```
using
CsvHelper;
              using CsvHelper.Configuration;
              using System;
              using System.Collections.Generic;
              using System.Globalization;
              using System.IO;
              using System.Linq;
              using System.Text;

              namespace App
              {
                  class Program
                  {
                      static void Main(string[] args)
                      {
                          // uložení hodnoty času spuštění programu, pro diagnostické
              účely
                          var time = DateTime.Now;
                          // vytvoření instance globálního slovníku, kde kličem je icao
              stroje a hodnotou je kolekce sumářů aktivity stroje
                          var data = new Dictionary<string, List<EventData>>();
                          // počítadlo zpracovaných řádků
                          var rows = 0;

                          // iterace kolekcí souborů s patternem states_*.csv
                          foreach (var file in System.IO.Directory.GetFiles(@"i:\Michal",
              "states_*.csv").OrderBy(x => x))
                              // zvýšení počtu zpracovaných řádků o hodnotu vrácenou
              funkcí pro zpracování souboru s raw daty letového provozu
                              rows += ProcessFile(file, data);

                          // uložení slovníku do souboru result.csv
                          SaveData(@"c:\Temp\MichalApp\data\result.csv", data);

                          // informace na std. výstup
                          Console.WriteLine("Zpracování trvalo: {0}",
              DateTime.Now.Subtract(time));
                          Console.WriteLine("Celkem řádků: {0}", rows);
                          Console.WriteLine("Letů s přistáním celkem: {0}", data.Count);
                          Console.WriteLine("Všech přistáním celkem: {0}", data.Sum(x =>
              x.Value.Count(y => y.onground)));
                          Console.WriteLine("Všech letů celkem: {0}", data.Sum(x =>
              x.Value.Count(y => !y.onground)));
```

```
            Console.WriteLine("Průměrná délka přistáním: {0} minut",
data.SelectMany(x => x.Value.Where(y => y.onground).Select(y =>
y.till.Subtract(y.from).TotalMinutes)).Average());
            Console.WriteLine("Průměrná délka letu: {0} minut",
data.SelectMany(x => x.Value.Where(y => !y.onground).Select(y =>
y.till.Subtract(y.from).TotalMinutes)).Average());
            Console.WriteLine("Top 10 podle počtu přistání: {0}",
String.Join(", ", data.OrderByDescending(x => x.Value.Count(y =>
y.onground)).Select(x => x.Key).Take(10)));
            Console.WriteLine("Hit ENTER to exit...");
            Console.ReadLine();
        }


        // funkce pro zápis globálního slovníku všech sumářů do souboru
        static void SaveData(string file, Dictionary<string,
List<EventData>> data)
        {
            // informace na std. výstup
            Console.WriteLine("Zapisuji do souboru: {0}", file);

            // instancování objektu konfigurace pro správne nastavení
parametrů zápisu do výstupního souboru typu CSV
            var config = new CsvConfiguration(new CultureInfo("en-US"))
            {
                HasHeaderRecord = true,
                Delimiter = ",",
            };

            // vytvoření instance objektu pro zápis do proudu souboru
            using (var writer = new StreamWriter(file, false,
Encoding.UTF8))
            // vytvoření instance objektu pro zápis CSV souboru s
požadovanou konfigurací
            using (var csv = new CsvWriter(writer, config))
            {
                // linearizace sumářů z globálního slovníku
                var records = data.SelectMany(x => x.Value)
                    .Select(x => new EventRecord
                    {
                        callsign = x.onground ? null : x.callsign.Trim(),
                        icao24 = x.icao24,
                        from =
(int)Math.Floor(x.from.Subtract(DateTime.UnixEpoch).TotalSeconds),
                        till =
(int)Math.Floor(x.till.Subtract(DateTime.UnixEpoch).TotalSeconds),
```

```
                        onground = x.onground,
                        lat = x.lat,
                        lon = x.lon,
                        latmax = x.latmax,
                        latmin = x.latmin,
                        lonmax = x.lonmax,
                        lonmin = x.lonmin,
                        latfirst = x.latfirst,
                        lonfirst = x.lonfirst,
                        latlast = x.latlast,
                        lonlast = x.lonlast,
                        ByDay = x.byDay,
                        ByNight = x.byNight,
                        Distance = x.distance,
                        ByAfternoon = x.byAfternoon,
                        ByMorning = x.byMorning,
                    });

                // zápis hlavičky sloupců do souboru
                csv.WriteHeader<EventRecord>();
                // přesun na další řádek
                csv.NextRecord();
                // itarace kolekcí linearizovaných sumářů
                foreach (var record in records)
                {
                    // zápis řádku sumáře
                    csv.WriteRecord(record);
                    // přesun na další řádek
                    csv.NextRecord();
                }
            }

            // informace na std. výstup
            Console.WriteLine();
            Console.WriteLine("-- Dokončeno --");
        }


        // Funkce pro zpracování souboru s raw daty letového provozu, jehož
název je předaný paramertem file
        // druhý parametr data je globální slovník, kde kličem je icao
stroje a hodnotou je kolekce sumářů aktivity stroje
        static int ProcessFile(string file, Dictionary<string,
List<EventData>> data)
        {
            // počítadlo zpracovaných řádků
```

```
            var rows = 0;
            // informace na std. výstup
            Console.WriteLine("Zpracovávám soubor: {0}", file);
            // instancování objektu konfigurace pro správne nastavení
parametrů čtení ze vstupního souboru typu CSV
            var config = new CsvConfiguration(new CultureInfo("en-US"))
            {
                // první řádek je hlavička - názvy sloupců
                HasHeaderRecord = true,
                // oddělovačem sloupců je čárka
                Delimiter = ",",
            };

            // vytvoření instance objektu pro čtení z proudu ze souboru
            using (var reader = new StreamReader(file, Encoding.UTF8))
            // vytvoření instance objektu pro čtení CSV souboru s
požadovanou konfigurací
            using (var csv = new CsvReader(reader, config))
            {
                // instancování prázdného modelu který zrcadlí strukturu
CSV souboru
                var record = new State();
                // vytvoření instance a naplnění kolekce ze souboru file
                var records = csv.EnumerateRecords(record);
                // iterace kolekcí záznamů, kde vlastnosti modelu r
odpovídají sloupcům v CSV souboru
                foreach (var r in records)
                {
                    // preskoc záznamy, kde příznak onground nemá hodnotu
                    if (!r.onground.HasValue)
                        continue;
                    // přeskoč záznam, kde příznak polohy (lat, lon) nemá
hodnotu
                    if (!r.lat.HasValue || !r.lon.HasValue)
                        continue;

                    // pokud ve slovníku data neexistuje klíč icao, tak
bude založen a hodnotou je prázdná kolekce sumářů aktivit
                    if (!data.ContainsKey(r.icao24))
                        data[r.icao24] = new List<EventData>();

                    // do proměnné og ulož poslední zpracovaný sumář
aktivity
                    var og = data[r.icao24].LastOrDefault();
```

```
                    // pokud poslední sumář aktivity pro aktuálně
zpracovávané icao neexistuje
                    if (og == null)
                    {
                        // příznak aktuaktuálního záznamu je na zemi -
počátek je vždy na zemi
                        if (r.onground.Value)
                        {
                            // informace na std. výstup
                            Console.Write(".");
                            // do kolekce je založen nový sumář s aktuálním
stavem
                            AddNewOnGround(data, r);
                        }
                    }

                    // jinak pokud se od posledniho vyskytu zmenil onground
                    else if (og.onground != r.onground.Value)
                    {
                        //  informace na std. výstup
                        Console.Write(r.onground.Value ? "_" : "^");
                        // poslední sumář je označen za ukončený
                        og.open = false;
                        // do kolekce je založen nový sumář s aktuálním
stavem
                        AddNewOnGround(data, r);
                    }

                    // jinak pokud jse záznam otevřený
                    else if (og.open)
                    {
                        // do diffDays je uložen pořet dní od posledního
sumáře
                        var diffDays =
DateTime.UnixEpoch.AddSeconds(r.time).Subtract(og.from).TotalDays;

                        // pokud je rozdíl víc než den - jde o přechod mezi
nespojitými daty
                        if (diffDays > 1)
                        {
                            //  informace na std. výstup
                            Console.Write("X");
                            // poslední sumář je označen za ukončený
                            og.open = false;
                            // do kolekce je založen nový sumář s aktuálním
stavem
```

```
                            AddNewOnGround(data, r);
                        }
                        // jinak je sumář otevřeny a je možné ho
aktualizovat
                        else
                        {
                            // prodlouží čas trvání sumáře
                            og.till =
DateTime.UnixEpoch.AddSeconds(r.time);

                            // pokud má záznam polohu
                            if (r.lat.HasValue && r.lon.HasValue)
                            {
                                // pokud je to první naplnění sumáře
                                if (og.count == 0)
                                {
                                    // nastavení polohy
                                    og.lat = r.lat;
                                    og.lon = r.lon;
                                    // nastavení nejmenšího "obdélníku", ve
kterém aktivita probíhala
                                    og.latmax = r.lat;
                                    og.latmin = r.lat;
                                    og.lonmax = r.lon;
                                    og.lonmin = r.lon;
                                    // nastavení první známe polohy
                                    og.latfirst = r.lat;
                                    og.lonfirst = r.lon;
                                    // nastavení poslední známé polohy
                                    og.latlast = r.lat;
                                    og.lonlast = r.lon;
                                }
                                // jinak jde o další aktualizaci polohy
                                else
                                {
                                    // rozšíření nejmenšího "obdélníku", ve
kterém aktivita probíhala
                                    og.latmax = Math.Max(og.latmax.Value,
r.lat.Value);
                                    og.latmin = Math.Min(og.latmin.Value,
r.lat.Value);
                                    og.lonmax = Math.Max(og.lonmax.Value,
r.lon.Value);
                                    og.lonmin = Math.Min(og.lonmin.Value,
r.lon.Value);
```

```
                                    // nastavení polohy - představuje
průměr
                                    og.lat = (og.lat.Value * (og.count - 1)
+ r.lat.Value) / og.count;
                                    og.lon = (og.lon.Value * (og.count - 1)
+ r.lon.Value) / og.count;
                                    // aktualizace poslední známé polohy
                                    og.latlast = r.lat;
                                    og.lonlast = r.lon;
                                }

                                // počítadlo počtu zpracovavných záznamů
                                og.count++;

                                // pokud jde o každé 6. zpracování - tj.
cca každou minutu
                                if (og.count % 6 == 0)
                                {
                                    // pokud jsou hodnoty dočasné polohy
dostupné
                                    if (og.latTmp.HasValue &&
og.lonTmp.HasValue)
                                    {
                                        // inkrementuje překonanou
vzdálenost o rozdíl mezi aktuální a dočasnou polohou
                                        og.distance +=
Geolocation.GeoCalculator.GetDistance(og.latTmp.Value, og.lonTmp.Value,
r.lat.Value, r.lon.Value, 1, Geolocation.DistanceUnit.Meters);
                                        // dočasnou polohu nastaví na
aktuální
                                        og.latTmp = r.lat;
                                        og.lonTmp = r.lon;
                                    }
                                }

                                // pokud jde o každé 180. zpracování - tj.
cca každou půlhodinu
                                if (og.count % 180 == 0)
                                    // aktualizuj kontextové inkrementy
závislé na poloze
                                    UpdateIncrements(r, og);

                                // dočasná poloha nemá hodnotu
                                if (!og.latTmp.HasValue &&
!og.lonTmp.HasValue)
                                {
```

```
                                    // dočasnou polohu nastaví na aktuální
                                    og.latTmp = r.lat;
                                    og.lonTmp = r.lon;
                                    // aktualizuj kontextové inkrementy
závislé na poloze
                                    UpdateIncrements(r, og);
                                }
                            }
                        }
                    }

                    else
                    {
                        //  informace na std. výstup
                        Console.Write("-");
                        // do kolekce je založen nový sumář s aktuálním
stavem
                        AddNewOnGround(data, r);
                    }

                    // počítadlo řádků je zvednuto o 1
                    rows++;
                }
            }

            //  informace na std. výstup
            Console.WriteLine();
            Console.WriteLine("-- Dokončeno --");

            // vrací počet zpracovaných řádků
            return rows;
        }


        // fukce pro založení nového sumáře z proměnné r
        static void AddNewOnGround(Dictionary<string, List<EventData>>
data, State r)
        {
            // vytvoření a nastavení hodnot nového sumáře
            var e = new EventData
            {
                icao24 = r.icao24,
                from = DateTime.UnixEpoch.AddSeconds(r.time),
                till = DateTime.UnixEpoch.AddSeconds(r.time),
                open = true,
                callsign = r.callsign,
```

```
                lat = r.lat,
                lon = r.lon,
                latmax = r.lat,
                latmin = r.lat,
                lonmax = r.lon,
                lonmin = r.lon,
                latfirst = r.lat,
                lonfirst = r.lon,
                latlast = r.lat,
                lonlast = r.lon,
                latTmp = r.lat,
                lonTmp = r.lon,
                count = r.lat.HasValue && r.lon.HasValue ? 1 : 0,
                onground = r.onground.Value,
            };

            // pokud má polohu
            if (r.lat.HasValue && r.lon.HasValue)
                // aktualizuj kontextové inkrementy závislé na poloze
                UpdateIncrements(r, e);

            // do kolekce přidá nový sumář
            data[r.icao24].Add(e);
        }


        // funkce pro aktualizaci inkrementů vázaných na polohu
        static void UpdateIncrements(State r, EventData e)
        {
            // z unixoveho krátkého času vytvoří UTC
            var time = DateTime.UnixEpoch.AddSeconds(r.time);
            // vytvoření instance objektu, který nese informace o poloze
Slunce na souřadnicích k UTC
            var cel = new CoordinateSharp.Coordinate(r.lat.Value,
r.lon.Value, time);
            // pokud je Slunce nad horizontem
            if (cel.CelestialInfo.IsSunUp)
            {
                // pokud je aktuální čas menší než čas pro poledne na
souřadnicích
                if (cel.CelestialInfo.SolarNoon > time)
                    // počítadlo dopoledních intervalů zvednuto o 1
                    e.byMorning++;
                else
                    // počítadlo odpoledních intervalů zvednuto o 1
                    e.byAfternoon++;
```

```
                // počítadlo denních intervalů zvednuto o 1
                e.byDay++;
            }
            else
                // počítadlo nočních intervalů je zvednuto o 1
                e.byNight++;
        }
    }
}
```