

Operatyvioji atmintis

```
1 using System;
2 using System.Collections.Generic;
3 using System.Diagnostics;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace Alg1
9 {
10     0 references
11     class BucketSort
12     {
13         0 references
14         private static void Main(string[] args)
15         {
16             int seed = (int)DateTime.Now.Ticks & 0x0000FFFF;
17             int[] numOfData = { 1000, 2000, 3000, 20000, 60000 };
18
19             //Test_Array_List(seed); //Rikiavimas
20             Benchmark(seed, numOfData);
21         }
22         #region Array
23         2 references
24         private static void BucketSortArray(DataArray x)
25         {
26             int numOfBuckets = 10;
27
28             List<Objektas>[] buckets = new List<Objektas>[numOfBuckets];
29             for (int i = 0; i < numOfBuckets; i++)
30             {
31                 buckets[i] = new List<Objektas>();
32             }
33
34             for (int i = 0; i < x.Length; i++)
35             {
36                 int bucket = (int)(x[i].flo * numOfBuckets);
37                 buckets[bucket].Add(x[i]);
38             }
39
40             int a = 0;
41             for (int i = 0; i < numOfBuckets; i++)
42             {
43                 BubbleSort(buckets[i]);
44                 for (int j = 0; j < buckets[i].Count; j++)
45                 {
46                     x.Change(a, buckets[i][j]);
47                     a++;
48                 }
49             }
50         }
51     }
52 }
53
54 #endregion
55 2 references
56 private static void BucketSortList(DataList x)
57 {
58     int numOfBuckets = 10;
59
60     List<Objektas>[] buckets = new List<Objektas>[numOfBuckets];
61     for (int i = 0; i < numOfBuckets; i++)
62     {
63         buckets[i] = new List<Objektas>();
64     }
65
66     Objektas temp = x.Head();
67     buckets[(int)(temp.flo * numOfBuckets)].Add(temp);
68     for (int i = 1; i < x.Length; i++)
69     {
70         temp = x.Next();
71         int bucket = (int)(temp.flo * 10);
72         buckets[bucket].Add(temp);
73     }
74
75     int a = 0;
76     x.clear();
77     for (int i = 0; i < numOfBuckets; i++)
78     {
79         BubbleSort(buckets[i]);
80         x.addAll(buckets[i]);
81     }
82 }
83
84 2 references
85 public static void BubbleSort(List<Objektas> items)
86 {
87     Objektas prevdata, currentdata;
88     for (int i = items.Count - 1; i >= 0; i--)
89     {
90         currentdata = items[0];
91         for (int j = 1; j <= i; j++)
92         {
93             prevdata = currentdata;
94             currentdata = items[j];
95             if (prevdata > currentdata)
96             {
97                 items[j - 1] = currentdata;
98                 items[j] = prevdata;
99             }
100         }
101     }
102 }
```

```
45     }
46 }
47
48 #endregion
49 2 references
50 private static void BucketSortList(DataList x)
51 {
52     int numOfBuckets = 10;
53
54     List<Objektas>[] buckets = new List<Objektas>[numOfBuckets];
55     for (int i = 0; i < numOfBuckets; i++)
56     {
57         buckets[i] = new List<Objektas>();
58     }
59
60     Objektas temp = x.Head();
61     buckets[(int)(temp.flo * numOfBuckets)].Add(temp);
62     for (int i = 1; i < x.Length; i++)
63     {
64         temp = x.Next();
65         int bucket = (int)(temp.flo * 10);
66         buckets[bucket].Add(temp);
67     }
68
69     int a = 0;
70     x.clear();
71     for (int i = 0; i < numOfBuckets; i++)
72     {
73         BubbleSort(buckets[i]);
74         x.addAll(buckets[i]);
75     }
76 }
77
78 2 references
79 public static void BubbleSort(List<Objektas> items)
80 {
81     Objektas prevdata, currentdata;
82     for (int i = items.Count - 1; i >= 0; i--)
83     {
84         currentdata = items[0];
85         for (int j = 1; j <= i; j++)
86         {
87             prevdata = currentdata;
88             currentdata = items[j];
89             if (prevdata > currentdata)
90             {
91                 items[j - 1] = currentdata;
92                 items[j] = prevdata;
93             }
94         }
95     }
96 }
```

```

89         currentdata = prevdata;
90     }
91 }
92 }
93 }
94 0 references
95 public static void Test_Array_List(int seed)
96 {
97     int n = 12;
98     MyDataArray myarray = new MyDataArray(n, seed);
99     Console.WriteLine("\n ARRAY \n");
100    myarray.Print(n);
101    BucketSortArray(myarray);
102    Console.WriteLine("\n SORTED \n");
103    myarray.Print(n);
104
105    MyDataList mylist = new MyDataList(n, seed);
106    Console.WriteLine("\n LIST \n");
107    mylist.Print(n);
108    Console.WriteLine("\n SORTED \n");
109    BucketSortList(mylist);
110    mylist.Print(n);
111 }
112 1 reference
113 public static void Benchmark(int seed, int[] dataCount)
114 {
115     Console.WriteLine("Array");
116     for (int i = 0; i < dataCount.Length; i++)
117     {
118         int n = dataCount[i];
119         MyDataArray myarray = new MyDataArray(n, seed);
120         var benchmark = Stopwatch.StartNew();
121         BucketSortArray(myarray);
122         benchmark.Stop();
123         Console.WriteLine("{0} - {1}", dataCount[i], benchmark.Elapsed);
124     }
125     Console.WriteLine("LinkedList");
126     for (int i = 0; i < dataCount.Length; i++)
127     {
128         int n = dataCount[i];
129         MyDataList mylist = new MyDataList(n, seed);
130         var benchmark = Stopwatch.StartNew();
131         BucketSortList(mylist);
132         benchmark.Stop();
133         Console.WriteLine("{0} - {1}", dataCount[i], benchmark.Elapsed);
134     }
135 }
136 }

```

```

133 }
134 }
135 }
136 }

```

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Alg1
8 {
9     abstract class DataArray
10     {
11         protected int length;
12
13         public int Length { get { return length; } }
14
15         public abstract Objktas this[int index] { get; }
16
17         public abstract void Change(int index, Objktas naujas);
18
19         public abstract void Swap(int j, Objktas a, Objktas b);
20
21         public void Print(int n)
22         {
23             for (int i = 0; i < n; i++)
24             {
25                 Console.WriteLine(" {0:F5} {1}", this[i].flo, this[i].str);
26                 Console.WriteLine();
27             }
28         }
29     }
30 }

```

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Alg1
8  {
9      2 references
10     abstract class Datalist
11     {
12         protected int length;
13         1 reference
14         public int Length { get { return length; } }
15         4 references
16         public abstract Objektas Head();
17         4 references
18         public abstract Objektas Next();
19         1 reference
20         public abstract void Swap(Objektas a, Objektas b);
21
22         public abstract void addAll(List<Objektas> items);
23
24         public abstract void clear();
25
26         public void Print(int n)
27         {
28             Console.WriteLine(" {0:F5} {1}", Head().flo, Head().str);
29             Objektas tee = Next();
30             while (tee != null)
31             {
32                 Console.WriteLine(" {0:F5} {1}", tee.flo, tee.str);
33                 tee = Next();
34             }
35             Console.WriteLine();
36         }
37     }
38 }

```

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Alg1
8  {
9      class Objektas
10     {
11         public string str { get; set; }
12         public float flo { get; set; }
13
14
15         public Objektas(float newF, string newS)
16         {
17             str = newS;
18             flo = newF;
19         }
20
21         public static bool operator <(Objektas lhs, Objektas rhs)
22         {
23             if (lhs.flo == rhs.flo)
24                 return lhs.str.CompareTo(rhs.str) == 1;
25
26             return lhs.flo < rhs.flo;
27         }
28
29         public static bool operator >(Objektas lhs, Objektas rhs)
30         {
31             if (lhs.flo == rhs.flo)
32                 return lhs.str.CompareTo(rhs.str) == -1;
33
34             return lhs.flo > rhs.flo;
35         }
36     }
37 }

```

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Alg1
8  {
9      5 references
10     class MyDataArray : DataArray
11     {
12         2 references
13         Objektas[] data;
14         public MyDataArray(int n, int seed)
15         {
16             data = new Objektas[n];
17             length = n;
18             Random rand = new Random(seed);
19             for(int i = 0; i < length; i++)
20             {
21                 Objektas temp = new Objektas((float)rand.NextDouble(), CreateString(4, rand));
22                 data[i] = temp;
23             }
24         }
25         1 reference
26         internal static string CreateString(int stringLength, Random rd)
27         {
28             const string allowedChars = "ABCDEFGHJKLMNOPQRSTUVWXYZ";
29             char[] chars = new char[stringLength];
30
31             for (int i = 0; i < stringLength; i++)
32             {
33                 chars[i] = allowedChars[rd.Next(0, allowedChars.Length)];
34             }
35             return new string(chars);
36         }
37         5 references
38         public override Objektas this[int index]
39         {
40             get { return data[index]; }
41         }
42         1 reference
43         public override void Swap(int j, Objektas a, Objektas b)
44         {
45             data[j - 1] = a;
46             data[j] = b;
47         }
48     }
49
50     public override void Change(int index, Objektas naujas)
51     {
52         data[index] = naujas;
53     }
54 }

```

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Alg1
8  {
9      class MyDataList : DataList
10     {
11         class MyLinkedListNode
12         {
13             public MyLinkedListNode nextNode { get; set; }
14             public Objektas data { get; set; }
15             public MyLinkedListNode(Objektas data)
16             {
17                 this.data = data;
18             }
19         }
20         MyLinkedListNode headNode;
21         MyLinkedListNode prevNode;
22         MyLinkedListNode currentNode;
23
24         public MyDataList(int n, int seed)
25         {
26             length = n;
27             Random rand = new Random(seed);
28             headNode = new MyLinkedListNode(new Objektas((float)rand.NextDouble(), CreateString(4, rand)));
29             currentNode = headNode;
30             for (int i = 1; i < length; i++)
31             {
32                 prevNode = currentNode;
33                 currentNode.nextNode = new MyLinkedListNode(new Objektas((float)rand.NextDouble(), CreateString(4, rand)));
34                 currentNode = currentNode.nextNode;
35             }
36             currentNode.nextNode = null;
37         }
38     }
39 }

```

```

37  public override void clear()
38  {
39      headNode = null;
40      prevNode = null;
41      currentNode = null;
42  }
43
44  2 references
45  public override void addAll(List<Objektas> items)
46  {
47      foreach (Objektas item in items)
48      {
49          if (headNode == null)
50          {
51              headNode = new MyLinkedListNode(item);
52              currentNode = headNode;
53              continue;
54          }
55          prevNode = currentNode;
56          currentNode.nextNode = new MyLinkedListNode(item);
57          currentNode = currentNode.nextNode;
58      }
59      currentNode.nextNode = null;
60  }
61
62  4 references
63  public override Objektas Head()
64  {
65      currentNode = headNode;
66      prevNode = null;
67      return currentNode.data;
68  }
69
70  4 references
71  public override Objektas Next()
72  {
73      prevNode = currentNode;
74      currentNode = currentNode.nextNode;
75      if (currentNode == null) return null;
76      return currentNode.data;
77  }
78
79  1 reference
80  public override void Swap(Objektas a, Objektas b)
81  {
82      prevNode.data = a;
83      currentNode.data = b;
84  }
85  }
86  }

```

```
78     internal static string CreateString(int stringLength, Random rd)
79     {
80         const string allowedChars = "ABCDEFGHJKLMNOPQRSTUVWXYZ";
81         char[] chars = new char[stringLength];
82         for (int i = 0; i < stringLength; i++)
83             chars[i] = allowedChars[rd.Next(0, allowedChars.Length)];
84         return new string(chars);
85     }
86 }
87 }
88 }
```

Išorinė atmintis

```

1  using System;
2  using System.Collections.Generic;
3  using System.Diagnostics;
4  using System.IO;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace Alg1_2
10 {
11     0 references
12     class BucketSort
13     {
14         0 references
15         class Bubble_Sort
16         {
17             0 references
18             private static void Main(string[] args)
19             {
20                 int[] numOfData = { 100, 200, 300, 2000, 6000 };
21                 int seed = (int)DateTime.Now.Ticks & 0x0000FFFF;
22                 Benchmark(seed, numOfData);
23                 //Test_File_Array_List(seed);
24             }
25             1 reference
26             private static void Benchmark(int seed, int[] dataCount)
27             {
28                 Console.WriteLine("Array");
29                 for (int i = 0; i < dataCount.Length; i++)
30                 {
31                     int n = dataCount[i];
32                     string filename = @"mydataarray.dat";
33                     MyFileArray myfilearray = new MyFileArray(filename, n, seed);
34                     var benchmark = Stopwatch.StartNew();
35                     using (myfilearray.fs = new FileStream(filename, FileMode.Open, FileAccess.ReadWrite))
36                         BucketSortArray(myfilearray);
37                     benchmark.Stop();
38                     Console.WriteLine("{0} - {1}", dataCount[i], benchmark.Elapsed);
39                 }
40                 Console.WriteLine("LinkedList");
41                 for (int i = 0; i < dataCount.Length; i++)
42                 {
43                     int n = dataCount[i];
44                     string filename = @"mydatalist.dat";
45                     MyFileList myfilelist = new MyFileList(filename, n, seed);
46
47                     var benchmark = Stopwatch.StartNew();
48                     using (myfilelist.fs = new FileStream(filename, FileMode.Open, FileAccess.ReadWrite))
49                         BucketSortList(myfilelist);
50                     benchmark.Stop();
51                     Console.WriteLine("{0} - {1}", dataCount[i], benchmark.Elapsed);
52                 }
53             }
54             0 references
55             public static void Test_File_Array_List(int seed)
56             {
57                 int n = 12;
58                 string filename = @"mydataarray.dat";
59                 MyFileArray myfilearray = new MyFileArray(filename, n, seed);
60                 using (myfilearray.fs = new FileStream(filename, FileMode.Open, FileAccess.ReadWrite))
61                 {
62                     Console.WriteLine("\n FILE ARRAY \n");
63                     myfilearray.Print(n);
64
65                     Console.WriteLine("\n SORTED FILE ARRAY \n");
66                     MyFileArray ats = BucketSortArray(myfilearray);
67                     using (ats.fs = new FileStream(@"ats.dat", FileMode.Open, FileAccess.ReadWrite))
68                         ats.Print(n);
69                 }
70                 filename = @"mydatalist.dat";
71                 MyFileList myfilelist = new MyFileList(filename, n, seed);
72                 using (myfilelist.fs = new FileStream(filename, FileMode.Open, FileAccess.ReadWrite))
73                 {
74                     Console.WriteLine("\n FILE LIST \n");
75                     myfilelist.Print(n);
76                     BucketSortList(myfilelist);
77                     Console.WriteLine("\n SORTED FILE LIST \n");
78                     MyFileList atss = BucketSortList(myfilelist);
79                     using (atss.fs = new FileStream(@"Lats.dat", FileMode.Open, FileAccess.ReadWrite))
80                         atss.Print(n);
81                 }
82             }
83         }
84     }
85 }

```

```

79 private static MyFileList BucketSortList(DataList x)
80 {
81     DirectoryInfo di = new DirectoryInfo(@"..\..\data2\");
82     foreach (FileInfo file in di.GetFiles())
83         if (file.Name.Contains("LBucket"))
84             file.Delete();
85
86     int[] lengths = new int[10];
87     for (int i = 0; i < 10; i++)
88         lengths[i] = 0;
89
90     for (int i = 0; i < x.Length; i++)
91     {
92         Objektas temp;
93         int bucket;
94         if (i == 0)
95         {
96             temp = x.Head();
97             bucket = (int)(temp.flo * 10);
98         }
99         else
100         {
101             temp = x.Next();
102             bucket = (int)(temp.flo * 10);
103         }
104
105         string fileName = "LBucket" + bucket + ".dat";
106         string path = @"..\..\data2\" + fileName;
107
108         if (!File.Exists(path))
109             using (BinaryWriter writer = new BinaryWriter(File.Open(path, FileMode.Create)))
110             {
111                 writer.Write(4);
112                 Byte[] str = Encoding.ASCII.GetBytes(temp.str);
113                 writer.Write(str);
114                 writer.Write(temp.flo);
115                 writer.Write((lengths[bucket] + 1) * 12 + 4);
116                 lengths[bucket] += 1;
117             }
118         else
119             using (BinaryWriter writer = new BinaryWriter(File.Open(path, FileMode.Append)))
120             {
121                 Byte[] str = Encoding.ASCII.GetBytes(temp.str);
122                 writer.Write(str);
123                 writer.Write(temp.flo);
124                 writer.Write((lengths[bucket] + 1) * 12 + 4);
125                 lengths[bucket] += 1;
126             }
127
128     MyFileList ats = new MyFileList(@"Lats.dat", x.Length);
129     using (BinaryWriter writer = new BinaryWriter(File.Open(@"Lats.dat", FileMode.Create)))
130     {
131         writer.Write(4);
132         int ind = 0;
133         foreach (FileInfo file in di.GetFiles())
134         {
135             int length = lengths[int.Parse(file.Name.Substring(7, 1))];
136             MyFileList buck = new MyFileList(@"..\..\data2\" + file.Name, length);
137
138             using (buck.fs = new FileStream(@"..\..\data2\" + file.Name, FileMode.Open, FileAccess.ReadWrite))
139             {
140                 //buck.Print(buck.Length);
141                 InsertionSort(buck);
142
143                 for (int j = 0; j < length; j++)
144                 {
145                     Objektas t;
146                     if (j == 0)
147                         t = buck.Head();
148                     else
149                         t = buck.Next();
150                     Byte[] str = Encoding.ASCII.GetBytes(t.str);
151                     writer.Write(str);
152                     writer.Write(t.flo);
153                     writer.Write((ind + 1) * 12 + 4);
154                     ind++;
155                 }
156             }
157         }
158     }
159     return ats;
160 }
161

```



```

162 private static MyFileArray BucketSortArray(DataArray x)
163 {
164     DirectoryInfo di = new DirectoryInfo(@"..\..\data\");
165     foreach (FileInfo file in di.GetFiles())
166         if (file.Name.Contains("ABucket"))
167             file.Delete();
168
169     int[] lengths = new int[10];
170
171     for (int i = 0; i < x.Length; i++)
172     {
173         Objektas key = x[i];
174         int bucket = (int)(key.flo * 10);
175         string fileName = "ABucket" + bucket + ".dat";
176         string path = @"..\..\data\" + fileName;
177
178         if (!File.Exists(@"..\..\data\" + fileName))
179             using (BinaryWriter writer = new BinaryWriter(File.Open(path, FileMode.Create)))
180             {
181                 Byte[] str = Encoding.ASCII.GetBytes(key.str);
182                 writer.Write(str);
183                 writer.Write(key.flo);
184                 lengths[bucket] = 1;
185             }
186         else
187             using (BinaryWriter writer = new BinaryWriter(File.Open(path, FileMode.Append)))
188             {
189                 Byte[] str = Encoding.ASCII.GetBytes(key.str);
190                 writer.Write(str);
191                 writer.Write(key.flo);
192                 lengths[bucket] += 1;
193             }
194     }
195     MyFileArray ats = new MyFileArray(@"ats.dat", x.Length);
196     using (BinaryWriter writer = new BinaryWriter(File.Open(@"ats.dat", FileMode.Create)))
197     {
198         foreach (FileInfo file in di.GetFiles())
199         {
200             int length = lengths[int.Parse(file.Name.Substring(7, 1))];
201             MyFileArray buck = new MyFileArray(@"..\..\data\" + file.Name, length);
202
203             using (buck.fs = new FileStream(@"..\..\data\" + file.Name, FileMode.Open, FileAccess.ReadWrite))
204             {
205                 InsertionSort(buck);
206
207                 for (int j = 0; j < length; j++)
208                 {
209                     Byte[] str = Encoding.ASCII.GetBytes(buck[j].str);
210                     writer.Write(str);
211                     writer.Write(buck[j].flo);
212                 }
213             }
214         }
215     }
216     return ats;
217 }
218
219 1 reference
220 public static void InsertionSort(DataArray myArray)
221 {
222     int n = myArray.Length;
223     for (int i = 1; i < n; ++i)
224     {
225         Objektas key = myArray[i];
226         int j = i - 1;
227
228         while (j >= 0 && CompareV1(myArray[j], key) > 0)
229         {
230             myArray.SetValue(j + 1, myArray[j]);
231             j = j - 1;
232         }
233         myArray.SetValue(j + 1, key);
234     }
235 }
236
237 1 reference
238 public static void InsertionSort(DataList myList)
239 {
240     int n = myList.Length;
241     for (int i = 1; i < n; ++i)
242     {
243         Objektas key = myList.ElementAt(i);
244         int j = i - 1;
245
246         while (j >= 0 && CompareV1(myList.ElementAt(j), key) > 0)
247         {
248             myList.SetValue(j + 1, myList.ElementAt(j));
249             j = j - 1;
250         }
251         myList.SetValue(j + 1, key);
252     }
253 }

```

```

251     public static int CompareV1(Objektas a, Objektas b)
252     {
253         if (a.flo == b.flo)
254             return a.str.CompareTo(b.str);
255         else
256             return a.flo.CompareTo(b.flo);
257     }
258 }
259 }
260 }

```

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Alg1_2
8  {
9      abstract class DataArray
10     {
11         protected int length;
12
13         public int Length { get { return length; } }
14
15         public abstract Objektas this[int index] { get; }
16
17         public abstract void Swap(int j, Objektas a, Objektas b);
18
19         public abstract void SetValue(int i, Objektas v);
20
21         public abstract void Print(int n);
22     }
23 }

```

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Alg1_2
8  {
9      abstract class DataList
10     {
11         protected int length;
12         public int Length { get { return length; } }
13         public abstract Objektas Head();
14         public abstract Objektas Next();
15         public abstract void Swap(Objektas a, Objektas b);
16         public abstract void SetValue(int i, Objektas v);
17         public abstract Objektas ElementAt(int n);
18         public void Print(int n)
19         {
20             Console.Write("{0:F5} {1}", Head().flo, Head().str);
21             for (int i = 1; i < n; i++)
22             {
23                 Objektas temp = Next();
24                 Console.Write("{0:F5} {1}", temp.flo, temp.str);
25             }
26             Console.WriteLine();
27         }
28     }
29 }

```

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Alg1_2
8  {
9      37 references
10     class Objektas
11     {
12         25 references
13         public float flo { get; set; }
14         16 references
15         public string str { get; set; }
16
17         4 references
18         public Objektas(string nstring, float nfloat)
19         {
20             flo = nfloat;
21             str = nstring;
22         }
23
24         0 references
25         public override string ToString()
26         {
27             return String.Format("{0}, {1:F5}\n", str, flo);
28         }
29
30         0 references
31         public string ToFileString()
32         {
33             return String.Format("{0}{1:F5}\n", str, flo);
34         }
35     }
36 }

```

```

1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Alg1_2
9  {
10     12 references
11     class MyFileList : DataList
12     {
13         int prevNode;
14         int currentNode;
15         int nextNode;
16
17         2 references
18         public MyFileList(string filename, int n, int seed)
19         {
20             length = n;
21             Random rand = new Random(seed);
22             if (File.Exists(filename)) File.Delete(filename);
23             try
24             {
25                 using (BinaryWriter writer = new BinaryWriter(File.Open(filename, FileMode.Create)))
26                 {
27                     writer.Write(4);
28                     for (int j = 0; j < length; j++)
29                     {
30                         Byte[] str = Encoding.ASCII.GetBytes(CreateString(4, rand));
31                         writer.Write(str);
32                         writer.Write((float)rand.NextDouble());
33                         writer.Write((j + 1) * 12 + 4);
34                     }
35                 }
36             }
37             catch (IOException ex)
38             {
39                 Console.WriteLine(ex.ToString());
40             }
41         }
42
43         2 references
44         public MyFileList(string filename, int n)
45         {
46             length = n;
47         }
48     }
49 }

```

```

43 public FileStream fs { get; set; }
44 7 references
45 public override Objektas Head()
46 {
47     Byte[] data = new Byte[12];
48     fs.Seek(0, SeekOrigin.Begin);
49     fs.Read(data, 0, 4);
50     currentNode = BitConverter.ToInt32(data, 0);
51     prevNode = -1;
52     fs.Seek(currentNode, SeekOrigin.Begin);
53     fs.Read(data, 0, 12);
54     string str = Encoding.ASCII.GetString(data.Take(4).ToArray());
55     float flo = BitConverter.ToSingle(data, 4);
56     nextNode = BitConverter.ToInt32(data, 8);
57     return new Objektas(str, flo);
58 }
59 6 references
60 public override Objektas Next()
61 {
62     Byte[] data = new Byte[12];
63     fs.Seek(nextNode, SeekOrigin.Begin); fs.Read(data, 0, 12);
64     prevNode = currentNode;
65     currentNode = nextNode;
66     string str = Encoding.ASCII.GetString(data.Take(4).ToArray());
67     float flo = BitConverter.ToSingle(data, 4);
68     nextNode = BitConverter.ToInt32(data, 8);
69     return new Objektas(str, flo);
70 }
71 1 reference
72 public override void Swap(Objektas a, Objektas b)
73 {
74     Byte[] data;
75     fs.Seek(prevNode, SeekOrigin.Begin);
76     data = BitConverter.GetBytes(a.flo);
77     fs.Write(data, 0, 8);
78     fs.Seek(currentNode, SeekOrigin.Begin);
79     data = BitConverter.GetBytes(b.flo);
80     fs.Write(data, 0, 8);
81 }
82 6 references
83 public override void SetValue(int i, Objektas v)
84 {
85     Objektas temp = Head();
86     for (int x = 0; x < Length; x++)
87     {
88         if (x == i)
89         {
90             Byte[] dataStr = Encoding.ASCII.GetBytes(v.str);
91             Byte[] dataFloat = new Byte[8];
92             BitConverter.GetBytes(v.flo).CopyTo(dataFloat, 4);
93
94             fs.Seek(currentNode, SeekOrigin.Begin);
95
96             fs.Write(dataStr, 0, 4);
97             fs.Write(dataFloat, 4, 4);
98
99             break;
100         }
101         temp = Next();
102     }
103 }
104 4 references
105 public override Objektas ElementAt(int n)
106 {
107     Objektas temp = Head();
108     for (int i = 0; i < Length; i++)
109     {
110         if (i == n)
111             return temp;
112         temp = Next();
113     }
114     return temp;
115 }
116 1 reference
117 internal static string CreateString(int stringLength, Random rd)
118 {
119     const string allowedChars = "ABCDEFGHJKLMNOPQRSTUVWXYZ";
120     //const string allowedChars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
121     //const string allowedChars = "A";
122     char[] chars = new char[stringLength];
123
124     for (int i = 0; i < stringLength; i++)
125     {
126         chars[i] = allowedChars[rd.Next(0, allowedChars.Length)];
127     }
128 }

```

```
123  
124     return new string(chars);  
125 }  
126  
127  
128 }  
129
```

```

45 internal static string CreateString(int stringLength, Random rd)
46 {
47     const string allowedChars = "ABCDEFGHJKLMNOPQRSTUVWXYZ";
48     //const string allowedChars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
49     //const string allowedChars = "A";
50     char[] chars = new char[stringLength];
51
52     for (int i = 0; i < stringLength; i++)
53     {
54         chars[i] = allowedChars[rd.Next(0, allowedChars.Length)];
55     }
56
57     return new string(chars);
58 }
59
60 public FileStream fs { get; set; }
61
62 public override Objektas this[int index]
63 {
64     get
65     {
66         Byte[] data = new Byte[8];
67         fs.Seek(8 * index, SeekOrigin.Begin);
68         fs.Read(data, 0, 8);
69         string s = Encoding.ASCII.GetString(data.Take(4).ToArray());
70         float dataFloat = BitConverter.ToSingle(data, 4);
71         return new Objektas(s, dataFloat);
72     }
73 }
74
75 public override void Swap(int j, Objektas a, Objektas b)
76 {
77     Byte[] data = new Byte[16];
78     BitConverter.GetBytes(a.flo).CopyTo(data, 0);
79     BitConverter.GetBytes(b.flo).CopyTo(data, 8);
80     fs.Seek(8 * (j - 1), SeekOrigin.Begin);
81     fs.Write(data, 0, 16);
82 }

```

```

1 using System;
2 using System.Collections.Generic;
3 using System.IO;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace Algl1_2
9 {
10     class MyFileArray : DataArray
11     {
12         public MyFileArray(string filename, int n, int seed)
13         {
14             Objektas[] data = new Objektas[n];
15             length = n;
16             Random rand = new Random(seed);
17             for (int i = 0; i < length; i++)
18             {
19                 data[i] = new Objektas(CreateString(4, rand), (float)rand.NextDouble());
20             }
21             if (File.Exists(filename)) File.Delete(filename);
22             try
23             {
24                 using (BinaryWriter writer = new BinaryWriter(File.Open(filename, FileMode.Create)))
25                 {
26                     for (int j = 0; j < length; j++)
27                     {
28                         Byte[] str = Encoding.ASCII.GetBytes(data[j].str);
29                         writer.Write(str);
30                         writer.Write(data[j].flo);
31                     }
32                     //writer.Write(data[j]);
33                 }
34             }
35             catch (IOException ex)
36             {
37                 Console.WriteLine(ex.ToString());
38             }
39         }
40
41         public MyFileArray(string filename, int n)
42         {
43             length = n;
44         }
45     }
46 }

```

```

45 1 reference
46 internal static string CreateString(int stringLength, Random rd)
47 {
48     const string allowedChars = "ABCDEFGHJKLMNOPQRSTUVWXYZ";
49     //const string allowedChars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
50     //const string allowedChars = "A";
51     char[] chars = new char[stringLength];
52
53     for (int i = 0; i < stringLength; i++)
54     {
55         chars[i] = allowedChars[rd.Next(0, allowedChars.Length)];
56     }
57
58     return new string(chars);
59 }
60 13 references
61 public FileStream fs { get; set; }
62 7 references
63 public override Objektas this[int index]
64 {
65     get
66     {
67         Byte[] data = new Byte[8];
68         fs.Seek(8 * index, SeekOrigin.Begin);
69         fs.Read(data, 0, 8);
70         string s = Encoding.ASCII.GetString(data.Take(4).ToArray());
71         float dataFloat = BitConverter.ToSingle(data, 4);
72         return new Objektas(s, dataFloat);
73     }
74 }
75 1 reference
76 public override void Swap(int j, Objektas a, Objektas b)
77 {
78     Byte[] data = new Byte[16];
79     BitConverter.GetBytes(a.flo).CopyTo(data, 0);
80     BitConverter.GetBytes(b.flo).CopyTo(data, 8);
81     fs.Seek(8 * (j - 1), SeekOrigin.Begin);
82     fs.Write(data, 0, 16);
83 }
84 3 references
85 public override void SetValue(int i, Objektas v)
86 {
87     Byte[] dataStr = Encoding.ASCII.GetBytes(v.str);
88     Byte[] dataFloat = new Byte[8];
89     BitConverter.GetBytes(v.flo).CopyTo(dataFloat, 4);
90
91     fs.Seek(8 * i, SeekOrigin.Begin);
92     fs.Write(dataStr, 0, 4);
93     fs.Write(dataFloat, 4, 4);
94 }
95 3 references
96 public override void Print(int n)
97 {
98     for (int i = 0; i < n; i++)
99     {
100         Byte[] data = new Byte[8];
101         fs.Seek(8 * i, SeekOrigin.Begin);
102         fs.Read(data, 0, 8);
103
104         string s = Encoding.ASCII.GetString(data.Take(4).ToArray());
105         float dataFloat = BitConverter.ToSingle(data, 4);
106         Console.WriteLine("{0}, {1:F5}", s, dataFloat);
107     }
108 }
109 }
110

```