

LinksPlatform's Platform.RegularExpressions.Transformer.CSharpToCpp Class Library

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

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text.RegularExpressions;
5
6  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
7
8  namespace Platform.RegularExpressions.Transformer.CSharpToCpp
9  {
10     /// <summary>
11     /// <para>
12     /// Represents the sharp to cpp transformer.
13     /// </para>
14     /// <para></para>
15     /// </summary>
16     /// <seealso cref="TextTransformer"/>
17     public class CSharpToCppTransformer : TextTransformer
18     {
19         /// <summary>
20         /// <para>
21         /// The to list.
22         /// </para>
23         /// <para></para>
24         /// </summary>
25         public static readonly IList<ISubstitutionRule> FirstStage = new List<SubstitutionRule>
26     {
27         // // ...
28         //
29         (new Regex(@"(\r?\n)?[ \t]+//.+"), "", 0),
30         // #pragma warning disable CS1591 // Missing XML comment for publicly visible type
31         // or member
32         (new Regex(@"^s*?#pragma[\sa-zA-Z0-9]+"), "", 0),
33         // {\n\n\n
34         //
35         (new Regex(@"{s+[\r\n]+"), "{" + Environment.NewLine, 0),
36         // Platform.Collections.Methods.Lists
37         // Platform::Collections::Methods::Lists
38         (new Regex(@"(namespace[^r\n]+?)\.([^r\n]+?)"), "$1:$2", 20),
39         // nameof(numbers)
40         // "numbers"
41         (new
42             Regex(@"(?<before>\W)nameof\(((^\n)+\.)?(?<name>[a-zA-Z0-9_]+)(<^\n>+)?\)", //,
43             "${before}" + ${name} "", 0),
44             // Insert markers
45             // EqualityComparer<T> _equalityComparer = EqualityComparer<T>.Default;
46             // EqualityComparer<T> _equalityComparer =
47             // EqualityComparer<T>.Default; /*~_comparer*/
48             (new Regex(@"(?<declaration>EqualityComparer<(?:type>[^>\n]+)>
49             (?<comparer>[a-zA-Z0-9_]+) = EqualityComparer<k<type>>\.Default;)", //,
50             "${declaration}/*~${comparer}~*/", 0),
51             // /*~_equalityComparer~*..._equalityComparer.Equals(Minimum, value)
52             // /*~_equalityComparer~*...Minimum == value
53             (new Regex(@"(?<before>/\*~(?<comparer>[a-zA-Z0-9_]+)\*/(.|\n)+\W)\k<comparer>\.Equ
54             als\((?<left>[^,\n]+, (?<right>[^,\n]+)\))", "${before}${left} == ${right}", //,
55             50),
56             // Remove markers
57             // /*~_equalityComparer~*/
58             //
59             (new Regex(@"\r?\n[^n]+\*/[^a-zA-Z0-9_]+\*/", "", 10),
60             // Insert markers
61             // Comparer<T> _comparer = Comparer<T>.Default;
62             // Comparer<T> _comparer = Comparer<T>.Default; /*~_comparer*/
63             (new Regex(@"(?<declaration>Comparer<(?:type>[^>\n]+)> (?<comparer>[a-zA-Z0-9_]+) =
64             Comparer<k<type>>\.Default;)", "${declaration}/*~${comparer}~*/", 0),
65             // /*~_comparer~*..._comparer.Compare(Minimum, value) <= 0
66             // /*~_comparer~*...Minimum <= value
67             (new Regex(@"(?<before>/\*~(?<comparer>[a-zA-Z0-9_]+)\*/(.|\n)+\W)\k<comparer>\.Com
68             pare\((?<left>[^,\n]+,
69             (?<right>[^,\n]+)\)\s*(?<comparison>[>|=])\s*0(?<after>\D)", //,
70             "${before}${left} ${comparison} ${right}${after}", 50),
71             // Remove markers
72             // private static readonly Comparer<T> _comparer =
73             // Comparer<T>.Default; /*~_comparer*/
74             //
75         }
76     }
77 }
```

```

63   (new Regex(@"\r?\n[^\\n]+/*[a-zA-Z0-9_]+/*"), "", 10),
64   // Comparer<TArgument>.Default.Compare(maximumArgument, minimumArgument) < 0
65   // maximumArgument < minimumArgument
66   (new Regex(@"Comparer<[^>\\n]>.Default.Compare\\((\\s*(?<first>[^,]\\n)+),\\s*(?<second>
67   >[^\\n]+)\\s*)\\s*(?<comparison>[<>]=?)\\s*0(?<after>\\D)", "{$first}
68   ${comparison} ${second}${after}", 0),
69   // public static bool operator ==(Range<T> left, Range<T> right) =>
70   left.Equals(right);
71   //
72   (new Regex(@"\r?\n[^\\n]+bool operator ==\\(((?<type>[^\\n]+) (?<left>[a-zA-Z0-9]+),
73   \k<type> (?<right>[a-zA-Z0-9]+)) =>
74   (\k<left>|\\k<right>)\\ Equals\\((\\k<left>|\\k<right>)\\);", "", 10),
75   // public static bool operator !=(Range<T> left, Range<T> right) => !(left == right);
76   //
77   (new Regex(@"\r?\n[^\\n]+bool operator !=\\(((?<type>[^\\n]+) (?<left>[a-zA-Z0-9]+),
78   \k<type> (?<right>[a-zA-Z0-9]+)) => !\\((\\k<left>|\\k<right>) ==
79   (\k<left>|\\k<right>)\\);", "", 10),
80   // public override bool Equals(object obj) => obj is Range<T> range ? Equals(range)
81   : false;
82   //
83   (new Regex(@"\r?\n[^\\n]+override bool Equals\\((System\\.)?[0o]bject
84   (?<this>[a-zA-Z0-9]+)\\) => \\k<this> is [^\\n]+ (?<other>[a-zA-Z0-9]+) \\?
85   Equals\\(\\k<other>) : false;", "", 10),
86   // out TProduct
87   // TProduct
88   (new Regex(@"(?<before>(<|, ))(in|out)
89   (?<typeParameter>[a-zA-Z0-9]+)(?<after>(>|, )), "
90   "${before}${typeParameter}${after}", 10),
91   // public ...
92   // public: ...
93   (new Regex(@"(?<newLineAndIndent>\\r?\\n?[
94   \\t]*)(?<before>[^\\{\\(\\r\\n)*](?<access>private|protected|public)[ \\t]+(?![^\\{\\(\\r\\n)*
95   \\n]*((?<=\\s) |\\W)(interface|class|struct)(\\W)[^\\{\\(\\r\\n)*[\\{\\(\\r\\n)]"),
96   "${newLineAndIndent}${access}: ${before}", 0),
97   // public: static bool CollectExceptions { get; set; }
98   // public: inline static bool CollectExceptions;
99   (new Regex(@"(?<access>(private|protected|public): )(?<before>(static )?[\\r\\n]+
100   )(?<name>[a-zA-Z0-9]+) {[^;]*(?<=\\W)get;[^;]*(?<=\\W)set;[^;]*}"),
101   "${access}inline ${before}${name};", 0),
102   // public abstract class
103   // class
104   (new Regex(@"((public|protected|private|internal|abstract|static)
105   )*(?<category>interface|class|struct)", "${category}", 0),
106   // class GenericCollectionMethodsBase<TElement> {
107   // template <typename TEElement> class GenericCollectionMethodsBase {
108   (new Regex(@"(?<before>\\r?\\n)(?<indent>[ \\t]*)(?<type>class|struct)
109   (?<typeName>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9
110   ,]+)>(?<typeDefinitionEnding>[^{}]+)", "${before}${indent}template <typename
111   ...> ${type} ${typeName};" + Environment.NewLine + "${indent}template <typename
112   ${typeParameters}> ${type}
113   ${typeName}<${typeParameters}>${typeDefinitionEnding}{", 0),
114   // static void
115   TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
116   tree, TEElement* root)
117   // template<typename T> static void
118   TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
119   tree, TEElement* root)
120   (new Regex(@"static ([a-zA-Z0-9]+) ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>\\(([\\r\\n]+)\\)", "
121   "template <typename $3> static $1 $2($4)", 0),
122   // interface IFactory<out TProduct> {
123   // template <typename...> class IFactory; \\ntemplate <typename TProduct> class
124   IFactory<TProduct>
125   (new Regex(@"(?<before>\\r?\\n)(?<indent>[ \\t]*)interface
126   (?<interface>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9
127   ,]+)>(?<typeDefinitionEnding>[^{}]+)", "${before}${indent}template <typename
128   ...> class ${interface};" + Environment.NewLine + "${indent}template <typename
129   ${typeParameters}> class
130   ${interface}<${typeParameters}>${typeDefinitionEnding}{" + Environment.NewLine +
131   "    public:", 0),
132   // template <typename TObject, TProperty, TValue>
133   // template <typename TObject, typename TProperty, typename TValue>
134   (new Regex(@"(?<before>template <((, )?typename [a-zA-Z0-9]+),
135   (?<typeParameter>[a-zA-Z0-9]+)(?<after>(|>))", "${before}typename
136   ${typeParameter}${after}", 10),
137   // Insert markers

```

```

101 // private: static void BuildExceptionString(this StringBuilder sb, Exception
102   exception, int level)
103 // /*extensionMethod~BuildExceptionString~*/private: static void
104   BuildExceptionString(this StringBuilder sb, Exception exception, int level)
105 (new Regex(@"private: static [^\r\n]+ (?<name>[a-zA-Z0-9]+)\((this [^\\]\r\n)+\)", 
106   "//*extensionMethod~${name}~*/$0", 0),
107 // Move all markers to the beginning of the file.
108 (new Regex(@"\A(?<before>[^\\r\\n]+\\r?\\n(.|\\n+))(?<marker>/\*~extensionMethod~(?<name>
109   [a-zA-Z0-9]+)~*/)", "${marker}${before}", 
110   10),
111 // /*~extensionMethod~BuildExceptionString~*/...sb.BuildExceptionString(exception.In_
112   nerException, level +
113   1);
114 // /*~extensionMethod~BuildExceptionString~*/...BuildExceptionString(sb,
115   exception.InnerException, level + 1);
116 (new Regex(@"(?<before>/\*~extensionMethod~(?<name>[a-zA-Z0-9]+)~*/(.|\\n+\\W)(?<var_
117   iable>[_a-zA-Z0-9]+)\\.\\k<name>\\("), "${before}${name}(${variable}, ",
118   50),
119 // Remove markers
120 // /*~extensionMethod~BuildExceptionString~*/
121 // 
122 (new Regex(@"/\*~extensionMethod~[a-zA-Z0-9]+~*/"), "", 0),
123 // (this
124 // (
125 (new Regex(@"(this ", "(", 0),
126 // private: static readonly Disposal _emptyDelegate = (manual, wasDisposed) => { };
127 // private: inline static std::function<Disposal> _emptyDelegate = [](auto manual,
128   auto wasDisposed) { };
129 (new Regex(@"(?<access>(private|protected|public): )?static readonly
130   (?<type>[a-zA-Z][a-zA-Z0-9]*) (?<name>[a-zA-Z_][a-zA-Z0-9_]*) =
131   \((?<firstArgument>[a-zA-Z_][a-zA-Z0-9_]*),
132   (?<secondArgument>[a-zA-Z_][a-zA-Z0-9_]*)\) => {\s*};"), "${access}inline static
133   std::function<${type}> ${name} = [](auto ${firstArgument}, auto
134   ${secondArgument}) { };", 0),
135 // public: static readonly EnsureAlwaysExtensionRoot Always = new
136   EnsureAlwaysExtensionRoot();
137 // public: inline static EnsureAlwaysExtensionRoot Always;
138 (new Regex(@"(?<access>(private|protected|public): )?static readonly
139   (?<type>[a-zA-Z0-9]+(<[a-zA-Z0-9]+>)?) (?<name>[a-zA-Z0-9_+] = new
140   \k<type>\(\);"), "${access}inline static ${type} ${name};", 0),
141 // public: static readonly Range<int> SByte = new
142   Range<int>(std::numeric_limits<int>::min(), std::numeric_limits<int>::max());
143 // public: inline static Range<int> SByte =
144   Range<int>(std::numeric_limits<int>::min(), std::numeric_limits<int>::max());
145 (new Regex(@"(?<access>(private|protected|public): )?static readonly
146   (?<type>[a-zA-Z0-9]+(<[a-zA-Z0-9]+>)?) (?<name>[a-zA-Z0-9_+] = new
147   \k<type>\(((?<arguments>[^\\n]+)\)\);"), "${access}inline static ${type} ${name} =
148   ${type}(${arguments});", 0),
149 // public: static readonly string ExceptionContentsSeparator = "---";
150 // public: inline static std::string ExceptionContentsSeparator = "---";
151 (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly) string
152   (?<name>[a-zA-Z0-9_+] = ""(?<string>(\\"|[^\\\"\\r\\n])+"");"), "${access}inline
153   static std::string ${name} = "${string}";", 0),
154 // private: const int MaxPath = 92;
155 // private: inline static const int MaxPath = 92;
156 (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly)
157   (?<type>[a-zA-Z0-9_+)(?<name>[_a-zA-Z0-9_+)= (?<value>[^;\\r\\n]+);"),
158   "${access}inline static const ${type} ${name} = ${value};", 0),
159 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
160   TArgument : class
161 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
162 (new Regex(@"(?<before> [a-zA-Z]+(([a-zA-Z *,], |))(?<type>[a-zA-Z]+)(?<after>(|_
163   [a-zA-Z *,],)) [ \\r\\n]+where \\k<type> : class"), "${before}${type}*${after}", 
164   0),
165 // protected: abstract TElement GetFirst();
166 // protected: virtual TElement GetFirst() = 0;
167 (new Regex(@"(?<access>(private|protected|public): )?abstract
168   (?<method>[^;\\r\\n]+);"), "${access}virtual ${method} = 0;", 0),
169 // TElement GetFirst();
170 // virtual TElement GetFirst() = 0;
171 (new Regex(@"(?<before>[\\r\\n]+[ ]+)(?<methodDeclaration>(?!return)[a-zA-Z0-9]+
172   [a-zA-Z0-9]+\\([^\]\\r\\n]*\\)) (?<after>[ ]*[\\r\\n]+)", "${before}virtual
173   ${methodDeclaration} = 0${after}", 1),
174 // protected: readonly TreeElement[] _elements;
175 // protected: TreeElement _elements[N];

```

```

142   (new Regex(@"(?<access>(private|protected|public): )?readonly
143     (?<type>[a-zA-Z<>0-9]+)([\[\]]+) (?<name>[_a-zA-Z0-9+];"), "${access}${type}
144     ${name}[N; ", 0),
145   // protected: readonly TElement Zero;
146   // protected: TElement Zero;
147   (new Regex(@"(?<access>(private|protected|public): )?readonly
148     (?<type>[a-zA-Z<>0-9]+) (?<name>[_a-zA-Z0-9+];"), "${access}${type} ${name};",
149     0),
150   // internal
151   //
152   (new Regex(@"(\W)internal\s+"), "$1", 0),
153   // static void NotImplementedException(ThrowExtensionRoot root) => throw new
154   // NotImplementedException();
155   // static void NotImplementedException(ThrowExtensionRoot root) { return throw new
156   // NotImplementedException(); }
157   (new Regex(@"(^s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static
158   // )?(override )?([a-zA-Z0-9]+
159   // )([a-zA-Z0-9]+)\(([^\(\r\n\*)])\s+=>\s+throw([^\r\n+);",
160   // $"$1$2$3$4$5$6$7$8($9) { throw$10; }", 0),
161   // SizeBalancedTree(int capacity) => a = b;
162   // SizeBalancedTree(int capacity) { a = b; }
163   (new Regex(@"(^s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static
164   // )?(override )?(void )?([a-zA-Z0-9]+)\(([^\(\r\n\*)])\s+=>\s+([^\r\n+);",
165   // $"$1$2$3$4$5$6$7$8($9) { $10; }", 0),
166   // int SizeBalancedTree(int capacity) => a;
167   // int SizeBalancedTree(int capacity) { return a; }
168   (new Regex(@"(^s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static
169   // )?(override )?([a-zA-Z0-9]+
170   // )([a-zA-Z0-9]+)\(([^\(\r\n\*)])\s+=>\s+([^\r\n+);",
171   // $"$1$2$3$4$5$6$7$8($9) { return $10; }", 0),
172   // OnDispose = (manual, wasDisposed) =>
173   // OnDispose = [&](auto manual, auto wasDisposed)
174   (new Regex(@"(?<variable>[a-zA-Z_][a-zA-Z0-9_]*)(?<operator>\s*+\?=*\s*)\((?<firstArg>
175   // ument>[a-zA-Z_][a-zA-Z0-9_]*),
176   // (?<secondArgument>[a-zA-Z_][a-zA-Z0-9_]*))\s*=>"),
177   // $"${variable}${operator}& (auto ${firstArgument}, auto ${secondArgument})", 0),
178   // () => Integer<TElement>.Zero,
179   // () { return Integer<TElement>.Zero; },
180   (new Regex(@"(\(\)\s+=>\s+(?<expression>[^(),;\r\n]+(\(((?<parenthesis>\()|(?<-parent
181   // hesis>\())|[^(),;\r\n]*?\)*?\))?[^\(\),;\r\n]*)(?<after>,|\))", "()" { return
182   // ${expression}; }${after}", 0),
183   // ~DisposableBase() => Destruct();
184   // ~DisposableBase() { Destruct(); }
185   (new Regex(@"~(?<class>[a-zA-Z_][a-zA-Z0-9_]*)(\(\)\s+=>\s+([^\r\n+?);",
186   // $"~${class}() { $1; }", 0),
187   // => Integer<TElement>.Zero;
188   // { return Integer<TElement>.Zero; }
189   (new Regex(@"()\s+=>\s+([^\r\n+?);"), " ") { return $1; }, 0),
190   // () { return avlTree.Count; }
191   // [&] ()-> auto { return avlTree.Count; }
192   (new Regex(@"(?<before>, |\(\)\(\) { return (?<expression>[^;\r\n+]); }"),
193   // $"${before}& ()-> auto { return ${expression}; }", 0),
194   // Count => GetSizeOrZero(Root);
195   // Count() { return GetSizeOrZero(Root); }
196   (new Regex(@"(\W)([A-Z][a-zA-Z_]+)\s+=>\s+([^\r\n+);"), $"$1$2() { return $3; }", 0),
197   // Insert scope borders.
198   // interface IDisposable { ... }
199   // interface IDisposable /*~start~interface~IDisposable~*/
200   // /*~end~interface~IDisposable~*/
201   (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)interface[\t
202   // ]*(?<type>[a-zA-Z][a-zA-Z0-9]*(<[^>\n]*>))?[^\{\}]*{}(?<middle>(.|\n*))(?<beforeE
203   // nd>(?<=r\n)\k<indent>)(?<end>))",
204   // $"${classDeclarationBegin}/*~start~interface~${type}~*/${middle}${beforeEnd}/*~en
205   // d~interface~${type}~*/${end}", 0),
206   // Inside the scope replace:
207   // /*~start~interface~IDisposable~*/ ... bool IsDisposed { get; } ...
208   // /*~end~interface~IDisposable~*/
209   // /*~start~interface~IDisposable~*/ ... virtual bool IsDisposed() = 0;
210   // /*~end~interface~IDisposable~*/
211   (new Regex(@"(?<before>(?<typeScopeStart>/\/*~start~interface~(?<type>[^~\n\*]+)\~\*/)
212   // (.|\n)?)(?<propertyDeclaration>(?<access>(private|protected|public):
213   // )?(<propertyType>[a-zA-Z_][a-zA-Z0-9_:<>]*)(?<property>[a-zA-Z_][a-zA-Z0-9_]*)
214   // (?<blockOpen>[\n\s]*{[\n\s]*}(\[[^\n]+]\)[\n\s]*)?get;(?<blockClose>[\n\s*]))(?<
215   // after>(.|\n)?)(?<typeScopeEnd>/\/*~end~interface~\k<type>~\*/))",
216   // $"${before}virtual ${propertyType} ${property}() = 0; ${after}", 20),

```

```

184 // Remove scope borders.
185 // /*~start~interface~IDisposable~*/
186 //
187 (new Regex(@"/*~[^~*\n]+(~[^~*\n]+)*~*/"), "", 0),
188 // public: T Object { get; }
189 // public: const T Object;
190 (new Regex(@"(?<before>[^r]\r?\n[ \t]*) (?<access>(private|protected|public):
191     )?(?<type>[a-zA-Z_][a-zA-Z0-9_:>]*)
192     (?<property>[a-zA-Z_][a-zA-Z0-9_*] (?<blockOpen>[\n\s]*{[\n\s]*}(\[[^\n]+\] [\n\s]
193     ]*)?get; (?<blockClose>[\n\s]*)(?<after>[\n\s*])", "${before}${access}const
194     ${type} ${property};${after}", 2),
195     // public: bool IsDisposed { get => _disposed > 0; }
196     // public: bool IsDisposed() { return _disposed > 0; }
197     (new Regex(@"(?<before>[^r]\r?\n[ \t]*) (?<access>(private|protected|public):
198         )?(?<virtual>virtual )?bool
199         (?<property>[a-zA-Z_][a-zA-Z0-9_*] (?<blockOpen>[\n\s]*{[\n\s]*}(\[[^\n]+\] [\n\s]
200         ]*)?get\$*=>\$*(?<expression>[^n]+); (?<blockClose>[\n\s*]{[\n\s*})",
201         "${before}${access}${virtual}bool ${property}() ${blockOpen}return
202         ${expression};${blockClose}", 2),
203     // protected: virtual std::string ObjectName { get => GetType().Name; }
204     // protected: virtual std::string ObjectName() { return GetType().Name; }
205     (new Regex(@"(?<before>[^r]\r?\n[ \t]*) (?<access>(private|protected|public):
206         )?(?<virtual>virtual )?(?<type>[a-zA-Z_][a-zA-Z0-9_:>]*)
207         (?<property>[a-zA-Z_][a-zA-Z0-9_*] (?<blockOpen>[\n\s]*{[\n\s]*}(\[[^\n]+\] [\n\s]
208         ]*)?get\$*=>\$*(?<expression>[^n]+); (?<blockClose>[\n\s*]{[\n\s*})",
209         "${before}${access}${virtual}${type} ${property}() ${blockOpen}return
210         ${expression};${blockClose}", 2),
211     // ArgumentInRange(string message) { string messageBuilder() { return message; }
212     // ArgumentInRange(string message) { auto messageBuilder = [&]() -> string { return
213         message; }; }
214     (new Regex(@"(?<before>\W[a-zA-Z0-9]+(\[^)\n]*)[\s\n]*{[\s\n]*([^\{\}]|\n)*?(\r?\n)
215         ?[ \t]*)(?<returnType>[_a-zA-Z0-9*:]+[_a-zA-Z0-9*: ]*)
216         (?<methodName>[_a-zA-Z0-9]+)\((?<arguments>[^)\n]*))\$*{(?<body>("")[""\n]+"" | ]
217         [^]| \n)+?)", "${before}auto ${methodName} = [&]() -> ${returnType}
218         ${body};", 10),
219     // Func<TElement> treeCount
220     // std::function<TElement()> treeCount
221     (new Regex(@"Func<[a-zA-Z0-9]+> ([a-zA-Z0-9]+)", "std::function<$1()> $2", 0),
222     // Action<TElement> free
223     // std::function<void(TElement)> free
224     (new Regex(@"Action(<(?<typeParameters>[a-zA-Z0-9]+(
225         [a-zA-Z0-9]+)*>)?(?<after>)| (?<variable>[a-zA-Z0-9]+))",
226         "std::function<void(${typeParameters})>${after}", 0),
227     // Predicate<TArgument> predicate
228     // std::function<bool(TArgument)> predicate
229     (new Regex(@"Predicate<[a-zA-Z0-9]+> ([a-zA-Z0-9]+)", "std::function<bool($1)>
230         $2", 0),
231     // var
232     // auto
233     (new Regex(@"(\W)var(\W)", "$1auto$2", 0),
234     // unchecked
235     //
236     (new Regex(@"[\r\n]{2}\s*?unchecked\s*?$$"), "", 0),
237     // throw new
238     // throw
239     (new Regex(@"(\W)throw new(\W)", "$1throw$2", 0),
240     // void RaiseExceptionIgnoredEvent(Exception exception)
241     // void RaiseExceptionIgnoredEvent(const std::exception& exception)
242     (new Regex(@"(\(| )System\.\Exception|Exception)( |\))", "${1const
243         std::exception&$3", 0),
244     // EventHandler<Exception>
245     // EventHandler<std::exception>
246     (new Regex(@"(\W)(System\.\Exception|Exception)(\W)", "$1std::exception$3", 0),
247     // override void PrintNode(TElement node, StringBuilder sb, int level)
248     // void PrintNode(TElement node, StringBuilder sb, int level) override
249     (new Regex(@"override ([a-zA-Z0-9]*+)\((\([^\)\r\n]+?\))", "$1$2 override", 0),
250     // return (range.Minimum, range.Maximum)
251     // return {range.Minimum, range.Maximum}
252     (new Regex(@"(?<before>return\$*)\((?<values>[^)\n]+)\)(?!\\() (?<after>\W)",
253         "${before}{{$values}}${after}", 0),
254     // string
255     // std::string
256     (new Regex(@"(?<before>\W)(?<!:>string(?<after>\W)", "
257         ${before}std::string${after}", 0),
258     // System.ValueTuple
259     // std::tuple

```

```

235     (new Regex(@"(?<before>\W)(System\.)?ValueTuple(?!\s*=|\() (?<after>\W)") ,
236         +"${before}std::tuple${after}", 0),
237         // sbyte
238         // std::int8_t
239     (new Regex(@"(?<before>\W)((System\.)?SB|sb)yte(?!\s*=|\() (?<after>\W)") ,
240         +"${before}std::int8_t${after}", 0),
241         // short
242         // std::int16_t
243     (new Regex(@"(?<before>\W)((System\.)?Int16|short)(?!\\s*=|\\() (?<after>\W)") ,
244         +"${before}std::int16_t${after}", 0),
245         // int
246         // std::int32_t
247     (new Regex(@"(?<before>\W)((System\.)?I|i)nt(32)?(?!\s*=|\\() (?<after>\W)") ,
248         +"${before}std::int32_t${after}", 0),
249         // long
250         // std::int64_t
251     (new Regex(@"(?<before>\W)((System\.)?Int64|long)(?!\\s*=|\\() (?<after>\W)") ,
252         +"${before}std::int64_t${after}", 0),
253         // byte
254         // std::uint8_t
255     (new Regex(@"(?<before>\W)((System\.)?Byte|byte)(?!\\s*=|\\() (?<after>\W)") ,
256         +"${before}std::uint8_t${after}", 0),
257         // ushort
258         // std::uint16_t
259     (new Regex(@"(?<before>\W)((System\.)?UInt16|ushort)(?!\\s*=|\\() (?<after>\W)") ,
260         +"${before}std::uint16_t${after}", 0),
261         // uint
262         // std::uint32_t
263     (new Regex(@"(?<before>\W)((System\.)?UI|ui)nt(32)?(?!\s*=|\\() (?<after>\W)") ,
264         +"${before}std::uint32_t${after}", 0),
265         // ulong
266         // std::uint64_t
267     (new Regex(@"(?<before>\W)((System\.)?UInt64|ulong)(?!\\s*=|\\() (?<after>\W)") ,
268         +"${before}std::uint64_t${after}", 0),
269         // char*[] args
270         // char* args[]
271     (new Regex(@"( [a-zA-Z0-9:\*]?)\\[\\] ([a-zA-Z0-9]+)", "$1 $2[]", 0),
272         // float.MinValue
273         // std::numeric_limits<float>::lowest()
274     (new Regex(@"(?<before>\W)(?<type>std::[a-zA-Z_]+|float|double)\.MinValue(?<after>\W") ,
275         +"${before}std::numeric_limits<${type}>::lowest()${after}", 0),
276         // double.MaxValue
277         // std::numeric_limits<float>::max()
278     (new Regex(@"(?<before>\W)(?<type>std::[a-zA-Z_]+|float|double)\.MaxValue(?<after>\W") ,
279         +"${before}std::numeric_limits<${type}>::max()${after}", 0),
280         // using Platform.Numbers;
281         //
282     (new Regex(@"([\\r\\n]{2}|~)\\s*?using [\\a-zA-Z0-9]+;\\s*?\"", "", 0),
283         // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
284         // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
285     (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(<[a-zA-Z0-9 ,+]?) : ([a-zA-Z0-9]+)" ,
286         +"$1 $2$3 : public $4", 0),
287         // System.IDisposable
288         // System::IDisposable
289     (new Regex(@"(?<before>System(\\:[a-zA-Z_]\\w*)*)\\.(?<after>[a-zA-Z_]\\w*)") ,
290         +"${before}::${after}", 20),
291         // class IProperty : ISetter< TValue, TObject >, IProvider< TValue, TObject >
292         // class IProperty : public ISetter< TValue, TObject >, public IProvider< TValue,
293             // TObject >
294     (new Regex(@"(?<before>(interface|struct|class) [a-zA-Z_]\\w* : ((public
295             [a-zA-Z_][\\w:]*(<[a-zA-Z0-9 ,+]?)?,
296             )+)?(?<inheritedType>(?!public)[a-zA-Z_][\\w:]*(<[a-zA-Z0-9 ,+]?)?) (?<after>(
297             [a-zA-Z_][\\w:]*(?!>)|[\\r\\n]+))", "${before}public ${inheritedType}${after}" ,
298             10),
299             // interface IDisposable {
300             // class IDisposable { public:
301     (new Regex(@"(?<before>\\r?\\n)(?<indent>[\\t]*)interface
302             (?<interface>[a-zA-Z_]\\w*)(?<typeDefinitionEnding>[^{}]+{}) ,
303             +"${before}${indent}class ${interface}${typeDefinitionEnding}{"
304                 + Environment.NewLine + "    public:", 0),
305                 // struct TreeElement { }
306                 // struct TreeElement { };
307     (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\\s+){([\\sa-zA-Z0-9;:_]+?)\\([^-;])"}, "$1
308             +"$2$3{$4};$5", 0),

```

```

287 // class Program { }
288 // class Program { };
289 (new Regex(@"(?<type>struct|class)
290     (?<name>[a-zA-Z0-9]+[^r\n]*)(?<beforeBody>[\r\n]+(?<indentLevel>[\t
291     ]*)?)\{(?<body>[\S\s]+[\r\n]+\k<indentLevel>)\}(?<afterBody>[^;]|$)", "${type}
292     ${name}${beforeBody}${body};${afterBody}", 0),
293     // Insert scope borders.
294     // ref TElement root
295     // ~!root!~ref TElement root
296     (new Regex(@"(?<definition>(?<= | \() (ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!ref>
297         (?<variable>[a-zA-Z0-9]+)(?= \) |, | =))", "~!${variable}!~${definition}", 0),
298     // Inside the scope of ~!root!~ replace:
299     // root
300     // *root
301     (new Regex(@"(?<definition>~! (?<pointer>[a-zA-Z0-9]+)!~ref [a-zA-Z0-9]+
302         \k<pointer>(?= \) |, | =)(?<before>((?<= !\k<pointer>!)~(.|\n)*?) (?<prefix>(\W
303         |\() \k<pointer>(?<suffix>(| \) |; |,)), "
304         "${definition}${before}${prefix}*${pointer}${suffix}", 70),
305     // Remove scope borders.
306     // ~!root!~
307     //
308     (new Regex(@"~! (?<pointer>[a-zA-Z0-9]+)!~", "", 5),
309     // ref auto root = ref
310     // ref auto root =
311     (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)", "$1* $2 = $3", 0),
312     // *root = ref left;
313     // root = left;
314     (new Regex(@"*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)", "$1 = $2$3", 0),
315     // (ref left)
316     // (left)
317     (new Regex(@"(ref ([a-zA-Z0-9]+)(\)|\(|,))", "($1$2", 0),
318     // ref TElement
319     // TElement*
320     (new Regex(@"( | \() ref ([a-zA-Z0-9]+) ", "$1$2*", 0),
321     // ref sizeBalancedTree.Root
322     // &sizeBalancedTree->Root
323     (new Regex(@"ref ([a-zA-Z0-9]+)\.( [a-zA-Z0-9\*]+)", "&$1->$2", 0),
324     // ref GetElement(node).Right
325     // &GetElement(node)->Right
326     (new Regex(@"ref ([a-zA-Z0-9]+)\((( [a-zA-Z0-9\*]+)\)\.( [a-zA-Z0-9]+)", "
327         "&$1($2)->$3", 0),
328     // GetElement(node).Right
329     // GetElement(node)->Right
330     (new Regex(@"([a-zA-Z0-9]+)\((( [a-zA-Z0-9\*]+)\)\.( [a-zA-Z0-9]+)", "$1($2)->$3", 0),
331     // [Fact]\npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
332     // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
333     (new Regex(@"\[Fact\] [\s\n]+(public: )?(static )?void ([a-zA-Z0-9]+)\(\)", "public:
334         TEST_METHOD($3)", 0),
335     // class TreesTests
336     // TEST_CLASS(TreesTests)
337     (new Regex(@"class ([a-zA-Z0-9]+Tests)", "TEST_CLASS($1)", 0),
338     // Assert.Equal
339     // Assert::AreEqual
340     (new Regex(@"(?<type>Assert)\.(?<method>(Not)?Equal)", "${type}::Are${method}", 0),
341     // Assert.Throws
342     // Assert::ExpectException
343     (new Regex(@"(Assert)\.\.Throws", "$1::ExpectException", 0),
344     // Assert.True
345     // Assert::IsTrue
346     (new Regex(@"(Assert)\.(True|False)", "$1::Is$2", 0),
347     // $"Argument {argumentName} is null."
348     // std::string("Argument
349         ").append(Platform::Converters::To<std::string>(argumentName)).append(" is
350         null.")
351     (new Regex(@"\$\"(?<left>(\\"|[^"\r\n])*{(?<expression>[_a-zA-Z0-9]+)}(?<right>(\_
352         \\"|[^"\r\n])*\""),
353         "std::string($"${left}\").append(Platform::Converters::To<std::string>(${expres
354         sion})).append("${right}\")",
355         10),
356     // $
357     // "
358     (new Regex(@"\$\"", "\\"", 0),
359     // std::string(std::string("[").append(Platform::Converters::To<std::string>(Minimum_
360         )).append(", "
361         ")).append(Platform::Converters::To<std::string>(Maximum)).append("]")
362     // std::string("[").append(Platform::Converters::To<std::string>(Minimum)).append(",
363         ").append(Platform::Converters::To<std::string>(Maximum)).append("]")

```

```

346     (new Regex(@"std::string\((?<begin>std::string\(""(\\\"|[^"])*""\)(\.append\((Platform<
347         +>orm::Converters::To<std::string>\([^\n]+|\[^\n]+\)))+)\)\.\append"),
348         +>"\${begin}.append", 10),
349     // Console.WriteLine("...")
350     // printf(...\n")
351     (new Regex(@"Console\.\WriteLine\(""(^"\r\n]+)""\)", "printf(\"$1\\n\"", 0),
352     // TElement Root;
353     // TElement Root = 0;
354     (new Regex(@"(?<before>\r?\n[\t ]+)(?<access>(private|protected|public)(:
355         +>)?)?(?<type>[a-zA-Z0-9:_]+(?<!return>) (?<name>[_a-zA-Z0-9]+);"),
356         +>"\${before}\${access}\${type} \${name} = 0;", 0),
357     // TreeElement _elements[N];
358     // TreeElement _elements[N] = { {0} };
359     (new Regex(@"(\r?\n[\t ]+)(private|protected|public)?(: )?([a-zA-Z0-9]+
360         +> ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9]+)\];)", "$1$2$3$4 $5[$6] = { {0} };", 0),
361     // auto path = new TElement[MaxPath];
362     // TElement path[MaxPath] = { {0} };
363     (new Regex(@"(\r?\n[\t ]+)[a-zA-Z0-9]+ ([a-zA-Z0-9]+) = new
364         +> ([a-zA-Z0-9]+)\[([_a-zA-Z0-9]+)\];)", "$1$3 $2[$4] = { {0} };", 0),
365     // bool Equals(Range<T> other) { ... }
366     // bool operator ==(const Key &other) const { ... }
367     (new Regex(@"(?<before>\r?\n[^n]+bool )Equals\((?<type>[^n]+
368         +> (?<variable>[a-zA-Z0-9]+)\)(?<after>(\s|\n)*{})", "\${before}operator ==(const
369         +> ${type} &${variable}) const${after}", 0),
370     // Insert scope borders.
371     // class Range { ... public: override std::string ToString() { return ...; }
372     // class Range /*~Range<T>~*/ { ... public: override std::string ToString() { return
373         +> ...; }
374     (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)template <typename
375         +> (?<typeParameter>[^>\n]+)> (struct|class)
376         +> (?<type>[a-zA-Z0-9]+<>\k<typeParameter>>)(\s*: \s*[^{\n}]+)?[\t ]*(\r?\n)?[\t
377         +> ]*{})(?<middle>((?!class|struct).|\n)+?) (?<toStringDeclaration>(?<access>(private|
378         +> |protected|public): )override std::string ToString()\(\)), "
379         +> "\${classDeclarationBegin}/*~${type}~*/${middle}${toStringDeclaration}", 0),
380     // Inside the scope of ~!Range!~ replace:
381     // public: override std::string ToString() { return ...; }
382     // public: operator std::string() const { return ...; }\n\npublic: friend
383         +> std::ostream & operator <<(std::ostream &out, const A &obj) { return out <<
384             +> (std::string)obj; }
385     (new Regex(@"(?<scope>/\*~(?<type>[_a-zA-Z0-9<>:]+)\*/)(?<separator>.|\n)(?<before>_
386         +> ((?<!/\*~\k<type>~\*/(.|\n))*?) (?<toStringDeclaration>\r?\n(?<indent>[
387             +> \t]*)(?<access>(private|protected|public): )override std::string ToString()\()
388             +> (?<toStringMethodBody>{[^}\n]+}))", "\${scope}\${separator}\${before}" +
389             +> Environment.NewLine + "\${indent}\${access}operator std::string() const
390             +> \${toStringMethodBody}" + Environment.NewLine + Environment.NewLine +
391             +> "\${indent}\${access}friend std::ostream & operator <<(std::ostream &out, const
392             +> ${type} &obj) { return out << (std::string)obj; }", 0),
393     // Remove scope borders.
394     // /*~Range~*/
395     //
396     (new Regex(@"/\*~[_a-zA-Z0-9<>:]+\*/"), "", 0),
397     // private: inline static ConcurrentBag<std::exception> _exceptionsBag;
398     // private: inline static std::mutex _exceptionsBag_mutex; \n\n private: inline
399     +> static std::vector<std::exception> _exceptionsBag;
400     (new Regex(@"(?<begin>\r?\n?(?<indent>[\t ]+))(?<access>(private|protected|public):
401         +> ?inline static ConcurrentBag<(?<argumentType>[^;\r\n]+)>
402         +> (?<name>[_a-zA-Z0-9]+);"), "\${begin}private: inline static std::mutex
403             +> ${name}_mutex;" + Environment.NewLine + Environment.NewLine +
404             +> "\${indent}\${access}inline static std::vector<${argumentType}> ${name};", 0),
405     // public: static IReadOnlyCollection<std::exception> GetCollectedExceptions() {
406         +> return _exceptionsBag; }
407     // public: static std::vector<std::exception> GetCollectedExceptions() { return
408         +> std::vector<std::exception>(_exceptionsBag); }
409     (new Regex(@"(?<access>(private|protected|public): )?static
410         +> IReadOnlyCollection<(?<argumentType>[^;\r\n]+)> (?<methodName>[_a-zA-Z0-9]+)\(\)
411         +> { return (?<fieldName>[_a-zA-Z0-9]+); }", "\${access}static
412             +> std::vector<${argumentType}> ${methodName}() { return
413                 +> std::vector<${argumentType}>(${fieldName}); }", 0),
414     // public: static event EventHandler<std::exception> ExceptionIgnored =
415         +> OnExceptionIgnored; ... );
416     // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
417         +> const std::exception&)> ExceptionIgnored = OnExceptionIgnored; );

```

```

382 (new Regex(@"(?<begin>\r?\n(\r?\n)?(?<halfIndent>[
    \t]+)\k<halfIndent>) (?<access>(private|protected|public): )?static event
→ EventHandler<(?<argumentType>[^; \r\n]+)> (?<name>[_a-zA-Z0-9]+) = (?<defaultDelete>[_a-zA-Z0-9]+); (?<middle>(.|\n)+?) (?<end>\r?\n\k<halfIndent>}); )",
→ "{$middle}" + Environment.NewLine + Environment.NewLine +
→ "{$halfIndent}{$halfIndent}{$access}static inline
→ Platform::Delegates::MulticastDelegate<void(void*, const ${argumentType}&)>
→ ${name} = ${defaultDelegate}; ${end}", 0),
// public: event Disposal OnDispose;
// public: Platform::Delegates::MulticastDelegate<Disposal> OnDispose;
(new Regex(@"(?<begin>(?<access>(private|protected|public): )?(static )?)event
→ (?<type>[a-zA-Z] [:_a-zA-Z0-9]+) (?<name>[a-zA-Z] [_a-zA-Z0-9]+); ),
→ "{$begin}Platform::Delegates::MulticastDelegate<${type}> ${name}; ", 0),
// Insert scope borders.
// class IgnoredExceptions { ... private: inline static std::vector<std::exception>
→ _exceptionsBag;
// class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: inline static
→ std::vector<std::exception> _exceptionsBag;
(new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n}]+\r\n[\t
→ ]*{} (?<middle>((?!class).|\n)+?) (?<vectorFieldDeclaration>(?<access>(private|pro
→ tected|public): )inline static std::vector<(?<argumentType>[^; \r\n]+)>
→ (?<fieldName>[_a-zA-Z0-9]+); ),
→ "{$classDeclarationBegin}/*~${fieldName}~*/${middle}${vectorFieldDeclaration}", 0),
// Inside the scope of ~!_exceptionsBag!~ replace:
// _exceptionsBag.Add(exception);
// _exceptionsBag.push_back(exception);
(new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)\*~/ (?<separator>.|\n)(?<before>
→ e>((?<!/*\~\k<fieldName>~\*/)(.|\n)*?)\k<fieldName>.\Add"),
→ "{$scope}${separator}${before}${fieldName}.push_back", 10),
// Remove scope borders.
// /*~_exceptionsBag~*/
//
(new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
// Insert scope borders.
// class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
// class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: static std::mutex
→ _exceptionsBag_mutex;
(new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n}]+\r\n[\t
→ ]*{} (?<middle>((?!class).|\n)+?) (?<mutexDeclaration>private: inline static
→ std::mutex (?<fieldName>[_a-zA-Z0-9]+)_mutex; )",
→ "{$classDeclarationBegin}/*~${fieldName}~*/${middle}${mutexDeclaration}", 0),
// Inside the scope of ~!_exceptionsBag!~ replace:
// return std::vector<std::exception>(_exceptionsBag);
// std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
→ std::vector<std::exception>(_exceptionsBag);
(new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)\*~/ (?<separator>.|\n)(?<before>
→ e>((?<!/*\~\k<fieldName>~\*/)(.|\n)*?) (?<after>((?!lock_guard)[^{\}; \r\n])*?\k<fieldName>[^; \r\n]*; )"),
→ "{$scope}${separator}${before}{${fieldName}_mutex}; ${after}", 10),
// Inside the scope of ~!_exceptionsBag!~ replace:
// _exceptionsBag.Add(exception);
// std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
→ _exceptionsBag.Add(exception);
(new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)\*~/ (?<separator>.|\n)(?<before>
→ e>((?<!/*\~\k<fieldName>~\*/)(.|\n)*?) (?<after>((?!lock_guard)([^{\}; \r\n])*?\r
→ ?\n(?<indent>[\t ]*)\k<fieldName>[^; \r\n]*; )"),
→ "{$scope}${separator}${before}{${fieldName}_mutex}; ${after}", 10),
// Remove scope borders.
// /*~_exceptionsBag~*/
//
(new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
// Insert scope borders.
// class IgnoredExceptions { ... public: static inline
→ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
→ ExceptionIgnored = OnExceptionIgnored;
// class IgnoredExceptions {/*~ExceptionIgnored~*/ ... public: static inline
→ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
→ ExceptionIgnored = OnExceptionIgnored;
(new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n}]+\r\n[\t
→ ]*{} (?<middle>((?!class).|\n)+?) (?<eventDeclaration>(?<access>(private|protected
→ |public): )static inline
→ Platform::Delegates::MulticastDelegate<(?<argumentType>[^; \r\n]+)>
→ (?<name>[_a-zA-Z0-9]+) = (?<defaultDelegate>[_a-zA-Z0-9]+; )",
→ "{$classDeclarationBegin}/*~${name}~*/${middle}${eventDeclaration}", 0),

```

```

418 // Inside the scope of ~!ExceptionIgnored!~ replace:
419 // ExceptionIgnored.Invoke(NULL, exception);
420 // ExceptionIgnored(NULL, exception);
421 (new Regex(@"(?<scope>/\*~(?<eventName>[a-zA-Z0-9]+)\*/)(?<separator>.|\n)(?<before>
422 → >((?<!/*\k<eventName>*\()(.|\n)*?\)\k<eventName>\.Invoke"),
423 → "${scope}${separator}${before}${eventName}", 10),
424 // Remove scope borders.
425 // /*~ExceptionIgnored~*/
426 //
427 (new Regex(@"/\*~[a-zA-Z0-9]+~\*/"), "", 0),
428 // Insert scope borders.
429 // auto added = new StringBuilder();
430 // /*~sb~*/std::string added;
431 (new Regex(@"(auto|System\.Text\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new
432 → (System\.Text\.)?StringBuilder\(\);", "/*~${variable}~*/std::string
433 → ${variable};", 0),
434 // static void Indent(StringBuilder sb, int level)
435 // static void Indent(/*~sb~*/StringBuilder sb, int level)
436 (new Regex(@"(?<start>, |\() (System\.Text\.)?StringBuilder
437 → (?<variable>[a-zA-Z0-9]+)(?<end>, |\())", "${start}/*~${variable}~*/std::string&
438 → ${variable}${end}", 0),
439 // Inside the scope of ~!added!~ replace:
440 // sb.ToString()
441 // sb
442 (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)\*/)(?<separator>.|\n)(?<before>
443 → ((?<!/*\k<variable>*\()(.|\n)*?\)\k<variable>\.ToString\(\)", ,
444 → "${scope}${separator}${before}${variable}", 10),
445 // sb.AppendLine(argument)
446 // sb.append(Platform::Converters::To<std::string>(argument)).append(1, '\n')
447 (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)\*/)(?<separator>.|\n)(?<before>
448 → ((?<!/*\k<variable>*\()(.|\n)*?\)\k<variable>\.AppendLine\((?<argument>[^),\r\n]+
449 → r\n]+\)\)", ,
450 → "${scope}${separator}${before}${variable}.append(Platform::Converters::To<std::s
451 → tring>(${argument})).append(1, '\n'", ,
452 → 10),
453 // sb.Append('\t', level);
454 // sb.append(level, '\t');
455 (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)\*/)(?<separator>.|\n)(?<before>
456 → ((?<!/*\k<variable>*\()(.|\n)*?\)\k<variable>\.Append\('(?<character>[^'\r\n]+
457 → +)', (?<count>[^),\r\n]+\)\)", ,
458 → "${scope}${separator}${before}${variable}.append(${count}, '${character}')", 10),
459 // sb.Append(argument)
460 // sb.append(Platform::Converters::To<std::string>(argument))
461 (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)\*/)(?<separator>.|\n)(?<before>
462 → ((?<!/*\k<variable>*\()(.|\n)*?\)\k<variable>\.Append\((?<argument>[^),\r\n]+
463 → +)\)\)", ,
464 → "${scope}${separator}${before}${variable}.append(Platform::Converters::To<std::s
465 → tring>(${argument})), ,
466 → 10),
467 // Remove scope borders.
468 // /*~sb~*/
469 //
470 (new Regex(@"/\*~[a-zA-Z0-9]+~\*/"), "", 0),
471 // Insert scope borders.
472 // auto added = new HashSet<TElement>();
473 // ~!added!~std::unordered_set<TElement> added;
474 (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
475 → HashSet<(?<element>[a-zA-Z0-9]+)>\(\);",
476 → "~~!${variable}!~std::unordered_set<${element}> ${variable};", 0),
477 // Inside the scope of ~!added!~ replace:
478 // added.Add(node)
479 // added.insert(node)
480 (new Regex(@"(?<scope>~! (?<variable>[a-zA-Z0-9]+) !~)(?<separator>.|\n)(?<before>((?<
481 → !~!\k<variable>!*~(.|\n)*?\)\k<variable>\.Add\((?<argument>[a-zA-Z0-9]+)\)\)",
482 → "${scope}${separator}${before}${variable}.insert(${argument})", 10),
483 // Inside the scope of ~!added!~ replace:
484 // added.Remove(node)
485 // added.erase(node)
486 (new Regex(@"(?<scope>~! (?<variable>[a-zA-Z0-9]+) !~)(?<separator>.|\n)(?<before>((?<
487 → !~!\k<variable>!*~(.|\n)*?\)\k<variable>\.Remove\((?<argument>[a-zA-Z0-9]+)\)\)",
488 → "${scope}${separator}${before}${variable}.erase(${argument})", 10),
489 // if (added.insert(node)) {
490 // if (!added.contains(node)) { added.insert(node);

```

```

464 (new Regex(@"if \((?<variable>[a-zA-Z0-9]+)\.insert\((?<argument>[a-zA-Z0-9]+)\)\) (?_
465     <separator>[\t ]*[\\r\\n]+)(?<indent>[\t ]*){", "if
466     (!${variable}.contains(${argument}))${separator}${indent}{"
467     + Environment.NewLine + "${indent}    ${variable}.insert(${argument});", 0),
468     // Remove scope borders.
469     // ~!added!
470     //
471     (new Regex(@"^![a-zA-Z0-9]+!~"), "", 5),
472     // Insert scope borders.
473     // auto random = new System::Random(0);
474     // std::srand(0);
475     (new Regex(@"[a-zA-Z0-9\.]+ ([a-zA-Z0-9]+) = new
476     (System::)Random\(([a-zA-Z0-9]+)\);", "~!$1!~std::srand($3);", 0),
477     // Inside the scope of ~!random!~ replace:
478     // random.Next(1, N)
479     // (std::rand() % N) + 1
480     (new Regex(@"(?<scope>^!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.\\n)(?<before>((?<
481         !^!\\k<variable>!~)(.|\\n)*?)\\k<variable>.Next\((?<from>[a-zA-Z0-9]+),
482         (?<to>[a-zA-Z0-9]+)\\)", "${scope}${separator}${before}(std::rand() % ${to}) +
483         ${from}", 10),
484     // Remove scope borders.
485     // ~!random!
486     //
487     (new Regex(@"^![a-zA-Z0-9]+!~"), "", 5),
488     // Insert method body scope starts.
489     // void PrintNodes(TElement node, StringBuilder sb, int level) {
490     // void PrintNodes(TElement node, StringBuilder sb, int level) {/*method-start*/
491     (new Regex(@"(?<start>\\r?\\n[\\t ]+)(?<prefix>((private|protected|public): )?(virtual
492         )?[a-zA-Z0-9:_]+
493         )?(?<method>[a-zA-Z][a-zA-Z0-9]*\\((?<arguments>[^\\)]*)\\)(?<override>(
494             override)?)(?<separator>[ \\t\\r\\n]*){(?<end>[^\\])}", "${start}${prefix}${method}_
495             ${arguments}${override}${separator}/*method-start*/${end}",
496             0),
497     // Insert method body scope ends.
498     // /*method-start*/...
499     // /*method-start*/.../*method-end*/
500     (new Regex(@"{/*method-start*/(?<body>((?<bracket>\\{}|(?<-bracket>\\})|[^\{\}]*)+)
501         \\}"), "{/*method-start*/${body}/*method-end*/",
502         0),
503     // Inside method bodies replace:
504     // GetFirst(
505     // this->GetFirst(
506     (new
507         Regex(@"(?<scope>/\\*method-start\\*)(?<before>((?<!--\\*method-end\\*/(.|\\n)*?) (?_
508             <separator>[\\W](?<!--\\:\\.|->|throw\\s+)))(?<method>(?!sizeof)[a-zA-Z0-9]+)\\((?<!--\\
509             \\{})(?<after>(.|\\n)*?) (?<scopeEnd>/\\*method-end\\*/)",
510             "${scope}${before}${separator}this->${method}(${after}${scopeEnd}", 100),
511     // Remove scope borders.
512     // /*method-start*/
513     //
514     (new Regex(@"\\/*method-(start|end)\\*/"), "", 0),
515     // Insert scope borders.
516     // const std::exception& ex
517     // const std::exception& ex/*~ex~*/
518     (new Regex(@"(?<before>\\(| )(?<variableDefinition>(const )?(std::)?exception&?
519         (?<variable>[_a-zA-Z0-9]+))(?<after>\\W)",
520         "${before}${variableDefinition}/*~${variable}~*${after}", 0),
521     // Inside the scope of ~!ex!~ replace:
522     // ex.Message
523     // ex.what()
524     (new Regex(@"(?<scope>/\\*~(?<variable>[_a-zA-Z0-9]+)~\\*)(?<separator>.\\n)(?<before>
525         >((?<!--\\*~\\k<variable>~\\*)(.|\\n)*?)(Platform::Converters::To<std::string>\\((\\k<_
526         variable>.Message)|\\k<variable>.Message)",
527         "${scope}${separator}${before}${variable}.what()", 10),
528     // Remove scope borders.
529     // /*~ex~*/
530     //
531     (new Regex(@"\\/*~[_a-zA-Z0-9]+~\\*/"), "", 0),
532     // throw ObjectDisposedException(objectName, message);
533     // throw std::runtime_error(std::string("Attempt to access disposed object
534     // [").append(objectName).append(": ").append(message).append("."));
535     (new Regex(@"throw ObjectDisposedException\\((?<objectName>[a-zA-Z_][a-zA-Z0-9_]*) ,
536         (?<message>[a-zA-Z0-9_]*[Mm]essage[a-zA-Z0-9_]*\\((\\))?[\\a-zA-Z_][a-zA-Z0-9_]*)\\)
537         ;", "throw std::runtime_error(std::string(\"Attempt to access disposed object
538         [\\]).append(${objectName}).append(": ").append(${message}).append("\\.\"));",
539         0),

```

```

512 // throw ArgumentNullException(argumentName, message);
513 // throw std::invalid_argument(std::string("Argument
514     " ).append(argumentName).append(" is null: ").append(message).append("."));
515     (new Regex(@"throw
516         ArgumentException\(((?<argument>[a-zA-Z]*[Aa]rgument[a-zA-Z]*),
517             (?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\))?\)\);"), "throw
518             std::invalid_argument(std::string(\"Argument \").append(${argument}).append(\""
519                 is null: \").append(${message}).append(\".\\"));", 0),
520             // throw ArgumentException(message, argumentName);
521             // throw std::invalid_argument(std::string("Invalid ").append(argumentName).append(
522                 " argument: ").append(message).append("."));
523             (new Regex(@"throw ArgumentException\(((?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\))?), "
524                 (?<argument>[a-zA-Z]*[Aa]rgument[a-zA-Z]*)\);"), "throw
525                     std::invalid_argument(std::string(\"Invalid \").append(${argument}).append(\""
526                         argument: \").append(${message}).append(\".\\"));", 0),
527                     // throw ArgumentOutOfRangeException(argumentName, argumentValue, messageBuilder());
528                     // throw std::invalid_argument(std::string("Value
529                         [").append(Platform::Converters::To<std::string>(argumentValue)).append("] of
530                         argument [").append(argumentName).append("] is out of range:
531                         ").append(messageBuilder()).append("."));
532             (new Regex(@"throw ArgumentOutOfRangeException\(((?<argument>[a-zA-Z]*[Aa]rgument[a-zA-Z]*[Nn]ame[a-zA-Z]*),
533                 (?<argumentValue>[a-zA-Z]*[Aa]rgument[a-zA-Z]*[Vv]alue[a-zA-Z]*?), "
534                     (?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\))?\)\);"), "throw
535                     std::invalid_argument(std::string(\"Value
536                         [").append(Platform::Converters::To<std::string>(${argumentValue})).append("] of argument [").append(${argument}).append("] is out of range:
537                         ").append(${message}).append(\".\\"));", 0),
538                     // throw NotSupportedException();
539                     // throw std::logic_error("Not supported exception.");
540                     (new Regex(@"throw NotSupportedException\(\);"), "throw std::logic_error(\"Not
541                         supported exception.\");", 0),
542                     // throw NotImplementedException();
543                     // throw std::logic_error("Not implemented exception.");
544                     (new Regex(@"throw NotImplementedException\(\);"), "throw std::logic_error(\"Not
545                         implemented exception.\");", 0),
546                     // Insert scope borders.
547                     // const std::string& message
548                     // const std::string& message/*~message~*/
549                     (new Regex(@"(?<before>\(|)(?<variableDefinition>(const )?((std::)?string&|char\*)"
550                         (?<variable>[_a-zA-Z0-9]+))(?<after>\W"),
551                         "${before}${variableDefinition}/*~${variable}~*${after}", 0),
552                     // Inside the scope of /*~message~*/ replace:
553                     // Platform::Converters::To<std::string>(message)
554                     // message
555                     (new Regex(@"(?<scope>/\*~(?<variable>[_a-zA-Z0-9]+)~*/)(?<separator>.|\n)(?<before>
556                         >((?<!/(*~\k<variable>~*/)(.|\\n))*?)Platform::Converters::To<std::string>\\(\k<v
557                             ariable>\\)", "${scope}${separator}${before}${variable}", 10),
558                     // Remove scope borders.
559                     // /*~ex~*/
560                     //
561                     (new Regex(@"/\*~[_a-zA-Z0-9]+~*/"), "", 0),
562                     // Insert scope borders.
563                     // std::tuple<T, T> tuple
564                     // std::tuple<T, T> tuple/*~tuple~*/
565                     (new Regex(@"(?<before>\(|)(?<variableDefinition>(const )?(std::)?tuple<[^\\n]+>&?
566                         (?<variable>[_a-zA-Z0-9]+))(?<after>\W"),
567                         "${before}${variableDefinition}/*~${variable}~*${after}", 0),
568                     // Inside the scope of ~!ex!~ replace:
569                     // tuple.Item1
570                     // std::get<1-1>(tuple)
571                     (new Regex(@"(?<scope>/\*~(?<variable>[_a-zA-Z0-9]+)~*/)(?<separator>.|\n)(?<before>
572                         >((?<!/(*~\k<variable>~*/)(.|\\n))*?)\\k<variable>.Item(?<itemNumber>\\d+)(?<afte
573                             r>\W"),
574                         "${scope}${separator}${before}std::get<${itemNumber}-1>(${variable})${after}", 10),
575                     // Remove scope borders.
576                     // /*~ex~*/
577                     //
578                     (new Regex(@"/\*~[_a-zA-Z0-9]+~*/"), "", 0),
579                     // Insert scope borders.
580                     // class Range<T> {
581                     // class Range<T> {/*~Range<T>~*/}

```

```

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

```

```

582 (new Regex(@"(?<namespaceScopeStart>/\*~start~namespace~(?<namespace>[^~\n\*]+)~\*/)
      (?<betweenStartScopes>(.|\n)+)(?<typeScopeStart>/\*~start~type~(?<type>[^~\n\*]+)
      )~(?<typeParameter>[^~\n\*]+)~\*/)(?<before>(.|\n)+?) (?<hashMethodDeclaration>\r
      \r\n[ \t]*(?<access>(private|protected|public): )override std::int32_t
      GetHashCode\(\)(\$s|\n)*{\$s*(?<methodBody>[^s][^n]+[^s])\$s*)?<after>(.|\n
      )+?) (?<typeScopeEnd>/\*~end~type~\k<type>~\k<typeParameter>~\*/)(?<betweenEndSco
      pes>(.|\n)+)(?<namespaceScopeEnd>/\*~end~namespace~\k<namespace>~\*/)\r?\n"),
      "${namespaceScopeStart}${betweenStartScopes}${typeScopeStart}${before}${after}${
      typeScopeEnd}${betweenEndScopes}${namespaceScopeEnd}" + Environment.NewLine +
      Environment.NewLine + "namespace std" + Environment.NewLine + "{" +
      Environment.NewLine + "    template <typename ${typeParameter}>" +
      Environment.NewLine + "        struct hash<${namespace}: ${type}>" +
      Environment.NewLine + "            {" + Environment.NewLine + "                std::size_t
      operator()(const ${namespace}: ${type} &obj) const" + Environment.NewLine + "
      /*~start~method~*//${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>[\n.\.(a-zA-Z0-9)((?!/*~end~method~\*/
      )[^n]+)(?<methodScopeEnd>/\*~end~method~\*/)",
      "${methodScopeStart}${before}obj.${name}${after}${methodScopeEnd}", 10),
      // Remove scope borders.
      // /*~start~type~Range<T>~*/
      //
      (new Regex(@"/*~[^~\n]+(~[^~\n]+)*~\*/"), "", 0),
      // class Disposable<T> : public Disposable
      // class Disposable<T> : public Disposable<>
      (new Regex(@"(?<before>(struct|class) (?<type>[a-zA-Z][a-zA-Z0-9]*)<[^>\n+> :
      (?<access>(private|protected|public) )?\k<type>)(?<after>\b(?!<))",
      "${before}>&${after}", 0),
      // Insert scope borders.
      // class Disposable<T> : public Disposable<> { ... };
      // class Disposable<T> : public Disposable<>
      // /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/
      // /*~end~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/
      (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)template[\t
      ]*<(?<typeParameters>[^n]*)>[\t ]*(struct|class)[\t
      ]+(<fullType>(?<type>[a-zA-Z][a-zA-Z0-9]*)(<[^>\n*>)?)[\t ]*:[:[\t
      ]*(?<access>(private|protected|public)[\t
      ]+)?(<fullBaseType>(?<baseType>[a-zA-Z][a-zA-Z0-9]*)(<[^>\n*>)?)[\t
      ]*(\r?\n)?[\t
      ]*{)(?<middle>(.|\n*)(?<beforeEnd>(?<=r?\n)\k<indent>)(?<end>);)",
      "${classDeclarationBegin}/*~start~type~${type}~${fullType}~${baseType}~${fullBas
      eType}~*/${middle}&${beforeEnd}/*~end~type~${type}~${fullType}~${baseType}~${full
      BaseType}~*/${end}",
      0),
      // Inside the scope replace:
      // /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/ ... ) : base(
      ... /*~end~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/
      // /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/ ... ) :
      // Disposable<>(<fullType>~Disposable~Disposable<T>~Disposable~Disposable<>~*/
      (new Regex(@"(?<before>(?<typeScopeStart>/\*~start~type~(?<types>(?<type>[^~\n\*]+)~
      (?<fullType>[^~\n\*]+)~\k<type>~(?<fullBaseType>[^~\n\*]+)~\*/)(.|\n)+?)\$s:\$s
      )base(?<after>\((.|\n)+?(?<typeScopeEnd>/\*~end~type~\k<types>~\*/))",
      "${before}&${fullBaseType}${after}", 20),
      // Inside the scope 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 the scope replace:
      // /*~start~type~Disposable~Disposable<T>~X~X<>~*/ ... public: Disposable(T object)
      { Object = object; } ... public: Disposable(T object) : Disposable(object) { }
      ... /*~end~type~Disposable~Disposable<T>~X~X<>~*/
      // /*~start~type~Disposable~Disposable<T>~X~X<>~*/ ... public: Disposable(T object)
      { Object = object; } /*~end~type~Disposable~Disposable<T>~X~X<>~*/

```

```

609   (new Regex(@"(?<before>(?<typeScopeStart>/\*~start~type~(?<types>(?<type>[^~\n\*]+)+~
610     (?<fullType>[^~\n\*]+)+~(?<baseType>[^~\n\*]+)+~(?<fullBaseType>[^~\n\*]+)+~\*/)(.|_
611     |\n)+?~(?<constructor>(?<access>(private|protected|public):[\t
612     ]*)?~\k<type>\((?<arguments>[^~\n\*]+)\)\s*\{[^~\n\*]+\}(.|\n)+?)~(?<duplicateConstru_
613     ctor>(?<access>(private|protected|public):[\t
614     ]*)?~\k<type>\((\k<arguments>)\)\s*:[^~\n\*]+\s*\{[^~\n\*]+\})~(?<after>(.|\n)+?~(?<typeS_
615     copeEnd>/\*~end~type~\k<types>~\*/))", "${before}${after}",
616     20),
617   // Remove scope borders.
618   // /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/
619   //
620   (new Regex(@"/\*~[^~\n\*]+\n+~[^~\n\*]\n+~\*/", "", 0),
621   // Insert scope borders.
622   // private: inline static const AppDomain _currentDomain = AppDomain.CurrentDomain;
623   // private: inline static const AppDomain _currentDomain =
624   //   AppDomain.CurrentDomain; /*~app-domain~_currentDomain~*/
625   (new Regex(@"(?<declaration>(?<access>(private|protected|public):[\t ]*)?(inline[\t
626     ]+)?~(static[\t ]+)?~(const[\t ]+)?~AppDomain[\t
627     ]+~(?<field>[a-zA-Z_][a-zA-Z0-9_]*)[\t ]*=~[\t ]*AppDomain\.CurrentDomain;)",
628     "${declaration}/*~app-domain~${field}~*/", 0),
629   // Inside the scope replace:
630   // /*~app-domain~_currentDomain~*/ ... _currentDomain.ProcessExit += OnProcessExit;
631   // /*~app-domain~_currentDomain~*/ ... std::atexit(OnProcessExit);
632   (new Regex(@"(?<before>(?<fieldScopeStart>/\*~app-domain~(?<field>[^~\n\*]+)+~\*/)(.|_
633     \n)+?)~\k<field>\.ProcessExit[\t ]*+=~[\t
634     ]*~(?<eventHandler>[a-zA-Z_][a-zA-Z0-9_]*);", "${before}std::atexit(${eventHandler}~",
635     20),
636   // Inside the scope replace:
637   // /*~process-exit-handler~OnProcessExit~*/ ... static void OnProcessExit(void
638     *sender, EventArgs e)
639   // /*~process-exit-handler~OnProcessExit~*/ ... static void OnProcessExit()
640   (new Regex(@"(?<before>(?<fieldScopeStart>/\*~process-exit-handler~(?<handler>[^~\n\*
641     ]*)~\*/)(.|n)+?~static[\t ]+void[\t ]+\k<handler>\()([^\n]+)\)", "${before}"),
642     20),
643   // Remove scope borders.
644   // /*~app-domain~_currentDomain~*/
645   //
646   (new Regex(@"/\*~[^~\n\*]+\n+~[^~\n\*]\n+~\*/", "", 0),
647   // AppDomain.CurrentDomain.ProcessExit -= OnProcessExit;
648   // /* No translation. It is not possible to unsubscribe from std::atexit. */
649   (new Regex(@"AppDomain\.CurrentDomain\.ProcessExit -= ([a-zA-Z_][a-zA-Z0-9_]*);",
650     "/* No translation. It is not possible to unsubscribe from std::atexit. */", 0),
651   }.Cast<ISubstitutionRule>().ToList();
652
653   /// <summary>
654   /// <para>
655   /// The to list.
656   /// </para>
657   /// <para></para>
658   /// </summary>
659   public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
660   {
661     // IDisposable disposable)
662     // IDisposable &disposable)
663     (new Regex(@"(?<argumentAbstractType>I[A-Z][a-zA-Z0-9]+(<[^>\r\n]+>)?"
664       ~<?<argument>[_a-zA-Z0-9]+(<?<after>, |\n))", "${argumentAbstractType}~
665       &${argument}${after}", 0),
666     // ICounter<int, int> c1;
667     // ICounter<int, int>* c1;
668     (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^>\r\n]+>)?"
669       ~<?<variable>[_a-zA-Z0-9]+(<?<after> = null)?;)", "${abstractType}~
670       *${variable}${after};", 0),
671     // (expression)
672     // expression
673     (new Regex(@"(\(| ))\(([a-zA-Z0-9_:*:]+)\)(,| | ; |\n)", "$1$2$3", 0),
674     // (method(expression))
675     // method(expression)

```

```

658
659     (new Regex(@"(?<firstSeparator>(\(|\)
660         ))\((?<method>[a-zA-Z0-9_>\-*:]+)\)\((?<expression>((?<parenthesis>\()|(?<-parent>
661             hesis>\))|[a-zA-Z0-9_>\-*:]*)+)(?<parenthesis>(?!))\)\)(?<lastSeparator>(, |
662             ;|\))"), "${firstSeparator}${method}(${expression})${lastSeparator}", 0),
663             // .append(".")
664             // .append(1, '.');
665             (new Regex(@"\.\append\(\"([^\\""]|\\\"")\"\)", ".append(1, '$1')", 0),
666             // return ref _elements[node];
667             // return &_elements[node];
668             (new Regex(@"return ref ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9*]+)\];"), "return &$1[$2];",
669                 0),
670                 // ((1, 2))
671                 // (1, 2)
672                 (new Regex(@"(?<before>\(|, )\((?<first>[^n()]+,
673                     (?<second>[^n()]+)\)(?<after>)|, )"), "${before}${first},
674                     ${second}${after}", 10),
675                     // {1, 2}.GetHashCode()
676                     // Platform::Hashing::Hash(1, 2)
677                     (new Regex(@"{(?<first>[^n{}]+, (?<second>[^n{}]+)}\.\GetHashCode\(\)", ,
678                         "Platform::Hashing::Hash(${first}, ${second})", 10),
679                         // range.ToString()
680                         // Platform::Converters::To<std::string>(range).data()
681                         (new Regex(@"(?<before>\W)(?<variable>[_a-zA-Z][_a-zA-Z0-9]+)\.\ToString\(\)", ,
682                             "${before}Platform::Converters::To<std::string>(${variable}).data()", 10),
683                             // new
684                             //
685                             (new Regex(@"(?<before>\r?\n[^"\r\n]*("(\\"|[^"\r\n])*[^"\r\n]*)*) (?<=\W)new\_
686                                 s+"), "${before}",
687                                 10),
688                                 // x == null
689                                 // x == nullptr
690                                 (new Regex(@"(?<before>\r?\n[^"\r\n]*("(\\"|[^"\r\n])*[^"\r\n]*)*) (?<=\W)(?<v
691                                     ariable>[_a-zA-Z][_a-zA-Z0-9]+)(?<operator>\s*(==|!=)\s*)null(?<after>\W)",
692                                     "${before}${variable}${operator}nullptr${after}", 10),
693                                     // null
694                                     // {}
695                                     (new Regex(@"(?<before>\r?\n[^"\r\n]*("(\\"|[^"\r\n])*[^"\r\n]*)*) (?<=\W)null\_
696                                         (?<after>\W)", "${before}{}${after}",
697                                         10),
698                                         // default
699                                         // 0
700                                         (new Regex(@"(?<before>\r?\n[^"\r\n]*("(\\"|[^"\r\n])*[^"\r\n]*)*) (?<=\W)defa
701                                             ult(?<after>\W)", "${before}0${after}",
702                                             10),
703                                             // object x
704                                             // void *x
705                                             (new Regex(@"(?<before>\r?\n[^"\r\n]*("(\\"|[^"\r\n])*[^"\r\n]*)*) (?<=\W)(?<!
706                                                 @)(object|System\.\Object)(?<after>\w)", "${before}void *${after}",
707                                                 10),
708                                                 // <object>
709                                                 // <void*>
710                                                 (new Regex(@"(?<before>\r?\n[^"\r\n]*("(\\"|[^"\r\n])*[^"\r\n]*)*) (?<=\W)(?<!
711                                                 @)(object|System\.\Object)(?<after>\W)", "${before}void*${after}",
712                                                 10),
713                                                 // @object
714                                                 // object
715                                                 (new Regex(@"@([_a-zA-Z0-9]+)", "$1", 0),
716                                                 // this->GetType().Name
717                                                 // typeid(this).name()
718                                                 (new Regex(@"(this)->GetType\(\)\.Name"), "typeid($1).name()", 0),
719                                                 // ArgumentNullException
720                                                 // std::invalid_argument
721                                                 (new Regex(@"(?<before>\r?\n[^"\r\n]*("(\\"|[^"\r\n])*[^"\r\n]*)*) (?<=\W)(Sys
722                                                     tem\.)?ArgumentNullException(?<after>\W)",
723                                                     "${before}std::invalid_argument${after}", 10),
724                                                     // InvalidOperationException
725                                                     // std::runtime_error
726                                                     (new Regex(@"(\W)(InvalidOperationException|Exception)(\W)", ,
727                                                         "$1std::runtime_error$3", 0),
728                                                         // ArgumentException
729                                                         // std::invalid_argument
730                                                         (new Regex(@"(\W)(ArgumentException|ArgumentOutOfRangeException)(\W)", ,
731                                                             "$1std::invalid_argument$3", 0),
732                                                             // template <typename T> struct Range : IEquatable<Range<T>>
733                                                             // template <typename T> struct Range {

```

```

709     (new Regex(@"(?<before>template <typename> (?<typeParameter>[^`n]+)> (struct|class)
710     (?<type>[a-zA-Z0-9]+<[^`n]+>)) : (public
711     )?IEquatable<\k<type>>(?<after>(\s|\n)*{})", "${before}${after}", 0),
712     // public: delegate void Disposal(bool manual, bool wasDisposed);
713     // public: delegate void Disposal(bool, bool);
714     (new Regex(@"(?<before>(?<access>(private|protected|public): )delegate
715     (?<returnType>[a-zA-Z][a-zA-Z0-9:]++)
716     (?<delegate>[a-zA-Z][a-zA-Z0-9:]+)\((?<leftArgumentType>[a-zA-Z][a-zA-Z0-9:]+, 
717     *)?(<argumentType>[a-zA-Z][a-zA-Z0-9:]++)
718     (?<argumentName>[a-zA-Z][a-zA-Z0-9:]?)?<after>(
719     (?<rightArgumentType>[a-zA-Z][a-zA-Z0-9:]++)
720     (?<rightArgumentName>[a-zA-Z][a-zA-Z0-9:]+)\)*\);)", "${before}${argumentType}${after}", 20),
721     // public: delegate void Disposal(bool, bool);
722     // using Disposal = void(bool, bool);
723     (new Regex(@"(?<access>(private|protected|public): )delegate
724     (?<returnType>[a-zA-Z][a-zA-Z0-9:]++)
725     (?<delegate>[a-zA-Z][a-zA-Z0-9:]+)\((?<argumentTypes>[^(\n)*])\);", "using
726     ${delegate} = ${returnType}(${argumentTypes});", 20),
727     // <4-1>
728     // <3>
729     (new Regex(@"(?<before><)4-1(?<after>>)", "${before}3${after}", 0),
730     // <3-1>
731     // <2>
732     (new Regex(@"(?<before><)3-1(?<after>>)", "${before}2${after}", 0),
733     // <2-1>
734     // <1>
735     (new Regex(@"(?<before><)2-1(?<after>>)", "${before}1${after}", 0),
736     // <1-1>
737     // <0>
738     (new Regex(@"(?<before><)1-1(?<after>>)", "${before}0${after}", 0),
739     // #region Always
740     // 
741     (new Regex(@"(^|\r?\n)[ \t]*#\{region|endregion\}[^\r\n]*(\r?\n|$)", "", 0),
742     // //define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
743     // 
744     (new Regex(@"^/\/*[ \t]*#\{define[ \t]+[_a-zA-Z0-9]+[ \t]*", "", 0),
745     // #if USEARRAYPOOL\r\n#\endif
746     // 
747     (new Regex(@"#\{if [a-zA-Z0-9]+\s+\#\{endif\}", "", 0),
748     // [Fact]
749     // 
750     (new Regex(@"(?<firstNewLine>\r?\n|\A)(?<indent>[\t
751     ]+)\[[a-zA-Z0-9]+(\((?<expression>(?<parenthesis>\()|(?<-parenthesis>\))|[^()\r
752     \n]*\))+)?(parenthesis\(!\))\)\)?\] [ \t]*(\r?\n\k<indent>)?", "${firstNewLine}${indent}", 5),
753     // \A \n ... namespace
754     // \Anamespace
755     (new Regex(@"(\A)(\r?\n)+namespace", "$1namespace", 0),
756     // \A \n ... class
757     // \Aclass
758     (new Regex(@"(\A)(\r?\n)+class", "$1class", 0),
759     // \n\n\n
760     // \n\n
761     (new Regex(@"\r?\n[ \t]*\r?\n[ \t]*\r?\n", Environment.NewLine +
762     Environment.NewLine, 50),
763     // {\n\n
764     // {\n
765     (new Regex(@"{[ \t]*\r?\n[ \t]*\r?\n", "{" + Environment.NewLine, 10),
766     // \n\n
767     // \n
768     (new Regex(@"\r?\n[ \t]*\r?\n(\?<end>[ \t]*)", Environment.NewLine + "${end}", 10),
769     }.Cast<ISubstitutionRule>().ToList();
770 
771     /// <summary>
772     /// <para>
773     /// Initializes a new <see cref="CSharpToCppTransformer"/> instance.
774     /// </para>
775     /// <para></para>
776     /// </summary>
777     /// <param name="extraRules">
778     /// <para>A extra rules.</para>
779     /// <para></para>
780     /// </param>
781     public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
782     base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }

```

```

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

1.2 ./csharp/Platform.RegularExpressions.Transformer.CSharpToCppClass.Tests/CSharpToCppClassTransformerTests.cs
1  using Xunit;
2
3 namespace Platform.RegularExpressions.Transformer.CSharpToCppClass.Tests
4 {
5   public class CSharpToCppClassTransformerTests
6   {
7     [Fact]
8     public void EmptyLineTest()
9     {
10       // This test can help to test basic problems with regular expressions like incorrect
11       // syntax
12       var transformer = new CSharpToCppClassTransformer();
13       var actualResult = transformer.Transform("");
14       Assert.Equal("", actualResult);
15     }
16
17     [Fact]
18     public void HelloWorldTest()
19     {
20       const string helloWorldCode = @"using System;
21 class Program
22 {
23   public static void Main(string[] args)
24   {
25     Console.WriteLine("Hello, world!");
26   }
27 }
28
29 const string expectedResult = @"class Program
30 {
31   public: static void Main(std::string args[])
32   {
33     printf("Hello, world!\n");
34   }
35 }
36
37 var transformer = new CSharpToCppClassTransformer();
38 var actualResult = transformer.Transform(helloWorldCode);
39 Assert.Equal(expectedResult, actualResult);
40   }
41 }
42 }

```

Index

./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs, 18
./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1