

LinksPlatform's Platform.Memory Class Library

1.1 ./Platform.Memory/ArrayMemory.cs

```
1 using System.Runtime.CompilerServices;
2
3 namespace Platform.Memory
4 {
5     /// <summary>
6     /// <para>Represents a memory block with access via indexer.</para>
7     /// <para>Представляет блок памяти с доступом через индекатор.</para>
8     /// </summary>
9     /// <typeparam name="TElement"><para>Element type.</para><para>Тип
10     ↪ элемента.</para></typeparam>
11     public class ArrayMemory<TElement> : IArrayMemory<TElement>
12     {
13         #region Fields
14
15         private readonly TElement[] _array;
16
17         #endregion
18
19         #region Properties
20
21         /// <inheritdoc/>
22         /// <include file='bin\Release\netstandard2.0\Platform.Memory.xml'
23         ↪ path='doc/members/member[@name="P:Platform.Memory.IMemory.Size"]/*' />
24         public long Size
25         {
26             [MethodImpl(MethodImplOptions.AggressiveInlining)]
27             get => _array.Length;
28         }
29
30         /// <inheritdoc/>
31         /// <include file='bin\Release\netstandard2.0\Platform.Memory.xml' path='doc/members/mem_
32         ↪ ber[@name="P:Platform.Memory.IArrayMemory`1.Item(System.Int64)"]/*' />
33         public TElement this[long index]
34         {
35             [MethodImpl(MethodImplOptions.AggressiveInlining)]
36             get => _array[index];
37             [MethodImpl(MethodImplOptions.AggressiveInlining)]
38             set => _array[index] = value;
39         }
40
41         #endregion
42
43         #region Constructors
44
45         /// <summary>
46         /// <para>Initializes a new instance of the <see cref="ArrayMemory{TElement}" />
47         ↪ class.</para>
48         /// <para>Инициализирует новый экземпляр класса <see
49         ↪ cref="ArrayMemory{TElement}" />.</para>
50         /// </summary>
51         /// <param name="size"><para>Size in bytes.</para><para>Размер в байтах.</para></param>
52         [MethodImpl(MethodImplOptions.AggressiveInlining)]
53         public ArrayMemory(long size) => _array = new TElement[size];
54
55         #endregion
56     }
57 }
```

1.2 ./Platform.Memory/DirectMemoryAsArrayMemoryAdapter.cs

```
1 using System;
2 using System.Runtime.CompilerServices;
3 using Platform.Disposables;
4 using Platform.Exceptions;
5 using Platform.Unsafe;
6
7 namespace Platform.Memory
8 {
9     /// <summary>
10     /// <para>Represents adapter to a memory block with access via indexer.</para>
11     /// <para>Представляет адаптер к блоку памяти с доступом через индекатор.</para>
12     /// </summary>
13     /// <typeparam name="TElement"><para>Element type.</para><para>Тип
14     ↪ элемента.</para></typeparam>
15     public class DirectMemoryAsArrayMemoryAdapter<TElement> : DisposableBase,
16     ↪ IArrayMemory<TElement>, IDirectMemory
17     {
18         where TElement : struct
19     }
```

```

17 #region Fields
18
19 private readonly IDirectMemory _memory;
20
21 #endregion
22
23 #region Properties
24
25 /// <inheritdoc>
26 /// <include file='bin\Release\netstandard2.0\Platform.Memory.xml'
27   ↳ path='doc/members/member[@name="P:Platform.Memory.IMemory.Size"]/*' />
28 public long Size
29 {
30     [MethodImpl(MethodImplOptions.AggressiveInlining)]
31     get => _memory.Size;
32 }
33
34 /// <inheritdoc>
35 /// <include file='bin\Release\netstandard2.0\Platform.Memory.xml'
36   ↳ path='doc/members/member[@name="P:Platform.Memory.IDirectMemory.Pointer"]/*' />
37 public IntPtr Pointer
38 {
39     [MethodImpl(MethodImplOptions.AggressiveInlining)]
40     get => _memory.Pointer;
41 }
42
43 /// <inheritdoc>
44 /// <include file='bin\Release\netstandard2.0\Platform.Memory.xml' path='doc/members/mem_
45   ↳ ber[@name="P:Platform.Memory.IArrayMemory`1.Item(System.Int64)"]/*' />
46 public TElement this[long index]
47 {
48     [MethodImpl(MethodImplOptions.AggressiveInlining)]
49     get => Pointer.ReadElementValue<TElement>(index);
50     [MethodImpl(MethodImplOptions.AggressiveInlining)]
51     set => Pointer.WriteElementValue(index, value);
52 }
53
54 #endregion
55
56 #region DisposableBase Properties
57
58 /// <inheritdoc>
59 protected override string ObjectName
60 {
61     [MethodImpl(MethodImplOptions.AggressiveInlining)]
62     get => $"Array as memory block at '{Pointer}' address.";
63 }
64
65 #endregion
66
67 #region Constructors
68
69 /// <summary>
70 /// <para>Initializes a new instance of the <see
71   ↳ cref="DirectMemoryAsArrayMemoryAdapter{TElement}> class.</para>
72 /// <para>Инициализирует новый экземпляр класса <see
73   ↳ cref="DirectMemoryAsArrayMemoryAdapter{TElement}>.</para>
74 /// </summary>
75 /// <param name="memory"><para>An object implementing <see cref="IDirectMemory">
76   ↳ interface.</para><para>Объект, реализующий интерфейс <see
77   ↳ cref="IDirectMemory">.</para></param>
78 [MethodImpl(MethodImplOptions.AggressiveInlining)]
79 public DirectMemoryAsArrayMemoryAdapter(IDirectMemory memory)
80 {
81     Ensure.Always.ArgumentNotNull(memory, nameof(memory));
82     Ensure.Always.ArgumentMeetsCriteria(memory, m => (m.Size % Structure<TElement>.Size)
83       ↳ == 0, nameof(memory), "Memory is not aligned to element size.");
84     _memory = memory;
85 }
86
87 #endregion
88
89 #region DisposableBase Methods
90
91 /// <inheritdoc>
92 [MethodImpl(MethodImplOptions.AggressiveInlining)]
93 protected override void Dispose(bool manual, bool wasDisposed)
94 {
95     if (!wasDisposed)

```

```

88         {
89             _memory.DisposeIfPossible();
90         }
91     }
92
93     #endregion
94 }
95 }

```

1.3 ./Platform.Memory/FileArrayMemory.cs

```

1  using System.IO;
2  using System.Runtime.CompilerServices;
3  using Platform.Disposables;
4  using Platform.Unsafe;
5  using Platform.IO;
6
7  namespace Platform.Memory
8  {
9      /// <summary>
10     /// <para>Represents a memory block with access via indexer and stored as file on
11     ///   ↳ disk.</para>
12     /// <para>Представляет блок памяти с доступом через индексатор и хранящийся в виде файла на
13     ///   ↳ диске.</para>
14     /// </summary>
15     /// <typeparam name="TElement"><para>Element type.</para><para>Тип
16     ///   ↳ элемента.</para></typeparam>
17     public class FileArrayMemory<TElement> : DisposableBase, IArrayMemory<TElement> //-V3073
18     where TElement : struct
19     {
20         #region Fields
21
22         private readonly FileStream _file;
23
24         #endregion
25
26         #region Properties
27
28         /// <inheritdoc/>
29         /// <include file='bin\Release\netstandard2.0\Platform.Memory.xml'
30         ///   ↳ path='doc/members/member[@name="P:Platform.Memory.IMemory.Size"]/*' />
31         public long Size
32         {
33             [MethodImpl(MethodImplOptions.AggressiveInlining)]
34             get => _file.Length;
35         }
36
37         /// <inheritdoc/>
38         /// <include file='bin\Release\netstandard2.0\Platform.Memory.xml' path='doc/members/mem
39         ///   ↳ ber[@name="P:Platform.Memory.IArrayMemory`1.Item(System.Int64)"]/*' />
40         public TElement this[long index]
41         {
42             [MethodImpl(MethodImplOptions.AggressiveInlining)]
43             get
44             {
45                 _file.Seek(Structure<TElement>.Size * index, SeekOrigin.Begin);
46                 return _file.ReadOrDefault<TElement>();
47             }
48             [MethodImpl(MethodImplOptions.AggressiveInlining)]
49             set
50             {
51                 _file.Seek(Structure<TElement>.Size * index, SeekOrigin.Begin);
52                 _file.Write(value);
53             }
54         }
55
56         #endregion
57
58         #region DisposableBase Properties
59
60         /// <inheritdoc/>
61         protected override string ObjectName
62         {
63             [MethodImpl(MethodImplOptions.AggressiveInlining)]
64             get => $"File stored memory block at '{_file.Name}' path.";
65         }
66
67         #endregion
68
69         #region Constructors

```

```

66     /// <summary>
67     /// <para>Initializes a new instance of the <see cref="FileArrayMemory{TElement}" />
    ↪ class.</para>
68     /// <para>Инициализирует новый экземпляр класса <see
    ↪ cref="FileArrayMemory{TElement}" />.</para>
69     /// </summary>
70     /// <param name="file"><para>File stream.</para><para>Файловый поток.</para></param>
71     [MethodImpl(MethodImplOptions.AggressiveInlining)]
72     public FileArrayMemory(FileStream file) => _file = file;
73
74     /// <summary>
75     /// <para>Initializes a new instance of the <see cref="FileArrayMemory{TElement}" />
    ↪ class.</para>
76     /// <para>Инициализирует новый экземпляр класса <see
    ↪ cref="FileArrayMemory{TElement}" />.</para>
77     /// </summary>
78     /// <param name="path"><para>An path to file.</para><para>Путь к файлу.</para></param>
79     [MethodImpl(MethodImplOptions.AggressiveInlining)]
80     public FileArrayMemory(string path) : this(File.Open(path, FileMode.OpenOrCreate)) { }
81
82     #endregion
83
84     #region DisposableBase Methods
85
86     /// <inheritdoc>
87     [MethodImpl(MethodImplOptions.AggressiveInlining)]
88     protected override void Dispose(bool manual, bool wasDisposed)
89     {
90         if (!wasDisposed)
91         {
92             _file.DisposeIfPossible();
93         }
94     }
95
96     #endregion
97 }
98 }

```

1.4 ./Platform.Memory/FileMappedResizableDirectMemory.cs

```

1  using System;
2  using System.IO;
3  using System.IO.MemoryMappedFiles;
4  using System.Runtime.CompilerServices;
5  using Platform.Disposables;
6  using Platform.Exceptions;
7  using Platform.Collections;
8  using Platform.IO;
9
10 namespace Platform.Memory
11 {
12     /// <summary>
13     /// <para>Represents a memory block stored as a file on disk.</para>
14     /// <para>Представляет блок памяти, хранящийся в виде файла на диске.</para>
15     /// </summary>
16     public unsafe class FileMappedResizableDirectMemory : ResizableDirectMemoryBase
17     {
18         #region Fields
19
20         private MemoryMappedFile _file;
21         private MemoryMappedViewAccessor _accessor;
22
23         /// <summary>
24         /// <para>Gets path to memory mapped file.</para>
25         /// <para>Получает путь к отображенному в памяти файлу.</para>
26         /// </summary>
27         protected readonly string Path;
28
29         #endregion
30
31         #region DisposableBase Properties
32
33         /// <inheritdoc>
34         protected override string ObjectName
35         {
36             [MethodImpl(MethodImplOptions.AggressiveInlining)]
37             get => $"File stored memory block at '{Path}' path.";
38         }
39
40         #endregion
41

```

```
#region Constructors
```

```
/// <summary>
/// <para>Initializes a new instance of the <see
→ cref="FileMappedResizableDirectMemory"/> class.</para>
/// <para>Инициализирует новый экземпляр класса <see
→ cref="FileMappedResizableDirectMemory"/>.</para>
/// </summary>
/// <param name="path"><para>An path to file.</para><para>Путь к файлу.</para></param>
/// <param name="minimumReservedCapacity"><para>Minimum file size in
→ bytes.</para><para>Минимальный размер файла в байтах.</para></param>
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public FileMappedResizableDirectMemory(string path, long minimumReservedCapacity)
{
    Ensure.Always.ArgumentNotEmptyAndNotWhiteSpace(path, nameof(path));
    if (minimumReservedCapacity < MinimumCapacity)
    {
        minimumReservedCapacity = MinimumCapacity;
    }
    Path = path;
    var size = FileHelpers.GetSize(Path);
    ReservedCapacity = size > minimumReservedCapacity ? ((size /
→ minimumReservedCapacity) + 1) * minimumReservedCapacity :
→ minimumReservedCapacity;
    UsedCapacity = size;
}

/// <summary>
/// <para>Initializes a new instance of the <see
→ cref="FileMappedResizableDirectMemory"/> class.</para>
/// <para>Инициализирует новый экземпляр класса <see
→ cref="FileMappedResizableDirectMemory"/>.</para>
/// </summary>
/// <param name="address"><para>An path to file.</para><para>Путь к файлу.</para></param>
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public FileMappedResizableDirectMemory(string address) : this(address, MinimumCapacity)
→ { }
```

```
#endregion
```

```
#region Methods
```

```
[MethodImpl(MethodImplOptions.AggressiveInlining)]
private void MapFile(long capacity)
{
    if (Pointer != IntPtr.Zero)
    {
        return;
    }
    _file = MemoryMappedFile.CreateFromFile(Path, FileMode.Open, mapName: null,
→ capacity, MemoryMappedFileAccess.ReadWrite);
    _accessor = _file.CreateViewAccessor();
    byte* pointer = null;
    _accessor.SafeMemoryMappedViewHandle.AcquirePointer(ref pointer);
    Pointer = new IntPtr(pointer);
}
```

```
[MethodImpl(MethodImplOptions.AggressiveInlining)]
private void UnmapFile()
{
    if (UnmapFile(Pointer))
    {
        Pointer = IntPtr.Zero;
    }
}
```

```
[MethodImpl(MethodImplOptions.AggressiveInlining)]
private bool UnmapFile(IntPtr pointer)
{
    if (pointer == IntPtr.Zero)
    {
        return false;
    }
    if (_accessor != null)
    {
        _accessor.SafeMemoryMappedViewHandle.ReleasePointer();
        Disposable.TryDisposeAndResetToDefault(ref _accessor);
    }
}
```

```

111         Disposable.TryDisposeAndResetToDefault(ref _file);
112         return true;
113     }
114
115     #endregion
116
117     #region ResizableDirectMemoryBase Methods
118
119     /// <inheritdoc/>
120     /// <include file='bin\Release\netstandard2.0\Platform.Memory.xml'
121     → path='doc/members/member[@name="M:Platform.Memory.ResizableDirectMemoryBase.OnReservedCapacityChanged(System.Int64,System.Int64)"]/*' />
122     [MethodImpl(MethodImplOptions.AggressiveInlining)]
123     protected override void OnReservedCapacityChanged(long oldReservedCapacity, long
124     → newReservedCapacity)
125     {
126         UnmapFile();
127         FileHelpers.SetSize(Path, newReservedCapacity);
128         MapFile(newReservedCapacity);
129     }
130
131     /// <inheritdoc/>
132     /// <include file='bin\Release\netstandard2.0\Platform.Memory.xml'
133     → path='doc/members/member[@name="M:Platform.Memory.ResizableDirectMemoryBase.DisposePointer(System.IntPtr,System.Int64)"]/*' />
134     [MethodImpl(MethodImplOptions.AggressiveInlining)]
135     protected override void DisposePointer(IntPtr pointer, long usedCapacity)
136     {
137         if (UnmapFile(pointer))
138         {
139             FileHelpers.SetSize(Path, usedCapacity);
140         }
141     }
142
143     #endregion
144 }

```

1.5 ./Platform.Memory/HeapResizableDirectMemory.cs

```

1 using System;
2 using System.Runtime.CompilerServices;
3 using System.Runtime.InteropServices;
4 using Platform.Unsafe;
5
6 namespace Platform.Memory
7 {
8     /// <summary>
9     /// <para>Represents a memory block allocated in Heap.</para>
10    /// <para>Представляет блок памяти, выделенный в "куче".</para>
11    /// </summary>
12    public unsafe class HeapResizableDirectMemory : ResizableDirectMemoryBase
13    {
14        #region DisposableBase Properties
15
16        /// <inheritdoc/>
17        protected override string ObjectName
18        {
19            [MethodImpl(MethodImplOptions.AggressiveInlining)]
20            get => $"Heap stored memory block at {Pointer} address.";
21        }
22
23        #endregion
24
25        #region Constructors
26
27        /// <summary>
28        /// <para>Initializes a new instance of the <see cref="HeapResizableDirectMemory"/>
29        → class.</para>
30        /// <para>Инициализирует новый экземпляр класса <see
31        → cref="HeapResizableDirectMemory"/>.</para>
32        /// </summary>
33        /// <param name="minimumReservedCapacity"><para>Minimum file size in
34        → bytes.</para><para>Минимальный размер файла в байтах.</para></param>
35        [MethodImpl(MethodImplOptions.AggressiveInlining)]
36        public HeapResizableDirectMemory(long minimumReservedCapacity)
37        {
38            if (minimumReservedCapacity < MinimumCapacity)
39            {
40                minimumReservedCapacity = MinimumCapacity;
41            }
42        }
43    }
44 }

```

```

38     }
39     ReservedCapacity = minimumReservedCapacity;
40     UsedCapacity = 0;
41 }
42
43 /// <summary>
44 /// <para>Initializes a new instance of the <see cref="HeapResizableDirectMemory"/>
45   ↳ class.</para>
46 /// <para>Инициализирует новый экземпляр класса <see
47   ↳ cref="HeapResizableDirectMemory"/>.</para>
48 /// </summary>
49 [MethodImpl(MethodImplOptions.AggressiveInlining)]
50 public HeapResizableDirectMemory() : this(MinimumCapacity) { }
51
52 #endregion
53
54 #region ResizableDirectMemoryBase Methods
55
56 /// <inheritdoc/>
57 /// <include file='bin\Release\netstandard2.0\Platform.Memory.xml'
58   ↳ path='doc/members/member[@name="M:Platform.Memory.ResizableDirectMemoryBase.DisposeP
59   ↳ ointer(System.IntPtr,System.Int64)"]/*' />
60 [MethodImpl(MethodImplOptions.AggressiveInlining)]
61 protected override void DisposePointer(IntPtr pointer, long usedCapacity) =>
62   ↳ Marshal.FreeHGlobal(pointer);
63
64 /// <inheritdoc/>
65 /// <include file='bin\Release\netstandard2.0\Platform.Memory.xml'
66   ↳ path='doc/members/member[@name="M:Platform.Memory.ResizableDirectMemoryBase.OnReserv
67   ↳ edCapacityChanged(System.Int64,System.Int64)"]/*' />
68 [MethodImpl(MethodImplOptions.AggressiveInlining)]
69 protected override void OnReservedCapacityChanged(long oldReservedCapacity, long
70   ↳ newReservedCapacity)
71 {
72     if (Pointer == IntPtr.Zero)
73     {
74         Pointer = Marshal.AllocHGlobal(new IntPtr(newReservedCapacity));
75         MemoryBlock.Zero((void*)Pointer, newReservedCapacity);
76     }
77     else
78     {
79         Pointer = Marshal.ReAllocHGlobal(Pointer, new IntPtr(newReservedCapacity));
80         var pointer = (byte*)Pointer + oldReservedCapacity;
81         MemoryBlock.Zero(pointer, newReservedCapacity - oldReservedCapacity);
82     }
83 }
84
85 #endregion
86 }
87 }

```

1.6 ./Platform.Memory/IArrayMemory.cs

```

1 using System.Runtime.CompilerServices;
2
3 namespace Platform.Memory
4 {
5     /// <summary>
6     /// <para>Represents a memory block interface with access via indexer.</para>
7     /// <para>Представляет интерфейс блока памяти с доступом через индексатор.</para>
8     /// </summary>
9     /// <typeparam name="TElement"><para>Element type.</para><para>Тип
10     ↳ элемента.</para></typeparam>
11     public interface IArrayMemory<TElement> : IMemory
12     {
13         /// <summary>
14         /// <para>Gets or sets the element at the specified index.</para>
15         /// <para>Возвращает или устанавливает элемент по указанному индексу.</para>
16         /// </summary>
17         /// <param name="index"><para>The index of the element to get or set.</para><para>Индекс
18         ↳ элемента, который нужно получить или установить.</para></param>
19         TElement this[long index]
20         {
21             [MethodImpl(MethodImplOptions.AggressiveInlining)]
22             get;
23             [MethodImpl(MethodImplOptions.AggressiveInlining)]
24             set;
25         }
26     }
27 }

```

```
25 }
```

1.7 ./Platform.Memory/IDirectMemory.cs

```
1 using System;
2 using System.Runtime.CompilerServices;
3
4 namespace Platform.Memory
5 {
6     /// <summary>
7     /// <para>Represents a memory block interface with direct access (via unmanaged
8     ///   ↳ pointers).</para>
9     /// <para>Представляет интерфейс блока памяти с прямым доступом (через неуправляемые
10    ///   ↳ указатели).</para>
11    /// </summary>
12    public interface IDirectMemory : IMemory, IDisposable
13    {
14        /// <summary>
15        /// <para>Gets the pointer to the beginning of this memory block.</para>
16        /// <para>Возвращает указатель на начало блока памяти.</para>
17        /// </summary>
18        IntPtr Pointer
19        {
20            [MethodImpl(MethodImplOptions.AggressiveInlining)]
21            get;
22        }
23    }
24 }
```

1.8 ./Platform.Memory/IMemory.cs

```
1 using System.Runtime.CompilerServices;
2
3 namespace Platform.Memory
4 {
5     /// <summary>
6     /// <para>Represents a memory block interface with size in bytes.</para>
7     /// <para>Представляет интерфейс блока памяти с размером в байтах.</para>
8     /// </summary>
9     public interface IMemory
10    {
11        /// <summary>
12        /// <para>Gets the size in bytes of this memory block.</para>
13        /// <para>Возвращает размер блока памяти в байтах.</para>
14        /// </summary>
15        long Size
16        {
17            [MethodImpl(MethodImplOptions.AggressiveInlining)]
18            get;
19        }
20    }
21 }
```

1.9 ./Platform.Memory/IResizableDirectMemory.cs

```
1 using System.Runtime.CompilerServices;
2
3 namespace Platform.Memory
4 {
5     /// <summary>
6     /// <para>Represents a resizable memory block interface with direct access (via unmanaged
7     ///   ↳ pointers).</para>
8     /// <para>Представляет интерфейс блока памяти с изменяемым размером и прямым доступом (через
9     ///   ↳ неуправляемые указатели).</para>
10    /// </summary>
11    public interface IResizableDirectMemory : IDirectMemory
12    {
13        /// <summary>
14        /// <para>Gets or sets the reserved capacity in bytes of this memory block.</para>
15        /// <para>Возвращает или устанавливает зарезервированный размер блока памяти в
16        ///   ↳ байтах.</para>
17        /// </summary>
18        /// <remarks>
19        /// <para>
20        ///   If less than zero the value is replaced with zero.
21        ///   Cannot be less than the used capacity of this memory block.
22        /// </para>
23        /// <para>
24        ///   Если меньше нуля, значение заменяется на ноль.
25        ///   Не может быть меньше используемой емкости блока памяти.
26        /// </para>
27        long Capacity
28        {
29            [MethodImpl(MethodImplOptions.AggressiveInlining)]
30            get;
31            set;
32        }
33    }
34 }
```



```

24     /// </remarks>
25     long ReservedCapacity
26     {
27         [MethodImpl(MethodImplOptions.AggressiveInlining)]
28         get;
29         [MethodImpl(MethodImplOptions.AggressiveInlining)]
30         set;
31     }
32
33     /// <summary>
34     /// <para>Gets or sets the used capacity in bytes of this memory block.</para>
35     /// <para>Возвращает или устанавливает используемый размер в блоке памяти (в
36     ↪ байтах).</para>
37     /// </summary>
38     /// <remarks>
39     /// <para>
40     /// If less then zero the value is replaced with zero.
41     /// Cannot be greater than the reserved capacity of this memory block.
42     /// </para>
43     /// <para>
44     /// It is recommended to reduce the reserved capacity of the memory block to the used
45     ↪ capacity (specified in this property) after the completion of the use of the memory
46     ↪ block.
47     /// </para>
48     /// <para>
49     /// Если меньше нуля, значение заменяется на ноль.
50     /// Не может быть больше, чем зарезервированная емкость этого блока памяти.
51     /// </para>
52     /// <para>
53     /// Рекомендуется уменьшать фактический размер блока памяти до используемого размера
54     ↪ (указанного в этом свойстве) после завершения использования блока памяти.
55     /// </para>
56     /// </remarks>
57     long UsedCapacity
58     {
59         [MethodImpl(MethodImplOptions.AggressiveInlining)]
60         get;
61         [MethodImpl(MethodImplOptions.AggressiveInlining)]
62         set;
63     }
64 }

```

1.10 ./Platform.Memory/ResizableDirectMemoryBase.cs

```

1  using System;
2  using System.Threading;
3  using System.Runtime.CompilerServices;
4  using Platform.Exceptions;
5  using Platform.Disposables;
6  using Platform.Ranges;
7
8  namespace Platform.Memory
9  {
10     /// <summary>
11     /// <para>Provides a base implementation for the resizable memory block with direct access
12     ↪ (via unmanaged pointers).</para>
13     /// <para>Предоставляет базовую реализацию для блока памяти с изменяемым размером и прямым
14     ↪ доступом (через неуправляемые указатели).</para>
15     /// </summary>
16     public abstract class ResizableDirectMemoryBase : DisposableBase, IResizableDirectMemory
17     {
18         #region Constants
19
20         /// <summary>
21         /// <para>Gets minimum capacity in bytes.</para>
22         /// <para>Возвращает минимальную емкость в байтах.</para>
23         /// </summary>
24         public static readonly long MinimumCapacity = 4096;
25
26         #endregion
27
28         #region Fields
29
30         private IntPtr _pointer;
31         private long _reservedCapacity;
32         private long _usedCapacity;
33
34         #endregion
35
36         #region Properties

```

```

35
36 /// <inheritdoc/>
37 /// <include file='bin\Release\netstandard2.0\Platform.Memory.xml'
    ↳ path='doc/members/member[@name="P:Platform.Memory.IMemory.Size"]/*' />
38 /// <exception cref="ObjectDisposedException"><para>The memory block is
    ↳ disposed.</para><para>Блок памяти уже высвобожден.</para></exception>
39 public long Size
40 {
41     [MethodImpl(MethodImplOptions.AggressiveInlining)]
42     get
43     {
44         Ensure.Always.NotDisposed(this);
45         return UsedCapacity;
46     }
47 }
48
49 /// <inheritdoc/>
50 /// <include file='bin\Release\netstandard2.0\Platform.Memory.xml'
    ↳ path='doc/members/member[@name="P:Platform.Memory.IDirectMemory.Pointer"]/*' />
51 /// <exception cref="ObjectDisposedException"><para>The memory block is
    ↳ disposed.</para><para>Блок памяти уже высвобожден.</para></exception>
52 public IntPtr Pointer
53 {
54     [MethodImpl(MethodImplOptions.AggressiveInlining)]
55     get
56     {
57         Ensure.Always.NotDisposed(this);
58         return _pointer;
59     }
60     [MethodImpl(MethodImplOptions.AggressiveInlining)]
61     protected set
62     {
63         Ensure.Always.NotDisposed(this);
64         _pointer = value;
65     }
66 }
67
68 /// <inheritdoc/>
69 /// <include file='bin\Release\netstandard2.0\Platform.Memory.xml' path='doc/members/mem
    ↳ ber[@name="P:Platform.Memory.IResizableDirectMemory.ReservedCapacity"]/*' />
70 /// <exception cref="ObjectDisposedException"><para>The memory block is
    ↳ disposed.</para><para>Блок памяти уже высвобожден.</para></exception>
71 /// <exception cref="ArgumentOutOfRangeException"><para>Attempted to set the reserved
    ↳ capacity to a value that is less than the used capacity.</para><para>Была выполнена
    ↳ попытка установить зарезервированную емкость на значение, которое меньше
    ↳ используемой емкости.</para></exception>
72 public long ReservedCapacity
73 {
74     [MethodImpl(MethodImplOptions.AggressiveInlining)]
75     get
76     {
77         Ensure.Always.NotDisposed(this);
78         return _reservedCapacity;
79     }
80     [MethodImpl(MethodImplOptions.AggressiveInlining)]
81     set
82     {
83         Ensure.Always.NotDisposed(this);
84         if (value != _reservedCapacity)
85         {
86             Ensure.Always.ArgumentInRange(value, new Range<long>(_usedCapacity,
    ↳ long.MaxValue));
87             OnReservedCapacityChanged(_reservedCapacity, value);
88             _reservedCapacity = value;
89         }
90     }
91 }
92
93 /// <inheritdoc/>
94 /// <include file='bin\Release\netstandard2.0\Platform.Memory.xml' path='doc/members/mem
    ↳ ber[@name="P:Platform.Memory.IResizableDirectMemory.UsedCapacity"]/*' />
95 /// <exception cref="ObjectDisposedException"><para>The memory block is
    ↳ disposed.</para><para>Блок памяти уже высвобожден.</para></exception>
96 /// <exception cref="ArgumentOutOfRangeException"><para>Attempted to set the used
    ↳ capacity to a value that is greater than the reserved capacity or less than
    ↳ zero.</para><para>Была выполнена попытка установить используемую емкость на
    ↳ значение, которое больше, чем зарезервированная емкость или меньше
    ↳ нуля.</para></exception>

```

```

97 public long UsedCapacity
98 {
99     [MethodImpl(MethodImplOptions.AggressiveInlining)]
100     get
101     {
102         Ensure.Always.NotDisposed(this);
103         return _usedCapacity;
104     }
105     [MethodImpl(MethodImplOptions.AggressiveInlining)]
106     set
107     {
108         Ensure.Always.NotDisposed(this);
109         if (value != _usedCapacity)
110         {
111             Ensure.Always.ArgumentInRange(value, new Range<long>(0, _reservedCapacity));
112             _usedCapacity = value;
113         }
114     }
115 }
116
117 #endregion
118
119 #region DisposableBase Properties
120
121 /// <inheritdoc>
122 protected override bool AllowMultipleDisposeCalls
123 {
124     [MethodImpl(MethodImplOptions.AggressiveInlining)]
125     get => true;
126 }
127
128 #endregion
129
130 #region Methods
131
132 /// <summary>
133 /// <para>Executed on the event of change for <see cref="ReservedCapacity"/>
134   ↳ property.</para>
135 /// <para>Выполняется в случае изменения свойства <see cref="ReservedCapacity"/>.</para>
136 /// </summary>
137 /// <param name="oldReservedCapacity"><para>The old reserved capacity of the memory
138   ↳ block in bytes.</para><para>Старая зарезервированная емкость блока памяти в
139   ↳ байтах.</para></param>
140 /// <param name="newReservedCapacity"><para>The new reserved capacity of the memory
141   ↳ block in bytes.</para><para>Новая зарезервированная емкость блока памяти в
142   ↳ байтах.</para></param>
143 [MethodImpl(MethodImplOptions.AggressiveInlining)]
144 protected abstract void OnReservedCapacityChanged(long oldReservedCapacity, long
145   ↳ newReservedCapacity);
146
147 /// <summary>
148 /// <para>Executed when it is time to dispose <see cref="Pointer"/>.</para>
149 /// <para>Выполняется, когда пришло время высвободить <see cref="Pointer"/>.</para>
150 /// </summary>
151 /// <param name="pointer"><para>The pointer to a memory block.</para><para>Указатель на
152   ↳ блок памяти.</para></param>
153 /// <param name="usedCapacity"><para>The used capacity of the memory block in
154   ↳ bytes.</para><para>Используемая емкость блока памяти в байтах.</para></param>
155 [MethodImpl(MethodImplOptions.AggressiveInlining)]
156 protected abstract void DisposePointer(IntPtr pointer, long usedCapacity);
157
158 #endregion
159
160 #region DisposableBase Methods
161
162 /// <inheritdoc>
163 [MethodImpl(MethodImplOptions.AggressiveInlining)]
164 protected override void Dispose(bool manual, bool wasDisposed)
165 {
166     if (!wasDisposed)
167     {
168         var pointer = Interlocked.Exchange(ref _pointer, IntPtr.Zero);
169         if (pointer != IntPtr.Zero)
170         {
171             DisposePointer(pointer, _usedCapacity);
172         }
173     }
174 }
175
176 }
177

```

```
#endregion
```

```
168 }  
169 }  
170 }
```

1.11 ./Platform.Memory/TemporaryFileMappedResizableDirectMemory.cs

```
1 using System.IO;  
2 using System.Runtime.CompilerServices;  
3  
4 namespace Platform.Memory  
5 {  
6     /// <summary>  
7     /// <para>Represents a memory block stored as a temporary file on disk.</para>  
8     /// <para>Представляет блок памяти, хранящийся в виде временного файла на диске.</para>  
9     /// </summary>  
10    public class TemporaryFileMappedResizableDirectMemory : FileMappedResizableDirectMemory  
11    {  
12        #region DisposableBase Properties  
13  
14        /// <inheritdoc/>  
15        protected override string ObjectName  
16        {  
17            [MethodImpl(MethodImplOptions.AggressiveInlining)]  
18            get => $"Temporary file stored memory block at '{Path}' path.";  
19        }  
20  
21        #endregion  
22  
23        #region Constructors  
24  
25        /// <summary>  
26        /// <para>Initializes a new instance of the <see  
27        ↪ cref="TemporaryFileMappedResizableDirectMemory"/> class.</para>  
28        /// <para>Инициализирует новый экземпляр класса <see  
29        ↪ cref="TemporaryFileMappedResizableDirectMemory"/>.</para>  
30        /// </summary>  
31        /// <param name="minimumReservedCapacity"><para>Minimum file size in  
32        ↪ bytes.</para><para>Минимальный размер файла в байтах.</para></param>  
33        [MethodImpl(MethodImplOptions.AggressiveInlining)]  
34        public TemporaryFileMappedResizableDirectMemory(long minimumReservedCapacity) :  
35        ↪ base(System.IO.Path.GetTempFileName(), minimumReservedCapacity) { }  
36  
37        /// <summary>  
38        /// <para>Initializes a new instance of the <see  
39        ↪ cref="TemporaryFileMappedResizableDirectMemory"/> class.</para>  
40        /// <para>Инициализирует новый экземпляр класса <see  
41        ↪ cref="TemporaryFileMappedResizableDirectMemory"/>.</para>  
42        /// </summary>  
43        [MethodImpl(MethodImplOptions.AggressiveInlining)]  
44        public TemporaryFileMappedResizableDirectMemory() : this(MinimumCapacity) { }  
45  
46        #endregion  
47  
48        #region DisposableBase Methods  
49  
50        /// <inheritdoc/>  
51        [MethodImpl(MethodImplOptions.AggressiveInlining)]  
52        protected override void Dispose(bool manual, bool wasDisposed)  
53        {  
54            base.Dispose(manual, wasDisposed);  
55            if (!wasDisposed)  
56            {  
57                File.Delete(Path);  
58            }  
59        }  
60  
61        #endregion  
62    }  
63 }
```

1.12 ./Platform.Memory.Tests/HeapResizableDirectMemoryTests.cs

```
1 using Xunit;  
2  
3 namespace Platform.Memory.Tests  
4 {  
5     public unsafe class HeapResizableDirectMemoryTests  
6     {  
7         [Fact]  
8         public void CorrectMemoryReallocationTest()  
9         {
```

```
10     using var heapMemory = new HeapResizableDirectMemory();
11     var value1 = GetLastByte(heapMemory);
12     heapMemory.ReservedCapacity *= 2;
13     var value2 = GetLastByte(heapMemory);
14     Assert.Equal(value1, value2);
15     Assert.Equal(0, value1);
16 }
17
18 private static byte GetLastByte(HeapResizableDirectMemory heapMemory)
19 {
20     var pointer1 = (void*)heapMemory.Pointer;
21     return *((byte*)pointer1 + heapMemory.ReservedCapacity - 1);
22 }
23 }
24 }
```

Index

- ./Platform.Memory.Tests/HeapResizableDirectMemoryTests.cs, 12
- ./Platform.Memory/ArrayMemory.cs, 1
- ./Platform.Memory/DirectMemoryAsArrayMemoryAdapter.cs, 1
- ./Platform.Memory/FileArrayMemory.cs, 3
- ./Platform.Memory/FileMappedResizableDirectMemory.cs, 4
- ./Platform.Memory/HeapResizableDirectMemory.cs, 6
- ./Platform.Memory/IArrayMemory.cs, 7
- ./Platform.Memory/IDirectMemory.cs, 8
- ./Platform.Memory/IMemory.cs, 8
- ./Platform.Memory/IResizableDirectMemory.cs, 8
- ./Platform.Memory/ResizableDirectMemoryBase.cs, 9
- ./Platform.Memory/TemporaryFileMappedResizableDirectMemory.cs, 12