Operatyvioji atmintis

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

namespace AlgL1
{
    0 references
    class BucketSort
    {
        0 references
        private static void Main(string[] args)
        {
            int seed = (int)DateTime.Now.Ticks & 0x0000FFFF;
            int[] numOfData = { 1000, 2000, 3000, 20000, 60000 };


            //Test_Array_List(seed);//Rikiavimas
            Benchmark(seed, numOfData);
        }
        #region Array
        2 references
        private static void BucketSortArray(DataArray x)
        {
            int numOfBuckets = 10;

            List<Objektas>[] buckets = new List<Objektas>[numOfBuckets];
            for (int i = 0; i < numOfBuckets; i++)
            {
                buckets[i] = new List<Objektas>();
            }

            for (int i = 0; i < x.Length; i++)
            {
                int bucket = (int)(x[i].flo * numOfBuckets);
                buckets[bucket].Add(x[i]);
            }
            int a = 0;
            for (int i = 0; i < numOfBuckets; i++)
            {
                BubbleSort(buckets[i]);
                for (int j = 0; j < buckets[i].Count; j++)
                {
                    x.Change(a, buckets[i][j]);
                    a++;
```

```
                }
            }
        }
        #endregion
        2 references
        private static void BucketSortList(DataList x)
        {
            int numOfBuckets = 10;

            List<Objektas>[] buckets = new List<Objektas>[numOfBuckets];
            for (int i = 0; i < numOfBuckets; i++)
            {
                buckets[i] = new List<Objektas>();
            }

            Objektas temp = x.Head();
            buckets[(int)(temp.flo * numOfBuckets)].Add(temp);
            for (int i = 1; i < x.Length; i++)
            {
                temp = x.Next();
                int bucket = (int)(temp.flo * 10);
                buckets[bucket].Add(temp);
            }
            int a = 0;
            x.clear();
            for (int i = 0; i < numOfBuckets; i++)
            {
                BubbleSort(buckets[i]);
                x.addAll(buckets[i]);
            }
        }
        2 references
        public static void BubbleSort(List<Objektas> items)
        {
            Objektas prevdata, currentdata;
            for (int i = items.Count - 1; i >= 0; i--)
            {
                currentdata = items[0];
                for (int j = 1; j <= i; j++)
                {
                    prevdata = currentdata;
                    currentdata = items[j];
                    if (prevdata > currentdata)
                    {
                        items[j - 1] = currentdata;
                        items[j] = prevdata;
```

```
                        currentdata = prevdata;
                    }
                }
            }
        }
        0 references
        public static void Test_Array_List(int seed)
        {
            int n = 12;
            MyDataArray myarray = new MyDataArray(n, seed);
            Console.WriteLine("\n ARRAY \n");
            myarray.Print(n);
            BucketSortArray(myarray);
            Console.WriteLine("\n SORTED \n");
            myarray.Print(n);

            MyDataList mylist = new MyDataList(n, seed);
            Console.WriteLine("\n LIST \n");
            mylist.Print(n);
            Console.WriteLine("\n SORTED \n");
            BucketSortList(mylist);
            mylist.Print(n);
        }
        1 reference
        public static void Benchmark(int seed, int[] dataCount)
        {
            Console.WriteLine("Array");
            for (int i = 0; i < dataCount.Length; i++)
            {
                int n = dataCount[i];
                MyDataArray myarray = new MyDataArray(n, seed);
                var benchmark = Stopwatch.StartNew();
                BucketSortArray(myarray);
                benchmark.Stop();
                Console.WriteLine("{0} - {1}", dataCount[i], benchmark.Elapsed);
            }
            Console.WriteLine("LinkedList");
            for (int i = 0; i < dataCount.Length; i++)
            {
                int n = dataCount[i];
                MyDataList mylist = new MyDataList(n, seed);
                var benchmark = Stopwatch.StartNew();
                BucketSortList(mylist);
                benchmark.Stop();
                Console.WriteLine("{0} - {1}", dataCount[i], benchmark.Elapsed);
            }
```

```
        }
    }

}
```

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

namespace AlgL1
{

    abstract class DataArray
    {
        protected int length;

        public int Length { get { return length; } }

        public abstract Objektas this[int index] { get; }

        public abstract void Change(int index, Objektas naujas);

        public abstract void Swap(int j, Objektas a, Objektas b);

        public void Print(int n)
        {
            for (int i = 0; i < n; i++)
            {
                Console.WriteLine(" {0:F5} {1}", this[i].flo, this[i].str);
                Console.WriteLine();
            }
        }
    }
}
```

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

namespace AlgL1
{
    2 references
    abstract class DataList
    {
        protected int length;
        1 reference
        public int Length { get { return length; } }
        4 references
        public abstract Objektas Head();
        4 references
        public abstract Objektas Next();
        1 reference
        public abstract void Swap(Objektas a, Objektas b);

        public abstract void addAll(List<Objektas> items);

        public abstract void clear();

        public void Print(int n)
        {
            Console.Write(" {0:F5} {1}", Head().flo, Head().str);
            Objektas tee = Next();
            while (tee != null)
            {
                Console.Write(" {0:F5} {1}", tee.flo, tee.str);
                tee = Next();
            }
            Console.WriteLine();
        }
    }
}
```

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

namespace AlgL1
{

    class Objektas
    {

        public string str { get; set; }

        public float flo { get; set; }



        public Objektas(float newF, string newS)
        {
            str = newS;
            flo = newF;
        }


        public static bool operator <(Objektas lhs, Objektas rhs)
        {
            if (lhs.flo == rhs.flo)
                return lhs.str.CompareTo(rhs.str) == 1;

            return lhs.flo < rhs.flo;
        }

        public static bool operator >(Objektas lhs, Objektas rhs)
        {
            if (lhs.flo == rhs.flo)
                return lhs.str.CompareTo(rhs.str) == -1;

            return lhs.flo > rhs.flo;
        }
    }
}
```

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

namespace AlgL1
{
    5 references
    class MyDataArray : DataArray
    {
        Objektas[] data;
        2 references
        public MyDataArray(int n, int seed)
        {
            data = new Objektas[n];
            length = n;
            Random rand = new Random(seed);
            for(int i = 0; i < length; i++)
            {
                Objektas temp = new Objektas((float)rand.NextDouble(), CreateString(4, rand));
                data[i] = temp;
            }
        }
        1 reference
        internal static string CreateString(int stringLength, Random rd)
        {
            const string allowedChars = "ABCDEFGHJKLMNOPQRSTUVWXYZ";
            char[] chars = new char[stringLength];

            for (int i = 0; i < stringLength; i++)
            {
                chars[i] = allowedChars[rd.Next(0, allowedChars.Length)];
            }
            return new string(chars);
        }
        5 references
        public override Objektas this[int index]
        {
            get { return data[index]; }
        }
        1 reference
        public override void Swap(int j, Objektas a, Objektas b)
        {
            data[j - 1] = a;
            data[j] = b;
        }
        2 references
        public override void Change(int index, Objektas naujas)
        {
            data[index] = naujas;
        }
    }
}
```

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

namespace AlgL1
{

    class MyDataList : DataList
    {

        class MyLinkedListNode
        {

            public MyLinkedListNode nextNode { get; set; }

            public Objektas data { get; set; }

            public MyLinkedListNode(Objektas data)
            {
                this.data = data;
            }
        }
        MyLinkedListNode headNode;
        MyLinkedListNode prevNode;
        MyLinkedListNode currentNode;

        public MyDataList(int n, int seed)
        {
            length = n;
            Random rand = new Random(seed);
            headNode = new MyLinkedListNode(new Objektas((float)rand.NextDouble(), CreateString(4, rand)));
            currentNode = headNode;
            for (int i = 1; i < length; i++)
            {
                prevNode = currentNode;
                currentNode.nextNode = new MyLinkedListNode(new Objektas((float)rand.NextDouble(), CreateString(4, rand)));
                currentNode = currentNode.nextNode;
            }
            currentNode.nextNode = null;
        }
```

```
        2 references
        public override void clear()
        {
            headNode = null;
            prevNode = null;
            currentNode = null;
        }
        2 references
        public override void addAll(List<Objektas> items)
        {
            foreach (Objektas item in items)
            {
                if (headNode == null)
                {
                    headNode = new MyLinkedListNode(item);
                    currentNode = headNode;
                    continue;
                }
                prevNode = currentNode;
                currentNode.nextNode = new MyLinkedListNode(item);
                currentNode = currentNode.nextNode;
            }
            currentNode.nextNode = null;
        }
        4 references
        public override Objektas Head()
        {
            currentNode = headNode;
            prevNode = null;
            return currentNode.data;
        }
        4 references
        public override Objektas Next()
        {
            prevNode = currentNode;
            currentNode = currentNode.nextNode;
            if (currentNode == null) return null;
            return currentNode.data;
        }
        1 reference
        public override void Swap(Objektas a, Objektas b)
        {
            prevNode.data = a;
            currentNode.data = b;
        }
```

```
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

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

namespace AlgL1_2
{
    0 references
    class BucketSort
    {
        0 references
        class Bubble_Sort
        {
            0 references
            private static void Main(string[] args)
            {
                int[] numOfData = { 100, 200, 300, 2000, 6000 };
                int seed = (int)DateTime.Now.Ticks & 0x0000FFFF;
                Benchmark(seed, numOfData);
                //Test_File_Array_List(seed);
            }
            1 reference
            private static void Benchmark(int seed, int[] dataCount)
            {
                Console.WriteLine("Array");
                for (int i = 0; i < dataCount.Length; i++)
                {
                    int n = dataCount[i];
                    string filename = @"mydataarray.dat";
                    MyFileArray myfilearray = new MyFileArray(filename, n, seed);
                    var benchmark = Stopwatch.StartNew();
                    using (myfilearray.fs = new FileStream(filename, FileMode.Open, FileAccess.ReadWrite))
                        BucketSortArray(myfilearray);
                    benchmark.Stop();
                    Console.WriteLine("{0} - {1}", dataCount[i], benchmark.Elapsed);
                }
                Console.WriteLine("LinkedList");
                for (int i = 0; i < dataCount.Length; i++)
                {
                    int n = dataCount[i];
                    string filename = @"mydatalist.dat";
                    MyFileList myfilelist = new MyFileList(filename, n, seed);

                    var benchmark = Stopwatch.StartNew();
                    using (myfilelist.fs = new FileStream(filename, FileMode.Open, FileAccess.ReadWrite))
                        BucketSortList(myfilelist);
                    benchmark.Stop();
                    Console.WriteLine("{0} - {1}", dataCount[i], benchmark.Elapsed);
                }
            }
            0 references
            public static void Test_File_Array_List(int seed)
            {
                int n = 12;
                string filename = @"mydataarray.dat";
                MyFileArray myfilearray = new MyFileArray(filename, n, seed);
                using (myfilearray.fs = new FileStream(filename, FileMode.Open, FileAccess.ReadWrite))
                {
                    Console.WriteLine("\n FILE ARRAY \n");
                    myfilearray.Print(n);

                    Console.WriteLine("\n SORTED FILE ARRAY \n");
                    MyFileArray ats = BucketSortArray(myfilearray);
                    using (ats.fs = new FileStream(@"ats.dat", FileMode.Open, FileAccess.ReadWrite))
                        ats.Print(n);
                }
                filename = @"mydatalist.dat";
                MyFileList myfilelist = new MyFileList(filename, n, seed);
                using (myfilelist.fs = new FileStream(filename, FileMode.Open, FileAccess.ReadWrite))
                {
                    Console.WriteLine("\n FILE LIST \n");
                    myfilelist.Print(n);
                    BucketSortList(myfilelist);
                    Console.WriteLine("\n SORTED FILE LIST \n");
                    MyFileList atss = BucketSortList(myfilelist);
                    using (atss.fs = new FileStream(@"Lats.dat", FileMode.Open, FileAccess.ReadWrite))
                        atss.Print(n);
                }
            }

```

```
3 references
private static MyFileList BucketSortList(DataList x)
{
    DirectoryInfo di = new DirectoryInfo(@"..\..\data2\");
    foreach (FileInfo file in di.GetFiles())
        if (file.Name.Contains("LBucket"))
            file.Delete();

    int[] lengths = new int[10];
    for (int i = 0; i < 10; i++)
        lengths[i] = 0;

    for (int i = 0; i < x.Length; i++)
    {
        Objektas temp;
        int bucket;
        if (i == 0)
        {
            temp = x.Head();
            bucket = (int)(temp.flo * 10);
        }
        else
        {
            temp = x.Next();
            bucket = (int)(temp.flo * 10);
        }

        string fileName = "LBucket" + bucket + ".dat";
        string path = @"..\..\data2\" + fileName;

        if (!File.Exists(path))
            using (BinaryWriter writer = new BinaryWriter(File.Open(path, FileMode.Create)))
            {
                writer.Write(4);
                Byte[] str = Encoding.ASCII.GetBytes(temp.str);
                writer.Write(str);
                writer.Write(temp.flo);
                writer.Write((lengths[bucket] + 1) * 12 + 4);
                lengths[bucket] += 1;
            }
        else
            using (BinaryWriter writer = new BinaryWriter(File.Open(path, FileMode.Append)))
            {
                Byte[] str = Encoding.ASCII.GetBytes(temp.str);
                writer.Write(str);
                writer.Write(temp.flo);
                writer.Write((lengths[bucket] + 1) * 12 + 4);
                lengths[bucket] += 1;
            }
    }
    MyFileList ats = new MyFileList(@"Lats.dat", x.Length);
    using (BinaryWriter writer = new BinaryWriter(File.Open(@"Lats.dat", FileMode.Create)))
    {
        writer.Write(4);
        int ind = 0;
        foreach (FileInfo file in di.GetFiles())
        {
            int length = lengths[int.Parse(file.Name.Substring(7, 1))];
            MyFileList buck = new MyFileList(@"..\..\data2\" + file.Name, length);

            using (buck.fs = new FileStream(@"..\..\data2\" + file.Name, FileMode.Open, FileAccess.ReadWrite))
            {
                //buck.Print(buck.Length);
                InsertionSort(buck);

                for (int j = 0; j < length; j++)
                {
                    Objektas t;
                    if (j == 0)
                        t = buck.Head();
                    else
                        t = buck.Next();
                    Byte[] str = Encoding.ASCII.GetBytes(t.str);
                    writer.Write(str);
                    writer.Write(t.flo);
                    writer.Write((ind + 1) * 12 + 4);
                    ind++;
                }
            }
        }
    }
    return ats;
}
```

```
2 references
private static MyFileArray BucketSortArray(DataArray x)
{
    DirectoryInfo di = new DirectoryInfo(@"..\..\data\");
    foreach (FileInfo file in di.GetFiles())
        if (file.Name.Contains("ABucket"))
            file.Delete();

    int[] lengths = new int[10];

    for (int i = 0; i < x.Length; i++)
    {
        Objektas key = x[i];
        int bucket = (int)(key.flo * 10);
        string fileName = "ABucket" + bucket + ".dat";
        string path = @"..\..\data\" + fileName;

        if (!File.Exists(@"..\..\data\" + fileName))
            using (BinaryWriter writer = new BinaryWriter(File.Open(path, FileMode.Create)))
            {
                Byte[] str = Encoding.ASCII.GetBytes(key.str);
                writer.Write(str);
                writer.Write(key.flo);
                lengths[bucket] = 1;
            }
        else
            using (BinaryWriter writer = new BinaryWriter(File.Open(path, FileMode.Append)))
            {
                Byte[] str = Encoding.ASCII.GetBytes(key.str);
                writer.Write(str);
                writer.Write(key.flo);
                lengths[bucket] += 1;
            }
    }
    MyFileArray ats = new MyFileArray(@"ats.dat", x.Length);
    using (BinaryWriter writer = new BinaryWriter(File.Open(@"ats.dat", FileMode.Create)))
    {
        foreach (FileInfo file in di.GetFiles())
        {
            int length = lengths[int.Parse(file.Name.Substring(7, 1))];
            MyFileArray buck = new MyFileArray(@"..\..\data\" + file.Name, length);

            using (buck.fs = new FileStream(@"..\..\data\" + file.Name, FileMode.Open, FileAccess.ReadWrite))
            {
                InsertionSort(buck);

                for (int j = 0; j < length; j++)
                {
                    Byte[] str = Encoding.ASCII.GetBytes(buck[j].str);
                    writer.Write(str);
                    writer.Write(buck[j].flo);
                }
            }
        }
    }
    return ats;
}
1 reference
public static void InsertionSort(DataArray myArray)
{
    int n = myArray.Length;
    for (int i = 1; i < n; ++i)
    {
        Objektas key = myArray[i];
        int j = i - 1;

        while (j >= 0 && CompareV1(myArray[j], key) > 0)
        {
            myArray.SetValue(j + 1, myArray[j]);
            j = j - 1;
        }
        myArray.SetValue(j + 1, key);
    }
}
1 reference
public static void InsertionSort(DataList myList)
{
    int n = myList.Length;
    for (int i = 1; i < n; ++i)
    {
        Objektas key = myList.ElementAt(i);
        int j = i - 1;

        while (j >= 0 && CompareV1(myList.ElementAt(j), key) > 0)
        {
            myList.SetValue(j + 1, myList.ElementAt(j));
            j = j - 1;
        }
        myList.SetValue(j + 1, key);
    }
}
```

```
        public static int CompareV1(Objektas a, Objektas b)
        {
            if (a.flo == b.flo)
                return a.str.CompareTo(b.str);
            else
                return a.flo.CompareTo(b.flo);
        }
    }
}
```

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

namespace AlgL1_2
{
    abstract class DataArray
    {
        protected int length;

        public int Length { get { return length; } }

        public abstract Objektas this[int index] { get; }

        public abstract void Swap(int j, Objektas a, Objektas b);

        public abstract void SetValue(int i, Objektas v);

        public abstract void Print(int n);
    }

}
```

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

namespace AlgL1_2
{
    3 references
    abstract class DataList
    {
        protected int length;
        5 references
        public int Length { get { return length; } }
        7 references
        public abstract Objektas Head();
        6 references
        public abstract Objektas Next();
        1 reference
        public abstract void Swap(Objektas a, Objektas b);
        3 references
        public abstract void SetValue(int i, Objektas v);
        4 references
        public abstract Objektas ElementAt(int n);
        2 references
        public void Print(int n)
        {
            Console.Write(" {0:F5} {1}", Head().flo, Head().str);
            for (int i = 1; i < n; i++)
            {
                Objektas temp = Next();
                Console.Write(" {0:F5} {1}", temp.flo, temp.str);
            }
            Console.WriteLine();
        }
    }
}
```

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

namespace AlgL1_2
{
    37 references
    class Objektas
    {
        25 references
        public float flo { get; set; }
        16 references
        public string str { get; set; }

        4 references
        public Objektas(string nstring, float nfloat)
        {
            flo = nfloat;
            str = nstring;
        }
        0 references
        public override string ToString()
        {
            return String.Format(" {0}, {1:F5}\n", str, flo);
        }

        0 references
        public string ToFileString()
        {
            return String.Format("{0}{1:F5}\n", str, flo);
        }
    }
}
```

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

namespace AlgL1_2
{
    12 references
    class MyFileList : DataList
    {
        int prevNode;
        int currentNode;
        int nextNode;
        2 references
        public MyFileList(string filename, int n, int seed)
        {
            length = n;
            Random rand = new Random(seed);
            if (File.Exists(filename)) File.Delete(filename);
            try
            {
                using (BinaryWriter writer = new BinaryWriter(File.Open(filename, FileMode.Create)))
                {
                    writer.Write(4);
                    for (int j = 0; j < length; j++)
                    {
                        Byte[] str = Encoding.ASCII.GetBytes(CreateString(4, rand));
                        writer.Write(str);
                        writer.Write((float)rand.NextDouble());
                        writer.Write((j + 1) * 12 + 4);
                    }
                }
            }
            catch (IOException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
        2 references
        public MyFileList(string filename, int n)
        {
            length = n;
        }
```

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

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

```
1 reference
internal static string CreateString(int stringLength, Random rd)
{
    const string allowedChars = "ABCDEFGHJKLMNOPQRSTUVWXYZ";
    //const string allowedChars = "ABCDEFGHJKLMNOPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz";
    //const string allowedChars = "A";
    char[] chars = new char[stringLength];

    for (int i = 0; i < stringLength; i++)
    {
        chars[i] = allowedChars[rd.Next(0, allowedChars.Length)];
    }

    return new string(chars);
}
13 references
public FileStream fs { get; set; }
7 references
public override Objektas this[int index]
{
    get
    {
        Byte[] data = new Byte[8];
        fs.Seek(8 * index, SeekOrigin.Begin);
        fs.Read(data, 0, 8);
        string s = Encoding.ASCII.GetString(data.Take(4).ToArray());
        float dataFloat = BitConverter.ToSingle(data, 4);
        return new Objektas(s, dataFloat);
    }
}
1 reference
public override void Swap(int j, Objektas a, Objektas b)
{
    Byte[] data = new Byte[16];
    BitConverter.GetBytes(a.flo).CopyTo(data, 0);
    BitConverter.GetBytes(b.flo).CopyTo(data, 8);
    fs.Seek(8 * (j - 1), SeekOrigin.Begin);
    fs.Write(data, 0, 16);
}
```

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

namespace AlgL1_2
{
    class MyFileArray : DataArray
    {
        public MyFileArray(string filename, int n, int seed)
        {
            Objektas[] data = new Objektas[n];
            length = n;
            Random rand = new Random(seed);
            for (int i = 0; i < length; i++)
            {
                data[i] = new Objektas(CreateString(4,rand),(float)rand.NextDouble());
            }
            if (File.Exists(filename)) File.Delete(filename);
            try
            {
                using (BinaryWriter writer = new BinaryWriter(File.Open(filename, FileMode.Create)))
                {
                    for (int j = 0; j < length; j++)
                    {
                        Byte[] str = Encoding.ASCII.GetBytes(data[j].str);
                        writer.Write(str);
                        writer.Write(data[j].flo);
                    }
                        //writer.Write(data[j]);
                }
            }
            catch (IOException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }

        public MyFileArray(string filename, int n)
        {
            length = n;
        }

```

```
1 reference
internal static string CreateString(int stringLength, Random rd)
{
    const string allowedChars = "ABCDEFGHJKLMNOPQRSTUVWXYZ";
    //const string allowedChars = "ABCDEFGHJKLMNOPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz";
    //const string allowedChars = "A";
    char[] chars = new char[stringLength];

    for (int i = 0; i < stringLength; i++)
    {
        chars[i] = allowedChars[rd.Next(0, allowedChars.Length)];
    }

    return new string(chars);
}
13 references
public FileStream fs { get; set; }
7 references
public override Objektas this[int index]
{
    get
    {
        Byte[] data = new Byte[8];
        fs.Seek(8 * index, SeekOrigin.Begin);
        fs.Read(data, 0, 8);
        string s = Encoding.ASCII.GetString(data.Take(4).ToArray());
        float dataFloat = BitConverter.ToSingle(data, 4);
        return new Objektas(s, dataFloat);
    }
}
1 reference
public override void Swap(int j, Objektas a, Objektas b)
{
    Byte[] data = new Byte[16];
    BitConverter.GetBytes(a.flo).CopyTo(data, 0);
    BitConverter.GetBytes(b.flo).CopyTo(data, 8);
    fs.Seek(8 * (j - 1), SeekOrigin.Begin);
    fs.Write(data, 0, 16);
}
3 references
public override void SetValue(int i, Objektas v)
{
    Byte[] dataStr = Encoding.ASCII.GetBytes(v.str);
    Byte[] dataFloat = new Byte[8];
    BitConverter.GetBytes(v.flo).CopyTo(dataFloat, 4);

    fs.Seek(8 * i, SeekOrigin.Begin);
    fs.Write(dataStr, 0, 4);
    fs.Write(dataFloat, 4, 4);
}
3 references
public override void Print(int n)
{
    for (int i = 0; i < n; i++)
    {
        Byte[] data = new Byte[8];
        fs.Seek(8 * i, SeekOrigin.Begin);
        fs.Read(data, 0, 8);

        string s = Encoding.ASCII.GetString(data.Take(4).ToArray());
        float dataFloat = BitConverter.ToSingle(data, 4);
        Console.WriteLine("{0}, {1:F5}", s, dataFloat);
    }
}
}
}
```