

## 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]+\((([^\r\n]*\r\n)*\)) (?<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]+\((([^\r\n]*\r\n)*\))\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 // Action action
164 // std::function<void()> action
165 (new Regex(@"Action ([a-zA-Z0-9]+)"), "std::function<void()> $1", 0),
166 // Predicate<TArgument> predicate
167 // std::function<bool(TArgument)> predicate
168 (new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<bool($1)>
    ↳ $2", 0),
169 // var
170 // auto
171 (new Regex(@"(\\W)var(\\W)"), "$1auto$2", 0),
172 // unchecked
173 //
174 (new Regex(@"[\\r\\n]{2}\\s*?unchecked\\s*?$"), "", 0),
175 // throw new
176 // throw
177 (new Regex(@"(\\W)throw new(\\W)"), "$1throw$2", 0),
178 // void RaiseExceptionIgnoredEvent(Exception exception)
179 // void RaiseExceptionIgnoredEvent(const std::exception& exception)
180 (new Regex(@"\\(|, )(System\\.Exception|Exception)( |\\)\\)"), "$1const
    ↳ std::exception&$3", 0),
181 // EventHandler<Exception>
182 // EventHandler<std::exception>
183 (new Regex(@"(\\W)(System\\.Exception|Exception)(\\W)"), "$1std::exception$3", 0),
184 // override void PrintNode(TElement node, StringBuilder sb, int level)
185 // void PrintNode(TElement node, StringBuilder sb, int level) override
186 (new Regex(@"override ([a-zA-Z0-9 \\*+]*)\\(((\\^\\)\\r\\n)+?)\\)"), "$1$2 override", 0),
187 // return (range.Minimum, range.Maximum)
188 // return {range.Minimum, range.Maximum}
189 (new Regex(@"(?<before>return\\s*)\\(((?<values>[\\^\\)\\n]+)\\)(?!\\() (?<after>\\W)"),
    ↳ "${before}${values}}${after}", 0),
190 // string
191 // std::string
192 (new Regex(@"(?<before>\\W)(?!::)string(?<after>\\W)"),
    ↳ "${before}std::string${after}", 0),
193 // System.ValueTuple
194 // std::tuple

```

```

195 (new Regex(@"(?<before>\W) (System\.)?ValueTuple(?:\s*=\|() (?<after>\W)"),
196     ↳ "${before}std::tuple${after}", 0),
197 // sbyte
198 // std::int8_t
199 (new Regex(@"(?<before>\W) ((System\.)?SB|sb)yte(?:\s*=\|() (?<after>\W)"),
200     ↳ "${before}std::int8_t${after}", 0),
201 // short
202 // std::int16_t
203 (new Regex(@"(?<before>\W) ((System\.)?Int16|short) (?:\s*=\|() (?<after>\W)"),
204     ↳ "${before}std::int16_t${after}", 0),
205 // int
206 // std::int32_t
207 (new Regex(@"(?<before>\W) ((System\.)?I|i)nt(32)?(?:\s*=\|() (?<after>\W)"),
208     ↳ "${before}std::int32_t${after}", 0),
209 // long
210 // std::int64_t
211 (new Regex(@"(?<before>\W) ((System\.)?Int64|long) (?:\s*=\|() (?<after>\W)"),
212     ↳ "${before}std::int64_t${after}", 0),
213 // byte
214 // std::uint8_t
215 (new Regex(@"(?<before>\W) ((System\.)?Byte|byte) (?:\s*=\|() (?<after>\W)"),
216     ↳ "${before}std::uint8_t${after}", 0),
217 // ushort
218 // std::uint16_t
219 (new Regex(@"(?<before>\W) ((System\.)?UInt16|ushort) (?:\s*=\|() (?<after>\W)"),
220     ↳ "${before}std::uint16_t${after}", 0),
221 // uint
222 // std::uint32_t
223 (new Regex(@"(?<before>\W) ((System\.)?UI|ui)nt(32)?(?:\s*=\|() (?<after>\W)"),
224     ↳ "${before}std::uint32_t${after}", 0),
225 // ulong
226 // std::uint64_t
227 (new Regex(@"(?<before>\W) ((System\.)?UInt64|ulong) (?:\s*=\|() (?<after>\W)"),
228     ↳ "${before}std::uint64_t${after}", 0),
229 // char*[] args
230 // char* args[]
231 (new Regex(@"([_a-zA-Z0-9:~*?])\[\] ([_a-zA-Z0-9]+)", "$1 $2[]", 0),
232 // float.MinValue
233 // std::numeric_limits<float>::lowest()
234 (new Regex(@"(?<before>\W) (?<type>std::[_a-z0-9_]+|float|double)\.MinValue(?<after>\W)",
235     ↳ "${before}std::numeric_limits<${type}>::lowest()${after}",
236     ↳ 0),
237 // double.MaxValue
238 // std::numeric_limits<float>::max()
239 (new Regex(@"(?<before>\W) (?<type>std::[_a-z0-9_]+|float|double)\.MaxValue(?<after>\W)",
240     ↳ "${before}std::numeric_limits<${type}>::max()${after}",
241     ↳ 0),
242 // using Platform.Numbers;
243 //
244 (new Regex(@"([\r\n]{2}|~)\s*using [_a-zA-Z0-9]+;\s*?${")", "", 0),
245 // struct TreeElement { }
246 // struct TreeElement { };
247 (new Regex(@"(struct|class) ([_a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([\^;])", "$1
248     ↳ $2$3{$4};$5", 0),
249 // class Program { }
250 // class Program { };
251 (new Regex(@"(?<type>struct|class)
252     ↳ (?<name>[_a-zA-Z0-9]+[~\r\n]*) (?<beforeBody>[\r\n]+(?<indentLevel>[\t
253     ↳ ]*)?)\{(?<body>[\S\s]+?[~\r\n]+\k<indentLevel>)\}(?<afterBody>[~;]|$)", "${type}
254     ↳ ${name}${beforeBody}${body}};${afterBody}", 0),
255 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
256 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
257 (new Regex(@"(struct|class) ([_a-zA-Z0-9]+)(<[_a-zA-Z0-9_,]+>)? : ([_a-zA-Z0-9]+)",
258     ↳ "$1 $2$3 : public $4", 0),
259 // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
260 // class IProperty : public ISetter<TValue, TObject>, public IProvider<TValue,
261     ↳ TObject>
262 (new Regex(@"(?<before>(struct|class) [_a-zA-Z0-9]+ : ((public
263     ↳ [_a-zA-Z0-9]+(<[_a-zA-Z0-9_,]+>)?,
264     ↳ )+)?(?<inheritedType>(?!public) [_a-zA-Z0-9]+(<[_a-zA-Z0-9_,]+>)?(?<after>(,
265     ↳ [_a-zA-Z0-9]+(?!>)|[\r\n]+)))", "${before}public ${inheritedType}${after}", 10),
266 // Insert scope borders.
267 // ref TElement root
268 // ~!root!~ref TElement root
269 (new Regex(@"(?<definition>(?!<= \|() (ref [_a-zA-Z0-9]+|[_a-zA-Z0-9]+(?<ref>))
270     ↳ (?<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[$4] = { {0} };", 0),
313 // bool Equals(Range<T> other) { ... }
314 // bool operator ==(const Key &other) const { ... }
315 (new Regex(@"(?<before>\\r?\\n[\\t ]+bool )Equals\\((?<type>[\\t ]+
→ (?<variable>[_a-zA-Z0-9]+)\\)(?<after>(\\s|\\n)*\\)", "${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 // public: event Disposal OnDispose;
338 // public: Platform::Delegates::MulticastDelegate<Disposal> OnDispose;
339 (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),
340 // Insert scope borders.

```



```

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

```



```

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

```

```

430 (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}) +
    ↳ ${from}", 10),
431 // Remove scope borders.
432 // ~!random!~
433 //
434 (new Regex(@"~![a-zA-Z0-9]+!~"), "", 5),
435 // Insert method body scope starts.
436 // void PrintNodes(TElement node, StringBuilder sb, int level) {
437 // void PrintNodes(TElement node, StringBuilder sb, int level) { /*method-start*/
438 (new Regex(@"(?<start>\r?\n[\\t ]+)(?<prefix>((private|protected|public): )?(virtual
    ↳ )?[a-zA-Z0-9:_]+
    ↳ )?(?<method>[a-zA-Z][a-zA-Z0-9]*)\\((?<arguments>[^\|\\]*)\\)(?<override>(
    ↳ override)?)(?<separator>[\\t\\r\\n]*)\\{(?<end>[~])")", "${start}${prefix}${method}
    ↳ (${arguments})${override}${separator}{ /*method-start*/${end}",
    ↳ 0),
439 // Insert method body scope ends.
440 // { /*method-start*/...}
441 // { /*method-start*/... /*method-end*/}
442 (new Regex(@"{ /\*method-start\*/ (?<body>((?<bracket>\\{) | (?<-bracket>\\}) | [^\{\\}]+) )
    ↳ \\}"), "{ /*method-start*/${body} /*method-end*/}",
    ↳ 0),
443 // Inside method bodies replace:
444 // GetFirst(
445 // this->GetFirst(
446 (new
    ↳ Regex(@"(?<scope> /\*method-start\*/)(?<before>((?! /\*method-end\*/)(.\|\\n))*)?(?
    ↳ <separator>[\\W](?! (:|\\.|->|throw\\s+)))(?<method>(?!sizeof)[a-zA-Z0-9]+)\\((?!\\
    ↳ \\{)(?<after>(.\|\\n)*)?(?<scopeEnd> /\*method-end\*/)",
    ↳ "${scope}${before}${separator}this->${method}(${after}${scopeEnd}", 100),
447 // Remove scope borders.
448 // /*method-start*/
449 //
450 (new Regex(@" /\*method-(start|end)\*/"), "", 0),
451 // Insert scope borders.
452 // const std::exception& ex
453 // const std::exception& ex/*~ex~*/
454 (new Regex(@"(?<before>\\(| )(?<variableDefinition>(const )?(std::)?exception&?
    ↳ (?<variable>[_a-zA-Z0-9]+))(?<after>\\W)",
    ↳ "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
455 // Inside the scope of ~!ex!~ replace:
456 // ex.Message
457 // ex.what()
458 (new Regex(@"(?<scope> /\*~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↳ >((?! /\*~\k<variable>~\*/)(.\|\\n)*)?(Platform::Converters::To<std::string>\\(\k<
    ↳ variable>\\.Message\\) | \k<variable>\\.Message)",
    ↳ "${scope}${separator}${before}${variable}.what()", 10),
459 // Remove scope borders.
460 // /*~ex~*/
461 //
462 (new Regex(@" /\*~[_a-zA-Z0-9]+~\*/"), "", 0),
463 // throw ArgumentNullException(argumentName, message);
464 // throw std::invalid_argument(std::string("Argument
    ↳ ").append(argumentName).append(" is null: ").append(message).append("."));
465 (new Regex(@"throw
    ↳ ArgumentNullException\\((?<argument>[a-zA-Z]*[Aa]rgument[a-zA-Z]*),
    ↳ (?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*\\(\\|\\)?\\)");", "throw
    ↳ std::invalid_argument(std::string(\"Argument \").append(${argument}).append(\"
    ↳ is null: \").append(${message}).append(\".\"));";", 0),
466 // throw ArgumentException(message, argumentName);
467 // throw std::invalid_argument(std::string("Invalid ").append(argumentName).append("
    ↳ argument: ").append(message).append("."));
468 (new Regex(@"throw
    ↳ ArgumentException\\((?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*\\(\\|\\)?),
    ↳ (?<argument>[a-zA-Z]*[Aa]rgument[a-zA-Z]*\\)");", "throw
    ↳ std::invalid_argument(std::string(\"Invalid \").append(${argument}).append(\"
    ↳ argument: \").append(${message}).append(\".\"));";", 0),
469 // throw ArgumentOutOfRangeException(argumentName, argumentValue, messageBuilder());
470 // 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("."));

```

```

471 (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),
472 // throw NotSupportedException();
473 // throw std::logic_error("Not supported exception.");
474 (new Regex(@"throw NotSupportedException\(\);"), "throw std::logic_error(\"Not
    → supported exception.\");", 0),
475 // throw NotImplementedException();
476 // throw std::logic_error("Not implemented exception.");
477 (new Regex(@"throw NotImplementedException\(\);"), "throw std::logic_error(\"Not
    → implemented exception.\");", 0),
478 // Insert scope borders.
479 // const std::string& message
480 // const std::string& message/*~message~/
481 (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?string&?|char\*)
    → (?<variable>[_a-zA-Z0-9]+)(?<after>\W)",
    → "${before}${variableDefinition}/*~${variable}~/${after}", 0),
482 // Inside the scope of /*~message~/ replace:
483 // Platform::Converters::To<std::string>(message)
484 // message
485 (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),
486 // Remove scope borders.
487 // /*~ex~/
488 //
489 (new Regex(@"\/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
490 // Insert scope borders.
491 // std::tuple<T, T> tuple
492 // std::tuple<T, T> tuple/*~tuple~/
493 (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?tuple<[^\n]+>&?
    → (?<variable>[_a-zA-Z0-9]+)(?<after>\W)",
    → "${before}${variableDefinition}/*~${variable}~/${after}", 0),
494 // Inside the scope of ~!ex!~ replace:
495 // tuple.Item1
496 // std::get<1-1>(tuple)
497 (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),
498 // Remove scope borders.
499 // /*~ex~/
500 //
501 (new Regex(@"\/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
502 // Insert scope borders.
503 // class Range<T> {
504 // class Range<T> {/*~type~Range<T>~/
505 (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),
506 // Inside the scope of /*~type~Range<T>~/ insert inner scope and replace:
507 // public: static implicit operator std::tuple<T, T>(Range<T> range)
508 // public: operator std::tuple<T, T>() const {/*~variable~Range<T>~/
509 (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),
510 // Inside the scope of /*~type~Range<T>~/ replace:
511 // 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)); }
512 // public: Range(std::tuple<T, T> tuple) : Range(std::get<1-1>(tuple),
    → std::get<2-1>(tuple)) { }

```

```

513 (new Regex(@"(?<scope>/\s*~type~(?<type>(?(typeName>[_a-zA-Z0-9]+)[^\n\s]*~\s*/)(?<se
    eparator>.\s|\n)(?<before>((?!/\s*~type~\k<type>~\s*/)(.\s|\n))*?(?<access>(private|
    ↪ protected|public): )static implicit operator
    ↪ \k<type>\k<arguments>[^\s\n]+\s)\s|\n)*{(\s|\n)*return (new
    ↪ )? \k<type>\k<passedArguments>[^\s\n]+\s)\s|\n)*"}),
    ↪ "${scope}${separator}${before}${access}${typeName}(${arguments}) :
    ↪ ${typeName}(${passedArguments}) { }", 10),
514 // Inside the scope of /*~variable~range~*/ replace:
515 // range.Minimum
516 // this->Minimum
517 (new Regex(@"(?<scope>{/\s*~variable~(?<variable>[^\n\s]+)~\s*/)(?<separator>.\s|\n)(?<be
    fore>(?(beforeExpression>(?(bracket>{)|(?(<-bracket>})|[\s\n])*?)\k<variable>\.
    ↪ (?(field>[_a-zA-Z0-9]+)(?<after>(,|;|}|
    ↪ |)))(?<afterExpression>(?(bracket>{)|(?(<-bracket>})|[\s\n])*?)") ,
    ↪ "${scope}${separator}${before}this->${field}${after}", 10),
518 // Remove scope borders.
519 // /*~ex~*/
520 //
521 (new Regex(@"/*~[^\n\s]+~[^\n\s]+~\s*/"), "", 0),
522 // Insert scope borders.
523 // namespace Platform::Ranges { ... }
524 // namespace Platform::Ranges {/\s*~start~namespace~Platform::Ranges~*/ ...
    ↪ /\s*~end~namespace~Platform::Ranges~*/}
525 (new Regex(@"(?<namespaceDeclarationBegin>\r?\n(?<indent>[\t ]*)namespace
    ↪ (?<namespaceName>(?(namePart>[_a-zA-Z][_a-zA-Z0-9]+)(?(nextNamePart>: [_a-zA-Z][_a-z
    ↪ A-Z0-9]+)+)\s|\n)*)(?(middle>(\s|\n)*)(?(end>(?(<=\r?\n)\k<indent>)(?!;)))"),
    ↪ "${namespaceDeclarationBegin}/*~start~namespace~${namespaceName}~*/${middle}~*e
    ↪ nd~namespace~${namespaceName}~*/${end}]",
    ↪ 0),
526 // Insert scope borders.
527 // class Range<T> { ... };
528 // class Range<T> {/\s*~start~type~Range<T>~T~*/ ... /\s*~start~type~Range<T>~T~*/};
529 (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)template <typename
    ↪ (?<typeParameter>[^\s\n]+)> (struct|class)
    ↪ (?(type>[_a-zA-Z0-9]+<\k<typeParameter>>)\s*:\s*[^\s\n]+)?[\t ]*(\r?\n)?[\t
    ↪ ]*(?(middle>(\s|\n)*)(?(endIndent>(?(<=\r?\n)\k<indent>)(?(end>);)))"),
    ↪ "${classDeclarationBegin}/*~start~type~${type}~${typeParameter}~*/${middle}~${end
    ↪ Indent}/*~end~type~${type}~${typeParameter}~*/${end}]",
    ↪ 0),
530 // Inside scopes replace:
531 // /*~start~namespace~Platform::Ranges~*/ ... /\s*~start~type~Range<T>~T~*/ ...
    ↪ public: override std::int32_t GetHashCode() { return {Minimum,
    ↪ Maximum}.GetHashCode(); } ... /\s*~start~type~Range<T>~T~*/ ...
    ↪ /\s*~end~namespace~Platform::Ranges~*/
532 // /\s*~start~namespace~Platform::Ranges~*/ ... /\s*~start~type~Range<T>~T~*/ ...
    ↪ /\s*~start~type~Range<T>~T~*/ ... /\s*~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(); } }; }
533 (new Regex(@"(?<namespaceScopeStart>/\s*~start~namespace~(?<namespace>[^\n\s]*~\s*/)
    ↪ (?<betweenStartScopes>(\s|\n)+)(?<typeScopeStart>/\s*~start~type~(?<type>[^\n\s]*~\s
    ↪ )~(?<typeParameter>[^\n\s]*~\s*/)(?<before>(\s|\n)+)?(?<hashMethodDeclaration>\r
    ↪ ?\n[\t ]*(?<access>(private|protected|public): )override std::int32_t
    ↪ GetHashCode\(\s\)(\s|\n)*{(\s*(?(methodBody>[^\s\n]+[^\s\n])\s*)\s*(?(after>(\s|\n
    ↪ )+)?)(?<typeScopeEnd>/\s*~end~type~\k<type>~\k<typeParameter>~\s*/)(?<betweenEndSco
    ↪ pes>(\s|\n)+)(?<namespaceScopeEnd>/\s*~end~namespace~\k<namespace>~\s*/)}\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 + "
    ↪ /\s*~start~method~*/${methodBody}/*~end~method~*/" + Environment.NewLine + "
    ↪ }" + Environment.NewLine + "        };" + Environment.NewLine + "    }" +
    ↪ Environment.NewLine, 10),
534 // Inside scope of /*~start~method~*/ replace:
535 // /*~start~method~*/ ... Minimum ... /\s*~end~method~*/
536 // /*~start~method~*/ ... obj.Minimum ... /\s*~end~method~*/
537 (new Regex(@"(?<methodScopeStart>/\s*~start~method~\s*/)(?<before>.({|,
    ↪ ))(?(name>[_a-zA-Z][_a-zA-Z0-9]+)(?(after>[^\n\s]\.([a-zA-Z0-9](?!/\s*~end~method~\s*/
    ↪ )|[\s\n])+)(?<methodScopeEnd>/\s*~end~method~\s*/)"),
    ↪ "${methodScopeStart}${before}obj.${name}${after}${methodScopeEnd}", 10),
538 // Remove scope borders.

```

```

539 // /*~start~type~Range<T>~*/
540 //
541 (new Regex(@"/*~[~\*\n]+(~[~\*\n]+)*~\*/"), "", 0),
542 }.Cast<ISubstitutionRule>().ToList();
543
544 public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
545 {
546     // ICounter<int, int> c1;
547     // ICounter<int, int>* c1;
548     (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^\r\n]+>)?)
549     ↪ (?<variable>[_a-zA-Z0-9]+)(?<after> = null)?;"), "${abstractType}*
550     ↪ ${variable}${after};", 0),
551     // (expression)
552     // expression
553     (new Regex(@"(\(|\)|\((([a-zA-Z0-9_\*:]+)\)|(|;|\\))", "$1$2$3", 0),
554     // (method(expression))
555     // method(expression)
556     (new Regex(@"(?<firstSeparator>(\(|
557     ↪ ))\(((?<method>[a-zA-Z0-9_\*:]+)\(((?<expression>((?<parenthesis>\(|(?<-parent
558     ↪ hesis>\\)|[a-zA-Z0-9_\*:]+)(?(parenthesis)(?!))\\)\)(?<lastSeparator>(,|
559     ↪ |;|\\)))", "${firstSeparator}${method}(${expression})${lastSeparator}", 0),
560     // .append(".")
561     // .append(1, '.');
562     (new Regex(@"\.(append\\("([^\\""]|\\[^\\""])"\\")", ".append(1, '$1')", 0),
563     // return ref _elements[node];
564     // return &elements[node];
565     (new Regex(@"return ref ([_a-zA-Z0-9]+)\\([([_a-zA-Z0-9_\*:]+)\\];)", "return &$1[$2];",
566     ↪ 0),
567     // ((1, 2))
568     // ({1, 2})
569     (new Regex(@"(?<before>\\(|, )\\(((?<first>[^\n()]+),
570     ↪ (?<second>[^\n()]+)\\)(?<after>\\(|, ))", "${before}${{first},
571     ↪ ${second}}${after}", 10),
572     // {1, 2}.GetHashCode()
573     // Platform::Hashing::Hash(1, 2)
574     (new Regex(@"{(?<first>[^\n{}]+), (?<second>[^\n{}]+)}\\.GetHashCode\\(\\")",
575     ↪ "Platform::Hashing::Hash(${first}, ${second})", 10),
576     // range.ToString()
577     // Platform::Converters::To<std::string>(range).data()
578     (new Regex(@"(?<before>\\W)(?<variable>[_a-zA-Z][_a-zA-Z0-9]+)\\.ToString\\(\\")",
579     ↪ "${before}Platform::Converters::To<std::string>(${variable}).data()", 10),
580     // new
581     //
582     (new Regex(@"(?<before>\\r?\\n[^\r\n]*(\\"\\|\\[^\r\n])*"\\[^\r\n]*)(?<=\\W)new\\_
583     ↪ s+)", "${before}",
584     ↪ 10),
585     // x == null
586     // x == nullptr
587     (new Regex(@"(?<before>\\r?\\n[^\r\n]*(\\"\\|\\[^\r\n])*"\\[^\r\n]*)(?<=\\W)(?<v
588     ↪ ariable>[_a-zA-Z][_a-zA-Z0-9]+)(?<operator>\\s*(==|!=)\\s*)null(?<after>\\W)",
589     ↪ "${before}${variable}${operator}nullptr${after}", 10),
590     // null
591     // {}
592     (new Regex(@"(?<before>\\r?\\n[^\r\n]*(\\"\\|\\[^\r\n])*"\\[^\r\n]*)(?<=\\W)null_
593     ↪ (?<after>\\W)", "${before}{}${after}",
594     ↪ 10),
595     // default
596     // 0
597     (new Regex(@"(?<before>\\r?\\n[^\r\n]*(\\"\\|\\[^\r\n])*"\\[^\r\n]*)(?<=\\W)defa_
598     ↪ ult(?<after>\\W)", "${before}0${after}",
599     ↪ 10),
600     // object x
601     // void *x
602     (new Regex(@"(?<before>\\r?\\n[^\r\n]*(\\"\\|\\[^\r\n])*"\\[^\r\n]*)(?<=\\W)(?!_
603     ↪ @)(object|System\\.Object) (?<after>\\W)", "${before}void *${after}",
604     ↪ 10),
605     // <object>
606     // <void*>
607     (new Regex(@"(?<before>\\r?\\n[^\r\n]*(\\"\\|\\[^\r\n])*"\\[^\r\n]*)(?<=\\W)(?!_
608     ↪ @)(object|System\\.Object) (?<after>\\W)", "${before}void*${after}",
609     ↪ 10),
610     // @object
611     // object
612     (new Regex(@"@([_a-zA-Z0-9]+)", "$1", 0),
613     // ArgumentNullException
614     // std::invalid_argument

```

```

593 (new Regex(@"(?<before>\r?\n[~""\r\n]*(""\\"""|["""\r\n])*""["""\r\n]*)(?<=\\W)(Sys_
    ↳ tem\\.)?ArgumentNullException(?<after>\\W)"),
    ↳ "${before}std::invalid_argument${after}", 10),
594 // InvalidOperationException
595 // std::runtime_error
596 (new Regex(@"\\W)(InvalidOperationException|Exception)\\W)"),
    ↳ "$1std::runtime_error$3", 0),
597 // ArgumentException
598 // std::invalid_argument
599 (new Regex(@"\\W)(ArgumentException|ArgumentOutOfRangeException)\\W)"),
    ↳ "$1std::invalid_argument$3", 0),
600 // template <typename T> struct Range : IEquatable<Range<T>>
601 // template <typename T> struct Range {
602 (new Regex(@"(?<before>template <typename (?<typeParameter>[~\n]+> (struct|class)
    ↳ (?<type>[a-zA-Z0-9]+[~\n]+>)) : (public
    ↳ )?IEquatable<\k<type>>(?(after>\\s|\\n)*{)"), "${before}${after}", 0),
603 // public: delegate void Disposal(bool manual, bool wasDisposed);
604 // public: delegate void Disposal(bool, bool);
605 (new Regex(@"(?<before>(?(access>(private|protected|public): )delegate
    ↳ (?<returnType>[a-zA-Z][a-zA-Z0-9:]+)
    ↳ (?<delegate>[a-zA-Z][a-zA-Z0-9:]+)\\(((?<leftArgumentType>[a-zA-Z][a-zA-Z0-9:]+),
    ↳ *)?(?<argumentType>[a-zA-Z][a-zA-Z0-9:]+)
    ↳ (?<argumentName>[a-zA-Z][a-zA-Z0-9:]+)(?(after>(,
    ↳ (?<rightArgumentType>[a-zA-Z][a-zA-Z0-9:]+)
    ↳ (?<rightArgumentName>[a-zA-Z][a-zA-Z0-9:]+))*\\);)"),
    ↳ "${before}${argumentType}${after}", 20),
606 // public: delegate void Disposal(bool, bool);
607 // using Disposal = void(bool, bool);
608 (new Regex(@"(?<access>(private|protected|public): )delegate
    ↳ (?<returnType>[a-zA-Z][a-zA-Z0-9:]+)
    ↳ (?<delegate>[a-zA-Z][a-zA-Z0-9:]+)\\(((?<argumentTypes>[~\\(\\)\n]*\\);)", "using
    ↳ ${delegate} = ${returnType}${argumentTypes};", 20),
609 // #region Always
610 //
611 (new Regex(@"(^[~\r?\n][ \t]*#(region|endregion)[~\r\n]*(\r?\n|$)"), "", 0),
612 // // #define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
613 //
614 (new Regex(@"\\/[ \t]*#define[ \t]+[_a-zA-Z0-9]+[ \t]*"), "", 0),
615 // #if USEARRAYPOOL\r\n#endif
616 //
617 (new Regex(@"#if [a-zA-Z0-9]+\s+#endif"), "", 0),
618 // [Fact]
619 //
620 (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),
621 // \A \n ... namespace
622 // \Anamespace
623 (new Regex(@"(\\A)(\\r?\n)+namespace"), "$1namespace", 0),
624 // \A \n ... class
625 // \Aclass
626 (new Regex(@"(\\A)(\\r?\n)+class"), "$1class", 0),
627 // \n\n\n
628 // \n\n
629 (new Regex(@"\r?\n[ \t]*\r?\n[ \t]*\r?\n"), Environment.NewLine +
    ↳ Environment.NewLine, 50),
630 // {\n\n
631 // {\n
632 (new Regex(@"{[ \t]*\r?\n[ \t]*\r?\n"), "{" + Environment.NewLine, 10),
633 // \n\n}
634 // \n}
635 (new Regex(@"\r?\n[ \t]*\r?\n(?<end>[ \t]*)"), Environment.NewLine + "${end}", 10),
636 }.Cast<ISubstitutionRule>().ToList();
637
638 public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
    ↳ base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
639
640 public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
641 }
642 }

```

```

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
11             ↪ syntax
12             var transformer = new CSharpToCppTransformer();
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             const string expectedResult = @"class Program
29 {
30     public: static void Main(std::string args[])
31     {
32         printf("Hello, world!\n");
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