

## 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 : Transformer
11     {
12         public static readonly IList<ISubstitutionRule> FirstStage = new List<SubstitutionRule>
13         {
14             // // ...
15             //
16             (new Regex(@"(\r?\n)?[ \t]+//+.+"), "", null, 0),
17             // #pragma warning disable CS1591 // Missing XML comment for publicly visible type
18             // or member
19             //
20             (new Regex(@"^-s*?#pragma\[sa-zA-Z0-9]+\$"), "", null, 0),
21             // {\n\n\n
22             // {
23             (new Regex(@"{\s+[\r\n]+") , "{" + Environment.NewLine, null, 0),
24             // Platform.Collections.Methods.Lists
25             // Platform::Collections::Methods::Lists
26             (new Regex(@"(namespace[^\r\n]+?)\.([^\r\n]+?)") , "$1::$2", null, 20),
27             // out TProduct
28             // TProduct
29             (new Regex(@"(<?before>(<|, ))(in|out)
30             → (<?typeParameter>[a-zA-Z0-9]+)(<?after>(>|,))") ,
31             → "${before}${typeParameter}${after}", null, 10),
32             // public ...
33             // public: ...
34             (new Regex(@"(<?newlineAndIndent>\r?\n?[
35             → \t]*)(<?before>[^\{\\(\r\n)*](<?access>private|protected|public)[
36             → \t]+(?![^\{\\(\r\n)*](<interface>|<class>|<struct>)[^\{\\(\r\n)*]([^\{\\(\r\n)*)") ,
37             → "${newlineAndIndent}${access}: ${before}", null, 0),
38             // public: static bool CollectExceptions { get; set; }
39             // public: inline static bool CollectExceptions;
40             (new Regex(@"(<?access>(private|protected|public): )(<?before>(static )?[^\r\n]+
41             → )(<?name>[a-zA-Z0-9]+) {[^;]}*(?<=\\W)get;[^;]}*(?<=\\W)set;[^;]}*") ,
42             → "${access}inline ${before}${name};", null, 0),
43             // public abstract class
44             // class
45             (new Regex(@"((public|protected|private|internal|abstract|static)
46             → )*(?<category>interface|class|struct)", "${category}", null, 0),
47             // class GenericCollectionMethodsBase<TElement> {
48             // template <typename TElement> class GenericCollectionMethodsBase {
49             (new Regex(@"class ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>([^\{]+){", "template <typename $2>
50             → class $1$3{", null, 0),
51             // static void
52             → TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
53             → tree, TElement* root)
54             // template<typename T> static void
55             → TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
56             → tree, TElement* root)
57             (new Regex(@"static ([a-zA-Z0-9]+) ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>\\(((^\\)\\r\\n)+)\\)",
58             → "template <typename $3> static $1 $2($4)", null, 0),
59             // interface IFactory<out TProduct> {
60             // template <typename TProduct> class IFactory { public:
61             (new Regex(@"interface (<?interface>[a-zA-Z0-9]+)<(<?typeParameters>[a-zA-Z0-9
62             → ,]+)>(<?whitespace>[^\{]+){", "template <typename...> class ${interface};
63             → template <typename ${typeParameters}> class
64             → ${interface}<${typeParameters}>${whitespace}{", + Environment.NewLine + "
65             → public:", null, 0),
66             // template <typename TObject, TProperty, TValue>
67             // template <typename TObject, typename TProperty, TValue>
68             (new Regex(@"(<?before>template <((, )?typename [a-zA-Z0-9]+)+,
69             → )(<?typeParameter>[a-zA-Z0-9]+)(<?after>(,|>))") , "${before}typename
70             → ${typeParameter}${after}", null, 10),
71             // Insert markers
72             // private: static void BuildExceptionString(this StringBuilder sb, Exception
73             → exception, int level)
74             // /*~extensionMethod~BuildExceptionString~*/private: static void
75             → BuildExceptionString(this StringBuilder sb, Exception exception, int level)

```

```

53 (new Regex(@"private: static [^\r\n]+ (?<name>[a-zA-Z0-9]+)\(this [^\]\r\n]+\)"),
54     ↳ "/*~extensionMethod~${name}~*/$0", null, 0),
55 // Move all markers to the beginning of the file.
56 (new Regex(@"\A(?<before>[^\r\n]+\r?\n(.|\n)+)(?<marker>/\*~extensionMethod~(?<name>[a-zA-Z0-9]+)~\*/)", "${marker}${before}", null,
57     ↳ 10),
58 // /*~extensionMethod~BuildExceptionString~*/...sb.BuildExceptionString(exception.InnerException, level +
59     ↳ 1);
60 // /*~extensionMethod~BuildExceptionString~*/...BuildExceptionString(sb,
61     ↳ exception.InnerException, level + 1);
62 (new Regex(@"(?<before>/\*~extensionMethod~(?<name>[a-zA-Z0-9]+)~\*/(.|\n)+\W)(?<variable>[_a-zA-Z0-9]+\)\. \k<name>\(", "${before}${name}(${variable}, ", null,
63     ↳ 50),
64 // Remove markers
65 // /*~extensionMethod~BuildExceptionString~*/
66 //
67 (new Regex(@"/*~extensionMethod~[a-zA-Z0-9]+~\*/", "", null, 0),
68 // (this
69 // (
70 (new Regex(@"(this ", "(", null, 0),
71 // public: static readonly EnsureAlwaysExtensionRoot Always = new
72     ↳ EnsureAlwaysExtensionRoot();
73 // public: inline static EnsureAlwaysExtensionRoot Always;
74 (new Regex(@"(?<access>(private|protected|public): )?static readonly
75     ↳ (?<type>[a-zA-Z0-9]+) (?<name>[_a-zA-Z0-9_]+) = new \k<type>\(\);"),
76     ↳ "${access}inline static ${type} ${name};", null, 0),
77 // public: static readonly string ExceptionContentsSeparator = "---";
78 // public: inline static const char* ExceptionContentsSeparator = "---";
79 (new Regex(@"(?<access>(private|protected|public): )?static readonly string
80     ↳ (?<name>[a-zA-Z0-9_]+) = ""(?<string>(\\"|"[^\r\n"])+)"";"), "${access}inline
81     ↳ static const char* ${name} = \"${string}\";", null, 0),
82 // private: const int MaxPath = 92;
83 // private: static const int MaxPath = 92;
84 (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly)
85     ↳ (?<type>[a-zA-Z0-9]+) (?<name>[_a-zA-Z0-9_]+) = (?<value>[^\r\n]+);"),
86     ↳ "${access}static const ${type} ${name} = ${value};", null, 0),
87 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
88     ↳ TArgument : class
89 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
90 (new Regex(@"(?<before> [a-zA-Z]+\((([a-zA-Z *,]+, |)))(?<type>[a-zA-Z]+)(?<after>(|
91     ↳ [a-zA-Z *,]+)\))) [ \r\n]+where \k<type> : class", "${before}${type}*${after}",
92     ↳ null, 0),
93 // protected: abstract TElement GetFirst();
94 // protected: virtual TElement GetFirst() = 0;
95 (new Regex(@"(?<access>(private|protected|public): )?abstract
96     ↳ (?<method>[^\r\n]+);"), "${access}virtual ${method} = 0;", null, 0),
97 // TElement GetFirst();
98 // virtual TElement GetFirst() = 0;
99 (new Regex(@"([\r\n]+[ ]+)((?!return)[a-zA-Z0-9]+ [a-zA-Z0-9]+\([^\]\r\n]*\))(\;|
100     ↳ ]*\[ \r\n]+)", "$1virtual $2 = 0$3", null, 1),
101 // protected: readonly TreeElement[] _elements;
102 // protected: TreeElement _elements[N];
103 (new Regex(@"(?<access>(private|protected|public): )?readonly
104     ↳ (?<type>[a-zA-Z0-9]+)(\[\]\+)(?<name>[_a-zA-Z0-9_]+);"), "${access}${type}
105     ↳ ${name}[N];", null, 0),
106 // protected: readonly TElement Zero;
107 // protected: TElement Zero;
108 (new Regex(@"(?<access>(private|protected|public): )?readonly
109     ↳ (?<type>[a-zA-Z0-9]+) (?<name>[_a-zA-Z0-9_]+);"), "${access}${type} ${name};",
110     ↳ null, 0),
111 // internal
112 //
113 (new Regex(@"(\W)internal\s+"), "$1", null, 0),
114 // static void NotImplementedException(ThrowExtensionRoot root) => throw new
115     ↳ NotImplementedException();
116 // static void NotImplementedException(ThrowExtensionRoot root) { return throw new
117     ↳ NotImplementedException(); }
118 (new Regex(@"(^s+)(private|protected|public)?(: )?(template \<[^^\r\n]+\> )?(static
119     ↳ )?(override )?([a-zA-Z0-9]+
120     ↳ )([a-zA-Z0-9]+\)\(((\^[^\r\n]*)\)\s+=>\s+throw([^\r\n]+);"),
121     ↳ "$1$2$3$4$5$6$7$8($9) { throw$10; }", null, 0),
122 // SizeBalancedTree(int capacity) => a = b;
123 // SizeBalancedTree(int capacity) { a = b; }

```

```

98 (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; }", null, 0),
99 // int SizeBalancedTree(int capacity) => a;
100 // int SizeBalancedTree(int capacity) { return a; }
101 (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; }", null, 0),
102 // () => Integer<TElement>.Zero,
103 // () { return Integer<TElement>.Zero; },
104 (new Regex(@"\(\)\s+=>\s+(?<expression>[~() ,;\r\n]+(\(((?<parenthesis>\()|(?<-parent
   ↳ hesis>~\)|[~() ,;\r\n]*~)?[~() ,;\r\n]*)(?<after>,|~);)"), "()" { return
   ↳ ${expression}; }${after}", null, 0),
105 // => Integer<TElement>.Zero;
106 // { return Integer<TElement>.Zero; }
107 (new Regex(@"\)\s+=>\s+([~;\r\n]+?);"), "()" { return $1; }", null, 0),
108 // () { return avlTree.Count; }
109 // [&]() -> auto { return avlTree.Count; }
110 (new Regex(@"(?<before>, |)\(\)\{ return (?<expression>[~;\r\n]+); }"),
   ↳ "${before}&[]() -> auto { return ${expression}; }", null, 0),
111 // Count => GetSizeOrZero(Root);
112 // GetCount() { return GetSizeOrZero(Root); }
113 (new Regex(@"(\W)([A-Z][a-zA-Z]+)\s+=>\s+([~;\r\n]+);"), "$1Get$2() { return $3; }",
   ↳ null, 0),
114 // Func<TElement> treeCount
115 // std::function<TElement()> treeCount
116 (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", null,
   ↳ 0),
117 // Action<TElement> free
118 // std::function<void(TElement)> free
119 (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<void($1)> $2",
   ↳ null, 0),
120 // Predicate<TArgument> predicate
121 // std::function<bool(TArgument)> predicate
122 (new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<bool($1)>
   ↳ $2", null, 0),
123 // var
124 // auto
125 (new Regex(@"(\W)var(\W)"), "$1auto$2", null, 0),
126 // unchecked
127 //
128 (new Regex(@"[\r\n]{2}\s*?unchecked\s*?$"), "", null, 0),
129 // throw new InvalidOperationException
130 // throw std::runtime_error
131 (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
   ↳ std::runtime_error", null, 0),
132 // void RaiseExceptionIgnoredEvent(Exception exception)
133 // void RaiseExceptionIgnoredEvent(const std::exception& exception)
134 (new Regex(@"\(|, )(System\.Exception|Exception)(\|)"), "$1const
   ↳ std::exception&$3", null, 0),
135 // EventHandler<Exception>
136 // EventHandler<std::exception>
137 (new Regex(@"(\W)(System\.Exception|Exception)(\W)"), "$1std::exception$3", null, 0),
138 // override void PrintNode(TElement node, StringBuilder sb, int level)
139 // void PrintNode(TElement node, StringBuilder sb, int level) override
140 (new Regex(@"override ([a-zA-Z0-9 \*+]+)\(((~\(\r\n)+~?))"), "$1$2 override", null,
   ↳ 0),
141 // string
142 // const char*
143 (new Regex(@"(\W)string(\W)"), "$1const char*$2", null, 0),
144 // sbyte
145 // std::int8_t
146 (new Regex(@"(?<before>\W)((System\.)?SB|sbyte)(?!s*)(?<after>\W)"),
   ↳ "${before}std::int8_t${after}", null, 0),
147 // sbyte.MinValue
148 // INT8_MIN
149 (new Regex(@"(?<before>\W)std::int8_t\.MinValue(?<after>\W)"),
   ↳ "${before}INT8_MIN${after}", null, 0),
150 // sbyte.MaxValue
151 // INT8_MAX
152 (new Regex(@"(?<before>\W)std::int8_t\.MaxValue(?<after>\W)"),
   ↳ "${before}INT8_MAX${after}", null, 0),
153 // short
154 // std::int16_t
155 (new Regex(@"(?<before>\W)((System\.)?Int16|short)(?!s*)(?<after>\W)"),
   ↳ "${before}std::int16_t${after}", null, 0),

```

```

156 // short.MinValue
157 // INT16_MIN
158 (new Regex(@"(?<before>\W)std::int16_t\.MinValue(?<after>\W)"),
159     ↪ "${before}INT16_MIN${after}", null, 0),
160 // short.MaxValue
161 // INT16_MAX
162 (new Regex(@"(?<before>\W)std::int16_t\.MaxValue(?<after>\W)"),
163     ↪ "${before}INT16_MAX${after}", null, 0),
164 // int
165 // std::int32_t
166 (new Regex(@"(?<before>\W)((System\.)?I|i)nt(32)?(?!\s*)(?<after>\W)"),
167     ↪ "${before}std::int32_t${after}", null, 0),
168 // int.MinValue
169 // INT32_MIN
170 (new Regex(@"(?<before>\W)std::int32_t\.MinValue(?<after>\W)"),
171     ↪ "${before}INT32_MIN${after}", null, 0),
172 // int.MaxValue
173 // INT32_MAX
174 (new Regex(@"(?<before>\W)std::int32_t\.MaxValue(?<after>\W)"),
175     ↪ "${before}INT32_MAX${after}", null, 0),
176 // long
177 // std::int64_t
178 (new Regex(@"(?<before>\W)((System\.)?Int64|long)(?!\\s*)(?<after>\W)"),
179     ↪ "${before}std::int64_t${after}", null, 0),
180 // long.MinValue
181 // INT64_MIN
182 (new Regex(@"(?<before>\W)std::int64_t\.MinValue(?<after>\W)"),
183     ↪ "${before}INT64_MIN${after}", null, 0),
184 // long.MaxValue
185 // INT64_MAX
186 (new Regex(@"(?<before>\W)std::int64_t\.MaxValue(?<after>\W)"),
187     ↪ "${before}INT64_MAX${after}", null, 0),
188 // byte
189 // std::uint8_t
190 (new Regex(@"(?<before>\W)((System\.)?Byte|byte)(?!\\s*)(?<after>\W)"),
191     ↪ "${before}std::uint8_t${after}", null, 0),
192 // byte.MinValue
193 // (std::uint8_t)0
194 (new Regex(@"(?<before>\W)std::uint8_t\.MinValue(?<after>\W)"),
195     ↪ "${before}(std::uint8_t)0${after}", null, 0),
196 // byte.MaxValue
197 // UINT8_MAX
198 (new Regex(@"(?<before>\W)std::uint8_t\.MaxValue(?<after>\W)"),
199     ↪ "${before}UINT8_MAX${after}", null, 0),
200 // ushort
201 // std::uint16_t
202 (new Regex(@"(?<before>\W)((System\.)?UInt16|ushort)(?!\\s*)(?<after>\W)"),
203     ↪ "${before}std::uint16_t${after}", null, 0),
204 // ushort.MinValue
205 // (std::uint16_t)0
206 (new Regex(@"(?<before>\W)std::uint16_t\.MinValue(?<after>\W)"),
207     ↪ "${before}(std::uint16_t)0${after}", null, 0),
208 // ushort.MaxValue
209 // UINT16_MAX
210 (new Regex(@"(?<before>\W)std::uint16_t\.MaxValue(?<after>\W)"),
211     ↪ "${before}UINT16_MAX${after}", null, 0),
212 // uint
213 // std::uint32_t
214 (new Regex(@"(?<before>\W)((System\.)?UI|ui)nt(32)?(?!\s*)(?<after>\W)"),
215     ↪ "${before}std::uint32_t${after}", null, 0),
216 // uint.MinValue
217 // (std::uint32_t)0
218 (new Regex(@"(?<before>\W)std::uint32_t\.MinValue(?<after>\W)"),
219     ↪ "${before}(std::uint32_t)0${after}", null, 0),
220 // uint.MaxValue
221 // UINT32_MAX
222 (new Regex(@"(?<before>\W)std::uint32_t\.MaxValue(?<after>\W)"),
223     ↪ "${before}UINT32_MAX${after}", null, 0),
224 // ulong
225 // std::uint64_t
226 (new Regex(@"(?<before>\W)((System\.)?UInt64|ulong)(?!\\s*)(?<after>\W)"),
227     ↪ "${before}std::uint64_t${after}", null, 0),
228 // ulong.MinValue
229 // (std::uint64_t)0
230 (new Regex(@"(?<before>\W)std::uint64_t\.MinValue(?<after>\W)"),
231     ↪ "${before}(std::uint64_t)0${after}", null, 0),

```

```

213 // ulong.MaxValue
214 // UINT64_MAX
215 (new Regex(@"(?<before>\W)std::uint64_t\.MaxValue(?<after>\W)"),
  → "${before}UINT64_MAX${after}", null, 0),
216 // char*[] args
217 // char* args[]
218 (new Regex(@"([_a-zA-Z0-9:*\?])\[\] ([a-zA-Z0-9]+)"), "$1 $2[]", null, 0),
219 // @object
220 // object
221 (new Regex(@"@([_a-zA-Z0-9]+)"), "$1", null, 0),
222 // using Platform.Numbers;
223 //
224 (new Regex(@"([\r\n]{2}|~)\s*using [\a-zA-Z0-9]+;\s*?$"), "", null, 0),
225 // struct TreeElement { }
226 // struct TreeElement { };
227 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([~;])"), "$1
  → $2$3{$4};$5", null, 0),
228 // class Program { }
229 // class Program { };
230 (new Regex(@"(struct|class) ([a-zA-Z0-9]+[~\r\n]*)([\r\n]+(?<indentLevel>[\t
  → ]*)?)\{([\S\s]+?[\r\n]+\k<indentLevel>)\}([~;]|$)"), "$1 $2$3{$4};$5", null, 0),
231 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
232 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
233 (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)"), "class $1 : public $2", null,
  → 0),
234 // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
235 // class IProperty : public ISetter<TValue, TObject>, IProvider<TValue, TObject>
236 (new Regex(@"(?<before>class [a-zA-Z0-9]+ : ((public [a-zA-Z0-9]+(<[a-zA-Z0-9
  → ,]+>)?, )+)?(?<inheritedType>(?!public) [a-zA-Z0-9]+(<[a-zA-Z0-9
  → ,]+>)?(?<after>(, [a-zA-Z0-9]+(?!>)|[\r\n]+)))"), "${before}public
  → ${inheritedType}${after}", null, 10),
237 // Insert scope borders.
238 // ref TElement root
239 // ~!root!~ref TElement root
240 (new Regex(@"(?<definition>(?!<= |\\() (ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<ref>))
  → (?<variable>[a-zA-Z0-9]+(?=\\)|, | =))"), "~!${variable}!~${definition}", null,
  → 0),
241 // Inside the scope of ~!root!~ replace:
242 // root
243 // *root
244 (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}", null, 70),
245 // Remove scope borders.
246 // ~!root!~
247 //
248 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
249 // ref auto root = ref
250 // ref auto root =
251 (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)"), "$1* $2 =$3", null, 0),
252 // *root = ref left;
253 // root = left;
254 (new Regex(@"*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", null, 0),
255 // (ref left)
256 // (left)
257 (new Regex(@"\ (ref ([a-zA-Z0-9]+)(\\|\\(|,))"), "($1$2", null, 0),
258 // ref TElement
259 // TElement*
260 (new Regex(@"( |\\()ref ([a-zA-Z0-9]+) "), "$1$2* ", null, 0),
261 // ref sizeBalancedTree.Root
262 // &sizeBalancedTree->Root
263 (new Regex(@"ref ([a-zA-Z0-9]+)\\.([a-zA-Z0-9\\*]+)"), "&$1->$2", null, 0),
264 // ref GetElement(node).Right
265 // &GetElement(node)->Right
266 (new Regex(@"ref ([a-zA-Z0-9]+)\\((([a-zA-Z0-9\\*]+)\\)\\.([a-zA-Z0-9]+)"),
  → "&$1($2)->$3", null, 0),
267 // GetElement(node).Right
268 // GetElement(node)->Right
269 (new Regex(@"([a-zA-Z0-9]+)\\((([a-zA-Z0-9\\*]+)\\)\\.([a-zA-Z0-9]+)"), "$1($2)->$3",
  → null, 0),
270 // [Fact] npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
271 // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
272 (new Regex(@"\[Fact\\] [\s\\n]+(public: )?(static )?void ([a-zA-Z0-9]+)\\(\\)"), "public:
  → TEST_METHOD($3)", null, 0),
273 // class TreesTests

```

```

274 // TEST_CLASS(TreesTests)
275 (new Regex(@"class ([a-zA-Z0-9]+)Tests"), "TEST_CLASS($1)", null, 0),
276 // Assert.Equal
277 // Assert::AreEqual
278 (new Regex(@"(Assert)\.Equal"), "$1::AreEqual", null, 0),
279 // Assert.Throws
280 // Assert::ExpectException
281 (new Regex(@"(Assert)\.Throws"), "$1::ExpectException", null, 0),
282 // $"Argument {argumentName} is null."
283 // ((std::string)"Argument ").append(argumentName).append(" is null.").data()
284 (new Regex(@"\$"("(?<left>\\\"|\"[^\r\n]*)\"{(?<expression>[_a-zA-Z0-9]+)}\"{(?<right>\\\"|\"[^\r\n]*)\"}"),
    ↪ "\"\"[^\r\n]*)\""),
    ↪ "\"((std::string)$\"${left}\").append(${expression}).append(\"${right}\").data()",
    ↪ null, 10),
285 // $"
286 // "
287 (new Regex(@"\$"""), "\", null, 0),
288 // Console.WriteLine("...")
289 // printf("...\n")
290 (new Regex(@"Console\.WriteLine\\(\"\"([^\r\n]+)\"\"\\)"), "printf(\"$1\\n\\n\")", null, 0),
291 // TElement Root;
292 // TElement Root = 0;
293 (new Regex(@"(\\r?\\n[\\t ]+)(private|protected|public)?(:| )?(?<[_a-zA-Z0-9:]+>\\[+<?!return>)([_a-zA-Z0-9]+);"), "$1$2$3$4 $5 = 0;", null, 0),
294 // TreeElement _elements[N];
295 // TreeElement _elements[N] = { {0} };
296 (new Regex(@"(\\r?\\n[\\t ]+)(private|protected|public)?(:| )?(?<[_a-zA-Z0-9]+>\\[+<?!return>)([_a-zA-Z0-9]+)\\[\\[([_a-zA-Z0-9]+)\\];\"), "$1$2$3$4 $5[$6] = { {0} };", null, 0),
297 // auto path = new TElement[MaxPath];
298 // TElement path[MaxPath] = { {0} };
299 (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[$4] = { {0} };", null, 0),
300 // private: static readonly ConcurrentBag<std::exception> _exceptionsBag = new
    ↪ ConcurrentBag<std::exception>();
301 // private: inline static std::mutex _exceptionsBag_mutex; \\n\\n private: inline
    ↪ static std::vector<std::exception> _exceptionsBag;
302 (new Regex(@"\"{?<begin>\\r?\\n?(?<indent>[ \\t]+)}\"{?<access>(private|protected|public):",
    ↪ )?static readonly ConcurrentBag<(?<argumentType>[~;\\r\\n]+)>
    ↪ (?<name>[_a-zA-Z0-9]+) = new ConcurrentBag<\\k<argumentType>>\\(\\);\"),
    ↪ \"${begin}private: inline static std::mutex ${name}_mutex;\" + Environment.NewLine
    ↪ + Environment.NewLine + \"${indent}${access}inline static
    ↪ std::vector<${argumentType}> ${name};\", null, 0),
303 // public: static IReadOnlyCollection<std::exception> GetCollectedExceptions() {
    ↪ return _exceptionsBag; }
304 // public: static std::vector<std::exception> GetCollectedExceptions() { return
    ↪ std::vector<std::exception>(_exceptionsBag); }
305 (new Regex(@"\"{?<access>(private|protected|public): }?static",
    ↪ IReadOnlyCollection<(?<argumentType>[~;\\r\\n]+)> (?<methodName>[_a-zA-Z0-9]+)\\(\\)
    ↪ { return (?<fieldName>[_a-zA-Z0-9]+); }\"), \"${access}static
    ↪ std::vector<${argumentType}> ${methodName}() { return
    ↪ std::vector<${argumentType}>({${fieldName}}); }\", null, 0),
306 // public: static event EventHandler<std::exception> ExceptionIgnored =
    ↪ OnExceptionIgnored; ... };
307 // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
    ↪ const std::exception&> ExceptionIgnored = OnExceptionIgnored; };
308 (new Regex(@"\"{?<begin>\\r?\\n(\\r?\\n)?(?<halfIndent>[ \\t]+)\\k<halfIndent>\"{?<access>(private|protected|public): }?static event",
    ↪ EventHandler<(?<argumentType>[~;\\r\\n]+)> (?<name>[_a-zA-Z0-9]+) = (?<defaultDele",
    ↪ gate>[_a-zA-Z0-9]+);(?<middle>(\\.\\n)+?)\"{?<end>\\r?\\n\\k<halfIndent>};)\"),
    ↪ \"${middle}\" + Environment.NewLine + Environment.NewLine +
    ↪ \"${halfIndent}${halfIndent}${access}static inline
    ↪ Platform::Delegates::MulticastDelegate<void(void*, const ${argumentType}&>
    ↪ ${name} = ${defaultDelegate};${end}\"", null, 0),
309 // Insert scope borders.
310 // class IgnoredExceptions { ... private: inline static std::vector<std::exception>
    ↪ _exceptionsBag;
311 // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: inline static
    ↪ std::vector<std::exception> _exceptionsBag;
312 (new Regex(@"\"{?<classDeclarationBegin>\\r?\\n(?<indent>[ \\t ]*)class [~{\\r\\n]+\\r\\n[ \\t",
    ↪ ]*}{?<middle>((?!class)\\.\\n)+?}\"{?<vectorFieldDeclaration>(\\k<access>(private|pro",
    ↪ tected|public): )inline static std::vector<(?<argumentType>[~;\\r\\n]+)>
    ↪ (?<fieldName>[_a-zA-Z0-9]+);)\"),
    ↪ \"${classDeclarationBegin}/*~${fieldName}~*/${middle}${vectorFieldDeclaration}\"",
    ↪ null, 0),
313 // Inside the scope of ~!_exceptionsBag!~ replace:
314 // _exceptionsBag.Add(exception);

```

```

315 // _exceptionsBag.push_back(exception);
316 (new Regex(@"(?<scope>/\~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ e>((?!/\~\k<fieldName>~\*/)(.\|\\n))*?)\k<fieldName>\.Add"),
    ↪ "${scope}${separator}${before}${fieldName}.push_back", null, 10),
317 // Remove scope borders.
318 // /*~_exceptionsBag~*/
319 //
320 (new Regex(@"/*~[_a-zA-Z0-9]+~\*/"), "", null, 0),
321 // Insert scope borders.
322 // class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
323 // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: static std::mutex
    ↪ _exceptionsBag_mutex;
324 (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}", null,
    ↪ 0),
325 // Inside the scope of ~!_exceptionsBag!~ replace:
326 // return std::vector<std::exception>(_exceptionsBag);
327 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
    ↪ std::vector<std::exception>(_exceptionsBag);
328 (new Regex(@"(?<scope>/\~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ e>((?!/\~\k<fieldName>~\*/)(.\|\\n))*?)\{(?<after>((?!lock_guard)[^{};\r\n])*k<f
    ↪ ieldName>[~;}\r\n]*;)\r\n";)", "${scope}${separator}${before}{
    ↪ std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", null, 10),
329 // Inside the scope of ~!_exceptionsBag!~ replace:
330 // _exceptionsBag.Add(exception);
331 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
    ↪ _exceptionsBag.Add(exception);
332 (new Regex(@"(?<scope>/\~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ e>((?!/\~\k<fieldName>~\*/)(.\|\\n))*?)\{(?<after>((?!lock_guard)([~{};]\|\\n))*?\r
    ↪ ?\n(?<indent>[\t ]*)\k<fieldName>[~;}\r\n]*;)\r\n";)",
    ↪ "${scope}${separator}${before}{", Environment.NewLine +
    ↪ "${indent}std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", null,
    ↪ 10),
333 // Remove scope borders.
334 // /*~_exceptionsBag~*/
335 //
336 (new Regex(@"/*~[_a-zA-Z0-9]+~\*/"), "", null, 0),
337 // Insert scope borders.
338 // class IgnoredExceptions { ... public: static inline
    ↪ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
    ↪ ExceptionIgnored = OnExceptionIgnored;
339 // class IgnoredExceptions {/*~ExceptionIgnored~*/ ... public: static inline
    ↪ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
    ↪ ExceptionIgnored = OnExceptionIgnored;
340 (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n}]+\r\n[\t
    ↪ ]*(?<middle>((?!class).\|\\n)+)?(?<eventDeclaration>(?(<access>(private|protected)
    ↪ |public): )static inline
    ↪ Platform::Delegates::MulticastDelegate<(?(<argumentType>[~;\r\n]+)>
    ↪ (?<name>[_a-zA-Z0-9]+) = (?<defaultDelegate>[_a-zA-Z0-9]+);)",
    ↪ "${classDeclarationBegin}/*~${name}~/${middle}${eventDeclaration}", null, 0),
341 // Inside the scope of ~!ExceptionIgnored!~ replace:
342 // ExceptionIgnored.Invoke(NULL, exception);
343 // ExceptionIgnored(NULL, exception);
344 (new Regex(@"(?<scope>/\~(?<eventName>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ >((?!/\~\k<eventName>~\*/)(.\|\\n))*?)\k<eventName>\.Invoke)",
    ↪ "${scope}${separator}${before}${eventName}", null, 10),
345 // Remove scope borders.
346 // /*~ExceptionIgnored~*/
347 //
348 (new Regex(@"/*~[_a-zA-Z0-9]+~\*/"), "", null, 0),
349 // Insert scope borders.
350 // auto added = new StringBuilder();
351 // /*~sb~*/std::string added;
352 (new Regex(@"(auto|(System\.\Text\.)?StringBuilder) (?<variable>[_a-zA-Z0-9]+) = new
    ↪ (System\.\Text\.)?StringBuilder\\(\\);)", "/*~${variable}~*/std::string
    ↪ ${variable};", null, 0),
353 // static void Indent(StringBuilder sb, int level)
354 // static void Indent(/*~sb~*/StringBuilder sb, int level)
355 (new Regex(@"(?<start>, \|\\() (System\.\Text\.)?StringBuilder
    ↪ (?<variable>[_a-zA-Z0-9]+)(?<end>, \|\\))", "${start}/*~${variable}~*/std::string&
    ↪ ${variable}${end}", null, 0),
356 // Inside the scope of ~!added!~ replace:
357 // sb.ToString()
358 // sb.data()

```



```

359 (new Regex(@"(?<scope>/\~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↳ ((?!/\~\k<variable>~\*/)(.|\n))*?)\k<variable>\.ToString\\(\\"),
    ↳ $"{scope}${separator}${before}${variable}.data()", null, 10),
360 // sb.AppendLine(argument)
361 // sb.append(argument).append('\\n')
362 (new Regex(@"(?<scope>/\~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↳ ((?!/\~\k<variable>~\*/)(.|\n))*?)\k<variable>\.AppendLine\\((?<argument>[^\r\n]
    ↳ r\\n]+)\\)"),
    ↳ $"{scope}${separator}${before}${variable}.append(${argument}).append(1, '\\n')",
    ↳ null, 10),
363 // sb.Append('\\t', level);
364 // sb.append(level, '\\t');
365 (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}')" ,
    ↳ null, 10),
366 // sb.Append(argument)
367 // sb.append(argument)
368 (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(${argument})", null,
    ↳ 10),
369 // Remove scope borders.
370 // /*~sb~*/
371 //
372 (new Regex(@"/*~[a-zA-Z0-9]+~\*/"), "", null, 0),
373 // Insert scope borders.
374 // auto added = new HashSet<TElement>();
375 // ~!added!~std::unordered_set<TElement> added;
376 (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
    ↳ HashSet<(?<element>[a-zA-Z0-9]+)>\\(\\"),
    ↳ "~!${variable}!~std::unordered_set<${element}> ${variable};", null, 0),
377 // Inside the scope of ~!added!~ replace:
378 // added.Add(node)
379 // added.insert(node)
380 (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})", null, 10),
381 // Inside the scope of ~!added!~ replace:
382 // added.Remove(node)
383 // added.erase(node)
384 (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(${argument})", null, 10),
385 // if (added.insert(node)) {
386 // if (!added.contains(node)) { added.insert(node);
387 (new Regex(@"if \\((?<variable>[a-zA-Z0-9]+)\\.insert\\((?<argument>[a-zA-Z0-9]+)\\)\\)(?
    ↳ <separator>[\\t ]*[\\r\\n]+)(?<indent>[\\t ]*){", "if
    ↳ (!${variable}.contains(${argument})) ${separator}${indent}{\" +
    ↳ Environment.NewLine + "${indent}    ${variable}.insert(${argument});", null, 0),
388 // Remove scope borders.
389 // ~!added!~
390 //
391 (new Regex(@"~![a-zA-Z0-9]+!~"), "", null, 5),
392 // Insert scope borders.
393 // auto random = new System.Random();
394 // std::srand(0);
395 (new Regex(@"[a-zA-Z0-9\\.]+ ([a-zA-Z0-9]+) = new
    ↳ (System\\.)?Random\\((([a-zA-Z0-9]+)\\)");, "~!$1!~std::srand($3);", null, 0),
396 // Inside the scope of ~!random!~ replace:
397 // random.Next(1, N)
398 // (std::rand() % N) + 1
399 (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}", null, 10),
400 // Remove scope borders.
401 // ~!random!~
402 //
403 (new Regex(@"~![a-zA-Z0-9]+!~"), "", null, 5),
404 // Insert method body scope starts.
405 // void PrintNodes(TElement node, StringBuilder sb, int level) {
406 // void PrintNodes(TElement node, StringBuilder sb, int level) {/*method-start*/

```



```

407 (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*/${end}", null,
    → 0),
408 // Insert method body scope ends.
409 // {/*method-start*/...}
410 // {/*method-start*/.../*method-end*/}
411 (new Regex(@"{/{/*method-start*/(?<body>((?<bracket>\{)|(?<-bracket>\})|[-\{\}]*)+)
    → \}"}), "{/{/*method-start*/${body}/*method-end*/}", null,
    → 0),
412 // Inside method bodies replace:
413 // GetFirst(
414 // this->GetFirst(
415 // (new Regex(@"(?<separator>(\(|\|([\W])|return ))(?<!(->|\*
    → ))(?<method>(?!sizeof)[a-zA-Z0-9]+)\(((?!\\) \{)"),
    → "${separator}this->${method}(", null, 1),
416 (new Regex(@"(?<scope>/{/*method-start*/})(?<before>((?<!(/*method-end*/)(\|\\n))*?) (
    → ?<separator>[\W] (?<!(::|\\.|->)))(?<method>(?!sizeof)[a-zA-Z0-9]+)\(((?!\\)
    → \{) (?<after>(\|\\n)*?) (?<scopeEnd>/{/*method-end*/})",
    → "${scope}${before}${separator}this->${method}(${after}${scopeEnd}", null, 100),
417 // Remove scope borders.
418 // /*method-start*/
419 //
420 (new Regex(@"/{/*method-(start|end)*/}"), "", null, 0),
421 // Insert scope borders.
422 // const std::exception& ex
423 // const std::exception& ex/*~ex~*/
424 (new Regex(@"(?<before>\\(|) (?<variableDefinition>(const )?(std::)?exception&?
    → (?<variable>[_a-zA-Z0-9]+)) (?<after>\\W)"),
    → "${before}${variableDefinition}/*~${variable}~/${after}", null, 0),
425 // Inside the scope of ~!ex!~ replace:
426 // ex.Message
427 // ex.what()
428 (new Regex(@"(?<scope>/{/*~(?<variable>[_a-zA-Z0-9]+)~*/})(?<separator>\\(|\\n) (?<before>
    → >((?!/*~\\k<variable>~*/)(\|\\n))*?)\\k<variable>\\.Message"),
    → "${scope}${separator}${before}${variable}.what()", null, 10),
429 // Remove scope borders.
430 // /*~ex~*/
431 //
432 (new Regex(@"/{/*~[_a-zA-Z0-9]+~*/}"), "", null, 0),
433 // throw new ArgumentNullException(argumentName, message);
434 // throw std::invalid_argument(((std::string)"Argument
    → ").append(argumentName).append(" is null: ").append(message).append("."));
435 (new Regex(@"throw new
    → 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(\".\");", null, 0),
436 // throw new ArgumentException(message, argumentName);
437 // throw std::invalid_argument(((std::string)"Invalid
    → ").append(argumentName).append(" argument: ").append(message).append("."));
438 (new Regex(@"throw new 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(\".\");", null, 0),
439 // throw new NotSupportedException();
440 // throw std::logic_error("Not supported exception.");
441 (new Regex(@"throw new NotSupportedException\\(\\);"), "throw std::logic_error(\"Not
    → supported exception.\");", null, 0),
442 // throw new NotImplementedException();
443 // throw std::logic_error("Not implemented exception.");
444 (new Regex(@"throw new NotImplementedException\\(\\);"), "throw std::logic_error(\"Not
    → implemented exception.\");", null, 0),
445 }.Cast<ISubstitutionRule>().ToList();
446
447 public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
448 {
449     // ICounter<int, int> c1;
450     // ICounter<int, int>* c1;
451     (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[~>\r\n]+>)?
    → (?<variable>[_a-zA-Z0-9]+);"), "${abstractType}* ${variable};", null, 0),
452     // (expression)
453     // expression
454     (new Regex(@"\\(\\(|)\\(((\\[a-zA-Z0-9_\\*:]*)\\)(\\(|\\|;|\\|)\\)\\)", "$1$2$3", null, 0),

```

```

455 // (method(expression))
456 // method(expression)
457 (new Regex(@"(?<firstSeparator>\(|
    ↳ ))\((?<method>[a-zA-Z0-9_\->\*:]*)\((?<expression>((?<parenthesis>\(|(?<-parenthesis>\)|[a-zA-Z0-9_\->\*:]*)+)(?<parenthesis>(?!))\)|(?<lastSeparator>(|;|;|)))") , "${firstSeparator}${method}(${expression})${lastSeparator}" , null , 0) ,
458 // return ref _elements[node];
459 // return &_elements[node];
460 (new Regex(@"return ref ([_a-zA-Z0-9]+)\([([_a-zA-Z0-9\*]+)\];") , "return &$1[$2];" ,
    ↳ null , 0) ,
461 // null
462 // nullptr
463 (new Regex(@"(?<before>\r?\n[~""\r\n]*(""(\\"""|~""\r\n))*""[~""\r\n]*)*(?<=\\W)null" ,
    ↳ (?<after>\\W)" , "${before}nullptr${after}" , null ,
    ↳ 10) ,
464 // default
465 // 0
466 (new Regex(@"(?<before>\r?\n[~""\r\n]*(""(\\"""|~""\r\n))*""[~""\r\n]*)*(?<=\\W)default" ,
    ↳ (?<after>\\W)" , "${before}0${after}" , null ,
    ↳ 10) ,
467 // object x
468 // void *x
469 (new Regex(@"(?<before>\r?\n[~""\r\n]*(""(\\"""|~""\r\n))*""[~""\r\n]*)*(?<=\\W)([O]b
    ↳ ject|System\\.Object) (?<after>\\w)" , "${before}void *${after}" , null ,
    ↳ 10) ,
470 // <object>
471 // <void*>
472 (new Regex(@"(?<before>\r?\n[~""\r\n]*(""(\\"""|~""\r\n))*""[~""\r\n]*)*(?<=\\W)(?!
    ↳ \\w)([O]bject|System\\.Object) (?<after>\\W)" , "${before}void*${after}" , null ,
    ↳ 10) ,
473 // ArgumentException
474 // std::invalid_argument
475 (new Regex(@"(?<before>\r?\n[~""\r\n]*(""(\\"""|~""\r\n))*""[~""\r\n]*)*(?<=\\W)(Sys
    ↳ tem\\.)?ArgumentException (?<after>\\W)" ,
    ↳ "${before}std::invalid_argument${after}" , null , 10) ,
476 // #region Always
477 //
478 (new Regex(@"(~\\r?\n)[ \t]*#(region|endregion)[~\\r\n]*(\\r?\n|$)" , "" , null , 0) ,
479 // //define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
480 //
481 (new Regex(@"\\[/[ \t]*#define[ \t]+[_a-zA-Z0-9]+[ \t]*") , "" , null , 0) ,
482 // #if USEARRAYPOOL\\r\\n#endif
483 //
484 (new Regex(@"#if [a-zA-Z0-9]+\\s+endif") , "" , null , 0) ,
485 // [Fact]
486 //
487 (new Regex(@"(?<firstNewLine>\\r?\n|\\A) (?<indent>[ \t
    ↳ ]+)[[a-zA-Z0-9]+\\((?<expression>((?<parenthesis>\(|(?<-parenthesis>\)|[~""\r\n]*
    ↳ )+)(?<parenthesis>(?!))\)|[~""\r\n]*\\k<indent>)?)" ,
    ↳ "${firstNewLine}${indent}" , null , 5) ,
488 // \\n ... namespace
489 // namespace
490 (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+namespace") , "$1namespace" , null , 0) ,
491 // \\n ... class
492 // class
493 (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+class") , "$1class" , null , 0) ,
494 }.Cast<ISubstitutionRule>().ToList();
495
496 public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
    ↳ base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
497
498 public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
499 }
500 }

```

## 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("", new Context(null));
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(const char* args[])
30     {
31         printf(""Hello, world!\n"");
32     }
33 };";
34         var transformer = new CSharpToCppTransformer();
35         var actualResult = transformer.Transform(helloWorldCode, new Context(null));
36         Assert.Equal(expectedResult, actualResult);
37     }
38 }
39 }
```

## Index

./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs, 10

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