A C# chunk of code using *minted*

```
using System;
using System.Runtime.InteropServices;

namespace Binarysharp.MemoryManagement.Memory
{
        /// <summary>
        /// Class representing a block of memory allocated in the
        ↪  local process.
        /// </summary>
        public class LocalUnmanagedMemory : IDisposable
        {
                #region Properties
                /// <summary>
                /// The address where the data is allocated.
                /// </summary>
                public IntPtr Address { get; private set; }
                /// <summary>
                /// The size of the allocated memory.
                /// </summary>
                public int Size { get; private set; }
                #endregion

                #region Constructor/Destructor
                /// <summary>
                /// Initializes a new instance of the <see
                ↪  cref="LocalUnmanagedMemory"/> class,
                ↪  allocating a block of memory in the local
                ↪  process.
                /// </summary>
                /// <param name="size">The size to
                ↪  allocate.</param>
                public LocalUnmanagedMemory(int size)
                {
                        // Allocate the memory
                        Size = size;
                        Address = Marshal.AllocHGlobal(Size);
                }
                /// <summary>
                /// Frees resources and perform other cleanup
                ↪  operations before it is reclaimed by garbage
                ↪  collection.
                /// </summary>
                ~LocalUnmanagedMemory()
                {
```

```
38                      Dispose();
39              }
40              #endregion
41
42              #region Methods
43              #region Dispose (implementation of IDisposable)
44              /// <summary>
45              /// Releases the memory held by the <see
                ↪  cref="LocalUnmanagedMemory"/> object.
46              /// </summary>
47              public virtual void Dispose()
48              {
49                      // Free the allocated memory
50                      Marshal.FreeHGlobal(Address);
51                      // Remove the pointer
52                      Address = IntPtr.Zero;
53                      // Avoid the finalizer
54                      GC.SuppressFinalize(this);
55              }
56              #endregion
57              #region Read
58              /// <summary>
59              /// Reads data from the unmanaged block of
                ↪  memory.
60              /// </summary>
61              /// <typeparam name="T">The type of data to
                ↪  return.</typeparam>
62              /// <returns>The return value is the block of
                ↪  memory casted in the specified
                ↪  type.</returns>
63              public T Read<T>()
64              {
65                      // Marshal data from the block of memory
                        ↪  to a new allocated managed object
66                      return (T)Marshal.PtrToStructure(Address,
                        ↪  typeof(T));
67              }
68              /// <summary>
69              /// Reads an array of bytes from the unmanaged
                ↪  block of memory.
70              /// </summary>
71              /// <returns>The return value is the block of
                ↪  memory.</returns>
72              public byte[] Read()
73              {
74                      // Allocate an array to store data
```

```
75                          var bytes = new byte[Size];
76                          // Copy the block of memory to the array
77                          Marshal.Copy(Address, bytes, 0, Size);
78                          // Return the array
79                          return bytes;
80                  }
81                  #endregion
82          }
83  }
```