

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<ISubstitutionRule> FirstStage = new List<SubstitutionRule>
13         {
14             // // ...
15             //
16             (new Regex(@"(\r?\n)?[ \t]++/.+"), "", 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]+$"), "", 0),
20             // {\n\n\n
21             // {
22             (new Regex(@"{\s+[\r\n]+") , "{" + Environment.NewLine, 0),
23             // Platform.Collections.Methods.Lists
24             // Platform::Collections::Methods::Lists
25             (new Regex(@"(namespace[^\r\n]+?)\.((~\r\n)+?)") , "$1::$2", 20),
26             // Comparer<TArgument>.Default.Compare(maximumArgument, minimumArgument) < 0
27             // maximumArgument < minimumArgument
28             (new Regex(@"Comparer<[^\n]+>\.Default\.Compare\\(s*(?<first>[^\n]+),s*(?<second>
29             >[^\n]+)\s*)\\s*(?<comparison>[<=>=?)\s*0") , "${first} ${comparison}
30             ${second}", 0),
31             // out TProduct
32             // TProduct
33             (new Regex(@"(?<before>( <| , ))(in|out)
34             > (?<typeParameter>[a-zA-Z0-9]+)(?<after>( >| , ))") ,
35             "${before}${typeParameter}${after}", 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}: ${before}", 0),
42             // public: static bool CollectExceptions { get; set; }
43             // public: inline static bool CollectExceptions;
44             (new Regex(@"(?<access>(private|protected|public): ) (?<before>(static )? [^\r\n]+
45             ) (?<name>[a-zA-Z0-9]+) { [^;]* (?<=\\W) get; [^;]* (?<=\\W) set; [^;]* }" ) ,
46             "${access}inline ${before}${name};" , 0),
47             // public abstract class
48             // class
49             (new Regex(@"((public|protected|private|internal|abstract|static)
50             )*(?<category>interface|class|struct)" , "${category}" , 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{" , 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)" , 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}>${whitespace}{" + Environment.NewLine + "
69             public:" , 0),
70             // template <typename TObject, TProperty, TValue>
71             // template <typename TObject, typename TProperty, TValue>
72             (new Regex(@"(?<before>template <(( , )?typename [a-zA-Z0-9]+)+,
73             > (?<typeParameter>[a-zA-Z0-9]+)(?<after>( , |>))" , "${before}typename
74             ${typeParameter}${after}" , 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}~*/$0", 0),
57 // Move all markers to the beginning of the file.
58 (new Regex(@"\A(?<before>[\r\n]+\r?\n(.|\n)+)(?<marker>\/\*~extensionMethod~(?<name>
    ↳ [a-zA-Z0-9]+)~\*/)", "${marker}${before}",
    ↳ 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}",
    ↳ 50),
62 // Remove markers
63 // /*~extensionMethod~BuildExceptionString~*/
64 //
65 (new Regex(@"\/\*~extensionMethod~[a-zA-Z0-9]+~\*/", "", 0),
66 // (this
67 // (
68 (new Regex(@"(this ", "(", 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} ${name};", 0),
72 // public: static readonly string ExceptionContentsSeparator = "---";
73 // public: inline static const char* ExceptionContentsSeparator = "---";
74 (new Regex(@"(?<access>(private|protected|public): )?static readonly string
    ↳ (?<name>[_a-zA-Z0-9_]+) = ""(?<string>\\\"|\\\"\\\r\n\\\"+)"";", "${access}inline
    ↳ static const char* ${name} = \"${string}\";", 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} = ${value};", 0),
78 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
    ↳ TArgument : class
79 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
80 (new Regex(@"(?<before> [a-zA-Z]+\\((([a-zA-Z *],+), |)))(?<type>[a-zA-Z]+)(?<after>(
    ↳ [a-zA-Z *,,]+)\\)) [ \r\n]+where \k<type> : class)", "${before}${type}*${after}",
    ↳ 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;", 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$3", 1),
87 // protected: readonly TreeElement[] _elements;
88 // protected: TreeElement _elements[N];
89 (new Regex(@"(?<access>(private|protected|public): )?readonly
    ↳ (?<type>[a-zA-Z<0-9]+)([ \r\n]+) (?<name>[_a-zA-Z0-9_]+);", "${access}${type}
    ↳ ${name}[N];", 0),
90 // protected: readonly TElement Zero;
91 // protected: TElement Zero;
92 (new Regex(@"(?<access>(private|protected|public): )?readonly
    ↳ (?<type>[a-zA-Z<0-9]+) (?<name>[_a-zA-Z0-9_]+);", "${access}${type} ${name};",
    ↳ 0),
93 // internal
94 //
95 (new Regex(@"(\\W)internal\s+)", "$1", 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; }", 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; }", 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 $10; }", 0),
105 // () => Integer<TElement>.Zero,
106 // () { return Integer<TElement>.Zero; },
107 (new Regex(@"(\\)\\s+=>\\s+(?<expression>[^(,;\\r\\n]+(\\(((?<parenthesis>\\(|(?<-parent_
   ↳ hesis>\\)|[^(,;\\r\\n]*?)*?\\))?[^(,;\\r\\n]*)(?<after>,\\|);)"), "$() { return
   ↳ ${expression}; }${after}", 0),
108 // => Integer<TElement>.Zero;
109 // { return Integer<TElement>.Zero; }
110 (new Regex(@"\\)\\s+=>\\s+([~;\\r\\n]+?);"), ") { return $1; }", 0),
111 // () { return avlTree.Count; }
112 // [&]() -> auto { return avlTree.Count; }
113 (new Regex(@"(?<before>, \\(|\\)\\(\\) { return (?<expression>[~;\\r\\n]+); }"),
   ↳ "${before}[&]() -> auto { return ${expression}; }", 0),
114 // Count => GetSizeOrZero(Root);
115 // GetCount() { return GetSizeOrZero(Root); }
116 (new Regex(@"(\\W)([A-Z][a-zA-Z]+)\\s+=>\\s+([~;\\r\\n]+);"), "$1Get$2() { return $3; }",
   ↳ 0),
117 // ArgumentInRange(const char* message) { const char* messageBuilder() { return
   ↳ message; }
118 // ArgumentInRange(const char* message) { auto messageBuilder = [&]() -> const char*
   ↳ { return message; };
119 (new Regex(@"(?<before>\\W[_a-zA-Z0-9]+\\(((\\^\\)\\n)*\\)[\\s\\n]*{[\\s\\n]*([~{}]|\\n)*?(\\r?\\n)
   ↳ ?[ \\t]*)(?<returnType>[_a-zA-Z0-9*:] +[_a-zA-Z0-9*:]*)
   ↳ (?<methodName>[_a-zA-Z0-9]+)\\(((?<arguments>[~\\)\\n]*))\\s*{(?<body>([~}]|\\n)+?)}"
   ↳ ), "${before}auto ${methodName} = [&]() -> ${returnType} {${body}};",
   ↳ 10),
120 // Func<TElement> treeCount
121 // std::function<TElement()> treeCount
122 (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", 0),
123 // Action<TElement> free
124 // std::function<void(TElement)> free
125 (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<void($1)> $2",
   ↳ 0),
126 // Predicate<TArgument> predicate
127 // std::function<bool(TArgument)> predicate
128 (new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<bool($1)>
   ↳ $2", 0),
129 // var
130 // auto
131 (new Regex(@"(\\W)var(\\W)"), "$1auto$2", 0),
132 // unchecked
133 //
134 (new Regex(@"[\\r\\n]{2}\\s*?unchecked\\s*?$"), "", 0),
135 // throw new InvalidOperationException
136 // throw std::runtime_error
137 (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
   ↳ std::runtime_error", 0),
138 // void RaiseExceptionIgnoredEvent(Exception exception)
139 // void RaiseExceptionIgnoredEvent(const std::exception& exception)
140 (new Regex(@"(\\(| ) (System\\.Exception|Exception) (|\\))"), "$1const
   ↳ std::exception&$3", 0),
141 // EventHandler<Exception>
142 // EventHandler<std::exception>
143 (new Regex(@"(\\W) (System\\.Exception|Exception) (\\W)"), "$1std::exception$3", 0),
144 // override void PrintNode(TElement node, StringBuilder sb, int level)
145 // void PrintNode(TElement node, StringBuilder sb, int level) override
146 (new Regex(@"override ([a-zA-Z0-9 \\*+]+)\\(((\\^\\)\\r\\n)+?\\)\\)"), "$1$2 override", 0),
147 // return (range.Minimum, range.Maximum)
148 // return {range.Minimum, range.Maximum}
149 (new Regex(@"(?<before>return\\s*)\\(((?<values>[~\\)\\n]+)\\)\\(?!\\(|(?<after>\\W)\\)"),
   ↳ "${before}${values}${after}", 0),
150 // string
151 // const char*

```

```

152 (new Regex(@"(\W)string(\W)"), "$1const char*$2", 0),
153 // System.ValueTuple
154 // std::tuple
155 (new Regex(@"(?<before>\W)(System\.)?ValueTuple(?:\s*)(?<after>\W)"),
156     ↳ "${before}std::tuple${after}", 0),
157 // sbyte
158 // std::int8_t
159 (new Regex(@"(?<before>\W)((System\.)?SB|sbyte(?:\s*)(?<after>\W)"),
160     ↳ "${before}std::int8_t${after}", 0),
161 // sbyte.MinValue
162 // INT8_MIN
163 (new Regex(@"(?<before>\W)std::int8_t\.MinValue(?<after>\W)"),
164     ↳ "${before}INT8_MIN${after}", 0),
165 // sbyte.MaxValue
166 // INT8_MAX
167 (new Regex(@"(?<before>\W)std::int8_t\.MaxValue(?<after>\W)"),
168     ↳ "${before}INT8_MAX${after}", 0),
169 // short
170 // std::int16_t
171 (new Regex(@"(?<before>\W)((System\.)?Int16|short(?:\s*)(?<after>\W)"),
172     ↳ "${before}std::int16_t${after}", 0),
173 // short.MinValue
174 // INT16_MIN
175 (new Regex(@"(?<before>\W)std::int16_t\.MinValue(?<after>\W)"),
176     ↳ "${before}INT16_MIN${after}", 0),
177 // short.MaxValue
178 // INT16_MAX
179 (new Regex(@"(?<before>\W)std::int16_t\.MaxValue(?<after>\W)"),
180     ↳ "${before}INT16_MAX${after}", 0),
181 // int
182 // std::int32_t
183 (new Regex(@"(?<before>\W)((System\.)?I|i)nt(32)?(?:\s*)(?<after>\W)"),
184     ↳ "${before}std::int32_t${after}", 0),
185 // int.MinValue
186 // INT32_MIN
187 (new Regex(@"(?<before>\W)std::int32_t\.MinValue(?<after>\W)"),
188     ↳ "${before}INT32_MIN${after}", 0),
189 // int.MaxValue
190 // INT32_MAX
191 (new Regex(@"(?<before>\W)std::int32_t\.MaxValue(?<after>\W)"),
192     ↳ "${before}INT32_MAX${after}", 0),
193 // long
194 // std::int64_t
195 (new Regex(@"(?<before>\W)((System\.)?Int64|long(?:\s*)(?<after>\W)"),
196     ↳ "${before}std::int64_t${after}", 0),
197 // long.MinValue
198 // INT64_MIN
199 (new Regex(@"(?<before>\W)std::int64_t\.MinValue(?<after>\W)"),
200     ↳ "${before}INT64_MIN${after}", 0),
201 // long.MaxValue
202 // INT64_MAX
203 (new Regex(@"(?<before>\W)std::int64_t\.MaxValue(?<after>\W)"),
204     ↳ "${before}INT64_MAX${after}", 0),
205 // byte
206 // std::uint8_t
207 (new Regex(@"(?<before>\W)((System\.)?Byte|byte(?:\s*)(?<after>\W)"),
208     ↳ "${before}std::uint8_t${after}", 0),
209 // byte.MinValue
210 // (std::uint8_t)0
211 (new Regex(@"(?<before>\W)std::uint8_t\.MinValue(?<after>\W)"),
212     ↳ "${before}(std::uint8_t)0${after}", 0),
213 // byte.MaxValue
214 // UINT8_MAX
215 (new Regex(@"(?<before>\W)std::uint8_t\.MaxValue(?<after>\W)"),
216     ↳ "${before}UINT8_MAX${after}", 0),
217 // ushort
218 // std::uint16_t
219 (new Regex(@"(?<before>\W)((System\.)?UInt16|ushort(?:\s*)(?<after>\W)"),
220     ↳ "${before}std::uint16_t${after}", 0),
221 // ushort.MinValue
222 // (std::uint16_t)0
223 (new Regex(@"(?<before>\W)std::uint16_t\.MinValue(?<after>\W)"),
224     ↳ "${before}(std::uint16_t)0${after}", 0),
225 // ushort.MaxValue
226 // UINT16_MAX

```

```

(new Regex(@"(?<before>\W)std::uint16_t\.MaxValue(?<after>\W)",
    ↳ "${before}UINT16_MAX${after}", 0),
// uint
// std::uint32_t
(new Regex(@"(?<before>\W)((System\.)?UI|ui)nt(32)?(?!\s*)(?<after>\W)",
    ↳ "${before}std::uint32_t${after}", 0),
// uint.MinValue
// (std::uint32_t)0
(new Regex(@"(?<before>\W)std::uint32_t\.MinValue(?<after>\W)",
    ↳ "${before}(std::uint32_t)0${after}", 0),
// uint.MaxValue
// UINT32_MAX
(new Regex(@"(?<before>\W)std::uint32_t\.MaxValue(?<after>\W)",
    ↳ "${before}UINT32_MAX${after}", 0),
// ulong
// std::uint64_t
(new Regex(@"(?<before>\W)((System\.)?UInt64|ulong)(?!\\s*)(?<after>\W)",
    ↳ "${before}std::uint64_t${after}", 0),
// ulong.MinValue
// (std::uint64_t)0
(new Regex(@"(?<before>\W)std::uint64_t\.MinValue(?<after>\W)",
    ↳ "${before}(std::uint64_t)0${after}", 0),
// ulong.MaxValue
// UINT64_MAX
(new Regex(@"(?<before>\W)std::uint64_t\.MaxValue(?<after>\W)",
    ↳ "${before}UINT64_MAX${after}", 0),
// char*[] args
// char* args[]
(new Regex(@"([_a-zA-Z0-9:\*]?)\\[\\] ([a-zA-Z0-9]+)", "$1 $2[]", 0),
// @object
// object
(new Regex(@"@([_a-zA-Z0-9]+)", "$1", 0),
// using Platform.Numbers;
//
(new Regex(@"([\\r\\n]{2}|~)\\s*?using [\\_a-zA-Z0-9]+;\\s*?$)", "", 0),
// struct TreeElement { }
// struct TreeElement { };
(new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\\s+){([\\sa-zA-Z0-9;:_]+?)}([^;])", "$1
    ↳ $2$3{$4};$5", 0),
// class Program { }
// class Program { };
(new Regex(@"(struct|class) ([a-zA-Z0-9]+)[^\\r\\n]*([\\r\\n]+(?<indentLevel>[\\t
    ↳ ]*)?)\\{([\\S\\s]+?[\\r\\n]+k<indentLevel>)\\}([^;]|$)", "$1 $2$3{$4};$5", 0),
// class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
// class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
(new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)", "class $1 : public $2", 0),
// class IProperty : ISetter<TValue, TObjcet>, IProvider<TValue, TObjcet>
// class IProperty : public ISetter<TValue, TObjcet>, IProvider<TValue, TObjcet>
(new Regex(@"(?<before>class [a-zA-Z0-9]+ : ((public [a-zA-Z0-9]+(<[a-zA-Z0-9
    ↳ ,]+>)?, )+)?(?<inheritedType>(?!public) [a-zA-Z0-9]+(<[a-zA-Z0-9
    ↳ ,]+>)?(?<after>(, [a-zA-Z0-9]+(?!>)[ \\r\\n]+)))", "${before}public
    ↳ ${inheritedType}${after}", 10),
// Insert scope borders.
// ref TElement root
// ~!root!~ref TElement root
(new Regex(@"(?<definition>(?!<= |\\() (ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<ref>
    ↳ (?<variable>[a-zA-Z0-9]+)(?=\\)|, | =))", "~!${variable}!~${definition}", 0),
// Inside the scope of ~!root!~ replace:
// root
// *root
(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),
// Remove scope borders.
// ~!root!~
//
(new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~", "", 5),
// ref auto root = ref
// ref auto root =
(new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\\W)", "$1* $2 =$3", 0),
// *root = ref left;
// root = left;
(new Regex(@"\\*( [a-zA-Z0-9]+ ) = ref ([a-zA-Z0-9]+)(\\W)", "$1 = $2$3", 0),
// (ref left)
// (left)

```

```

269 (new Regex(@"\ref ([a-zA-Z0-9]+)\(|\(|)", "$1$2", 0),
270 // ref TElement
271 // TElement*
272 (new Regex(@"\(|\ref ([a-zA-Z0-9]+)", "$1$2*", 0),
273 // ref sizeBalancedTree.Root
274 // &sizeBalancedTree->Root
275 (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)", "&$1->$2", 0),
276 // ref GetElement(node).Right
277 // &GetElement(node)->Right
278 (new Regex(@"ref ([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+)\)\)\.([a-zA-Z0-9]+)",
    ↳ "&$1($2)->$3", 0),
279 // GetElement(node).Right
280 // GetElement(node)->Right
281 (new Regex(@"([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+)\)\)\.([a-zA-Z0-9]+)", "$1($2)->$3", 0),
282 // [Fact]\npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
283 // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
284 (new Regex(@"\Fact\[\s\n]+(public: )?(static )?void ([a-zA-Z0-9]+)\(|\(|)", "public:
    ↳ TEST_METHOD($3)", 0),
285 // class TreesTests
286 // TEST_CLASS(TreesTests)
287 (new Regex(@"class ([a-zA-Z0-9]+)Tests)", "TEST_CLASS($1)", 0),
288 // Assert.Equal
289 // Assert::AreEqual
290 (new Regex(@"(Assert)\.Equal)", "$1::AreEqual", 0),
291 // Assert.Throws
292 // Assert::ExpectException
293 (new Regex(@"(Assert)\.Throws)", "$1::ExpectException", 0),
294 // $"Argument {argumentName} is null."
295 // ((std::string)"Argument ").append(argumentName).append(" is null.").data()
296 (new Regex(@"\$"(?<left>\(|\(|\r\n|[\^~"\r\n])*){(?<expression>[_a-zA-Z0-9]+)}{(?<right>\(|
    ↳ \)|\r\n|[\^~"\r\n])*)",
    ↳ "((std::string)$\"${left}\").append(${expression}).append(\"${right}\").data()",
    ↳ 10),
297 // $"
298 // "
299 (new Regex(@"\$"""), "\"", 0),
300 // Console.WriteLine("...")
301 // printf("...\n")
302 (new Regex(@"Console\.WriteLine\(\"([~""\r\n]+)\""\""), "printf(\"$1\\n\\n\"", 0),
303 // TEElement Root;
304 // TEElement Root = 0;
305 (new Regex(@"(\r?\n[\t ]+)(private|protected|public)?(:
    ↳ )?([a-zA-Z0-9:_]+(?<return>)) ([a-zA-Z0-9]+);", "$1$2$3$4 $5 = 0;", 0),
306 // TreeElement _elements[N];
307 // TreeElement _elements[N] = { {0} };
308 (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),
309 // auto path = new TEElement[MaxPath];
310 // TEelement path[MaxPath] = { {0} };
311 (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),
312 // private: static readonly ConcurrentBag<std::exception> _exceptionsBag = new
    ↳ ConcurrentBag<std::exception>();
313 // private: inline static std::mutex _exceptionsBag_mutex; \n\n private: inline
    ↳ static std::vector<std::exception> _exceptionsBag;
314 (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};", 0),
315 // public: static IReadOnlyCollection<std::exception> GetCollectedExceptions() {
    ↳ return _exceptionsBag; }
316 // public: static std::vector<std::exception> GetCollectedExceptions() { return
    ↳ std::vector<std::exception>(_exceptionsBag); }
317 (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),
318 // public: static event EventHandler<std::exception> ExceptionIgnored =
    ↳ OnExceptionIgnored; ... };
319 // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
    ↳ const std::exception&> ExceptionIgnored = OnExceptionIgnored; };

```

```

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

```



```

356 (new Regex(@"(?<scope>/\~(?<eventName>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ >((?<!/\\~\k<eventName>~\*/)(.\\n))*?)\k<eventName>\\.Invoke"),
    ↪ "${scope}${separator}${before}${eventName}", 10),
357 // Remove scope borders.
358 // /*~ExceptionIgnored~/
359 //
360 (new Regex(@"/*~[a-zA-Z0-9]+~\*/"), "", 0),
361 // Insert scope borders.
362 // auto added = new StringBuilder();
363 // /*~sb~/std::string added;
364 (new Regex(@"(auto|(System\\.Text\\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new
    ↪ (System\\.Text\\.)?StringBuilder\\(\\);", "/*~${variable}~/std::string
    ↪ ${variable};", 0),
365 // static void Indent(StringBuilder sb, int level)
366 // static void Indent(/*~sb~/StringBuilder sb, int level)
367 (new Regex(@"(?<start>, \\() (System\\.Text\\.)?StringBuilder
    ↪ (?<variable>[a-zA-Z0-9]+)(?<end>, \\))", "${start}/*~${variable}~/std::string&
    ↪ ${variable}${end}", 0),
368 // Inside the scope of ~!added!~ replace:
369 // sb.ToString()
370 // sb.data()
371 (new Regex(@"(?<scope>/\~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ ((?<!/\\~\k<variable>~\*/)(.\\n))*?)\k<variable>\\.ToString\\(\\)",
    ↪ "${scope}${separator}${before}${variable}.data()", 10),
372 // sb.AppendLine(argument)
373 // sb.append(argument).append('\\n')
374 (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(${argument}).append(1, '\\n')",
    ↪ 10),
375 // sb.Append('\\t', level);
376 // sb.append(level, '\\t');
377 (new Regex(@"(?<scope>/\~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ ((?<!/\\~\k<variable>~\*/)(.\\n))*?)\k<variable>\\.Append\\('( ?<character>[\\r\\n]
    ↪ +)\\', (?<count>[\\], \\r\\n)+\\)",
    ↪ "${scope}${separator}${before}${variable}.append(${count}, '${character}'))", 10),
378 // sb.Append(argument)
379 // sb.append(argument)
380 (new Regex(@"(?<scope>/\~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ ((?<!/\\~\k<variable>~\*/)(.\\n))*?)\k<variable>\\.Append\\((?<argument>[\\], \\r\\n]
    ↪ +)\\)", "${scope}${separator}${before}${variable}.append(${argument})",
    ↪ 10),
381 // Remove scope borders.
382 // /*~sb~/
383 //
384 (new Regex(@"/*~[a-zA-Z0-9]+~\*/"), "", 0),
385 // Insert scope borders.
386 // auto added = new HashSet<TElement>();
387 // ~!added!~std::unordered_set<TElement> added;
388 (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
    ↪ HashSet<(?<element>[a-zA-Z0-9]+)>\\(\\);",
    ↪ "~!${variable}!~std::unordered_set<${element}> ${variable};", 0),
389 // Inside the scope of ~!added!~ replace:
390 // added.Add(node)
391 // added.insert(node)
392 (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.\|\\n)(?<before>((?<
    ↪ !~!\k<variable>!~)(.\\n))*?)\k<variable>\\.Add\\((?<argument>[a-zA-Z0-9]+)\\)",
    ↪ "${scope}${separator}${before}${variable}.insert(${argument})", 10),
393 // Inside the scope of ~!added!~ replace:
394 // added.Remove(node)
395 // added.erase(node)
396 (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(${argument})", 10),
397 // if (added.insert(node)) {
398 // if (!added.contains(node)) { added.insert(node);
399 (new Regex(@"if \\((?<variable>[a-zA-Z0-9]+)\\.insert\\((?<argument>[a-zA-Z0-9]+)\\)\\)(?
    ↪ <separator>[\\t ]*[\\r\\n]+)(?<indent>[\\t ]*){", "if
    ↪ (!${variable}.contains(${argument})) ${separator}${indent}{ " +
    ↪ Environment.NewLine + "${indent}    ${variable}.insert(${argument});", 0),
400 // Remove scope borders.
401 // ~!added!~
402 //
403 (new Regex(@"~![a-zA-Z0-9]+!~"), "", 5),
404 // Insert scope borders.

```



```

405 // auto random = new System.Random(0);
406 // std::srand(0);
407 (new Regex(@"[a-zA-Z0-9\.] + ([a-zA-Z0-9]+) = new
→ (System\.)?Random\((([a-zA-Z0-9]+)\);", "!$1!~std::srand($3);", 0),
408 // Inside the scope of ~!random!~ replace:
409 // random.Next(1, N)
410 // (std::rand() % N) + 1
411 (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),
412 // Remove scope borders.
413 // ~!random!~
414 //
415 (new Regex(@"~![a-zA-Z0-9]+!~"), "", 5),
416 // Insert method body scope starts.
417 // void PrintNodes(TElement node, StringBuilder sb, int level) {
418 // void PrintNodes(TElement node, StringBuilder sb, int level) { /*method-start*/
419 (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),
420 // Insert method body scope ends.
421 // { /*method-start*/ ... }
422 // { /*method-start*/ ... /*method-end*/ }
423 (new Regex(@"\{ /\*method-start\*/ (?<body> ((?<bracket>\{) | (?<-bracket>\}) | [^\{\}]*) )+ )
→ \}", "{ /*method-start*/ ${body} /*method-end*/ }",
→ 0),
424 // Inside method bodies replace:
425 // GetFirst(
426 // this->GetFirst(
427 (new Regex(@"(?<separator>(\(| |([\\W]) |return ))(?<!(->|\\*
→ ))(?<method>(?!sizeof)[a-zA-Z0-9]+\)((?!\\) \{)",
→ "${separator}this->${method}(", 1),
428 (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}", 100),
429 // Remove scope borders.
430 // /*method-start*/
431 //
432 (new Regex(@" /\*method-(start|end)\*/"), "", 0),
433 // Insert scope borders.
434 // const std::exception& ex
435 // const std::exception& ex/*~ex~*/
436 (new Regex(@"(?<before>\\(| )(?<variableDefinition>(const)?(std::)?exception&
→ (?<variable>[_a-zA-Z0-9]+)) (?<after>\\W)",
→ "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
437 // Inside the scope of ~!ex!~ replace:
438 // ex.Message
439 // ex.what()
440 (new Regex(@"(?<scope> /\*~ (?<variable>[_a-zA-Z0-9]+) ~\*/ ) (?<separator>.\|\\n) (?<before>
→ >((?<!( /\*~<k<variable>~\*/ ) (.\|\\n))*?)\k<variable>\.Message)",
→ "${scope}${separator}${before}${variable}.what()", 10),
441 // Remove scope borders.
442 // /*~ex~*/
443 //
444 (new Regex(@" /\*~ [_a-zA-Z0-9]+ ~\*/"), "", 0),
445 // throw new ArgumentNullException(argumentName, message);
446 // throw std::invalid_argument(((std::string)"Argument
→ ").append(argumentName).append(" is null: ").append(message).append("."));
447 (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(\".\");", 0),
448 // throw new ArgumentException(message, argumentName);
449 // throw std::invalid_argument(((std::string)"Invalid
→ ").append(argumentName).append(" argument: ").append(message).append("."));
450 (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(\".\");", 0),

```

```

451 // throw new ArgumentOutOfRangeException(argumentName, argumentValue,
452     ↳ messageBuilder());
453 // throw std::invalid_argument(((std::string)"Value
454     ↳ [").append(std::to_string(argumentValue)).append("] of argument
455     ↳ [").append(argumentName).append("] is out of range:
456     ↳ ").append(messageBuilder()).append(".");
457 (new Regex(@"throw new ArgumentOutOfRangeException\((?<argument>[a-zA-Z]*[Aa]rgument
458     ↳ [a-zA-Z]*([Nn]ame[a-zA-Z]*)?)
459     ↳ (?<argumentValue>[a-zA-Z]*[Aa]rgument[a-zA-Z]*([Vv]alue[a-zA-Z]*)?)
460     ↳ (?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\(\)\)?\)\);)", "throw
461     ↳ std::invalid_argument(((std::string)"Value
462     ↳ [").append(std::to_string(${argumentValue}).append("\] of argument
463     ↳ [").append(${argument}).append("\] is out of range:
464     ↳ \").append(${message}).append("\.\");", 0),
465 // throw new NotSupportedException();
466 // throw std::logic_error("Not supported exception.");
467 (new Regex(@"throw new NotSupportedException\(\);", "throw std::logic_error(\"Not
468     ↳ supported exception.\");", 0),
469 // throw new NotImplementedException();
470 // throw std::logic_error("Not implemented exception.");
471 (new Regex(@"throw new NotImplementedException\(\);", "throw std::logic_error(\"Not
472     ↳ implemented exception.\");", 0),
473 }.Cast<ISubstitutionRule>().ToList();
474
475 public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
476 {
477     // ICounter<int, int> c1;
478     // ICounter<int, int>* c1;
479     (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[~>\r\n]+>?)
480     ↳ (?<variable>[_a-zA-Z0-9]+);", "${abstractType}* ${variable};", 0),
481     // (expression)
482     // expression
483     (new Regex(@"(\(|\)|)(([a-zA-Z0-9_]*:)+)\(|\)|;|\) )", "$1$2$3", 0),
484     // (method(expression))
485     // method(expression)
486     (new Regex(@"(?<firstSeparator>(\(|\)|
487     ↳ ))\(((?<method>[a-zA-Z0-9_]*->*:)+)\(((?<expression>((?<parenthesis>\(|\)|(?<-parent
488     ↳ hesis>)|[a-zA-Z0-9_]*->*:)+)(?(parenthesis)(?!))\)\)\((?<lastSeparator>(\(|\)|;|\)
489     ↳ );|\) )", "${firstSeparator}${method}(${expression})${lastSeparator}", 0),
490     // return ref _elements[node];
491     // return &elements[node];
492     (new Regex(@"return ref ([_a-zA-Z0-9]+)\([([_a-zA-Z0-9_]*+)\];", "return &$1[$2];",
493     ↳ 0),
494     // null
495     // nullptr
496     (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"""| [~""\r\n])*""[~""\r\n])*)(?<=\\W)null
497     ↳ (?<after>\\W)", "${before}nullptr${after}",
498     ↳ 10),
499     // default
500     // 0
501     (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"""| [~""\r\n])*""[~""\r\n])*)(?<=\\W)defa
502     ↳ ult(?<after>\\W)", "${before}0${after}",
503     ↳ 10),
504     // object x
505     // void *x
506     (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"""| [~""\r\n])*""[~""\r\n])*)(?<=\\W)([O|
507     ↳ o]bject|System\\.Object) (?<after>\\w)", "${before}void *${after}",
508     ↳ 10),
509     // <object>
510     // <void*>
511     (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"""| [~""\r\n])*""[~""\r\n])*)(?<=\\W)(?!
512     ↳ \\w)([O|o]bject|System\\.Object) (?<after>\\W)", "${before}void*${after}",
513     ↳ 10),
514     // ArgumentNullException
515     // std::invalid_argument
516     (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"""| [~""\r\n])*""[~""\r\n])*)(?<=\\W)(Sys
517     ↳ tem\\.)?ArgumentNullException(?<after>\\W)",
518     ↳ "${before}std::invalid_argument${after}", 10),
519     // #region Always
520     //
521     (new Regex(@"(^\r?\n)[ \t]*#(region|endregion)[^\r\n]*(\r?\n|$)", "", 0),
522     // // #define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
523     //
524     (new Regex(@"\\[/[ \t]*#define[ \t]+[_a-zA-Z0-9]+[ \t]*")", "", 0),
525     // #if USEARRAYPOOL\r\n#endif
526     //

```

```

499         (new Regex(@"#if [a-zA-Z0-9]+\s+#endif"), "", 0),
500         // [Fact]
501         //
502         (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),
503         // \n ... namespace
504         // namespace
505         (new Regex(@"\S[\r\n]{1,2}?[\r\n]+namespace"), "$1namespace", 0),
506         // \n ... class
507         // class
508         (new Regex(@"\S[\r\n]{1,2}?[\r\n]+class"), "$1class", 0),
509     }.Cast<ISubstitutionRule>().ToList();
510
511     public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
→      base(FirstStage.Concat(extraRules).Concat>LastStage).ToList()) { }
512
513     public CSharpToCppTransformer() : base(FirstStage.Concat>LastStage).ToList()) { }
514 }
515 }

```

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

```

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

```

Index

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