**LINQ**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace ConsoleApplication3**

**{**

**public class Racer : IComparable<Racer>, IFormattable**

**{**

**public Racer(string firstName = null, string lastName = null, string country = null, int starts = 0, int wins = 0, IEnumerable<int> years = null, IEnumerable<string> cars = null)**

**{**

**this.FirstName = firstName;**

**this.LastName = lastName;**

**this.Country = country;**

**this.Starts = starts;**

**this.Wins = wins;**

**var yearsList = new List<int>();**

**foreach (var year in years)**

**{**

**yearsList.Add(year);**

**}**

**this.Years = yearsList.ToArray();**

**var carList = new List<string>();**

**foreach (var car in cars)**

**{**

**carList.Add(car);**

**}**

**this.Cars = carList.ToArray();**

**}**

**public string FirstName { get; set; }**

**public string LastName { get; set; }**

**public string Country { get; set; }**

**public int Wins { get; set; }//夺冠场数**

**public int Starts { get; set; }//首发场数**

**public string[] Cars { get; private set; }//赛车手获得冠军那一年使用的所有车型**

**public int[] Years { get; private set; }//赛车手获得冠军的年份**

**public override string ToString()**

**{**

**return String.Format("{0} {1}", FirstName, LastName);**

**}**

**public int CompareTo(Racer other)**

**{**

**if (other == null) throw new ArgumentNullException("other");**

**return this.LastName.CompareTo(other.LastName);**

**}**

**public string ToString(string format)**

**{**

**return ToString(format, null);**

**}**

**public string ToString(string format,**

**IFormatProvider formatProvider)**

**{**

**switch (format)**

**{**

**case null:**

**case "N":**

**return ToString();**

**case "F":**

**return FirstName;**

**case "L":**

**return LastName;**

**case "C":**

**return Country;**

**case "S":**

**return Starts.ToString();**

**case "W":**

**return Wins.ToString();**

**case "A":**

**return String.Format("{0} {1}, {2}; starts: {3}, wins: {4}",**

**FirstName, LastName, Country, Starts, Wins);**

**default:**

**throw new FormatException(String.Format("Format {0} not supported", format));**

**}**

**}**

**public static IList<Racer> GetChampions()//1950-2008年一级方程式锦标赛冠军**

**{**

**IList<Racer> racers = new List<Racer>(40);**

**racers.Add(new Racer("Nino", "Farina", "Italy", 33, 5, new int[] { 1950 }, new string[] { "Alfa Romeo" }));**

**racers.Add(new Racer("Alberto", "Ascari", "Italy", 32, 10, new int[] { 1952, 1953 }, new string[] { "Ferrari" }));**

**racers.Add(new Racer("Juan Manuel", "Fangio", "Argentina", 51, 24, new int[] { 1951, 1954, 1955, 1956, 1957 }, new string[] { "Alfa Romeo", "Maserati", "Mercedes", "Ferrari" }));**

**racers.Add(new Racer("Mike", "Hawthorn", "UK", 45, 3, new int[] { 1958 }, new string[] { "Ferrari" }));**

**racers.Add(new Racer("Phil", "Hill", "USA", 48, 3, new int[] { 1961 }, new string[] { "Ferrari" }));**

**racers.Add(new Racer("John", "Surtees", "UK", 111, 6, new int[] { 1964 }, new string[] { "Ferrari" }));**

**racers.Add(new Racer("Jim", "Clark", "UK", 72, 25, new int[] { 1963, 1965 }, new string[] { "Lotus" }));**

**racers.Add(new Racer("Jack", "Brabham", "Australia", 125, 14, new int[] { 1959, 1960, 1966 }, new string[] { "Cooper", "Brabham" }));**

**racers.Add(new Racer("Denny", "Hulme", "New Zealand", 112, 8, new int[] { 1967 }, new string[] { "Brabham" }));**

**racers.Add(new Racer("Graham", "Hill", "UK", 176, 14, new int[] { 1962, 1968 }, new string[] { "BRM", "Lotus" }));**

**racers.Add(new Racer("Jochen", "Rindt", "Austria", 60, 6, new int[] { 1970 }, new string[] { "Lotus" }));**

**racers.Add(new Racer("Jackie", "Stewart", "UK", 99, 27, new int[] { 1969, 1971, 1973 }, new string[] { "Matra", "Tyrrell" }));**

**racers.Add(new Racer("Emerson", "Fittipaldi", "Brazil", 143, 14, new int[] { 1972, 1974 }, new string[] { "Lotus", "McLaren" }));**

**racers.Add(new Racer("James", "Hunt", "UK", 91, 10, new int[] { 1976 }, new string[] { "McLaren" }));**

**racers.Add(new Racer("Mario", "Andretti", "USA", 128, 12, new int[] { 1978 }, new string[] { "Lotus" }));**

**racers.Add(new Racer("Jody", "Scheckter", "South Africa", 112, 10, new int[] { 1979 }, new string[] { "Ferrari" }));**

**racers.Add(new Racer("Alan", "Jones", "Australia", 115, 12, new int[] { 1980 }, new string[] { "Williams" }));**

**racers.Add(new Racer("Keke", "Rosberg", "Finland", 114, 5, new int[] { 1982 }, new string[] { "Williams" }));**

**racers.Add(new Racer("Niki", "Lauda", "Austria", 173, 25, new int[] { 1975, 1977, 1984 }, new string[] { "Ferrari", "McLaren" }));**

**racers.Add(new Racer("Nelson", "Piquet", "Brazil", 204, 23, new int[] { 1981, 1983, 1987 }, new string[] { "Brabham", "Williams" }));**

**racers.Add(new Racer("Ayrton", "Senna", "Brazil", 161, 41, new int[] { 1988, 1990, 1991 }, new string[] { "McLaren" }));**

**racers.Add(new Racer("Nigel", "Mansell", "UK", 187, 31, new int[] { 1992 }, new string[] { "Williams" }));**

**racers.Add(new Racer("Alain", "Prost", "France", 197, 51, new int[] { 1985, 1986, 1989, 1993 }, new string[] { "McLaren", "Williams" }));**

**racers.Add(new Racer("Damon", "Hill", "UK", 114, 22, new int[] { 1996 }, new string[] { "Williams" }));**

**racers.Add(new Racer("Jacques", "Villeneuve", "Canada", 165, 11, new int[] { 1997 }, new string[] { "Williams" }));**

**racers.Add(new Racer("Mika", "Hakkinen", "Finland", 160, 20, new int[] { 1998, 1999 }, new string[] { "McLaren" }));**

**racers.Add(new Racer("Michael", "Schumacher", "Germany", 250, 91, new int[] { 1994, 1995, 2000, 2001, 2002, 2003, 2004 }, new string[] { "Benetton", "Ferrari" }));**

**racers.Add(new Racer("Fernando", "Alonso", "Spain", 132, 21, new int[] { 2005, 2006 }, new string[] { "Renault" }));**

**racers.Add(new Racer("Kimi", "R?ikk?nen", "Finland", 148, 17, new int[] { 2007 }, new string[] { "Ferrari" }));**

**racers.Add(new Racer("Lewis", "Hamilton", "UK", 44, 9, new int[] { 2008 }, new string[] { "McLaren" }));**

**return racers;**

**}**

**}**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**IList<Racer> racers = Racer.GetChampions();**

**var query = from r in racers**

**where r.Country == "Brazil"**

**orderby r.Wins descending**

**select r;**

**Console.WriteLine("查询来自巴西的所有世界冠军，并按胜利场数降序排列");**

**query = racers.Where(x => x.Country == "Brazil").OrderByDescending(x=>x.Wins);**

**foreach(Racer racer in query)**

**Console.WriteLine("{0:A}", racer);**

**Console.WriteLine();**

**Console.WriteLine("查出所有胜利场数超过25，用foreach循环并打印");**

**var query2 = from r in racers**

**where r.Wins > 25**

**select r;**

**foreach(Racer racer in query2)**

**Console.WriteLine("{0:A}", racer);**

**//var query3 = from r in racers**

**// where r.Country == "UK"**

**// select r;**

**Console.WriteLine("然后将所有英国（UK）车手的国名改为“United Kingdom”,再次用foreach循环并打印");**

**foreach (Racer racer in racers)**

**{**

**if (racer.Country == "UK")**

**racer.Country = "United Kingdom";**

**Console.WriteLine("{0:A}", racer);**

**}**

**Console.WriteLine("查出首发场数超过100，并且胜利场数超过20的车手的姓名（包含名和姓）");**

**var query4 = from r in racers**

**where r.Starts > 100**

**where r.Wins > 25**

**select r;**

**foreach (Racer racer in query4)**

**Console.WriteLine("{0:N}", racer);**

**int num = 0;**

**var query5 = from r in racers**

**where r.Country=="Germany"**

**select r.Wins;**

**Console.WriteLine("德国人的胜利总场数"+query5.Sum());**

**Console.WriteLine("进行投影查询，输出人名与首发胜率，人名包含名和姓，首发胜率为Wins除以Starts");**

**var query6 = from r in racers**

**select new {Name = r.FirstName+" "+r.LastName,Ratio = r.Wins/r.Starts};**

**foreach(var racer in query6)**

**Console.WriteLine(racer);**

**Console.ReadKey();**

**}**

**}**

**}**

**输入文件或目录显示大小**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.IO;**

**namespace ConsoleApplication1**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**string path = Console.ReadLine();**

**if (Directory.Exists(path))**

**{**

**string[] str = Directory.GetDirectories(path);**

**foreach (string s in str)**

**{**

**Console.WriteLine("["+s+"]");**

**}**

**str = Directory.GetFiles(path);**

**foreach (string s in str)**

**{**

**Console.Write(s+" ");**

**OutputFileLength(s);**

**}**

**}**

**else if (File.Exists(path))**

**{**

**OutputFileLength(path);**

**}**

**else**

**Console.WriteLine("输入的路径无效");**

**Console.ReadKey();**

**}**

**static string[] size = { "B", "K", "M", "G", "T" };**

**static void OutputFileLength(string str)**

**{**

**FileInfo file = new FileInfo(str);**

**long len = file.Length;**

**int index = 0;**

**while (len >= 1024)**

**{**

**len = len >> 10;**

**index++;**

**}**

**Console.Write("{0:D3}", len);**

**Console.WriteLine(size[index]);**

**}**

**}**

**}**

**按照目录 文件 行 单词 字符的层次关系，逐步深入**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.IO;**

**namespace ConsoleApplication2**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**SortedDictionary<string, int> stats = new SortedDictionary<string, int>();**

**string path = @"D:\Code";**

**DirectoryInfo di = new DirectoryInfo(path);**

**FileInfo[] files = di.GetFiles();**

**foreach(FileInfo fi in files)**

**{**

**StreamReader sr = new StreamReader(fi.FullName);**

**while (!sr.EndOfStream)**

**{**

**string line;**

**line = sr.ReadLine();**

**string[] word=line.Split(new char[] { ' ', '<', '\_','>','[',']','(',')','=' });**

**foreach (string w in word)**

**{**

**if (w.Length < 4)**

**continue;**

**if (w[0] == 'I')**

**{**

**if (!stats.ContainsKey(w))**

**stats[w] = 1;**

**else**

**{**

**stats[w]++;**

**}**

**}**

**}**

**}**

**foreach(KeyValuePair<string, int>s in stats)**

**Console.WriteLine(s.Key+":"+s.Value);**

**}**

**Console.ReadKey();**

**}**

**}**

**}**

**//数据库**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Data;**

**using System.Data.OleDb;**

**namespace ConsoleApplication1**

**{**

**class Program**

**{**

**static string \_mdbFile = "D:/Northwind.mdb";**

**static OleDbConnection \_connection;**

**static OleDbDataAdapter \_adapter;**

**static DataSet \_dataset;**

**static void Main(string[] args)**

**{**

**while (true)**

**{**

**int index = 1;**

**Console.WriteLine("=====Menu=====");**

**Console.WriteLine(index++ + ". 连接数据库");**

**Console.WriteLine(index++ + ". 查询数据行");**

**Console.WriteLine(index++ + ". 数据统计");**

**Console.WriteLine(index++ + ". 更新数据库");**

**Console.WriteLine(index++ + ". 数据库离线查询");**

**Console.WriteLine(index++ + ". 离线LINQ查询");**

**Console.WriteLine(index++ + ". 断开数据库");**

**Console.WriteLine(index++ + ". 退出");**

**Console.WriteLine("=====End======");**

**Console.Write("请选择一个操作：");**

**string input = Console.ReadLine();**

**int choice;**

**bool isDigit = int.TryParse(input, out choice);**

**if (!isDigit || choice >= index || choice < 1)**

**{**

**Console.WriteLine("错误选项");**

**}**

**if (choice == index - 1)**

**break;**

**switch (choice)**

**{**

**case 1:**

**ConnectToDB();**

**break;**

**case 2:**

**QueryWithReader();**

**break;**

**case 3:**

**QueryScalar();**

**break;**

**case 4:**

**UpdateOnLine();**

**break;**

**case 5:**

**QueryOffLine();**

**break;**

**case 6:**

**LINQQuery();**

**break;**

**case 7:**

**CloseDB();**

**break;**

**}**

**Console.WriteLine();**

**}**

**}**

**public static bool IsConnected()**

**{**

**if (\_connection == null || \_connection.State != ConnectionState.Open)**

**{**

**Console.WriteLine("数据库未连接");**

**return false;**

**}**

**return true;**

**}**

**public static void ConnectToDB()**

**{**

**string connString = @"Provider=Microsoft.ACE.OLEDB.12.0; Data Source=" + \_mdbFile;**

**\_connection = new OleDbConnection(connString);**

**\_connection.Open();**

**try**

**{**

**if (\_connection.State == ConnectionState.Open)**

**{**

**Console.WriteLine("数据库已经连接");**

**}**

**}**

**catch**

**{**

**Console.WriteLine("连接数据库失败");**

**}**

**}**

**public static void QueryWithReader()**

**{**

**if (!IsConnected())**

**return;**

**var command = \_connection.CreateCommand();**

**//补全**

**command.CommandText = "select \* from 客户 where 城市='南京'";**

**command.CommandType = CommandType.Text;**

**OleDbDataReader data = command.ExecuteReader();**

**while (data.Read())**

**{**

**Console.WriteLine(data[0] + " " + data[1] + " " + data[2] + " " + data[3] + " ");**

**}**

**}**

**public static void QueryScalar()**

**{**

**if (!IsConnected())**

**return;**

**var command = \_connection.CreateCommand();**

**command.CommandText = "select count(\*) from 客户 where 城市='南京'";**

**command.CommandType = CommandType.Text;**

**string s = command.ExecuteScalar().ToString();**

**Console.WriteLine("一共查询到了" + s + "行数据");**

**}**

**public static void UpdateOnLine()**

**{**

**if (!IsConnected())**

**return;**

**var command = \_connection.CreateCommand();**

**command.CommandText = "update 客户 set 城市='天津' where 客户ID='ALFKI'";**

**command.CommandType = CommandType.Text;**

**int cnt = command.ExecuteNonQuery();**

**Console.WriteLine("一共修改了" + cnt + "行");**

**}**

**public static void QueryOffLine()**

**{**

**\_adapter = new OleDbDataAdapter();**

**\_dataset = new DataSet();**

**var command = \_connection.CreateCommand();**

**command.CommandText = "select \* from 客户 where 城市='南京'";**

**\_adapter.SelectCommand = command;**

**\_adapter.Fill(\_dataset, "客户");**

**foreach (DataTable table in \_dataset.Tables)**

**{**

**foreach (DataRow row in table.Rows)**

**{**

**foreach (DataColumn column in table.Columns)**

**{**

**Console.WriteLine(row[column]);**

**}**

**Console.WriteLine("");**

**}**

**}**

**}**

**public static void LINQQuery()**

**{**

**DataTable table = \_dataset.Tables[0];**

**IEnumerable<DataRow> rows = from p in table.AsEnumerable() select p;**

**Console.WriteLine("客户ID");**

**foreach (DataRow row in rows)**

**{**

**Console.WriteLine(row.Field<string>("客户ID"));**

**}**

**}**

**public static void CloseDB()**

**{**

**try**

**{**

**string str = "";**

**\_connection.Close();**

**if (\_connection.State == ConnectionState.Closed)**

**{**

**str = "数据库已经成功关闭";**

**}**

**Console.WriteLine(str);**

**}**

**catch**

**{**

**Console.WriteLine("数据库无法关闭");**

**}**

**}**

**}**

**}**

**struct SomeVal { public Int32 x; }**

**class SomeRef { public Int32 x; }**

**SomeVal v1; // 分配到 栈 （堆/栈）**

**//Console.WriteLine(v1.x); //能运行吗？ 不能，v1.x未赋值，要把下一行代码移到前面。**

**v1 = new SomeVal();**

**Console.WriteLine(v1.x); //输出 0**

**v1.x =5;**

**Console.WriteLine(v1.x); //输出 5**

**SomeRef r1;**

**//Console.WriteLine(r1.x); //能运行吗？ 不能，r1.x未赋值，要把下一行代码移到前面。r1 = new SomeRef(); // 分配到 堆 （堆/栈）**

**Console.WriteLine(r1.x); //输出 0**

**r1.x =5;**

**Console.WriteLine(r1.x); //输出 5**

**SomeVal v2 = v1;**

**SomeRef r2 = r1;**

**v1.x = 9;**

**r1.x = 8;**

**Console.WriteLine(r1.x); //输出 8**

**Console.WriteLine(r2.x); //输出 8**

**Console.WriteLine(v1.x); //输出 9**

**Console.WriteLine(v2.x); //输出 5**

**Code1 => Code3 => Code2 => Code5 => Code4**

**【运行顺序检测源代码】**

**using System;**

**namespace 实验二\_第四题**

**{**

**abstract class Server**

**{**

**protected const int DEFAULT\_PORT = 40000;//code1**

**public Server()**

**{**

**Console.WriteLine("code2");**

**int port = getPort();//code2**

**Console.WriteLine("Port: " + port);**

**}**

**protected abstract int getPort();**

**}**

**class SimpleServer : Server**

**{**

**private int port = 100;//code3**

**public SimpleServer(int \_port)**

**{**

**Console.WriteLine("code4");**

**port = \_port;//code4**

**}**

**protected override int getPort()**

**{**

**Console.WriteLine("code5");**

**return (new Random()).NextDouble() > 0.5 ? port : DEFAULT\_PORT;//code5**

**}**

**}**

**class Program**

**{**

**public int i = 0;**

**static void Main(string[] args)**

**{**

**Server s = new SimpleServer(1000);**

**Console.ReadKey();**

**}**

**}**

**}**