1.1 ./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;

#pragma warning disable CS1591 // Missing XML comment for publicly visible type or member

namespace Platform.RegularExpressions.Transformer.CSharpToCpp
{
    /// <summary>
    /// <para>
    /// Represents the sharp to cpp transformer.
    /// </para>
    /// <para></para>
    /// </summary>
    /// <seealso cref="TextTransformer"/>
    public class CSharpToCppTransformer : TextTransformer
    {
        /// <summary>
        /// <para>
        /// The to list.
        /// </para>
        /// <para></para>
        /// </summary>
        public static readonly IList<ISubstitutionRule> FirstStage = new List<SubstitutionRule>
        {
            // // ...
            //
            (new Regex(@"(\r?\n)?[ \t]+//+.+"), "", 0),
            // #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
            //
            (new Regex(@"^\s*?\#pragma[\sa-zA-Z0-9]+$"), "", 0),
            // {\n\n\n
            // {
            (new Regex(@"{\s+[\r\n]+"), "{" + Environment.NewLine, 0),
            // Platform.Collections.Methods.Lists
            // Platform::Collections::Methods::Lists
            (new Regex(@"(namespace[^\r\n]+?)\.([^\r\n]+?)"), "$1::$2", 20),
            // nameof(numbers)
            // "numbers"
            (new Regex(@"(?<before>\W)nameof\(([^)\n]+\.)?(?<name>[a-zA-Z0-9_]+)(<[^)\n]+>)?\)"), "${before}\"${name}\"", 0),
            // Insert markers
            // EqualityComparer<T> _equalityComparer = EqualityComparer<T>.Default;
            // EqualityComparer<T> _equalityComparer = EqualityComparer<T>.Default;/*~_comparer~*/
            (new Regex(@"(?<declaration>EqualityComparer<(?<type>[^>\n]+)> (?<comparer>[a-zA-Z0-9_]+) = EqualityComparer<\k<type>>\.Default;)"), "${declaration}/*~${comparer}~*/", 0),
            // /*~_equalityComparer~*/..._equalityComparer.Equals(Minimum, value)
            // /*~_equalityComparer~*/...Minimum == value
            (new Regex(@"(?<before>/\*~(?<comparer>[a-zA-Z0-9_]+)~\*/(.|\n)+\W)\k<comparer>\.Equals\((?<left>[^,\n]+), (?<right>[^)\n]+)\)"), "${before}${left} == ${right}", 50),
            // Remove markers
            // /*~_equalityComparer~*/
            //
            (new Regex(@"\r?\n[^\n]+/\*~[a-zA-Z0-9_]+~\*/"), "", 10),
            // Insert markers
            // Comparer<T> _comparer = Comparer<T>.Default;
            // Comparer<T> _comparer = Comparer<T>.Default;/*~_comparer~*/
            (new Regex(@"(?<declaration>Comparer<(?<type>[^>\n]+)> (?<comparer>[a-zA-Z0-9_]+) = Comparer<\k<type>>\.Default;)"), "${declaration}/*~${comparer}~*/", 0),
            // /*~_comparer~*/..._comparer.Compare(Minimum, value) <= 0
            // /*~_comparer~*/...Minimum <= value
            (new Regex(@"(?<before>/\*~(?<comparer>[a-zA-Z0-9_]+)~\*/(.|\n)+\W)\k<comparer>\.Compare\((?<left>[^,\n]+),
            (?<right>[^)\n]+)\)\s*(?<comparison>[<>=]=?)\s*0(?<after>\D)"),
            "${before}${left} ${comparison} ${right}${after}", 50),
            // Remove markers
            // private static readonly Comparer<T> _comparer = Comparer<T>.Default;/*~_comparer~*/
            //
```

```
63              (new Regex(@"\r?\n[^\n]+/\*~[a-zA-Z0-9_]+~\*/"), "", 10),
64              // Comparer<TArgument>.Default.Compare(maximumArgument, minimumArgument) < 0
65              // maximumArgument < minimumArgument
66              (new Regex(@"Comparer<[^>\n]+>\.Default\.Compare\(\s*(?<first>[^,)\n]+),\s*(?<second
                >[^\)\n]+)\s*\)\s*(?<comparison>[<>=]=?)\s*0(?<after>\D)"), "${first}
                ${comparison} ${second}${after}", 0),
67              // public static bool operator ==(Range<T> left, Range<T> right) =>
                left.Equals(right);
68              //
69              (new Regex(@"\r?\n[^\n]+bool operator ==\((?<type>[^\n]+) (?<left>[a-zA-Z0-9]+),
                \k<type> (?<right>[a-zA-Z0-9]+)\) =>
                (\k<left>|\k<right>)\.Equals\((\k<left>|\k<right>)\);"), "", 10),
70              // public static bool operator !=(Range<T> left, Range<T> right) => !(left == right);
71              //
72              (new Regex(@"\r?\n[^\n]+bool operator !=\((?<type>[^\n]+) (?<left>[a-zA-Z0-9]+),
                \k<type> (?<right>[a-zA-Z0-9]+)\) => !\((\k<left>|\k<right>) ==
                (\k<left>|\k<right>)\);"), "", 10),
73              // public override bool Equals(object obj) => obj is Range<T> range ? Equals(range)
                : false;
74              //
75              (new Regex(@"\r?\n[^\n]+override bool Equals\((System\.)?[Oo]bject
                (?<this>[a-zA-Z0-9]+)\) => \k<this> is [^\n]+ (?<other>[a-zA-Z0-9]+) \?
                Equals\(\k<other>\) : false;"), "", 10),
76              // out TProduct
77              // TProduct
78              (new Regex(@"(?<before>(<|, ))(in|out)
                (?<typeParameter>[a-zA-Z0-9]+)(?<after>(>|,))"),
                "${before}${typeParameter}${after}", 10),
79              // public ...
80              // public: ...
81              (new Regex(@"(?<newLineAndIndent>\r?\n?[
                \t]*)(?<before>[^\{\(\r\n]*)(?<access>private|protected|public)[ \t]+(?![^\{\(\r
                \n]*((?<=\s)|\W)(interface|class|struct)(\W)[^\{\(\r\n]*[\{\(\r\n])"),
                "${newLineAndIndent}${access}: ${before}", 0),
82              // public: static bool CollectExceptions { get; set; }
83              // public: inline static bool CollectExceptions;
84              (new Regex(@"(?<access>(private|protected|public): )(?<before>(static )?[^\r\n]+
                )(?<name>[a-zA-Z0-9]+) {[^;}]*(?<=\W)get;[^;}]*(?<=\W)set;[^;}]*}"),
                "${access}inline ${before}${name};", 0),
85              // public abstract class
86              // class
87              (new Regex(@"((public|protected|private|internal|abstract|static)
                )*(?<category>interface|class|struct)"), "${category}", 0),
88              // class GenericCollectionMethodsBase<TElement> {
89              // template <typename TElement> class GenericCollectionMethodsBase {
90              (new Regex(@"(?<before>\r?\n)(?<indent>[ \t]*)(?<type>class|struct)
                    (?<typeName>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9
                ,]+)>(?<typeDefinitionEnding>[^{]+){"), "${before}${indent}template <typename
                ...> ${type} ${typeName};" + Environment.NewLine + "${indent}template <typename
                ${typeParameters}> ${type}
                ${typeName}<${typeParameters}>${typeDefinitionEnding}{", 0),
91              // static void
                TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
                tree, TElement* root)
92              // template<typename T> static void
                TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
                tree, TElement* root)
93              (new Regex(@"static ([a-zA-Z0-9]+) ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>\(([^\)\r\n]+)\)"),
                "template <typename $3> static $1 $2($4)", 0),
94              // interface IFactory<out TProduct> {
95              // template <typename...> class IFactory;\ntemplate <typename TProduct> class
                IFactory<TProduct>
96              (new Regex(@"(?<before>\r?\n)(?<indent>[ \t]*)interface
                    (?<interface>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9
                ,]+)>(?<typeDefinitionEnding>[^{]+){"), "${before}${indent}template <typename
                ...> class ${interface};" + Environment.NewLine + "${indent}template <typename
                ${typeParameters}> class
                ${interface}<${typeParameters}>${typeDefinitionEnding}{" + Environment.NewLine +
                "    public:", 0),
97              // template <typename TObject, TProperty, TValue>
98              // template <typename TObject, typename TProperty, typename TValue>
99              (new Regex(@"(?<before>template <((, )?typename [a-zA-Z0-9]+)+,
                )(?<typeParameter>[a-zA-Z0-9]+)(?<after>(,|>))"), "${before}typename
                ${typeParameter}${after}", 10),
100             // Insert markers
```

```
101             // private: static void BuildExceptionString(this StringBuilder sb, Exception
                ↪  exception, int level)
102             // /*~extensionMethod~BuildExceptionString~*/private: static void
                ↪  BuildExceptionString(this StringBuilder sb, Exception exception, int level)
103             (new Regex(@"private: static [^\r\n]+ (?<name>[a-zA-Z0-9]+)\(this [^\)\r\n]+\)"),
                ↪  "/*~extensionMethod~${name}~*/$0", 0),
104             // Move all markers to the beginning of the file.
105             (new Regex(@"\A(?<before>[^\r\n]+\r?\n(.|\n)+)(?<marker>/\*~extensionMethod~(?<name>
                ↪  [a-zA-Z0-9]+)~\*/)"), "${marker}${before}",
                ↪  10),
106             // /*~extensionMethod~BuildExceptionString~*/...sb.BuildExceptionString(exception.In
                ↪  nerException, level +
                ↪  1);
107             // /*~extensionMethod~BuildExceptionString~*/...BuildExceptionString(sb,
                ↪  exception.InnerException, level + 1);
108             (new Regex(@"(?<before>/\*~extensionMethod~(?<name>[a-zA-Z0-9]+)~\*/(.|\n)+\W)(?<var
                ↪  iable>[_a-zA-Z0-9]+)\.\k<name>\("), "${before}${name}(${variable}, ",
                ↪  50),
109             // Remove markers
110             // /*~extensionMethod~BuildExceptionString~*/
111             //
112             (new Regex(@"/\*~extensionMethod~[a-zA-Z0-9]+~\*/"), "", 0),
113             // (this
114             // (
115             (new Regex(@"\(this "), "(", 0),
116             // private: static readonly Disposal _emptyDelegate = (manual, wasDisposed) => { };
117             // private: inline static std::function<Disposal> _emptyDelegate = [](auto manual,
                ↪  auto wasDisposed) { };
118             (new Regex(@"(?<access>(private|protected|public): )?static readonly
                    (?<type>[a-zA-Z][a-zA-Z0-9]*) (?<name>[a-zA-Z_][a-zA-Z0-9_]*) =
                ↪  \((?<firstArgument>[a-zA-Z_][a-zA-Z0-9_]*),
                ↪  (?<secondArgument>[a-zA-Z_][a-zA-Z0-9_]*)\) => {\s*};"), "${access}inline static
                ↪  std::function<${type}> ${name} = [](auto ${firstArgument}, auto
                ↪  ${secondArgument}) { };", 0),
119             // public: static readonly EnsureAlwaysExtensionRoot Always = new
                ↪  EnsureAlwaysExtensionRoot();
120             // public: inline static EnsureAlwaysExtensionRoot Always;
121             (new Regex(@"(?<access>(private|protected|public): )?static readonly
                ↪  (?<type>[a-zA-Z0-9]+(<[a-zA-Z0-9]+>)?) (?<name>[a-zA-Z0-9_]+) = new
                ↪  \k<type>\(\);"), "${access}inline static ${type} ${name};", 0),
122             // public: static readonly Range<int> SByte = new
                ↪  Range<int>(std::numeric_limits<int>::min(), std::numeric_limits<int>::max());
123             // public: inline static Range<int> SByte =
                ↪  Range<int>(std::numeric_limits<int>::min(), std::numeric_limits<int>::max());
124             (new Regex(@"(?<access>(private|protected|public): )?static readonly
                ↪  (?<type>[a-zA-Z0-9]+(<[a-zA-Z0-9]+>)?) (?<name>[a-zA-Z0-9_]+) = new
                ↪  \k<type>\((?<arguments>[^\n]+)\);"), "${access}inline static ${type} ${name} =
                ↪  ${type}(${arguments});", 0),
125             // public: static readonly string ExceptionContentsSeparator = "---";
126             // public: inline static std::string ExceptionContentsSeparator = "---";
127             (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly) string
                ↪  (?<name>[a-zA-Z0-9_]+) = ""(?<string>(\\""|[^""\r\n])+)"";"), "${access}inline
                ↪  static std::string ${name} = \"${string}\";", 0),
128             // private: const int MaxPath = 92;
129             // private: inline static const int MaxPath = 92;
130             (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly)
                ↪  (?<type>[a-zA-Z0-9]+) (?<name>[_a-zA-Z0-9]+) = (?<value>[^;\r\n]+);"),
                ↪  "${access}inline static const ${type} ${name} = ${value};", 0),
131             //  ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
                ↪  TArgument : class
132             //  ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
133             (new Regex(@"(?<before> [a-zA-Z]+\(([a-zA-Z *,]+, |))(?<type>[a-zA-Z]+)(?<after>(|
                ↪  [a-zA-Z *,]+)\))[ \r\n]+where \k<type> : class"), "${before}${type}*${after}",
                ↪  0),
134             // protected: abstract TElement GetFirst();
135             // protected: virtual TElement GetFirst() = 0;
136             (new Regex(@"(?<access>(private|protected|public): )?abstract
                ↪  (?<method>[^;\r\n]+);"), "${access}virtual ${method} = 0;", 0),
137             // TElement GetFirst();
138             // virtual TElement GetFirst() = 0;
139             (new Regex(@"(?<before>[\r\n]+[ ]+)(?<methodDeclaration>(?!return)[a-zA-Z0-9]+
                ↪  [a-zA-Z0-9]+\([^\)\r\n]*\))(?<after>;[ ]*[\r\n]+)"), "${before}virtual
                ↪  ${methodDeclaration} = 0${after}", 1),
140             // protected: readonly TreeElement[] _elements;
141             // protected: TreeElement _elements[N];
```

```
(new Regex(@"(?<access>(private|protected|public): )?readonly (?<type>[a-zA-Z<>0-9]+)([\[\]]+) (?<name>[_a-zA-Z0-9]+);"), "${access}${type} ${name}[N];", 0),
// protected: readonly TElement Zero;
// protected: TElement Zero;
(new Regex(@"(?<access>(private|protected|public): )?readonly (?<type>[a-zA-Z<>0-9]+) (?<name>[_a-zA-Z0-9]+);"), "${access}${type} ${name};", 0),
// internal
//
(new Regex(@"(\W)internal\s+"), "$1", 0),
// static void NotImplementedException(ThrowExtensionRoot root) => throw new NotImplementedException();
// static void NotImplementedException(ThrowExtensionRoot root) { return throw new NotImplementedException(); }
(new Regex(@"(^\s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static )?(override )?([a-zA-Z0-9]+
)([a-zA-Z0-9]+)\(([^\(\r\n]*)\)\s+=>\s+throw([^;\r\n]+);"), "$1$2$3$4$5$6$7$8($9) { throw$10; }", 0),
// SizeBalancedTree(int capacity) => a = b;
// SizeBalancedTree(int capacity) { a = b; }
(new Regex(@"(^\s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static )?(override )?(void )?([a-zA-Z0-9]+)\(([^\(\r\n]*)\)\s+=>\s+([^;\r\n]+);"), "$1$2$3$4$5$6$7$8($9) { $10; }", 0),
// int SizeBalancedTree(int capacity) => a;
// int SizeBalancedTree(int capacity) { return a; }
(new Regex(@"(^\s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static )?(override )?([a-zA-Z0-9]+
)([a-zA-Z0-9]+)\(([^\(\r\n]*)\)\s+=>\s+([^;\r\n]+);"), "$1$2$3$4$5$6$7$8($9) { return $10; }", 0),
// OnDispose = (manual, wasDisposed) =>
// OnDispose = [&](auto manual, auto wasDisposed)
(new Regex(@"(?<variable>[a-zA-Z_][a-zA-Z0-9_]*)(?<operator>\s*\+?=\s*)\((?<firstArgument>[a-zA-Z_][a-zA-Z0-9_]*),
(?<secondArgument>[a-zA-Z_][a-zA-Z0-9_]*)\)\s*=>"),
"${variable}${operator}[&](auto ${firstArgument}, auto ${secondArgument})", 0),
// () => Integer<TElement>.Zero,
// () { return Integer<TElement>.Zero; },
(new Regex(@"\(\)\s+=>\s+(?<expression>[^(),;\r\n]+(\(((?<parenthesis>\()|(?<-parenthesis>\))|[^();\r\n]*?)*?\))?[^(),;\r\n]*)(?<after>,|\);)"), "() { return ${expression}; }${after}", 0),
// ~DisposableBase() => Destruct();
// ~DisposableBase() { Destruct(); }
(new Regex(@"~(?<class>[a-zA-Z_][a-zA-Z0-9_]*)\(\)\s+=>\s+([^;\r\n]+?);"), "~${class}() { $1; }", 0),
// => Integer<TElement>.Zero;
// { return Integer<TElement>.Zero; }
(new Regex(@"\)\s+=>\s+([^;\r\n]+?);"), ") { return $1; }", 0),
// () { return avlTree.Count; }
// [&]()-> auto { return avlTree.Count; }
(new Regex(@"(?<before>, |\()\(\) { return (?<expression>[^;\r\n]+); }"), "${before}[&]()-> auto { return ${expression}; }", 0),
// Count => GetSizeOrZero(Root);
// Count() { return GetSizeOrZero(Root); }
(new Regex(@"(\W)([A-Z][a-zA-Z]+)\s+=>\s+([^;\r\n]+);"), "$1$2() { return $3; }", 0),
// Insert scope borders.
// interface IDisposable { ... }
// interface IDisposable {/*~start~interface~IDisposable~*/ ... /*~end~interface~IDisposable~*/}
(new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)interface[\t ]*(?<type>[a-zA-Z][a-zA-Z0-9]*(<[^<>\n]*>)?)[^{}]*{)(?<middle>(.|\n)*)(?<beforeEnd>(?<=\r?\n)\k<indent>)(?<end>})"),
"${classDeclarationBegin}/*~start~interface~${type}~*/${middle}${beforeEnd}/*~end~interface~${type}~*/${end}",
0),
// Inside the scope replace:
// /*~start~interface~IDisposable~*/ ... bool IsDisposed { get; } ... /*~end~interface~IDisposable~*/
// /*~start~interface~IDisposable~*/ ... virtual bool IsDisposed() = 0; /*~end~interface~IDisposable~*/
(new Regex(@"(?<before>(?<typeScopeStart>/\*~start~interface~(?<type>[^~\n\*]+)~\*/)(.|\n)+?)(?<propertyDeclaration>(?<access>(private|protected|public): )?(?<propertyType>[a-zA-Z_][a-zA-Z0-9_:<>]*) (?<property>[a-zA-Z_][a-zA-Z0-9_]*)
(?<blockOpen>[\n\s]*{[\n\s]*)(\[[^\n]+\][\n\s]*)?get;(?<blockClose>[\n\s]*}))(?<after>(.|\n)+?(?<typeScopeEnd>/\*~end~interface~\k<type>~\*/))"),
"${before}virtual ${propertyType} ${property}() = 0;${after}", 20),
```

```
                // Remove scope borders.
                // /*~start~interface~IDisposable~*/
                //
                (new Regex(@"/\*~[^~\*\n]+(~[^~\*\n]+)*~\*/"), "", 0),
                // public: T Object { get; }
                // public: const T Object;
                (new Regex(@"(?<before>[^\r]\r?\n[ \t]*)(?<access>(private|protected|public):
                    )?(?<type>[a-zA-Z_][a-zA-Z0-9_:<>]*)
                    (?<property>[a-zA-Z_][a-zA-Z0-9_]*)(?<blockOpen>[\n\s]*{[\n\s]*)(\[[^\n]+\][\n\s
                    ]*)?get;(?<blockClose>[\n\s]*})(?<after>[\n\s]*)"), "${before}${access}const
                    ${type} ${property};${after}", 2),
                // public: bool IsDisposed { get => _disposed > 0; }
                // public: bool IsDisposed() { return _disposed > 0; }
                (new Regex(@"(?<before>[^\r]\r?\n[ \t]*)(?<access>(private|protected|public):
                    )?(?<virtual>virtual )?bool
                    (?<property>[a-zA-Z_][a-zA-Z0-9_]*)(?<blockOpen>[\n\s]*{[\n\s]*)(\[[^\n]+\][\n\s
                    ]*)?get\s*=>\s*(?<expression>[^\n]+);(?<blockClose>[\n\s]*}[\n\s]*)"),
                    "${before}${access}${virtual}bool ${property}()${blockOpen}return
                    ${expression};${blockClose}", 2),
                // protected: virtual std::string ObjectName { get => GetType().Name; }
                // protected: virtual std::string ObjectName() { return GetType().Name; }
                (new Regex(@"(?<before>[^\r]\r?\n[ \t]*)(?<access>(private|protected|public):
                    )?(?<virtual>virtual )?(?<type>[a-zA-Z_][a-zA-Z0-9_:<>]*)
                    (?<property>[a-zA-Z_][a-zA-Z0-9_]*)(?<blockOpen>[\n\s]*{[\n\s]*)(\[[^\n]+\][\n\s
                    ]*)?get\s*=>\s*(?<expression>[^\n]+);(?<blockClose>[\n\s]*}[\n\s]*)"),
                    "${before}${access}${virtual}${type} ${property}()${blockOpen}return
                    ${expression};${blockClose}", 2),
                // ArgumentInRange(string message) { string messageBuilder() { return message; }
                // ArgumentInRange(string message) { auto messageBuilder = [&]() -> string { return
                    message; };
                (new Regex(@"(?<before>\W[_a-zA-Z0-9]+\([^\)\n]*\)[\s\n]*{[\s\n]*([^{}]|\n)*?(\r?\n)
                    ?[ \t]*)(?<returnType>[_a-zA-Z0-9*:]+[_a-zA-Z0-9*: ]*)
                    (?<methodName>[_a-zA-Z0-9]+)\((?<arguments>[^\)\n]*)\)\s*{(?<body>(""[^""\n]+""|
                    [^}]|\n)+?)}"), "${before}auto ${methodName} = [&]() -> ${returnType}
                    {${body}};", 10),
                // Func<TElement> treeCount
                // std::function<TElement()> treeCount
                (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", 0),
                // Action<TElement> free
                // std::function<void(TElement)> free
                (new Regex(@"Action(<(?<typeParameters>[a-zA-Z0-9]+(,
                    ([a-zA-Z0-9]+))*)>)?(?<after>>| (?<variable>[a-zA-Z0-9]+))"),
                    "std::function<void(${typeParameters})>${after}", 0),
                // Predicate<TArgument> predicate
                // std::function<bool(TArgument)> predicate
                (new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<bool($1)>
                    $2", 0),
                // var
                // auto
                (new Regex(@"(\W)var(\W)"), "$1auto$2", 0),
                // unchecked
                //
                (new Regex(@"[\r\n]{2}\s*?unchecked\s*?$"), "", 0),
                // throw new
                // throw
                (new Regex(@"(\W)throw new(\W)"), "$1throw$2", 0),
                // void RaiseExceptionIgnoredEvent(Exception exception)
                // void RaiseExceptionIgnoredEvent(const std::exception& exception)
                (new Regex(@"(\(|, )(System\.Exception|Exception)( |\))"), "$1const
                    std::exception&$3", 0),
                // EventHandler<Exception>
                // EventHandler<std::exception>
                (new Regex(@"(\W)(System\.Exception|Exception)(\W)"), "$1std::exception$3", 0),
                // override void PrintNode(TElement node, StringBuilder sb, int level)
                // void PrintNode(TElement node, StringBuilder sb, int level) override
                (new Regex(@"override ([a-zA-Z0-9 \*\+]+)(\([^\)\r\n]+?\))"), "$1$2 override", 0),
                // return (range.Minimum, range.Maximum)
                // return {range.Minimum, range.Maximum}
                (new Regex(@"(?<before>return\s*)\((?<values>[^\)\n]+)\)(?!\()(?<after>\W)"),
                    "${before}{${values}}${after}", 0),
                // string
                // std::string
                (new Regex(@"(?<before>\W)(?<!::)string(?<after>\W)"),
                    "${before}std::string${after}", 0),
                // System.ValueTuple
                // std::tuple
```

```
235             (new Regex(@"(?<before>\W)(System\.)?ValueTuple(?!\s*=|\()(?<after>\W)"),
                ↪  "${before}std::tuple${after}", 0),
236             // sbyte
237             // std::int8_t
238             (new Regex(@"(?<before>\W)((System\.)?SB|sb)yte(?!\s*=|\()(?<after>\W)"),
                ↪  "${before}std::int8_t${after}", 0),
239             // short
240             // std::int16_t
241             (new Regex(@"(?<before>\W)((System\.)?Int16|short)(?!\s*=|\()(?<after>\W)"),
                ↪  "${before}std::int16_t${after}", 0),
242             // int
243             // std::int32_t
244             (new Regex(@"(?<before>\W)((System\.)?I|i)nt(32)?(?!\s*=|\()(?<after>\W)"),
                ↪  "${before}std::int32_t${after}", 0),
245             // long
246             // std::int64_t
247             (new Regex(@"(?<before>\W)((System\.)?Int64|long)(?!\s*=|\()(?<after>\W)"),
                ↪  "${before}std::int64_t${after}", 0),
248             // byte
249             // std::uint8_t
250             (new Regex(@"(?<before>\W)((System\.)?Byte|byte)(?!\s*=|\()(?<after>\W)"),
                ↪  "${before}std::uint8_t${after}", 0),
251             // ushort
252             // std::uint16_t
253             (new Regex(@"(?<before>\W)((System\.)?UInt16|ushort)(?!\s*=|\()(?<after>\W)"),
                ↪  "${before}std::uint16_t${after}", 0),
254             // uint
255             // std::uint32_t
256             (new Regex(@"(?<before>\W)((System\.)?UI|ui)nt(32)?(?!\s*=|\()(?<after>\W)"),
                ↪  "${before}std::uint32_t${after}", 0),
257             // ulong
258             // std::uint64_t
259             (new Regex(@"(?<before>\W)((System\.)?UInt64|ulong)(?!\s*=|\()(?<after>\W)"),
                ↪  "${before}std::uint64_t${after}", 0),
260             // char*[] args
261             // char* args[]
262             (new Regex(@"([_a-zA-Z0-9:\*]?)\[\] ([a-zA-Z0-9]+)"), "$1 $2[]", 0),
263             // float.MinValue
264             // std::numeric_limits<float>::lowest()
265             (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MinValue(?<after>\W
                ↪  )"), "${before}std::numeric_limits<${type}>::lowest()${after}",
                ↪  0),
266             // double.MaxValue
267             // std::numeric_limits<float>::max()
268             (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MaxValue(?<after>\W
                ↪  )"), "${before}std::numeric_limits<${type}>::max()${after}",
                ↪  0),
269             // using Platform.Numbers;
270             //
271             (new Regex(@"([\r\n]{2}|^)\s*?using [\.a-zA-Z0-9]+;\s*?$"), "", 0),
272             // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
273             // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
274             (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(<[a-zA-Z0-9 ,]+>)? : ([a-zA-Z0-9]+)"),
                ↪  "$1 $2$3 : public $4", 0),
275             // System.IDisposable
276             // System::IDisposable
277             (new Regex(@"(?<before>System(::[a-zA-Z_]\w*)*)\.(?<after>[a-zA-Z_]\w*)"),
                ↪  "${before}::${after}", 20),
278             // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
279             // class IProperty : public ISetter<TValue, TObject>, public IProvider<TValue,
                ↪  TObject>
280             (new Regex(@"(?<before>(interface|struct|class) [a-zA-Z_]\w* : ((public
                ↪  [a-zA-Z_][\w:]*(<[a-zA-Z0-9 ,]+>)?,
                ↪  )+)?)(?<inheritedType>(?!public)[a-zA-Z_][\w:]*(<[a-zA-Z0-9 ,]+>)?)(?<after>(,
                ↪  [a-zA-Z_][\w:]*(?!>)|[ \r\n]+))"), "${before}public ${inheritedType}${after}",
                ↪  10),
281             // interface IDisposable {
282             // class IDisposable { public:
283             (new Regex(@"(?<before>\r?\n)(?<indent>[ \t]*)interface
                ↪  (?<interface>[a-zA-Z_]\w*)(?<typeDefinitionEnding>[^{]+){"),
                ↪  "${before}${indent}class ${interface}${typeDefinitionEnding}{" +
                ↪  Environment.NewLine + "    public:", 0),
284             // struct TreeElement { }
285             // struct TreeElement { };
286             (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([^;])"), "$1
                ↪  $2$3{$4};$5", 0),
```

```
287             // class Program { }
288             // class Program { };
289             (new Regex(@"(?<type>struct|class)
                ↪  (?<name>[a-zA-Z0-9]+[^\r\n]*)(?<beforeBody>[\r\n]+(?<indentLevel>[\t
                ↪  ]*)?)\{(?<body>[\S\s]+?[\r\n]+\k<indentLevel>)\}(?<afterBody>[^;]|$)"), "${type}
                ↪  ${name}${beforeBody}{${body}};${afterBody}", 0),
290             // Insert scope borders.
291             // ref TElement root
292             // ~!root!~ref TElement root
293             (new Regex(@"(?<definition>(?<= |\()(ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!ref))
                ↪  (?<variable>[a-zA-Z0-9]+)(?=\)|, | =))"), "~!${variable}!~${definition}", 0),
294             // Inside the scope of ~!root!~ replace:
295             // root
296             // *root
297             (new Regex(@"(?<definition>~!(?<pointer>[a-zA-Z0-9]+)!~ref [a-zA-Z0-9]+
                ↪  \k<pointer>(?=\)|, | =))(?<before>((?<!~!\k<pointer>!~)(.|\n))*?)(?<prefix>(\W
                ↪  |\())\k<pointer>(?<suffix>( |\)|;|,))"),
                ↪  "${definition}${before}${prefix}*${pointer}${suffix}", 70),
298             // Remove scope borders.
299             // ~!root!~
300             //
301             (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", 5),
302             // ref auto root = ref
303             // ref auto root =
304             (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)"), "$1* $2 =$3", 0),
305             // *root = ref left;
306             // root = left;
307             (new Regex(@"\*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", 0),
308             // (ref left)
309             // (left)
310             (new Regex(@"\(ref ([a-zA-Z0-9]+)(\)|\(|,)"), "($1$2", 0),
311             //  ref TElement
312             //  TElement*
313             (new Regex(@"( |\()ref ([a-zA-Z0-9]+) "), "$1$2* ", 0),
314             // ref sizeBalancedTree.Root
315             // &sizeBalancedTree->Root
316             (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)"), "&$1->$2", 0),
317             // ref GetElement(node).Right
318             // &GetElement(node)->Right
319             (new Regex(@"ref ([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"),
                ↪  "&$1($2)->$3", 0),
320             // GetElement(node).Right
321             // GetElement(node)->Right
322             (new Regex(@"([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"), "$1($2)->$3", 0),
323             // [Fact]\npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
324             // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
325             (new Regex(@"\[Fact\][\s\n]+(public: )?(static )?void ([a-zA-Z0-9]+)\(\)"), "public:
                ↪  TEST_METHOD($3)", 0),
326             // class TreesTests
327             // TEST_CLASS(TreesTests)
328             (new Regex(@"class ([a-zA-Z0-9]+Tests)"), "TEST_CLASS($1)", 0),
329             // Assert.Equal
330             // Assert::AreEqual
331             (new Regex(@"(?<type>Assert)\.(?<method>(Not)?Equal)"), "${type}::Are${method}", 0),
332             // Assert.Throws
333             // Assert::ExpectException
334             (new Regex(@"(Assert)\.Throws"), "$1::ExpectException", 0),
335             // Assert.True
336             // Assert::IsTrue
337             (new Regex(@"(Assert)\.(True|False)"), "$1::Is$2", 0),
338             // $"Argument {argumentName} is null."
339             // std::string("Argument
                ↪  ").append(Platform::Converters::To<std::string>(argumentName)).append(" is
                ↪  null.")
340             (new Regex(@"\$""(?<left>(\\""|[^""\r\n])*){(?<expression>[_a-zA-Z0-9]+)}(?<right>(\
                ↪  \""|[^""\r\n])*)"""),
                ↪  "std::string($\"${left}\").append(Platform::Converters::To<std::string>(${expres
                ↪  sion})).append(\"${right}\")",
                ↪  10),
341             // $"
342             // "
343             (new Regex(@"\$"""), "\"", 0),
344             // std::string(std::string("[").append(Platform::Converters::To<std::string>(Minimum
                ↪  )).append(",
                ↪  ")).append(Platform::Converters::To<std::string>(Maximum)).append("]")
345             // std::string("[").append(Platform::Converters::To<std::string>(Minimum)).append(",
                ↪  ").append(Platform::Converters::To<std::string>(Maximum)).append("]")
```

```
(new Regex(@"std::string\((?<begin>std::string\(""(\\""|[^""])*""\)(\.append\((Platform::Converters::To<std::string>\([^)\n]+\)|[^)\n]+)\))+)\)\.append"), "${begin}.append", 10),
// Console.WriteLine("...")
// printf("...\n")
(new Regex(@"Console\.WriteLine\(""([^""\r\n]+)""\)"), "printf(\"$1\\n\")", 0),
// TElement Root;
// TElement Root = 0;
(new Regex(@"(?<before>\r?\n[\t ]+)(?<access>(private|protected|public)(: )?)?(?<type>[a-zA-Z0-9:_]+(?<!return)) (?<name>[_a-zA-Z0-9]+);"), "${before}${access}${type} ${name} = 0;", 0),
// TreeElement _elements[N];
// TreeElement _elements[N] = { {0} };
(new Regex(@"(\r?\n[\t ]+)(private|protected|public)?(: )?([a-zA-Z0-9]+) ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9]+)\];"), "$1$2$3$4 $5[$6] = { {0} };", 0),
// auto path = new TElement[MaxPath];
// TElement path[MaxPath] = { {0} };
(new Regex(@"(\r?\n[\t ]+)[a-zA-Z0-9]+ ([a-zA-Z0-9]+) = new ([a-zA-Z0-9]+)\[([_a-zA-Z0-9]+)\];"), "$1$3 $2[$4] = { {0} };", 0),
// bool Equals(Range<T> other) { ... }
// bool operator ==(const Key &other) const { ... }
(new Regex(@"(?<before>\r?\n[^\n]+bool )Equals\((?<type>[^\n{]+) (?<variable>[a-zA-Z0-9]+)\)(?<after>(\s|\n)*{)"), "${before}operator ==(const ${type} &${variable}) const${after}", 0),
// Insert scope borders.
// class Range { ... public: override std::string ToString() { return ...; }
// class Range {/*~Range<T>~*/ ... public: override std::string ToString() { return ...; }
(new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)template <typename (?<typeParameter>[^<>\n]+)> (struct|class) (?<type>[a-zA-Z0-9]+<\k<typeParameter>>)(\s*:\s*[^{\n]+)?[\t ]*(\r?\n)?[\t ]*{)(?<middle>((?!class|struct).|\n)+?)(?<toStringDeclaration>(?<access>(private|protected|public): )override std::string ToString\(\))"), "${classDeclarationBegin}/*~${type}~*/${middle}${toStringDeclaration}", 0),
// Inside the scope of ~!Range!~ replace:
// public: override std::string ToString() { return ...; }
// public: operator std::string() const { return ...; }\n\npublic: friend std::ostream & operator <<(std::ostream &out, const A &obj) { return out << (std::string)obj; }
(new Regex(@"(?<scope>/\*~(?<type>[_a-zA-Z0-9<>:]+)~\*/)(?<separator>.|\n)(?<before>((?<!/\*~\k<type>~\*/)(.|\n))*?)(?<toStringDeclaration>\r?\n(?<indent>[\t]*)(?<access>(private|protected|public): )override std::string ToString\(\) (?<toStringMethodBody>{[^}\n]+}))"), "${scope}${separator}${before}" + Environment.NewLine + "${indent}${access}operator std::string() const ${toStringMethodBody}" + Environment.NewLine + Environment.NewLine + "${indent}${access}friend std::ostream & operator <<(std::ostream &out, const ${type} &obj) { return out << (std::string)obj; }", 0),
// Remove scope borders.
// /*~Range~*/
//
(new Regex(@"/\*~[_a-zA-Z0-9<>:]+~\*/"), "", 0),
// private: inline static ConcurrentBag<std::exception> _exceptionsBag;
// private: inline static std::mutex _exceptionsBag_mutex; \n\n private: inline static std::vector<std::exception> _exceptionsBag;
(new Regex(@"(?<begin>\r?\n?(?<indent>[ \t]+))(?<access>(private|protected|public): )?inline static ConcurrentBag<(?<argumentType>[^;\r\n]+)> (?<name>[_a-zA-Z0-9]+);"), "${begin}private: inline static std::mutex ${name}_mutex;" + Environment.NewLine + Environment.NewLine + "${indent}${access}inline static std::vector<${argumentType}> ${name};", 0),
// public: static IReadOnlyCollection<std::exception> GetCollectedExceptions() { return _exceptionsBag; }
// public: static std::vector<std::exception> GetCollectedExceptions() { return std::vector<std::exception>(_exceptionsBag); }
(new Regex(@"(?<access>(private|protected|public): )?static IReadOnlyCollection<(?<argumentType>[^;\r\n]+)> (?<methodName>[_a-zA-Z0-9]+)\(\) { return (?<fieldName>[_a-zA-Z0-9]+); }"), "${access}static std::vector<${argumentType}> ${methodName}() { return std::vector<${argumentType}>(${fieldName}); }", 0),
// public: static event EventHandler<std::exception> ExceptionIgnored = OnExceptionIgnored; ... };
// ... public: static inline Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)> ExceptionIgnored = OnExceptionIgnored; };
```

```
382             (new Regex(@"(?<begin>\r?\n(\r?\n)?(?<halfIndent>[
                    \t]+)\k<halfIndent>)(?<access>(private|protected|public): )?static event
                ↪  EventHandler<(?<argumentType>[^;\r\n]+)> (?<name>[_a-zA-Z0-9]+) = (?<defaultDele
                ↪  gate>[_a-zA-Z0-9]+);(?<middle>(.|\n)+?)(?<end>\r?\n\k<halfIndent>};)"),
                ↪  "${middle}" + Environment.NewLine + Environment.NewLine +
                ↪  "${halfIndent}${halfIndent}${access}static inline
                ↪  Platform::Delegates::MulticastDelegate<void(void*, const ${argumentType}&)>
                ↪  ${name} = ${defaultDelegate};${end}", 0),
383             // public: event Disposal OnDispose;
384             // public: Platform::Delegates::MulticastDelegate<Disposal> OnDispose;
385             (new Regex(@"(?<begin>(?<access>(private|protected|public): )?(static )?)event
                ↪  (?<type>[a-zA-Z][:_a-zA-Z0-9]+) (?<name>[a-zA-Z][_a-zA-Z0-9]+);"),
                ↪  "${begin}Platform::Delegates::MulticastDelegate<${type}> ${name};", 0),
386             // Insert scope borders.
387             // class IgnoredExceptions { ... private: inline static std::vector<std::exception>
                ↪  _exceptionsBag;
388             // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: inline static
                ↪  std::vector<std::exception> _exceptionsBag;
389             (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n]+\r\n[\t
                    ]*{)(?<middle>((?!class).|\n)+?)(?<vectorFieldDeclaration>(?<access>(private|pro
                ↪  tected|public): )inline static std::vector<(?<argumentType>[^;\r\n]+)>
                ↪  (?<fieldName>[_a-zA-Z0-9]+);)"),
                ↪  "${classDeclarationBegin}/*~${fieldName}~*/${middle}${vectorFieldDeclaration}",
                ↪  0),
390             // Inside the scope of ~!_exceptionsBag!~ replace:
391             // _exceptionsBag.Add(exception);
392             // _exceptionsBag.push_back(exception);
393             (new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<befor
                ↪  e>((?<!/\*~\k<fieldName>~\*/)(.|\n))*?)\k<fieldName>\.Add"),
                ↪  "${scope}${separator}${before}${fieldName}.push_back", 10),
394             // Remove scope borders.
395             // /*~_exceptionsBag~*/
396             //
397             (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
398             // Insert scope borders.
399             // class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
400             // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: static std::mutex
                ↪  _exceptionsBag_mutex;
401             (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n]+\r\n[\t
                ↪  ]*{)(?<middle>((?!class).|\n)+?)(?<mutexDeclaration>private: inline static
                ↪  std::mutex (?<fieldName>[_a-zA-Z0-9]+)_mutex;)"),
                ↪  "${classDeclarationBegin}/*~${fieldName}~*/${middle}${mutexDeclaration}", 0),
402             // Inside the scope of ~!_exceptionsBag!~ replace:
403             // return std::vector<std::exception>(_exceptionsBag);
404             // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
                ↪  std::vector<std::exception>(_exceptionsBag);
405             (new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<befor
                ↪  e>((?<!/\*~\k<fieldName>~\*/)(.|\n))*?){(?<after>((?!lock_guard)[^{};\r\n])*\k<f
                ↪  ieldName>[^;}\r\n]*;)"), "${scope}${separator}${before}{
                ↪  std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", 10),
406             // Inside the scope of ~!_exceptionsBag!~ replace:
407             // _exceptionsBag.Add(exception);
408             // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
                ↪  _exceptionsBag.Add(exception);
409             (new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<befor
                ↪  e>((?<!/\*~\k<fieldName>~\*/)(.|\n))*?){(?<after>((?!lock_guard)([^{};]|\n))*?\r
                ↪  ?\n(?<indent>[ \t]*)\k<fieldName>[^;}\r\n]*;)"),
                ↪  "${scope}${separator}${before}{" + Environment.NewLine +
                ↪  "${indent}std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", 10),
410             // Remove scope borders.
411             // /*~_exceptionsBag~*/
412             //
413             (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
414             // Insert scope borders.
415             // class IgnoredExceptions { ... public: static inline
                ↪  Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
                ↪  ExceptionIgnored = OnExceptionIgnored;
416             // class IgnoredExceptions {/*~ExceptionIgnored~*/ ... public: static inline
                ↪  Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
                ↪  ExceptionIgnored = OnExceptionIgnored;
417             (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n]+\r\n[\t
                    ]*{)(?<middle>((?!class).|\n)+?)(?<eventDeclaration>(?<access>(private|protected
                ↪  |public): )static inline
                ↪  Platform::Delegates::MulticastDelegate<(?<argumentType>[^;\r\n]+)>
                ↪  (?<name>[_a-zA-Z0-9]+) = (?<defaultDelegate>[_a-zA-Z0-9]+);)"),
                ↪  "${classDeclarationBegin}/*~${name}~*/${middle}${eventDeclaration}", 0),
```

```
            // Inside the scope of ~!ExceptionIgnored!~ replace:
            // ExceptionIgnored.Invoke(NULL, exception);
            // ExceptionIgnored(NULL, exception);
            (new Regex(@"(?<scope>/\*~(?<eventName>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>((?<!/\*~\k<eventName>~\*/)(.|\n))*?)\k<eventName>\.Invoke"), "${scope}${separator}${before}${eventName}", 10),
            // Remove scope borders.
            // /*~ExceptionIgnored~*/
            //
            (new Regex(@"/\*~[a-zA-Z0-9]+~\*/"), "", 0),
            // Insert scope borders.
            // auto added = new StringBuilder();
            // /*~sb~*/std::string added;
            (new Regex(@"(auto|(System\.Text\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new (System\.Text\.)?StringBuilder\(\);"), "/*~${variable}~*/std::string ${variable};", 0),
            // static void Indent(StringBuilder sb, int level)
            // static void Indent(/*~sb~*/StringBuilder sb, int level)
            (new Regex(@"(?<start>, |\()(System\.Text\.)?StringBuilder (?<variable>[a-zA-Z0-9]+)(?<end>,|\))"), "${start}/*~${variable}~*/std::string& ${variable}${end}", 0),
            // Inside the scope of ~!added!~ replace:
            // sb.ToString()
            // sb
            (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.ToString\(\)"), "${scope}${separator}${before}${variable}", 10),
            // sb.AppendLine(argument)
            // sb.append(Platform::Converters::To<std::string>(argument)).append(1, '\n')
            (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.AppendLine\((?<argument>[^\),\r\n]+)\)"), "${scope}${separator}${before}${variable}.append(Platform::Converters::To<std::string>(${argument})).append(1, '\\n')", 10),
            // sb.Append('\t', level);
            // sb.append(level, '\t');
            (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.Append\('(?<character>[^'\r\n]+)', (?<count>[^\),\r\n]+)\)"), "${scope}${separator}${before}${variable}.append(${count}, '${character}')", 10),
            // sb.Append(argument)
            // sb.append(Platform::Converters::To<std::string>(argument))
            (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.Append\((?<argument>[^\),\r\n]+)\)"), "${scope}${separator}${before}${variable}.append(Platform::Converters::To<std::string>(${argument}))", 10),
            // Remove scope borders.
            // /*~sb~*/
            //
            (new Regex(@"/\*~[a-zA-Z0-9]+~\*/"), "", 0),
            // Insert scope borders.
            // auto added = new HashSet<TElement>();
            // ~!added!~std::unordered_set<TElement> added;
            (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new HashSet<(?<element>[a-zA-Z0-9]+)>\(\);"), "~!${variable}!~std::unordered_set<${element}> ${variable};", 0),
            // Inside the scope of ~!added!~ replace:
            // added.Add(node)
            // added.insert(node)
            (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?<!~!\k<variable>!~)(.|\n))*?)\k<variable>\.Add\((?<argument>[a-zA-Z0-9]+)\)"), "${scope}${separator}${before}${variable}.insert(${argument})", 10),
            // Inside the scope of ~!added!~ replace:
            // added.Remove(node)
            // added.erase(node)
            (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?<!~!\k<variable>!~)(.|\n))*?)\k<variable>\.Remove\((?<argument>[a-zA-Z0-9]+)\)"), "${scope}${separator}${before}${variable}.erase(${argument})", 10),
            // if (added.insert(node)) {
            // if (!added.contains(node)) { added.insert(node);
```

```
            (new Regex(@"if \((?<variable>[a-zA-Z0-9]+)\.insert\((?<argument>[a-zA-Z0-9]+)\)\)(?
                <separator>[\t ]*[\r\n]+)(?<indent>[\t ]*){"), "if
                (!${variable}.contains(${argument}))${separator}${indent}{" +
                Environment.NewLine + "${indent}    ${variable}.insert(${argument});", 0),
            // Remove scope borders.
            // ~!added!~
            //
            (new Regex(@"~![a-zA-Z0-9]+!~"), "", 5),
            // Insert scope borders.
            // auto random = new System::Random(0);
            // std::srand(0);
            (new Regex(@"[a-zA-Z0-9\.]+ ([a-zA-Z0-9]+) = new
                (System::)?Random\(([a-zA-Z0-9]+)\);"), "~!$1!~std::srand($3);", 0),
            // Inside the scope of ~!random!~ replace:
            // random.Next(1, N)
            // (std::rand() % N) + 1
            (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?<
                !~!\k<variable>!~)(.|\n))*?)\k<variable>\.Next\((?<from>[a-zA-Z0-9]+),
                (?<to>[a-zA-Z0-9]+)\)"), "${scope}${separator}${before}(std::rand() % ${to}) +
                ${from}", 10),
            // Remove scope borders.
            // ~!random!~
            //
            (new Regex(@"~![a-zA-Z0-9]+!~"), "", 5),
            // Insert method body scope starts.
            // void PrintNodes(TElement node, StringBuilder sb, int level) {
            // void PrintNodes(TElement node, StringBuilder sb, int level) {/*method-start*/
            (new Regex(@"(?<start>\r?\n[\t ]+)(?<prefix>((private|protected|public): )?(virtual
                )?[a-zA-Z0-9:_]+
                )?(?<method>[a-zA-Z][a-zA-Z0-9]*)\((?<arguments>[^\)]*)\)(?<override>(
                override)?)(?<separator>[ \t\r\n]*)\{(?<end>[^~])"), "${start}${prefix}${method}
                (${arguments})${override}${separator}{/*method-start*/${end}",
                0),
            // Insert method body scope ends.
            // {/*method-start*/...}
            // {/*method-start*/.../*method-end*/}
            (new Regex(@"\{/\*method-start\*/(?<body>((?<bracket>\{)|(?<-bracket>\})|[^\{\}]*)+)
                \}"), "{/*method-start*/${body}/*method-end*/}",
                0),
            // Inside method bodies replace:
            // GetFirst(
            // this->GetFirst(
            (new
                Regex(@"(?<scope>/\*method-start\*/)(?<before>((?<!/\*method-end\*/)(.|\n))*?)(?
                <separator>[\W](?<!(::|\.|->|throw\s+)))(?<method>(?!sizeof)[a-zA-Z0-9]+)\((?!\)
                \{)(?<after>(.|\n)*?)(?<scopeEnd>/\*method-end\*/)"),
                "${scope}${before}${separator}this->${method}(${after}${scopeEnd}", 100),
            // Remove scope borders.
            // /*method-start*/
            //
            (new Regex(@"/\*method-(start|end)\*/"), "", 0),
            // Insert scope borders.
            // const std::exception& ex
            // const std::exception& ex/*~ex~*/
            (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?exception&?
                (?<variable>[_a-zA-Z0-9]+))(?<after>\W)"),
                "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
            // Inside the scope of ~!ex!~ replace:
            // ex.Message
            // ex.what()
            (new Regex(@"(?<scope>/\*~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before
                >((?<!/\*~\k<variable>~\*/)(.|\n))*?)(Platform::Converters::To<std::string>\(\k<
                variable>\.Message\)|\k<variable>\.Message)"),
                "${scope}${separator}${before}${variable}.what()", 10),
            // Remove scope borders.
            // /*~ex~*/
            //
            (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
            // throw ObjectDisposedException(objectName, message);
            // throw std::runtime_error(std::string("Attempt to access disposed object
                [").append(objectName).append("]: ").append(message).append("."));
            (new Regex(@"throw ObjectDisposedException\((?<objectName>[a-zA-Z_][a-zA-Z0-9_]*),
                (?<message>[a-zA-Z0-9_]*[Mm]essage[a-zA-Z0-9_]*(\(\))?|[a-zA-Z_][a-zA-Z0-9_]*)\)
                ;"), "throw std::runtime_error(std::string(\"Attempt to access disposed object
                [\").append(${objectName}).append(\"]: \").append(${message}).append(\".\"));",
                0),
```

```
512                 // throw ArgumentNullException(argumentName, message);
513                 // throw std::invalid_argument(std::string("Argument
                    ↪  ").append(argumentName).append(" is null: ").append(message).append("."));
514                 (new Regex(@"throw
                    ↪  ArgumentNullException\((?<argument>[a-zA-Z]*[Aa]rgument[a-zA-Z]*),
                    ↪  (?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\))?)\);"), "throw
                    ↪  std::invalid_argument(std::string(\"Argument \").append(${argument}).append(\"
                    ↪  is null: \").append(${message}).append(\".\"));", 0),
515                 // throw ArgumentException(message, argumentName);
516                 // throw std::invalid_argument(std::string("Invalid ").append(argumentName).append("
                    ↪  argument: ").append(message).append("."));
517                 (new Regex(@"throw
                    ↪  ArgumentException\((?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\))?),
                    ↪  (?<argument>[a-zA-Z]*[Aa]rgument[a-zA-Z]*)\);"), "throw
                    ↪  std::invalid_argument(std::string(\"Invalid \").append(${argument}).append(\"
                    ↪  argument: \").append(${message}).append(\".\"));", 0),
518                 // throw ArgumentOutOfRangeException(argumentName, argumentValue, messageBuilder());
519                 // throw std::invalid_argument(std::string("Value
                    ↪  [").append(Platform::Converters::To<std::string>(argumentValue)).append("] of
                    ↪  argument [").append(argumentName).append("] is out of range:
                    ↪  ").append(messageBuilder()).append("."));
520                 (new Regex(@"throw ArgumentOutOfRangeException\((?<argument>[a-zA-Z]*[Aa]rgument[a-z
                    A-Z]*([Nn]ame[a-zA-Z]*)?),
                    ↪  (?<argumentValue>[a-zA-Z]*[Aa]rgument[a-zA-Z]*([Vv]alue[a-zA-Z]*)?),
                    ↪  (?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\))?)\);"), "throw
                    ↪  std::invalid_argument(std::string(\"Value
                    ↪  [\").append(Platform::Converters::To<std::string>(${argumentValue})).append(\"]
                    ↪  of argument [\").append(${argument}).append(\"] is out of range:
                    ↪  \").append(${message}).append(\".\"));", 0),
521                 // throw NotSupportedException();
522                 // throw std::logic_error("Not supported exception.");
523                 (new Regex(@"throw NotSupportedException\(\);"), "throw std::logic_error(\"Not
                    ↪  supported exception.\");", 0),
524                 // throw NotImplementedException();
525                 // throw std::logic_error("Not implemented exception.");
526                 (new Regex(@"throw NotImplementedException\(\);"), "throw std::logic_error(\"Not
                    ↪  implemented exception.\");", 0),
527                 // Insert scope borders.
528                 // const std::string& message
529                 // const std::string& message/*~message~*/
530                 (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?((std::)?string&?|char\*)
                    ↪  (?<variable>[_a-zA-Z0-9]+))(?<after>\W)"),
                    ↪  "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
531                 // Inside the scope of /*~message~*/ replace:
532                 // Platform::Converters::To<std::string>(message)
533                 // message
534                 (new Regex(@"(?<scope>/\*~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before
                    ↪  >((?<!/\*~\k<variable>~\*/)(.|\n))*?)Platform::Converters::To<std::string>\(\k<v
                    ↪  ariable>\)"), "${scope}${separator}${before}${variable}",
                    ↪  10),
535                 // Remove scope borders.
536                 // /*~ex~*/
537                 //
538                 (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
539                 // Insert scope borders.
540                 // std::tuple<T, T> tuple
541                 // std::tuple<T, T> tuple/*~tuple~*/
542                 (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?tuple<[^\n]+>&?
                    ↪  (?<variable>[_a-zA-Z0-9]+))(?<after>\W)"),
                    ↪  "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
543                 // Inside the scope of ~!ex!~ replace:
544                 // tuple.Item1
545                 // std::get<1-1>(tuple)
546                 (new Regex(@"(?<scope>/\*~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before
                    ↪  >((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.Item(?<itemNumber>\d+)(?<afte
                    ↪  r>\W)"),
                    ↪  "${scope}${separator}${before}std::get<${itemNumber}-1>(${variable})${after}",
                    ↪  10),
547                 // Remove scope borders.
548                 // /*~ex~*/
549                 //
550                 (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
551                 // Insert scope borders.
552                 // class Range<T> {
553                 // class Range<T> {/*~type~Range<T>~*/
```

```
554             (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)(template\s*<[^<>\n]*>
                ↪  )?(struct|class)
                ↪  (?<fullType>(?<typeName>[a-zA-Z0-9]+)(<[^:\n]*>)?)(\s*:\s*[^{\n]+)?[\t
                ↪  ]*(\r?\n)?[\t ]*{)"),
                ↪  "${classDeclarationBegin}/*~type~${typeName}~${fullType}~*/", 0),
555             // Inside the scope of /*~type~Range<T>~*/ insert inner scope and replace:
556             // public: static implicit operator std::tuple<T, T>(Range<T> range)
557             // public: operator std::tuple<T, T>() const {/*~variable~Range<T>~*/
558             (new Regex(@"(?<scope>/\*~type~(?<typeName>[^~\n\*]+)~(?<fullType>[^~\n\*]+)~\*/)(?<
                    separator>.|\n)(?<before>((?<!/\*~type~\k<typeName>~\k<fullType>~\*/)(.|\n))*?)(
                ↪  ?<access>(private|protected|public): )static implicit operator
                ↪  (?<targetType>[^\(\n]+)\((?<argumentDeclaration>\k<fullType>
                ↪  (?<variable>[a-zA-Z0-9]+))\)(?<after>\s*\n?\s*{)"),
                ↪  "${scope}${separator}${before}${access}operator ${targetType}()
                ↪  const${after}/*~variable~${variable}~*/", 10),
559             // Inside the scope of /*~type~Range<T>~*/ replace:
560             // public: static implicit operator Range<T>(std::tuple<T, T> tuple) { return new
                ↪  Range<T>(std::get<1-1>(tuple), std::get<2-1>(tuple)); }
561             // public: Range(std::tuple<T, T> tuple) : Range(std::get<1-1>(tuple),
                ↪  std::get<2-1>(tuple)) { }
562             (new Regex(@"(?<scope>/\*~type~(?<typeName>[^~\n\*]+)~(?<fullType>[^~\n\*]+)~\*/)(?<
                    separator>.|\n)(?<before>((?<!/\*~type~\k<typeName>~\k<fullType>~\*/)(.|\n))*?)(
                ↪  ?<access>(private|protected|public): )static implicit operator
                ↪  (\k<fullType>|\k<typeName>)\((?<arguments>[^{}\n]+)\)(\s|\n)*{(\s|\n)*return
                ↪  (new )?(\k<fullType>|\k<typeName>)\((?<passedArguments>[^\n]+)\);(\s|\n)*}"),
                ↪  "${scope}${separator}${before}${access}${typeName}(${arguments}) :
                ↪  ${typeName}(${passedArguments}) { }", 10),
563             // Inside the scope of /*~variable~range~*/ replace:
564             // range.Minimum
565             // this->Minimum
566             (new Regex(@"(?<scope>{/\*~variable~(?<variable>[^~\n]+)~\*/)(?<separator>.|\n)(?<be
                    fore>(?<beforeExpression>(?<bracket>{)|(?<-bracket>})|[^{}]|\n)*?)\k<variable>\.
                ↪  (?<field>[_a-zA-Z0-9]+)(?<after>(,|;|}|
                ↪  |\))(?<afterExpression>(?<bracket>{)|(?<-bracket>})|[^{}]|\n)*?})"),
                ↪  "${scope}${separator}${before}this->${field}${after}", 10),
567             // Remove scope borders.
568             // /*~ex~*/
569             //
570             (new Regex(@"/\*~[^~\n]+~[^~\n]+~\*/"), "", 0),
571             // Insert scope borders.
572             // namespace Platform::Ranges { ... }
573             // namespace Platform::Ranges {/*~start~namespace~Platform::Ranges~*/ ...
                ↪  /*~end~namespace~Platform::Ranges~*/}
574             (new Regex(@"(?<namespaceDeclarationBegin>\r?\n(?<indent>[\t ]*)namespace
                    (?<namespaceName>(?<namePart>[a-zA-Z][a-zA-Z0-9]+)(?<nextNamePart>::[a-zA-Z][a-z
                ↪  A-Z0-9]+)+)(\s|\n)*{)(?<middle>(.|\n)*)(?<end>(?<=\r?\n)\k<indent>}(?!;))"),
                ↪  "${namespaceDeclarationBegin}/*~start~namespace~${namespaceName}~*/${middle}/*~e
                ↪  nd~namespace~${namespaceName}~*/${end}",
                ↪  0),
575             // Insert scope borders.
576             // class Range<T> { ... };
577             // class Range<T> {/*~start~type~Range<T>~T~*/ ... /*~end~type~Range<T>~T~*/};
578             (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)template <typename
                    (?<typeParameter>[^\n]+)> (struct|class)
                ↪  (?<type>[a-zA-Z0-9]+<\k<typeParameter>>)(\s*:\s*[^{\n]+)?[\t ]*(\r?\n)?[\t
                ↪  ]*{(?<middle>(.|\n)*)(?<endIndent>(?<=\r?\n)\k<indent>)(?<end>};)"),
                ↪  "${classDeclarationBegin}/*~start~type~${type}~${typeParameter}~*/${middle}${end
                ↪  Indent}/*~end~type~${type}~${typeParameter}~*/${end}",
                ↪  0),
579             // Inside the scope replace:
580             // /*~start~namespace~Platform::Ranges~*/ ... /*~start~type~Range<T>~T~*/ ...
                ↪  public: override std::int32_t GetHashCode() { return {Minimum,
                ↪  Maximum}.GetHashCode(); } ... /*~end~type~Range<T>~T~*/ ...
                ↪  /*~end~namespace~Platform::Ranges~*/
581             // /*~start~namespace~Platform::Ranges~*/ ... /*~start~type~Range<T>~T~*/ ...
                    /*~end~type~Range<T>~T~*/ ... /*~end~namespace~Platform::Ranges~*/ namespace std
                ↪  { template <typename T> struct hash<Platform::Ranges::Range<T>> { std::size_t
                ↪  operator()(const Platform::Ranges::Range<T> &obj) const { return {Minimum,
                ↪  Maximum}.GetHashCode(); } }; }
```

```
582             (new Regex(@"(?<namespaceScopeStart>/\*~start~namespace~(?<namespace>[^~\n\*]+)~\*/)(?<betweenStartScopes>(.|\n)+)(?<typeScopeStart>/\*~start~type~(?<type>[^~\n\*]+)~(?<typeParameter>[^~\n\*]+)~\*/)(?<before>(.|\n)+?)(?<hashMethodDeclaration>\r?\n[ \t]*(?<access>(private|protected|public): )override std::int32_t GetHashCode\(\)(\s|\n)*{\s*(?<methodBody>[^\s][^\n]+[^\s])\s*}\s*)(?<after>(.|\n)+?)(?<typeScopeEnd>/\*~end~type~\k<type>~\k<typeParameter>~\*/)(?<betweenEndScopes>(.|\n)+)(?<namespaceScopeEnd>/\*~end~namespace~\k<namespace>~\*/)}\r?\n"),
                "${namespaceScopeStart}${betweenStartScopes}${typeScopeStart}${before}${after}${typeScopeEnd}${betweenEndScopes}${namespaceScopeEnd}}" + Environment.NewLine +
                Environment.NewLine + "namespace std" + Environment.NewLine + "{" +
                Environment.NewLine + "    template <typename ${typeParameter}>" +
                Environment.NewLine + "    struct hash<${namespace}::${type}>" +
                Environment.NewLine + "    {" + Environment.NewLine + "        std::size_t operator()(const ${namespace}::${type} &obj) const" + Environment.NewLine + "        {" + Environment.NewLine + "
                /*~start~method~*/${methodBody}/*~end~method~*/" + Environment.NewLine + "    }" + Environment.NewLine + "    };" + Environment.NewLine + "}" + Environment.NewLine, 10),
583             // Inside scope of /*~start~method~*/ replace:
584             // /*~start~method~*/ ... Minimum ... /*~end~method~*/
585             // /*~start~method~*/ ... obj.Minimum ... /*~end~method~*/
586             (new Regex(@"(?<methodScopeStart>/\*~start~method~\*/)(?<before>.+({|,))(?<name>[a-zA-Z][a-zA-Z0-9]+)(?<after>[^\n\.\(a-zA-Z0-9]((?!/\*~end~method~\*/)[^\n])+)(?<methodScopeEnd>/\*~end~method~\*/)"),
                "${methodScopeStart}${before}obj.${name}${after}${methodScopeEnd}", 10),
587             // Remove scope borders.
588             // /*~start~type~Range<T>~*/
589             //
590             (new Regex(@"/\*~[^~\*\n]+(~[^~\*\n]+)*~\*/"), "", 0),
591             // class Disposable<T> : public Disposable
592             // class Disposable<T> : public Disposable<>
593             (new Regex(@"(?<before>(struct|class) (?<type>[a-zA-Z][a-zA-Z0-9]*)<[^<>\n]+> : (?<access>(private|protected|public) )?\k<type>)(?<after>\b(?!<))"),
                "${before}<>${after}", 0),
594             // Insert scope borders.
595             // class Disposable<T> : public Disposable<> { ... };
596             // class Disposable<T> : public Disposable<>
                {/*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/ ...
                /*~end~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/};
597             (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)template[\t ]*<(?<typeParameters>[^\n]*)>[\t ]*(struct|class)[\t ]+(?<fullType>(?<type>[a-zA-Z][a-zA-Z0-9]*)(<[^<>\n]*>)?)[\t ]*:[\t ]*(?<access>(private|protected|public)[\t ]+)?(?<fullBaseType>(?<baseType>[a-zA-Z][a-zA-Z0-9]*)(<[^<>\n]*>)?)[\t ]*(\r?\n)?[\t ]*{)(?<middle>(.|\n)*)(?<beforeEnd>(?<=\r?\n)\k<indent>)(?<end>};)"),
                "${classDeclarationBegin}/*~start~type~${type}~${fullType}~${baseType}~${fullBaseType}~*/${middle}${beforeEnd}/*~end~type~${type}~${fullType}~${baseType}~${fullBaseType}~*/${end}",
                0),
598             // Inside the scope replace:
599             // /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/ ... ) : base(
                ... /*~end~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/
600             // /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/ ... ) :
                Disposable<>( /*~end~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/
601             (new Regex(@"(?<before>(?<typeScopeStart>/\*~start~type~(?<types>(?<type>[^~\n\*]+)~(?<fullType>[^~\n\*]+)~\k<type>~(?<fullBaseType>[^~\n\*]+))~\*/)(.|\n)+?\)\s*:\s)base(?<after>\((.|\n)+?(?<typeScopeEnd>/\*~end~type~\k<types>~\*/))"),
                "${before}${fullBaseType}${after}", 20),
602             // Inside the scope replace:
603             // /*~start~type~Disposable~Disposable<T>~X~X<>~*/ ... ) : base( ...
                /*~end~type~Disposable~Disposable<T>~X~X<>~*/
604             // /*~start~type~Disposable~Disposable<T>~X~X<>~*/ ... ) : X(
                /*~end~type~Disposable~Disposable<T>~X~X<>~*/
605             (new Regex(@"(?<before>(?<typeScopeStart>/\*~start~type~(?<types>(?<type>[^~\n\*]+)~(?<fullType>[^~\n\*]+)~(?<baseType>[^~\n\*]+)~(?<fullBaseType>[^~\n\*]+))~\*/)(.|\n)+?\)\s*:\s)base(?<after>\((.|\n)+?(?<typeScopeEnd>/\*~end~type~\k<types>~\*/))"), "${before}${baseType}${after}",
                20),
606             // Inside the scope replace:
607             // /*~start~type~Disposable~Disposable<T>~X~X<>~*/ ... public: Disposable(T object) { Object = object; } ... public: Disposable(T object) : Disposable(object) { } ... /*~end~type~Disposable~Disposable<T>~X~X<>~*/
608             // /*~start~type~Disposable~Disposable<T>~X~X<>~*/ ... public: Disposable(T object) { Object = object; } /*~end~type~Disposable~Disposable<T>~X~X<>~*/
```

```
609             (new Regex(@"(?<before>(?<typeScopeStart>/\*~start~type~(?<types>(?<type>[^~\n\*]+)~
                    (?<fullType>[^~\n\*]+)~(?<baseType>[^~\n\*]+)~(?<fullBaseType>[^~\n\*]+))~\*/)(.
                ↪  |\n)+?(?<constructor>(?<access>(private|protected|public):[\t
                ↪  ]*)?\k<type>\((?<arguments>[^()\n]+)\)\s*{[^{}\n]+})(.|\n)+?)(?<duplicateConstru
                ↪  ctor>(?<access>(private|protected|public):[\t
                ↪  ]*)?\k<type>\(\k<arguments>\)\s*:[^{}\n]+\s*{[^{}\n]+})(?<after>(.|\n)+?(?<typeS
                ↪  copeEnd>/\*~end~type~\k<types>~\*/))"), "${before}${after}",
                ↪  20),
610             // Remove scope borders.
611             // /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/
612             //
613             (new Regex(@"/\*~[^~\*\n]+(~[^~\*\n]+)*~\*/"), "", 0),
614             // Insert scope borders.
615             // private: inline static const AppDomain _currentDomain = AppDomain.CurrentDomain;
616             // private: inline static const AppDomain _currentDomain =
                ↪  AppDomain.CurrentDomain;/*~app-domain~_currentDomain~*/
617             (new Regex(@"(?<declaration>(?<access>(private|protected|public):[\t ]*)?(inline[\t
                ↪  ]+)?(static[\t ]+)?(const[\t ]+)?AppDomain[\t
                ↪  ]+(?<field>[a-zA-Z_][a-zA-Z0-9_]*)[\t ]*=[\t ]*AppDomain\.CurrentDomain;)"),
                ↪  "${declaration}/*~app-domain~${field}~*/", 0),
618             // Inside the scope replace:
619             // /*~app-domain~_currentDomain~*/ ... _currentDomain.ProcessExit += OnProcessExit;
620             // /*~app-domain~_currentDomain~*/ ... std::atexit(OnProcessExit);
621             (new Regex(@"(?<before>(?<fieldScopeStart>/\*~app-domain~(?<field>[^~\n\*]+)~\*/)(.|
                    \n)+?)\k<field>\.ProcessExit[\t ]*\+=[\t
                ↪  ]*(?<eventHandler>[a-zA-Z_][a-zA-Z0-9_]*);"), "${before}std::atexit(${eventHandl
                ↪  er});/*~process-exit-handler~${eventHandler}~*/",
                ↪  20),
622             // Inside the scope replace:
623             // /*~app-domain~_currentDomain~*/ ... _currentDomain.ProcessExit -= OnProcessExit;
624             // /*~app-domain~_currentDomain~*/ ... /* No translation. It is not possible to
                ↪  unsubscribe from std::atexit. */
625             (new Regex(@"(?<before>(?<fieldScopeStart>/\*~app-domain~(?<field>[^~\n\*]+)~\*/)(.|
                ↪  \n)+?\r?\n[\t ]*)\k<field>\.ProcessExit[\t ]*\-=[\t
                ↪  ]*(?<eventHandler>[a-zA-Z_][a-zA-Z0-9_]*);"), "${before}/* No translation. It is
                ↪  not possible to unsubscribe from std::atexit. */", 20),
626             // Inside the scope replace:
627             // /*~process-exit-handler~OnProcessExit~*/ ... static void OnProcessExit(void
                ↪  *sender, EventArgs e)
628             // /*~process-exit-handler~OnProcessExit~*/ ... static void OnProcessExit()
629             (new Regex(@"(?<before>(?<fieldScopeStart>/\*~process-exit-handler~(?<handler>[^~\n\|
                ↪  *]+)~\*/)(.|\n)+?static[\t ]+void[\t ]+\k<handler>\()[^()\n]+\)"), "${before})",
                ↪  20),
630             // Remove scope borders.
631             // /*~app-domain~_currentDomain~*/
632             //
633             (new Regex(@"/\*~[^~\*\n]+(~[^~\*\n]+)*~\*/"), "", 0),
634             // AppDomain.CurrentDomain.ProcessExit -= OnProcessExit;
635             // /* No translation. It is not possible to unsubscribe from std::atexit. */
636             (new Regex(@"AppDomain\.CurrentDomain\.ProcessExit -= ([a-zA-Z_][a-zA-Z0-9_]*);"),
                ↪  "/* No translation. It is not possible to unsubscribe from std::atexit. */", 0),
637         }.Cast<ISubstitutionRule>().ToList();
638
639         /// <summary>
640         /// <para>
641         /// The to list.
642         /// </para>
643         /// <para></para>
644         /// </summary>
645         public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
646         {
647             // IDisposable disposable)
648             // IDisposable &disposable)
649             (new Regex(@"(?<argumentAbstractType>I[A-Z][a-zA-Z0-9]+(<[^>\r\n]+>)?)
                ↪  (?<argument>[_a-zA-Z0-9]+)(?<after>,|\))"), "${argumentAbstractType}
                ↪  &${argument}${after}", 0),
650             // ICounter<int, int> c1;
651             // ICounter<int, int>* c1;
652             (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^>\r\n]+>)?)
                ↪  (?<variable>[_a-zA-Z0-9]+)(?<after> = null)?;"), "${abstractType}
                ↪  *${variable}${after};", 0),
653             // (expression)
654             // expression
655             (new Regex(@"(\(| )\(([a-zA-Z0-9_\*:]+)\)(,| |;|\))"), "$1$2$3", 0),
656             // (method(expression))
657             // method(expression)
```

```
            (new Regex(@"(?<firstSeparator>(\(|
            ↪  ))\((?<method>[a-zA-Z0-9_\->\*:]+)\((?<expression>((?<parenthesis>\()|(?<-parent
            ↪  hesis>\))|[a-zA-Z0-9_\->\*:]*)+)(?(parenthesis)(?!))\)\)(?<lastSeparator>(,|
            ↪  |;|\)))"), "${firstSeparator}${method}(${expression})${lastSeparator}", 0),
            // .append(".")
            // .append(1, '.');
            (new Regex(@"\.append\(""([^\\""]|\\[^""])""\)"), ".append(1, '$1')", 0),
            // return ref _elements[node];
            // return &_elements[node];
            (new Regex(@"return ref ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9\*]+)\];"), "return &$1[$2];",
            ↪  0),
            // ((1, 2))
            // ({1, 2})
            (new Regex(@"(?<before>\(|, )\((?<first>[^\n()]+),
            ↪  (?<second>[^\n()]+)\)(?<after>\)|, )"), "${before}{${first},
            ↪  ${second}}${after}", 10),
            // {1, 2}.GetHashCode()
            // Platform::Hashing::Hash(1, 2)
            (new Regex(@"{(?<first>[^\n{}]+), (?<second>[^\n{}]+)}\.GetHashCode\(\)"),
            ↪  "Platform::Hashing::Hash(${first}, ${second})", 10),
            // range.ToString()
            // Platform::Converters::To<std::string>(range).data()
            (new Regex(@"(?<before>\W)(?<variable>[_a-zA-Z][_a-zA-Z0-9]+)\.ToString\(\)"),
            ↪  "${before}Platform::Converters::To<std::string>(${variable}).data()", 10),
            // new
            //
            (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)new\
            ↪  s+"), "${before}",
            ↪  10),
            // x == null
            // x == nullptr
            (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(?<v
            ↪  ariable>[_a-zA-Z][_a-zA-Z0-9]+)(?<operator>\s*(==|!=)\s*)null(?<after>\W)"),
            ↪  "${before}${variable}${operator}nullptr${after}", 10),
            // null
            // {}
            (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)null
            ↪  (?<after>\W)"), "${before}{}${after}",
            ↪  10),
            // default
            // 0
            (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)defa
            ↪  ult(?<after>\W)"), "${before}0${after}",
            ↪  10),
            // object x
            // void *x
            (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(?<!
            ↪  @)(object|System\.Object) (?<after>\w)"), "${before}void *${after}",
            ↪  10),
            // <object>
            // <void*>
            (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(?<!
            ↪  @)(object|System\.Object)(?<after>\W)"), "${before}void*${after}",
            ↪  10),
            // @object
            // object
            (new Regex(@"@([_a-zA-Z0-9]+)"), "$1", 0),
            // this->GetType().Name
            // typeid(this).name()
            (new Regex(@"(this)->GetType\(\)\.Name"), "typeid($1).name()", 0),
            // ArgumentNullException
            // std::invalid_argument
            (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(Sys
            ↪  tem\.)?ArgumentNullException(?<after>\W)"),
            ↪  "${before}std::invalid_argument${after}", 10),
            // InvalidOperationException
            // std::runtime_error
            (new Regex(@"(\W)(InvalidOperationException|Exception)(\W)"),
            ↪  "$1std::runtime_error$3", 0),
            // ArgumentException
            // std::invalid_argument
            (new Regex(@"(\W)(ArgumentException|ArgumentOutOfRangeException)(\W)"),
            ↪  "$1std::invalid_argument$3", 0),
            // template <typename T> struct Range : IEquatable<Range<T>>
            // template <typename T> struct Range {
```

```
            (new Regex(@"(?<before>template <typename (?<typeParameter>[^\n]+)> (struct|class)
            ↪  (?<type>[a-zA-Z0-9]+<[^\n]+>)) : (public
            ↪  )?IEquatable<\k<type>>(?<after>(\s|\n)*{)"), "${before}${after}", 0),
            // public: delegate void Disposal(bool manual, bool wasDisposed);
            // public: delegate void Disposal(bool, bool);
            (new Regex(@"(?<before>(?<access>(private|protected|public): )delegate
                (?<returnType>[a-zA-Z][a-zA-Z0-9:]+)
            ↪  (?<delegate>[a-zA-Z][a-zA-Z0-9]+)\(((?<leftArgumentType>[a-zA-Z][a-zA-Z0-9:]+),
            ↪  )*)(?<argumentType>[a-zA-Z][a-zA-Z0-9:]+)
            ↪  (?<argumentName>[a-zA-Z][a-zA-Z0-9]+)(?<after>(,
            ↪  (?<rightArgumentType>[a-zA-Z][a-zA-Z0-9:]+)
            ↪  (?<rightArgumentName>[a-zA-Z][a-zA-Z0-9]+))*\);)"),
            ↪  "${before}${argumentType}${after}", 20),
            // public: delegate void Disposal(bool, bool);
            // using Disposal = void(bool, bool);
            (new Regex(@"(?<access>(private|protected|public): )delegate
            ↪  (?<returnType>[a-zA-Z][a-zA-Z0-9:]+)
            ↪  (?<delegate>[a-zA-Z][a-zA-Z0-9]+)\((?<argumentTypes>[^\(\)\n]*)\);"), "using
            ↪  ${delegate} = ${returnType}(${argumentTypes});", 20),
            // <4-1>
            // <3>
            (new Regex(@"(?<before><)4-1(?<after>>)"), "${before}3${after}", 0),
            // <3-1>
            // <2>
            (new Regex(@"(?<before><)3-1(?<after>>)"), "${before}2${after}", 0),
            // <2-1>
            // <1>
            (new Regex(@"(?<before><)2-1(?<after>>)"), "${before}1${after}", 0),
            // <1-1>
            // <0>
            (new Regex(@"(?<before><)1-1(?<after>>)"), "${before}0${after}", 0),
            // #region Always
            //
            (new Regex(@"(^|\r?\n)[ \t]*\#(region|endregion)[^\r\n]*(\r?\n|$)"), "", 0),
            // //#define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
            //
            (new Regex(@"\/\/[ \t]*\#define[ \t]+[_a-zA-Z0-9]+[ \t]*"), "", 0),
            // #if USEARRAYPOOL\r\n#endif
            //
            (new Regex(@"#if [a-zA-Z0-9]+\s+#endif"), "", 0),
            // [Fact]
            //
            (new Regex(@"(?<firstNewLine>\r?\n|\A)(?<indent>[\t
            ↪  ]+)\[[a-zA-Z0-9]+(\((?<expression>((?<parenthesis>\()|(?<-parenthesis>\))|[^()\r
            ↪  \n]*)+)(?(parenthesis)(?!))\))?\][ \t]*(\r?\n\k<indent>)?"),
            ↪  "${firstNewLine}${indent}", 5),
            // \A \n ... namespace
            // \Anamespace
            (new Regex(@"(\A)(\r?\n)+namespace"), "$1namespace", 0),
            // \A \n ... class
            // \Aclass
            (new Regex(@"(\A)(\r?\n)+class"), "$1class", 0),
            // \n\n\n
            // \n\n
            (new Regex(@"\r?\n[ \t]*\r?\n[ \t]*\r?\n"), Environment.NewLine +
            ↪  Environment.NewLine, 50),
            // {\n\n
            // {\n
            (new Regex(@"{[ \t]*\r?\n[ \t]*\r?\n"), "{" + Environment.NewLine, 10),
            // \n\n}
            // \n}
            (new Regex(@"\r?\n[ \t]*\r?\n(?<end>[ \t]*})"), Environment.NewLine + "${end}", 10),
        }.Cast<ISubstitutionRule>().ToList();

        /// <summary>
        /// <para>
        /// Initializes a new <see cref="CSharpToCppTransformer"/> instance.
        /// </para>
        /// <para></para>
        /// </summary>
        /// <param name="extraRules">
        /// <para>A extra rules.</para>
        /// <para></para>
        /// </param>
        public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
        ↪  base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
```

```
768
769         /// <summary>
770         /// <para>
771         /// Initializes a new <see cref="CSharpToCppTransformer"/> instance.
772         /// </para>
773         /// <para></para>
774         /// </summary>
775         public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
776     }
777 }
```

1.2 ./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs

```
1  using Xunit;
2
3  namespace Platform.RegularExpressions.Transformer.CSharpToCpp.Tests
4  {
5      /// <summary>
6      /// <para>
7      /// Represents the sharp to cpp transformer tests.
8      /// </para>
9      /// <para></para>
10     /// </summary>
11     public class CSharpToCppTransformerTests
12     {
13         /// <summary>
14         /// <para>
15         /// Tests that empty line test.
16         /// </para>
17         /// <para></para>
18         /// </summary>
19         [Fact]
20         public void EmptyLineTest()
21         {
22             // This test can help to test basic problems with regular expressions like incorrect
                ↪  syntax
23             var transformer = new CSharpToCppTransformer();
24             var actualResult = transformer.Transform("");
25             Assert.Equal("", actualResult);
26         }
27
28         /// <summary>
29         /// <para>
30         /// Tests that hello world test.
31         /// </para>
32         /// <para></para>
33         /// </summary>
34         [Fact]
35         public void HelloWorldTest()
36         {
37             const string helloWorldCode = @"using System;
38 class Program
39 {
40     public static void Main(string[] args)
41     {
42         Console.WriteLine(""Hello, world!"");
43     }
44 }";
45             const string expectedResult = @"class Program
46 {
47     public: static void Main(std::string args[])
48     {
49         printf(""Hello, world!\n"");
50     }
51 };";
52             var transformer = new CSharpToCppTransformer();
53             var actualResult = transformer.Transform(helloWorldCode);
54             Assert.Equal(expectedResult, actualResult);
55         }
56     }
57 }
```

Index