

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             (new Regex(@"^-s*?#pragma[sa-zA-Z0-9]+$"), "", null, 0),
20             // {\n\n\n
21             // {
22             (new Regex(@"{\s+[\r\n]+") , "{" + Environment.NewLine, null, 0),
23             // Platform.Collections.Methods.Lists
24             // Platform::Collections::Methods::Lists
25             (new Regex(@"(namespace[^\r\n]+?)\.((^\r\n)+?)") , "$1::$2", null, 20),
26             // Comparer<TArgument>.Default.Compare(maximumArgument, minimumArgument) < 0
27             // maximumArgument < minimumArgument
28             (new Regex(@"Comparer<[^>\n]+>\.Default\.Compare\\(s*(?<first>[^,)\n]+),s*(?<second>
29             >[^>\n]+)s*)\\s*(?<comparison>[<>=]=?)s*0") , "${first} ${comparison}
30             ${second}", null, 0),
31             // out TProduct
32             // TProduct
33             (new Regex(@"(?<before>(<|, ))(in|out)
34             > (?<typeParameter>[a-zA-Z0-9]+)(?<after>(>|,))") ,
35             "${before}${typeParameter}${after}", null, 10),
36             // public ...
37             // public: ...
38             (new Regex(@"(?<newLineAndIndent>\r?\n?[
39             \t]*) (?<before>[^\{\\(\r\n)*] (?<access>private|protected|public) [
40             \t]+(?! [^\{\\(\r\n)* (interface|class|struct) [^\{\\(\r\n)* [^\{\\(\r\n)"] )",
41             "${newLineAndIndent}${access}: ${before}", null, 0),
42             // public: static bool CollectExceptions { get; set; }
43             // public: inline static bool CollectExceptions;
44             (new Regex(@"(?<access>(private|protected|public): ) (?<before>(static )? [^\r\n]+
45             ) (?<name>[a-zA-Z0-9]+) { [^;]* (?<=\\W) get; [^;]* (?<=\\W) set; [^;]* }" ,
46             "${access}inline ${before}${name};", null, 0),
47             // public abstract class
48             // class
49             (new Regex(@"((public|protected|private|internal|abstract|static)
50             )*(?<category>interface|class|struct)" , "${category}", null, 0),
51             // class GenericCollectionMethodsBase<TElement> {
52             // template <typename TElement> class GenericCollectionMethodsBase {
53             (new Regex(@"class ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>([^\{]+){", "template <typename $2>
54             class $1$3{", null, 0),
55             // static void
56             > TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
57             > tree, TElement* root)
58             // template<typename T> static void
59             > TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
60             > tree, TElement* root)
61             (new Regex(@"static ([a-zA-Z0-9]+) ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>\\(((^\\)\r\n)+\\)" ,
62             "template <typename $3> static $1 $2($4)", null, 0),
63             // interface IFactory<out TProduct> {
64             // template <typename TProduct> class IFactory { public:
65             (new Regex(@"interface (?<interface>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9
66             ,]+)> (?<whitespace>[^\{]+){", "template <typename...> class ${interface};
67             template <typename ${typeParameters}> class
68             ${interface}<${typeParameters}>${whitespace}{ " + Environment.NewLine + "
69             public:", null, 0),
70             // template <typename TObject, TProperty, TValue>
71             // template <typename TObject, typename TProperty, TValue>
72             (new Regex(@"(?<before>template <((, )?typename [a-zA-Z0-9]+)+,
73             ) (?<typeParameter>[a-zA-Z0-9]+) (?<after>(>|,))" , "${before}typename
74             ${typeParameter}${after}", null, 10),

```

```

53 // Insert markers
54 // private: static void BuildExceptionString(this StringBuilder sb, Exception
    ↳ exception, int level)
55 // /*~extensionMethod~BuildExceptionString~*/private: static void
    ↳ BuildExceptionString(this StringBuilder sb, Exception exception, int level)
56 (new Regex(@"private: static [\r\n]+ (?<name>[a-zA-Z0-9]+)\(this [\r\n]+\)",
    ↳ "/*~extensionMethod~${name}~*/$0", null, 0),
57 // Move all markers to the beginning of the file.
58 (new Regex(@"\A(?<before>[\r\n]+\r?\n(.|\n)+)(?<marker>\/\*~extensionMethod~(?<name>
    ↳ [a-zA-Z0-9]+)~\*/)", "${marker}${before}", null,
    ↳ 10),
59 // /*~extensionMethod~BuildExceptionString~*/...sb.BuildExceptionString(exception.In
    ↳ nerException, level +
    ↳ 1);
60 // /*~extensionMethod~BuildExceptionString~*/...BuildExceptionString(sb,
    ↳ exception.InnerException, level + 1);
61 (new Regex(@"(?<before>\/\*~extensionMethod~(?<name>[a-zA-Z0-9]+)~\*/(.|\n)+\W)(?<var
    ↳ iable>[_a-zA-Z0-9]+\.\k<name>\(", "${before}${name}(${variable}, ", null,
    ↳ 50),
62 // Remove markers
63 // /*~extensionMethod~BuildExceptionString~*/
64 //
65 (new Regex(@"\/\*~extensionMethod~[a-zA-Z0-9]+~\*/", "", null, 0),
66 // (this
67 // (
68 (new Regex(@"\(this ", "(", null, 0),
69 // public: static readonly EnsureAlwaysExtensionRoot Always = new
    ↳ EnsureAlwaysExtensionRoot();
70 // public:inline static EnsureAlwaysExtensionRoot Always;
71 (new Regex(@"(?<access>(private|protected|public): )?static readonly
    ↳ (?<type>[a-zA-Z0-9]+) (?<name>[a-zA-Z0-9_]+) = new \k<type>\(\);",
    ↳ "${access}inline static ${type} ${name};", null, 0),
72 // public: static readonly string ExceptionContentsSeparator = "---";
73 // public: inline static const char* ExceptionContentsSeparator = "---";
74 (new Regex(@"(?<access>(private|protected|public): )?static readonly string
    ↳ (?<name>[a-zA-Z0-9_]+) = ""(?<string>\\\"|\\\"\\\r\n]+)"";", "${access}inline
    ↳ static const char* ${name} = \"${string}\";", null, 0),
75 // private: const int MaxPath = 92;
76 // private: static const int MaxPath = 92;
77 (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly)
    ↳ (?<type>[a-zA-Z0-9]+) (?<name>[_a-zA-Z0-9]+) = (?<value>[~;\r\n]+);",
    ↳ "${access}static const ${type} ${name} = ${value};", null, 0),
78 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
    ↳ TArgument : class
79 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
80 (new Regex(@"(?<before> [a-zA-Z]+\((([a-zA-Z *,,]+, |)))(?<type>[a-zA-Z]+)(?<after>(
    ↳ [a-zA-Z *,,]+\))) [ \r\n]+where \k<type> : class)", "${before}${type}*${after}",
    ↳ null, 0),
81 // protected: abstract TElement GetFirst();
82 // protected: virtual TElement GetFirst() = 0;
83 (new Regex(@"(?<access>(private|protected|public): )?abstract
    ↳ (?<method>[~;\r\n]+);", "${access}virtual ${method} = 0;", null, 0),
84 // TElement GetFirst();
85 // virtual TElement GetFirst() = 0;
86 (new Regex(@"([\r\n]+[ ]+)((?!return)[a-zA-Z0-9]+ [a-zA-Z0-9]+\((~\r\n)*\))(\;[
    ↳ ]*\r\n+)", "$1virtual $2 = 0$3", null, 1),
87 // protected: readonly TreeElement[] _elements;
88 // protected: TreeElement _elements[N];
89 (new Regex(@"(?<access>(private|protected|public): )?readonly
    ↳ (?<type>[a-zA-Z<0-9]+)([\[]+) (?<name>[_a-zA-Z0-9]+);", "${access}${type}
    ↳ ${name}[N];", null, 0),
90 // protected: readonly TElement Zero;
91 // protected: TElement Zero;
92 (new Regex(@"(?<access>(private|protected|public): )?readonly
    ↳ (?<type>[a-zA-Z<0-9]+) (?<name>[_a-zA-Z0-9]+);", "${access}${type} ${name};",
    ↳ null, 0),
93 // internal
94 //
95 (new Regex(@"(\W)internal\s+)", "$1", null, 0),
96 // static void NotImplementedException(ThrowExtensionRoot root) => throw new
    ↳ NotImplementedException();
97 // static void NotImplementedException(ThrowExtensionRoot root) { return throw new
    ↳ NotImplementedException(); }

```

```

98 (new Regex(@"(^s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static
   ↳ )?(override )?([a-zA-Z0-9]+
   ↳ )([a-zA-Z0-9]+)\(((^(\r\n)*)\)\s+=>\s+throw([~;\r\n]+);"),
   ↳ "$1$2$3$4$5$6$7$8($9) { throw$10; }", null, 0),
99 // SizeBalancedTree(int capacity) => a = b;
100 // SizeBalancedTree(int capacity) { a = b; }
101 (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),
102 // int SizeBalancedTree(int capacity) => a;
103 // int SizeBalancedTree(int capacity) { return a; }
104 (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),
105 // () => Integer<TElement>.Zero,
106 // () { return Integer<TElement>.Zero; },
107 (new Regex(@"(\)\s+=>\s+(?<expression>[^(,;~;\r\n]+(\(((?<parenthesis>\()|(?<-parent
   ↳ hesis>))|([^(,;~;\r\n]*)*?)?([^(,;~;\r\n]*)?(?<after>,|~;))"), "() { return
   ↳ ${expression}; }${after}", null, 0),
   ↳ // => Integer<TElement>.Zero;
108 // { return Integer<TElement>.Zero; }
109 (new Regex(@"\)\s+=>\s+([~;\r\n]+?);"), ") { return $1; }", null, 0),
110 // () { return avlTree.Count; }
111 // [&]() -> auto { return avlTree.Count; }
112 (new Regex(@"(?<before>, |() \() { return (?<expression>[~;\r\n]+); }"),
   ↳ "${before}[&]() -> auto { return ${expression}; }", null, 0),
113 // Count => GetSizeOrZero(Root);
114 // GetCount() { return GetSizeOrZero(Root); }
115 (new Regex(@"(\W)([A-Z][a-zA-Z]+)\s+=>\s+([~;\r\n]+);"), "$1Get$2() { return $3; }",
   ↳ null, 0),
116 // Func<TElement> treeCount
117 // std::function<TElement()> treeCount
118 (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", null,
   ↳ 0),
119 // Action<TElement> free
120 // std::function<void(TElement)> free
121 (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<void($1)> $2",
   ↳ null, 0),
122 // Predicate<TArgument> predicate
123 // std::function<bool(TArgument)> predicate
124 (new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<bool($1)>
   ↳ $2", null, 0),
125 // var
126 // auto
127 (new Regex(@"(\W)var(\W)"), "$1auto$2", null, 0),
128 // unchecked
129 //
130 (new Regex(@"[\r\n]{2}\s*?unchecked\s*?$"), "", null, 0),
131 // throw new InvalidOperationException
132 // throw std::runtime_error
133 (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
   ↳ std::runtime_error", null, 0),
134 // void RaiseExceptionIgnoredEvent(Exception exception)
135 // void RaiseExceptionIgnoredEvent(const std::exception& exception)
136 (new Regex(@"(\(|, )(\System\.Exception|Exception)( |\\)"), "$1const
   ↳ std::exception&$3", null, 0),
137 // EventHandler<Exception>
138 // EventHandler<std::exception>
139 (new Regex(@"(\W)(\System\.Exception|Exception)(\W)"), "$1std::exception$3", null, 0),
140 // override void PrintNode(TElement node, StringBuilder sb, int level)
141 // void PrintNode(TElement node, StringBuilder sb, int level) override
142 (new Regex(@"override ([a-zA-Z0-9 \*+]+)\(((^(\r\n)+?)\))"), "$1$2 override", null,
   ↳ 0),
143 // return (range.Minimum, range.Maximum)
144 // return {range.Minimum, range.Maximum}
145 (new Regex(@"(?<before>return\s*)\(((?<values>[^\n]+\n+)\)(?!\\)(?<after>\W)"),
   ↳ "${before}${values}${after}", null, 0),
146 // string
147 // const char*
148 (new Regex(@"(\W)string(\W)"), "$1const char*$2", null, 0),
149 // System.ValueTuple
150 // std::tuple
151 (new Regex(@"(?<before>\W)(\System\.)?ValueTuple(?:\s*=)(?<after>\W)"),
   ↳ "${before}std::tuple${after}", null, 0),
152 // sbyte
153 // std::int8_t

```

```

155 (new Regex(@"(?<before>\W)((System\.)?SB|sb)yte(?:\s*)(?<after>\W)"),
156     ↳ "${before}std::int8_t${after}", null, 0),
157 // sbyte.MinValue
158 // INT8_MIN
159 (new Regex(@"(?<before>\W)std::int8_t\.MinValue(?<after>\W)"),
160     ↳ "${before}INT8_MIN${after}", null, 0),
161 // sbyte.MaxValue
162 // INT8_MAX
163 (new Regex(@"(?<before>\W)std::int8_t\.MaxValue(?<after>\W)"),
164     ↳ "${before}INT8_MAX${after}", null, 0),
165 // short
166 // std::int16_t
167 (new Regex(@"(?<before>\W)((System\.)?Int16|short)(?:\s*)(?<after>\W)"),
168     ↳ "${before}std::int16_t${after}", null, 0),
169 // short.MinValue
170 // INT16_MIN
171 (new Regex(@"(?<before>\W)std::int16_t\.MinValue(?<after>\W)"),
172     ↳ "${before}INT16_MIN${after}", null, 0),
173 // short.MaxValue
174 // INT16_MAX
175 (new Regex(@"(?<before>\W)std::int16_t\.MaxValue(?<after>\W)"),
176     ↳ "${before}INT16_MAX${after}", null, 0),
177 // int
178 // std::int32_t
179 (new Regex(@"(?<before>\W)((System\.)?I|i)nt(32)?(?:\s*)(?<after>\W)"),
180     ↳ "${before}std::int32_t${after}", null, 0),
181 // int.MinValue
182 // INT32_MIN
183 (new Regex(@"(?<before>\W)std::int32_t\.MinValue(?<after>\W)"),
184     ↳ "${before}INT32_MIN${after}", null, 0),
185 // int.MaxValue
186 // INT32_MAX
187 (new Regex(@"(?<before>\W)std::int32_t\.MaxValue(?<after>\W)"),
188     ↳ "${before}INT32_MAX${after}", null, 0),
189 // long
190 // std::int64_t
191 (new Regex(@"(?<before>\W)((System\.)?Int64|long)(?:\s*)(?<after>\W)"),
192     ↳ "${before}std::int64_t${after}", null, 0),
193 // long.MinValue
194 // INT64_MIN
195 (new Regex(@"(?<before>\W)std::int64_t\.MinValue(?<after>\W)"),
196     ↳ "${before}INT64_MIN${after}", null, 0),
197 // long.MaxValue
198 // INT64_MAX
199 (new Regex(@"(?<before>\W)std::int64_t\.MaxValue(?<after>\W)"),
200     ↳ "${before}INT64_MAX${after}", null, 0),
201 // byte
202 // std::uint8_t
203 (new Regex(@"(?<before>\W)((System\.)?Byte|byte)(?:\s*)(?<after>\W)"),
204     ↳ "${before}std::uint8_t${after}", null, 0),
205 // byte.MinValue
206 // (std::uint8_t)0
207 (new Regex(@"(?<before>\W)std::uint8_t\.MinValue(?<after>\W)"),
208     ↳ "${before}(std::uint8_t)0${after}", null, 0),
209 // byte.MaxValue
210 // UINT8_MAX
211 (new Regex(@"(?<before>\W)std::uint8_t\.MaxValue(?<after>\W)"),
    ↳ "${before}UINT8_MAX${after}", null, 0),
// ushort
// std::uint16_t
(new Regex(@"(?<before>\W)((System\.)?UInt16|ushort)(?:\s*)(?<after>\W)"),
    ↳ "${before}std::uint16_t${after}", null, 0),
// ushort.MinValue
// (std::uint16_t)0
(new Regex(@"(?<before>\W)std::uint16_t\.MinValue(?<after>\W)"),
    ↳ "${before}(std::uint16_t)0${after}", null, 0),
// ushort.MaxValue
// UINT16_MAX
(new Regex(@"(?<before>\W)std::uint16_t\.MaxValue(?<after>\W)"),
    ↳ "${before}UINT16_MAX${after}", null, 0),
// uint
// std::uint32_t
(new Regex(@"(?<before>\W)((System\.)?UI|ui)nt(32)?(?:\s*)(?<after>\W)"),
    ↳ "${before}std::uint32_t${after}", null, 0),
// uint.MinValue
// (std::uint32_t)0

```

```

212 (new Regex(@"(?<before>\W)std::uint32_t\.MinValue(?<after>\W)"),
213     ↳ "${before}(std::uint32_t)0${after}", null, 0),
214 // uint.MaxValue
215 // UINT32_MAX
216 (new Regex(@"(?<before>\W)std::uint32_t\.MaxValue(?<after>\W)"),
217     ↳ "${before}UINT32_MAX${after}", null, 0),
218 // ulong
219 // std::uint64_t
220 (new Regex(@"(?<before>\W)((System\.)?UInt64|ulong)(?!\\s*)(?<after>\W)"),
221     ↳ "${before}std::uint64_t${after}", null, 0),
222 // ulong.MinValue
223 // (std::uint64_t)0
224 (new Regex(@"(?<before>\W)std::uint64_t\.MinValue(?<after>\W)"),
225     ↳ "${before}(std::uint64_t)0${after}", null, 0),
226 // ulong.MaxValue
227 // UINT64_MAX
228 (new Regex(@"(?<before>\W)std::uint64_t\.MaxValue(?<after>\W)"),
229     ↳ "${before}UINT64_MAX${after}", null, 0),
230 // char*[] args
231 // char* args[]
232 (new Regex(@"([_a-zA-Z0-9:\*\?]?)\\[\\] ([_a-zA-Z0-9]+)"), "$1 $2[]", null, 0),
233 // @object
234 // object
235 (new Regex(@"@([_a-zA-Z0-9]+)"), "$1", null, 0),
236 // using Platform.Numbers;
237 //
238 (new Regex(@"([\\r\\n]{2}|~)\\s*?using [\\_a-zA-Z0-9]+;\\s*?$"), "", null, 0),
239 // struct TreeElement { }
240 // struct TreeElement { };
241 (new Regex(@"(struct|class) ([_a-zA-Z0-9]+)(\\s+){([\\sa-zA-Z0-9;:_]+?)}([~;])"), "$1
242     ↳ $2$3{$4};$5", null, 0),
243 // class Program { }
244 // class Program { };
245 (new Regex(@"(struct|class) ([_a-zA-Z0-9]+)[~\\r\\n]*([\\r\\n]+(?<indentLevel>[\\t
246     ↳ ]*)?)\\{([\\S\\s]+?[\\r\\n]+\\k<indentLevel>)\\}([~;]|$)", "$1 $2$3{$4};$5", null, 0),
247 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
248 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
249 (new Regex(@"class ([_a-zA-Z0-9]+) : ([_a-zA-Z0-9]+)"), "class $1 : public $2", null,
250     ↳ 0),
251 // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
252 // class IProperty : public ISetter<TValue, TObject>, IProvider<TValue, TObject>
253 (new Regex(@"(?<before>class [_a-zA-Z0-9]+ : ((public [_a-zA-Z0-9]+(<[_a-zA-Z0-9
254     ↳ ,]+>)?, )+)?)(?<inheritedType>(?!public) [_a-zA-Z0-9]+(<[_a-zA-Z0-9
255     ↳ ,]+>)?)(?<after>(, [_a-zA-Z0-9]+(?>!)| [\\r\\n]+))"), "${before}public
256     ↳ ${inheritedType}${after}", null, 10),
257 // Insert scope borders.
258 // ref TEElement root
259 // ~!root!~ref TEElement root
260 (new Regex(@"(?<definition>(?!<= |\\() (ref [_a-zA-Z0-9]+|[_a-zA-Z0-9]+(?<!ref))
261     ↳ (?<variable>[_a-zA-Z0-9]+)(?!<= |\\() (ref [_a-zA-Z0-9]+|[_a-zA-Z0-9]+(?<!ref))
262     ↳ (?<variable>[_a-zA-Z0-9]+)(?!<= |\\() (ref [_a-zA-Z0-9]+|[_a-zA-Z0-9]+(?<!ref))", null,
263     ↳ 0),
264 // Inside the scope of ~!root!~ replace:
265 // root
266 // *root
267 (new Regex(@"(?<definition>~!(?<pointer>[_a-zA-Z0-9]+)!~ref [_a-zA-Z0-9]+
268     ↳ \\k<pointer>(?!<= |\\() (ref [_a-zA-Z0-9]+|[_a-zA-Z0-9]+(?<!ref))", null,
269     ↳ 0),
270     ↳ \\k<pointer>(?!<= |\\() (ref [_a-zA-Z0-9]+|[_a-zA-Z0-9]+(?<!ref))", null,
271     ↳ 0),
272     ↳ "${definition}${before}${prefix}*${pointer}${suffix}", null, 70),
273 // Remove scope borders.
274 // ~!root!~
275 //
276 (new Regex(@"~!(?<pointer>[_a-zA-Z0-9]+)!~"), "", null, 5),
277 // ref auto root = ref
278 // ref auto root =
279 (new Regex(@"ref ([_a-zA-Z0-9]+) ([_a-zA-Z0-9]+) = ref(\\W)"), "$1* $2 =$3", null, 0),
280 // *root = ref left;
281 // root = left;
282 (new Regex(@"\\*([_a-zA-Z0-9]+) = ref ([_a-zA-Z0-9]+)(\\W)"), "$1 = $2$3", null, 0),
283 // (ref left)
284 // (left)
285 (new Regex(@"\\(ref ([_a-zA-Z0-9]+)(\\) |\\(|,)", "($1$2", null, 0),
286 // ref TEElement
287 // TEElement*
288 (new Regex(@"( |\\()ref ([_a-zA-Z0-9]+) "), "$1$2* ", null, 0),
289 // ref sizeBalancedTree.Root
290 // &sizeBalancedTree->Root

```

```

272 (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)", "&$1->$2", null, 0),
273 // ref GetElement(node).Right
274 // &GetElement(node)->Right
275 (new Regex(@"ref ([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+))\)\.([a-zA-Z0-9]+)",
    ↳ "&$1($2)->$3", null, 0),
276 // GetElement(node).Right
277 // GetElement(node)->Right
278 (new Regex(@"([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+))\)\.([a-zA-Z0-9]+)", "$1($2)->$3",
    ↳ null, 0),
279 // [Fact]\npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
280 // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
281 (new Regex(@"\[Fact\]\[s\n\]+(public:)?(static)?void ([a-zA-Z0-9]+)\(\)", "public:
    ↳ TEST_METHOD($3)", null, 0),
282 // class TreesTests
283 // TEST_CLASS(TreesTests)
284 (new Regex(@"class ([a-zA-Z0-9]+)Tests", "TEST_CLASS($1)", null, 0),
285 // Assert.Equal
286 // Assert::AreEqual
287 (new Regex(@"(Assert)\.Equal", "$1::AreEqual", null, 0),
288 // Assert.Throws
289 // Assert::ExpectException
290 (new Regex(@"(Assert)\.Throws", "$1::ExpectException", null, 0),
291 // $"Argument {argumentName} is null."
292 // ((std::string)"Argument ").append(argumentName).append(" is null.").data()
293 (new Regex(@"\$""(?<left>\\"""|["""\r\n])*{(?<expression>[_a-zA-Z0-9]+)}{(?<right>(\
    ↳ \""|["""\r\n])*}""",
    ↳ "((std::string)$\"${left}\").append(${expression}).append(\"${right}\").data()",
    ↳ null, 10),
294 // $"
295 // "
296 (new Regex(@"\$""", "\"", null, 0),
297 // Console.WriteLine(...)
298 // printf(...)
299 (new Regex(@"Console\.WriteLine\(\"([~""\r\n]+)\""\)", "printf(\"$1\\n\")", null, 0),
300 // TElement Root;
301 // TElement Root = 0;
302 (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),
303 // TreeElement _elements[N];
304 // TreeElement _elements[N] = { {0} };
305 (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} };", null, 0),
306 // auto path = new TElement[MaxPath];
307 // TElement path[MaxPath] = { {0} };
308 (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),
309 // private: static readonly ConcurrentBag<std::exception> _exceptionsBag = new
    ↳ ConcurrentBag<std::exception>();
310 // private: inline static std::mutex _exceptionsBag_mutex; \n\n private: inline
    ↳ static std::vector<std::exception> _exceptionsBag;
311 (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),
312 // public: static IReadOnlyCollection<std::exception> GetCollectedExceptions() {
    ↳ return _exceptionsBag; }
313 // public: static std::vector<std::exception> GetCollectedExceptions() { return
    ↳ std::vector<std::exception>(_exceptionsBag); }
314 (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),
315 // public: static event EventHandler<std::exception> ExceptionIgnored =
    ↳ OnExceptionIgnored; ... };
316 // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
    ↳ const std::exception&> ExceptionIgnored = OnExceptionIgnored; };

```

```

317 (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),
318 // Insert scope borders.
319 // class IgnoredExceptions { ... private: inline static std::vector<std::exception>
→ _exceptionsBag;
320 // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: inline static
→ std::vector<std::exception> _exceptionsBag;
321 (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}",
→ null, 0),
322 // Inside the scope of ~!_exceptionsBag!~ replace:
323 // _exceptionsBag.Add(exception);
324 // _exceptionsBag.push_back(exception);
325 (new Regex(@"(?<scope>/\s*(?<fieldName>[_a-zA-Z0-9]+)~\s*/)(?<separator>.\n)(?<befor_
→ e>((?!/\s*\k<fieldName>~\s*/)(.\n))*?)\k<fieldName>\.Add"),
→ "${scope}${separator}${before}${fieldName}.push_back", null, 10),
326 // Remove scope borders.
327 // /*~_exceptionsBag~*/
328 //
329 (new Regex(@"/\s*~[_a-zA-Z0-9]+\s*/"), "", null, 0),
330 // Insert scope borders.
331 // class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
332 // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: static std::mutex
→ _exceptionsBag_mutex;
333 (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),
334 // Inside the scope of ~!_exceptionsBag!~ replace:
335 // return std::vector<std::exception>(_exceptionsBag);
336 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
→ std::vector<std::exception>(_exceptionsBag);
337 (new Regex(@"(?<scope>/\s*(?<fieldName>[_a-zA-Z0-9]+)~\s*/)(?<separator>.\n)(?<befor_
→ 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}", null, 10),
338 // Inside the scope of ~!_exceptionsBag!~ replace:
339 // _exceptionsBag.Add(exception);
340 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
→ _exceptionsBag.Add(exception);
341 (new Regex(@"(?<scope>/\s*(?<fieldName>[_a-zA-Z0-9]+)~\s*/)(?<separator>.\n)(?<befor_
→ e>((?!/\s*\k<fieldName>~\s*/)(.\n))*?){(?<after>((?!lock_guard)\[~\];\n)]*\k<f_
→ ?\n(?<indent>[\t ]*)\k<fieldName>[~;\r\n]*;)",
→ "${scope}${separator}${before}" + Environment.NewLine +
→ "${indent}std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", null,
→ 10),
342 // Remove scope borders.
343 // /*~_exceptionsBag~*/
344 //
345 (new Regex(@"/\s*~[_a-zA-Z0-9]+\s*/"), "", null, 0),
346 // Insert scope borders.
347 // class IgnoredExceptions { ... public: static inline
→ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
→ ExceptionIgnored = OnExceptionIgnored;
348 // class IgnoredExceptions {/*~ExceptionIgnored~*/ ... public: static inline
→ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
→ ExceptionIgnored = OnExceptionIgnored;
349 (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),
350 // Inside the scope of ~!ExceptionIgnored!~ replace:
351 // ExceptionIgnored.Invoke(NULL, exception);
352 // ExceptionIgnored(NULL, exception);

```



```

353 (new Regex(@"(?<scope>/\~*(?<eventName>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↳ >((?<!/\\~*\k<eventName>~\*/)(.\|\\n))*?)\k<eventName>\.Invoke"),
    ↳ "${scope}${separator}${before}${eventName}", null, 10),
354 // Remove scope borders.
355 // /*~ExceptionIgnored~*/
356 //
357 (new Regex(@"/*~[a-zA-Z0-9]+~\*/"), "", null, 0),
358 // Insert scope borders.
359 // auto added = new StringBuilder();
360 // /*~sb~*/std::string added;
361 (new Regex(@"(auto|(System\.Text\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new
    ↳ (System\.Text\.)?StringBuilder\\(\\);", "/*~${variable}~*/std::string
    ↳ ${variable};", null, 0),
362 // static void Indent(StringBuilder sb, int level)
363 // static void Indent(/*~sb~*/StringBuilder sb, int level)
364 (new Regex(@"(?<start>, \\() (System\.Text\.)?StringBuilder
    ↳ (?<variable>[a-zA-Z0-9]+)(?<end>, \\))", "${start}/*~${variable}~*/std::string&
    ↳ ${variable}${end}", null, 0),
365 // Inside the scope of ~!added!~ replace:
366 // sb.ToString()
367 // sb.data()
368 (new Regex(@"(?<scope>/\~*(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↳ ((?<!/\\~*\k<variable>~\*/)(.\|\\n))*?)\k<variable>\.ToString\\(\\)",
    ↳ "${scope}${separator}${before}${variable}.data()", null, 10),
369 // sb.AppendLine(argument)
370 // sb.append(argument).append('\\n')
371 (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(${argument}).append(1, '\\n')",
    ↳ null, 10),
372 // sb.Append('\\t', level);
373 // sb.append(level, '\\t');
374 (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),
375 // sb.Append(argument)
376 // sb.append(argument)
377 (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(${argument})", null,
    ↳ 10),
378 // Remove scope borders.
379 // /*~sb~*/
380 //
381 (new Regex(@"/*~[a-zA-Z0-9]+~\*/"), "", null, 0),
382 // Insert scope borders.
383 // auto added = new HashSet<TElement>();
384 // ~!added!~std::unordered_set<TElement> added;
385 (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
    ↳ HashSet<?<element>[a-zA-Z0-9]+>\\(\\);",
    ↳ "~!${variable}!~std::unordered_set<${element}> ${variable};", null, 0),
386 // Inside the scope of ~!added!~ replace:
387 // added.Add(node)
388 // added.insert(node)
389 (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.\|\\n)(?<before>((?<
    ↳ !~!\k<variable>!~)(.\|\\n))*?)\k<variable>\.Add\\((?<argument>[a-zA-Z0-9]+)\\)",
    ↳ "${scope}${separator}${before}${variable}.insert(${argument})", null, 10),
390 // Inside the scope of ~!added!~ replace:
391 // added.Remove(node)
392 // added.erase(node)
393 (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(${argument})", null, 10),
394 // if (added.insert(node)) {
395 // if (!added.contains(node)) { added.insert(node);
396 (new Regex(@"if \\((?<variable>[a-zA-Z0-9]+)\\.insert\\((?<argument>[a-zA-Z0-9]+)\\)\\)(?
    ↳ <separator>[\\t ]*[\\r\\n]+)(?<indent>[\\t ]*){", "if
    ↳ (!${variable}).contains(${argument})${separator}${indent}{ " +
    ↳ Environment.NewLine + "${indent} ${variable}.insert(${argument});", null, 0),
397 // Remove scope borders.
398 // ~!added!~
399 //
400 (new Regex(@"~![a-zA-Z0-9]+!~"), "", null, 5),

```



```

401 // Insert scope borders.
402 // auto random = new System.Random(0);
403 // std::srand(0);
404 (new Regex(@"[a-zA-Z0-9\.] + ([a-zA-Z0-9]+) = new
    ↳ (System\.)?Random\((([a-zA-Z0-9]+)\);", "!$!~std::srand($3);", null, 0),
405 // Inside the scope of ~!random!~ replace:
406 // random.Next(1, N)
407 // (std::rand() % N) + 1
408 (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),
409 // Remove scope borders.
410 // ~!random!~
411 //
412 (new Regex(@"~![a-zA-Z0-9]+!~"), "", null, 5),
413 // Insert method body scope starts.
414 // void PrintNodes(TElement node, StringBuilder sb, int level) {
415 // void PrintNodes(TElement node, StringBuilder sb, int level) { /*method-start*/
416 (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*/${end}", null,
    ↳ 0),
417 // Insert method body scope ends.
418 // { /*method-start*/...}
419 // { /*method-start*/... /*method-end*/}
420 (new Regex(@"{ /\*method-start\*/(?<body>((?<bracket>\{) | (?<-bracket>\}) | [^\{\}])*)+ )
    ↳ \}"), "{ /*method-start*/${body} /*method-end*/", null,
    ↳ 0),
421 // Inside method bodies replace:
422 // GetFirst(
423 // this->GetFirst(
424 // (new Regex(@"(?<separator>(\|, | ([\W]) |return ))(?<!( ->| \*
    ↳ ))(?<method>(?!sizeof)[a-zA-Z0-9]+)\(((?! \) \{)"),
    ↳ "${separator}this->${method}(", null, 1),
425 (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),
426 // Remove scope borders.
427 // /*method-start*/
428 //
429 (new Regex(@" /\*method-(start|end)\*/"), "", null, 0),
430 // Insert scope borders.
431 // const std::exception& ex
432 // const std::exception& ex/*~ex~*/
433 (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?exception&
    ↳ (?<variable>[_a-zA-Z0-9]+)) (?<after>\W)",
    ↳ "${before}${variableDefinition}/*~${variable}~*/${after}", null, 0),
434 // Inside the scope of ~!ex!~ replace:
435 // ex.Message
436 // ex.what()
437 (new Regex(@"(?<scope> /\*~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↳ >((?<!( /\*~\k<variable>~\*/)(.\|\\n))*?)\k<variable>\.Message"),
    ↳ "${scope}${separator}${before}${variable}.what()", null, 10),
438 // Remove scope borders.
439 // /*~ex~*/
440 //
441 (new Regex(@" /\*~[_a-zA-Z0-9]+~\*/"), "", null, 0),
442 // throw new ArgumentNullException(argumentName, message);
443 // throw std::invalid_argument(((std::string)"Argument
    ↳ ").append(argumentName).append(" is null: ").append(message).append("."));
444 (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),
445 // throw new ArgumentException(message, argumentName);
446 // throw std::invalid_argument(((std::string)"Invalid
    ↳ ").append(argumentName).append(" argument: ").append(message).append("."));
447 (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),

```

```

448 // throw new NotSupportedException();
449 // throw std::logic_error("Not supported exception.");
450 (new Regex(@"throw new NotSupportedException\(\);", "throw std::logic_error(\"Not
    ↳ supported exception.\");", null, 0),
451 // throw new NotImplementedException();
452 // throw std::logic_error("Not implemented exception.");
453 (new Regex(@"throw new NotImplementedException\(\);", "throw std::logic_error(\"Not
    ↳ implemented exception.\");", null, 0),
454 }.Cast<ISubstitutionRule>().ToList();
455
456 public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
457 {
458     // ICounter<int, int> c1;
459     // ICounter<int, int>* c1;
460     (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^\r\n]+>)?)
    ↳ (?<variable>[_a-zA-Z0-9]+);", "${abstractType}* ${variable};", null, 0),
461     // (expression)
462     // expression
463     (new Regex(@"(\(|\)|)(([a-zA-Z0-9_\*:]+)\(|\)|\;|\)|)", "$1$2$3", null, 0),
464     // (method(expression))
465     // method(expression)
466     (new Regex(@"(?<firstSeparator>(\(|
    ↳ ))\((?<method>[a-zA-Z0-9_\->\*:]+)\((?<expression>((?<parenthesis>\(|(?<-parent
    ↳ hesis>\)|[a-zA-Z0-9_\->\*:]*)+)(?(parenthesis)(?!))\)\)(?<lastSeparator>(\(|
    ↳ \;|\)|))", "${firstSeparator}${method}(${expression})${lastSeparator}", null, 0),
467     // return ref _elements[node];
468     // return &elements[node];
469     (new Regex(@"return ref ([_a-zA-Z0-9]+)\([[_a-zA-Z0-9\*]+]\);", "return &$1[$2];",
    ↳ null, 0),
470     // null
471     // nullptr
472     (new Regex(@"(?<before>\r?\n[~""\r\n]*(""(\\"""| [~""\r\n])*""[~""\r\n]*)*)(<=\W)null
    ↳ (?<after>\W)", "${before}nullptr${after}", null,
    ↳ 10),
473     // default
474     // 0
475     (new Regex(@"(?<before>\r?\n[~""\r\n]*(""(\\"""| [~""\r\n])*""[~""\r\n]*)*)(<=\W)defa
    ↳ ult(?<after>\W)", "${before}0${after}", null,
    ↳ 10),
476     // object x
477     // void *x
478     (new Regex(@"(?<before>\r?\n[~""\r\n]*(""(\\"""| [~""\r\n])*""[~""\r\n]*)*)(<=\W)([O|
    ↳ o]bject|System\.Object) (?<after>\w)", "${before}void *${after}", null,
    ↳ 10),
479     // <object>
480     // <void*>
481     (new Regex(@"(?<before>\r?\n[~""\r\n]*(""(\\"""| [~""\r\n])*""[~""\r\n]*)*)(<=\W)(?!
    ↳ \w)([O|o]bject|System\.Object)(<after>\W)", "${before}void*${after}", null,
    ↳ 10),
482     // ArgumentNullException
483     // std::invalid_argument
484     (new Regex(@"(?<before>\r?\n[~""\r\n]*(""(\\"""| [~""\r\n])*""[~""\r\n]*)*)(<=\W)(Sys
    ↳ tem\.)?ArgumentNullException(<after>\W)",
    ↳ "${before}std::invalid_argument${after}", null, 10),
485     // #region Always
486     //
487     (new Regex(@"(~|\r?\n)[\t]*#(region|endregion)[~\r\n]*(\r?\n|$)", "", null, 0),
488     // // #define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
489     //
490     (new Regex(@"\\\/\[\t]*#define[\t]+[_a-zA-Z0-9]+[\t]*")", "", null, 0),
491     // #if USEARRAYPOOL\r\n#endif
492     //
493     (new Regex(@"#if [a-zA-Z0-9]+\s+endif)", "", null, 0),
494     // [Fact]
495     //
496     (new Regex(@"(?<firstNewLine>\r?\n\A)(?<indent>[\t
    ↳ ]+)\[[a-zA-Z0-9]+(\((?<expression>((?<parenthesis>\(|(?<-parent
    ↳ hesis>\)|[~""\r\n]*)+)(?(parenthesis)(?!))\)\)?\][\t]*(\r?\n\k<indent>?)")",
    ↳ "${firstNewLine}${indent}", null, 5),
497     // \n ... namespace
498     // namespace
499     (new Regex(@"(\\S[\r\n]{1,2})?[\r\n]+namespace)", "$1namespace", null, 0),
500     // \n ... class
501     // class
502     (new Regex(@"(\\S[\r\n]{1,2})?[\r\n]+class)", "$1class", null, 0),
503 }.Cast<ISubstitutionRule>().ToList();

```

```

504     public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
505         ↪ base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
506
507     public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
508 }
509 }

```

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

```

Index

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