```
LinksPlatform's Platform RegularExpressions Transformer CSharpToCpp Class Library
     ./csharp/Platform.Regular Expressions. Transformer. CSharp To Cpp/CSharp To Cpp Transformer. cs
   using System;
   using System.Collections.Generic;
2
   using System.Linq;
   using System. Text. Regular Expressions;
4
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform.RegularExpressions.Transformer.CSharpToCpp
        public class CSharpToCppTransformer : TextTransformer
10
11
            public static readonly IList<ISubstitutionRule> FirstStage = new List<SubstitutionRule>
12
13
14
                //
15
                (new Regex(0"(\r?\n)?[\t]+//+.+"), "", 0),
16
                // #pragma warning disable CS1591 // Missing XML comment for publicly visible type
                    or member
18
                 (new Regex(0"^\s*?\#pragma[\sa-zA-Z0-9]+$"), "", 0),
19
                // \{ n \in \mathbb{N} 
                // {
                (new Regex(0"\{\s+[\r\n]+"\}, "{" + Environment.NewLine, 0),
22
                // Platform.Collections.Methods.Lists
                // Platform::Collections::Methods::Lists
                 (new Regex(0"(namespace[^{r}_1+?)\.([^{r}_1+?)"), "$1::$2", 20),
25
                // nameof(numbers)
26
                 // "numbers"
27
                 (new
2.8
                    Regex(@"(?\before>\begin{picture}((-)\n]+\.)?(?\name>[a-zA-ZO-9_]+)(<[^)\n]+>)?()"),
                     "${before}\"${name}\"", 0),
                // Insert markers
2.9
                // EqualityComparer<T> _equalityComparer = EqualityComparer<T>.Default;
// EqualityComparer<T> _equalityComparer =
30

→ EqualityComparer<T>.Default; /*~_comparer~*/
                 (new Regex(0"(?<declaration>EqualityComparer<(?<type>[^>\n]+)>
32
                     (?<comparer>[a-zA-Z0-9_]+) = EqualityComparer<\k<type>>\.Default;)"),
                     "${declaration}/*~${comparer}~*/", 0),
                // /*~_equalityComparer~*/...equalityComparer.Equals(Minimum, value)
// /*~_equalityComparer~*/...Minimum == value
33
                 (new Regex(0"(?<before>/\*^(?<comparer>[a-zA-Z0-9_]+)^\*/(.|\n)+\W)\k<comparer>\.Equ_|
35
                    als((?<left>[^, \n]+), (?<right>[^)\n]+)))), "${before}${left} == ${right}",
                 \hookrightarrow
                     50),
                // Remove markers
36
                // /*~_equalityComparer~*/
38
                (new Regex(0"\r?\n[^\n]+/\*[a-zA-Z0-9_]+^{*}\*/"), "", 10),
39
                // Insert markers
40
                // Comparer<T> _comparer = Comparer<T>.Default;
// Comparer<T> _comparer = Comparer<T>.Default;
                                 _comparer = Comparer<T>.Default;/*~_comparer~*/
42
                (new Regex(@"(?<declaration>Comparer<(?<type>[^>\n]+)> (?<comparer>[a-zA-Z0-9_]+) =
43
                    Comparer < \k < type >> \. Default;)"), "$ {declaration} / * ~ $ {comparer} ~ * / ", 0),
                // /*~_comparer~*/..._comparer.Compare(Minimum, value) <= 0</pre>
                // /*~_comparer~*/...Minimum <= value
                 (new Regex(@"(?<before>/\*~(?<comparer>[a-zA-Z0-9_]+)~\*/(.|\n)+\W)\k<comparer>\.Com_
46
                    pare\((?<left>[^,\n]+)
                     "${before}${left} ${comparison} ${right}${after}", 50),
                // Remove markers
47
                // private static readonly Comparer<T> _comparer =
                     Comparer<T>.Default;/*~_comparer~*/
                //
                (new Regex(0"\r?\n[^\n]+/\*^[a-zA-Z0-9_]+^\*/"), "", 10),
50
                // Comparer<TArgument>.Default.Compare(maximumArgument, minimumArgument) < 0
                // maximumArgument < minimumArgument</pre>
                 (new Regex(@"Comparer<[^>\n]+>\.Default\.Compare\(\s*(?<first>[^,)\n]+),\s*(?<second |</pre>
53
                    \ >[^{\n}+)\s*(\comparison>[<>=]=?)\s*0(?<after>\D)"), "${first}
                    ${comparison} ${second}${after}", 0)
                // public static bool operator ==(Range<T> left, Range<T> right) =>
54
                    left.Equals(right);
                 (\text{new Regex}(@''\r')\n[^\n] + \text{bool operator} == ((?<type>[^\n]+) (?<teft>[a-zA-Z0-9]+),
                     \k < type > (? < right > [a-zA-Z0-9]+) \) = >
                    (\k<left>|\k<right>)\. Equals\((\k<left>|\k<right>)\);"), "", 10)
                // public static bool operator !=(Range<T> left, Range<T> right) => !(left == right);
```

```
(\text{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),
                         // public override bool Equals(object obj) => obj is Range<T> range ? Equals(range)
                                : false;
                         (new Regex(@"\r?\n[^\n]+override bool Equals\((System\.)?[Oo]bject
                               // out TProduct
                         // TProduct
64
                         (new Regex(@"(?<before>(<|, ))(in|out)</pre>
65
                                (?<typeParameter>[a-zA-Z0-9]+)(?<after>(>|,))"),
                                "${before}${typeParameter}${after}", 10),
                         // public ...
66
                         // public:
67
                         (new Regex(0"(?<newLineAndIndent>\r?\n?[
68
                                \t^* (?<before>[^{{\(\r\n]*)}(?<access>private|protected|public)[ \t]+(?![^{{\(\r\n)}*)}
                                \n]*((?<=\s)|\W)(interface|class|struct)(\W)[^{{(\r\n]}*[{(\r\n])"},
                                "${newLineAndIndent}${access}: ${before}", 0),
                         // public: static bool CollectExceptions { get; set; }
                         // public: inline static bool CollectExceptions;
70
                          (new Regex(@"(?<access>(private|protected|public): )(?<before>(static )?[^\r\n]+
71
                               )(?<ame>[a-zA-Z0-9]+) {[^;}]*(?<=\\W)get;[^;\]*(?<=\\W)set;[^;\]*\"),
                               "${access}inline ${before}${name};", 0),
                         // public abstract class
                         // class
73
                         (new Regex(@"((public|protected|private|internal|abstract|static)
74
                               )*(?<category>interface|class|struct)"), "${category}", 0),
                         // class GenericCollectionMethodsBase<TElement>
75
                         // template <typename TElement> class GenericCollectionMethodsBase {
76
                          (new Regex(0"(?<before>\r?\n)(?<indent>[ \t]*)(?<type>class|struct)
77
                                (?<typeName>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9]+)
                                ,]+)>(?<typeDefinitionEnding>[^{\{}]+){"), "${before}${indent}template <typename
                                ...> ${type} ${typeName};" + Environment.NewLine + "${indent}template <typename
                                ${typeParameters}> ${type}
                                $\{\typeName}<\$\{\typeParameters}>\$\{\typeDefinitionEnding}\{\t", 0),
                         // static void
                          TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
                               tree, TElement* root)
                         // template<typename T> static void
                          TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>

    tree, TElement* root)

                         (\text{new Regex}(0"\text{static}([a-zA-Z0-9]+)([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>(([^\)\r\n]+)\)"),
80
                                "template <typename $3> static $1 $2($4)", 0),
                         // interface IFactory<out TProduct> {
                         // template <typename...> class IFactory;\ntemplate <typename TProduct> class
                              IFactory<TProduct>
                          (new Regex(@"(?<before>\r?\n)(?<indent>[ \t]*)interface
83
                                (?<interface>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9]
                                ,]+)>(?<typeDefinitionEnding>[^{]+){"}, "${before}${indent}template <typename
                                 ...> class ${interface};" + Environment.NewLine + "${indent}template <typename
                                ${typeParameters}> class
                                ${interface}<${typeParameters}>${typeDefinitionEnding}{" + Environment.NewLine +
                                       public:", 0),
                         // template <typename TObject, TProperty, TValue>
// template <typename TObject, typename TProperty, typename TValue>
(new Regex(@"(?<before>template <((, )?typename [a-zA-ZO-9]+)+,</pre>
85
                                )(?<typeParameter>[a-zA-Z0-9]+)(?<after>(,|>))"), "${before}typename
                               ${typeParameter}${after}", 10),
                         // Insert markers
                         // private: static void BuildExceptionString(this StringBuilder sb, Exception
                               exception, int level)
                         // /*~extensionMethod~BuildExceptionString~*/private: static void
                          → BuildExceptionString(this StringBuilder sb, Exception exception, int level)
                          (new Regex(@"private: static [^{r}] + (?^{a-20-9}) + (this [^{)}r^{+})),
                               "/*~extensionMethod~${name}~*/$0", 0),
                         // Move all markers to the beginning of the file.
                         (\text{new Regex}(@''\A(?<\text{before})^r\n] + r?\n(.|\n) +) (?<\text{marker}/\*`extensionMethod}^*(?<\text{name}) + r?\n(.|\n) +) (?<\text{marker}/\*`extensionMethod}^*(?<\text{name}) + r?\n(.|\n) +) (?<\text{marker}/\*`extensionMethod}^*(?<\text{name}) + r?\n(.|\n) +) (?<\text{marker}/\n) + r?\n(.|\n) + r?\n(.|\n) +) (?<\text{marker}/\n) + r?\n(.|\n) + r?\n(.|\n) +) (?<\text{marker}/\n) + r?\n(.|\n) + r?\n(.|\
92
                                [a-zA-Z0-9]+)^*/", "${marker}${before}",
                                10),
                         // /*~extensionMethod~BuildExceptionString~*/...sb.BuildExceptionString(exception.In |
                              nerException, level +
                               1):
```

```
// /*~extensionMethod~BuildExceptionString~*/...BuildExceptionString(sb,

→ exception.InnerException, level + 1);

                            (\underline{new Regex(@"(?<before>/\*^extensionMethod^(?<\underline{name>[a-zA-Z0-9]+)^*/(.|\n)+\W)(?<\underline{var})}
                                   iable>[_a-zA-Z0-9]+)\.\k<name>\("), "${before}${name}(${variable}, ",
                                  50),
                           // Remove markers
96
                           // /*~extensionMethod~BuildExceptionString~*/
9.8
                            (new Regex(0"/\*~extensionMethod~[a-zA-Z0-9]+~\*/"), "", 0),
99
                           // (this
100
                           // (
                           (new Regex(0"\(this "), "(", 0),
102
                           // private: static readonly Disposal _emptyDelegate = (manual, wasDisposed) => { };
103
                           // private: inline static std::function<Disposal> _emptyDelegate = [](auto manual,
                            → auto wasDisposed) { };
                            (new Regex(@"(?<access>(private|protected|public): )?static readonly
105
                                   (?<type>[a-zA-Z][a-zA-Z0-9]*) (?<name>[a-zA-Z_][a-zA-Z0-9_]*) =
                                   ((?\langle firstArgument\rangle [a-zA-Z_] [a-zA-Z0-9_]*)
                                   (?\langle secondArgument \rangle [a-zA-Z_{-}][a-zA-Z0-9_{-}]*) \rangle) => \{\s*\};"), "$\{access\}inline static \} 
                                  std::function<${type}> ${name} = [](auto ${firstArgument}, auto
                                  ${secondArgument}) { };", 0),
                           // public: static readonly EnsureAlwaysExtensionRoot Always = new
106
                                  EnsureAlwaysExtensionRoot();
                            // public: inline static EnsureAlwaysExtensionRoot Always;
                            (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),
                           // public: static readonly Range<int> SByte = new
109
                                  Range<int>(std::numeric_limits<int>::min(), std::numeric_limits<int>::max());
                           // public: inline static Range<int> SByte =
110
                                 Range<int>(std::numeric_limits<int>::min(), std::numeric_limits<int>::max());
                            (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),
                           // public: static readonly string ExceptionContentsSeparator = "---"
112
                           // public: inline static std::string ExceptionContentsSeparator = "---";
113
                            (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly) string
                                   (?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) = ""(?\langle \bar{n} = 2A - Z0 - 9] + ) 
                                 static std::string ${name} = \"${string}\";", 0),
                           // private: const int MaxPath = 92;
115
                           // private: inline static const int MaxPath = 92;
116
                            (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly)
                                   (?\langle type \rangle [a-zA-Z0-9]+) (?\langle name \rangle [a-zA-Z0-9]+) = (?\langle value \rangle [^; \r\n]+);"),
                                  "${access}inline static const ${type} ${name} = ${value};", 0),
                           //
                                  ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
                                  TArgument : class
                            // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
(new Regex(@"(?<before> [a-zA-Z]+\(([a-zA-Z *,]+, |))(?<type>[a-zA-Z]+)(?<after>(|
119
                                   [a-zA-Z *,]+)))[ \r\n]+where \k<type> : class"), "${before}${type}*${after}",
                                  0),
                           // protected: abstract TElement GetFirst();
121
                           // protected: virtual TElement GetFirst() = 0;
122
                            (new Regex(@"(?<access>(private|protected|public): )?abstract
                                   (?<method>[^; \r\n]+);"), "${access}virtual ${method} = 0;", 0),
                           // TElement GetFirst();
                           // virtual TElement GetFirst() = 0;
125
                            (new Regex(0"(?<br/>before>[r]+[]+)(?<methodDeclaration>(?!return)[a-zA-Z0-9]+
126
                           127
                           // protected: TreeElement _elements[N];
128
                            (new Regex(0"(?<access>(private|protected|public): )?readonly
129
                                 (?<type>[a-zA-Z<>0-9]+)([\[\]]+) (?<name>[a-zA-Z0-9]+);"), "${access}${type}
                                  ${name}[N];", 0),
                           // protected: readonly TElement Zero;
130
                            // protected: TElement Zero;
131
                            (new Regex(@"(?<access>(private|protected|public): )?readonly
132
                                  (?<type>[a-zA-Z<>0-9]+) (?<name>[_a-zA-Z0-9]+);"), "${access}${type} ${name};",
                                  0),
                           // internal
133
134
                            (new Regex(0"(\W)internal\s+"), "$1", 0),
135
                           // 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
138
                               )?(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;
139
                        // SizeBalancedTree(int capacity) { a = b;
140
                         (new Regex(@"(^\s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static
                               )?(\bar{o}verride )?(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))
144
                               )?(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*)\/((?<firstArg_l)))
147
                              ument>[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; }
149
                         (new Regex(@"\())\s+=>\s+(?<expression>[^(),;\r\n]+(\(((?<parenthesis>\())|(?<-parent_|</pre>
150
                              hesis>\))|[^();\r\n]*?\*?\))?[^(),;\r\n]*)(?<after>,|\);)"), "() { return
                               ${expression}; }${after}", 0)
                        // ~DisposableBase() => Destruct();
151
                         // ~DisposableBase() { Destruct();
                         (new Regex(0"~(?<class>[a-zA-Z_][a-zA-Z0-9_]*)\(\)\s+=>\s+([^;\r\n]+?);"),
153
                               "~${class}() { $1; }", 0),
                        // => Integer<TElement>.Zero;
154
                        // { return Integer<TElement>.Zero; }
155
                         (new Regex(0"\)\\ddot{s}+=>\s+([^;\r\n]+?);"), ") { return $1; }", 0),
                        // () { return avlTree.Count; }
157
                        // [&]()-> auto { return avlTree.Count; }
158
                         (new Regex(@"(?<before>, |\()\(\) { return (?<expression>[^;\r\n]+); }"),
159
                              "${before}[&]()-> auto { return ${expression}; }", 0),
                        // Count => GetSizeOrZero(Root);
                        // Count() { return GetSizeOrZero(Root); }
161
                         (\text{new Regex}(@"(\W)([A-Z][a-zA-Z]+)\s+=>\s+([^;\r\n]+);"), "$1$2() { return $3; }", 0),
162
                        // public: T Object { get; }
                        // public: const T Object;
164
                         (new Regex(@"(?<before>[^\r]\r?\n[ \t]*)(?<access>(private|protected|public):
165
                               )?(?<type>[a-zA-Z_][a-zA-Z0-9_:<>]*)
                                (?\property>[a-zA-Z_][a-zA-Z0-9_]*) (?\block0pen>[\n\s]*{[\n\s]*) (\[[^\n]+\][\n\s]*) (\property>[a-zA-Z_][a-zA-Z0-9_]*) (?\block0pen>[\n\s]*{[\n\s]*) (\property>[a-zA-Z_][a-zA-Z0-9_]*) (?\block0pen>[\n\s]*{[\n\s]*) (\property>[a-zA-Z_][a-zA-Z0-9_]*) (?\block0pen>[\n\s]*{[\n\s]*) (\property>[a-zA-Z_][a-zA-Z0-9_]*) (?\block0pen>[\n\s]*{[\n\s]*) (\property>[a-zA-Z_][a-zA-Z0-9_]*) (?\block0pen>[\n\s]*{[\n\s]*}) (\property>[a-zA-Z0-9_]*) (?\block0pen>[\n\s]*{[\n\s]*}) (\property>[a-zA-Z0-9_]*) (?\block0pen>[\n\s]*{[\n\s]*}) (\property>[\n\s]*{[\n\s]*}) (\propert
                              ]*)?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; }
167
                         (new Regex(@"(?<before>[^\r]\r?\n[ \t]*)(?<access>(private|protected|public):
168
                               )?(?<virtual>virtual )?bool
                               (?\property>[a-zA-Z_][a-zA-Z0-9_]*)(?\block0pen>[\n\s]*{[\n\s]*)(\[[^\n]+)][\n\s_1](\n\s_1)}
                              ]*)?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;
169
170
                         (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_]*)(?\block0pen>[\n\s]*{[\n\s]*)(\[[^\n]+\][\n\s]*)
                               ]*)?get\s*=>\s*(?<expression>[^\n]+);(?<blockClose>[\n\s]*}[\n\s]*)"),
                               "${before}${access}${virtual}${type} ${property}()${blockOpen}return
                        172
                        // ArgumentInRange(string message) { auto messageBuilder = [&]() -> string { return
173

→ message: }

                         (\text{new Regex}(@"(?\before>\W[_a-zA-ZO-9]+\([^\)\n]*\)[\s\n]*{[\s\n]*([^{}]|\n)*?(\r?\n)_{}})
                               ?[ \t]*)(?<returnType>[_a-zA-Z0-9*:]+[_a-zA-Z0-9*:]*)
                               [^{]}|^{n}+?)^{"}
                                                       "${before}auto ${methodName} = [&]() -> ${returnType}
                               {${body}};", 10),
                        // Func<TElement> treeCount
                         // std::function<TElement()> treeCount
176
                         (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", 0),
```

```
// Action<TElement> free
                 // std::function<void(TElement)> free
                 (\text{new Regex}(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<void($1)> $2",
180
                     0),
                 // Action<TPrimary, TAuxiliary> action
                 // std::function<void(TPrimary, TAuxiliary)> action
182
                 (\text{new Regex}(@"Action<([a-zA-Z0-9]+), ([a-zA-Z0-9]+)>([a-zA-Z0-9]+)"),
                     "std::function<void($1, $2)> $3", 0),
                 // , Action<TPrimary, TAuxiliary>>
// , std::function<void(TPrimary, TAuxiliary)>>
184
185
                 (new Regex(0"(, )Action<([a-zA-Z0-9]+), ([a-zA-Z0-9]+)>(>)"),
186
                     "$1std::function<void($2, $3)>$4", 0),
                 // Action action
                 // std::function<void()> action
188
                 (new Regex(@"Action ([a-zA-Z0-9]+)"), "std::function<void()> $1", 0),
189
                 // , Action>
                 // ,std::function<void()>>
191
                 (new Regex(@"(, )Action(>)"), "$1std::function<void()>$2", 0),
192
                 // Predicate<TArgument> predicate
193
                 (new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<bool($1)>
195
                     $2", 0),
                 // var
196
                 // auto
                 (new Regex(@"(\W)var(\W)"), "$1auto$2", 0),
198
                 // unchecked
199
200
                 (new Regex(0"[\r\n]{2}\s*?unchecked\s*?$"), "", 0),
                 // throw new
202
                 // throw
203
                 (new Regex(@"(\W)throw new(\W)"), "$1throw$2", 0),
                 // void RaiseExceptionIgnoredEvent(Exception exception)
205
                 // void RaiseExceptionIgnoredEvent(const std::exception& exception)
206
                 (new Regex(@"(\(|, ))(System\.Exception|Exception)( |\))"), "$1const
207
                     std::exception&$3"
                                          0),
                 // EventHandler<Exception>
                 // EventHandler<std::exception>
20.9
                 (new Regex(@"(\W)(System\.Exception|Exception)(\W)"), "$1std::exception$3", 0),
210
                 // override void PrintNode(TElement node, StringBuilder sb, int level)
211
                 // void PrintNode(TElement node, StringBuilder sb, int level) override
                 (new Regex(0"override ([a-zA-Z0-9 \*\-]+)(\([^\)\r\n]+?\))"), "$1$2 override", 0),
213
                 // return (range.Minimum, range.Maximum)
214
                 // return {range.Minimum, range.Maximum}
                 (new Regex(@"(?<before>return\s*)\((?<values>[^\)\n]+)\)(?!\()(?<after>\W)"),
216
                     "${before}{${values}}${after}", 0),
                 // string
217
                 // std::string
218
                 (new Regex(@"(?<before>\W)(?<!::)string(?<after>\W)"),
                     "${before}std::string${after}", 0),
                 // System.ValueTuple
220
                 // std::tuple
221
                 (new Regex(@"(?<before>\W)(System\.)?ValueTuple(?!\s*=|\()(?<after>\W)"),
                     "${before}std::tuple${after}", 0),
                 // sbyte
                 // std::int8_t
224
                 (\text{new Regex}(0"(?<\text{before}))((\text{System}.)?\text{SB}|\text{sb})\text{yte}(?!/\text{s*=}|\()(?<\text{after})))),
225
                     "${before}std::int8_t${after}", 0),
                 // short
                 // std::int16_t
227
                 (new\ Regex(@"(?<before>\W)((System\.)?Int16|short)(?!\s*=|\()(?<after>\W)"),
228
                     "${before}std::int16_t${after}", 0),
                 // int
229
                 // std::int32_t
                 (\text{new Regex}(@"(?<\text{before}\W)((System\.)?I|i)nt(32)?(?!\s*=|\()(?<\text{after}\W)"),
231
                     "${before}std::int32_t${after}", 0),
                 // long
232
                 // std::int64_t
233
                 (new Regex(@"(?<before>\W)((System\.)?Int64|long)(?!\s*=|\()(?<after>\W)"),
                     "${before}std::int64_t${after}", 0),
                 // byte
235
                 // std::uint8_t
236
                 (new Regex(@"(?<before>\W)((System\.)?Byte|byte)(?!\s*=|\()(?<after>\W)"),
                     "${before}std::uint8_t${after}", 0),
                 // ushort
                 // std::uint16_t
239
```

```
(new Regex(@"(?<before>\W)((System\.)?UInt16|ushort)(?!\s*=|\()(?<after>\W)"),
240
                                 "${before}std::uint16_t${after}", 0),
                           // uint
241
                           // std::uint32_t
242
                           (new Regex(@"(?<before>\W)((System\.)?UI|ui)nt(32)?(?!\s*=|\()(?<after>\W)"),
243
                                  "${before}std::uint32_t${after}", 0),
                           // ulong
244
                           // std::uint64_t
                           (new Regex(@"(?<before>\W)((System\.)?UInt64|ulong)(?!\s*=|\()(?<after>\W)"),
246
                                  "${before}std::uint64_t${after}", 0),
                           // char*[] args
247
                           // char* args[]
248
                            (\text{new Regex}(@"([_a-zA-Z0-9:\*]?)\[\] ([a-zA-Z0-9]+)"), "$1 $2[]", 0),
                           // float.MinValue
250
                           // std::numeric_limits<float>::lowest()
251
                           (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MinValue(?<after>\W|
                            _, )"), "${before}std::numeric_limits<${type}>::lowest()${after}",
                                 0),
                           // double.MaxValue
253
                           // std::numeric_limits<float>::max()
254
255
                           (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MaxValue(?<after>\W|
                                  )"), "${before}std::numeric_limits<${type}>::max()${after}",
                                  0),
                           // using Platform.Numbers;
                           //
257
                           (new Regex(0"([\r\n]{2}|^)\s*?using [\.a-zA-Z0-9]+;\s*?$"), "", 0),
258
                           // struct TreeElement { }
                           // struct TreeElement { };
260
                           (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([^;])"), "$1
261
                                  $2$3{$4};$5", 0),
262
                           // class Program {
                           // class Program { };
263
                           (new Regex(0"(?<type>struct|class)
264
                                  (?\name>[a-zA-Z0-9]+[^\r\n]*) (?\name>[\r\n]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\name)[\tau]+(?\n
                                  ]*)?)\{(?<body>[\S\s]+?[\r\n]+\k<indentLevel>)\}(?<afterBody>[^;]|$)"), "${type}
                                  ${name}${beforeBody}{${body}};${afterBody}", 0),
                           // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
265
                           // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
266
                           (\text{new Regex}(@"(struct|class) ([a-zA-Z0-9]+)(<[a-zA-Z0-9 ,]+>)? : ([a-zA-Z0-9]+)"),
267
                                  "$1 $2$3 : public $4", 0),
                           // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
268
                           // class IProperty : public ISetter<TValue, TObject>, public IProvider<TValue,
269
                                 TObject>
                           (new Regex(0"(?<before>(struct|class) [a-zA-Z0-9]+ : ((public
270
                                   [a-zA-Z0-9]+(<[a-zA-Z0-9],]+>)?
                                  )+)?)(?<inheritedType>(?!public)[a-zA-Z0-9]+(<[a-zA-Z0-9 ,]+>)?)(?<after>(,
                                 [a-zA-ZO-9]+(?!>)|[ \r\n]+))"), "\{before\}public \{inheritedType\}\{after\}", 10)
                           // Insert scope borders.
271
                           // ref TElement root
272
                           // ~!root!~ref TElement root
273
                           (\text{new Regex}(@"(?<\text{definition}>(?<= |\()(\text{ref }[a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!\text{ref}))))))
274
                                   \begin{tabular}{ll} (?<\variable>[a-zA-Z0-9]+)(?=\)|, | =))"), "~!${\rm variable}!~${\rm definition}", 0), \\ \end{tabular} 
                           // Inside the scope of ~!root!~ replace:
275
                           // root
276
                           // *root
                           (new Regex(0"(?<definition>~!(?<pointer>[a-zA-Z0-9]+)!~ref [a-zA-Z0-9]+
                                  \k<pointer>(?=\)|, | =))(?<before>((?<!~!\k<pointer>!~)(.|\n))*?)(?<prefix>(\W
                                  |\cdot()\rangle = (?\langle suffix\rangle ( |\cdot\rangle |; |,))"),
                                  "${definition}${before}${prefix}*${pointer}${suffix}", 70),
                           // Remove scope borders.
279
                                 ~!root!~
280
                           (new Regex(0"^{!}(?<pointer>[a-zA-Z0-9]+)!^{"}), "", 5),
282
                           // ref auto root = ref
283
                           // ref auto root =
                           (new Regex(0"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)"), "$1* $2 =$3", 0),
285
                               *root = ref left:
286
                           // root = left;
287
                           (\text{new Regex}(@"\*([a-zA-ZO-9]+) = \text{ref}([a-zA-ZO-9]+)(\W)"), "$1 = $2$3", 0),
288
                           // (ref left)
289
                           // (left)
290
                           (new Regex(0"\(ref ([a-zA-Z0-9]+)(\)|\(|,)"), "($1$2", 0),
                                ref<sup>TElement</sup>
292
                                  TElement*
293
                            (new Regex(0"(|\)()ref ([a-zA-Z0-9]+) "), "$1$2* ", 0),
                           // ref sizeBalancedTree.Root
295
```

```
// &sizeBalancedTree->Root
296
                                   (\text{new Regex}(0"\text{ref }([a-zA-Z0-9]+)).([a-zA-Z0-9]*]+)"), "&$1->$2", 0),
                                   // ref GetElement(node).Right
298
                                   // &GetElement(node)->Right
299
                                   (new Regex(0"ref ([a-zA-\bar{Z}0-9]+)\(([a-zA-\bar{Z}0-9\*]+)\)\.([a-zA-\bar{Z}0-9]+)"),
300
                                           "&$1($2)->$3", 0)
                                   // GetElement(node).Right
                                   // GetElement(node) ->Right
302
                                   (\text{new Regex}(@"([a-zA-Z0-9]+)\(([a-zA-Z0-9]*]+)\)).([a-zA-Z0-9]+)"), "$1($2)->$3", 0),
303
                                         [Fact]\npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
                                   // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
305
                                   (new Regex(Q^*\setminus [Fact][\s\n] + (public: )?(static )?void ([a-zA-ZO-9]+)\(\)"), "public:
306
                                          TEST_METHOD(\$3)", 0),
                                   // class TreesTests
307
                                   // TEST_CLASS(TreesTests)
                                   (new Regex(@"class ([a-zA-Z0-9]+Tests)"), "TEST_CLASS($1)", 0),
309
                                   // Assert.Equal
310
                                   // Assert::AreEqual
311
                                   (new Regex(@"(?<type>Assert)\.(?<method>(Not)?Equal)"), "${type}::Are${method}", 0),
312
                                   // Assert.Throws
313
                                   // Assert::ExpectException
314
                                   (new Regex(0"(Assert)\.Throws"), "$1::ExpectException", 0),
                                   // Assert.True
316
                                   // Assert::IsTrue
317
                                   (new Regex(@"(Assert)\.(True|False)"), "$1::Is$2", 0),
318
                                   // $"Argument {argumentName} is null."
319
                                   // std::string("Argument
320
                                           ").append(Platform::Converters::To<std::string>(argumentName)).append(" is
                                    \rightarrow null.")
                                   (new Regex(@"\$""(?<left>(\\""|[^""\r\n])*){(?<expression>[_a-zA-Z0-9]+)}(?<right>(\_
321
                                            \""<sup>[</sup>[^""\r\n])*)""")
                                           "std::string($\"${left}\").append(Platform::Converters::To<std::string>(${expres_
                                          sion})).append(\"${right}\")"
                                           10),
                                   // $"
322
                                   // "
                                   (new Regex(@"\$"""), "\"", 0)
324
                                   // std::string(std::string("[").append(Platform::Converters::To<std::string>(Minimum)
325
                                           )).append("
                                           ")).append(Platform::Converters::To<std::string>(Maximum)).append("]")
                                   // std::string("[").append(Platform::Converters::To<std::string>(Minimum)).append(",
326
                                           ").append(Platform::Converters::To<std::string>(Maximum)).append("]")
                                   (new Regex(@"std::string\((!"(\\""|[^""])*""\)(\.append\((Platf)))
                                           orm::Converters::To\langle std::string \rangle (([^) \n]+())([^) \n]+()) \rangle.append(),
                                           "${begin}.append", 10)
                                   // Console.WriteLine("...
328
                                   // printf("...\n")
329
                                   (new Regex(@"Console\.WriteLine\(""([^""\r\n]+)""\)"), "printf(\"$1\\n\")", 0),
330
                                       TElement Root;
                                   // TElement Root = 0;
332
                                   (new Regex(@"(?<before>\r?\n[\t ]+)(?<access>(private|protected|public)(:
333
                                           )?)?(?<type>[a-zA-Z0-9:_]+(?<!return)) (?<name>[_a-zA-Z0-9]+);"),
                                           "${before}${access}${type} ${name} = 0;", 0),
                                   // TreeElement _elements[N];
                                   // TreeElement _elements[N] = { {0} };
335
                                   (new Regex(@"(\r?\n[\t]+)(private|protected|public)?(: )?([a-zA-Z0-9]+)
336
                                           ([_a-zA-ZO-9]+)\setminus[([_a-zA-ZO-9]+)\setminus];"), "$1$2$3$4 $5[$6] = { {0} };", 0),
                                         auto path = new TElement[MaxPath];
337
                                   // TElement path[MaxPath] = { {0} };
                                   (\text{new Regex}(0^{-}(\r?\n[\t]+)[a-zA-Z0-9]+([a-zA-Z0-9]+) = \text{new})
339
                                           ([a-zA-Z0-9]+)\setminus[([_a-zA-Z0-9]+)\setminus];"), "$1$3 $2[$4] = { {0} };", 0),
                                   // bool Equals(Range<T> other) { ... }
340
                                   // bool operator ==(const Key &other) const { ...
341
                                   (new Regex(0"(?<before>\r?\n[^\n]+bool )Equals\((?<type>[^\n{]+)
                                             (?<\variable>[a-zA-Z0-9]+)\) (?<\after>(\s|\n)*{})"), "${before}\ operator == (constructions) ( | \color=0.5cm | \color=0.5
                                           $\{\type\} &\{\variable\}\) const\{\(\after\}\), 0),
                                   // Insert scope borders.
343
                                   // class Range { ... public: override std::string ToString() { return ...
344
                                   // class Range {/*~Range<T>~*/ ... public: override std::string ToString() { return
                                           . . . ;
                                   (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)template <typename</pre>
                                            (?<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).|\n)+?) (?<toStringDeclaration>(?<access>(private).|\n)+?) (?<toStringDeclaration>(?<access>(private).|\n)+?) (?<toStringDeclaration>(?<access>(private).|\n)+?) (?<toStringDeclaration>(?<access>(private).|\n)+?) (?<toStringDeclaration>(?<access>(private).|\n)+?) (?<access>(private).|\n)+?) (?<access>(privat
                                           |protected|public): )override std::string ToString\(\\))"),
                                           "${classDeclarationBegin}/*~${type}~*/${middle}${toStringDeclaration}", 0),
```

```
// Inside the scope of ~!Range!~ replace:
347
                 // public: override std::string ToString() { return ...
348
                 // public: operator std::string() const { return ...; }\n\npublic: friend
349
                    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>_
350
                     ((?<!/*^{\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\} &\text{obj} \{ \text{return out << (std::string)obj; }", 0),</pre>
                 // Remove scope borders.
351
                 // /*~Range~*/
352
                 //
                 (new Regex(0"/\*^[_a-zA-Z0-9<>:]+^\*/"), "", 0),
                 // private: inline static ConcurrentBag<std::exception> _exceptionsBag;
355
                 // private: inline static std::mutex _exceptionsBag_mutex; \n\n private: inline
356

    static std::vector<std::exception> _exceptionsBag;

                 (new Regex(@"(?<begin>\r?\n?(?<indent>[ \t]+))(?<access>(private|protected|public):
357
                    )?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}> ${name};", 0)
                 // public: static IReadOnlyCollection<std::exception> GetCollectedExceptions() {
                     return _exceptionsBag; }
                 // public: static std::vector<std::exception> GetCollectedExceptions() { return
359
                    std::vector<std::exception>(_exceptionsBag); }
                 (new Regex(0"(?<access>(private|protected|public): )?static
360
                     { return (?<fieldName>[_a-zA-Z0-9]+); }"),
                                                                  "${access}static
                    std::vector<${argumentType}> ${methodName}() { return
                    std::vector<${argumentType}>(${fieldName}); }", 0),
                 // public: static event EventHandler<std::exception> ExceptionIgnored =
361
                    OnExceptionIgnored; ... };
                    ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
362

→ const std::exception&)> ExceptionIgnored = OnExceptionIgnored; };

                 (new Regex(0"(?<br/>hegin>\r?\n(\r?\n)?(?<halfIndent>[
                     \t]+)\k<halfIndent>)(?<access>(private|protected|public): )?static event
                      EventHandler < (? < argumentType > [^; \\ r | +) > (? < name > [_a-zA-Z0-9] +) = (? < defaultDele_l) 
                     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),
                 // public: event Disposal OnDispose;
// public: Platform::Delegates::MulticastDelegate<Disposal> OnDispose;
364
                 (new Regex(@"(?<begin>(?<access>(private|protected|public): )?(static )?)event
366
                     (?<type>[a-zA-Z][:_a-zA-Z0-9]+) (?<name>[a-zA-Z][_a-zA-Z0-9]+);")
                    "${begin}Platform::Delegates::MulticastDelegate<${type}> ${name};",
                 // Insert scope borders.
367
                 // class IgnoredExceptions { ... private: inline static std::vector<std::exception>
                      _exceptionsBag;
                 // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: inline static
369
                    std::vector<std::exception> _exceptionsBag;
                 (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t]*)class [^{\r\n]+\r\n[\t
                     ]*{)(?<middle>((?!class).|\n)+?)(?<vectorFieldDeclaration>(?<access>(private|pro|
                     tected | public): )inline static std::vector<(?<argumentType>[^;\r\n]+)>
                     (?<fieldName>[_a-zA-Z0-9]+);)")
                     "${classDeclarationBegin}/*~${fieldName}~*/${middle}${vectorFieldDeclaration}",
                     0),
                 // Inside the scope of ~!_exceptionsBag!~ replace:
                 // _exceptionsBag.Add(exception);
372
                 // _exceptionsBag.push_back(exception);
373
                 (\text{new Regex}(@"(?<scope>/)*^(?<fieldName>[_a-zA-Z0-9]+)^**/)(?<separator>.|\n)(?<befor_1)()
374
                     e>((?<!/\*~\k<fieldName>~\*/)(.|\n))*?)\k<fieldName>\.Add"),
                     "${scope}${separator}${before}${fieldName}.push_back", 10),
                 // Remove scope borders.
375
                 // /*~_exceptionsBag~*/
                 //
377
                 (new Regex(0"/*[_a-zA-Z0-9]+*\*/"), "", 0),
378
                 // Insert scope borders.
379
                // class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
// class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: static std::mutex
380
381
                    _exceptionsBag_mutex;
```

```
(new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t]*)class [^{\r\n]+\r\n[\t
382
                               ]*{)(?<middle>((?!class).|\n)+?)(?<mutexDeclaration>private: inline static)}
                               std::mutex (?<fieldName>[_a-zA-Z0-9]+)_mutex;)"),
"${classDeclarationBegin}/*~${fieldName}~*/${mutexDeclaration}", 0),
                         // Inside the scope of ~!_exceptionsBag!~ replace:
383
                         // return std::vector<std::exception>(_exceptionsBag);
                         // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
385

    std::vector<std::exception>(_exceptionsBag);
                         (\text{new Regex}(@"(?<scope>//*^{(?<fieldName>[_a-zA-Z0-9]+)^*/})(?<separator>.|\n)(?<befor_left)
386
                                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);${after}", 10),
Inside the scope of ~!_exceptionsBag!~ replace:
387
                              _exceptionsBag.Add(exception);
                         // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
389
                                _exceptionsBag.Add(exception);
                          (new Regex(0"(?<scope>/\times~(?<fieldName>[_a-zA-Z0-9]+)~\times/)(?<separator>.|\setminusn)(?<befor
390
                                e>((?<!/*^k<fieldName>^**/)(.|n))*?){(?<after>((?!lock_guard)([^{};]|n))*?}r_1
                                n(?<indent>[ \t]*)\k<fieldName>[^;}\r\n]*;)")
                                "${scope}${separator}${before}{" + Environment.NewLine +
                                "${indent}std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", 10),
                         // Remove scope borders.
391
                         // /*~_exceptionsBag~*/
392
                         (new Regex(0"/\*^[_a-zA-Z0-9]+^*\*/"), "", 0),
                         // Insert scope borders.
395
                         // class IgnoredExceptions { ... public: static inline
396
                               Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
                               ExceptionIgnored = OnExceptionIgnored;
                         // class IgnoredExceptions {/*~ExceptionIgnored~*/ ... public: static inline
397
                                Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
                               ExceptionIgnored = OnExceptionIgnored;
                          (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n]+\r\n[\t
398
                               |public): )static inline
                               Platform::Delegates::MulticastDelegate<(?<argumentType>[^;\r\n]+)>
                                (?\langle name \rangle [_a-zA-ZO-9]+) = (?\langle defaultDelegate \rangle [_a-zA-ZO-9]+);)"),
                                "${classDeclarationBegin}/*~${name}~*/${middle}${eventDeclaration}", 0),
                         // Inside the scope of ~!ExceptionIgnored!~ replace:
                         // ExceptionIgnored.Invoke(NULL, exception);
                         // ExceptionIgnored(NULL, exception);
401
                         (\text{new Regex}(@"(?<scope>//*^(?<eventName>[a-zA-Z0-9]+)^/*/)(?<separator>.|\n)(?<before_|
402
                               >((?<!/*^k<eventName>^*/)(.|n))*?)k<eventName>|.Invoke||),
                               "${scope}${separator}${before}${eventName}", 10),
                         // Remove scope borders.
403
                         // /*~ExceptionIgnored~*/
404
                         (new Regex(0"/\*^[a-zA-Z0-9]+^\*/"), "", 0),
406
                         // Insert scope borders.
407
                             auto added = new StringBuilder();
                         // /*~sb~*/std::string added;
409
                         (new Regex(@"(auto|(System\.Text\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new
410
                                (System\.Text\.)?StringBuilder\(\);"), "/*~${variable}~*/std::string
                               ${variable}; ", 0),
                         // static void Indent(StringBuilder sb, int level)
411
                         // static void Indent(/*~sb~*/StringBuilder sb, int level)
                         (new Regex(@"(?<start>, |\()(System\.Text\.)?StringBuilder
413
                                (?<variable>[a-zA-Z0-9]+)(?<end>,|\))"), "${start}/*~${variable}~*/std::string&
                         $\ \square\ \text{\text{variable}\$\{\text{end}\}\", 0),}
// Inside the scope of \ \text{\text{!added!}\" replace:}
414
                         // sb.ToString()
415
                         // sb
416
                         (\text{new Regex}(@"(?<scope>/<math>*"(?<variable>[a-zA-Z0-9]+)")*/)(?<separator>.|\n)(?<before>|
417
                                ((? <!/*^k < variable > ^k/)(.|\n)) *?) \k < variable > \. To String \((\)"),
                                "${scope}${separator}${before}${variable}", 10),
                         // sb.AppendLine(argument)
418
                         // sb.append(Platform::Converters::To<std::string>(argument)).append(1, '\n')
                         (\text{new Regex}(@"(?<scope>//*^(?<variable>[a-zA-Z0-9]+)^*/*/)(?<separator>.|\n)(?<before>|)(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/*/)(?<separator>.|\n)(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/*/)(?<separator>.|\n)(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/*/)(?<separator>.|\n)(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/*/)(?<separator>.|\n)(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/*/)(?<separator>.|\n)(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/*/)(?<separator>.|\n)(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/*/)(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/*/)(?<separator>.|\n)(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/*/)(?<separator>.|\n)(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/*/)(?<separator>.|\n)(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/*/)(?<separator>.|\n)(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/*/)(?<separator>.|\n)(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/*/)(?<separator>.|\n)(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/*/)(?<separator>.|\n)(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/(?<scope>//*(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9]+)^*/(?<variable>[a-zA-Z0-9
420
                                ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.AppendLine\((?<argument>[^\),\<sub>|</sub>
                               r\n]+)\)"),
"${scope}${separator}${before}${variable}.append(Platform::Converters::To<std::s|
                               tring>(${argument})).append(1, '\\n')",
                               10),
                         // sb.Append('\t'
                                                     , level);
                         // sb.append(level, '\t');
422
```

```
(\text{new Regex}(@"(?<scope>/)*^(?<variable>[a-zA-Z0-9]+)^**/)(?<separator>.|\n)(?<before>|
423
                                  ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.Append\('(?<character>[^'\r\n]
                                        (?<count>[^{\}, r^{\}), r^{\})
                                 "${scope}${separator}${before}${variable}.append(${count}, '${character}')", 10),
                           // sb.Append(argument)
42.4
                           // sb.append(Platform::Converters::To<std::string>(argument))
425
                           (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>|
                                   ((? < !/* \land \texttt{k} < \texttt{variable} > `` +/) (. | \land n)) *?) \land \texttt{variable} \land \texttt{Append} \land ((? < \texttt{argument} > [^ \land) , \land r \land n] ) 
                                 +)\)"),
                                 "${scope}${separator}${before}${variable}.append(Platform::Converters::To<std::s
                                 tring>(${argument}))",
                                 10).
                           // Remove scope borders.
427
                           // /*~sb~*/
                           11
429
                           (\text{new Regex}(@"/\*^[a-zA-Z0-9]+^\*/"), "", 0),
430
                           // Insert scope borders.
                           // auto added = new HashSet<TElement>();
432
                           // ~!added!~std::unordered_set<TElement> added;
433
                           (new Regex(@"auto (?<variable>[a-zA-Z0-9]+)
434
                                 HashSet < (? < element > [a-zA-Z0-9] +) > ( ); "),
                                 "~!${variable}!~std::unordered_set<${element}> ${variable};", 0),
                           // Inside the scope of ~!added!~ replace:
435
                           // added.Add(node)
436
                           // added.insert(node)
437
                           (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?< |</pre>
438
                                  !~!\k<variable>!~)(.|\n))*?)\k<variable>\.Add\((?<argument>[a-zA-Z0-9]+)\)"),
                                 "${scope}${separator}${before}${variable}.insert(${argument})", 10),
                           // Inside the scope of ~!added!~ replace:
439
                           // added.Remove(node)
                           // added.erase(node)
441
                           (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?<|</pre>
442
                                  !~!\k<variable>!~)(.|\n))*?)\k<variable>\.Remove\((?<argument>[a-zA-Z0-9]+)\)"),
                                 "${scope}${separator}${before}${variable}.erase(${argument})", 10),
                           // if (added.insert(node)) {
                           // if (!added.contains(node)) { added.insert(node);
444
                           (\text{new Regex}(@"if \setminus ((?<\text{variable}=a-zA-Z0-9]+) \setminus (?<\text{argument}=a-zA-Z0-9]+) \setminus) (?_{argument}=a-zA-Z0-9]+))))
445
                                 \operatorname{separator}[\t] *[\r\n] +) (? \operatorname{sindent}[\t] *) {"}, "if
                                  (!${variable}.contains(${argument}))${separator}${indent}{" +
                                 Environment.NewLine + "${indent}
                                                                                              ${variable}.insert(${argument});", 0),
                           // Remove scope borders.
446
                           // ~!added!^
447
448
                           (new Regex(0"^{-1}[a-zA-Z0-9]+!^{-1}), "", 5),
449
                           // Insert scope borders.
                           // auto random = new System.Random(0);
451
                           // std::srand(0);
452
                           (\text{new Regex}(@"[a-zA-Z0-9]] + ([a-zA-Z0-9]] + ) = \text{new}
453
                                  (System\.)?Random\(([a-zA-Z0-9]+)\);"), "~!$1!~std::srand($3);", 0),
                           // Inside the scope of "!random!" replace:
                              random.Next(1, N)
455
                           // (std::rand() % N) + 1
456
                           (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?< |</pre>
                                  !^*[\k<\variable>!^*)(.\n))*?)\k<\variable>\.Next\((?<from>[a-zA-ZO-9]+)
                                  (?<to>[a-zA-Z0-9]+)\)"), "${scope}${separator}${before}(std::rand() % ${to}) + (?<to>[a-zA-Z0-9]+)\)"), "${scope}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${separator}${
                                 ${from}", 10),
                           // Remove scope borders.
458
                               ~!random!
                           //
                           //
460
                           (new Regex(0"^{-1}[a-zA-Z0-9]+!^{-1}), "", 5),
461
                           // Insert method body scope starts.
                           // void PrintNodes(TElement node, StringBuilder sb, int level) {
463
                           // void PrintNodes(TElement node, StringBuilder sb, int level) {/*method-start*/
464
                           (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*/${end}",
                                 0),
                           // Insert method body scope ends.
466
                           // {/*method-start*/...}
467
                           // {/*method-start*/.../*method-end*/}
468
                           (new\ Regex(@"\{/\*method-start\*/(?<body>((?<bracket>\{) | (?<-bracket>\{}) | [^\{\}]*)+)_{|}})
469
                                 \}"),
                                           "{/*method-start*/${body}/*method-end*/}",
                                 0)
                           // Inside method bodies replace:
```

```
// GetFirst(
                 // this->GetFirst(
                 (new
473
                     Regex(@"(?<scope>/\mbox{*method-start}*/)(?<before>((?<!/\mbox{*method-end}*/)(.|\n))*?)(?|
                     \ensuremath{$\langle (::|\.|->| throw\s+))(?(method>(?!sizeof)[a-zA-Z0-9]+)((?!\))$}
                     "${scope}${before}${separator}this->${method}(${after}${scopeEnd}", 100),
                 // Remove scope borders.
474
                 // /*method-start*/
                 //
476
                 (new Regex(0"/\timesmethod-(start|end)\times/"), "", 0),
477
                 // Insert scope borders.
478
                   const std::exception& ex
                 // const std::exception& ex/*~ex~*/
480
                 (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?exception&?
481
                     (?<variable>[_a-zA-Z0-9]+))(?<after>\W)")
                     "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
                 // Inside the scope of ~!ex!~ replace:
482
                 // ex.Message
483
                 // ex.what()
                 (new Regex(@"(?<scope>/\*~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before</pre>
485
                     >((?<!/*^k<variable>^k/)(.|n))*?)(Platform::Converters::To<std::string>\(\k<|)**
                     variable>\.Message\) | \k<variable>\.Message) ");
                    "${scope}${separator}${before}${variable}.what()", 10),
                 // Remove scope borders.
486
                 // /*~ex~*/
                 //
                 (new Regex(0"/\*^[_a-zA-Z0-9]+^*\*/"), "", 0),
489
                 // throw ObjectDisposedException(objectName, message);
490
                 // 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_]*);
492
                     (?<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
                     \hookrightarrow
                    0),
                 // throw ArgumentNullException(argumentName, message);
493
                 // throw std::invalid_argument(std::string("Argument
494
                    ").append(argumentName).append(" is null: ").append(message).append("."));
                 (new Regex(@"throw
                     ArgumentNullException ((?\langle argument \rangle [a-zA-Z] * [Aa] rgument [a-zA-Z] *),
                     (?\langle message\rangle[a-zA-Z]*[Mm]essage[a-zA-Z]*((())?));"), "throw
                    std::invalid_argument(std::string(\"Argument \").append(${argument}).append(\"
                    is null: \").append(${message}).append(\".\"));", 0),
                 // throw ArgumentException(message, argumentName);
496
                 // throw std::invalid_argument(std::string("Invalid ").append(argumentName).append("
497
                    argument: ").append(message).append("."));
                 (new Regex(@"throw
                     ArgumentException \setminus ((?<message>[a-zA-Z]*[Mm] essage[a-zA-Z]*(\setminus(\setminus))?),
                     (?\langle argument \rangle [a-zA-Z] * [Aa] rgument [a-zA-Z] *) \rangle;"), "thrown"
                     std::invalid_argument(std::string(\"Invalid \").append(${argument}).append(\"
                     argument: \").append(${message}).append(\".\"));", 0),
                 // throw ArgumentOutOfRangeException(argumentName, argumentValue, messageBuilder());
499
                 // throw std::invalid_argument(std::string("Value
500
                     [").append(Platform::Converters::To<std::string>(argumentValue)).append("] of
                     argument [").append(argumentName).append("] is out of range:
                     ").append(messageBuilder()).append("."));
                 (new Regex(@"throw ArgumentOutOfRangeException\((?<argument>[a-zA-Z]*[Aa]rgument[a-z]
                     A-Z] * ([Nn] ame [a-zA-Z] *)?)
                     (?\langle \texttt{argumentValue} \rangle [\texttt{a-zA-Z}] * [\texttt{Aa}] \texttt{rgument} [\texttt{a-zA-Z}] * ([\texttt{Vv}] \texttt{alue} [\texttt{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);
                 // throw NotSupportedException();
502
                 // throw std::logic_error("Not supported exception.");
503
                 (new Regex(@"throw NotSupportedException\(\);"), "throw std::logic_error(\"Not
                    supported exception.\");", 0);
                 // throw NotImplementedException();
505
                 // throw std::logic_error("Not implemented exception.");
506
                 (new Regex(@"throw NotImplementedException\(\);"), "throw std::logic_error(\"Not
507
                     implemented exception.\");", 0),
                 // Insert scope borders.
508
                 // const std::string& message
509
```

```
// const std::string& message/*~message~*/
510
                                   (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?((std::)?string&?|char\*)
                                            (?<variable>[_a-zA-Z0-9]+))(?<after>\W)")
                                            "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
                                   // Inside the scope of /*~message~*/ replace:
512
                                   // Platform::Converters::To<std::string>(message)
                                   // message
514
                                   (\text{new Regex}(@"(?<scope>/*"(?<variable>[_a-zA-Z0-9]+)"\*/)(?<separator>.|\n)(?<before_1)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?<scope)(?
515
                                           ((<!/*^k<variable^*/*)(.|n))*?
                                           ariable>\)"), "${scope}${separator}${before}${variable}",
                                           10),
                                   // Remove scope borders.
                                   // /*~ex~*/
517
                                   //
518
                                   (\text{new Regex}(@"/\*^[_a-zA-ZO-9]+^\*/"), "", 0),
519
                                   // Insert scope borders.
                                   // std::tuple<T, T> tuple
// std::tuple<T, T> tuple/*~tuple~*/
521
522
523
                                   (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?tuple<[^\n]+>&?
                                            (?<variable>[_a-zA-Z0-9]+))(?<after>\W)")
                                            "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
                                   // Inside the scope of ~!ex!~ replace:
                                   // tuple.Item1
                                   // std::get<1-1>(tuple)
526
                                   (\text{new Regex}(@"(?<scope>/)*^(?<variable>[_a-zA-Z0-9]+)^*)*(?<separator>.|\n)(?<before)
527
                                           >((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.Item(?<itemNumber>\d+)(?<afte_
                                           r>\W)")
                                            "${scope}${separator}${before}std::get<${itemNumber}-1>(${variable})${after}",
                                    \hookrightarrow
                                   // Remove scope borders.
528
                                   // /*~ex~*/
529
                                   //
                                   (new Regex(0"/\*^[_a-zA-Z0-9]+^*\*/"), "", 0),
531
                                   // Insert scope borders.
532
                                   // class Range<T>
                                   // class Range<T> {/*~type~Range<T>~*/
534
                                   (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t]*)(template\s*<[^<>\n]*>
535
                                           )?(struct|class)
                                            (?<fullType>(?<typeName>[a-zA-Z0-9]+)(<[^:\n]*>)?)(\s*:\s*[^{\n]+)?[\t
                                   | *(\r?\n)?[\t]*{)"),
| "${classDeclarationBegin}/*~type~${typeName}~${fullType}~*/", 0),
| Inside the scope of /*~type~Range<T>~*/ insert inner scope and replace:
                                   // public: static implicit operator std::tuple<T, T>(Range<T> range)
537
                                   // public: operator std::tuple<T, T>() const {/*~variable~Range<T>~*,
538
                                   (new Regex(@"(?<scope>/\*~type~(?<typeName>[^~\n\*]+)~(?<fullType>[^~\n\*]+)~\*/)(?<
                                            ?<access>(private|protected|public): )static implicit operator
                                            (?<targetType>[^\(\n]+)\((?<argumentDeclaration>\k<fullType>
                                            (?\langle variable \rangle [a-zA-Z0-9]+))))(?\langle after \rangle * n?\langle s*{})")
                                            "${scope}${separator}${before}${access}operator ${targetType}()
                                            const${after}/*~variable~${variable}~*/", 10),
                                   // Inside the scope of /*~type~Range<T>~*/ replace:
540
                                   // public: static implicit operator Range<T>(std::tuple<T, T> tuple) { return new
541
                                           Range<T>(std::get<1-1>(tuple), std::get<2-1>(tuple)); }
                                   // public: Range(std::tuple<T, T> tuple) : Range(std::get<1-1>(tuple),
542
                                           std::get<2-1>(tuple)) { }
                                   (new Regex(@"(?<scope>/\*~type~(?<typeName>[^~\n\*]+)~(?<fullType>[^~\n\*]+)~(*/)
543
                                            separator >. \ |\ ) \ (?<before > ((?<!/*~type^k<typeName > ^k<fullType > ^* +/) (. \ |\ )) *?) (|
                                            ?<access>(private|protected|public): )static implicit operator
                                            (new )?(k<fullType>|k<typeName>)((?<passedArguments>[^\n]+)\);(\s|\n)*}"),
                                            "${scope}${separator}${before}${access}${typeName}(${arguments}) :
                                            $\{\typeName\}(\$\{\passedArguments\}) \{\}\", 10),
                                   // Inside the scope of /*~variable~range~*/ replace:
545
                                   // range.Minimum
                                   // this->Minimum
546
                                   (new Regex(@"(?<scope>{/\*~variable~(?<variable>[^~\n]+)~\*/)(?<separator>.|\n)(?<be |</pre>
                                           fore>(?\langle beforeExpression>(?\langle bracket> \{) | (?\langle -bracket> \}) | [^{ }] | \n) *?) \\ \\ \langle constraint | constrai
                                             \begin{tabular}{ll} (?&field>[_a-zA-Z0-9]+) (?&after>(,|;||)| \\ ||)) (?&afterExpression>(?&bracket>||(?&-bracket>|)|[^{}||n)*?|)"), \\ \end{tabular} 
                                            "${scope}${separator}${before}this->${field}${after}", 10),
                                   // Remove scope borders.
548
                                   // /*~ex~*/
550
                                   (\text{new Regex}(@"/\*"[^"\n]+"[^"\n]+"\*/"), "", 0),
551
                                   // Insert scope borders.
```

```
// namespace Platform::Ranges {
553
               // namespace Platform::Ranges {/*~start~namespace~Platform::Ranges~*/ ...
                   /*~end~namespace~Platform::Ranges~*/}
               (new Regex(@"(?<namespaceDeclarationBegin>\r?\n(?<indent>[\t ]*)namespace
555
                   (?<name>name>(?<namePart>[a-zA-Z][a-zA-Z0-9]+)(?<nextNamePart>::[a-zA-Z][a-z]
                   nd~namespace~${namespaceName}~*/${end}",
               // Insert scope borders.
               // class Range<T> { ... };
557
               // class Range<T> {/*~start~type~Range<T>~T~*/ ... /*~start~type~Range<T>~T~*/};
558
               (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)template <typename</pre>
559
                   (?<typeParameter>[^\n]+)> (struct|class)
                   (?<type>[a-zA-Z0-9]+<k<typeParameter>>)(\s*:\s*[^{\n]+)?[\t]*(\r?\n)?[\t]
                   Indent}/*~end~type~${type}~${typeParameter}~*/${end}",
               // Inside scopes replace:
560
               // /*~start~namespace~Platform::Ranges~*/ ... /*~start~type~Range<T>~T~*/ ...
                   public: override std::int32_t GetHashCode() { return {Minimum,
                  Maximum}.GetHashCode(); } ... /*~start~type~Range<T>~T~*/ ...
                   /*~end~namespace~Platform::Ranges~*/
               // /*~start~namespace~Platform::Ranges~*/ ... /*~start~type~Range<T>~T~*/ ...
562
                   /*~start~type~Range<T>~T~*/ ... /*~end~namespace~Platform::Ranges~*/ namespace
                   std { template <typename T> struct hash<Platform::Ranges::Range<T>> {
                   std::size_t operator()(const Platform::Ranges::Range<T> &obj) const { return
               {Minimum, Maximum}.GetHashCode(); } }; }
(new Regex(@"(?<namespaceScopeStart>/\*~start~namespace~(?<namespace>[^~\n\*]+)~\*/)
                   (?<betweenStartScopes>(.|\n)+)(?<typeScopeStart>/\*~start~type~(?<type>[^~\n\*]+<sub>|</sub>
                   )^(?<typeParameter>[^{\n}*]+)^{*/}(?<before>(.|\n)+?)(?<hashMethodDeclaration>\r_1
                   ?\n[ \t]*(?<access>(private|protected|public): )override std::int32_t
                   )+?)(?<typeScopeEnd>/\*~end~type~\k<type>~\k<typeParameter>~\*/)(?<betweenEndSco
                   pes>(. | \n)+) (?\newpaceScopeEnd>/\*"end"namespace"\k<namespace>"\*/) \r")
                   "${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 + "
                   operator()(const ${namespace}::${type} &obj) const" + Environment.NewLine + "
                       {" + Environment.NewLine + "
                   /*~start~method~*/${methodBody}/*~end~method~*/" + Environment.NewLine + "
                    }" + Environment.NewLine + "
                                                   };" + Environment.NewLine + "}" +
                   Environment.NewLine, 10),
               // Inside scope of /*~start~method~*/ replace:
               // /*~start~method~*/ ... Minimum ... /*~end~method~*/
565
               // /*~start~method~*/ ... obj.Minimum ... /*~end~method~*/
566
               (new Regex(@"(?<methodScopeStart>/\*~start~method~\*/)(?<before>.+({|,
                   ))(<name>[a-zA-Z][a-zA-Z0-9]+)(<after>[^\\\.\(a-zA-Z0-9]((<!/\*~end~method~\*/|
                   ) [^n]) +) (?<methodScopeEnd>/\*~end~method~\*/)")
                   \verb| "\$\{methodScopeStart\}\$\{before\}obj.\$\{name\}\$\{after\}\$\{methodScopeEnd\}", 10), \\
               // Remove scope borders.
568
               // /*~start~type~Range<T>~*/
569
570
               (new Regex(0"/\*~[^~\*\n]+(~[^~\*\n]+)*~\*/"), "", 0),
           }.Cast<ISubstitutionRule>().ToList();
572
           public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
574
               // ICounter<int, int> c1;
576
               // ICounter<int, int>* c1;
577
               (new Regex(0"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^>\r\n]+>)?)
                   (?<variable>[_a-zA-Z0-9]+)(?<after> = null)?;"), "${abstractType}*
                   ${variable}${after};", 0),
               // (expression)
579
               // expression
580
               (\text{new Regex}(@"((| )(([a-zA-Z0-9_{*:}]+)))(,| |;|))"), "$1$2$3", 0),
581
               // (method(expression))
               // method(expression)
583
```

```
(new Regex(0"(?<firstSeparator>(\( \) |
584
                                                                       ))\((?<method>[a-zA-Z0-9_\->\*:]+)\((?<expression>((?<parenthesis>\()|(?<-parent
                                                                       |;|\)))")
                                                                                                                "${firstSeparator}${method}(${expression})${lastSeparator}", 0),
                                                                    .append(".")
                                                          // .append(1,
                                                                                                           '.');
586
                                                          (new Regex(0"\.append\(""([^\\""]|\\[^""])""\)", ".append(1, '$1')", 0),
587
                                                         // return ref _elements[node];
588
                                                          // return &_elements[node];
                                                          (new Regex(@"return ref ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9\*]+)\];"), "return &$1[$2];",
590
                                                                      0),
                                                         // ((1, 2))
// ({1, 2})
591
                                                          (new Regex(0"(?<before>\(|, )\((?<first>[^\n()]+),
593
                                                                         (?\langle second \rangle [^n()] +) (?\langle after \rangle) |, )"), "$\{before\} {\{first\}, \}
                                                                        ${second}}${after}", 10),
                                                         // {1, 2}.GetHashCode()
                                                         // Platform::Hashing::Hash(1,
                                                                                                                                                                   2)
595
                                                         (new Regex(@"{(?<first>[^\n{}]+), (?<second>[^\n{}]+)}\.GetHashCode\(\)"),
596
                                                                        "Platform::Hashing::Hash(${first}, ${second})", 10),
                                                         // range.ToString()
597
                                                          // Platform::Converters::To<std::string>(range).data()
                                                          (new Regex(@"(?<before>\W)(?<variable>[_a-zA-Z][_a-zA-Z0-9]+)\.ToString\(\)"),
599
                                                                        "${before}Platform::Converters::To<std::string>(${variable}).data()", 10),
                                                         // new
600
                                                         //
601
                                                          s+"), "${before}",
                                                                      10),
                                                          // x == null
                                                         // x == nullptr
604
                                                          (\text{new Regex}(@"(?\before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(?<=\W)(
605
                                                                       ariable > [_a-zA-Z] [_a-zA-Z0-9] +) (? < perator > x < (== | !=) x ) null (? < after > \\ \\ \) "),
                                                                       "${before}${variable}${operator}nullptr${after}", 10),
                                                         // null
606
                                                          // {}
607
                                                           (\text{new Regex}(@"(?<\text{before}\r?\n[^""\r\n]*(""(\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W) \\ \text{null}_{\parallel}(\text{new Regex}(@"(?<\text{before}\r)))*"(?<=\W)_{\parallel}(\text{new Regex}(@"(?<\text{before}\r)))*"(?<=\W)_{\parallel}(\text{new Regex}(@"(?<\text{before}\r)))*"(?<=\W)_{\parallel}(\text{new Regex}(@"(?<\text{before}\r)))*"(?<=\W)_{\parallel}(\text{new Regex}(@"(?<\text{before}\r)))*"(?<=\W)_{\parallel}(\text{new Regex}(@"(?<\text{before}\r)))*"(?<=\W)_{\parallel}(\text{new Regex}(@"(?<\text{before}\r)))*"(?<=\W)_{\parallel}(\text{new Regex}(@"(?<\text{before}\r)))*"(?<=\W)_{\parallel}(\text{new Regex}(@"(?<\text{helps:}\w)))*"(?<=\W)_{\parallel}(\text{new Regex}
608
                                                                        (?<after>\W)"), "${before}{}${after}",
                                                                       10)
                                                         // default
609
610
                                                           (\text{new Regex}(@"(?<\text{before}\r?\n[^""\r\n]*(""(\""|[^""\r\n])*""[^""\r\n]*)*) (?<=\W) \\ \text{defa}_{\text{local}}(\text{local}_{\text{local}}(\text{local}_{\text{local}}(\text{local}_{\text{local}}(\text{local}_{\text{local}}(\text{local}_{\text{local}}(\text{local}_{\text{local}}(\text{local}_{\text{local}}(\text{local}_{\text{local}}(\text{local}_{\text{local}}(\text{local}_{\text{local}}(\text{local}_{\text{local}}(\text{local}_{\text{local}}(\text{local}_{\text{local}}(\text{local}(\text{local}_{\text{local}}(\text{local}(\text{local}_{\text{local}}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local}(\text{local
611
                                                                      ult(?<after>\W)"), "${before}0${after}",
                                                                       10)
                                                         // object x
612
                                                          // void *x
                                                          (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(?<! |</pre>
614
                                                                       @)(object|System\.Object) (?<after>\w)"), "${before}void *${after}",
                                                                       10),
                                                         // <object>
615
                                                         // <void*>
616
                                                           (\text{new Regex}(@"(?<\text{before}\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(?<!_{|} ) ) \\
                                                                       @) (object|System\.Object) (?<after>\W)"), "${before}void*${after}",
                                                                       10),
                                                         // @object
618
                                                         // object
619
                                                          (new Regex(0"0([_a-zA-Z0-9]+)"), "$1", 0),
620
                                                         // this->GetType().Name
621
                                                         // typeid(this).name()
622
                                                          (new Regex(@"(this)->GetType\(\)\.Name"), "typeid($1).name()", 0),
623
                                                         // ArgumentNullException
                                                         // std::invalid_argument
625
                                                          (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(Sys
626
                                                                        tem\.)?ArgumentNullException(?<after>\W)");
                                                                        "${before}std::invalid_argument${after}", 10),
                                                         // InvalidOperationException
627
                                                         // std::runtime_error
628
                                                          (new Regex(@"(\W)(InvalidOperationException|Exception)(\W)"),
629
                                                                       "$1std::runtime_error$3", 0),
                                                         // ArgumentException
630
                                                         // std::invalid_argument
                                                         (new Regex(@"(\W)(ArgumentException|ArgumentOutOfRangeException)(\W)"),
632
                                                                        "$1std::invalid_argument$3", 0),
                                                         // template <typename T> struct Range : IEquatable<Range<T>>
633
                                                         // template <typename T> struct Range {
```

```
(new Regex(@"(?<before>template <typename (?<typeParameter>[^\n]+)> (struct|class)
635
                                                             (?<type>[a-zA-Z0-9]+<[^\n]+>)) : (public)
                                                            // public: delegate void Disposal(bool manual, bool wasDisposed);
                                                 // public: delegate void Disposal(bool, bool);
637
                                                 (new Regex(@"(?<before>(?<access>(private|protected|public): )delegate
638
                                                             (?\langle returnType\rangle[a-zA-Z][a-zA-Z0-9:]+)
                                                             (?< delegate > [a-zA-Z][a-zA-Z0-9]+) \setminus (((?< leftArgumentType > [a-zA-Z][a-zA-Z0-9:]+),
                                                            )*)(?<argumentType>[a-zA-Z][a-zA-Z0-9:]+)
                                                             (?\langle argumentName \rangle [a-zA-Z] [a-zA-Z0-9] +) (?\langle after \rangle (, 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);
639
                                                 // using Disposal = void(bool, bool);
640
                                                 (new Regex(@"(?<access>(private|protected|public): )delegate
                                                               (?<returnType>[a-zA-Z][a-zA-Z0-9:]+)
                                                             (?\langle elegate = [a-zA-Z] [a-zA-Z0-9] +) ((?\langle elegate = [a-zA-Z0-9] +)
                                                  \hookrightarrow
                                                            ${delegate} = ${returnType}(${argumentTypes});", 20),
                                                 // #region Always
642
                                                 //
643
                                                 (\text{new Regex}(@"(^|\r?\n)[ \t]*\#(\text{region}|\text{endregion})[^\r\n]*(\r?\n|\$)"), "", 0),
644
                                                 // //#define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
646
                                                 (\text{new Regex}(@")//[ \t]*\define[ \t]+[_a-zA-Z0-9]+[ \t]*"), "", 0),
647
                                                 // #if USEARRAYPOOL\r\n#endif
649
                                                 (new Regex(0"#if [a-zA-Z0-9]+\s+\#endif"), "", 0),
650
                                                 // [Fact]
651
652
                                                 (new Regex(@"(?<firstNewLine>\r?\n|\A)(?<indent>[\t
653
                                                            ]+)\[[a-zA-Z0-9]+(\((?<expression>((?<parenthesis>\()|(?<-parenthesis>\))|[^()\r_{\perp}
                                                             \n]*)+)(?(parenthesis)(?!)))))?][ \t]*(\r?\n\k<indent>)?"),
                                                            "${firstNewLine}${indent}", 5),
                                                 // \A \n ... namespace
654
                                                 // \Anamespace
655
                                                 (new Regex(Q''(A)(r?n)+namespace"), "$1namespace", 0),
656
                                                        \A \n ... class
657
                                                 // \Aclass
658
                                                 (new Regex(0"(\A)(\r?\n)+class"), "$1class", 0),
659
                                                // \n\n\n
// \n\n
660
661
                                                 (new Regex(0"\r?\n[\t]*\r?\n[\t]. Environment.NewLine +
662
                                                            Environment.NewLine, 50),
                                                 // {\n\n
                                                 // {\n
664
                                                 (\text{new Regex}(@"{[ \t]*\r?\n"}, "{" + Environment.NewLine, 10}),
665
                                                 // \n n
                                                 // \n}
667
                                                 (\text{new Regex}(@"\r?\n[\t]*\r?\n(?<\text{end}[\t]*)"), \ \text{Environment.NewLine} \ + \ "\$\{\text{end}\}", \ 10),
668
                                     }.Cast<ISubstitutionRule>().ToList();
670
                                    public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
671
                                      → base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
672
                                    public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
673
                        }
674
675
                ./csharp/Platform.Regular Expressions. Transformer. CSharp To Cpp. Tests/CSharp To Cpp Transformer Tests. cs. \\
  1.2
            using Xunit;
    2
            namespace Platform.RegularExpressions.Transformer.CSharpToCpp.Tests
    4
                        public class CSharpToCppTransformerTests
    5
    6
                                     [Fact]
                                    public void EmptyLineTest()
                                                 // This test can help to test basic problems with regular expressions like incorrect
  10
                                                            syntax
                                                var transformer = new CSharpToCppTransformer();
  11
                                                 var actualResult = transformer.Transform("");
  12
                                                 Assert.Equal("", actualResult);
                                    }
  14
  15
```

[Fact]

```
public void HelloWorldTest()
17
18
                  const string helloWorldCode = @"using System;
19
    class Program
^{20}
21
        public static void Main(string[] args)
22
23
             Console.WriteLine(""Hello, world!"");
^{24}
25
    }";
26
                  const string expectedResult = @"class Program
27
28
    {
        public: static void Main(std::string args[])
29
30
             printf(""Hello, world!\n"");
31
32
    };";
33
                  var transformer = new CSharpToCppTransformer();
var actualResult = transformer.Transform(helloWorldCode);
34
35
                  Assert.Equal(expectedResult, actualResult);
36
             }
37
        }
38
    }
39
```

Index

./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs, 15 ./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1