Veiksmai su objektais vienkrypčiame sąraše

- Sąrašo klasė.
- Formuojamas atvirkštinis sąrašas.
- Spausdinamos sąrašo elementų reikšmės.
- Formuojamas naujas sąrašas.
- Rikiuojamas sąrašas.

Užduotis.

Tekstiniuose failuose "`Darius.txt`" ir "`Mikas.txt`" yra duomenys apie dviejų studentų turimus mobiliosios elektronikos įrenginius. Pirmoje eilutėje yra studento vardas ir pavardė. Kitose eilutėse yra informacija apie įrenginius: modelis, tipas, baterijos veikimo trukmė. Parašykite programą, kuri:

- surastų kiekvieno studento vieną ilgiausiai veikiantį įrenginį,
- suformuotų pasirinkto tipo įrenginių sąrašą.

Kiekvieną sąrašą surikiuokite pagal veikimo trukmę ir modelį abėcėliškai.

Duomenų failo `Darius.txt` pavyzdys

```
Darius Elektronikas
Sony Ericsson K610i;Mobilusis telefonas;  480
Motorola Razor V3;Mobilusis telefonas;    400
Creative Zen;MP3 grotuvas;                40
Apple iPod Shuffle;MP3 grotuvas;           40
```

Duomenų failo `Mikas.txt` pavyzdys

```
Mikas Humanitaras
Motorola W156;Mobilusis telefonas;    465
Nokia 1200;Mobilusis telefonas;       390
Samsung B520;Mobilusis telefonas;     350
Sony Walkman NWZ-A826B;MP3 grotuvas;   45
Sony Walkman NWZ-E436FB;MP3 grotuvas;  45
Garmin Oregon 300;Navigacine sistema;  16
```

Rezultatų failo pavyzdys

```
Mikas Humanitaras
+------------------------------+----------------------+--------------+
|             Modelis          |        Tipas         | Veik. trukmė |
+------------------------------+----------------------+--------------+
|Garmin Oregon 300             | Navigacine sistema   |           16 |
|Sony Walkman NWZ-E436FB       | MP3 grotuvas         |           45 |
|Sony Walkman NWZ-A826B        | MP3 grotuvas         |           45 |
|Samsung B520                  | Mobilusis telefonas  |          350 |
|Nokia 1200                    | Mobilusis telefonas  |          390 |
|Motorola W156                 | Mobilusis telefonas  |          465 |
+------------------------------+----------------------+--------------+

Darius Elektronikas
+------------------------------+----------------------+--------------+
|             Modelis          |        Tipas         | Veik. trukmė |
+------------------------------+----------------------+--------------+
|Apple iPod Shuffle            | MP3 grotuvas         |           40 |
|Creative Zen                  | MP3 grotuvas         |           40 |
|Motorola Razor V3             | Mobilusis telefonas  |          400 |
|Sony Ericsson K610i           | Mobilusis telefonas  |          480 |
+------------------------------+----------------------+--------------+

Studentas: Mikas Humanitaras, ilgiausiai veikianti baterija
  modelis: Motorola W156, tipas:  Mobilusis telefonas, trukmė: 465.

Studentas: Darius Elektronikas, ilgiausiai veikianti baterija
  modelis: Sony Ericsson K610i, tipas:  Mobilusis telefonas, trukmė: 480.
Atrinkti nerikiuoti
+------------------------------+----------------------+--------------+
|            Modelis           |        Tipas         | Veik. Trukmė |
+------------------------------+----------------------+--------------+
|Sony Ericsson K610i           | Mobilusis telefonas  |          480 |
|Motorola Razor V3             | Mobilusis telefonas  |          400 |
```

```
|Motorola W156                 | Mobilusis telefonas  |          465 |
|Nokia 1200                    | Mobilusis telefonas  |          390 |
|Samsung B520                  | Mobilusis telefonas  |          350 |
+------------------------------+----------------------+--------------+

Atrinkti surikiuoti
+------------------------------+----------------------+--------------+
|             Modelis          |        Tipas         | Veik. trukmė |
+------------------------------+----------------------+--------------+
|Samsung B520                  | Mobilusis telefonas  |          350 |
|Nokia 1200                    | Mobilusis telefonas  |          390 |
|Motorola Razor V3             | Mobilusis telefonas  |          400 |
|Motorola W156                 | Mobilusis telefonas  |          465 |
|Sony Ericsson K610i           | Mobilusis telefonas  |          480 |
+------------------------------+----------------------+--------------+

Rimas Kurpius
+------------------------------+----------------------+--------------+
|             Modelis          |        Tipas         | Veik. trukmė |
+------------------------------+----------------------+--------------+
|Motorola Razor V4             | Mobilusis telefonas  |     490,00   |
|Tavo                          | Mobilusis telefonas  |     300,00   |
|Sony Ericsson K6109           | Mobilusis telefonas  |     455,00   |
|Apple iPod Shuffle            | Grotuvas             |     400,00   |
+------------------------------+----------------------+--------------+

Rikiuotas po papildymo
+------------------------------+----------------------+--------------+
|             Modelis          |        Tipas         | Veik. trukmė |
+------------------------------+----------------------+--------------+
|Tavo                          | Mobilusis telefonas  |     300,00   |
|Samsung B520                  | Mobilusis telefonas  |     350,00   |
|Nokia 1200                    | Mobilusis telefonas  |     390,00   |
|Motorola Razor V3             | Mobilusis telefonas  |     400,00   |
|Sony Ericsson K6109           | Mobilusis telefonas  |     455,00   |
|Motorola W156                 | Mobilusis telefonas  |     465,00   |
|Sony Ericsson K610i           | Mobilusis telefonas  |     480,00   |
|Motorola Razor V4             | Mobilusis telefonas  |     490,00   |
+------------------------------+----------------------+--------------+
```

Programos kūrimo eiga.

- Sukuriama įrenginio klasė.
- Sukuriama sąrašo klasė.
- Formuojami du atvirkštiniai sąrašai.
- Sąrašų duomenys spausdinami lentelėmis.
- Atliekami skaičiavimai.
- Formuojamas naujas sąrašas.
- Rikiuojamas sąrašas.

Pirmas žingsnis.

- Parašykite klasę įrenginio duomenims saugoti ir apdoroti. Užrašykite užklotus operatorius, kuriuos vėliau panaudosite metoduose.

```
//-------------------------------------------------------------
// Mobiliojo įrenginio duomenų saugojimo klasė
public class Mobilus
{   // mobiliosios elektronikos įrenginys
    public string modelis { get; set; }  // modelio pavadinimas
    public string tipas  { get; set; }   // įrenginio tipas
    public int baterija  { get; set; }   // baterijos veikimo trukmė

    //  konstruktorius
    public Mobilus(string modelis ="", string tipas = "", int baterija = 0)
    {
```

```
        this.modelis = modelis;
        this.tipas = tipas;
        this.baterija = baterija;
    }

    //  objekto naujos reikšmės
    // a – modelio pavadinimas
    // b – įrenginio tipas
    // c – baterijos veikimo trukmė
    public void Dėti(string a, string b, int c)
    {
        modelis = a;
        tipas = b;
        baterija = c;
    }

    public override string ToString()
    {
        string eilute;
        eilute = string.Format("|{0, -30}| {1, -20} |   {2, 8:f}   |",
                 modelis, tipas, baterija);
        return eilute;
    }

    public override bool Equals(object objektas)
    {
        Mobilus telef = objektas as Mobilus;
        return telef.tipas == tipas && telef.modelis == modelis && telef.baterija ==
               baterija;
    }

    // Užklotas metodas GetHashCode()
    public override int GetHashCode()
    {
        return base.GetHashCode();
    }

    // Užklotas operatorius >= (dviejų įrenginių palyginimui pagal baterijos veikimo trukmę
    // ir modelio pavadinimą)
    public static bool operator >=(Mobilus pirmas, Mobilus antras)
    {
        int poz = String.Compare(pirmas.modelis, antras.modelis,
                  StringComparison.CurrentCulture);
        return pirmas.baterija > antras.baterija || pirmas.baterija == antras.baterija &&
                poz > 0;
    }

    // Užklotas operatorius <= (dviejų įrenginių palyginimui pagal baterijos veikimo trukmę
    // ir modelio pavadinimą)
    public static bool operator <=(Mobilus pirmas, Mobilus antras)
    {
        int poz = String.Compare(pirmas.modelis, antras.modelis,
                  StringComparison.CurrentCulture);
        return pirmas.baterija < antras.baterija || pirmas.baterija == antras.baterija &&
               poz < 0;
    }

    // Užklotas operatorius == (įrenginių tipui palyginti)
    public static bool operator ==(Mobilus pirmas, Mobilus antras)
    {
        return pirmas.tipas == antras.tipas;
    }

    // Užklotas operatorius != (įrenginių tipui palyginti)
    public static bool operator !=(Mobilus pirmas, Mobilus antras)
    {
        return pirmas.tipas != antras.tipas;
    }
```

```
    // Užklotas operatorius > (dviejų įrenginių palyginimui pagal baterijos veikimo trukmę)
    public static bool operator >(Mobilus pirmas, Mobilus antras)
    {
        return pirmas.baterija > antras.baterija;
    }

    // Užklotas operatorius < (dviejų įrenginių palyginimui pagal baterijos veikimo trukmę)
    public static bool operator <(Mobilus pirmas, Mobilus antras)
    {
        return pirmas.baterija < antras.baterija;
    }
}
//---------------------------------------------------------------
```

- Parašykite pagrindinį metodą `Main()` sukurtos klasės teisingumui patikrinti. Tam skelbkite porą objektų, kuriems suteikite reikšmes, panaudodami konstruktorių. Spausdinkite objektų reikšmes ir objektų palyginimo rezultatą, įsitikindami, kad klasės metodai dirba teisingai. Panaudodami metodą `Dėti()` pakeiskite objektuose reikšmes. Patikrinkite spausdindami objektų reikšmes ir jų palyginimo rezultatą.

Antras žingsnis.

- Sukurkite sąrašo elemento mazgą.

```
//---------------------------------------------------------------
public sealed class Mazgas
{
    public Mobilus Duomenys { get; set; }
    public Mazgas Kitas { get; set; }
    public Mazgas() { }
    public Mazgas(Mobilus duomenys, Mazgas adresas)
    {
        this.Duomenys = duomenys;
        this.Kitas = adresas;
    }
}
//---------------------------------------------------------------
```

- Parašykite klasę studento įrenginiams saugoti. Ši klasė **<u>nėra konteinerinė klasė</u>** (pagalvokite kodėl). Dažnai sąrašo klasės būna konteinerinės.

```
//---------------------------------------------------------------
// Mobiliųjų įrenginių vienkryptis sąrašas
public sealed class Studentas
{
    public string pv { get; set; }        // studento pavardė ir vardas
    Mazgas pr;      // sąrašo pradžios nuoroda
    Mazgas d;       // sąsajos nuoroda

    // Konstruktorius be parametrų
    public Studentas()
    {
        this.pr = null;
        this.d = null;
    }

    // Sąsajos metodai
    // Sąsajai priskiriama sąrašo pradžia
    public void Pradžia()
    {
        d = pr;
    }
```

```
    // Sąsajai priskiriamas sąrašo sekantis elementas
    public void Kitas()
    {
        d = d.Kitas;
    }

    // Grąžina true, jeigu sąsaja netuščia; false - priešingu atveju
    public bool Yra()
    {
        return d != null;
    }

    // Grąžina pagalbinės rodyklės rodomo elemento reikšmę
    public Mobilus ImtiDuomenis() { return d.Duomenys; }

    // Sukuriamas sąrašo elementas ir prijungiamas prie sąrašo PRADŽIOS
    // inf – naujo elemento reikšmė (duomenys)
    public void DėtiDuomenisA(Mobilus inf)
    {
        var d = new Mazgas(inf, null);
        d.Kitas = pr;
        pr = d;
        // arba
        //pr = new Mazgas(inf, pr);
    }

    // Sunaikinamas sąrašas
    public void Naikinti()
    {
        while (pr != null)
        {
            d = pr;
            // sąrašo elementų duomenų dalyje yra nuoroda į objektą,
            // ten taip pat reikėtų įrašyti reikšmę null.
            // pr.Duomenys = null;
            pr = pr.Kitas;
            d = null;
        }
        d = pr;
    }
}
//-------------------------------------------------------------
```

- Parašykite pagrindinio metodo `Main()` fragmentą sąrašo klasės metodų patikrinimui.

```
//-------------------------------------------------------------
class Program
{
    const string CFd1 = "..\\..\\Mikas.txt";
    const string CFd2 = "..\\..\\Darius.txt";
    const string CFr = "..\\..\\Rezultatai.txt";

    static void Main(string[] args)
    {
        Studentas A,        // pirmojo studento duomenys
                  B;        // antrojo studento duomenys
        A = SkaitytiAtv(CFd1);
        B = SkaitytiAtv(CFd2);

        if (File.Exists(CFr))
            File.Delete(CFr);

        Spausdinti (CFr, A, A.pv);
        Spausdinti (CFr, B, B.pv);

    }

    // Skaitomi duomenys iš failo ir sudedami į sąrašą ATVIRKŠTINE tvarka
```

```
    // fv – duomenų failo vardas
    static Studentas SkaitytiAtv(string fv)
    {
        var A = new Studentas();
        using (var failas = new StreamReader(fv))
        {
            string eilute;
            A.pv = eilute = failas.ReadLine();
            while ((eilute = failas.ReadLine()) != null)
            {
                string[] eilDalis = eilute.Split(';');
                string modelis = eilDalis[0];
                string tipas = eilDalis[1];
                int baterija = int.Parse(eilDalis[2]);
                Mobilus elem = new Mobilus(modelis, tipas, baterija);
                A.DėtiDuomenisA(elem);
            }
        }
        return A;
    }


    // Sąrašo duomenys spausdinami faile
    // fv – duomenų failo vardas
    // A  - sąrašo objekto nuoroda
    // koment  - komentaras
    static void Spausdinti(string fv, Studentas A, string koment)
    {
        using (var failas = new StreamWriter(fv, true))
        {
            failas.WriteLine(koment);
            failas.WriteLine("+------------------------------+--------------" +
                             "-------+--------------+");
            failas.WriteLine("|              Modelis          |        Tipas  " +
                             "       | Veik. trukmė |");
            failas.WriteLine("+------------------------------+--------------" +
                             "-------+--------------+");

            // Sąrašo peržiūra, panaudojant sąsajos metodus
            for (A.Pradžia(); A.Yra(); A.Kitas())
            {
                failas.WriteLine("{0}", A.ImtiDuomenis().ToString());
            }
            failas.WriteLine("+------------------------------+--------------" +
                             "-------+--------------+");
            failas.WriteLine();
        }
    }
}
//---------------------------------------------------------------
```

- Patikrinkite, ar duomenys gerai surašomi į sąrašą, ar sėkmingai spausdinami faile.

Trečias žingsnis.

- Papildykite sąrašo klasę rikiavimo išrinkimo būdu metodu:

```
//---------------------------------------------------------------
// Rikiavimas išrinkimo būdu
public void Rikiuoti()
{
    for(Mazgas d1 = pr; d1.Kitas != null; d1 = d1.Kitas)
    {
        Mazgas maxv= d1;
        for(Mazgas d2 = d1; d2 != null; d2 = d2.Kitas)
              if(d2.Duomenys <= maxv.Duomenys)
                maxv= d2;
        Mobilus St = d1.Duomenys;
        d1.Duomenys = maxv.Duomenys;
```

```
            maxv.Duomenys= St;
        }
    }
    //-------------------------------------------------------------
```

- Patikrinkite, ar sąrašai gerai rikiuojami. Tam pagrindiniame metode `Main()` parašykite sakinius, kuriais būtų duomenų sąrašai rikiuojami ir spausdinami. Po to tuos sakinius išmeskite.

Ketvirtas žingsnis.

- Papildykite sąrašo klasę metodu, kuris surastų sąraše įrenginį, kurio veikimo trukmė didžiausia.

```
//-------------------------------------------------------------
// Suranda ir grąžina ilgiausiai veikiančio įrenginio duomenis
public Mobilus MaxTrukmė()
{
    Mobilus max;
    max = pr.Duomenys;
    for (Mazgas d1 = pr; d1 != null; d1 = d1.Kitas)
        if (d1.Duomenys > max)
            max = d1.Duomenys;
    return max;
}
//-------------------------------------------------------------
```

- Patikrinkite, ar gerai surandami kiekviename iš sąrašų įrenginiai, kurių veikimo trukmė didžiausia. Tam pagrindiniame metode `Main()` parašykite sakinius, kuriais būtų randami įrenginiai ir gauti rezultatai spausdinami. Tai gali būti padaroma taip:

```
//-------------------------------------------------------------
static void Main(string[] args)
{
    Studentas A,        // pirmojo studento duomenys
              B;        // antrojo studento duomenys
    A = SkaitytiAtv(CFd1);
    B = SkaitytiAtv(CFd2);

    if (File.Exists(CFr))
        File.Delete(CFr);

    Spausdinti (CFr, A, A.pv);
    Spausdinti (CFr, B, B.pv);
    using (var failas = new StreamWriter(CFr, true))
    {
         Mobilus max;
         max = A.MaxTrukmė();
         failas.WriteLine("Studentas: {0}, ilgiausiai veikianti baterija  \r\n  modelis: " +
                          "{1}, tipas:  {2}, trukmė: {3}.",
                          A.pv, max.modelis, max.tipas, max.baterija);
         failas.WriteLine();
         max = B.MaxTrukmė();
         failas.WriteLine("Studentas: {0}, ilgiausiai veikianti baterija  \r\n  modelis: " +
                          " {1}, tipas:  {2}, trukmė: {3}.",
                          B.pv, max.modelis, max.tipas, max.baterija);
    }
}
//-------------------------------------------------------------
```

Penktas žingsnis.

- Papildykite sąrašo klasę metodu, kuris sąrašą papildo nauju elementu

```
//-------------------------------------------------------------
// Į sąrašo pradžią įterpia naują elementą ir įrašo į jį duom
// duom – įrenginių sąrašas papildomas nauju objektu
public void Papildyti(Mobilus duom)
{
    Mazgas d1 = new Mazgas();
    d1.Duomenys = duom;
    d1.Kitas = pr;
```

```
    pr = d1;
}
//-------------------------------------------------------------
```

- Parašykite pagrindinėje klasėje `Program` metodą naujo sąrašo formavimui, kuris atrenka nurodytas savybes turinčius objektus iš nurodyto sąrašo į naują.

```
//-------------------------------------------------------------
// Iš sąrašo senas kopijuoja objektus į sąrašą naujas
// senas   įrenginių sąrašas
// tipas   atrenkamų įrenginių tipas
// naujas  naujo objektų sąrašo adresas
static void Atrinkti(Studentas senas, string tipas, Studentas naujas)
{
    for (senas.Pradžia(); senas.Yra(); senas.Kitas())
    {
        Mobilus duom = senas.ImtiDuomenis();
        if (duom.tipas == tipas)
            naujas.Papildyti(duom);
    }
}
//-------------------------------------------------------------
```

- Patikrinkite, ar gerai formuojamas naujas sąrašas. Tam pagrindiniame metode `Main()` parašykite sakinius, kuriais būtų atrenkami nurodytą tipą *(Mobilusis telefonas*) turintys objektai. Gautą sąrašą spausdinkite prieš rikiavimą ir po rikiavimo. Tipas įvedamas iš konsolės.

```
//-------------------------------------------------------------
static void Main(string[] args)
{
    Studentas A,        // pirmojo studento duomenys
              B;        // antrojo studento duomenys
    A = SkaitytiAtv(CFd1);
    B = SkaitytiAtv(CFd2);

    if (File.Exists(CFr))
        File.Delete(CFr);

    Spausdinti (CFr, A, A.pv);
    Spausdinti (CFr, B, B.pv);
    using (var failas = new StreamWriter(CFr, true))
    {
         Mobilus max;
         max = A.MaxTrukmė();
         failas.WriteLine("Studentas: {0}, ilgiausiai veikianti baterija  \r\n  modelis: " +
                          "{1}, tipas:  {2}, trukmė: {3}.",
                          A.pv, max.modelis, max.tipas, max.baterija);
         failas.WriteLine();
         max = B.MaxTrukmė();
         failas.WriteLine("Studentas: {0}, ilgiausiai veikianti baterija  \r\n  modelis: " +
                          " {1}, tipas:  {2}, trukmė: {3}.",
                          B.pv, max.modelis, max.tipas, max.baterija);
    }
// --- Nurodyto tipo įrenginių atrinkimas ---
    Studentas Naujas = new Studentas();    // atrinkti duomenys
    Console.WriteLine("Įveskite norimą įrenginio tipą:");
    string tipas = Console.ReadLine();  // Įvedamas norimas įrenginio tipas
    Atrinkti(A, tipas, Naujas);
    Atrinkti(B, tipas, Naujas);
    // --- Suformuoto sąrašo spausdinimas ir rikiavimas ---
    Naujas.Pradžia();
    if (Naujas.Yra() )
    {
        Spausdinti(CFr, Naujas, "Atrinkti nerikiuoti");
        Naujas.Rikiuoti();
        Spausdinti(CFr, Naujas, "Atrinkti surikiuoti");
    }
    else
    {
```

```
        using (var failas = new StreamWriter(CFr, true))
        {
            failas.WriteLine("Naujas sąrašas nesudarytas.");
        }
    }
}
//---------------------------------------------------------------
```

Šeštas žingsnis.

Tekstiniame faile "`Rimas.txt`" yra dar vieno studento turimos mobiliosios elektronikos įrenginių sąrašas. Reikia gautą rezultatų sąrašą papildyti to paties tipo įrenginių sąrašu. Sąrašas yra surikiuotas, todėl naujus duomenis reikia įterpti taip, kad nereikėtų rikiuoti iš naujo.

Duomenų failo `Rimas.txt` **pavyzdys**
`Rimas Kurpius`
`Apple iPod Shuffle;Grotuvas;  400`
`Sony Ericsson K6109;Mobilusis telefonas;  455`
`Tavo;Mobilusis telefonas;  650`
`Motorola Razor V4;Mobilusis telefonas;  490`

- Programą papildykite failo vardo aprašu: `const string CFd3 = "..\\..\\Rimas.txt";`
- Parašykite sąrašą papildančią naujais duomenimis metodą.

```
//---------------------------------------------------------------
// Iš sąrašo senas kopijuoja objektus į sąrašą naujas
// senas   įrenginių sąrašas
// tipas   atrenkamų įrenginių tipas
// naujas  naujo objektų sąrašo adresas
static void Atrinkti_Į_Rikiuotą(Studentas senas, string tipas, Studentas naujas)
{
    for (senas.Pradžia(); senas.Yra(); senas.Kitas())
    {
        Mobilus duom = senas.ImtiDuomenis();
        if (duom.tipas == tipas)
            naujas.Įterpti(duom);
     }
}
//---------------------------------------------------------------
```

Klasę `Studentas` papildykite metodu, kuris randa elementą, už kurio reikia įterpti naują. Šį metodą aprašome `private`, nes `public` neturi būti metodo, kuris per savo vardą ar per parametrus grąžintų kurio tai sąrašo elemento adresą. Tai pažeidžia OP principus!

```
//---------------------------------------------------------------
// Ieškoma naujo elemento įterpimo vieta.
// Vieta objektui duom ieškoma, naudojant rikiavimui sukurtu operatoriumi
// duom – objektas
private Mazgas Vieta(Mobilus duom)
{
    Mazgas  dd = pr;
    while (dd != null && dd.Kitas != null && duom >= dd.Kitas.Duomenys)
        dd = dd.Kitas;
    return dd;
}
//---------------------------------------------------------------
```

- Klasę `Studentas` papildykite nauju įterpimo metodu.

```
//---------------------------------------------------------------
// Suranda vietą, kurioje reikia įterpti naują elementą ir įterpia
// duom – objektas papildymui
public void Įterpti(Mobilus duom)
{
    Mazgas d = new Mazgas();
    d.Duomenys = duom;
    d.Kitas = null;
    if (pr == null) pr = d;  // jei sąrašas tuščias
    else
        if (pr.Duomenys >= duom)
        {  // jeigu elementą reikia sukurti sąrašo pradžioje
```

```
            d.Kitas = pr;
            pr = d;
        }
        else
        {  // jeigu elementą reikia įterpti sąraše
           // randama įterpimo vieta – elementas, už kurio reikia įterpti
            Mazgas dd = Vieta(duom);
            d.Kitas = dd.Kitas; // naujas elementas įterpiamas už surasto elemento
            dd.Kitas = d;
        }
}
//---------------------------------------------------------------
```

- `Main()` metodą papildykite sakiniais:

```
Studentas C;    // trečio studento duomenys
C = SkaitytiAtv(CFd3);
Spausdinti(CFr, C, C.pv);

Atrinkti_Į_Rikiuotą(C, tipas, Naujas);
Naujas.Pradžia();
if (Naujas.Yra())
    Spausdinti(CFr, Naujas, "Rikiuotas po papildymo");
else
    using (var failas = new StreamWriter(CFr, true))
    {
        failas.WriteLine("Naujas sąrašas liko nesudarytas.");
    }
```

- Įvykdykite programą. Pasitikrinkite atsakymus.
- `Main()` metodą papildykite sakiniais, kad išnaikintų sąrašą *Naujas*. Atspausdinkite jį. Po to šiuos sakinius užkomentuokite. Įsitikinkite, ar gautas rezultatas teisingas.
- Dar kartą išspręskite programą, nurodant tipą *Grotuvas*.

Papildant vienkryptį sąrašą naujais elementais nagrinėjamos dvi situacijos:

- naujas elementas jungiamas prie sąrašo pradžios; tuomet sąrašo pradžios rodyklės reikšmė pakinta;
- naujas elementas įterpiamas; tuomet reikia žinoti po kurio elemento reikia įterpti.

Situacija, kai reikia naują elementą jungti sąrašo pabaigoje, sutampa su antrąja: jungti po paskutinio elemento sąraše.
Sąrašo pradžioje padėjus fiktyvų elementą (be duomenų), visos sąrašo papildymo situacijos tampa „po“.

Savarankiško darbo užduotis.

- Sukurkite projekto kopiją. Pertvarkykite programą, kad būtų formuojamas tiesioginis sąrašas.
- Rikiavimui parašykite burbuliuko metodą. Burbuliuko metodas - porinių sukeitimų metodas. Todėl jį galima modifikuoti: atlikti porinius sukeitimus nuo sąrašo pradžios iki galo.
- Pertvarkykite programą, kad fiktyvus elementas sąrašo pradžioje būtų visuomet. Tam reikia modifikuoti visus metodus, kurie sąrašą formuoja ir atlieka veiksmus su duomenimis.
- Pagalvokite, ar įmanoma atsisakyti sąrašo klasės metodo *Papildyti*(). Jeigu taip, kur ir kokius pataisymus reikėtų atlikti. (Realizuoti nebūtina)
- Pagalvokite, kaip reiktų aprašyti sąrašo klasę *Studentai* konteinerine. Kokius pataisymus dar reikėtų atlikti programoje. (Realizuoti nebūtina)

Uždavinio sprendimas, naudojant susieto sąrašo konteinerį *LinkedList<T>*

Sukurkite naują projektą. Pertvarkykite pirmoje dalyje išspręstą uždavinį. Panaudokite susieto sąrašo konteinerį *LinkedList<T>* (*System.Collections.Generic*).

Pirmas žingsnis.

Nusikopijuokite klasę įrenginio duomenims saugoti ir apdoroti. Užrašykite užklotus operatorius, kuriuos vėliau panaudosite metoduose.

```
//-------------------------------------------------------------
// Mobiliojo įrenginio duomenų saugojimo klasė
public class Mobilus
{   // mobiliosios elektronikos įrenginys
    public string modelis { get; set; }  // modelio pavadinimas
    public string tipas  { get; set; }   // įrenginio tipas
    public int baterija  { get; set; }   // baterijos veikimo trukmė

    //  konstruktorius
    public Mobilus(string modelis ="", string tipas = "", int baterija = 0)
    {
        this.modelis = modelis;
        this.tipas = tipas;
        this.baterija = baterija;
    }

    //  objekto naujos reikšmės
    // a – modelio pavadinimas
    // b – įrenginio tipas
    // c – baterijos veikimo trukmė
    void Dėti(string a, string b, int c)
    {
        modelis = a;
        tipas = b;
        baterija = c;
    }

    public override string ToString()
    {
        string eilute;
        eilute = string.Format("|{0, -30}| {1, -20} |   {2, 8:f}   |",
                 modelis, tipas, baterija);
        return eilute;
    }

    public override bool Equals(object objektas)
    {
        Mobilus telef = objektas as Mobilus;
        return telef.tipas == tipas && telef.modelis == modelis && telef.baterija ==
               baterija;
    }

    // Užklotas metodas GetHashCode()
    public override int GetHashCode()
    {
        return base.GetHashCode();
    }

    // Užklotas operatorius >= (dviejų įrenginių palyginimui pagal baterijos veikimo trukmę
    // ir modelio pavadinimą)
    public static bool operator >=(Mobilus pirmas, Mobilus antras)
    {
        int poz = String.Compare(pirmas.modelis, antras.modelis,
                  StringComparison.CurrentCulture);
        return pirmas.baterija > antras.baterija || pirmas.baterija == antras.baterija &&
                poz > 0;
    }
```

```
    // Užklotas operatorius <= (dviejų įrenginių palyginimui pagal baterijos veikimo trukmę
    // ir modelio pavadinimą)
    public static bool operator <=(Mobilus pirmas, Mobilus antras)
    {
        int poz = String.Compare(pirmas.modelis, antras.modelis,
                  StringComparison.CurrentCulture);
        return pirmas.baterija < antras.baterija || pirmas.baterija == antras.baterija &&
               poz < 0;
    }

    // Užklotas operatorius == (įrenginių tipui palyginti)
    public static bool operator ==(Mobilus pirmas, Mobilus antras)
    {
        return pirmas.tipas == antras.tipas;
    }

    // Užklotas operatorius != (įrenginių tipui palyginti)
    public static bool operator !=(Mobilus pirmas, Mobilus antras)
    {
        return pirmas.tipas != antras.tipas;
    }

    // Užklotas operatorius > (dviejų įrenginių palyginimui pagal baterijos veikimo trukmę)
    public static bool operator >(Mobilus pirmas, Mobilus antras)
    {
        return pirmas.baterija > antras.baterija;
    }

    // Užklotas operatorius < (dviejų įrenginių palyginimui pagal baterijos veikimo trukmę)
    public static bool operator <(Mobilus pirmas, Mobilus antras)
    {
        return pirmas.baterija < antras.baterija;
    }
}
//-------------------------------------------------------------
```

Antras žingsnis.

Nusikopijuokite pagrindinę klasę *Program*. Nukopijuokite reikiamus jūsų sukurtos klasės *Studentai* metodus, kurie bus reikalingi spendžiant uždavinį. Rikiavimo metodo nekopijuokite. Sąrašo rikiavimui panaudosime *LinkedList<T>* klasės užklotą metodą *OrderBy*(). Pertvarkykite metodus darbui su susieto sąrašo konteineriu *LinkedList<T>*.

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Pratybos_4_1  // Pakeiskite paryškintą vardą!
{
    class Program
    {
        const string CFd1 = "..\\..\\Mikas.txt";
        const string CFd2 = "..\\..\\Darius.txt";
        const string CFd3 = "..\\..\\Rimas.txt";
        const string CFr = "..\\..\\Rezultatai.txt";

        static void Main(string[] args)
        {
            string[] Vardai = new string[3];
            LinkedList<Mobilus> A = new LinkedList<Mobilus>();
            LinkedList<Mobilus> B = new LinkedList<Mobilus>();
            SkaitytiAtv(CFd1, 0, Vardai, A);
            SkaitytiAtv(CFd2, 1, Vardai, B);
```

```
        if (File.Exists(CFr))
            File.Delete(CFr);

        Spausdinti(CFr, A, Vardai[0]);
        Spausdinti(CFr, B, Vardai[1]);

        using (var failas = new StreamWriter(CFr, true))
        {
            Mobilus max;
            max = MaxTrukmė(A);
            failas.WriteLine("Studentas: {0}, ilgiausiai veikianti baterija  \r\n" +
                             "  modelis: {1}, tipas:  {2}, trukmė: {3}.",
                             Vardai[0], max.modelis, max.tipas, max.baterija);
            failas.WriteLine();
            max = MaxTrukmė(B);
            failas.WriteLine("Studentas: {0}, ilgiausiai veikianti baterija  \r\n" +
                             "  modelis: {1}, tipas:  {2}, trukmė: {3}.",
                             Vardai[1], max.modelis, max.tipas, max.baterija);
        }

        // --- Nurodyto tipo įrenginių atrinkimas ---
        LinkedList<Mobilus> Naujas = new LinkedList<Mobilus>();   // atrinkti duomenys
        Console.WriteLine("Įveskite norimą įrenginio tipą:");
        string tipas = Console.ReadLine();  // Įvedamas norimas įrenginio tipas
        Atrinkti(A, tipas, Naujas);
        Atrinkti(B, tipas, Naujas);
        // --- Suformuoto sąrašo spausdinimas ir rikiavimas ---
        if (Naujas.Count > 0)
        {
            Spausdinti(CFr, Naujas, "Atrinkti nerikiuoti");
            // -Suformuoto sąrašo rikiavimas
            Naujas = new LinkedList<Mobilus>
                         (Naujas.OrderBy(p => p.baterija).ThenBy(p => p.modelis));
            Spausdinti(CFr, Naujas, "Atrinkti surikiuoti");
        }
        else
        {
            using (var failas = new StreamWriter(CFr, true))
            {
                failas.WriteLine("Naujas sąrašas nesudarytas.");
            }
        }

        LinkedList<Mobilus> C = new LinkedList<Mobilus>();  // trečio studento duomenys
        SkaitytiAtv(CFd3, 2, Vardai, C);
        Spausdinti(CFr, C, Vardai[2]);
        Atrinkti_Į_Rikiuotą(C, tipas, Naujas);
        if (Naujas.Count() > 0)
            Spausdinti(CFr, Naujas, "Rikiuotas po papildymo");
        else
            using (var failas = new StreamWriter(CFr, true))
            {
                failas.WriteLine("Naujas sąrašas liko nesudarytas.");
            }
    }

    // Skaitomi duomenys iš failo ir sudedami į sąrašą ATVIRKŠČIA tvarka
    // fv – duomenų failo vardas
    // vardo numeris Vardai masyve
    static void SkaitytiAtv(string fv, int indeksas, string[] Vardai,
                            LinkedList<Mobilus> A)
    {

        using (var failas = new StreamReader(fv))
        {
            string eilute;
            Vardai[indeksas] = eilute = failas.ReadLine();
            while ((eilute = failas.ReadLine()) != null)
```

```
            {
                string[] eilDalis = eilute.Split(';');
                string modelis = eilDalis[0];
                string tipas = eilDalis[1];
                int baterija = int.Parse(eilDalis[2]);
                Mobilus elem = new Mobilus(modelis, tipas, baterija);
                A.AddFirst(elem);
            }
        }
    }

    // Sąrašo duomenys spausdinami faile
    // fv – duomenų failo vardas
    // A  - sąrašo objekto nuoroda
    // koment  - komentaras
    static void Spausdinti(string fv, LinkedList<Mobilus> A, string koment)
    {
        using (var failas = new StreamWriter(fv, true))
        {
            failas.WriteLine(koment);
            failas.WriteLine("+-----------------------------+--------------" +
                             "-------+--------------+");
            failas.WriteLine("|             Modelis          |        Tipas  " +
                             "       | Veik. trukmė |");
            failas.WriteLine("+-----------------------------+--------------" +
                             "-------+--------------+");

            // Sąrašo peržiūra, panaudojant sąsajos metodus
            foreach (Mobilus elem in A)
            {
                failas.WriteLine("{0}", elem.ToString());
            }
            failas.WriteLine("+-----------------------------+--------------" +
                             "-------+--------------+");
            failas.WriteLine();
        }
    }

    // Suranda ir grąžina ilgiausiai veikiančio įrenginio duomenis
    static Mobilus MaxTrukmė(LinkedList<Mobilus> A)
    {
        Mobilus max;
        max = A.First();
        foreach (Mobilus elem in A)
            if (elem > max)
                max = elem;
        return max;
    }


    // Iš sąrašo senas kopijuoja objektus į sąrašą naujas
    // senas   įrenginių sąrašas
    // tipas   atrenkamų įrenginių tipas
    // naujas  naujo objektų sąrašo adresas
    static void Atrinkti(LinkedList<Mobilus> senas, string tipas,
                         LinkedList<Mobilus> naujas)
    {
        foreach (Mobilus elem in senas)
        {
            if (elem.tipas == tipas)
                naujas.AddLast(elem);
        }
    }
```

```
        // Iš sąrašo senas kopijuoja objektus į sąrašą naujas
        // senas   įrenginių sąrašas
        // tipas   atrenkamų įrenginių tipas
        // naujas  naujo objektų sąrašo adresas
        static void Atrinkti_Į_Rikiuotą(LinkedList<Mobilus> senas, string tipas,
                                        LinkedList<Mobilus> naujas)
        {
            foreach (Mobilus elem in senas)
            {
                if (elem.tipas == tipas)
                {
                    Mobilus pagalb = Vieta(naujas, elem);
                    if (pagalb.baterija == -1) naujas.AddFirst(elem);
                    else
                    {
                        LinkedListNode<Mobilus> mazgas = naujas.Find(pagalb);
                        naujas.AddAfter(mazgas, elem);
                    }
                }
            }
        }

        // Ieškoma naujo elemento įterpimo vieta.
        // Vieta objektui elementas ieškoma, naudojantis sukurtu operatoriumi
        // sar – susietas sąrašas
        // elementas – objektas
        static Mobilus Vieta(LinkedList<Mobilus> sar, Mobilus elementas)
        {
            Mobilus rastasElem = new Mobilus();
            rastasElem.baterija = -1;
            foreach (Mobilus elem in sar)
            {
                if (elem <= elementas)
                    rastasElem = elem;
            }
            return rastasElem;
        }
    }
}
```

Trečias žingsnis.

Nusikopijuokite duomenų failus. Išspręskite uždavinį. Pasitikrinkite atsakymus.