stream.cm

```
/*
    ========================================================================

    Copyright (c) 2012-2016 Seppo Laakko
    http://sourceforge.net/projects/cmajor/

    Distributed under the GNU General Public License, version 3 (GPLv3).
    (See accompanying LICENSE.txt or http://www.gnu.org/licenses/gpl.html
        )

    ========================================================================
    */

// Copyright (c) 1994
// Hewlett-Packard Company
// Copyright (c) 1996
// Silicon Graphics Computer Systems, Inc.
// Copyright (c) 2009 Alexander Stepanov and Paul McJones

using System;
using System.Collections;
using System.Concepts;

namespace System.IO
{
    public abstract class InputStream
    {
        public nothrow InputStream() {}
        public nothrow virtual ~InputStream()
        {
        }
        suppress InputStream(const InputStream&);
        suppress void operator=(const InputStream&);
        public nothrow default InputStream(InputStream&&);
        public nothrow default void operator=(InputStream&&);
        public abstract string ReadLine();
        public abstract string ReadToEnd();
        public nothrow abstract bool EndOfStream() const;
    }

    public abstract class OutputStream
    {
        public nothrow OutputStream() {}
        public nothrow virtual ~OutputStream()
        {
        }
        suppress OutputStream(const OutputStream&);
        suppress void operator=(const OutputStream&);
        public nothrow default OutputStream(OutputStream&&);
        public nothrow default void operator=(OutputStream&&);
        public abstract void Write(const char* s);
```

```
        public abstract void Write(const string& s);
        public abstract void Write(const wstring& s);
        public abstract void Write(const ustring& s);
        public abstract void Write(char c);
        public abstract void Write(wchar c);
        public abstract void Write(uchar c);
        public abstract void Write(byte b);
        public abstract void Write(sbyte b);
        public abstract void Write(short s);
        public abstract void Write(ushort u);
        public abstract void Write(int i);
        public abstract void Write(uint i);
        public abstract void Write(long l);
        public abstract void Write(ulong u);
        public abstract void Write(bool b);
        public abstract void Write(float f);
        public abstract void Write(double d);
        public abstract void WriteLine();
        public abstract void WriteLine(const char* s);
        public abstract void WriteLine(const string& s);
        public abstract void WriteLine(const wstring& s);
        public abstract void WriteLine(const ustring& s);
        public abstract void WriteLine(char c);
        public abstract void WriteLine(wchar c);
        public abstract void WriteLine(uchar c);
        public abstract void WriteLine(byte b);
        public abstract void WriteLine(sbyte b);
        public abstract void WriteLine(short s);
        public abstract void WriteLine(ushort u);
        public abstract void WriteLine(int i);
        public abstract void WriteLine(uint i);
        public abstract void WriteLine(long l);
        public abstract void WriteLine(ulong u);
        public abstract void WriteLine(bool b);
        public abstract void WriteLine(float f);
        public abstract void WriteLine(double d);
    }

    public OutputStream& operator<<(OutputStream& s, const string& str)
    {
        s.Write(str);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, const wstring& str)
    {
        s.Write(str);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, const ustring& str)
    {
        s.Write(str);
```

```
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, const char* str)
    {
        s.Write(str);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, char c)
    {
        s.Write(c);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, wchar c)
    {
        s.Write(c);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, uchar c)
    {
        s.Write(c);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, byte b)
    {
        s.Write(b);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, sbyte b)
    {
        s.Write(b);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, short x)
    {
        s.Write(x);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, ushort u)
    {
        s.Write(u);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, int i)
    {
```

```
        s.Write(i);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, uint u)
    {
        s.Write(u);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, long l)
    {
        s.Write(l);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, ulong u)
    {
        s.Write(u);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, bool b)
    {
        s.Write(b);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, float f)
    {
        s.Write(f);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, double d)
    {
        s.Write(d);
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, Date date)
    {
        s.Write(ToString(date));
        return s;
    }

    public OutputStream& operator<<(OutputStream& s, EndLine)
    {
        s.WriteLine();
        return s;
    }
```

```
    public OutputStream& operator<<<C>(OutputStream& s, const C& c) where
        C is ForwardContainer and C.ValueType is int
    {
        bool first = true;
        for (int i : c)
        {
            if (first)
            {
                first = false;
            }
            else
            {
                s.Write(", ");
            }
            s.Write(i);
        }
        return s;
    }
}

namespace System
{

//  note: EndLine class and endl() function is placed to System namespace
//        so that they will be available to code using System.Console.

    public class EndLine
    {
        public nothrow inline EndLine()
        {
        }
        public nothrow inline EndLine(const EndLine&)
        {
        }
    }

    public nothrow inline EndLine endl()
    {
        return EndLine();
    }
}
```