

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),
108 // => Integer<TElement>.Zero;
109 // { return Integer<TElement>.Zero; }
110 (new Regex(@"\\)\\s+=>\\s+([\\^;\\r\\n]+?);"), ") { return $1; }", null, 0),
111 // () { return avlTree.Count; }
112 // [&]() -> auto { return avlTree.Count; }
113 (new Regex(@"(?<before>, |\\()\\(\\) { return (?<expression>[\\^;\\r\\n]+); }"),
   ↳ "${before}[&]() -> auto { return ${expression}; }", null, 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 $3; }",
   ↳ null, 0),
117 // ArgumentInRange(const char* message) { const char* messageBuilder() { return
   ↳ message; }
118 // ArgumentInRange(const char* message) { auto messageBuilder = [&]() -> const char*
   ↳ { return message; };
119 (new Regex(@"(?<before>\\W[_a-zA-Z0-9]+\\((\\^\\)\\n)*\\)[\\s\\n]*{[\\s\\n]*([\\^{}]|\\n)*?(\\r?\\n)
   ↳ ?[ \\t]*}?(?<returnType>[_a-zA-Z0-9*:] +[_a-zA-Z0-9*:]*)
   ↳ (?<methodName>[_a-zA-Z0-9]+)\\(((?<arguments>[\\^\\)\\n]*\\)\\s*{(?<body>([\\^{}]|\\n)+?)}"
   ↳ ), "${before}auto ${methodName} = [&]() -> ${returnType} {${body}};", null,
   ↳ 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", null,
   ↳ 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",
   ↳ null, 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", null, 0),
129 // var
130 // auto
131 (new Regex(@"(\\W)var(\\W)"), "$1auto$2", null, 0),
132 // unchecked
133 //
134 (new Regex(@"[\\r\\n]{2}\\s*?unchecked\\s*?$"), "", null, 0),
135 // throw new InvalidOperationException
136 // throw std::runtime_error
137 (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
   ↳ std::runtime_error", null, 0),
138 // void RaiseExceptionIgnoredEvent(Exception exception)
139 // void RaiseExceptionIgnoredEvent(const std::exception& exception)
140 (new Regex(@"(\\(| ) (System\\.Exception|Exception) (|\\))"), "$1const
   ↳ std::exception&$3", null, 0),
141 // EventHandler<Exception>
142 // EventHandler<std::exception>
143 (new Regex(@"(\\W) (System\\.Exception|Exception) (\\W)"), "$1std::exception$3", null, 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$2 override", null,
   ↳ 0),
147 // return (range.Minimum, range.Maximum)
148 // return {range.Minimum, range.Maximum}
149 (new Regex(@"(?<before>return\\s*)\\(((?<values>[\\^\\)\\n]+)\\)(?!\\() (?<after>\\W)"),
   ↳ "${before}${values}${after}", null, 0),

```

```

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

```

```

208 // UINT16_MAX
209 (new Regex(@"(?<before>\W)std::uint16_t\.MaxValue(?<after>\W)"),
    ↳ "${before}UINT16_MAX${after}", null, 0),
210 // uint
211 // std::uint32_t
212 (new Regex(@"(?<before>\W)((System\.)?UI|ui)nt(32)?(?!\s*)(?<after>\W)"),
    ↳ "${before}std::uint32_t${after}", null, 0),
213 // uint.MinValue
214 // (std::uint32_t)0
215 (new Regex(@"(?<before>\W)std::uint32_t\.MinValue(?<after>\W)"),
    ↳ "${before}(std::uint32_t)0${after}", null, 0),
216 // uint.MaxValue
217 // UINT32_MAX
218 (new Regex(@"(?<before>\W)std::uint32_t\.MaxValue(?<after>\W)"),
    ↳ "${before}UINT32_MAX${after}", null, 0),
219 // ulong
220 // std::uint64_t
221 (new Regex(@"(?<before>\W)((System\.)?UInt64|ulong)(?!\\s*)(?<after>\W)"),
    ↳ "${before}std::uint64_t${after}", null, 0),
222 // ulong.MinValue
223 // (std::uint64_t)0
224 (new Regex(@"(?<before>\W)std::uint64_t\.MinValue(?<after>\W)"),
    ↳ "${before}(std::uint64_t)0${after}", null, 0),
225 // ulong.MaxValue
226 // UINT64_MAX
227 (new Regex(@"(?<before>\W)std::uint64_t\.MaxValue(?<after>\W)"),
    ↳ "${before}UINT64_MAX${after}", null, 0),
228 // char*[] args
229 // char* args[]
230 (new Regex(@"([_a-zA-Z0-9:~*\?]\\\[\] ([a-zA-Z0-9]+)"), "$1 $2[]", null, 0),
231 // @object
232 // object
233 (new Regex(@"@([_a-zA-Z0-9]+)"), "$1", null, 0),
234 // using Platform.Numbers;
235 //
236 (new Regex(@"([\r\n]{2}|~)\s*?using [\\a-zA-Z0-9]+;\s*?$"), "", null, 0),
237 // struct TreeElement { }
238 // struct TreeElement { };
239 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([~;])"), "$1
    ↳ $2$3{$4};$5", null, 0),
240 // class Program { }
241 // class Program { };
242 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)[~\r\n*](\r\n+)?(<indentLevel>[\\t
    ↳ ]*)?\\{([\\S\\s]?+[\r\n]+\k<indentLevel>)\}([~;]|$)", "$1 $2$3{$4};$5", null, 0),
243 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
244 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
245 (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)", "class $1 : public $2", null,
    ↳ 0),
246 // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
247 // class IProperty : public ISetter<TValue, TObject>, IProvider<TValue, TObject>
248 (new Regex(@"(?<before>class [a-zA-Z0-9]+ : ((public [a-zA-Z0-9]+(<[a-zA-Z0-9
    ↳ ,]+>)?, )+)?(?<inheritedType>(?!public) [a-zA-Z0-9]+(<[a-zA-Z0-9
    ↳ ,]+>)?(?<after>([a-zA-Z0-9]+(?!>)|[ \r\n]+))", "${before}public
    ↳ ${inheritedType}${after}", null, 10),
249 // Insert scope borders.
250 // ref TElement root
251 // ~!root!~ref TElement root
252 (new Regex(@"(?<definition>(?!\\() (ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!ref))
    ↳ (?<variable>[a-zA-Z0-9]+)(?=\\)|, | =))", "~!${variable}!~${definition}", null,
    ↳ 0),
253 // Inside the scope of ~!root!~ replace:
254 // root
255 // *root
256 (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),
257 // Remove scope borders.
258 // ~!root!~
259 //
260 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
261 // ref auto root = ref
262 // ref auto root =
263 (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\\W)", "$1* $2 =$3", null, 0),
264 // *root = ref left;
265 // root = left;

```

```

266 (new Regex(@"*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)", "$1 = $2$3", null, 0),
267 // (ref left)
268 // (left)
269 (new Regex(@"\((ref ([a-zA-Z0-9]+)(\)|\(|,))", "($1$2", null, 0),
270 // ref TElement
271 // TElement*
272 (new Regex(@"(\(|\()ref ([a-zA-Z0-9]+) ", "$1$2* ", null, 0),
273 // ref SizeBalancedTree.Root
274 // &sizeBalancedTree->Root
275 (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)", "&$1->$2", null, 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]+)",
279     ↳ "$1($2)->$3", null, 0),
280 // GetElement(node).Right
281 // GetElement(node)->Right
282 (new Regex(@"([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)", "$1($2)->$3",
283     ↳ null, 0),
284 // [Fact]\npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
285 // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
286 (new Regex(@"\[Fact\]\[s\n\]+(public: )?(static )?void ([a-zA-Z0-9]+)\(\)", "public:
287     ↳ TEST_METHOD($3)", null, 0),
288 // class TreesTests
289 // TEST_CLASS(TreesTests)
290 (new Regex(@"class ([a-zA-Z0-9]+)Tests", "TEST_CLASS($1)", null, 0),
291 // Assert.Equal
292 // Assert::AreEqual
293 (new Regex(@"(Assert)\.Equal", "$1::AreEqual", null, 0),
294 // Assert.Throws
295 // Assert::ExpectException
296 (new Regex(@"(Assert)\.Throws", "$1::ExpectException", null, 0),
297 // $"Argument {argumentName} is null."
298 // ((std::string)"Argument ").append(argumentName).append(" is null.").data()
299 (new Regex(@"\$"("(?<left>\\"|"[^""\r\n])*" ){(?<expression>[_a-zA-Z0-9]+)}(?<right>\\"|
300     ↳ "\"|"[^""\r\n])*" )"),
301     ↳ "((std::string)$\"${left}\" ).append(${expression}).append(\"${right}\").data()",
302     ↳ null, 10),
303 // $"
304 // "
305 (new Regex(@"\$"""), "\"", null, 0),
306 // Console.WriteLine("...")
307 // printf("...\n")
308 (new Regex(@"Console\.WriteLine\(\"([^\r\n]+)\" \)", "printf(\"$1\\n\")", null, 0),
309 // TElement Root;
310 // TElement Root = 0;
311 (new Regex(@"(\r?\n\t )+(private|protected|public)?(: )?([a-zA-Z0-9]+)
312     ↳ )?([a-zA-Z0-9:_]+)(?!return)) ([a-zA-Z0-9]+);", "$1$2$3$4 $5 = 0;", null, 0),
313 // TreeElement _elements[N];
314 // TreeElement _elements[N] = { {0} };
315 (new Regex(@"(\r?\n\t )+(private|protected|public)?(: )?([a-zA-Z0-9]+)
316     ↳ ([a-zA-Z0-9]+)\([([a-zA-Z0-9]+)\];", "$1$2$3$4 $5[$6] = { {0} };", null, 0),
317 // auto path = new TElement[MaxPath];
318 // TElement path[MaxPath] = { {0} };
319 (new Regex(@"(\r?\n\t )+[a-zA-Z0-9]+ ([a-zA-Z0-9]+) = new
320     ↳ ([a-zA-Z0-9]+)\([([a-zA-Z0-9]+)\];", "$1$3 $2[$4] = { {0} };", null, 0),
321 // private: static readonly ConcurrentBag<std::exception> _exceptionsBag = new
322     ↳ ConcurrentBag<std::exception>();
323 // private: inline static std::mutex _exceptionsBag_mutex; \n\n private: inline
324     ↳ static std::vector<std::exception> _exceptionsBag;
325 (new Regex(@"(?<begin>\r?\n?(?<indent>[ \t]+))(?<access>(private|protected|public):
326     ↳ )?static readonly ConcurrentBag<(?<argumentType>[^\r\n]+)>
327     ↳ (?<name>[_a-zA-Z0-9]+) = new ConcurrentBag<k<argumentType>>\(\);",
328     ↳ "${begin}private: inline static std::mutex ${name}_mutex;" + Environment.NewLine
329     ↳ + Environment.NewLine + "${indent}${access}inline static
330     ↳ std::vector<${argumentType}> ${name};", null, 0),
331 // public: static IReadOnlyCollection<std::exception> GetCollectedExceptions() {
332     ↳ return _exceptionsBag; }
333 // public: static std::vector<std::exception> GetCollectedExceptions() { return
334     ↳ std::vector<std::exception>(_exceptionsBag); }
335 (new Regex(@"(?<access>(private|protected|public): )?static
336     ↳ IReadOnlyCollection<(?<argumentType>[^\r\n]+)> (?<methodName>[_a-zA-Z0-9]+)\(\)
337     ↳ { return (?<fieldName>[_a-zA-Z0-9]+); }", "${access}static
338     ↳ std::vector<${argumentType}> ${methodName}() { return
339     ↳ std::vector<${argumentType}>(${fieldName}); }", null, 0),
340 // public: static event EventHandler<std::exception> ExceptionIgnored =
341     ↳ OnExceptionIgnored; ... };

```

```

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

```



```

354 // ExceptionIgnored.Invoke(NULL, exception);
355 // 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}", null, 10),
357 // Remove scope borders.
358 // /*~ExceptionIgnored~*/
359 //
360 (new Regex(@"/*~[a-zA-Z0-9]+~\*/"), "", null, 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};", null, 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}", null, 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()", null, 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')",
→ null, 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}'))",
→ null, 10),
378 // sb.AppendLine(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})", null,
→ 10),
381 // Remove scope borders.
382 // /*~sb~*/
383 //
384 (new Regex(@"/*~[a-zA-Z0-9]+~\*/"), "", null, 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};", null, 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})", null, 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})", null, 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});", null, 0),
400 // Remove scope borders.
401 // ~!added!~

```



```

402 //
403 (new Regex(@"~![a-zA-Z0-9]+!~"), "", null, 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\((([a-zA-Z0-9]+)\);", "~!$!~std::srand($3);", null, 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}", null, 10),
412 // Remove scope borders.
413 // ~!random!~
414 //
415 (new Regex(@"~![a-zA-Z0-9]+!~"), "", null, 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*/${end}", null,
    ↳ 0),
420 // Insert method body scope ends.
421 // { /*method-start*/...}
422 // { /*method-start*/... /*method-end*/}
423 (new Regex(@"{ /*method-start*/(?<body>((?<bracket>\\{)|(?!<-bracket>\\})|[^\\{\\}]+)+)
    ↳ \\}", "{ /*method-start*/${body} /*method-end*/}", null,
    ↳ 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}(", null, 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}", null, 100),
429 // Remove scope borders.
430 // /*method-start*/
431 //
432 (new Regex(@"\\/*method-(start|end)\\/*"), "", null, 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}", null, 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()", null, 10),
441 // Remove scope borders.
442 // /*~ex~*/
443 //
444 (new Regex(@"\\/*~[_a-zA-Z0-9]+~\\/*"), "", null, 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("\\.\\)");", null, 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("\.\\"));", null, 0),
451 // throw new ArgumentOutOfRangeException(argumentName, argumentValue,
    ↳ messageBuilder());
452 // throw std::invalid_argument(((std::string)"Value
    ↳ [").append(std::to_string(argumentValue)).append("] of argument
    ↳ [").append(argumentName).append("] is out of range:
    ↳ ").append(messageBuilder()).append("\.\\"));
453 (new Regex(@"throw new ArgumentOutOfRangeException\((?<argument>[a-zA-Z]*[Aa]rgument
    ↳ [a-zA-Z]*[([Nn]ame[a-zA-Z]*)?),
    ↳ (?<argumentValue>[a-zA-Z]*[Aa]rgument[a-zA-Z]*([Vv]alue[a-zA-Z]*)?),
    ↳ (?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\(\)?)\)\);", "throw
    ↳ std::invalid_argument(((std::string)"Value
    ↳ [").append(std::to_string(${argumentValue}).append("\] of argument
    ↳ [").append(${argument}).append("\] is out of range:
    ↳ \").append(${message}).append("\.\\"));", null, 0),
454 // throw new NotSupportedException();
455 // throw std::logic_error("Not supported exception.");
456 (new Regex(@"throw new NotSupportedException\(\);", "throw std::logic_error(\"Not
    ↳ supported exception.\");", null, 0),
457 // throw new NotImplementedException();
458 // throw std::logic_error("Not implemented exception.");
459 (new Regex(@"throw new NotImplementedException\(\);", "throw std::logic_error(\"Not
    ↳ implemented exception.\");", null, 0),
460 }.Cast<ISubstitutionRule>().ToList());
461
462 public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
463 {
464     // ICounter<int, int> c1;
465     // ICounter<int, int>* c1;
466     (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[~>\r\n]+)?)
    ↳ (?<variable>[_a-zA-Z0-9]+);", "${abstractType}* ${variable};", null, 0),
467     // (expression)
468     // expression
469     (new Regex(@"(\(|\)|)(([a-zA-Z0-9_\*:]+)\(|\)|;|\)|)"), "$1$2$3", null, 0),
470     // (method(expression))
471     // method(expression)
472     (new Regex(@"(?<firstSeparator>(\(|
    ↳ ))\((?<method>[a-zA-Z0-9_\*:]+)\((?<expression>((?<parenthesis>\(|(?<-parent
    ↳ hesis>)\)|[a-zA-Z0-9_\*:]+)(?(parenthesis)(?!))\)\)(?<lastSeparator>(\(|
    ↳ |;|\)|)"))", "${firstSeparator}${method}(${expression})${lastSeparator}", null, 0),
473     // return ref _elements[node];
474     // return &elements[node];
475     (new Regex(@"return ref ([a-zA-Z0-9]+)\([([a-zA-Z0-9_\*:]+)\];", "return &1[2];",
    ↳ null, 0),
476     // null
477     // nullptr
478     (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"""| [~""\r\n])*""[~""\r\n])*)(?<=\\W)null
    ↳ (?<after>\\W)", "${before}nullptr${after}", null,
    ↳ 10),
479     // default
480     // 0
481     (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"""| [~""\r\n])*""[~""\r\n])*)(?<=\\W)defa
    ↳ ult(?<after>\\W)", "${before}0${after}", null,
    ↳ 10),
482     // object x
483     // void *x
484     (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"""| [~""\r\n])*""[~""\r\n])*)(?<=\\W)([O|
    ↳ o]bject|System\\.Object) (?<after>\\W)", "${before}void *${after}", null,
    ↳ 10),
485     // <object>
486     // <void*>
487     (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),
488     // ArgumentNullException
489     // std::invalid_argument
490     (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"""| [~""\r\n])*""[~""\r\n])*)(?<=\\W)(Sys
    ↳ tem\\.)?ArgumentNullException(?<after>\\W)",
    ↳ "${before}std::invalid_argument${after}", null, 10),
491     // #region Always
492     //
493     (new Regex(@"(^\r?\n)[ \t]*#(region|endregion)[^\r\n]*(\r?\n|$)", "", null, 0),

```

```

494 // //define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
495 //
496 (new Regex(@"\\\/[ \t]*#define[ \t]+[_a-zA-Z0-9]+[ \t]*"), "", null, 0),
497 // #if USEARRAYPOOL\r\n#endif
498 //
499 (new Regex(@"#if [a-zA-Z0-9]+\s+#endif"), "", null, 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}", null, 5),
503 // \n ... namespace
504 // namespace
505 (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+namespace"), "$1namespace", null, 0),
506 // \n ... class
507 // class
508 (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+class"), "$1class", null, 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("", 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, 11
./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1