```
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 n n
                // {
                (new Regex(0"\{\s+[\r\n]+"\}, "{" + Environment.NewLine, 0),
22
                // Platform.Collections.Methods.Lists
                // Platform::Collections::Methods::Lists
                (new Regex(0"(namespace[^{r}]+?)\.([^{r}]+?)"), "$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-ZO-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
62
                                                        // 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: (\cdot,\cdot) = 
                                                          "${newLineAndIndent}${access}: ${before}", 0),
                                              // public: static bool CollectExceptions { get; set; }
69
                                              // public: inline static bool CollectExceptions;
70
                                               (new Regex(@"(?<access>(private|protected|public): )(?<before>(static )?[^\r\n]+
71
                                                       )(?<name>[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)
                                                        )*(?<category>interface|class|struct)"), "${category}", 0),
                                              // class GenericCollectionMethodsBase<TElement>
                                              // template <typename TElement> class GenericCollectionMethodsBase {
76
                                              (new Regex(0"class ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>([^{{]+}}{"}), "template <typename $2>)
                                               \rightarrow class $1$3{", 0),
                                              // static void
                                                       TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
                                                       tree, TElement* root)
                                              // template<typename T> static void
                                               _{\hookrightarrow} \quad \texttt{TestMultipleCreationsAndDeletions} < \texttt{TElement} > (\texttt{SizedBinaryTreeMethodsBase} < \texttt{TElement} > \texttt{TEl
                                                 → tree, TElement* root)
                                               (\text{new Regex}(@"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 TProduct> class IFactory { public:
                                              (new Regex(@"interface (?<interface>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9]
83
                                                          ,]+)>(?<whitespace>[^{]+){"}, "template <typename...> class ${interface};
                                                          template <typename ${typeParameters}> class
                                                         $\{\interface\} < \{\text{typeParameters}} \$\{\text{whitespace}\{\text{" + Environment.NewLine + \text{"}}}\]</pre>
                                                         public:", 0),
                                              // template <typename TObject, TProperty, TValue>
                                              // template <typename TObject, typename TProperty, TValue>
                                              (new Regex(0"(?<before>template <((, )?typename [a-zA-Z0-9]+)+,</pre>
86
                                                         )(?<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
                                              "/*~extensionMethod~${name}~*/$0", 0),
                                              // Move all markers to the beginning of the file.
                                               (\text{new Regex}(@"\A(?<\text{before}[^\r]+\r?\n(.|\n)+)(?<\text{marker}/\*^extensionMethod}^{(?<\text{name})})
92
                                                        [a-zA-Z0-9]+)^*/", "${marker}${before}",
                                                         10),
                                              // /*~extensionMethod~BuildExceptionString~*/...sb.BuildExceptionString(exception.In]
93

    nerException, level +

                                                        1);
                                              // /*~extensionMethod~BuildExceptionString~*/...BuildExceptionString(sb,
                                                        exception.InnerException, level + 1);
                                              (new Regex(@"(?<before>\bar{\ \ \ \ }\*~extensionMethod~(?<name>[a-zA-Z0-9]+)~\*/(.|\n)+\W)(?<var_1)
95
                                                         50),
                                              // Remove markers
                                              // /*~extensionMethod~BuildExceptionString~*/
97
                                              //
```

```
(new Regex(0"/*extensionMethod[a-zA-Z0-9]+<math>**/"), "", 0),
                          // (this
                          // (
101
                          (new Regex(0"\(this "), "(", 0),
102
                          // public: static readonly EnsureAlwaysExtensionRoot Always = new
                              EnsureAlwaysExtensionRoot();
                          // public:inline static EnsureAlwaysExtensionRoot Always;
                           (new Regex(@"(?<access>(private|protected|public): )?static readonly
105
                                 (?<type>[a-zA-Z0-9]+) (?<name>[a-zA-Z0-9_]+) = new k<type>(\);"),
                                 "${access}inline static ${type} ${name}; ", 0),
                          // public: static readonly string ExceptionContentsSeparator = "---";
106
                          // public: inline static const char* ExceptionContentsSeparator = "---";
                           (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly) string
108
                                 (?\langle name \rangle [a-zA-Z0-9_]+) = ""(?\langle string \rangle (\""|[^""\r\n])+)"";"), "$\{access\}inline\}
                                static const char* ${name} = \"${string}\";", 0),
                          // private: const int MaxPath = 92;
109
                          // private: inline static const int MaxPath = 92;
110
                          (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly)
                                 (?<type>[a-zA-Z0-9]+) (?<name>[a-zA-Z0-9]+) = (?<value>[^;\r\n]+);"),
                                 "${access}inline static const ${type} ${name} = ${value}; ", 0),
                          //
                                 ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
112
                                 TArgument : class
                                 ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
                           (\text{new Regex}(@"(?<\text{before}> [a-zA-Z]+\(([a-zA-Z *,]+, |))(?<\text{type}>[a-zA-Z]+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{after}>(|a-zA-Z|+)(?<\text{aft
114
                                 [a-zA-Z *,]+)))[ \r\n]+where \k<type> : class"), "${before}${type}*${after}",
                                0),
                          // protected: abstract TElement GetFirst();
115
                          // protected: virtual TElement GetFirst() = 0;
116
                          (new Regex(@"(?<access>(private|protected|public): )?abstract
                                 (?<method>[^;\r\n]+);"), "${access}virtual ${method} = 0;", 0),
                              TElement GetFirst();
118
                          // virtual TElement GetFirst() = 0;
119
                          (\text{new Regex}(@"([\r\n]+[ ]+)((?!\text{return})[a-zA-Z0-9]+ [a-zA-Z0-9]+\([^\)\r\n]*\))(;[
120
                                ]*[\r\n]+)"), "$1virtual $2 = 0$3", 1),
                          // protected: readonly TreeElement[]
                          // protected: TreeElement _elements[N];
122
                          (new Regex(0"(?<access>(private|protected|public): )?readonly
123
                                 (?<type>[a-zA-Z<>0-9]+)([\[\]]+) (?<name>[_a-zA-Z0-9]+);"), "${access}${type}
                                 ${name}[N];", 0),
                          // protected: readonly TElement Zero;
                          // protected: TElement Zero;
125
                          (new Regex(@"(?<access>(private|protected|public): )?readonly
126
                                 (?<type>[a-zA-Z<>0-9]+) (?<name>[_a-zA-Z0-9]+);"), "${access}${type} ${name};",
                                0),
                          // internal
                          //
                          (new Regex(@"(\W)internal\s+"), "$1", 0),
129
                          // static void NotImplementedException(ThrowExtensionRoot root) => throw new
130
                                NotImplementedException();
                          // static void NotImplementedException(ThrowExtensionRoot root) { return throw new
                           → NotImplementedException(); }
                          (new Regex(@"(^\s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static
132
                                // SizeBalancedTree(int capacity) => a = b;
133
                          // SizeBalancedTree(int capacity) { a = b; }
                          (new Regex(@"(^\s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static
135
                                 )?(override )?(void )?([a-zA-Z0-9]+)(([^\(\r\n]*)))s+=>s+([^;\r\n]+);"),
                                 "$1$2$3$4$5$6$7$8($9) { $10; }"
                          // int SizeBalancedTree(int capacity) => a;
                          // int SizeBalancedTree(int capacity) { return a; }
137
                          (new Regex(@"(^\s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static
138
                                 )?(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),
                               () => Integer<TElement>.Zero,
                          // () { return Integer<TElement>.Zero; }
140
                          (new Regex(0"\(\)\s+=>\s+(?<expression>[^(),;\r\n]+(\(((?<parenthesis>\()|(?<-parent
141
                                hesis>\))|[^();\r\n]*?\*?\))?[^(),;\r\n]*)(?<after>,|\);)"), "() { return
                                 ${expression}; \}${after}",
                                                                             0),
                          // => Integer<TElement>.Zero;
142
                          // { return Integer<TElement>.Zero; }
143
                           (new Regex(0"\)\\ddot{s}+=>\s+([^;\r\n]+?);"), ") { return $1; }", 0),
                          // () { return avlTree.Count; }
145
                          // [&]()-> auto { return avlTree.Count; }
146
```

```
(new Regex(@"(?<before>, |\()\(\) { return (?<expression>[^;\r\n]+); }"),
147
                    "${before}[&]()-> auto { return ${expression}; }", 0),
                // Count => GetSizeOrZero(Root);
148
                // GetCount() { return GetSizeOrZero(Root); }
149
                (\text{new Regex}(@"(\W)([A-Z][a-zA-Z]+)\s+=>\s+([^;\r\n]+);"), "$1Get$2() { return $3; }",
150
                    0),
                // ArgumentInRange(const char* message) { const char* messageBuilder() { return
151
                    message; }
                // ArgumentInRange(const char* message) { auto messageBuilder = [&]() -> const char*
                    { return message; };
                 (\text{new Regex}(@"(?<\text{before})W[_a-zA-ZO-9]+\([^\)\n]*\)[\s\n]*{[\s\n]*([^{}]|\n)*?(\r?\n)_{})}
153
                    ?[ \t]*)(?<returnType>[_a-zA-Z0-9*:]+[_a-zA-Z0-9*:]*)
                    (?<methodName>[_a-zA-Z0-9]+)((?<arguments>[^\)\n]*))\s*{(?<body>([^}]|\n)+?)}"_1
                    ),
                       "${before}auto ${methodName} = [&]() -> ${returnType} {${body}};",
                 \hookrightarrow
                    10),
                // Func<TElement> treeCount
154
                // std::function<TElement()> treeCount
155
                 (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", 0),
                // Action<TElement> free
157
                // std::function<void(TElement)> free
158
                (new Regex(0"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<void($1)> $2",
                    0),
                // Predicate<TArgument> predicate
                // std::function < bool (TArgument) > predicate
161
                (new Regex(0"Predicate<((\bar{a}-zA-Z0-9]+)> ((\bar{a}-zA-Z0-9]+)"), "std::function<br/>bool($1)>
162
                    $2", 0),
                // var
                // auto
164
                (new Regex(@"(\W)var(\W)"), "$1auto$2", 0),
165
                // unchecked
166
                //
                (new Regex(@"[\r\n]{2}\s*?unchecked\s*?$"), "", 0),
168
                // throw new InvalidOperationException
169
                // throw std::runtime_error
170
                (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
171
                    std::runtime_error", 0),
                // void RaiseExceptionIgnoredEvent(Exception exception)
172
                // void RaiseExceptionIgnoredEvent(const std::exception& exception)
173
                (new Regex(@"(\(|, )(System\.Exception|Exception)( |\))"), "$1const
                    std::exception&$3", 0),
                // EventHandler<Exception>
175
                // EventHandler<std::exception>
176
                 (new Regex(@"(\W)(System\.Exception|Exception)(\W)"), "$1std::exception$3", 0),
177
                // override void PrintNode(TElement node, StringBuilder sb, int level)
                // void PrintNode(TElement node, StringBuilder sb, int level) override
179
                (new Regex(0"override ([a-zA-Z0-9 \times +]+)(([^\)rn]+?())"), "$1$2 override", 0),
180
                // return (range.Minimum, range.Maximum)
                // return {range.Minimum, range.Maximum}
182
                (new Regex(@"(?<before>return\s*)\((?<values>[^\)\n]+)\)(?!\()(?<after>\W)"),
183
                    "${before}{${values}}${after}", 0),
                // string
184
                // const char*
                (new Regex(@"(\W)string(\W)"), "$1const char*$2", 0),
186
                // System.ValueTuple
187
                // std::tuple
188
                (new Regex(@"(?<before>\W)(System\.)?ValueTuple(?!\s*=)(?<after>\W)"),
189
                    "${before}std::tuple${after}", 0),
                // sbyte
190
                // std::int8_t
191
                192
                    "${before}std::int8_t${after}", 0),
                // short
193
                // std::int16_t
194
                (new Regex(@"(?<before>\W)((System\.)?Int16|short)(?!\s*=)(?<after>\W)"),
195
                    "${before}std::int16_t${after}", 0),
                // int
                // std::int32_t
197
                (new Regex(@"(?<before>\W)((System\.)?I|i)nt(32)?(?!\s*=)(?<after>\W)"),
198
                    "${before}std::int32_t${after}", 0),
                // long
199
                // std::int64_t
200
                (new Regex(@"(?<before>\W)((System\.)?Int64|long)(?!\s*=)(?<after>\W)"),
201
                    "${before}std::int64_t${after}", 0),
                // byte
202
                // std::uint8_t
203
```

```
(\text{new Regex}(@"(?<\text{before}\W)((System\.)?Byte|byte)(?!\s*=)(?<\text{after}\W)"),
204
                                  "${before}std::uint8_t${after}", 0),
                           // ushort
                            // std::uint16_t
206
                            (new Regex(@"(?<before>\W)((System\.)?UInt16|ushort)(?!\s*=)(?<after>\W)"),
207
                                  "${before}std::uint16_t${after}", 0),
                           // uint
208
                            // std::uint32_t
                            (new Regex(@"(?<before>\W)((System\.)?UI|ui)nt(32)?(?!\s*=)(?<after>\W)"),
210
                                  "${before}std::uint32_t${after}", 0),
                           // ulong
211
                           // std::uint64_t
212
                            (new Regex(@"(?<before>\W)((System\.)?UInt64|ulong)(?!\s*=)(?<after>\W)"),
                                  "${before}std::uint64_t${after}", 0),
                           // char*[] args
214
                           // char* args[]
215
                            (\text{new Regex}(@"([_a-zA-ZO-9:\*]?)\[\] ([a-zA-ZO-9]+)"), "$1 $2[]", 0),
216
217
                           // @object
                           // object
218
                            (new Regex(@"@([_a-zA-Z0-9]+)"), "$1", 0),
219
                           // float.MinValue
                            // std::numeric_limits<float>::min()
221
                            (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MinValue(?<after>\W|
222
                                  )"), "${before}std::numeric_limits<${type}>::min()${after}",
                                  0),
                           // double.MaxValue
                           // std::numeric_limits<float>::max()
                            (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MaxValue(?<after>\W]
225
                                 )"), "${before}std::numeric_limits<${type}>::max()${after}",
                                 0),
                           // using Platform.Numbers;
226
                            //
                            (new Regex(0"([\r\n]{2}|^)\s*?using [\.a-zA-Z0-9]+;\s*?$"), "", 0),
228
                           // struct TreeElement { }
229
                           // struct TreeElement { };
230
                           (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([^;])"), "$1
231
                                 $2$3{$4};$5", 0),
                           // class Program {
232
                            // class Program { }
233
                            (new Regex(0"(struct|class) ([a-zA-Z0-9]+[^r]*)([^r]+(?<indentLevel>[\t
                                  ]*)?)\{([\S\s]+?[\r\n]+\k<indentLevel>)\}([^;]|$)"), "$1 $2$3{$4};$5", 0),
                           // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
235
                           // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
236
                            (\text{new Regex}(@"class})([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)"), "class $1 : public $2", 0),
237
                           // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
                           // class IProperty : public ISetter<TValue, TObject>, IProvider<TValue, TObject> (new Regex(@"(?<before>class [a-zA-ZO-9]+ : ((public [a-zA-ZO-9]+(<[a-zA-ZO-9]+)) | ((public [a-zA-ZO-9]+)) | ((public 
239
240
                                  ,]+>)?)(?(after)(, [a-zA-Z0-9]+(?!>)|[ \r\n]+))"), "${before}public
                                  ${inheritedType}${after}", 10),
                            // Insert scope borders.
                               ref TElement root
242
                           // ~!root!~ref TElement root
243
                            (\text{new Regex}(0"(?<\text{definition}>(?<= |\setminus()(\text{ref }[a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!\text{ref})))))
244
                                  (?\langle variable \rangle [a-zA-Z0-9]+)(?= \rangle |, | = ))"), "^! \{ variable \}!^{ \{definition\}", 0 \}, }
                           // Inside the scope of ~!root!~ replace:
                           // root
246
                            // *root
247
                            (\text{new Regex}(@"(?<\text{definition}>^!(?<\text{pointer})[a-zA-Z0-9]+)!^ref [a-zA-Z0-9]+)
                                   \k<pointer>(?=\)|, | =))(?<before>((?<!~!\k<pointer>!~)(.|\n))*?)(?<prefix>(\W
                                  |\())\k<pointer>(?<suffix>( |\)|;|
                                                                                              ,))"),
                                  "${definition}${before}${prefix}*${pointer}${suffix}", 70),
                           // Remove scope borders.
249
                           // ~!root!~
250
                           //
                           (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", 5),
252
                           // ref auto root = ref
253
                           // ref auto root
254
                            (\text{new Regex}(@"\text{ref}([a-zA-Z0-9]+)([a-zA-Z0-9]+) = \text{ref}(\W)"), "$1* $2 =$3", 0),
                           // *root = ref left;
256
                           // root = left;
257
                            (\text{new Regex}(@"\*([a-zA-Z0-9]+) = \text{ref}([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", 0),
                           // (ref left)
259
                           // (left)
260
                            (\text{new Regex}(@"\(\text{ref}([a-zA-Z0-9]+)(\)|\(|,)"), "($1$2", 0),
261
                           // ref TElement
262
```

```
TElement*
263
                  (new Regex(0"( |\cdot|) ref ([a-zA-Z0-9]+) "), "$1$2* ", 0),
                 // ref sizeBalancedTree.Root
265
                 // &sizeBalancedTree->Root
266
                 (new Regex(0"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)"), "&1->2", 0),
                 // ref GetElement(node).Right
268
                 // &GetElement(node)->Right
269
                 (\text{new Regex}(0"\text{ref}([a-zA-Z0-9]+))(([a-zA-Z0-9]*)+))).([a-zA-Z0-9]+)"),
270
                     "&$1($2) ->$3", 0),
                 // GetElement(node).Right
                 // GetElement(node) ->Right
272
                 (\text{new Regex}(@"([a-zA-Z0-9]+))(([a-zA-Z0-9]+))).([a-zA-Z0-9]+)"), "$1($2)->$3", 0),
273
                 // [Fact̄]\npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
                 // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
275
                 (\text{new Regex}(0"\setminus[\text{Fact}][\s\n]+(\text{public}:)?(\text{static})?\text{void}([a-zA-Z0-9]+)\(\)"), "public:
276
                     TEST_METHOD(\$3)", 0),
                 // class TreesTests
277
                  // TEST_CLASS(TreesTests)
                  (new Regex(@"class ([a-zA-Z0-9]+)Tests"), "TEST_CLASS($1)", 0),
                 // Assert.Equal
280
                 // Assert::AreEqual
281
                 (new Regex(@"(Assert)\.Equal"), "$1::AreEqual", 0),
                 // Assert.Throws
283
                 // Assert::ExpectException
284
                  (new Regex(@"(Assert)\.Throws"), "$1::ExpectException", 0),
285
                     $"Argument {argumentName} is null."
286
                 // ((std::string) "Argument ").append(argumentName).append(" is null.").data()
287
                 (\text{new Regex}(@'')^{"''}(?<\text{left}>()''''|[^"''])*){(?<\text{expression}=a-zA-Z0-9]+)}(?<\text{right}>()_1
288
                      \""|[^""\r\n])*)""")
                      \hookrightarrow
                     10),
                 // $"
289
                 // "
                 (new Regex(@"\$"""), "\"";
291
                 // Console.WriteLine("...")
292
                 // printf("...\n")
293
                 (new Regex(@"Console\.WriteLine\(""([^""\r\n]+)""\)"), "printf(\"$1\\n\")", 0),
                 // TElement Root;
295
                    TElement Root = 0;
296
                  (new Regex(@"(\r?\n[\t]+)(private|protected|public)?(:
                     )?([a-zA-Z0-9:_]+(?<!return)) ([_a-zA-Z0-9]+);"), "$1$2$3$4 $5 = 0;", 0),
                 // TreeElement _elements[N];
                 // TreeElement _elements[N] = { {0} };
299
                 (new Regex(@"(\r?\n[\t]+)(private|protected|public)?(: )?([a-zA-Z0-9]+)
300
                      ([_a-zA-Z0-9]+)\setminus[([_a-zA-Z0-9]+)\setminus];"), "$1$2$3$4 $5[$6] = { {0} };", 0),
                 // auto path = new TElement[MaxPath];
                 // TElement path[MaxPath] = { {0} }
302
                 (\text{new Regex}(@"(\r?\n[\t]+)[a-zA-Z0-9]+ ([a-zA-Z0-9]+) = \text{new})
303
                       ([a-zA-Z0-9]+) \setminus [([_a-zA-Z0-9]+) \setminus ];"), "$1$3 $2[$4] = { {0} };", 0), 
                 // bool Equals(Range<T> other) { ... }
304
                  // bool operator ==(const Key &other) const {
                 (new Regex(0"(?<before>\r?\n[^\n]+bool )Equals\((?<type>[^\n{]+)
306
                      (?variable>[a-zA-Z0-9]+))(?<after>(\s|\n)*{})"), "${before}operator ==(const)
                     $\{\type\} &\{\variable\}\) const\{\(\artarrow\) after\}", 0)
                 // private: static readonly ConcurrentBag<std::exception> _exceptionsBag = new
307
                     ConcurrentBag<std::exception>();
                 // private: inline static std::mutex _exceptionsBag_mutex; \n\n private: inline
308
                 \label{lem:bag} \begin{array}{ll} \Rightarrow & static \ std::vector < std::exception > \_exceptions Bag; \\ (new \ Regex(@"(? < begin > \r?\n?(? < indent > [ \t] +))(? < access > (private | protected | public): \\ \end{array}
309
                      )?static readonly ConcurrentBag<(?<argumentType>[^;\r\n]+)>
                      (?\langle name \rangle [_a-zA-z0-9]+) = new ConcurrentBag\langle \langle argumentType \rangle \rangle ();"),
                      "${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() {
310
                     return _exceptionsBag; }
                 // public: static std::vector<std::exception> GetCollectedExceptions() { return
311

    std::vector<std::exception>(_exceptionsBag); }

                  (new Regex(@"(?<access>(private|protected|public): )?static
                     { return (?<fieldName>[_a-zA-Z0-9]+); }"),
                                                                    "${access}static
                     std::vector<${argumentType}> ${methodName}() { return
                     std::vector<${argumentType}>(${fieldName}); }", 0),
                 // public: static event EventHandler<std::exception> ExceptionIgnored =
                     OnExceptionIgnored; ... };
```

```
// ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
314
                            const std::exception&)> ExceptionIgnored = OnExceptionIgnored; };
                           (new Regex(@"(?<begin>\r?\n(\r?\n)?(?<halfIndent>[
315
                                  \t]+)\k<halfIndent>)(?<access>(private|protected|public): )?static event
                                 ${middle}" + Environment.NewLine + Environment.NewLine +
                                 "${halfIndent}${halfIndent}${access}static inline
                                 Platform::Delegates::MulticastDelegate<void(void*, const ${argumentType}&)>
                                 ${name} = ${defaultDelegate};${end}", 0),
                          // Insert scope borders.
316
                          // class IgnoredExceptions { ... private: inline static std::vector<std::exception>
317
                                  _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
                                 ]*{)(?<middle>((?!class).|\n)+?)(?<vectorFieldDeclaration>(?<access>(private|pro_
                                 tected | public): )inline static std::vector<(?<argumentType>[^;\r\n]+)>
                                 (?<fieldName>[_a-zA-Z0-9]+);)"),
"${classDeclarationBegin}/*~${fieldName}~*/${middle}${vectorFieldDeclaration}",
                                 0),
                          // Inside the scope of ~!_exceptionsBag!~ replace:
320
                          // _exceptionsBag.Add(exception);
// exceptionsPag.add(exception);
                                _exceptionsBag.push_back(exception);
322
                           (new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<befor_</pre>
323
                                 e > ((?<!/*^k<fieldName>^**/)(.|n))*?)k<fieldName>^.Add"),
                                 "${scope}${separator}${before}${fieldName}.push_back", 10),
                          // Remove scope borders.
324
                              /*~_exceptionsBag~*/
325
                           (new Regex(0"/\*^[_a-zA-Z0-9]+^\*/"), "", 0),
327
                          // Insert scope borders.
328
                          // class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
// class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: static std::mutex
330
                                 _exceptionsBag_mutex;
                           (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t]*)class [^{\r\n]+\r\n[\t
331
                                 ]*{)(?<middle>((?!class).|\n)+?)(?<mutexDeclaration>private: inline static
                                 // Inside the scope of ~!_exceptionsBag!~ replace:
332
                          // return std::vector<std::exception>(_exceptionsBag);
                          // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
334

    std::vector<std::exception>(_exceptionsBag);
                           (\text{new Regex}(@"(?<\text{scope})/*^{(?<\text{fieldName})}[_a-zA-Z0-9]+)^{*})(?<\text{separator}.|\n)(?<\text{befor})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{separator})(?<\text{sep
335
                                 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);${after}", 10),
Inside the scope of ~!_exceptionsBag!~ replace:
336
                                _exceptionsBag.Add(exception);
                          // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
338
                                  _exceptionsBag.Add(exception);
                           (new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<befor_</pre>
339
                                 e > ((?<!/*^k<fieldName>^*)(.|n))*?){(?<after>((?!lock_guard)([^{};]|n))*?}r_|
                                  ?\n(?<indent>[ \t]*)\k<fieldName>[^;}\r\n]*;)")
                                 "${scope}${separator}${before}{" + Environment.NewLine +
                                 "${indent}std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", 10),
                          // Remove scope borders.
340
                          // /*~_exceptionsBag~*/
341
342
                           (new Regex(0"/\*^[_a-zA-Z0-9]+^*\*/"), "", 0),
343
                          // Insert scope borders.
344
                          // class IgnoredExceptions { ... public: static inline
345
                                Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
                                ExceptionIgnored = OnExceptionIgnored;
                          // class IgnoredExceptions {/*~ExceptionIgnored~*/ ... public: static inline
346
                                 Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
                                 ExceptionIgnored = OnExceptionIgnored;
                           (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t]*)class [^{\r\n]+\r\n[\t
347
                                 ]*{)(?<middle>((?!class).|\n)+?)(?<eventDeclaration>(?<access>(private|protected_
                                 |public): )static inline
                                 Platform::Delegates::MulticastDelegate<(?<argumentType>[^;\r\n]+)>
                                  (?<name>[_a-zA-Z0-9]+) = (?<defaultDelegate>[_a-zA-Z0-9]+);)"),
                                 "${classDeclarationBegin}/*~${name}~*/${middle}${eventDeclaration}", 0),
                          // Inside the scope of "!ExceptionIgnored!" replace:
                          // ExceptionIgnored.Invoke(NULL, exception);
                          // ExceptionIgnored(NULL, exception);
350
```

```
(\text{new Regex}(@"(?<scope>/)*^(?<eventName>[a-zA-Z0-9]+)^*/)(?<separator>.|\n)(?<before_|
351
                               ((?<!/*^k<eventName>^**/)(.|n))*?)k<eventName>^.Invoke"),
                               "${scope}${separator}${before}${eventName}", 10),
                         // Remove scope borders.
352
                         // /*~ExceptionIgnored~*/
353
                         //
                         (new Regex(0"/*[a-zA-Z0-9]+^**/"), "", 0).
355
                         // Insert scope borders.
356
                             auto added = new StringBuilder();
                         // /*~sb~*/std::string added;
358
                         (new Regex(@"(auto|(System\.Text\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new
359
                               (System\.Text\.)?StringBuilder\(\);"), "/*~${variable}~*/std::string
                               ${variable}; ", 0),
                         // static void Indent(StringBuilder sb, int level)
360
                         // static void Indent(/*~sb~*/StringBuilder sb, int level)
                         (new Regex(@"(?<start>, |\()(System\.Text\.)?StringBuilder
362
                               (?<variable>[a-zA-Z0-9]+)(?<end>,|\))"), "${start}/*~${variable}~*/std::string&
                         363
                         // sb.ToString()
                         // sb.data()
365
                         (new Regex(0"(?<scope>/\*^(?<variable>[a-zA-Z0-9]+)^\*/)(?<separator>.|\n)(?<before>|
366
                               (((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.ToString\(\)"),
                               "${scope}${separator}${before}${variable}.data()", 10),
                         // sb.AppendLine(argument)
                         // sb.append(argument).append('\n')
                         (new Regex(0"(?<scope>/*(?<variable>[a-zA-Z0-9]+)^**/)(?<separator>.|\n)(?<before>|
369
                                r\n]+)\)")
                          \hookrightarrow
                               \label{thm:cope} $$\{separator\} \{before\} \{variable\}.append($\{argument\}).append(1, '\n')'', append(1, '\n')''', append(1, '\n')'', append(1, '\n')''', append(1, '\n')'', append(1, '\n')''', append(1, '\n
                          \hookrightarrow
                               10)
                         // sb.Append('\t', level);
370
                         // sb.append(level, '\t');
371
                         (\text{new Regex}(@"(?<scope>/)*^(?<variable>[a-zA-Z0-9]+)^**/)(?<separator>.|\n)(?<before>|
                               ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.Append\('(?<character>[^'\r\n]
                                     , (?<count>[^\),\r\n]+)\)")
                               "${scope}${separator}${before}${variable}.append(${count}, '${character}')", 10),
                         // sb.Append(argument)
373
                         // sb.append(argument)
                         (new Regex(0"(?<scope>/*"(?<variable>[a-zA-Z0-9]+)"\*/)(?<separator>.|\n)(?<before>|
375
                                ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.Append\((?<argument>[^\),\r\n]
                               +)\)", "${scope}${separator}${before}${variable}.append(${argument})",
                               10),
                         // Remove scope borders.
376
                         // /*~sb~*/
377
                         //
                         (new Regex(0"/\*^[a-zA-Z0-9]+^\*/"), "", 0),
379
                         // Insert scope borders.
380
                             auto added = new HashSet<TElement>();
381
                         // ~!added!~std::unordered_set<TElement>_ added;
                         (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
383
                               HashSet < (? < element > [a-zA-Z0-9] +) > ( ); "),
                               "~!${variable}!~std::unordered_set<${element}> ${variable};", 0),
                         // Inside the scope of ~!added!~ replace:
384
                         // added.Add(node)
385
386
                         // added.insert(node)
                         (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?<|</pre>
                               !^*[\bar{k}\leq 1] (.|\n))*?)\k<variable>\.Add\((?<argument>[a-zA-Z0-9]+)\)"),
                               "${scope}${separator}${before}${variable}.insert(${argument})", 10),
                         // Inside the scope of ~!added!~ replace:
388
                         // added.Remove(node)
389
                         // added.erase(node)
                         (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?< |</pre>
                               !~!\k<variable>!~)(.|\n))*?)\k<variable>\.Remove\((?<argument>[a-zA-Z0-9]+)\)"),
                               "${scope}${separator}${before}${variable}.erase(${argument})", 10),
                             if (added.insert(node)) {
392
                         // if (!added.contains(node)) { added.insert(node);
                         (new Regex(0"if \(((?\variable>[a-zA-Z0-9]+)\.insert\(((?\argument>[a-zA-Z0-9]+)\))\)(?_1
                               (!${variable}.contains(${argument}))${separator}${indent}{" +
                               Environment.NewLine + "${indent}
                                                                                        ${variable}.insert(${argument});", 0),
                         // Remove scope borders.
395
                         // ~!added!~
396
397
                         (new Regex(0"^{-1}[a-zA-Z0-9]+!^{-1}), "", 5),
398
                         // Insert scope borders.
399
```

```
// auto random = new System.Random(0);
400
                                    // std::srand(0);
                                    (\text{new Regex}(@"[a-zA-Z0-9]] + ([a-zA-Z0-9]] +) = \text{new}
402
                                             (System\.)?Random\(([a-zA-Z0-9]+)\);"), "~!$1!~std::srand($3);", 0),
                                    // Inside the scope of ~!random!~ replace:
403
                                    // random.Next(1, N)
404
                                    // (std::rand() % N) + 1
                                    (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?<|</pre>
406
                                             \hookrightarrow
                                             ${from}", 10),
                                    // Remove scope borders.
407
                                    // ~!random!^
408
                                    //
409
                                    (new Regex(0"^{-1}[a-zA-Z0-9]+!^{-1}), "", 5),
                                    // Insert method body scope starts.
411
                                    // void PrintNodes(TElement node, StringBuilder sb, int level) {
// void PrintNodes(TElement node, StringBuilder sb, int level) {/*method-start*/
412
413
                                    (new Regex(@"(?<start>\r?\n[\t]+)(?<prefix>((private|protected|public): )?(virtual)
414
                                             )?[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*/${end}",
                                            0),
                                    // Insert method body scope ends.
415
                                    // {/*method-start*/...}
416
                                    // {/*method-start*/.../*method-end*/}
417
                                    (new Regex(@"\{/\*method-start\*/(?<body>((?<bracket>\{)|(?<-bracket>\})|[^\{\}]*)+) |
                                             \"), "{/*method-start*/${body}/*method-end*/}",
                                            0)
                                    // Inside method bodies replace:
419
                                    // GetFirst(
420
                                    // this->GetFirst(
                                    //(new Regex(@"(?<separator>(\(|, |([\\]) |return ))(?<!(->|\*
422
                                            ))(?<method>(?!sizeof)[a-zA-Z0-9]+)\((?!\)\{)"),
                                             "${separator}this->${method}(", 1),
                                    (\texttt{new Regex}(@"(?<scope>/\\*method-start\\*/)(?<before>((?<!/\\*method-end\\*/)(.|\\n))*?)(_{|})()
                                             ?<separator>[\W](?<!(::|\.|->)))(?<method>(?!sizeof)[a-zA-Z0-9]+)\((?!\)
                                             \{\) (?\langle after\rangle(.|\n)*?) (?\langle scopeEnd\rangle/\method-end\*/)")
                                             \label{lem:cope} $$\{separator\}$ this->$\{method\}($\{after\}$\{scopeEnd\}", 100), for each of the context of the co
                                    // Remove scope borders.
424
                                    // /*method-start*/
425
                                    (new Regex(@"/\*method-(start|end)\*/"), "", 0),
427
                                    // Insert scope borders.
428
                                         const std::exception& ex
429
                                    // const std::exception& ex/*~ex~*/
430
                                    (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?exception&?
431
                                             (?<variable>[_a-zA-Z0-9]+))(?<after>\\\")
                                             "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
                                    // Inside the scope of ~!ex!~ replace:
432
                                    // ex.Message
433
434
                                    // ex.what()
                                    (new Regex(@"(?<scope>/\*~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before</pre>
435
                                            >((?<!/\*^\k<variable>^\*/)(.|\n))*?)\k<variable>\.Message"),
                                            "${scope}${separator}${before}${variable}.what()", 10),
                                    // Remove scope borders.
436
                                    // /*~ex~*/
437
                                    //
438
                                    (new Regex(0"/\*^[_a-zA-Z0-9]+^*\*/"), "", 0),
439
                                    // throw new ArgumentNullException(argumentName, message);
440
                                         throw std::invalid_argument(((std::string)"Argument
441
                                            ").append(argumentName).append(" is null: ").append(message).append("."));
                                    (new Regex(@"throw new
442
                                             ArgumentNullException\((?<argument>[a-zA-Z]*[Aa]rgument[a-zA-Z]*),
                                              (?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\))?)\);"), "throw "in the content of the content o
                                            std::invalid_argument(((std::string)\"Argument \").append(${argument}).append(\"
                                            is null: \").append(${message}).append(\".\"));"
                                    // throw new ArgumentException(message, argumentName);
443
                                    // throw std::invalid_argument(((std::string)"Invalid
444
                                            ").append(argumentName).append(" argument: ").append(message).append("."));
                                    (new Regex(@"throw new
                                            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;"), "throw"
                                            std::invalid_argument(((std::string)\"Invalid \").append(${argument}).append(\"
                                            argument: \").append(${message}).append(\".\"));", 0),
```

```
// throw new ArgumentOutOfRangeException(argumentName, argumentValue,
446
                               messageBuilder());
                         // throw std::invalid_argument(((std::string)"Value
                                [").append(std::to_string(argumentValue)).append("] of argument
                                [").append(argumentName).append("] is out of range:
                               ").append(messageBuilder()).append("."));
                          (new Regex(@"throw new ArgumentOutOfRangeException\((?<argument>[a-zA-Z]*[Aa]rgument]
                                [a-zA-Z]*([Nn]ame[a-zA-Z]*)?)
                                (?\langle argumentValue\rangle[a-zA-Z]*[Aa]rgument[a-zA-Z]*([VV]alue[a-zA-Z]*)?),
                                (?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\))?)\);"), "throw
                                std::invalid_argument(((std::string)\"Value
                                [\").append(std::to_string(${argumentValue})).append(\"] of argument
                                [\").append(${argument}).append(\"] is out of range:
                                \").append(${message}).append(\".\"));", 0),
                         // throw new NotSupportedException();
449
                         // throw std::logic_error("Not supported exception.");
450
                          (new Regex(@"throw new NotSupportedException\(\);"), "throw std::logic_error(\"Not
451
                               supported exception.\");", 0)
                         // throw new NotImplementedException();
452
                         // throw std::logic_error("Not implemented exception.");
453
                          (\texttt{new Regex}(\texttt{@"throw new NotImplementedException}\cdot{(`\);"), "throw std::logic\_error(\"NotImplementedException)\cdot{(`\);"), "throw std::logic\_error(\"NotImplementedException)\cdot{('\);"), "throw std::logic\_error(\"N

→ implemented exception.\");", 0),
                   }.Cast<ISubstitutionRule>().ToList();
455
456
                   public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
457
458
                         // ICounter<int, int> c1;
                         // ICounter<int, int>* c1;
460
                          (\text{new Regex}(@"(?\langle abstractType\rangle I[A-Z][a-zA-Z0-9]+(\langle [^{\rangle r]+\rangle)?})
461
                                (?<variable>[_a-zA-Z0-9]+);"), "${abstractType}* ${variable};", 0),
462
                              (expression)
                          // expression
                          (\text{new Regex}(@"((| )(([a-zA-Z0-9_{*:}]+)))(,| |;|))"), "$1$2$3", 0),
464
                         // (method(expression))
465
                         // method(expression)
466
                          (new Regex(@"(?<firstSeparator>(\()
                               ))\((?<method>[a-zA-Z0-9_\->\*:]+)\((?<expression>((?<parenthesis>\()|(?<-parent
                              |;|\)))"), "${firstSeparator}${method}(${expression})${lastSeparator}", 0),
                         // return ref _elements[node];
468
                          // return &_elements[node];
469
                          (\text{new Regex}(@"\text{return ref}([_a-zA-Z0-9]+))[([_a-zA-Z0-9]*]+))];"), "return &$1[$2];",
                               0)
                         // null
                         // nullptr
472
                          (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)null |</pre>
473
                                (?<after>\W)"), "${before}nullptr${after}",
                                10),
                         // default
                         // 0
475
                          (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)defa_</pre>
476
                               ult(?<after>\W)"), "${before}0${after}",
                               10)
                         // object x
477
                         // void *x
478
                          o]bject|System\.Object) (?<after>\w)"), "${before}void *${after}",
                               10),
                         // <object>
480
                         // <void*>
481
                           (\text{new Regex}(@"(?\before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(?<!_{||} ) ) 
482
                                \hookrightarrow
                               10),
                         // ArgumentNullException
483
                         // std::invalid_argument
484
                          (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(Sys |</pre>
485
                                tem\.)?ArgumentNullException(?<after>\W)"),
                               "${before}std::invalid_argument${after}", 10),
                         // struct Range<T> : IEquatable<Range<T>> {
                         // struct Range<T> {
487
                          (\text{new Regex}(@"(?<\text{before}>(\text{struct}|\text{class}) (?<\text{type}>[a-zA-Z0-9]+(<[^\n]+>)?)) :
488
                               // #region Always
489
                          (\text{new Regex}(@"(^|\r?\n)[ \t]*\#(\text{region}|\text{endregion})[^\r\n]*(\r?\n|\$)"), "", 0),
491
                         // //#define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
492
```

```
493
                 (new Regex(0"\/\/[\t]*\#define[\t]+[_a-zA-Z0-9]+[\t]*"), "", 0),
                 // #if USEARRAYPOOL\r\n#endif
495
496
                 (new Regex(0"#if [a-zA-Z0-9]+\s+\#endif"), "", 0),
                 // [Fact]
498
499
                 (new Regex(0"(?<firstNewLine>\r?\n|\A)(?<indent>[\t
500
                     ]+)\[[a-zA-Z0-9]+(\((?<expression>((?<parenthesis>\()|(?<-parenthesis>\))|[^{()}\r_
                     \n]*)+)(?(parenthesis)(?!)))))?][ \t]*(\r?\n\k<indent>)?"),
                     "${firstNewLine}${indent}", 5),
                 // \n ... namespace
                 // namespace
502
                 (new Regex(0"(S[\r]{1,2})?[\r]+namespace", "$1namespace", 0),
503
504
                 // \n ... class
                 // class
505
                 (new Regex(0"(S[\r\n]{1,2})?[\r\n]+class"), "$1class", 0),
506
                 // \n\n
507
                 // \n
508
                 (new Regex(@"\r?\n[ \t]*\r?\n"), Environment.NewLine, 50),
509
             }.Cast<ISubstitutionRule>().ToList();
510
511
            public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
512
             → base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
            public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
514
        }
515
516
1.2
     ./csharp/Platform.Regular Expressions.Transformer.CSharp ToCpp.Tests/CSharp ToCpp Transformer Tests.cs
    using Xunit;
    namespace Platform.RegularExpressions.Transformer.CSharpToCpp.Tests
 3
        public class CSharpToCppTransformerTests
 5
             [Fact]
            public void EmptyLineTest()
                 // This test can help to test basic problems with regular expressions like incorrect
10
                 var transformer = new CSharpToCppTransformer();
                 var actualResult = transformer.Transform("");
12
                 Assert.Equal("", actualResult);
13
             }
14
15
             [Fact]
16
             public void HelloWorldTest()
17
18
                 const string helloWorldCode = @"using System;
    class Program
20
21
        public static void Main(string[] args)
22
23
             Console.WriteLine(""Hello, world!"");
24
25
    }";
26
                 const string expectedResult = @"class Program
27
28
        public: static void Main(const char* args[])
29
30
            printf(""Hello, world!\n"");
31
32
    }:":
33
                 var transformer = new CSharpToCppTransformer();
                 var actualResult = transformer.Transform(helloWorldCode);
35
                 Assert.Equal(expectedResult, actualResult);
36
             }
37
        }
38
```

39 }

Index

./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs, 11 ./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1