```
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_]+^\x'), "", 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}~*/$0", 0),
                         // Move all markers to the beginning of the file.
                         (\text{new Regex}(@''\A(?<\text{before})^r\n] + r?\n(.|\n) +) (?<\text{marker}/\*`extensionMethod}^*(?<\text{name}) + r?\n(.|\n) +) (?<\text{marker}/\*`extensionMethod}^*(?<\text{name}) + r?\n(.|\n) +) (?<\text{marker}/\*`extensionMethod}^*(?<\text{name}) + r?\n(.|\n) +) (?<\text{marker}/\*`extensionMethod}^*(?<\text{name}) + r?\n(.|\n) +) (?<\text{marker}/\n) + r?\n(.|\n) + r?\n(.|\n) +) (?<\text{marker}/\n) + r?\n(.|\n) + r?\n(.|\n) +) (?<\text{marker}/\n) + r?\n(.|\n) + r?\n(.
92
                                [a-zA-Z0-9]+)^*/", "${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(@"\(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-ZO-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-Z][a-zA-Z0-9_]*) (?\block0pen>[\n\s]*{[\n\s]*}) (\property>[\n\s]*{[\n\s]*}) (\
                              ]*)?get;(?<blockClose>[\n\s]*))(?<after>[\n\s]*)"), "${before}${access}const
                              ${type} ${property};${after}", 2),
                        // public: bool IsDisposed { get => _disposed > 0; }
// public: bool IsDisposed() { return _disposed > 0; }
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
                 (new Regex(0"Action(<(?<typeParameters>[a-zA-Z0-9]+(,
180
                      ([a-zA-Z0-9]+))*))?(?\langle after >> | (?\langle variable > [a-zA-Z0-9]+))"),
                     "std::function<void(${typeParameters})>${after}", 0),
                 // Predicate<TArgument> predicate
181
                 // std::function<bool(TArgument)> predicate
182
                 (new Regex(0"Predicate<((\bar{a}-zA-Z0-9]+)> ((\bar{a}-zA-Z0-9]+)"), "std::function<br/>bool($1)>
183
                 // var
                 // auto
185
                 (new Regex(@"(\W)var(\W)"), "$1auto$2", 0),
186
                 // unchecked
187
                 (new Regex(@"[\r\n]{2}\s*?unchecked\s*?$"), "", 0),
189
                 // throw new
190
                 // throw
                 (new Regex(@"(\W)throw new(\W)"), "$1throw$2", 0),
192
                 // void RaiseExceptionIgnoredEvent(Exception exception)
193
                 // void RaiseExceptionIgnoredEvent(const std::exception& exception)
194
                 (new Regex(@"(\(|, )(System\.Exception|Exception)( |\))"), "$1const
                     std::exception&$3", 0),
                 // EventHandler<Exception>
196
                 // EventHandler<std::exception>
197
                 (new Regex(@"(\W)(System\.Exception|Exception)(\W)"), "$1std::exception$3", 0),
198
                 // override void PrintNode(TElement node, StringBuilder sb, int level)
199
                 // void PrintNode(TElement node, StringBuilder sb, int level) override
200
                 (\text{new Regex}(@"override}([a-zA-Z0-9 *++]+)(([^\)r\n]+?\))"), "$1$2 override", 0),
201
                 // return (range.Minimum, range.Maximum)
                 // return {range.Minimum, range.Maximum}
203
                 (\text{new Regex}(@"(?<\text{before}>\text{return}\s*)\((?<\text{values})\n]+)\)(?!\()(?<\text{after}\w)"),
204
                     "${before}{${values}}${after}", 0),
                 // string
205
                 // std::string
                 (new Regex(0"(?<before>\W)(?<!::)string(?<after>\W)"),
207
                      "${before}std::string${after}", 0),
                 // System.ValueTuple
208
                 // std::tuple
                 (new Regex(@"(?<before>\W)(System\.)?ValueTuple(?!\s*=|\()(?<after>\W)"),
210
                      "${before}std::tuple${after}", 0),
                 // sbyte
211
                 // std::int8_t
212
                 (new Regex(0"(?<before>\W)((System\.)?SB|sb)yte(?!\s*=|\()(?<after>\W)"),
                     "${before}std::int8_t${after}", 0),
                 // short
214
                 // std::int16_t
215
                 (new Regex(@"(?<before>\W)((System\.)?Int16|short)(?!\s*=|\()(?<after>\W)"),
216
                     "${before}std::int16_t${after}", 0),
                 // int
217
                 // std::int32_t
218
                 (\text{new Regex}(0"(?<bfore>\W)((System\.)?I|i)nt(32)?(?!\s*=|\()(?<after>\W)"),
219
                     "${before}std::int32_t${after}", 0),
                 // long
220
                 // std::int64_t
221
                 (new Regex(@"(?<before>\W)((System\.)?Int64|long)(?!\s*=|\()(?<after>\W)"),
222
                     "${before}std::int64_t${after}", 0),
                 // byte
223
                 // std::uint8_t
                 (new Regex(@"(?<before>\W)((System\.)?Byte|byte)(?!\s*=|\()(?<after>\W)"),
225
                     "${before}std::uint8_t${after}", 0),
                 // ushort
226
                 // std::uint16_t
227
                 (new Regex(@"(?<before>\W)((System\.)?UInt16|ushort)(?!\s*=|\()(?<after>\W)"),
                     "${before}std::uint16_t${after}", 0),
                 // uint
229
                 // std::uint32 t
230
                 (new Regex(@"(?<before>\W)((System\.)?UI|ui)nt(32)?(?!\s*=|\()(?<after>\W)"),
231
                     "${before}std::uint32_t${after}", 0),
                 // ulong
                 // std::uint64_t
233
                 (new Regex(@"(?<before>\W)((System\.)?UInt64|ulong)(?!\s*=|\()(?<after>\W)"),
234
                     "${before}std::uint64_t${after}", 0),
                 // char*[] args
235
                 // char* args[]
                 (\text{new Regex}(\bar{\mathbb{Q}}"([_a-zA-ZO-9:\*]?)\[\] ([a-zA-ZO-9]+)"), "$1 $2[]", 0),
237
                 // float.MinValue
238
```

```
// std::numeric_limits<float>::lowest()
239
                                                           (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MinValue(?<after>\W|
                                                                        )"), "${before}std::numeric_limits<${type}>::lowest()${after}",
                                                                        0).
                                                          // double.MaxValue
241
                                                          // std::numeric limits<float>::max()
242
                                                           (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MaxValue(?<after>\W|
                                                                      )"), "${before}std::numeric_limits<${type}>::max()${after}",
                                                                      0),
                                                          // using Platform.Numbers;
                                                          //
245
                                                           (new Regex(0"([\r\n]{2}|^)\s*?using [\.a-zA-Z0-9]+;\s*?$"), "", 0),
246
                                                          // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
247
                                                          // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
248
                                                          (new Regex(0"(struct|class) ([a-zA-Z0-9]+)(<[a-zA-Z0-9,]+>)? : ([a-zA-Z0-9]+)"),
249
                                                                       "$1 $2$3 : public $4", 0),
                                                          // System.IDisposable
250
                                                          // System::IDisposable
                                                          (\text{new Regex}(@"(?\before>System(::[a-zA-Z_]\w*)*).(?\after>[a-zA-Z_]\w*)"),
252
                                                                         "${before}::${after}", 20)
                                                          // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
253
                                                          // class IProperty : public ISetter<TValue, TObject>, public IProvider<TValue,
254
                                                                       TObject>
                                                           (\texttt{new} \ \texttt{Regex}(\texttt{0"}(?<\texttt{before}>(\texttt{interface}|\texttt{struct}|\texttt{class}) \ [\texttt{a-zA-Z}] \\ \texttt{w*} : ((\texttt{publice})) \\ \texttt{mew} \ \texttt{med} \\ \texttt{med} \ \texttt{med} \\ \texttt{med}
255
                                                                          [a-zA-Z_{-}][\w:]*(<[a-zA-Z0-9],]+>)?,
                                                                        )+)?)(?<inheritedType>(?!public)[a-zA-Z_][\w:]*(<[a-zA-Z0-9 ,]+>)?)(?<after>(,
                                                                       [a-zA-Z_][\w:]*(?!>)|[ \r\n]+))", "${before}public ${inheritedType}${after}",
                                                                       10),
                                                          // interface IDisposable {
256
                                                          // class IDisposable { public:
257
                                                           (new Regex(@"(?<before>\r?\n)(?<indent>[ \t]*)interface
                                                                          (?<interface>[a-zA-Z_]\w*)(?<typeDefinitionEnding>[^{]+){")
                                                                         "${before}${indent}class ${interface}${typeDefinitionEnding}{" +
                                                            \hookrightarrow
                                                                       Environment.NewLine + "
                                                                                                                                                                          public:", 0),
                                                          // struct TreeElement {
259
                                                          // struct TreeElement { };
260
                                                           (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([^;])"), "$1
                                                             \rightarrow $2$3{$4};$5", 0),
                                                          // class Program {
262
                                                          // class Program { };
263
                                                           (new Regex(@"(?<type>struct|class)
264
                                                                          (?\langle name \rangle [a-zA-Z0-9]+[^\r\n]*)(?\langle beforeBody \rangle [\r\n]+(?\langle indentLevel \rangle [\t
                                                                         ]*)?)\{(?<body>[\S\s]+?[\r\n]+\k<indentLevel>)\}(?<afterBody>[^;]|$)"), "${type}
                                                                         $\{\text{name}\$\{\text{beforeBody}\{\$\{\text{body}\}\};\$\{\text{afterBody}\", 0),
                                                           // Insert scope borders.
                                                          // ref TElement root
266
                                                          // ~!root!~ref TElement root
267
                                                           (\text{new Regex}(@"(?<\text{definition}>(?<= |\()(\text{ref }[a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!\text{ref})))))
                                                                         (?\langle variable \rangle [a-zA-Z0-9]+)(?= \rangle |, | = ))"), "^! {\{variable\}!^{\{definition\}}", 0\},
                                                          // Inside the scope of ~!root!~ replace:
269
                                                          // root
270
                                                           // *root
271
                                                           (\text{new Regex}(@"(?<\text{definition}^{"}(?<\text{pointer}[a-zA-Z0-9]+))" [a-zA-Z0-9]+)
                                                                         \k<pointer>(?=\)|, | =))(?<before>((?<!~!\k<pointer>!~)(.|\n))*?)(?<prefix>(\W
                                                                          | \langle () \rangle = (?\langle () \rangle () | \langle () \rangle | \langle (
                                                                        "${definition}${before}${prefix}*${pointer}${suffix}", 70),
                                                          // Remove scope borders.
273
                                                          // ~!root!~
                                                          //
                                                          (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", 5),
276
                                                          // ref auto root = ref
277
278
                                                               / ref auto root =
                                                           (new Regex(0"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)"), "$1* $2 =$3", 0),
279
                                                          // *root = ref left;
280
                                                          // root = left;
281
                                                           (\text{new Regex}(0"\*([a-zA-Z0-9]+) = \text{ref}([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", 0),
283
                                                          // (ref left)
                                                          // (left)
284
                                                           (new Regex(0"\(ref ([a-zA-Z0-9]+)(\)|\(|,)"), "($1$2", 0),
285
                                                                        ref TElement
286
                                                          // TElement*
287
                                                           (new Regex(0"(|\cdot|)ref ([a-zA-Z0-9]+)"), "$1$2*", 0),
288
                                                          // ref sizeBalancedTree.Root
289
                                                          // &sizeBalancedTree->Root
290
                                                           (\text{new Regex}(@"\text{ref}([a-zA-Z0-9]+)).([a-zA-Z0-9]*]+)"), "&$1->$2", 0),
291
                                                          // ref GetElement(node).Right
```

```
// &GetElement(node)->Right
293
                          (new Regex(0"ref ([a-zA-\bar{Z}0-9]+)\(([a-zA-\bar{Z}0-9\*]+)\)\.([a-zA-\bar{Z}0-9]+)"),
                                "&$1($2)->$3",
                          // GetElement(node).Right
295
                          // GetElement(node)->Right
296
                          (\text{new Regex}(@"([a-zA-Z0-9]+))(([a-zA-Z0-9]*)+))).([a-zA-Z0-9]+)"), "$1($2)->$3", 0),
297
                          // [Fact]\npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
                          // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
299
                          (\text{new Regex}(@'\[Fact\] [\s\n] + (\text{public}: )?(\text{static})?\text{void}([a-zA-ZO-9]+)\(\)"), "public: )
300
                                TEST_METHOD(\$3)", 0),
                          // class TreesTests
301
                          // TEST_CLASS(TreesTests)
302
                          (new Regex(0"class ([a-zA-ZO-9]+Tests)"), "TEST_CLASS(\$1)", 0),
303
                          // Assert.Equal
304
                          // Assert::AreEqual
                          (new Regex(@"(?<type>Assert)\.(?<method>(Not)?Equal)"), "${type}::Are${method}", 0),
306
                          // Assert.Throws
307
                          // Assert::ExpectException
308
                          (new Regex(@"(Assert)\\.Throws"), "$1::ExpectException", 0),
309
                          // Assert.True
310
                          // Assert::IsTrue
311
                          (new Regex(@"(Assert)\.(True|False)"), "$1::Is$2", 0),
                          // $"Argument {argumentName} is null.
313
                          // std::string("Argument
314
                                ").append(Platform::Converters::To<std::string>(argumentName)).append(" is
                               null.")
                           (\text{new Regex}(@"\s""(?<\text{left}>(\""|[^""\r\n])*){(?<\text{expression}>[_a-zA-Z0-9]+)}(?<\text{right}>(\|\|\|) ) ) 
315
                                 \""|[^""\r\n])*)""")
                                "std::string(\$)" \$ \{ left \} \setminus ") \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{joint} \} ) \ . append(Platform::Converters::To < std::string > (\$ \{ expres_{join
                                sion})).append(\"${right}\")",
                           \hookrightarrow
                                10),
                          // $"
316
                          // "
                          (new Regex(@"\$"""), "\"", 0)
318
                          // std::string(std::string("[").append(Platform::Converters::To<std::string>(Minimum)
319
                                )).append("
                                ")).append(Platform::Converters::To<std::string>(Maximum)).append("]")
                          // std::string("[").append(Platform::Converters::To<std::string>(Minimum)).append(",
                                ").append(Platform::Converters::To<std::string>(Maximum)).append("]")
                          321
                                orm::Converters::To<std::string>([^)\n]+()|[^)\n]+()))+)().append"),
                          "${begin}.append", 10),
// Console.WriteLine("...")
                                "${begin}.append"
322
                          // printf("...\n")
323
                          (new Regex(@"Console\.WriteLine\(""([^""\r\n]+)""\)"), "printf(\"$1\\n\")", 0),
                          // TElement Root;
325
                          // TElement Root = 0;
326
                          (new Regex(0"(?<before>\r?\n[\t ]+)(?<access>(private|protected|public)(:
327
                                )?)?(?<type>[a-zA-Z0-9:_]+(?<!return)) (?<name>[_a-zA-Z0-9]+);"),
                                "${before}${access}${type} ${name} = 0;", 0),
                          // TreeElement _elements[N]
328
                          // TreeElement _elements[N] = { {0} };
329
                          (new Regex(@"(\r?\n[\t]+)(private|protected|public)?(: )?([a-zA-Z0-9]+)
330
                                ([_a-zA-ZO-9]+)\setminus[([_a-zA-ZO-9]+)\setminus];"), "$1$2$3$4 $5[$6] = { (0} };", 0),
                          // auto path = new TElement[MaxPath];
                          // TElement path[MaxPath] = { {0} };
332
                          (\text{new Regex}(@"(\r?\n[\t]+)[a-zA-ZO-9]+ ([a-zA-ZO-9]+) = \text{new})
333
                                // bool Equals(Range<T> other) { ... }
334
                          // bool operator ==(const Key &other) const {
                          (new Regex(0"(?<before>\r?\n[^\n]+bool )Equals\((?<type>[^\n{]+)
336
                                (?\langle variable \rangle [a-zA-Z0-9]+) \rangle (?\langle after \rangle (\s|\n) *{})"), "${before} operator == (const.)
                                $\{\type\} &\{\text{variable}\}\) const\{\text{after}\", 0),
                          // Insert scope borders
337
                          // class Range { ... public: override std::string ToString() { return ...;
                          // class Range {/*~Range<T>~*/ ... public: override std::string ToString() { return
                                ...; }
                          (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)template <typename</pre>
340
                                 (?<typeParameter>[^<>\n]+)> (struct|class)
                                 ]*{)(?<middle>((?!class|struct).|\n)+?)(?<toStringDeclaration>(?<access>(private)
                                |protected|public): )override std::string ToString\(\\))"),
                                "${classDeclarationBegin}/*~${type}~*/${middle}${toStringDeclaration}", 0),
                          // Inside the scope of "!Range!" replace:
341
                          // public: override std::string ToString() { return ...; }
```

```
// public: operator std::string() const { return ...; }\n\npublic: friend
343
                   std::ostream & operator <<(std::ostream &out, const A &obj) { return out <<
                    (std::string)obj; }
                (new Regex(@"(?<scope>/\*~(?<type>[_a-zA-Z0-9<>:]+)~\*/)(?<separator>.|\n)(?<before>_
                    ((?<!/\*~\k<type>~\*/)(.|\n))*?)(?<toStringDeclaration>\r?\n(?<indent>[
                    \t]*)(?<access>(private|protected|public): )override std::string ToString\(\)
                    Environment.NewLine + "${indent}${access}operator std::string() const
                    $\{\toStringMethodBody\}\" + Environment.NewLine + Environment.NewLine +
                    "${indent}${access}friend std::ostream & operator <<(std::ostream &out, const
                    $\{\text{type}\} &\text{obj} \{ \text{return out << (std::string)obj; }", 0),</pre>
                // Remove scope borders.
345
                // /*~Range~*/
                //
347
                (new Regex(0"/\*~[_a-zA-Z0-9<>:]+~\*/"), "", 0),
// private: inline static ConcurrentBag<std::exception> _exceptionsBag;
348
349
                // private: inline static std::mutex _exceptionsBag_mutex; \n\n private: inline
350
                   static std::vector<std::exception> _exceptionsBag;
                (new Regex(@"(?<begin>\r?\n?(?<indent>[ \t]+))(?<access>(private|protected|public):
351
                    )?inline static ConcurrentBag<(?<argumentType>[^;\r\n]+)>
                    (?<name>[_a-zA-Z0-9]+);"), "${begin}private: inline static std::mutex
                    ${name}_mutex;" + Environment.NewLine + Environment.NewLine -
                    "${indent}${access}inline static std::vector<${argumentType}> ${name};", 0)
                // public: static IReadOnlyCollection<std::exception> GetCollectedExceptions() {
352
                    return _exceptionsBag; }
                // public: static std::vector<std::exception> GetCollectedExceptions() { return
                   std::vector<std::exception>(_exceptionsBag); }
                (new Regex(@"(?<access>(private|protected|public): )?static
                    std::vector<${argumentType}> ${methodName}() { return
                    std::vector<${argumentType}>(${fieldName}); }", 0),
                // public: static event EventHandler<std::exception> ExceptionIgnored =
                    OnExceptionIgnored; ... };
                // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
356
                const std::exception&)> ExceptionIgnored = OnExceptionIgnored; };
                (new Regex(@"(?<begin>\r?\n(\r?\n)?(?<halfIndent>[
357
                    \t]+)\k<halfIndent>)(?<access>(private|protected|public): )?static event
                    gate = [a-zA-Z0-9]+; (?\langle (.|\n)+?) (?\langle (x-y)-x\rangle (x-y)-x\rangle (x-y)-x
                    "${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;
359
                (new Regex(@"(?<begin>(?<access>(private|protected|public): )?(static )?)event
360
                    (?<type>[a-zA-Z][:_a-zA-Z0-9]+) (?<name>[a-zA-Z][_a-zA-Z0-9]+);"),
                    "${begin}Platform::Delegates::MulticastDelegate<${type}> ${name};", 0),
                // Insert scope borders.
                // class IgnoredExceptions { ... private: inline static std::vector<std::exception>
362
                     exceptionsBag;
                // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: inline static

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

                (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t]*)class [^{\r\n]+\r\n[\t
364
                    ]*{)(?<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}",
                \hookrightarrow
                    0),
                // Inside the scope of ~!_exceptionsBag!~ replace:
365
                // _exceptionsBag.Add(exception);
366
                // _exceptionsBag.push_back(exception);
                (new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<befor_</pre>
368
                    e>((?<!/\*~\k<fieldName>~\*/)(.|\n))*?)\k<fieldName>\.Add"),
                    "${scope}${separator}${before}${fieldName}.push_back", 10),
                // Remove scope borders.
                // /*~_exceptionsBag~*/
370
371
                (new Regex(0"/*^{[a-zA-Z0-9]+^**/"}), "", 0),
                // Insert scope borders.
373
                // class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
// class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: static std::mutex
374
375
                    _exceptionsBag_mutex;
```

```
(new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t]*)class [^{\r\n]+\r\n[\t
376
                               ]*{)(?<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:
                         // return std::vector<std::exception>(_exceptionsBag);
                         // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
379

    std::vector<std::exception>(_exceptionsBag);
                         (\text{new Regex}(@"(?<scope>//*^{(?<fieldName>[_a-zA-Z0-9]+)^*/})(?<separator>.|\n)(?<befor_left)
380
                               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);${after}", 10),
Inside the scope of ~!_exceptionsBag!~ replace:
381
                             _exceptionsBag.Add(exception);
                         // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
383
                               _exceptionsBag.Add(exception);
                         (new Regex(0"(?<scope>/\times~(?<fieldName>[_a-zA-Z0-9]+)~\times/)(?<separator>.|\setminusn)(?<befor
384
                               e>((?<!/*^k<fieldName>^**/)(.|n))*?){(?<after>((?!lock_guard)([^{};]|n))*?}r_1
                               n(?<indent>[ \t]*)\k<fieldName>[^;}\r\n]*;)")
                               "${scope}${separator}${before}{" + Environment.NewLine +
                               "${indent}std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", 10),
                         // Remove scope borders.
385
                         // /*~_exceptionsBag~*/
386
                         (new Regex(0"/\*^[_a-zA-Z0-9]+^*\*/"), "", 0),
                         // Insert scope borders.
389
                         // class IgnoredExceptions { ... public: static inline
390
                               Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
                               ExceptionIgnored = OnExceptionIgnored;
                         // class IgnoredExceptions {/*~ExceptionIgnored~*/ ... public: static inline
391
                               Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
                               ExceptionIgnored = OnExceptionIgnored;
                         (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n]+\r\n[\t
392
                               |public): )static inline
                               Platform::Delegates::MulticastDelegate<(?<argumentType>[^;\r\n]+)>
                               (?\langle name \rangle [_a-zA-ZO-9]+) = (?\langle defaultDelegate \rangle [_a-zA-ZO-9]+);)"),
                               "${classDeclarationBegin}/*~${name}~*/${middle}${eventDeclaration}", 0),
                         // Inside the scope of ~!ExceptionIgnored!~ replace:
                         // ExceptionIgnored.Invoke(NULL, exception);
                         // ExceptionIgnored(NULL, exception);
395
                         (\text{new Regex}(@"(?<scope>//*^(?<eventName>[a-zA-Z0-9]+)^/*/)(?<separator>.|\n)(?<before_|
396
                               >((?<!/*^k<eventName>^*/)(.|n))*?)k<eventName>|.Invoke||),
                               "${scope}${separator}${before}${eventName}", 10),
                         // Remove scope borders.
397
                         // /*~ExceptionIgnored~*/
398
                         (new Regex(0"/\*^[a-zA-Z0-9]+^\*/"), "", 0),
400
                         // Insert scope borders.
401
                            auto added = new StringBuilder();
                         // /*~sb~*/std::string added;
403
                         (new Regex(@"(auto|(System\.Text\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new
404
                               (System\.Text\.)?StringBuilder\(\);"), "/*~${variable}~*/std::string
                               ${variable}; ", 0),
                         // static void Indent(StringBuilder sb, int level)
405
                         // static void Indent(/*~sb~*/StringBuilder sb, int level)
                         (new Regex(@"(?<start>, |\()(System\.Text\.)?StringBuilder
407
                               (?<variable>[a-zA-Z0-9]+)(?<end>,|\))"), "${start}/*~${variable}~*/std::string&
                         408
                         // sb.ToString()
                         // sb
410
                         (\text{new Regex}(@"(?<scope>/<math>*"(?<variable>[a-zA-Z0-9]+)")*/)(?<separator>.|\n)(?<before>|
411
                               ((? <!/*^k < variable > ^k/)(.|\n)) *?) \k < variable > \. To String \((\)"),
                               "${scope}${separator}${before}${variable}", 10),
                         // sb.AppendLine(argument)
412
                         // 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]+)^*/(?<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]+)^*/(?<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]+)^*/(?<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]+)^*/(?<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
414
                               ((?<!/\*~\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');
416
```

```
(\text{new Regex}(@"(?<scope>/)*^(?<variable>[a-zA-Z0-9]+)^**/)(?<separator>.|\n)(?<before>|
417
                                  ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.Append\('(?<character>[^'\r\n]
                                        (?<count>[^{\}, r^{\}), r^{\})
                                 "${scope}${separator}${before}${variable}.append(${count}, '${character}')", 10),
                           // sb.Append(argument)
                           // sb.append(Platform::Converters::To<std::string>(argument))
419
                           (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>|
                                   ((? < !/* \land \texttt{k} < \texttt{variable} > `` +/) (. | \land n)) *?) \land \texttt{variable} \land \texttt{Append} \land ((? < \texttt{argument} > [^ \land) , \land r \land n] ) 
                                 +)\)"),
                                 "${scope}${separator}${before}${variable}.append(Platform::Converters::To<std::s
                                 tring>(${argument}))",
                                 10).
                           // Remove scope borders.
421
                           // /*~sb~*/
422
                           11
423
                           (\text{new Regex}(@"/\*^[a-zA-Z0-9]+^\*/"), "", 0),
424
                           // Insert scope borders.
                           // auto added = new HashSet<TElement>();
426
                           // ~!added!~std::unordered_set<TElement> added;
427
                           (new Regex(@"auto (?<variable>[a-zA-Z0-9]+)
428
                                 HashSet < (? < element > [a-zA-Z0-9] +) > ( ); "),
                                 "~!${variable}!~std::unordered_set<${element}> ${variable};", 0),
                           // Inside the scope of ~!added!~ replace:
429
                           // added.Add(node)
                           // added.insert(node)
431
                           (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?< |</pre>
432
                                  !~!\k<variable>!~)(.|\n))*?)\k<variable>\.Add\((?<argument>[a-zA-Z0-9]+)\)"),
                                 "${scope}${separator}${before}${variable}.insert(${argument})", 10),
                           // Inside the scope of ~!added!~ replace:
433
                           // added.Remove(node)
                           // added.erase(node)
435
                           (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?<|</pre>
436
                                  !~!\k<variable>!~)(.|\n))*?)\k<variable>\.Remove\((?<argument>[a-zA-Z0-9]+)\)"),
                                 "${scope}${separator}${before}${variable}.erase(${argument})", 10),
                           // if (added.insert(node)) {
                           // if (!added.contains(node)) { added.insert(node);
438
                           (\text{new Regex}(@"if \setminus ((?<\text{variable}=a-zA-Z0-9]+) \setminus (?<\text{argument}=a-zA-Z0-9]+) \setminus) (?_{argument}=a-zA-Z0-9]+))))
439
                                 \operatorname{separator}[\t] *[\r\n] +) (? \operatorname{sindent}[\t] *) {"}, "if
                                  (!${variable}.contains(${argument}))${separator}${indent}{" +
                                 Environment.NewLine + "${indent}
                                                                                              ${variable}.insert(${argument});", 0),
                           // Remove scope borders.
440
                           // ~!added!^
441
442
                           (new Regex(0"^{-1}[a-zA-Z0-9]+!^{-1}), "", 5),
443
                           // Insert scope borders.
                           // auto random = new System::Random(0);
445
                           // std::srand(0);
446
                           (\text{new Regex}(@"[a-zA-Z0-9]] + ([a-zA-Z0-9]] + ) = \text{new}
447
                                  (\text{System}::)?Random(([a-zA-Z0-9]+));"), "~!$1!~std::srand($3);", 0),
                           // Inside the scope of "!random!" replace:
                              random.Next(1, N)
449
                           // (std::rand() % N) + 1
450
                           (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.
452
                               ~!random!
                           //
453
                           //
454
                           (new Regex(0"^{-1}[a-zA-Z0-9]+!^{-1}), "", 5),
455
                           // Insert method body scope starts.
                           // void PrintNodes(TElement node, StringBuilder sb, int level) {
457
                           // void PrintNodes(TElement node, StringBuilder sb, int level) {/*method-start*/
458
                           (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*/${end}",
                                 0),
                           // Insert method body scope ends.
460
                           // {/*method-start*/...}
                           // {/*method-start*/.../*method-end*/}
462
                           (new\ Regex(@"\{/\*method-start\*/(?<body>((?<bracket>\{) | (?<-bracket>\{}) | [^\{\}]*)+)_{|}})
463
                                 \}"),
                                           "{/*method-start*/${body}/*method-end*/}",
                                 0)
                           // Inside method bodies replace:
```

```
// GetFirst(
465
                           // this->GetFirst(
                           (new
467
                                 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.
468
                           // /*method-start*/
                           //
470
                           (new Regex(0"/\astmethod-(start|end)\ast/"), "", 0),
471
                           // Insert scope borders.
472
                               const std::exception& ex
                           // const std::exception& ex/*~ex~*/
474
                           (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?exception&?
475
                                  (?\langle variable \rangle [ a-zA-Z0-9]+))(?\langle after \rangle \| )
                                  "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
                           // Inside the scope of ~!ex!~ replace:
476
                           // ex.Message
                           // ex.what()
                           (new Regex(@"(?<scope>/\*~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before</pre>
479
                                 >((?<!/*^k<variable>^k/)(.|n))*?)(Platform::Converters::To<std::string>\(\k<|)**
                                 variable>\.Message\) | \k<variable>\.Message) ");
                                 "${scope}${separator}${before}${variable}.what()", 10),
                           // Remove scope borders.
480
                           // /*~ex~*/
481
                           //
                           (new Regex(0"/\*^[_a-zA-Z0-9]+^*\*/"), "", 0),
483
                           // throw ObjectDisposedException(objectName, message);
484
                           // throw std::runtime_error(std::string("Attempt to access disposed object
485
                                  [").append(objectName).append("]: ").append(message).append("."));
                           (new Regex(@"throw ObjectDisposedException\((?<objectName>[a-zA-Z_][a-zA-Z0-9_]*);
                                  (?<message>[a-zA-Z0-9_]*[Mm]essage[a-zA-Z0-9_]*(\(\))?|[a-zA-Z_][a-zA-Z0-9_]*)\)|
                                  ;"), "throw std::runtime_error(std::string(\"Attempt to access disposed object
                                  [\"] . append(\{\{\{\}\}\}) . append(\{\{\}\}\}) . append(\{\{\}\}) . append(\{\{\}\}
                            \hookrightarrow
                                 0),
                           // throw ArgumentNullException(argumentName, message);
                           // throw std::invalid_argument(std::string("Argument
488
                                 ").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);
490
                           // throw std::invalid_argument(std::string("Invalid ").append(argumentName).append("
491
                                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());
493
                           // throw std::invalid_argument(std::string("Value
494
                                  [").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}] *)?) \;,
                                   (?\langle message \rangle [a-zA-Z] * [Mm] essage [a-zA-Z] * (\backslash (\backslash))?) \backslash);"), "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();
496
                           // throw std::logic_error("Not supported exception.");
497
                           (new Regex(@"throw NotSupportedException\(\);"), "throw std::logic_error(\"Not
                                supported exception.\");", 0);
                           // throw NotImplementedException();
499
                           // throw std::logic_error("Not implemented exception.");
500
                           (new Regex(@"throw NotImplementedException\(\);"), "throw std::logic_error(\"Not
501
                                 implemented exception.\");", 0),
                           // Insert scope borders.
                           // const std::string& message
503
```

```
// const std::string& message/*~message~*/
504
                          (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:
506
                          // Platform::Converters::To<std::string>(message)
507
                          // message
508
                          (\text{new Regex}(@"(?<scope>/*"(?<variable>[_a-zA-Z0-9]+)"\*/)(?<separator>.|\n)(?<before_1)(?<scope)()
509
                                >((?<!/\*~\k<variable>~\*/)(.|\n))*?)Platform::Converters::To<std::string>\(\k<v<sub>|</sub>
                                ariable>\)"), "${scope}${separator}${before}${variable}",
                                10),
                          // Remove scope borders.
510
                          // /*~ex~*/
511
                          //
512
                          (\text{new Regex}(@"/\*^[_a-zA-ZO-9]+^\*/"), "", 0),
513
                          // Insert scope borders.
                          // std::tuple<T, T> tuple
// std::tuple<T, T> tuple/*~tuple~*/
515
516
                          (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?tuple<[^\n]+>&?
517
                                 (?<variable>[_a-zA-Z0-9]+))(?<after>\W)")
                                "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
                          // Inside the scope of ~!ex!~ replace:
518
                          // tuple.Item1
                          // std::get<1-1>(tuple)
520
                          (\text{new Regex}(@"(?<scope>/)*^(?<variable>[_a-zA-Z0-9]+)^*/)(?<separator>.|\n)(?<before)
521
                                >((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.Item(?<itemNumber>\d+)(?<afte_
                                r>\W)")
                                "${scope}${separator}${before}std::get<${itemNumber}-1>(${variable})${after}",
                           \hookrightarrow
                          // Remove scope borders.
522
                          // /*~ex~*/
523
                          //
                          (new Regex(0"/\*^[_a-zA-Z0-9]+^*\*/"), "", 0),
525
                          // Insert scope borders.
526
                          // class Range<T>
                          // class Range<T> {/*~type~Range<T>~*/
528
                          (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t]*)(template\s*<[^<>\n]*>
529
                                )?(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)
531
                          // public: operator std::tuple<T, T>() const {/*~variable~Range<T>~*,
532
                          (new Regex(@"(?<scope>/\*~type~(?<typeName>[^~\n\*]+)~(?<fullType>[^~\n\*]+)~\*/)(?<
533
                                ?<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~${variable}~*/", 10),
                          // Inside the scope of /*~type~Range<T>~*/ replace:
534
                          // public: static implicit operator Range<T>(std::tuple<T, T> tuple) { return new
535
                                Range<T>(std::get<1-1>(tuple), std::get<2-1>(tuple)); }
                          // public: Range(std::tuple<T, T> tuple) : Range(std::get<1-1>(tuple),
536
                                std::get<2-1>(tuple)) { }
                          (new Regex(@"(?<scope>/\*~type~(?<typeName>[^~\n\*]+)~(?<fullType>[^~\n\*]+)~(*/)
537
                                separator >. \ |\ ) \ (?<before > ((?<!/*~type^k<typeName > ^k<fullType > ^* +/) (. \ |\ )) *?) (|
                                ?<access>(private|protected|public): )static implicit operator
                                 "${scope}${separator}${before}${access}${typeName}(${arguments}) :
                                $\{\typeName\}(\$\{\passedArguments\}) \{\}\", 10),
                          // Inside the scope of /*~variable~range~*/ replace:
539
                          // range.Minimum
                          // this->Minimum
540
                          (new Regex(@"(?<scope>{/\*~variable~(?<variable>[^~\n]+)~\*/)(?<separator>.|\n)(?<be |</pre>
                                fore>(?\langle beforeExpression>(?\langle bracket> \{) | (?\langle -bracket> \}) | [^{ }] | \n) *?) \\ \\ \langle constraint | constrai
                                  \begin{tabular}{ll} (?&field>[_a-zA-Z0-9]+) (?&after>(,|;||)| \\ ||)) (?&afterExpression>(?&bracket>||(?&-bracket>|)|[^{}||n)*?|)"), \\ \end{tabular} 
                                "${scope}${separator}${before}this->${field}${after}", 10),
                          // Remove scope borders.
542
                          // /*~ex~*/
543
544
                          (\text{new Regex}(@"/\*"[^"\n]+"[^"\n]+"\*/"), "", 0),
545
                          // Insert scope borders.
```

```
// namespace Platform::Ranges {
                                                  ...}
// namespace Platform::Ranges {/*~start~namespace~Platform::Ranges~*/ ...
     /*~end~namespace~Platform::Ranges~*/}
(new Regex(@"(?<namespaceDeclarationBegin>\r?\n(?<indent>[\t ]*)namespace
      (?<name>name>(?<namePart>[a-zA-Z][a-zA-Z0-9]+)(?<nextNamePart>::[a-zA-Z][a-z]
      _{\rightarrow} nd~namespace~${namespaceName}~*/${end}",
// Insert scope borders.
// class Range<T> { ... };
// class Range<T> {/*~start~type~Range<T>~T~*/ ... /*~end~type~Range<T>~T~*/};
(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>(.|\n)*)(?<endIndent>(?<=\r?\n)\k<indent>)(?<end>};)"), "${classDeclarationBegin}/*~start~type~${type}~${typeParameter}~*/${middle}${end}
      Indent}/*~end~type~${type}~${typeParameter}~*/${end}",
// Inside scopes replace:
// /*~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~*/ ...
     /*~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}${after}${<sub>|</sub>
      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~*/
// /*~start~method~*/ ... obj.Minimum ... /*~end~method~*/
(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~\*/)")
      "${methodScopeStart}$\(\frac{10}{5}\), $\(\frac{10}{5}\), $\(\frac{10}\), $\(\frac{10}{5}
// Remove scope borders.
// /*~start~type~Range<T>~*/
(new Regex(0"/\*[^{\sim} \times n] + (^{(\sim} \times n] + ) *^{<} / "), "", 0),
// class Disposable <T> : public Disposable
// class Disposable<T> : public Disposable<>
(\texttt{new Regex}(@"(?<\texttt{before}>(\texttt{struct}|\texttt{class}) \quad (?<\texttt{type}>[\texttt{a-zA-Z}][\texttt{a-zA-Z0-9}]*)<[^<<\\\texttt{n}]+> \ :
      (?<access>(private|protected|public) )?\k<type>)(?<after>\b(?!<))"),</pre>
     "${before}<>${after}", 0),
// Insert scope borders.
// class Disposable<T> : public Disposable<> { ... };
// class Disposable<T> : public Disposable<>
     {/*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/ ...
     /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/};
```

547

549

551

552

553

556

559

562

563

565

566

569

570

```
(new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t]*)template[\t
572
                                                ] *< (?< type Parameters > [^n] *) > [t] * (struct | class) [t]
                                                ]+(?<fullType>(?<type>[a-zA-Z][a-zA-Z0-9]*)(<[^<>\n]*>)?)[\t ]*:[\t
                                                ]*(?<access>(private|protected|public)[\t
                                                ]+)?(?<fullBaseType>(?<baseType>[a-zA-Z][a-zA-Z0-9]*)(<[^<>\n]*>)?)[\t
                                                 ]*(\r?\n)?[\t
                                                ]*{)(?<middle>(.|\n)*)(?<beforeEnd>(?<=\r?\n)\k<indent>)(?<end>};)"),
                                                "${classDeclarationBegin}/*~start~type~${type}~${fullType}~${baseType}~${fullBas_
                                                BaseType}~*/${end}",
                                                0),
                                       // Inside scopes replace:
573
                                       // /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/ ... ) : base(
574
                                                 ... /*~end~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/
                                       // /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/ ... ) :
                                        → Disposable<>( /*~end~type~Disposable~Disposable<T>~Disposable~Disposable<>>~*/
                                       (new Regex(@"(?<before>(?<typeScopeStart>/\**start~type~(?<types>(?<type>[^~\n\*]+)~]
                                                  (?<fullType>[^-\n\*]+)^-\k<type>^(?<fullBaseType>[^-\n\*]+))^-\*/)(.^\n)+?\) \
                                                )*base(?<after>\((.|\n)+?(?<typeScopeEnd>/\*~end~type~\k<types>~\*/))"),
                                                "${before}${fullBaseType}${after}", 20),
                                       // Inside scopes replace:
                                       // /*~start~type~Disposable~Disposable<T>~X~X<>~*/ ... ) : base( ...
                                                /*~end~type~Disposable~Disposable<T>~X~X<>~*/
                                       // /*~start~type~Disposable~Disposable<T>~X~X<>~*/ ... ) : X(
                                        → /*~end~type~Disposable~Disposable<T>~X~X<>~*/
                                       (new Regex(@"(?<before>(?<typeScopeStart>/\*~start~type~(?<types>(?<type>[^~\n\*]+)~|
                                                 (?<fullType>[^{^{n}}+)^{^{(?}}baseType>[^{^{n}}+)^{^{(?}}fullBaseType>[^{^{n}}+))^{*/}(...)
                                                 \n)+?\)\s*:\s)*base(?<after>\((.|\n)+?(?<typeScopeEnd>/\*~end~type~\k<types>~\*_
                                                /))"), "${before}${baseType}${after}",
                                                20),
                                       // Inside scopes replace:
                                       // /*~start~type~Disposable~Disposable<T>~X~X<>~*/ ... public: Disposable(T object)
582
                                              { Object = object; } ... public: Disposable(T object) : Disposable(object) { }
                                                  ... /*~end~type~Disposable~Disposable<T>~X~X<>~*/
                                       // /*~start~type~Disposable~Disposable<T>~X~X<>~*/ ... public: Disposable(T object)
583
                                                { Object = object; } /*~end~type~Disposable~Disposable<T>~X~X<>~*/
                                       (new Regex(@"(?<before>(?<typeScopeStart>/\*~start~type~(?<type>[^~\n\*]+)~
                                                  (?\langle \text{fullType}\rangle [^{\sim} \n \rangle +)^{\sim} (?\langle \text{fullBaseType}\rangle [^{\sim} \n \rangle +))^{\sim} (?\langle \text{fu
                                                 |\n)+?(?<constructor>(?<access>(private|protected|public):[\t
                                                ]*)?\k<type>\(((?<arguments>[^()\n]+)\)\s*{[^{}\n]+})(. |\n)+?)*(?<duplicateConstr
                                                uctor>(?<access>(private|protected|public):[\t
                                                copeEnd>/\*~end~type~\k<types>~\*/))"), "${before}${after}",
                                                20),
                                       // Remove scope borders.
                                       // /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/
586
587
                                       (new Regex(0"/\*[^{\sim} \times n] + (^{(\sim} \times n] + ) *^{\sim} */"), "", 0),
                             }.Cast<ISubstitutionRule>().ToList();
589
                             public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
591
592
                                       // ICounter<int, int> c1;
593
                                       // ICounter<int, int>* c1;
594
                                       (new Regex(0"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^>\setminusr\n]+>)?)
595
                                                 (?<variable>[_a-zA-Z0-9]+)(?<after> = null)?;"), "${abstractType}*
                                                ${variable}${after};", 0),
                                       // (expression)
                                       // expression
597
                                       (\text{new Regex}(@"((| )(([a-zA-Z0-9_{*:}]+)))(,| |;|))"), "$1$2$3", 0),
598
                                       // (method(expression))
                                       // method(expression)
                                       (new Regex(0"(?<firstSeparator>(\())
601
                                                ))\((?method>[a-zA-Z0-9_\->\*:]+)\((?expression>((?expression>()|(?expression>()
                                               hesis > )) | [a-zA-Z0-9_\-> *:] *) +) (?(parenthesis) (?!)) \) ) (?(lastSeparator > (, | Bartan | Contact | Contac
                                             |;|\)))"), "${firstSeparator}${method}(${expression})${lastSeparator}", 0), append(".")
602
                                              .append(1
                                       (new Regex(0"\.append\(""([^\\""]|\\[^""])""\)", ".append(1, '$1')", 0),
604
                                       // return ref _elements[node];
605
                                       // return &_elements[node];
606
                                       (new Regex(@"return ref ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9\*]+)\];"), "return &$1[$2];",
607
                                              0),
                                       // ((1,
608
                                       // ({1, 2})
609
```

```
(new Regex(@"(?<before>\(|, )\((?<first>[^\n()]+),
610
                                          (?\langle second \rangle [^n()] +) (?\langle after \rangle) |, )"), "$\{before\}{\{first\}, }
                                         ${second}}${after}", 10),
                                 // {1, 2}.GetHashCode()
611
                                 // Platform::Hashing::Hash(1, 2)
612
                                 (\text{new Regex}(@"\{(?<\text{first})^n{}\}+), (?<\text{second}(^n{})+)\}.GetHashCode(())"),
613
                                         "Platform::Hashing::Hash(${first}, ${second})", 10),
                                 // range.ToString()
614
                                 // Platform::Converters::To<std::string>(range).data()
615
                                 (new Regex(@"(?<before>\W)(?<variable>[_a-zA-Z][_a-zA-Z0-9]+)\.ToString\(\)"),
                                        "${before}Platform::Converters::To<std::string>(${variable}).data()", 10),
                                 // new
617
                                 //
618
                                 619
                                 \rightarrow s+"), "${before}",
                                        10),
                                 // x == null
620
                                 // x == nullptr
621
                                 (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(?<v|</pre>
622
                                         ariable > [_a-zA-Z][_a-zA-Z0-9]+) (? operator > (s*(==|!=) s*)null(? (after > \W)"),
                                         "${before}${variable}${operator}nullptr${after}", 10),
                                 // null
623
                                 // {}
624
                                 (\texttt{new Regex}(@"(?<\texttt{before}\r?\n[^""\r\n]*(""(\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)\ null_{||})
                                         (?<after>\W)"), "${before}{}${after}",
                                         10)
                                 // default
626
                                 // 0
627
                                 (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)defa|</pre>
628

    ult(?<after>\W)"), "${before}0${after}",
                                        10),
                                 // object x
629
                                 // void *x
630
                                 (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(?<! |</pre>
631
                                         @)(object|System\.Object) (?<after>\w)"), "${before}void *${after}",
                                         10),
                                 // <object>
                                 // <void*>
633
                                 (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*) (?<=\W) (?<! |</pre>
634
                                         @)(object|System\.Object)(?<after>\W)"), "${before}void*${after}",
                                         10).
                                 // @object
                                 // object
636
                                 (new Regex(0"0([_a-zA-Z0-9]+)"), "$1", 0).
637
                                      this->GetType().Name
638
                                 // typeid(this).name()
639
                                 (new Regex(@"(this)->GetType\(\)\.Name"), "typeid($1).name()", 0),
640
                                 // ArgumentNullException
641
                                 // std::invalid_argument
642
                                 (\texttt{new Regex}(@"(?<\texttt{before}\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(Sys_1)
643
                                        tem\.)?ArgumentNullException(?<after>\W)");
                                         "${before}std::invalid_argument${after}", 10),
                                 // InvalidOperationException
644
645
                                 // std::runtime_error
                                 (new Regex(@"(\W)(InvalidOperationException|Exception)(\W)"),
                                         "$1std::runtime_error$3", 0),
                                 // ArgumentException
647
                                 // std::invalid_argument
648
                                 (new Regex(@"(\W)(ArgumentException|ArgumentOutOfRangeException)(\W)"),
649
                                         "$1std::invalid_argument$3", 0),
                                 // template <typename T> struct Range : IEquatable<Range<T>>
650
                                 // template <typename T> struct Range {
(new Regex(@"(?<before>template <typename (?<typeParameter>[^\n]+)> (struct|class)
651
652
                                          (?<type>[a-zA-Z0-9]+<[^\n]+>)) : (public)
                                         // public: delegate void Disposal(bool manual, bool wasDisposed);
653
                                 // public: delegate void Disposal(bool, bool);
654
                                 (new Regex(@"(?<before>(?<access>(private|protected|public): )delegate
655
                                          (?\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);
656
```

```
// using Disposal = void(bool, bool);
657
                 (new Regex(@"(?<access>(private|protected|public): )delegate
                     (?< returnType>[a-zA-Z][a-zA-Z0-9:]+)
                     (?< delegate>[a-zA-Z][a-zA-Z0-9]+)((?< argumentTypes>[^\(\)\n]*)\);"), "using"
                     ${delegate} = ${returnType}(${argumentTypes});", 20),
                 // <4-1>
659
                 // <3>
660
                 (new Regex(@"(?<before><)4-1(?<after>>)"), "${before}3${after}", 0),
                 // <3-13
662
                 // <2>
663
                 (new Regex(@"(?<before><)3-1(?<after>>)"), "${before}2${after}", 0),
664
                 // <2-1>
                 // <1>
666
                 (new Regex(@"(?<before><)2-1(?<after>>)"), "${before}1${after}", 0),
667
                 // <1-1>
                 // <0>
669
                 (new Regex(@"(?<before><)1-1(?<after>>)"), "${before}0${after}", 0),
670
                 // #region Always
671
672
                 (\text{new Regex}(@"(^|\r?\n)[ \t]*\\ \text{(region|endregion)}[^\r\n]*\\ \text{(r?}\n|\$)"), "", 0),
673
                 // //#define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
674
                 (new Regex(0"\/\/[\t]*\#define[\t]+[_a-zA-Z0-9]+[\t]*"), "", 0),
676
                 // #if USEARRAYPOOL\r\n#endif
677
678
                 (new Regex(0"#if [a-zA-Z0-9]+\s+\#endif"), "", 0),
679
                 // [Fact]
680
                 //
681
                 (new Regex(0"(?<firstNewLine>\r?\n|\A)(?<indent>[\t
                     ]+)\[[a-zA-Z0-9]+(\((?<expression>((?<parenthesis>\())|(?<-parenthesis>\)))|[^()\r_1
                     \n] *) +) (?(parenthesis)(?!)))))?) [ \t] * (\r?\n\k<indent>)?"),
                     "${firstNewLine}${indent}", 5),
                 // \A \n ... namespace
683
                 // \Anamespace
684
                 (new Regex(0"(\A)(\r?\n)+namespace"), "$1namespace", 0),
685
                    \A \n ... class
                 // \Aclass
687
                 (new Regex(0"(\A)(\r?\n)+class"), "$1class", 0),
688
                 // \ln n
                 // \n n
690
                 (new Regex(@"\r?\n[ \t]*\r?\n"), Environment.NewLine +
691
                     Environment.NewLine, 50),
                    {n n}
692
                 // {\n
                 (new Regex(0"{[ t]*r?n[ t]*r?n"), "{" + Environment.NewLine, 10),
694
                 // \n n
695
                 // \n}
696
                 (new Regex(@"\r?\n[ \t]*\r?\n(?<end>[ \t]*})"), Environment.NewLine + "${end}", 10),
697
             }.Cast<ISubstitutionRule>().ToList();
698
699
            public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules)
700
                base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
701
            public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
702
        }
703
704
     ./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs
1.2
    using Xunit;
    {\tt namespace}\ \ {\tt Platform.RegularExpressions.Transformer.CSharpToCpp.Tests}
 3
 4
        public class CSharpToCppTransformerTests
 5
             [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("");
                 Assert.Equal("", actualResult);
13
             }
14
15
             [Fact]
16
            public void HelloWorldTest()
17
                 const string helloWorldCode = @"using System;
```

19

```
class Program
20
^{21}
         public static void Main(string[] args)
^{22}
23
              Console.WriteLine(""Hello, world!"");
    }";
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
              }
         }
38
    }
39
```

## Index

./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs, 16 ./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1