Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №3**

Массивы, строки, коллекции, кортежи

тема

Преподаватель \_\_\_\_\_\_\_\_\_ А.А. Чикизов

подпись, дата инициалы, фамилия

Студент КИ18-17/1б 031830504 \_\_\_\_\_\_\_\_\_ Е.В. Железкин

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2020

Содержание

[Содержание 2](#_Toc18956750)

[1 Цель работы 3](#_Toc18956751)

[2 Задача работы 3](#_Toc18956752)

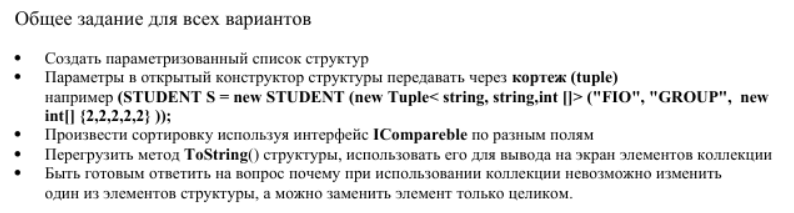
[3 Ход работы 3](#_Toc18956753)

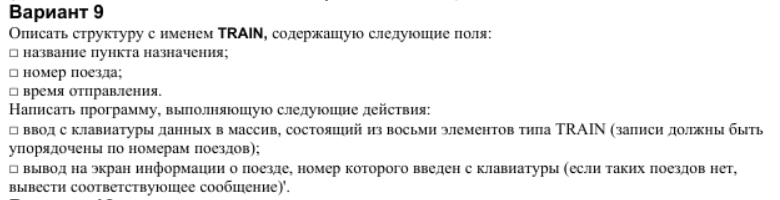
[4 Вывод 6](#_Toc18956754)

# Цель работы

Изучение массивов, строк, коллекций, кортежей.

# Задачи работы





# Ход работы

1. Был написан код лабораторной работы (Приложение А):

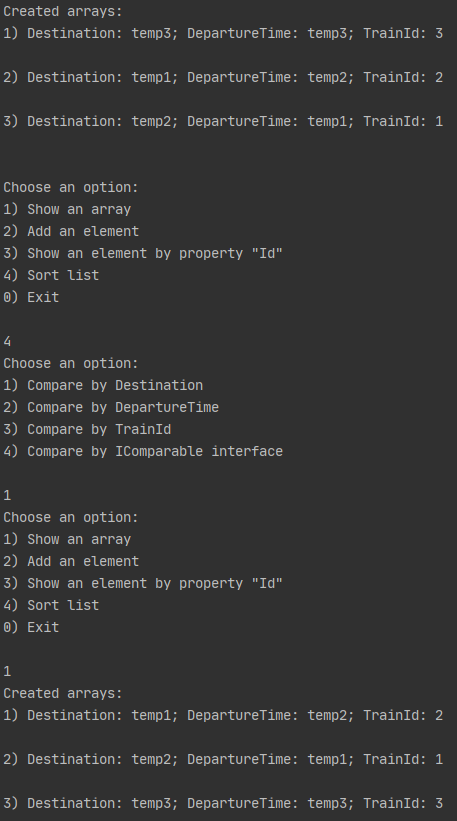


Рисунок 1 – Результат выполнения

# Вывод

В ходе данной лабораторной работы были изучены массивы, строки, коллекции, кортежи.

**ПРИЛОЖЕНИЕ А**

Листинг 1 – файл Program.cs

using System;  
using System.Collections.Generic;  
  
namespace Lab\_3  
{  
 internal static class Program  
 {  
   
 private static List<Train> Array { get; set; }  
  
 static void Main()  
 {  
  
 Array = new List<Train>  
 {  
 new Train(new Tuple<string, string, short>("temp3",  
 "temp3", 3)),  
 new Train(new Tuple<string, string, short>("temp1",  
 "temp2", 2)),  
 new Train(new Tuple<string, string, short>("temp2",  
 "temp1", 1))  
 };  
  
  
 while (true)  
 {  
 Console.Write("Choose an option:\n1) Show an array\n2) Add an element\n" +  
 "3) Show an element by property \"Id\"\n4) Sort list\n0) Exit\n\n");  
  
 var option = 0;  
 try  
 {  
 option = int.Parse(Console.ReadLine()!);  
 }  
 catch (Exception e)  
 {  
 Console.Error.WriteLine(e.Message);  
 }  
  
 switch (option)  
 {  
 case 1:  
 Console.Write("Created arrays: \n");  
 foreach (var item in Array)  
 {  
 Console.WriteLine($"{Array.IndexOf(item) + 1}) {item.ToString()}");  
 }  
 Console.WriteLine();  
 break;  
 case 2:  
 Console.Write("Enter a train destination: ");  
 var tempDestination = Console.ReadLine();  
 Console.Write("Enter a train departure time: ");  
 var tempDepartureTime = Console.ReadLine();  
 Console.Write("Enter a train id: ");  
 var tempTrainId = Convert.ToInt16(Console.ReadLine());  
 try  
 {  
 lock (Array)  
 {  
 Array.Add(new Train(new Tuple<string, string, short>(tempDestination,   
 tempDepartureTime, tempTrainId)));  
 }  
 }  
 catch (Exception e)  
 {  
 Console.Error.WriteLine(e.Message);  
 }  
 break;  
 case 3:  
 Console.Write("Enter a train id: ");  
   
 var inputTrainId = Convert.ToInt16(Console.ReadLine());  
 var flag = true;  
   
 foreach (var train in Array)  
 {  
 if (train.GetTrainInTuple().Item3.Equals(inputTrainId))  
 {  
 Console.Write(train.ToString());  
 flag = false;  
 }  
 }  
  
 if (flag)  
 {  
 Console.Write("Specified ID not found!\n\n");  
 }  
 break;  
 case 4:  
 try  
 {  
 Console.Write("Choose an option:\n1) Compare by Destination\n2) Compare by DepartureTime\n" +  
 "3) Compare by TrainId\n4) Compare by IComparable interface\n\n");  
   
 var sortOption = int.Parse(Console.ReadLine()!);  
 switch (sortOption)  
 {  
 case 1:  
 Array.Sort(Train.CompareByDestination);  
 break;  
 case 2:  
 Array.Sort(Train.CompareByDepartureTime);  
 break;  
 case 3:  
 Array.Sort(Train.CompareByTrainId);  
 break;  
 case 4:  
 Array.Sort();  
 break;  
 default:  
 continue;  
 }  
 }  
 catch (Exception e)  
 {  
 Console.WriteLine(e.Message);  
 }  
 break;  
 case 0:  
 return;  
 }  
 }  
 }  
 }  
}

Листинг 2 – файл Train.cs

using System;  
  
namespace Lab\_3  
{  
 public readonly struct Train : IComparable  
 {  
 private string Destination { get; }  
 private string DepartureTime { get; }  
 private short TrainId { get; }  
  
 public Train(Tuple<string, string, short> args)  
 {  
 var (item1, item2, item3) = args;  
 Destination = item1;  
 DepartureTime = item2;  
 TrainId = item3;  
 }  
  
 public Tuple<string, string, short> GetTrainInTuple()  
 {  
 return new Tuple<string, string, short>(this.Destination, this.DepartureTime, this.TrainId);  
 }  
  
 public override string ToString()  
 {  
 return $"Destination: {Destination}; DepartureTime: " +  
 $"{DepartureTime}; TrainId: {TrainId}\n";  
 }  
  
 public int CompareTo(object obj)  
 {  
 if (obj == null) return 1;  
   
 if (obj is Train temp)  
 {  
 var (tempDestination, tempDepartureTime, tempTrainId) = temp.GetTrainInTuple();  
  
 if (TrainId.CompareTo(tempTrainId) != 0) return TrainId.CompareTo(tempTrainId);  
   
 if (string.Compare(Destination, tempDestination, StringComparison.*Ordinal*) != 0)  
 return string.Compare(Destination, tempDestination, StringComparison.*Ordinal*);  
   
 if (string.Compare(DepartureTime, tempDepartureTime, StringComparison.*Ordinal*) != 0)  
 return string.Compare(DepartureTime, tempDepartureTime, StringComparison.*Ordinal*);  
 }  
 else  
 throw new ArgumentException("Object is not a Train");  
  
 return 0;  
 }  
  
 public static int CompareByDestination(Train train1, Train train2)  
 {  
 return !train1.GetTrainInTuple().Item1.Equals(train2.GetTrainInTuple().Item1) ?   
 string.CompareOrdinal(train1.GetTrainInTuple().Item1, train2.GetTrainInTuple().Item1) : 0;  
 }  
   
 public static int CompareByDepartureTime(Train train1, Train train2)  
 {  
 return !train1.GetTrainInTuple().Item2.Equals(train2.GetTrainInTuple().Item2) ?   
 string.CompareOrdinal(train1.GetTrainInTuple().Item2, train2.GetTrainInTuple().Item2) : 0;  
 }  
  
 public static int CompareByTrainId(Train train1, Train train2)  
 {  
 return train1.GetTrainInTuple().Item3.CompareTo(train2.GetTrainInTuple().Item3);  
 }  
   
 }  
}