

## 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     public class CSharpToCppTransformer : TextTransformer
11     {
12         public static readonly IList

```

```

58 //
59 (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),
60 // public override bool Equals(object obj) => obj is Range<T> range ? Equals(range)
    ↳ : false;
61 //
62 (new Regex(@"\r?\n[^\n]+override bool Equals\((System\.)?[Oo]bject
    ↳ (?<this>[a-zA-Z0-9]+)\) => \k<this> is [^\n]+ (?<other>[a-zA-Z0-9]+) \?
    ↳ Equals\(\k<other>\) : false;"), "", 10),
63 // out TProduct
64 // TProduct
65 (new Regex(@"(?<before><|, ))(in|out)
    ↳ (?<typeParameter>[a-zA-Z0-9]+)(?<after>>|,))"),
    ↳ "${before}${typeParameter}${after}", 10),
66 // public ...
67 // public: ...
68 (new Regex(@"(?<newLineAndIndent>\r?\n?[
    ↳ \t]*) (?<before>[^\{\\(\r\n)*) (?<access>private|protected|public) [ \t]+(?![^\{\\(\r
    ↳ \n]*((?<=\\s)|\\W) (interface|class|struct) (\\W) [^\{\\(\r\n)*[\\{\\(\r\n)]")",
    ↳ "${newLineAndIndent}${access}: ${before}", 0),
69 // public: static bool CollectExceptions { get; set; }
70 // public: inline static bool CollectExceptions;
71 (new Regex(@"(?<access>(private|protected|public): ) (?<before>(static )? [^\r\n]+
    ↳ ) (?<name>[a-zA-Z0-9]+) { [^;]* (?<=\\W) get; [^;]* (?<=\\W) set; [^;]* }"),
    ↳ "${access}inline ${before}${name};", 0),
72 // public abstract class
73 // class
74 (new Regex(@"((public|protected|private|internal|abstract|static)
    ↳ )*(?<category>interface|class|struct)", "${category}", 0),
75 //class GenericCollectionMethodsBase<TElement> {
76 // template <typename TElement> class GenericCollectionMethodsBase {
77 (new Regex(@"(?<before>\r?\n) (?<indent>[ \t]*) (?<type>class|struct)
    ↳ (?<typeName>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9
    ↳ ,]+)> (?<typeDefinitionEnding>[^\{]+){", "${before}${indent}template <typename
    ↳ ...> ${type} ${typeName};" + Environment.NewLine + "${indent}template <typename
    ↳ ${typeParameters}> ${type}
    ↳ ${typeName}<${typeParameters}>${typeDefinitionEnding}{", 0),
78 // static void
    ↳ TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
    ↳ tree, TElement* root)
79 // template<typename T> static void
    ↳ TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
    ↳ tree, TElement* root)
80 (new Regex(@"static ([a-zA-Z0-9]+) ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>\\(((\\r\\n)+)\\)",
    ↳ "template <typename $3> static $1 $2($4)", 0),
81 // interface IFactory<out TProduct> {
82 // template <typename...> class IFactory; \ntemplate <typename TProduct> class
    ↳ IFactory<TProduct>
83 (new Regex(@"(?<before>\r?\n) (?<indent>[ \t]*) interface
    ↳ (?<interface>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9
    ↳ ,]+)> (?<typeDefinitionEnding>[^\{]+){", "${before}${indent}template <typename
    ↳ ...> class ${interface};" + Environment.NewLine + "${indent}template <typename
    ↳ ${typeParameters}> class
    ↳ ${interface}<${typeParameters}>${typeDefinitionEnding}{", 0),
    ↳ "public:", 0),
84 // template <typename TObject, TProperty, TValue>
85 // template <typename TObject, typename TProperty, typename TValue>
86 (new Regex(@"(?<before>template <((, )?typename [a-zA-Z0-9]+)+,
    ↳ ) (?<typeParameter>[a-zA-Z0-9]+) (?<after>(,|>))"), "${before}typename
    ↳ ${typeParameter}${after}", 10),
87 // Insert markers
88 // private: static void BuildExceptionString(this StringBuilder sb, Exception
    ↳ exception, int level)
89 // /*~extensionMethod~BuildExceptionString~*/private: static void
    ↳ BuildExceptionString(this StringBuilder sb, Exception exception, int level)
90 (new Regex(@"private: static [^\r\n]+ (?<name>[a-zA-Z0-9]+)\\(this [^\r\n]+\\)",
    ↳ "/*~extensionMethod~${name}~*/$0", 0),
91 // Move all markers to the beginning of the file.
92 (new Regex(@"\A(?<before>[^\r\n]+\r?\n(.|\n)+) (?<marker>\/\*~extensionMethod~(?<name>
    ↳ [a-zA-Z0-9]+)~\*/)", "${marker}${before}",
    ↳ 10),
93 // /*~extensionMethod~BuildExceptionString~*/...sb.BuildExceptionString(exception.In_
    ↳ nerException, level +
    ↳ 1);

```

```

94 // /*~extensionMethod~BuildExceptionString~*/...BuildExceptionString(sb,
95   ↳ exception.InnerException, level + 1);
96 (new Regex(@"(?<before>/\/*~extensionMethod~(?<name>[a-zA-Z0-9]+)~\*/(.\|\\n)+\\W)(?<var_
97   ↳ iable>[_a-zA-Z0-9]+)\\.\\k<name>\\("), "${before}${name}${{variable}}, ",
98   ↳ 50),
99 // Remove markers
100 // /*~extensionMethod~BuildExceptionString~*/
101 //
102 (new Regex(@"\/*~extensionMethod~[a-zA-Z0-9]+~\*/"), "", 0),
103 // (this
104 // (
105 (new Regex(@"\((this ", "(", 0),
106 // public: static readonly EnsureAlwaysExtensionRoot Always = new
107   ↳ EnsureAlwaysExtensionRoot();
108 // public: inline static EnsureAlwaysExtensionRoot Always;
109 (new Regex(@"(?<access>(private|protected|public): )?static readonly
110   ↳ (?<type>[a-zA-Z0-9]+(<[a-zA-Z0-9]+>?)) (?<name>[a-zA-Z0-9_]+) = new
111   ↳ \k<type>\\(");", "${access}inline static ${type} ${name};", 0),
112 // public: static readonly Range<int> SByte = new
113   ↳ Range<int>(std::numeric_limits<int>::min(), std::numeric_limits<int>::max());
114 // public: inline static Range<int> SByte =
115   ↳ Range<int>(std::numeric_limits<int>::min(), std::numeric_limits<int>::max());
116 (new Regex(@"(?<access>(private|protected|public): )?static readonly
117   ↳ (?<type>[a-zA-Z0-9]+(<[a-zA-Z0-9]+>?)) (?<name>[a-zA-Z0-9_]+) = new
118   ↳ \k<type>\\((?<arguments>[^\n]+)\\");", "${access}inline static ${type} ${name} =
119   ↳ ${type}${{arguments}};", 0),
120 // public: static readonly string ExceptionContentsSeparator = "---";
121 // public: inline static std::string ExceptionContentsSeparator = "---";
122 (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly) string
123   ↳ (?<name>[a-zA-Z0-9_]+) = ""(?<string>(\\\\"|"[^"]*"\\r\\n))+"";", "${access}inline
124   ↳ static std::string ${name} = \"${string}\";", 0),
125 // private: const int MaxPath = 92;
126 // private: inline static const int MaxPath = 92;
127 (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly)
128   ↳ (?<type>[a-zA-Z0-9]+) (?<name>[_a-zA-Z0-9]+) = (?<value>[^\r\n]+);",
129   ↳ "${access}inline static const ${type} ${name} = ${value};", 0),
130 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
131   ↳ TArgument : class
132 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
133 (new Regex(@"(?<before> [a-zA-Z]+\\((([a-zA-Z *,]+, |)) (?<type>[a-zA-Z]+) (?<after>(\\
134   ↳ [a-zA-Z *,]+)\\))) [\\r\\n]+where \\k<type> : class", "${before}${type}*${after}",
135   ↳ 0),
136 // protected: abstract TElement GetFirst();
137 // protected: virtual TElement GetFirst() = 0;
138 (new Regex(@"(?<access>(private|protected|public): )?abstract
139   ↳ (?<method>[^\r\n]+);", "${access}virtual ${method} = 0;", 0),
140 // TElement GetFirst();
141 // virtual TElement GetFirst() = 0;
142 (new Regex(@"(?<before>[\\r\\n]+[ ]+)(?<methodDeclaration>(?!return) [a-zA-Z0-9]+
143   ↳ [a-zA-Z0-9]+\\(([^\\r\\n]*\\)) (?<after>[ ]*[\\r\\n]+)\"", "${before}virtual
144   ↳ ${methodDeclaration} = 0${after}", 1),
145 // protected: readonly TreeElement[] _elements;
146 // protected: TreeElement _elements[N];
147 (new Regex(@"(?<access>(private|protected|public): )?readonly
148   ↳ (?<type>[a-zA-Z<>0-9]+)([\\[]]+) (?<name>[_a-zA-Z0-9]+);", "${access}${type}
149   ↳ ${name}[N];", 0),
150 // protected: readonly TElement Zero;
151 // protected: TElement Zero;
152 (new Regex(@"(?<access>(private|protected|public): )?readonly
153   ↳ (?<type>[a-zA-Z<>0-9]+) (?<name>[_a-zA-Z0-9]+);", "${access}${type} ${name};",
154   ↳ 0),
155 // internal
156 //
157 (new Regex(@"(\\W)internal\\s+)", "$1", 0),
158 // static void NotImplementedException(ThrowExtensionRoot root) => throw new
159   ↳ NotImplementedException();
160 // static void NotImplementedException(ThrowExtensionRoot root) { return throw new
161   ↳ NotImplementedException(); }
162 (new Regex(@"(\\s+)(private|protected|public)?(: )?(template \\<[^\\r\\n]+\\> )?(static
163   ↳ )?(override )?(?<[a-zA-Z0-9]+
164   ↳ )([a-zA-Z0-9]+)\\(((\\r\\n)*\\)\\s+=>\\s+throw([^\r\n]+);)",
165   ↳ "$1$2$3$4$5$6$7$8($9) { throw$10; }", 0),
166 // SizeBalancedTree(int capacity) => a = b;
167 // SizeBalancedTree(int capacity) { a = b; }

```

```

138 (new Regex(@"(^s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static
    ↳ )?(override )?(void )?([a-zA-Z0-9]+)\(((^\\(\\r\\n)*)\\)\s+=>\s+([~;\r\n]+);"),
    ↳ "$1$2$3$4$5$6$7$8($9) { $10; }", 0),
139 // int SizeBalancedTree(int capacity) => a;
140 // int SizeBalancedTree(int capacity) { return a; }
141 (new Regex(@"(^s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static
    ↳ )?(override )?([a-zA-Z0-9]+
    ↳ )([a-zA-Z0-9]+)\(((^\\(\\r\\n)*)\\)\s+=>\s+([~;\r\n]+);"), "$1$2$3$4$5$6$7$8($9) {
    ↳ return $10; }", 0),
142 // () => Integer<TElement>.Zero,
143 // () { return Integer<TElement>.Zero; },
144 (new Regex(@"\\(\\)\s+=>\s+(?<expression>[~() ,;\r\n]+(\\(((?<parenthesis>\\()|(?<-parent
    ↳ hesis>\\))| [~() ,;\r\n]*?\\)))?[~() ,;\r\n]*(?<after>,|\\);)"), "() { return
    ↳ ${expression}; }${after}", 0),
145 // => Integer<TElement>.Zero;
146 // { return Integer<TElement>.Zero; }
147 (new Regex(@"\\)\s+=>\s+([~;\r\n]+?);"), ") { return $1; }", 0),
148 // () { return avlTree.Count; }
149 // [&]() -> auto { return avlTree.Count; }
150 (new Regex(@"(?<before>, |\\()\\(\\) { return (?<expression>[~;\r\n]+); }"),
    ↳ "${before}[&]() -> auto { return ${expression}; }", 0),
151 // Count => GetSizeOrZero(Root);
152 // GetCount() { return GetSizeOrZero(Root); }
153 (new Regex(@"\\(\\W)([A-Z][a-zA-Z]+)\s+=>\s+([~;\r\n]+);"), "$1Get$2() { return $3; }",
    ↳ 0),
154 // ArgumentInRange(string message) { string messageBuilder() { return message; }
155 // ArgumentInRange(string message) { auto messageBuilder = [&]() -> string { return
    ↳ message; };
156 (new Regex(@"(?<before>\\W[_a-zA-Z0-9]+\\((^\\)\n)*\\[\\s\\n]*{[\\s\\n]*([~{]}|\\n)*?(\\r?\\n)
    ↳ ?[ \\t]*)(?<returnType>[_a-zA-Z0-9*:] +[_a-zA-Z0-9*:] *)
    ↳ (?<methodName>[_a-zA-Z0-9]+)\\(((?<arguments>[~\\)\n]*\\)\s*{(?<body>("[^""\n]+""|
    ↳ [~}]|\\n)+?)}")", "${before}auto ${methodName} = [&]() -> ${returnType}
    ↳ {${body}};", 10),
157 // Func<TElement> treeCount
158 // std::function<TElement()> treeCount
159 (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", 0),
160 // Action<TElement> free
161 // std::function<void(TElement)> free
162 (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<void($1)> $2",
    ↳ 0),
163 // Predicate<TArgument> predicate
164 // std::function<bool(TArgument)> predicate
165 (new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<bool($1)>
    ↳ $2", 0),
166 // var
167 // auto
168 (new Regex(@"\\(\\W)var(\\W)"), "$1auto$2", 0),
169 // unchecked
170 //
171 (new Regex(@"[\\r\\n]{2}\\s*?unchecked\\s*?$"), "", 0),
172 // throw new
173 // throw
174 (new Regex(@"\\(\\W)throw new(\\W)"), "$1throw$2", 0),
175 // void RaiseExceptionIgnoredEvent(Exception exception)
176 // void RaiseExceptionIgnoredEvent(const std::exception& exception)
177 (new Regex(@"\\(|, )(System\\.Exception|Exception)( |\\)\\)", "$1const
    ↳ std::exception&$3", 0),
178 // EventHandler<Exception>
179 // EventHandler<std::exception>
180 (new Regex(@"\\(\\W)(System\\.Exception|Exception)(\\W)"), "$1std::exception$3", 0),
181 // override void PrintNode(TElement node, StringBuilder sb, int level)
182 // void PrintNode(TElement node, StringBuilder sb, int level) override
183 (new Regex(@"override ([a-zA-Z0-9 \\*+] +)\\((^\\)\r\n)+?\\)\\)", "$1$2 override", 0),
184 // return (range.Minimum, range.Maximum)
185 // return {range.Minimum, range.Maximum}
186 (new Regex(@"(?<before>return\\s*)\\(((?<values>[~\\)\n]+)\\)(?!\\() (?<after>\\W)",
    ↳ "${before}${values}${after}", 0),
187 // string
188 // std::string
189 (new Regex(@"(?<before>\\W)(?!::)string(?<after>\\W)"),
    ↳ "${before}std::string${after}", 0),
190 // System.ValueTuple
191 // std::tuple
192 (new Regex(@"(?<before>\\W)(System\\.)?ValueTuple(?:\\s*=|\\() (?<after>\\W)",
    ↳ "${before}std::tuple${after}", 0),
193 // sbyte

```

```

194 // std::int8_t
195 (new Regex(@"(?<before>\W)((System\.)?SB|sb)yte(?:\s*=\|()\ (?<after>\W)"),
196   ↳ "${before}std::int8_t${after}", 0),
197 // short
198 // std::int16_t
199 (new Regex(@"(?<before>\W)((System\.)?Int16|short)(?:\s*=\|()\ (?<after>\W)"),
200   ↳ "${before}std::int16_t${after}", 0),
201 // int
202 // std::int32_t
203 (new Regex(@"(?<before>\W)((System\.)?I|i)nt(32)?(?:\s*=\|()\ (?<after>\W)"),
204   ↳ "${before}std::int32_t${after}", 0),
205 // long
206 // std::int64_t
207 (new Regex(@"(?<before>\W)((System\.)?Int64|long)(?:\s*=\|()\ (?<after>\W)"),
208   ↳ "${before}std::int64_t${after}", 0),
209 // byte
210 // std::uint8_t
211 (new Regex(@"(?<before>\W)((System\.)?Byte|byte)(?:\s*=\|()\ (?<after>\W)"),
212   ↳ "${before}std::uint8_t${after}", 0),
213 // ushort
214 // std::uint16_t
215 (new Regex(@"(?<before>\W)((System\.)?UInt16|ushort)(?:\s*=\|()\ (?<after>\W)"),
216   ↳ "${before}std::uint16_t${after}", 0),
217 // uint
218 // std::uint32_t
219 (new Regex(@"(?<before>\W)((System\.)?UI|ui)nt(32)?(?:\s*=\|()\ (?<after>\W)"),
220   ↳ "${before}std::uint32_t${after}", 0),
221 // ulong
222 // std::uint64_t
223 (new Regex(@"(?<before>\W)((System\.)?UInt64|ulong)(?:\s*=\|()\ (?<after>\W)"),
224   ↳ "${before}std::uint64_t${after}", 0),
225 // char*[] args
226 // char* args[]
227 (new Regex(@"([_a-zA-Z0-9:\*]?)\[\] ([a-zA-Z0-9]+)", "$1 $2[]", 0),
228 // @object
229 // object
230 (new Regex(@"@([_a-zA-Z0-9]+)", "$1", 0),
231 // float.MinValue
232 // std::numeric_limits<float>::lowest()
233 (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MinValue(?<after>\W)",
234   ↳ "${before}std::numeric_limits<${type}>::lowest()${after}",
235   ↳ 0),
236 // double.MaxValue
237 // std::numeric_limits<float>::max()
238 (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MaxValue(?<after>\W)",
239   ↳ "${before}std::numeric_limits<${type}>::max()${after}",
240   ↳ 0),
241 // using Platform.Numbers;
242 //
243 (new Regex(@"([\r\n]{2}|~)\s*using [_a-zA-Z0-9]+;\s*?$)", "", 0),
244 // struct TreeElement { }
245 // struct TreeElement { };
246 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([~;])", "$1
247   ↳ $2$3{$4};$5", 0),
248 // class Program { }
249 // class Program { };
250 (new Regex(@"(?<type>struct|class)
251   ↳ (?<name>[a-zA-Z0-9]+[~\r\n]*) (?<beforeBody>[\r\n]+(?<indentLevel>[\t
252   ↳ ]*)?)\{(?<body>[\S\s]+?[~\r\n]+\k<indentLevel>)\}(?<afterBody>[~;]|$)", "${type}
253   ↳ ${name}${beforeBody}${body}};${afterBody}", 0),
254 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
255 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
256 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(<[a-zA-Z0-9 ,]+>)? : ([a-zA-Z0-9]+)",
257   ↳ "$1 $2$3 : public $4", 0),
258 // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
259 // class IProperty : public ISetter<TValue, TObject>, public IProvider<TValue,
260   ↳ TObject>
261 (new Regex(@"(?<before>(struct|class) [a-zA-Z0-9]+ : ((public
262   ↳ [a-zA-Z0-9]+(<[a-zA-Z0-9 ,]+>)?,
263   ↳ )+)?(?<inheritedType>(?!public)[a-zA-Z0-9]+(<[a-zA-Z0-9 ,]+>)?(?<after>(,
264   ↳ [a-zA-Z0-9]+(?>)|[~\r\n]+)))", "${before}public ${inheritedType}${after}", 10),
265 // Insert scope borders.
266 // ref TElement root
267 // ~!root!~ref TEElement root
268 (new Regex(@"(?<definition>(?!= \|()\ (ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!ref))
269   ↳ (?<variable>[a-zA-Z0-9]+)(?= \|, \| =)")), "~!${variable}!~${definition}", 0),

```

```

248 // Inside the scope of ~!root!~ replace:
249 // root
250 // *root
251 (new Regex(@"(?<definition>~!(?<pointer>[a-zA-Z0-9]+)!~ref [a-zA-Z0-9]+
    ↳ \k<pointer>(=?\)|,| =)) (?<before>((?<!\k<pointer>!~)(.|\\n))*?) (?<prefix>(\W
    ↳ |\\())\k<pointer>( ?<suffix>( |\\)|;|,))"),
    ↳ "$${definition}$$before}$$prefix}*${pointer}$$suffix}", 70),
252 // Remove scope borders.
253 // ~!root!~
254 //
255 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", 5),
256 // ref auto root = ref
257 // ref auto root =
258 (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\\W)", "$1* $2 =$3", 0),
259 // *root = ref left;
260 // root = left;
261 (new Regex(@"\\*( [a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\\W)", "$1 = $2$3", 0),
262 // (ref left)
263 // (left)
264 (new Regex(@"\\(ref ([a-zA-Z0-9]+)(\\)|\\(|,))", "($1$2", 0),
265 // ref TElement
266 // TElement*
267 (new Regex(@"( |\\())ref ([a-zA-Z0-9]+) ", "$1$2* ", 0),
268 // ref sizeBalancedTree.Root
269 // &sizeBalancedTree->Root
270 (new Regex(@"ref ([a-zA-Z0-9]+)\\.([a-zA-Z0-9\\*]+)", "&$1->$2", 0),
271 // ref GetElement(node).Right
272 // &GetElement(node)->Right
273 (new Regex(@"ref ([a-zA-Z0-9]+)\\(( [a-zA-Z0-9\\*]+)\\)\\.([a-zA-Z0-9]+)",
    ↳ "&$1($2)->$3", 0),
274 // GetElement(node).Right
275 // GetElement(node)->Right
276 (new Regex(@"([a-zA-Z0-9]+)\\(( [a-zA-Z0-9\\*]+)\\)\\.([a-zA-Z0-9]+)", "$1($2)->$3", 0),
277 // [Fact]npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
278 // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
279 (new Regex(@"\\[Fact\\] [\\s\\n]+(public: )?(static )?void ([a-zA-Z0-9]+)\\(\\)", "public:
    ↳ TEST_METHOD($3)", 0),
280 // class TreesTests
281 // TEST_CLASS(TreesTests)
282 (new Regex(@"class ([a-zA-Z0-9]+Tests)", "TEST_CLASS($1)", 0),
283 // Assert.Equal
284 // Assert::AreEqual
285 (new Regex(@"(?<type>Assert)\\. (?<method>(Not)?Equal)", "${type}::Are${method}", 0),
286 // Assert.Throws
287 // Assert::ExpectException
288 (new Regex(@"(Assert)\\.Throws", "$1::ExpectException", 0),
289 // Assert.True
290 // Assert::IsTrue
291 (new Regex(@"(Assert)\\. (True|False)", "$1::Is$2", 0),
292 // $"Argument {argumentName} is null."
293 // std::string("Argument
    ↳ ").append(Platform::Converters::To<std::string>(argumentName)).append(" is
    ↳ null.")
294 (new Regex(@"\\$"" (?<left>(\\\\" | [^""\\r\\n])* ) { (?<expression>[_a-zA-Z0-9]+) } { (?<right>(\\
    ↳ \\" | [^""\\r\\n])* ) """,
    ↳ "std::string(\\$""${left}\\").append(Platform::Converters::To<std::string>({expres
    ↳ sion})).append(\\$""${right}\\")",
    ↳ 10),
295 // $"
296 // "
297 (new Regex(@"\\$"""), "\\\"", 0),
298 // std::string(std::string("[").append(Platform::Converters::To<std::string>(Minimum
    ↳ )), append(",
    ↳ )), append(Platform::Converters::To<std::string>(Maximum)).append("]")
299 // std::string("[").append(Platform::Converters::To<std::string>(Minimum)).append(",
    ↳ ").append(Platform::Converters::To<std::string>(Maximum)).append("]")
300 (new Regex(@"std::string\\((?<begin>std::string\\( ""(\\\\" | [^""\\r\\n])* ""\\) (\\
    ↳ .append\\((Platf
    ↳ orm::Converters::To<std::string>\\((~)\\n]+\\)| [^~)\\n]+\\))\\)\\)\\.append)",
    ↳ "${begin}.append", 10),
301 // Console.WriteLine("...")
302 // printf("...\\n")
303 (new Regex(@"Console\\.WriteLine\\( ""([~""\\r\\n]+) ""\\)", "printf(\\$""$1\\n\\n)", 0),
304 // TElement Root;
305 // TElement Root = 0;
306 (new Regex(@"(?<before>\\r?\\n[\\t ]+) (?<access>(private|protected|public)(:
    ↳ ))? (?<type>[a-zA-Z0-9:]+) (?<return>) (?<name>[_a-zA-Z0-9]+);"),
    ↳ "${before}$$access}$$type} ${name} = 0;", 0),

```

```

307 // TreeElement _elements[N];
308 // TreeElement _elements[N] = { {0} };
309 (new Regex(@"(\\r?\\n[\\t ]+)(private|protected|public)?(: )?([a-zA-Z0-9]+)
    ↳ ([_a-zA-Z0-9]+)\\([[_a-zA-Z0-9]+)\\];") , "$1$2$3$4 $5[$6] = { {0} };", 0),
310 // auto path = new TElement[MaxPath];
311 // TElement path[MaxPath] = { {0} };
312 (new Regex(@"(\\r?\\n[\\t ]+)[a-zA-Z0-9]+ ([a-zA-Z0-9]+) = new
    ↳ ([_a-zA-Z0-9]+)\\([[_a-zA-Z0-9]+)\\];") , "$1$3 $2[$4] = { {0} };", 0),
313 // bool Equals(Range<T> other) { ... }
314 // bool operator ==(const Key &other) const { ... }
315 (new Regex(@"(?<before>\\r?\\n[\\n]+bool )Equals\\((?<type>[\\n]+)
    ↳ (?<variable>[_a-zA-Z0-9]+)\\)(?<after>(\\s|\\n)*{") , "${before}operator ==(const
    ↳ ${type} &${variable}) const${after}", 0),
316 // Insert scope borders.
317 // class Range { ... public: override std::string ToString() { return ...; }
318 // class Range { /*~Range<T>~*/ ... public: override std::string ToString() { return
    ↳ ...; }
319 (new Regex(@"(?<classDeclarationBegin>\\r?\\n(?<indent>[\\t ]*)template <typename
    ↳ (?<typeParameter>[~<>\\n]+> (struct|class)
    ↳ (?<type>[_a-zA-Z0-9]+<k<typeParameter>>)(\\s*:\\s*[~{\\n]+)?[\\t ]*(\\r?\\n)?[\\t
    ↳ ]*(?<middle>((?!class|struct)\\.\\n)+?)?(?<toStringDeclaration>(?!<access>(private|
    ↳ |protected|public): )override std::string ToString\\(\\))") ,
    ↳ "${classDeclarationBegin}/*~${type}~*/${middle}${toStringDeclaration}", 0),
320 // Inside the scope of ~!Range!~ replace:
321 // public: override std::string ToString() { return ...; }
322 // public: operator std::string() const { return ...; }\\n\\npublic: friend
    ↳ std::ostream & operator <<(std::ostream &out, const A &obj) { return out <<
    ↳ (std::string)obj; }
323 (new Regex(@"(?<scope>/\\s*(?<type>[_a-zA-Z0-9<>:]+)~\\s*/)(?<separator>\\.\\n)(?<before>
    ↳ ((?!/\\s*~\\k<type>~\\s*/)(\\.\\n)*)?(?<toStringDeclaration>\\r?\\n(?<indent>[
    ↳ \\t]*)?(?<access>(private|protected|public): )override std::string ToString\\(\\)
    ↳ (?<toStringMethodBody>[~}\\n]+))") , "${scope}${separator}${before}" +
    ↳ Environment.NewLine + "${indent}${access}operator std::string() const
    ↳ ${toStringMethodBody}" + Environment.NewLine + Environment.NewLine +
    ↳ "${indent}${access}friend std::ostream & operator <<(std::ostream &out, const
    ↳ ${type} &obj) { return out << (std::string)obj; }", 0),
324 // Remove scope borders.
325 // /*~Range~*/
326 //
327 (new Regex(@"/\\s*~[_a-zA-Z0-9<>:]+~\\s*/") , "", 0),
328 // private: inline static ConcurrentBag<std::exception> _exceptionsBag;
329 // private: inline static std::mutex _exceptionsBag_mutex; \\n\\n private: inline
    ↳ static std::vector<std::exception> _exceptionsBag;
330 (new Regex(@"(?<begin>\\r?\\n(?<indent>[\\t ]+))?(?<access>(private|protected|public):
    ↳ )?inline static ConcurrentBag<(?!<argumentType>[~;\\r\\n]+)>
    ↳ (?<name>[_a-zA-Z0-9]+);") , "${begin}private: inline static std::mutex
    ↳ ${name}_mutex;" + Environment.NewLine + Environment.NewLine +
    ↳ "${indent}${access}inline static std::vector<${argumentType}> ${name};", 0),
331 // public: static IReadonlyCollection<std::exception> GetCollectedExceptions() {
    ↳ return _exceptionsBag; }
332 // public: static std::vector<std::exception> GetCollectedExceptions() { return
    ↳ std::vector<std::exception>(_exceptionsBag); }
333 (new Regex(@"(?<access>(private|protected|public): )?static
    ↳ IReadonlyCollection<(?!<argumentType>[~;\\r\\n]+)> (?<methodName>[_a-zA-Z0-9]+)\\(\\)
    ↳ { return (?<fieldName>[_a-zA-Z0-9]+); }") , "${access}static
    ↳ std::vector<${argumentType}> ${methodName}() { return
    ↳ std::vector<${argumentType}>({${fieldName}}); }", 0),
334 // public: static event EventHandler<std::exception> ExceptionIgnored =
    ↳ OnExceptionIgnored; ... };
335 // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
    ↳ const std::exception&> ExceptionIgnored = OnExceptionIgnored; };
336 (new Regex(@"(?<begin>\\r?\\n(\\r?\\n)?(?<halfIndent>[
    ↳ \\t]+)\\k<halfIndent>)(?<access>(private|protected|public): )?static event
    ↳ EventHandler<(?!<argumentType>[~;\\r\\n]+)> (?<name>[_a-zA-Z0-9]+) = (?<defaultDele
    ↳ gate>[_a-zA-Z0-9]+);(?<middle>(\\.\\n)+?)?(?<end>\\r?\\n\\k<halfIndent>});") ,
    ↳ "${middle}" + Environment.NewLine + Environment.NewLine +
    ↳ "${halfIndent}${halfIndent}${access}static inline
    ↳ Platform::Delegates::MulticastDelegate<void(void*, const ${argumentType}&>
    ↳ ${name} = ${defaultDelegate};${end}", 0),
337 // Insert scope borders.
338 // class IgnoredExceptions { ... private: inline static std::vector<std::exception>
    ↳ _exceptionsBag;
339 // class IgnoredExceptions { /*~_exceptionsBag~*/ ... private: inline static
    ↳ std::vector<std::exception> _exceptionsBag;

```



```

340 (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),
341 // Inside the scope of ~!_exceptionsBag!~ replace:
342 // _exceptionsBag.Add(exception);
343 // _exceptionsBag.push_back(exception);
344 (new Regex(@"(?<scope>/\s*(?<fieldName>[_a-zA-Z0-9]+)~\s*/) (?<separator>.\n) (?<before>
→ e>((?!/\s*~\k<fieldName>~\s*/)(\n))*?)\k<fieldName>\.Add"),
→ "${scope}${separator}${before}${fieldName}.push_back", 10),
345 // Remove scope borders.
346 // /*~_exceptionsBag~*/
347 //
348 (new Regex(@"/\s*~[_a-zA-Z0-9]+~\s*/"), "", 0),
349 // Insert scope borders.
350 // class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
351 // class IgnoredExceptions { /*~_exceptionsBag~*/ ... private: static std::mutex
→ _exceptionsBag_mutex;
352 (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),
353 // Inside the scope of ~!_exceptionsBag!~ replace:
354 // return std::vector<std::exception>(_exceptionsBag);
355 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
→ std::vector<std::exception>(_exceptionsBag);
356 (new Regex(@"(?<scope>/\s*(?<fieldName>[_a-zA-Z0-9]+)~\s*/) (?<separator>.\n) (?<before>
→ e>((?!/\s*~\k<fieldName>~\s*/)(\n))*?) { (?<after>((?!lock_guard) [^{};\r\n]) * \k<f
→ ieldName>[^\r\n]*);)", "${scope}${separator}${before}{
→ std::lock_guard<std::mutex> guard(${fieldName}_mutex); ${after}", 10),
357 // Inside the scope of ~!_exceptionsBag!~ replace:
358 // _exceptionsBag.Add(exception);
359 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
→ _exceptionsBag.Add(exception);
360 (new Regex(@"(?<scope>/\s*(?<fieldName>[_a-zA-Z0-9]+)~\s*/) (?<separator>.\n) (?<before>
→ e>((?!/\s*~\k<fieldName>~\s*/)(\n))*?) { (?<after>((?!lock_guard) ([^{};]|\n))*? \r
→ ?\n (?<indent>[\t ]*) \k<fieldName>[^\r\n]*);)",
→ "${scope}${separator}${before}{ " + Environment.NewLine +
→ "${indent}std::lock_guard<std::mutex> guard(${fieldName}_mutex); ${after}", 10),
361 // Remove scope borders.
362 // /*~_exceptionsBag~*/
363 //
364 (new Regex(@"/\s*~[_a-zA-Z0-9]+~\s*/"), "", 0),
365 // Insert scope borders.
366 // class IgnoredExceptions { ... public: static inline
→ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
→ ExceptionIgnored = OnExceptionIgnored;
367 // class IgnoredExceptions { /*~ExceptionIgnored~*/ ... public: static inline
→ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
→ ExceptionIgnored = OnExceptionIgnored;
368 (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n}+\r\n[\t
→ ]*(?<middle>((?!class)\.|\n)+?) (?<eventDeclaration>(?<access>(private|protected|
→ public): )static inline
→ Platform::Delegates::MulticastDelegate<(?<argumentType>[^\r\n]+)>
→ (?<name>[_a-zA-Z0-9]+) = (?<defaultDelegate>[_a-zA-Z0-9]+);)",
→ "${classDeclarationBegin}/*~${name}~/${middle}${eventDeclaration}", 0),
369 // Inside the scope of ~!ExceptionIgnored!~ replace:
370 // ExceptionIgnored.Invoke(NULL, exception);
371 // ExceptionIgnored(NULL, exception);
372 (new Regex(@"(?<scope>/\s*(?<eventName>[_a-zA-Z0-9]+)~\s*/) (?<separator>.\n) (?<before>
→ >((?!/\s*~\k<eventName>~\s*/)(\n))*?) \k<eventName>\.Invoke"),
→ "${scope}${separator}${before}${eventName}", 10),
373 // Remove scope borders.
374 // /*~ExceptionIgnored~*/
375 //
376 (new Regex(@"/\s*~[_a-zA-Z0-9]+~\s*/"), "", 0),
377 // Insert scope borders.
378 // auto added = new StringBuilder();
379 // /*~sb~*/std::string added;
380 (new Regex(@"(auto|(System\.\Text\.)?StringBuilder) (?<variable>[_a-zA-Z0-9]+) = new
→ (System\.\Text\.)?StringBuilder\(\);)", "/*~${variable}~*/std::string
→ ${variable};", 0),
381 // static void Indent(StringBuilder sb, int level)
382 // static void Indent(/*~sb~*/StringBuilder sb, int level)

```



```

383 (new Regex(@"(?<start>, |\) (System\Text\.)?StringBuilder
    ↳ (?<variable>[a-zA-Z0-9]+)(?<end>, |\)") , "${start}/*~${variable}~/std::string&
    ↳ ${variable}${end}", 0),
384 // Inside the scope of ~!added!~ replace:
385 // sb.ToString()
386 // sb
387 (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↳ ((?!/*~\k<variable>~\*/)(.\|\\n))*?)\k<variable>\.ToString\\("),
    ↳ "${scope}${separator}${before}${variable}", 10),
388 // sb.AppendLine(argument)
389 // sb.append(Platform::Converters::To<std::string>(argument)).append(1, '\\n')
390 (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↳ ((?!/*~\k<variable>~\*/)(.\|\\n))*?)\k<variable>\.AppendLine\\((?<argument>[^\|\\n], \\
    ↳ r\\n)+\\)"),
    ↳ "${scope}${separator}${before}${variable}.append(Platform::Converters::To<std::s
    ↳ tring>(${argument}))".append(1, '\\n')",
    ↳ 10),
391 // sb.Append('\\t', level);
392 // sb.append(level, '\\t');
393 (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↳ ((?!/*~\k<variable>~\*/)(.\|\\n))*?)\k<variable>\.Append\\('(?<character>[^\|\\r\\n]
    ↳ +)', (?<count>[^\|\\n]+)\\)"),
    ↳ "${scope}${separator}${before}${variable}.append(${count}, '${character}')" , 10),
394 // sb.Append(argument)
395 // sb.append(Platform::Converters::To<std::string>(argument))
396 (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↳ ((?!/*~\k<variable>~\*/)(.\|\\n))*?)\k<variable>\.Append\\((?<argument>[^\|\\n], \\r\\n
    ↳ +)\\)"),
    ↳ "${scope}${separator}${before}${variable}.append(Platform::Converters::To<std::s
    ↳ tring>(${argument}))",
    ↳ 10),
397 // Remove scope borders.
398 // /*~sb~*/
399 //
400 (new Regex(@"/*~[a-zA-Z0-9]+~\*/") , "", 0),
401 // Insert scope borders.
402 // auto added = new HashSet<TElement>();
403 // ~!added!~std::unordered_set<TElement> added;
404 (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
    ↳ HashSet<(?<element>[a-zA-Z0-9]+)>\\(\\)");
    ↳ "~!${variable}!~std::unordered_set<${element}> ${variable};", 0),
405 // Inside the scope of ~!added!~ replace:
406 // added.Add(node)
407 // added.insert(node)
408 (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.\|\\n)(?<before>((?<
    ↳ !~!\k<variable>!~)(.\|\\n))*?)\k<variable>\\.Add\\((?<argument>[a-zA-Z0-9]+)\\)"),
    ↳ "${scope}${separator}${before}${variable}.insert(${argument})", 10),
409 // Inside the scope of ~!added!~ replace:
410 // added.Remove(node)
411 // added.erase(node)
412 (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.\|\\n)(?<before>((?<
    ↳ !~!\k<variable>!~)(.\|\\n))*?)\k<variable>\\.Remove\\((?<argument>[a-zA-Z0-9]+)\\)"),
    ↳ "${scope}${separator}${before}${variable}.erase(${argument})", 10),
413 // if (added.insert(node)) {
414 // if (!added.contains(node)) { added.insert(node);
415 (new Regex(@"if \\((?<variable>[a-zA-Z0-9]+)\\.insert\\((?<argument>[a-zA-Z0-9]+)\\)\\)(?
    ↳ <separator>[\\t ]*[\\r\\n]+)(?<indent>[\\t ]*){") , "if
    ↳ (!${variable}.contains(${argument})) ${separator}${indent}{ " +
    ↳ Environment.NewLine + "${indent}    ${variable}.insert(${argument});", 0),
416 // Remove scope borders.
417 // ~!added!~
418 //
419 (new Regex(@"~![a-zA-Z0-9]+!~") , "", 5),
420 // Insert scope borders.
421 // auto random = new System.Random(0);
422 // std::srand(0);
423 (new Regex(@"[a-zA-Z0-9\\.]+ ([a-zA-Z0-9]+) = new
    ↳ (System\\.)?Random\\((([a-zA-Z0-9]+)\\)");
    ↳ "~!$!~std::srand($3);", 0),
424 // Inside the scope of ~!random!~ replace:
425 // random.Next(1, N)
426 // (std::rand() % N) + 1
427 (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.\|\\n)(?<before>((?<
    ↳ !~!\k<variable>!~)(.\|\\n))*?)\k<variable>\\.Next\\((?<from>[a-zA-Z0-9]+),
    ↳ (?<to>[a-zA-Z0-9]+)\\)"),
    ↳ "${scope}${separator}${before}(std::rand() % ${to}) +
    ↳ ${from}", 10),
428 // Remove scope borders.

```

```

429 // ~!random!~
430 //
431 (new Regex(@"~![a-zA-Z0-9]+!~"), "", 5),
432 // Insert method body scope starts.
433 // void PrintNodes(TElement node, StringBuilder sb, int level) {
434 // void PrintNodes(TElement node, StringBuilder sb, int level) { /*method-start*/
435 (new Regex(@"(?<start>\r?\n[\t ]+)(?<prefix>((private|protected|public):)?(virtual
→ )?[a-zA-Z0-9:_]+
→ )?(?<method>[a-zA-Z][a-zA-Z0-9]*)\(((?<arguments>[^\)]*)\)(?<override>(
→ override)?)(?<separator>[ \t\r\n]*)\{(?<end>[~])\}", "${start}${prefix}${method}"
→ (${arguments})${override}${separator}{ /*method-start*/ ${end} ",
→ 0),
436 // Insert method body scope ends.
437 // { /*method-start*/ ... }
438 // { /*method-start*/ ... /*method-end*/ }
439 (new Regex(@"{ /*method-start*/ (?<body> ((?<bracket>\{) | (?<-bracket>\}) | [^\{\}]* )+ )
→ \}", " { /*method-start*/ ${body} /*method-end*/ } ",
→ 0),
440 // Inside method bodies replace:
441 // GetFirst(
442 // this->GetFirst(
443 (new
→ Regex(@"(?<scope> /*method-start*/ ) (?<before> ((?! /*method-end*/ ) ( . | \n ) ) * ) ( ?
→ <separator> [ \W ] ( ? ! ( : | \. | - > | throw \s + ) ) ) ( ? <method> ( ? ! sizeof ) [ a - z A - Z 0 - 9 ] + ) \ ( ( ? ! \
→ \{ ) ( ? <after> ( . | \n ) * ) ( ? <scopeEnd> /*method-end*/ ) " ,
→ "${scope}${before}${separator}this->${method} (${after}${scopeEnd} ", 100),
444 // Remove scope borders.
445 // /*method-start*/
446 //
447 (new Regex(@" /*method-(start|end) */"), "", 0),
448 // Insert scope borders.
449 // const std::exception& ex
450 // const std::exception& ex/*~ex~*/
451 (new Regex(@"(?<before>\(| ) (?<variableDefinition>(const )?(std::)?exception&
→ (?<variable>[_a-zA-Z0-9]+)) (?<after>\W)",
→ "${before}${variableDefinition}/*~${variable}~/${after} ", 0),
452 // Inside the scope of ~!ex!~ replace:
453 // ex.Message
454 // ex.what()
455 (new Regex(@"(?<scope> /*~ (?<variable>[_a-zA-Z0-9]+) ~ */ ) (?<separator> . | \n ) ( ? <before
→ > ((?! /*~\k<variable>~ */ ) ( . | \n ) ) * ) ( Platform::Converters::To<std::string>\( \k<
→ variable> \. Message \) | \k<variable> \. what ( ) " ,
→ "${scope}${separator}${before}${variable}.what()", 10),
456 // Remove scope borders.
457 // /*~ex~*/
458 //
459 (new Regex(@" /*~ [_a-zA-Z0-9]+ ~ */"), "", 0),
460 // throw ArgumentNullException(argumentName, message);
461 // throw std::invalid_argument(std::string("Argument
→ ").append(argumentName).append(" is null: ").append(message).append("."));
462 (new Regex(@"throw
→ ArgumentNullException\(((?<argument>[a-zA-Z]*[Aa]rgument[a-zA-Z]*) ,
→ (?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\(\)?)\));", "throw
→ std::invalid_argument(std::string(\"Argument \").append("${argument}").append(\"
→ is null: \").append("${message}").append(\".\"));";", 0),
463 // throw ArgumentException(message, argumentName);
464 // throw std::invalid_argument(std::string("Invalid ").append(argumentName).append("
→ argument: ").append(message).append("."));
465 (new Regex(@"throw
→ ArgumentException\(((?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\(\)?) ,
→ (?<argument>[a-zA-Z]*[Aa]rgument[a-zA-Z]*)\));", "throw
→ std::invalid_argument(std::string(\"Invalid \").append("${argument}").append(\"
→ argument: \").append("${message}").append(\".\"));";", 0),
466 // throw ArgumentOutOfRangeException(argumentName, argumentValue, messageBuilder());
467 // throw std::invalid_argument(std::string("Value
→ [").append(Platform::Converters::To<std::string>(argumentValue)).append("] of
→ argument [").append(argumentName).append("] is out of range:
→ ").append(messageBuilder()).append("."));
468 (new Regex(@"throw ArgumentOutOfRangeException\(((?<argument>[a-zA-Z]*[Aa]rgument[a-z
→ A-Z]*([Nn]ame[a-zA-Z]*)?) ,
→ (?<argumentValue>[a-zA-Z]*[Aa]rgument[a-zA-Z]*([Vv]alue[a-zA-Z]*)?) ,
→ (?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\(\)?)\));", "throw
→ std::invalid_argument(std::string(\"Value
→ [").append(Platform::Converters::To<std::string>("${argumentValue}")).append(\"
→ of argument [").append("${argument}").append(\"] is out of range:
→ \").append("${message}").append(\".\"));";", 0),

```

```

469 // throw NotSupportedException();
470 // throw std::logic_error("Not supported exception.");
471 (new Regex(@"throw NotSupportedException\(\);", "throw std::logic_error(\"Not
    ↳ supported exception.\");", 0),
472 // throw NotImplementedException();
473 // throw std::logic_error("Not implemented exception.");
474 (new Regex(@"throw NotImplementedException\(\);", "throw std::logic_error(\"Not
    ↳ implemented exception.\");", 0),
475 // Insert scope borders.
476 // const std::string& message
477 // const std::string& message/*~message~*/
478 (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?string&?|char*)
    ↳ (?<variable>[_a-zA-Z0-9]+))(?<after>\W)",
    ↳ "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
479 // Inside the scope of /*~message~*/ replace:
480 // Platform::Converters::To<std::string>(message)
481 // message
482 (new Regex(@"(?<scope>\/~(?<variable>[_a-zA-Z0-9]+)~\/)(?<separator>.\|\\n)(?<before>
    ↳ >((?!\/~\k<variable>~\/)(.\|\\n))*?)Platform::Converters::To<std::string>\\(\\k<v
    ↳ ariable>\\)", "${scope}${separator}${before}${variable}",
    ↳ 10),
483 // Remove scope borders.
484 // /*~ex~*/
485 //
486 (new Regex(@"\/~[_a-zA-Z0-9]+~\/", "", 0),
487 // Insert scope borders.
488 // std::tuple<T, T> tuple
489 // std::tuple<T, T> tuple/*~tuple~*/
490 (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?tuple<[^\\n]+>&?
    ↳ (?<variable>[_a-zA-Z0-9]+))(?<after>\W)",
    ↳ "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
491 // Inside the scope of ~!ex!~ replace:
492 // tuple.Item1
493 // std::get<1-1>(tuple)
494 (new Regex(@"(?<scope>\/~(?<variable>[_a-zA-Z0-9]+)~\/)(?<separator>.\|\\n)(?<before>
    ↳ >((?!\/~\k<variable>~\/)(.\|\\n))*?)\\k<variable>\\.Item(?<itemNumber>\\d+)(?<afte
    ↳ r>\\W)",
    ↳ "${scope}${separator}${before}std::get<${itemNumber}-1>(${variable})${after}",
    ↳ 10),
495 // Remove scope borders.
496 // /*~ex~*/
497 //
498 (new Regex(@"\/~[_a-zA-Z0-9]+~\/", "", 0),
499 // Insert scope borders.
500 // class Range<T> {
501 // class Range<T> {/*~type~Range<T>~*/
502 (new Regex(@"(?<classDeclarationBegin>\\r?\\n(?<indent>[\\t ]*)template <typename
    ↳ (?<typeParameter>[^\\n]+> (struct|class)
    ↳ (?<type>[_a-zA-Z0-9]+<\\k<typeParameter>>)\\s*:\\s*[^{\\n]+)?[\\t ]*(\\r?\\n)?[\\t
    ↳ ]*{)"), "${classDeclarationBegin}/*~type~${type}~*/", 0),
503 // Inside the scope of /*~type~Range<T>~*/ insert inner scope and replace:
504 // public: static implicit operator std::tuple<T, T>(Range<T> range)
505 // public: operator std::tuple<T, T>() const {/*~variable~Range<T>~*/
506 (new Regex(@"(?<scope>\/~type~(?<type>[~\\n\\+])~\/)(?<separator>.\|\\n)(?<before>((
    ↳ ?!\/~type~\\k<type>~\/)(.\|\\n))*?) (?<access>(private|protected|public): )static
    ↳ implicit operator (?<targetType>[^\\(\\n]+)\\((?<argumentDeclaration>\\k<type>
    ↳ (?<variable>[_a-zA-Z0-9]+))\\)(?<after>\\s*\\n?\\s*{)"),
    ↳ "${scope}${separator}${before}${access}operator ${targetType}()
    ↳ const${after}/*~variable~${variable}~*/", 10),
507 // Inside the scope of /*~type~Range<T>~*/ replace:
508 // public: static implicit operator Range<T>(std::tuple<T, T> tuple) { return new
    ↳ Range<T>(std::get<1-1>(tuple), std::get<2-1>(tuple)); }
509 // public: Range(std::tuple<T, T> tuple) : Range(std::get<1-1>(tuple),
    ↳ std::get<2-1>(tuple)) { }
510 (new Regex(@"(?<scope>\/~type~(?<type>(\\k<typeName>[_a-zA-Z0-9]+)[~\\n\\+])~\/)(?<s
    ↳ eparator>.\|\\n)(?<before>((?!\/~type~\\k<type>~\/)(.\|\\n))*?) (?<access>(private|
    ↳ protected|public): )static implicit operator
    ↳ \\k<type>\\((?<arguments>[{}\\n]+)\\)\\(\\s|\\n)*\\(\\s|\\n)*return (new
    ↳ )?\\k<type>\\((?<passedArguments>[~\\n\\+])\\);\\(\\s|\\n)*"),
    ↳ "${scope}${separator}${before}${access}${typeName}(${arguments}) :
    ↳ ${typeName}(${passedArguments}) { }", 10),
511 // Inside the scope of /*~variable~range~*/ replace:
512 // range.Minimum
513 // this->Minimum

```

```

(new Regex(@"(?<scope>[/\*\~variable~(?<variable>[~\n]+)~\*/)(?<separator>[.|\n](?<be
fore>(?(beforeExpression>(?(bracket>{)|(?(bracket>})|[^{}]|\\n)*?)\k<variable>\.
(?(field>[_a-zA-Z0-9]+)(?(after>(,|;|)|
|)))(?(afterExpression>(?(bracket>{)|(?(bracket>})|[^{}]|\\n)*?))"),
"$${scope}${separator}${before}this->${field}${after}", 10),
// Remove scope borders.
// /*~ex~*/
//
(new Regex(@"[/\*\~[~\n]+~[~\n]+~\*/"), "", 0),
// Insert scope borders.
// namespace Platform::Ranges { ... }
// namespace Platform::Ranges {/~start~namespace~Platform::Ranges~/ ...
// /*~end~namespace~Platform::Ranges~/}
(new Regex(@"(?<namespaceDeclarationBegin>\r?\n(?<indent>[\t ]*)namespace
(?<namespaceName>(?(namePart>[a-zA-Z][a-zA-Z0-9]+)(?(nextNamePart>:[a-zA-Z][a-z
A-Z0-9]+)))(\s|\n)*{)(?(middle>(\.|\n)*)(?(end>(?(=>\r?\n)\k<indent>}{?!;}))"),
"${namespaceDeclarationBegin}/~start~namespace~${namespaceName}~/~${middle}~/~e
nd~namespace~${namespaceName}~/~${end}%",
0),
// Insert scope borders.
// class Range<T> { ... };
// class Range<T> {/~start~type~Range<T>~T~/ ... /*~start~type~Range<T>~T~/};
(new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)template <typename
(?<typeParameter>[~\n]+)> (struct|class)
(?<type>[a-zA-Z0-9]+<\k<typeParameter>>)(\s*:\s*[~\n]+)?[\t ]*(\r?\n)?[\t
]*{)(?(middle>(\.|\n)*)(?(endIndent>(?(=>\r?\n)\k<indent>)(?(end>};))"),
"${classDeclarationBegin}/~start~type~${type}~${typeParameter}~/~${middle}~/~e
nd~type~${type}~${typeParameter}~/~${end}%",
0),
// Inside scopes replace:
// /*~start~namespace~Platform::Ranges~/ ... /*~start~type~Range<T>~T~/ ...
public: override std::int32_t GetHashCode() { return {Minimum,
Maximum}.GetHashCode(); } ... /*~start~type~Range<T>~T~/ ...
// /*~end~namespace~Platform::Ranges~/}
// /*~start~namespace~Platform::Ranges~/ ... /*~start~type~Range<T>~T~/ ...
// /*~start~type~Range<T>~T~/ ... /*~end~namespace~Platform::Ranges~/ namespace
std { template <typename T> struct hash<Platform::Ranges::Range<T>> {
std::size_t operator()(const Platform::Ranges::Range<T> &obj) const { return
{Minimum, Maximum}.GetHashCode(); } }; }
(new Regex(@"(?<namespaceScopeStart>/~start~namespace~(?<namespace>[~\n\*]+)~\*/)
(?<betweenStartScopes>(\.|\n)+)(?<typeScopeStart>/~start~type~(?<type>[~\n\*]+)
)~(?<typeParameter>[~\n\*]+)~\*/)(?<before>(\.|\n)+)?(?<hashMethodDeclaration>\r
?\n[ \t]*(?(access>(private|protected|public): )override std::int32_t
GetHashCode\(\)(\s|\n)*\s*(?(methodBody>[~\s][~\n]+[~\s])\s*\s*)(?(after>(\.|\n
)+)?)(?<typeScopeEnd>/~end~type~\k<type>~\k<typeParameter>~\*/)(?<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 + "
            {" + 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>~T~/
//
(new Regex(@"[/\*\~[~\n\*]+~[~\n\*]+~\*/"), "", 0),
}.Cast<ISubstitutionRule>().ToList();

public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
{
    // ICounter<int, int> c1;
    // ICounter<int, int>* c1;

```

```

545 (new Regex(@"(?<abstractType>I[A-Z] [a-zA-Z0-9]+(<[^\r\n]+>)?)
    ↳ (?<variable>[_a-zA-Z0-9]+)(?<after> = null)?;"), "${abstractType}*
    ↳ ${variable}${after};", 0),
546 // (expression)
547 // expression
548 (new Regex(@"\(|\)|((([a-zA-Z0-9_]*:)+))\(|\)|;|\)|)"), "$1$2$3", 0),
549 // (method(expression))
550 // method(expression)
551 (new Regex(@"(?<firstSeparator>\(|\)|
    ↳ ))\((?<method>[a-zA-Z0-9_]*->[a-zA-Z0-9_]*:)+\)((?<expression>((?<parenthesis>\(|\)|(?<-parent
    ↳ hesis>)|[a-zA-Z0-9_]*->[a-zA-Z0-9_]*:)+)(?(parenthesis)(?!))\)|(?<lastSeparator>(|,|
    ↳ |;|\)|)"))), "${firstSeparator}${method}${expression}${lastSeparator}", 0),
552 // .append(".")
553 // .append(1, '.');
554 (new Regex(@"\append\""([^\\""]|\\[^\"])"\""), ".append(1, '$1')", 0),
555 // return ref _elements[node];
556 // return &_elements[node];
557 (new Regex(@"return ref ([_a-zA-Z0-9]+)\([([_a-zA-Z0-9_]*)+\)");), "return &$1[$2];",
    ↳ 0),
558 // ((1, 2))
559 // ({1, 2})
560 (new Regex(@"(?<before>\(|, )\((?<first>[^\n()]+),
    ↳ (?<second>[^\n()]+)\)(?<after>\)|, )"), "${before}${first},
    ↳ ${second}${after}", 10),
561 // {1, 2}.GetHashCode()
562 // Platform::Hashing::Hash(1, 2)
563 (new Regex(@"{(?<first>[^\n{]+), (?<second>[^\n{]+)}\}.GetHashCode\(\)"),
    ↳ "Platform::Hashing::Hash(${first}, ${second})", 10),
564 // range.ToString()
565 // Platform::Converters::To<std::string>(range).data()
566 (new Regex(@"(?<before>\W)(?<variable>[_a-zA-Z] [_a-zA-Z0-9]+)\.ToString\(\)"),
    ↳ "${before}Platform::Converters::To<std::string>(${variable}).data()", 10),
567 // new
568 //
569 (new Regex(@"(?<before>\r?\n[^\r\n]*("(\\"| [^\r\n])*)*" [^\r\n]*)(?<=\W)new\
    ↳ s+"), "${before}",
    ↳ 10),
570 // x == null
571 // x == nullptr
572 (new Regex(@"(?<before>\r?\n[^\r\n]*("(\\"| [^\r\n])*)*" [^\r\n]*)(?<=\W)(?<v
    ↳ ariable>[_a-zA-Z] [_a-zA-Z0-9]+)(?<operator>s*(==|!=)\s*)null(?<after>\W)",
    ↳ "${before}${variable}${operator}nullptr${after}", 10),
573 // null
574 // {}
575 (new Regex(@"(?<before>\r?\n[^\r\n]*("(\\"| [^\r\n])*)*" [^\r\n]*)(?<=\W)null\
    ↳ (?<after>\W)", "${before}-${after}",
    ↳ 10),
576 // default
577 // 0
578 (new Regex(@"(?<before>\r?\n[^\r\n]*("(\\"| [^\r\n])*)*" [^\r\n]*)(?<=\W)defa
    ↳ ult(?<after>\W)", "${before}0${after}",
    ↳ 10),
579 // object x
580 // void *x
581 (new Regex(@"(?<before>\r?\n[^\r\n]*("(\\"| [^\r\n])*)*" [^\r\n]*)(?<=\W)([O|
    ↳ o]bject|System\.Object) (?<after>\w)", "${before}void *${after}",
    ↳ 10),
582 // <object>
583 // <void*>
584 (new Regex(@"(?<before>\r?\n[^\r\n]*("(\\"| [^\r\n])*)*" [^\r\n]*)(?<=\W)(?!
    ↳ \w )([Oo]bject|System\.Object)(?<after>\W)", "${before}void*${after}",
    ↳ 10),
585 // ArgumentException
586 // std::invalid_argument
587 (new Regex(@"(?<before>\r?\n[^\r\n]*("(\\"| [^\r\n])*)*" [^\r\n]*)(?<=\W)(Sys
    ↳ tem\.)?ArgumentException(?<after>\W)",
    ↳ "${before}std::invalid_argument${after}", 10),
588 // InvalidOperationException
589 // std::runtime_error
590 (new Regex(@"(\W)(InvalidOperationException|Exception)(\W)"),
    ↳ "$1std::runtime_error$3", 0),
591 // ArgumentException
592 // std::invalid_argument
593 (new Regex(@"(\W)(ArgumentException|ArgumentOutOfRangeException)(\W)"),
    ↳ "$1std::invalid_argument$3", 0),
594 // template <typename T> struct Range : IEquatable<Range<T>>

```

```

595 // template <typename T> struct Range {
596 (new Regex(@"(?<before>template <typename (?<typeParameter>[^\n]+)> (struct|class)
→ (?<type>[a-zA-Z0-9]+<[^\n]+>)) : (public
→ )?IEquatable<k<type>>(?(after>(\s|\n)*{)", "${before}${after}", 0),
597 // #region Always
598 //
599 (new Regex(@"(^\r?\n)[ \t]*\#(region|endregion)[^\r\n]*(\r?\n|$)", "", 0),
600 // // #define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
601 //
602 (new Regex(@"\\\/[ \t]*\#define[ \t]+[_a-zA-Z0-9]+[ \t]*"), "", 0),
603 // #if USEARRAYPOOL\r\n#endif
604 //
605 (new Regex(@"#if [a-zA-Z0-9]+\s+#endif"), "", 0),
606 // [Fact]
607 //
608 (new Regex(@"(?<firstNewLine>\r?\n|\A)(?<indent>[ \t
→ ]+)\[[a-zA-Z0-9]+\(((?<expression>((?<parenthesis>\()|(?<-parenthesis>\))|~()\r
→ \n)*+)(?(parenthesis)(?!))\))?\][ \t]*(\r?\n\k<indent>?)"),
→ "${firstNewLine}${indent}", 5),
609 // \A \n ... namespace
610 // \Anamespace
611 (new Regex(@"(\A)(\r?\n)+namespace"), "$1namespace", 0),
612 // \A \n ... class
613 // \Aclass
614 (new Regex(@"(\A)(\r?\n)+class"), "$1class", 0),
615 // \n\n\n
616 // \n\n
617 (new Regex(@"\r?\n[ \t]*\r?\n[ \t]*\r?\n"), Environment.NewLine +
→ Environment.NewLine, 50),
618 // {\n\n
619 // {\n
620 (new Regex(@"[ \t]*\r?\n[ \t]*\r?\n"), "{" + Environment.NewLine, 10),
621 // \n\n}
622 // \n}
623 (new Regex(@"\r?\n[ \t]*\r?\n(?<end>[ \t]*}")), Environment.NewLine + "${end}", 10),
624 }.Cast<ISubstitutionRule>().ToList();
625
626 public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
→ base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
627
628 public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
629 }
630 }

```

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

```

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

```

```
33     };";
34
35     var transformer = new CSharpToCppTransformer();
36     var actualResult = transformer.Transform(helloWorldCode);
37     Assert.Equal(expectedResult, actualResult);
38 }
39 }
```



## Index

./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs, 14

./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1