

1.1 ./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text.RegularExpressions;
5
6  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
7
8  namespace Platform.RegularExpressions.Transformer.CSharpToCpp
9  {
10     public class CSharpToCppTransformer : TextTransformer
11     {
12         public static readonly IList

```

```

58 //
59 (new Regex(@"\r?\n[^\n]+bool operator !=\(((?<type>[^\n]+) (?<left>[a-zA-Z0-9]+),
    ↳ \k<type> (?<right>[a-zA-Z0-9]+)\) => !\((\k<left>|\k<right>) ==
    ↳ (\k<left>|\k<right>)\);"), "", 10),
60 // public override bool Equals(object obj) => obj is Range<T> range ? Equals(range)
    ↳ : false;
61 //
62 (new Regex(@"\r?\n[^\n]+override bool Equals\((System\.)?[Oo]bject
    ↳ (?<this>[a-zA-Z0-9]+)\) => \k<this> is [^\n]+ (?<other>[a-zA-Z0-9]+) \?
    ↳ Equals\(\k<other>\) : false;"), "", 10),
63 // out TProduct
64 // TProduct
65 (new Regex(@"(?<before>(<|, ))(in|out)
    ↳ (?<typeParameter>[a-zA-Z0-9]+)(?<after>(>|,))"),
    ↳ "${before}${typeParameter}${after}", 10),
66 // public ...
67 // public: ...
68 (new Regex(@"(?<newLineAndIndent>\r?\n?[
    ↳ \t]*) (?<before>[^\{\\(\r\n)*) (?<access>private|protected|public) [
    ↳ \t]+(?! [^\{\\(\r\n)*(interface|class|struct) [^\{\\(\r\n)*[\\(\r\n)]"),
    ↳ "${newLineAndIndent}${access}: ${before}", 0),
69 // public: static bool CollectExceptions { get; set; }
70 // public: inline static bool CollectExceptions;
71 (new Regex(@"(?<access>(private|protected|public): )(?<before>(static )?[^\r\n]+
    ↳ )(?<name>[a-zA-Z0-9]+) { [^;]* (?<=\\W) get; [^;]* (?<=\\W) set; [^;]* }"),
    ↳ "${access}inline ${before}${name};", 0),
72 // public abstract class
73 // class
74 (new Regex(@"((public|protected|private|internal|abstract|static)
    ↳ )*(?<category>interface|class|struct)", "${category}", 0),
75 //class GenericCollectionMethodsBase<TElement> {
76 // template <typename TElement> class GenericCollectionMethodsBase {
77 (new Regex(@"class ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>([^\{]+){", "template <typename $2>
    ↳ class $1$3{", 0),
78 // static void
    ↳ TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
    ↳ tree, TElement* root)
79 // template<typename T> static void
    ↳ TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
    ↳ tree, TElement* root)
80 (new Regex(@"static ([a-zA-Z0-9]+) ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>\\(((^\\)\r\n)+\\)",
    ↳ "template <typename $3> static $1 $2($4)", 0),
81 // interface IFactory<out TProduct> {
82 // template <typename TProduct> class IFactory { public:
83 (new Regex(@"interface (?<interface>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9
    ↳ ,]+)>(?<whitespace>[^\{]+){", "template <typename...> class ${interface};
    ↳ template <typename ${typeParameters}> class
    ↳ ${interface}<${typeParameters}>${whitespace}{ " + Environment.NewLine + "
    ↳ public:", 0),
84 // template <typename TObject, TProperty, TValue>
85 // template <typename TObject, typename TProperty, TValue>
86 (new Regex(@"(?<before>template <((, )?typename [a-zA-Z0-9]+)+,
    ↳ )(?<typeParameter>[a-zA-Z0-9]+)(?<after>(>|,))"), "${before}typename
    ↳ ${typeParameter}${after}", 10),
87 // Insert markers
88 // private: static void BuildExceptionString(this StringBuilder sb, Exception
    ↳ exception, int level)
89 // /*~extensionMethod~BuildExceptionString~*/private: static void
    ↳ BuildExceptionString(this StringBuilder sb, Exception exception, int level)
90 (new Regex(@"private: static [^\r\n]+ (?<name>[a-zA-Z0-9]+)\\(this [^\)\r\n]+\\)",
    ↳ "/*~extensionMethod~${name}~*/$0", 0),
91 // Move all markers to the beginning of the file.
92 (new Regex(@"\A(?<before>[^\r\n]+\r?\n(?:.|\\n) )(?<marker>\/\*~extensionMethod~(?<name>
    ↳ [a-zA-Z0-9]+)~\*/)", "${marker}${before}",
    ↳ 10),
93 // /*~extensionMethod~BuildExceptionString~*/...sb.BuildExceptionString(exception.In
    ↳ nerException, level +
    ↳ 1);
94 // /*~extensionMethod~BuildExceptionString~*/...BuildExceptionString(sb,
    ↳ exception.InnerException, level + 1);
95 (new Regex(@"(?<before>\/\*~extensionMethod~(?<name>[a-zA-Z0-9]+)~\*/(?:.|\\n)+\\W )(?<var
    ↳ iable>[_a-zA-Z0-9]+)\\. \k<name>\\(", "${before}${name}(${variable}, ",
    ↳ 50),
96 // Remove markers
97 // /*~extensionMethod~BuildExceptionString~*/
98 //

```

```

99     (new Regex(@"/*~extensionMethod~[a-zA-Z0-9]+~\*/", "", 0),
100     // (this
101     // (
102     (new Regex(@"\((this ", "(", 0),
103     // public: static readonly EnsureAlwaysExtensionRoot Always = new
104     → EnsureAlwaysExtensionRoot();
105     // public: inline static EnsureAlwaysExtensionRoot Always;
106     (new Regex(@"(?:<access>(private|protected|public): )?static readonly
107     → (?<type>[a-zA-Z0-9]+) (?<name>[a-zA-Z0-9_]+) = new \k<type>\(\);",
108     → "${access}inline static ${type} ${name};", 0),
109     // public: static readonly string ExceptionContentsSeparator = "---";
110     // public: inline static const char* ExceptionContentsSeparator = "---";
111     (new Regex(@"(?:<access>(private|protected|public): )?(const|static readonly) string
112     → (?<name>[a-zA-Z0-9_]+) = ""(?:<string>(\\"|~\r\n))+"";", "${access}inline
113     → static const char* ${name} = \"${string}\";", 0),
114     // private: const int MaxPath = 92;
115     // private: inline static const int MaxPath = 92;
116     (new Regex(@"(?:<access>(private|protected|public): )?(const|static readonly)
117     → (?<type>[a-zA-Z0-9]+) (?<name>[_a-zA-Z0-9_]+) = (?<value>[~;\r\n]+);",
118     → "${access}inline static const ${type} ${name} = ${value};", 0),
119     // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
120     → TArgument : class
121     // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
122     (new Regex(@"(?:<before> [a-zA-Z]+\\((([a-zA-Z *],)+, |)))(?:<type>[a-zA-Z]+)(?:<after>(\\
123     → [a-zA-Z *,]+))\\)[ \r\n]+where \k<type> : class)", "${before}${type}*${after}",
124     → 0),
125     // protected: abstract TElement GetFirst();
126     // protected: virtual TElement GetFirst() = 0;
127     (new Regex(@"(?:<access>(private|protected|public): )?abstract
128     → (?<method>[~;\r\n]+);", "${access}virtual ${method} = 0;", 0),
129     // TElement GetFirst();
130     // virtual TElement GetFirst() = 0;
131     (new Regex(@"([ \r\n]+[ ]+)((?!return)[a-zA-Z0-9]+ [a-zA-Z0-9]+\\([~\\)\r\n]*))\\(;[
132     → ]*[ \r\n]+)", "$1virtual $2 = 0$3", 1),
133     // protected: readonly TreeElement[] _elements;
134     // protected: TreeElement _elements[N];
135     (new Regex(@"(?:<access>(private|protected|public): )?readonly
136     → (?<type>[a-zA-Z<0-9]+)\\([~\\])\\ (?<name>[_a-zA-Z0-9_]+);", "${access}${type}
137     → ${name}[N];", 0),
138     // protected: readonly TElement Zero;
139     // protected: TElement Zero;
140     (new Regex(@"(?:<access>(private|protected|public): )?readonly
141     → (?<type>[a-zA-Z<0-9]+) (?<name>[_a-zA-Z0-9_]+);", "${access}${type} ${name};",
142     → 0),
143     // internal
144     //
145     (new Regex(@"(\\W)internal\\s+)", "$1", 0),
146     // static void NotImplementedException(ThrowExtensionRoot root) => throw new
147     → NotImplementedException();
148     // static void NotImplementedException(ThrowExtensionRoot root) { return throw new
149     → NotImplementedException(); }
150     (new Regex(@"(^\\s+)(private|protected|public)?(?: )?(template \\<[^\\r\\n]+\\> )?(static
151     → )?(override )?([a-zA-Z0-9]+
152     → )([a-zA-Z0-9_+\\(((\\[~\\)\r\n]*))\\s+=>\\s+throw([~;\r\n]+);)",
153     → "$1$2$3$4$5$6$7$8($9) { throw$10; }", 0),
154     // SizeBalancedTree(int capacity) => a = b;
155     // SizeBalancedTree(int capacity) { a = b; }
156     (new Regex(@"(^\\s+)(private|protected|public)?(?: )?(template \\<[^\\r\\n]+\\> )?(static
157     → )?(override )?(void )?([a-zA-Z0-9]+\\(((\\[~\\)\r\n]*))\\s+=>\\s+([~;\r\n]+);)",
158     → "$1$2$3$4$5$6$7$8($9) { $10; }", 0),
159     // int SizeBalancedTree(int capacity) => a;
160     // int SizeBalancedTree(int capacity) { return a; }
161     (new Regex(@"(^\\s+)(private|protected|public)?(?: )?(template \\<[^\\r\\n]+\\> )?(static
162     → )?(override )?([a-zA-Z0-9]+
163     → )([a-zA-Z0-9_+\\(((\\[~\\)\r\n]*))\\s+=>\\s+([~;\r\n]+);)", "$1$2$3$4$5$6$7$8($9) {
164     → return $10; }", 0),
165     // () => Integer<TElement>.Zero,
166     // () { return Integer<TElement>.Zero; },
167     (new Regex(@"\\(\\)\\s+=>\\s+(?:<expression>[~(),;\\r\\n]+\\(((\\(?:<parenthesis>\\(|(?:<-parent
168     → hesis>\\)|[~(),;\\r\\n]*?))\\)?[~(),;\\r\\n]*)(?:<after>,|\\);)", "()" { return
169     → ${expression}; }${after}", 0),
170     // => Integer<TElement>.Zero;
171     // { return Integer<TElement>.Zero; }
172     (new Regex(@"\\)\\s+=>\\s+([~;\r\n]+?);", ") { return $1; }", 0),
173     // () { return avlTree.Count; }
174     // [&]()-> auto { return avlTree.Count; }

```

```

147 (new Regex(@"(?<before>, |\\)\(\) { return (?<expression>[~;\r\n]+); }"),
148     ↳ "${before}[&]() -> auto { return ${expression}; }", 0),
149 // Count => GetSizeOrZero(Root);
150 // GetCount() { return GetSizeOrZero(Root); }
151 (new Regex(@"(\\W)([A-Z][a-zA-Z]+)\\s+=>\\s+([~;\r\n]+);"), "$1Get$2() { return $3; }",
152     ↳ 0),
153 // ArgumentInRange(const char* message) { const char* messageBuilder() { return
154     ↳ message; }
155 // ArgumentInRange(const char* message) { auto messageBuilder = [&]() -> const char*
156     ↳ { return message; };
157 (new Regex(@"(?<before>\\W[_a-zA-Z0-9]+\\(([^\\]\\n)*\\)[\\s\\n]*{([\\s\\n]*([^{}]|\\n)*?(\\r?\\n)|
158     ↳ ?[ \\t]*)(?<returnType>[_a-zA-Z0-9*:] +[_a-zA-Z0-9*:]*)
159     ↳ (?<methodName>[_a-zA-Z0-9]+)\\((?<arguments>[^\\]\\n)*\\)\\s*{(?<body>("[^"\\n]+"|
160     ↳ [^}]|\\n)+?)}"), "${before}auto ${methodName} = [&]() -> ${returnType}
161     ↳ {${body}};", 10),
162 // Func<TElement> treeCount
163 // std::function<TElement()> treeCount
164 (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", 0),
165 // Action<TElement> free
166 // std::function<void(TElement)> free
167 (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<void($1)> $2",
168     ↳ 0),
169 // Predicate<TArgument> predicate
170 // std::function<bool(TArgument)> predicate
171 (new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<bool($1)>
172     ↳ $2", 0),
173 // var
174 // auto
175 (new Regex(@"(\\W)var(\\W)"), "$1auto$2", 0),
176 // unchecked
177 //
178 (new Regex(@"[\\r\\n]{2}\\s*?unchecked\\s*?$"), "", 0),
179 // throw new InvalidOperationException
180 // throw std::runtime_error
181 (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
182     ↳ std::runtime_error", 0),
183 // void RaiseExceptionIgnoredEvent(Exception exception)
184 // void RaiseExceptionIgnoredEvent(const std::exception& exception)
185 (new Regex(@"(\\(| ) (System\\.Exception|Exception) (|\\))"), "$1const
186     ↳ std::exception&$3", 0),
187 // EventHandler<Exception>
188 // EventHandler<std::exception>
189 (new Regex(@"(\\W) (System\\.Exception|Exception) (\\W)"), "$1std::exception$3", 0),
190 // override void PrintNode(TElement node, StringBuilder sb, int level)
191 // void PrintNode(TElement node, StringBuilder sb, int level) override
192 (new Regex(@"override ([a-zA-Z0-9 \\*\\+]+)\\(([^\\]\\r\\n|\\n)+?\\)"), "$1$2 override", 0),
193 // return (range.Minimum, range.Maximum)
194 // return {range.Minimum, range.Maximum}
195 (new Regex(@"(?<before>return\\s*)\\((?<values>[^\\]\\n)+\\)\\((?!\\() (?<after>\\W)"),
196     ↳ "${before}${values}}${after}", 0),
197 // string
198 // const char*
199 (new Regex(@"(\\W)string(\\W)"), "$1const char*$2", 0),
200 // System.ValueTuple
201 // std::tuple
202 (new Regex(@"(?<before>\\W) (System\\.)?ValueTuple(?:\\s*=) (?<after>\\W)"),
203     ↳ "${before}std::tuple${after}", 0),
204 // sbyte
205 // std::int8_t
206 (new Regex(@"(?<before>\\W) ((System\\.)?SB|sbyte(?:\\s*=) (?<after>\\W)"),
207     ↳ "${before}std::int8_t${after}", 0),
208 // short
209 // std::int16_t
210 (new Regex(@"(?<before>\\W) ((System\\.)?Int16|short(?:\\s*=) (?<after>\\W)"),
211     ↳ "${before}std::int16_t${after}", 0),
212 // int
213 // std::int32_t
214 (new Regex(@"(?<before>\\W) ((System\\.)?I|i)nt(32)?(?:\\s*=) (?<after>\\W)"),
215     ↳ "${before}std::int32_t${after}", 0),
216 // long
217 // std::int64_t
218 (new Regex(@"(?<before>\\W) ((System\\.)?Int64|long(?:\\s*=) (?<after>\\W)"),
219     ↳ "${before}std::int64_t${after}", 0),
220 // byte
221 // std::uint8_t

```

```

204 (new Regex(@"(?<before>\W)((System\.)?Byte|byte)(?!\\s*)(?<after>\W)",
    ↪ "${before}std::uint8_t${after}", 0),
205 // ushort
206 // std::uint16_t
207 (new Regex(@"(?<before>\W)((System\.)?UInt16|ushort)(?!\\s*)(?<after>\W)",
    ↪ "${before}std::uint16_t${after}", 0),
208 // uint
209 // std::uint32_t
210 (new Regex(@"(?<before>\W)((System\.)?UI|ui)nt(32)?(?!\\s*)(?<after>\W)",
    ↪ "${before}std::uint32_t${after}", 0),
211 // ulong
212 // std::uint64_t
213 (new Regex(@"(?<before>\W)((System\.)?UInt64|ulong)(?!\\s*)(?<after>\W)",
    ↪ "${before}std::uint64_t${after}", 0),
214 // char*[] args
215 // char* args[]
216 (new Regex(@"([_a-zA-Z0-9:\*]?)\\[\\] ([a-zA-Z0-9]+)", "$1 $2[]", 0),
217 // @object
218 // object
219 (new Regex(@"@([_a-zA-Z0-9]+)", "$1", 0),
220 // float.MinValue
221 // std::numeric_limits<float>::min()
222 (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+float|double)\\.MinValue(?<after>\W)",
    ↪ ")"), "${before}std::numeric_limits<${type}>::min()${after}",
    ↪ 0),
223 // double.MaxValue
224 // std::numeric_limits<float>::max()
225 (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+float|double)\\.MaxValue(?<after>\W)",
    ↪ ")"), "${before}std::numeric_limits<${type}>::max()${after}",
    ↪ 0),
226 // using Platform.Numbers;
227 //
228 (new Regex(@"([\\r\\n]{2}|^\\s*?using [\\_a-zA-Z0-9]+;\\s*?$)", "", 0),
229 // struct TreeElement { }
230 // struct TreeElement { };
231 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\\s+){([\\sa-zA-Z0-9;:_]+?)}([~;])", "$1
    ↪ $2$3{$4};$5", 0),
232 // class Program { }
233 // class Program { };
234 (new Regex(@"(struct|class) ([a-zA-Z0-9]+[~\\r\\n]*)([\\r\\n]+(?<indentLevel>[\\t
    ↪ ]*))?\\{([\\S\\s]+?[\\r\\n]+\\k<indentLevel>)\\}([~;]|$)", "$1 $2$3{$4};$5", 0),
235 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
236 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
237 (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)", "class $1 : public $2", 0),
238 // class IProperty : ISetter<TValue, TObj>, IProvider<TValue, TObj>
239 // class IProperty : public ISetter<TValue, TObj>, IProvider<TValue, TObj>
240 (new Regex(@"(?<before>class [a-zA-Z0-9]+ : ((public [a-zA-Z0-9]+(<[a-zA-Z0-9
    ↪ ,]+>)?, )+)?(?<inheritedType>(?!public)[a-zA-Z0-9]+(<[a-zA-Z0-9
    ↪ ,]+>)?(?<after>(, [a-zA-Z0-9]+(?!>)|[ \\r\\n]+))", "${before}public
    ↪ ${inheritedType}${after}", 10),
241 // Insert scope borders.
242 // ref TElement root
243 // ~!root!~ref TElement root
244 (new Regex(@"(?<definition>(?!\\() (ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<ref>))
    ↪ (?<variable>[a-zA-Z0-9]+)(?=\\)|, | =))", "~!${variable}!~${definition}", 0),
245 // Inside the scope of ~!root!~ replace:
246 // root
247 // *root
248 (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),
249 // Remove scope borders.
250 // ~!root!~
251 //
252 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~", "", 5),
253 // ref auto root = ref
254 // ref auto root =
255 (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\\W)", "$1* $2 =$3", 0),
256 // *root = ref left;
257 // root = left;
258 (new Regex(@"\\*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\\W)", "$1 = $2$3", 0),
259 // (ref left)
260 // (left)
261 (new Regex(@"\\(ref ([a-zA-Z0-9]+)(\\)|\\(|,)", "($1$2", 0),
262 // ref TElement

```

```

263 // TElement*
264 (new Regex(@"( | \()ref ([a-zA-Z0-9]+) ", "$1$2* ", 0),
265 // ref sizeBalancedTree.Root
266 // &sizeBalancedTree->Root
267 (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)", "&$1->$2", 0),
268 // ref GetElement(node).Right
269 // &GetElement(node)->Right
270 (new Regex(@"ref ([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+))\)\.([a-zA-Z0-9]+)",
    → "&$1($2)->$3", 0),
271 // GetElement(node).Right
272 // GetElement(node)->Right
273 (new Regex(@"([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+))\)\.([a-zA-Z0-9]+)", "$1($2)->$3", 0),
274 // [Fact]\npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
275 // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
276 (new Regex(@"\[Fact\]\[s\n\]+(public:)?(static)?void ([a-zA-Z0-9]+)\(\)\)", "public:
    → TEST_METHOD($3)", 0),
277 // class TreesTests
278 // TEST_CLASS(TreesTests)
279 (new Regex(@"class ([a-zA-Z0-9]+)Tests)", "TEST_CLASS($1)", 0),
280 // Assert.Equal
281 // Assert::AreEqual
282 (new Regex(@"(Assert)\.Equal)", "$1::AreEqual", 0),
283 // Assert.Throws
284 // Assert::ExpectException
285 (new Regex(@"(Assert)\.Throws)", "$1::ExpectException", 0),
286 // $"Argument {argumentName} is null."
287 // ((std::string)"Argument ").append(argumentName).append(" is null.").data()
288 (new Regex(@"\$" "(?<left>(\\" | [^"\\r\\n])*" (?<expression>[_a-zA-Z0-9]+) (?<right>(\\"
    → \\" | [^"\\r\\n])*" )",
    → "((std::string)$\" ${left}\").append(${expression}).append(\" ${right}\").data()",
    → 10),
289 // $"
290 // "
291 (new Regex(@"\$"""), "\"", 0),
292 // ((std::string)((std::string)"[").append(Minimum).append(",
    → ").data()).append(Maximum).append("]").data()
293 // ((std::string)"[").append(Minimum).append(", ").append(Maximum).append("]").data()
294 (new Regex(@"\\(\\(std::string\\)(?<begin>\\(\\(std::string\\)""(\\\\" | [^"\\])*"\\)\\)\\.append\\
    → \\(\\(\\n\\+\\))\\)\\.data\\(\\)\\)\\.append", "${begin}.append",
    → 10),
295 // Console.WriteLine("...")
296 // printf("...\n")
297 (new Regex(@"Console\\.WriteLine\\(""(\\\\" | [^"\\r\\n])*"\\)", "printf(\"$1\\n\\n\")", 0),
298 // TElement Root;
299 // TElement Root = 0;
300 (new Regex(@"(\\r?\\n[\\t ]+)(private|protected|public)?(:
    → )?([a-zA-Z0-9:]+(?<return>)) ([a-zA-Z0-9]+);", "$1$2$3$4 $5 = 0;", 0),
301 // TreeElement _elements[N];
302 // TreeElement _elements[N] = { {0} };
303 (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),
304 // auto path = new TElement[MaxPath];
305 // TElement path[MaxPath] = { {0} };
306 (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[$4] = { {0} };", 0),
307 // bool Equals(Range<T> other) { ... }
308 // bool operator ==(const Key &other) const { ... }
309 (new Regex(@"(?<before>\\r?\\n[\\n]+bool )Equals\\((?<type>[\\n]+)
    → (?<variable>[a-zA-Z0-9]+)\\) (?<after>\\(s|\\n)*{")", "${before}operator ==(const
    → ${type} &${variable}) const${after}", 0),
310 // Insert scope borders.
311 // class Range { ... public: override const char* ToString() { return ...; }
312 // class Range { /*~Range~/ ... public: override const char* ToString() { return
    → ...; }
313 (new Regex(@"(?<classDeclarationBegin>\\r?\\n(?<indent>[\\t ]*)(struct|class)
    → (?<type>[a-zA-Z0-9]+(<((?!\\s*:\\s*)[\\n]+>)?(\\s*:\\s*[\\n]+)?[\\t ]*(\\r?\\n)?[\\t
    → ]*(?<middle>((?!class|struct)\\.\\n)+)?(?<toStringDeclaration>(?<access>(private|
    → |protected|public): )override const char* ToString\\(\\))",
    → "${classDeclarationBegin}/${type}~/${middle}/${toStringDeclaration}", 0),
314 // Inside the scope of ~!_exceptionsBag!~ replace:
315 // public: override const char* ToString() { return ...; }
316 // public: operator std::string() const { return ...; }\\n\\npublic: friend
    → std::ostream & operator << (std::ostream &out, const A &obj) { return out <<
    → (std::string)obj; }

```

```

317 (new Regex(@"(?<scope>\/~(?<type>[_a-zA-Z0-9<>:~\*\/])(?<separator>.\|\\n)(?<before>
    ((?!\/~\k<type>~\*\/)(.\\n))*?(?<toStringDeclaration>\r?\\n(?<indent>[
    ↪ \t]*)?(?<access>(private|protected|public): )override const char\* ToString\\(\\
    ↪ (?<toStringMethodBody>{[~\\n]+}))"), "${scope}${separator}${before}" +
    ↪ Environment.NewLine + "${indent}${access}operator std::string() const
    ↪ ${toStringMethodBody}" + Environment.NewLine + Environment.NewLine +
    ↪ "${indent}${access}friend std::ostream & operator << (std::ostream &out, const
    ↪ ${type} &obj) { return out << (std::string)obj; }", 0),
318 // Remove scope borders.
319 // /*~Range~*/
320 //
321 (new Regex(@"\/~[_a-zA-Z0-9<>:~\*\/]"), "", 0),
322 // private: static readonly ConcurrentBag<std::exception> _exceptionsBag = new
    ↪ ConcurrentBag<std::exception>();
323 // private: inline static std::mutex _exceptionsBag_mutex; \\n\\n private: inline
    ↪ static std::vector<std::exception> _exceptionsBag;
324 (new Regex(@"(?<begin>\r?\\n?(?<indent>[ \t]+))(?<access>(private|protected|public):
    ↪ )?static readonly ConcurrentBag<(?<argumentType>[~;\\r\\n]+)>
    ↪ (?<name>[_a-zA-Z0-9]+) = new ConcurrentBag<\k<argumentType>>\\(\\);"),
    ↪ "${begin}private: inline static std::mutex ${name}_mutex;" + Environment.NewLine
    ↪ + Environment.NewLine + "${indent}${access}inline static
    ↪ std::vector<${argumentType}> ${name};", 0),
325 // public: static IReadonlyCollection<std::exception> GetCollectedExceptions() {
    ↪ return _exceptionsBag; }
326 // public: static std::vector<std::exception> GetCollectedExceptions() { return
    ↪ std::vector<std::exception>(_exceptionsBag); }
327 (new Regex(@"(?<access>(private|protected|public): )?static
    ↪ IReadonlyCollection<(?<argumentType>[~;\\r\\n]+)> (?<methodName>[_a-zA-Z0-9]+)\\(\\)
    ↪ { return (?<fieldName>[_a-zA-Z0-9]+); }"), "${access}static
    ↪ std::vector<${argumentType}> ${methodName}() { return
    ↪ std::vector<${argumentType}>(${fieldName}); }", 0),
328 // public: static event EventHandler<std::exception> ExceptionIgnored =
    ↪ OnExceptionIgnored; ... };
329 // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
    ↪ const std::exception&> ExceptionIgnored = OnExceptionIgnored; };
330 (new Regex(@"(?<begin>\r?\\n(\\r?\\n)?(?<halfIndent>[
    ↪ \t]+)\k<halfIndent>)(?<access>(private|protected|public): )?static event
    ↪ EventHandler<(?<argumentType>[~;\\r\\n]+)> (?<name>[_a-zA-Z0-9]+) = (?<defaultDele
    ↪ gate>[_a-zA-Z0-9]+); (?<middle>(.\\n)+?) (?<end>\\r?\\n\k<halfIndent>});"),
    ↪ "${middle}" + Environment.NewLine + Environment.NewLine +
    ↪ "${halfIndent}${halfIndent}${access}static inline
    ↪ Platform::Delegates::MulticastDelegate<void(void*, const ${argumentType}&>
    ↪ ${name} = ${defaultDelegate};${end}", 0),
331 // Insert scope borders.
332 // class IgnoredExceptions { ... private: inline static std::vector<std::exception>
    ↪ _exceptionsBag;
333 // class IgnoredExceptions { /*~_exceptionsBag~*/ ... private: inline static
    ↪ std::vector<std::exception> _exceptionsBag;
334 (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),
335 // Inside the scope of ~!_exceptionsBag!~ replace:
336 // _exceptionsBag.Add(exception);
337 // _exceptionsBag.push_back(exception);
338 (new Regex(@"(?<scope>\/~(?<fieldName>[_a-zA-Z0-9]+)~\*\/)(?<separator>.\|\\n)(?<befor
    ↪ e>((?!\/~\k<fieldName>~\*\/)(.\\n))*?\k<fieldName>\\.Add"),
    ↪ "${scope}${separator}${before}${fieldName}.push_back", 10),
339 // Remove scope borders.
340 // /*~_exceptionsBag~*/
341 //
342 (new Regex(@"\/~[_a-zA-Z0-9<>:~\*\/]"), "", 0),
343 // Insert scope borders.
344 // class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
345 // class IgnoredExceptions { /*~_exceptionsBag~*/ ... private: static std::mutex
    ↪ _exceptionsBag_mutex;
346 (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),
347 // Inside the scope of ~!_exceptionsBag!~ replace:
348 // return std::vector<std::exception>(_exceptionsBag);
349 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
    ↪ std::vector<std::exception>(_exceptionsBag);

```



```

350 (new Regex(@"(?<scope>/\~*(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ e>((?<!/\\*~\k<fieldName>~\*/)(.\|\\n))*?){(?<after>((?!lock_guard)[^{};\\r\\n])*\\k<f
    ↪ ieldName>[~;}\\r\\n*};)"), "${scope}${separator}${before}{
    ↪ std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", 10),
351 // Inside the scope of ~!_exceptionsBag!~ replace:
352 // _exceptionsBag.Add(exception);
353 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \\r\\n
    ↪ _exceptionsBag.Add(exception);
354 (new Regex(@"(?<scope>/\~*(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ e>((?<!/\\*~\k<fieldName>~\*/)(.\|\\n))*?){(?<after>((?!lock_guard)([^{};]\\|\\n))*?\\r\\n
    ↪ ?\\n(?<indent>[ \\t])*\\k<fieldName>[~;}\\r\\n*};)"),
    ↪ "${scope}${separator}${before}{\" + Environment.NewLine +
    ↪ \"${indent}std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}\", 10),
355 // Remove scope borders.
356 // /*~_exceptionsBag~*/
357 //
358 (new Regex(@"/*~[_a-zA-Z0-9]+~\*/"), "", 0),
359 // Insert scope borders.
360 // class IgnoredExceptions { ... public: static inline
    ↪ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
    ↪ ExceptionIgnored = OnExceptionIgnored;
361 // class IgnoredExceptions {/*~ExceptionIgnored~*/ ... public: static inline
    ↪ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
    ↪ ExceptionIgnored = OnExceptionIgnored;
362 (new Regex(@"(?<classDeclarationBegin>\\r?\\n(?<indent>[ \\t ]*)class [^\\{\\r\\n]+\\r\\n[ \\t
    ↪ ]*}{(?<middle>((?!class)\\.\\|\\n)+)?(?<eventDeclaration>(?<access>(private|protected|
    ↪ public): )static inline
    ↪ Platform::Delegates::MulticastDelegate<(?<argumentType>[~;\\r\\n]+)>
    ↪ (?<name>[_a-zA-Z0-9]+) = (?<defaultDelegate>[_a-zA-Z0-9]+);)"),
    ↪ "${classDeclarationBegin}/*~${name}~*/${middle}${eventDeclaration}", 0),
363 // Inside the scope of ~!ExceptionIgnored!~ replace:
364 // ExceptionIgnored.Invoke(NULL, exception);
365 // ExceptionIgnored(NULL, exception);
366 (new Regex(@"(?<scope>/\~*(?<eventName>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ >((?<!/\\*~\k<eventName>~\*/)(.\|\\n))*?\\k<eventName>\\.Invoke"),
    ↪ "${scope}${separator}${before}${eventName}", 10),
367 // Remove scope borders.
368 // /*~ExceptionIgnored~*/
369 //
370 (new Regex(@"/*~[_a-zA-Z0-9]+~\*/"), "", 0),
371 // Insert scope borders.
372 // auto added = new StringBuilder();
373 // /*~sb~*/std::string added;
374 (new Regex(@"(auto|\\(System\\.Text\\.)?StringBuilder) (?<variable>[_a-zA-Z0-9]+) = new
    ↪ (System\\.Text\\.)?StringBuilder\\(\\);)", "/*~${variable}~*/std::string
    ↪ ${variable};", 0),
375 // static void Indent(StringBuilder sb, int level)
376 // static void Indent(/*~sb~*/StringBuilder sb, int level)
377 (new Regex(@"(?<start>, \\|\\(\\(System\\.Text\\.)?StringBuilder
    ↪ (?<variable>[_a-zA-Z0-9]+)(?<end>, \\|\\))", "${start}/*~${variable}~*/std::string&
    ↪ ${variable}${end}", 0),
378 // Inside the scope of ~!added!~ replace:
379 // sb.ToString()
380 // sb.data()
381 (new Regex(@"(?<scope>/\~*(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ ((?<!/\\*~\k<variable>~\*/)(.\|\\n))*?\\k<variable>\\.ToString\\(\\)",
    ↪ "${scope}${separator}${before}${variable}.data()", 10),
382 // sb.AppendLine(argument)
383 // sb.append(argument).append('\\n')
384 (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(${argument}).append(1, '\\n')",
    ↪ 10),
385 // sb.Append('\\t', level);
386 // sb.append(level, '\\t');
387 (new Regex(@"(?<scope>/\~*(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ ((?<!/\\*~\k<variable>~\*/)(.\|\\n))*?\\k<variable>\\.Append\\('(?<character>[^\\r\\n]
    ↪ +)'\\', (?<count>[^\\),\\r\\n]+)\\)",
    ↪ "${scope}${separator}${before}${variable}.append(${count}, '${character}'))", 10),
388 // sb.Append(argument)
389 // sb.append(argument)
390 (new Regex(@"(?<scope>/\~*(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ ((?<!/\\*~\k<variable>~\*/)(.\|\\n))*?\\k<variable>\\.Append\\((?<argument>[^\\),\\r\\n]
    ↪ +)\\)", "${scope}${separator}${before}${variable}.append(${argument})",
    ↪ 10),

```



```

391 // Remove scope borders.
392 // /*~sb~/
393 //
394 (new Regex(@"/*~[a-zA-Z0-9]+~*/"), "", 0),
395 // Insert scope borders.
396 // auto added = new HashSet<TElement>();
397 // ~!added!~std::unordered_set<TElement> added;
398 (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
    ↳ HashSet<(?<element>[a-zA-Z0-9]+)>\(\(\);",
    ↳ "~!${variable}!~std::unordered_set<${element}> ${variable};", 0),
399 // Inside the scope of ~!added!~ replace:
400 // added.Add(node)
401 // added.insert(node)
402 (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),
403 // Inside the scope of ~!added!~ replace:
404 // added.Remove(node)
405 // added.erase(node)
406 (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),
407 // if (added.insert(node)) {
408 // if (!added.contains(node)) { added.insert(node);
409 (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),
410 // Remove scope borders.
411 // ~!added!~
412 //
413 (new Regex(@"~![a-zA-Z0-9]+!~"), "", 5),
414 // Insert scope borders.
415 // auto random = new System.Random();
416 // std::srand(0);
417 (new Regex(@"[a-zA-Z0-9\\.]+ ([a-zA-Z0-9]+) = new
    ↳ (System\.)?Random\((([a-zA-Z0-9]+)\)", "~!$!~std::srand($3);", 0),
418 // Inside the scope of ~!random!~ replace:
419 // random.Next(1, N)
420 // (std::rand() % N) + 1
421 (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),
422 // Remove scope borders.
423 // ~!random!~
424 //
425 (new Regex(@"~![a-zA-Z0-9]+!~"), "", 5),
426 // Insert method body scope starts.
427 // void PrintNodes(TElement node, StringBuilder sb, int level) {
428 // void PrintNodes(TElement node, StringBuilder sb, int level) { /*method-start*/
429 (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),
430 // Insert method body scope ends.
431 // { /*method-start*/...}
432 // { /*method-start*/.../*method-end*/}
433 (new Regex(@"\{ /*method-start*/ (?<body>((?<bracket>\{) | (?<-bracket>\}) | [^\{\}])* ) +
    ↳ \} )", "{ /*method-start*/ ${body} /*method-end*/ }",
    ↳ 0),
434 // Inside method bodies replace:
435 // GetFirst(
436 // this->GetFirst(
437 (new Regex(@"(?<separator>(\(| \|([\\W]) |return ))(?<!(->|\\*
    ↳ ))(?<method>(?!sizeof)[a-zA-Z0-9]+)\(((?!\\) \{)\"",
    ↳ "${separator}this->${method}(", 1),
438 (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),
439 // Remove scope borders.
440 // /*method-start*/
441 //

```

```

442 (new Regex(@"\/.*method-(start|end)\*/"), "", 0),
443 // Insert scope borders.
444 // const std::exception& ex
445 // const std::exception& ex/*~ex~/
446 (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?exception&?
    ↳ (?<variable>[_a-zA-Z0-9]+)))(?<after>\W)"),
    ↳ "${before}${variableDefinition}/*~${variable}*/${after}", 0),
447 // Inside the scope of ~!ex!~ replace:
448 // ex.Message
449 // ex.what()
450 (new Regex(@"(?<scope>\/.*~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↳ >((?!\/\.*~\k<variable>\*/)(.\|\\n))*?)\k<variable>\.Message"),
    ↳ "${scope}${separator}${before}${variable}.what()", 10),
451 // Remove scope borders.
452 // /*~ex~/
453 //
454 (new Regex(@"\/.*~[_a-zA-Z0-9]+~\*/"), "", 0),
455 // throw new ArgumentNullException(argumentName, message);
456 // throw std::invalid_argument(((std::string)"Argument
    ↳ ").append(argumentName).append(" is null: ").append(message).append("."));
457 (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),
458 // throw new ArgumentException(message, argumentName);
459 // throw std::invalid_argument(((std::string)"Invalid
    ↳ ").append(argumentName).append(" argument: ").append(message).append("."));
460 (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),
461 // throw new ArgumentOutOfRangeException(argumentName, argumentValue,
    ↳ messageBuilder());
462 // 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("."));
463 (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("\.\\"));", 0),
464 // throw new NotSupportedException();
465 // throw std::logic_error("Not supported exception.");
466 (new Regex(@"throw new NotSupportedException\(\);"), "throw std::logic_error(\\"Not
    ↳ supported exception.\\"");", 0),
467 // throw new NotImplementedException();
468 // throw std::logic_error("Not implemented exception.");
469 (new Regex(@"throw new NotImplementedException\(\);"), "throw std::logic_error(\\"Not
    ↳ implemented exception.\\"");", 0),
470 }.Cast<ISubstitutionRule>().ToList();
471
472 public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
473 {
474     // ICounter<int, int> c1;
475     // ICounter<int, int>* c1;
476     (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[~>\r\n]+)?))
    ↳ (?<variable>[_a-zA-Z0-9]+);"), "${abstractType}* ${variable}";", 0),
477     // (expression)
478     // expression
479     (new Regex(@"(\(| )(((a-zA-Z0-9_\*:)+)\(| |;|\\))"), "$1$2$3", 0),
480     // (method(expression))
481     // method(expression)
482     (new Regex(@"(?<firstSeparator>(\(|
    ↳ ))\((?<method>[a-zA-Z0-9_\*:]+)\((?<expression>((?<parenthesis>\(| )(?<-parent
    ↳ hesis>\\)|[a-zA-Z0-9_\*:]+)(?(parenthesis)(?!))\\)\)(?<lastSeparator>(|
    ↳ |;|\\))")", "${firstSeparator}${method}(${expression})${lastSeparator}", 0),
483     // return ref _elements[node];
484     // return &_elements[node];
485     (new Regex(@"return ref ([a-zA-Z0-9]+)\([([a-zA-Z0-9_\*:]+)\];"), "return &$1[$2];",
    ↳ 0),

```

```

486 // null
487 // nullptr
488 (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"|["~""\r\n])*~""\r\n)*)(?<=\\W)null_
  ↳ (?<after>\\W)"), "${before}nullptr${after}",
  ↳ 10),
489 // default
490 // 0
491 (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"|["~""\r\n])*~""\r\n)*)(?<=\\W)defa_
  ↳ ult(?<after>\\W)"), "${before}0${after}",
  ↳ 10),
492 // object x
493 // void *x
494 (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"|["~""\r\n])*~""\r\n)*)(?<=\\W)([O|_
  ↳ o]bject|System\\.Object) (?<after>\\w)"), "${before}void *${after}",
  ↳ 10),
495 // <object>
496 // <void*>
497 (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"|["~""\r\n])*~""\r\n)*)(?<=\\W)(?!_
  ↳ \\w)([O|o]bject|System\\.Object)(?<after>\\W)"), "${before}void*${after}",
  ↳ 10),
498 // ArgumentException
499 // std::invalid_argument
500 (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"|["~""\r\n])*~""\r\n)*)(?<=\\W)(Sys_
  ↳ tem\\.)?ArgumentException(?<after>\\W)"),
  ↳ "${before}std::invalid_argument${after}", 10),
501 // struct Range<T> : IEquatable<Range<T>> {
502 // struct Range<T> {
503 (new Regex(@"(?<before>(struct|class) (?<type>[a-zA-Z0-9]+(<[~\\n]+)?)) :
  ↳ IEquatable<\\k<type>>(?!<after>(\\s|\\n)*{)"), "${before}${after}", 0),
504 // #region Always
505 //
506 (new Regex(@"(~/\\r?\\n)[ \\t]*#(region|endregion)[~\\r\\n]*(\\r?\\n|$)"), "", 0),
507 // //define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
508 //
509 (new Regex(@"\\/\\/ [ \\t]*#define[ \\t]+[_a-zA-Z0-9]+[ \\t]*"), "", 0),
510 // #if USEARRAYPOOL\\r\\n#endif
511 //
512 (new Regex(@"#if [a-zA-Z0-9]+\\s+#endif"), "", 0),
513 // [Fact]
514 //
515 (new Regex(@"(?<firstNewLine>\\r?\\n|\\A)(?<indent>[\\t
  ↳ ]+)[[a-zA-Z0-9]+(\\((?!<expression>(?!<parenthesis>\\)|(?<-parenthesis>\\))|(?<)\\r_
  ↳ \\n)*+)(?!<parenthesis>(?!))\\)\\)?\\[ \\t]*(\\r?\\n\\k<indent>)?"),
  ↳ "${firstNewLine}${indent}", 5),
516 // \\n ... namespace
517 // namespace
518 (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+namespace"), "$1namespace", 0),
519 // \\n ... class
520 // class
521 (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+class"), "$1class", 0),
522 // \\n\\n\\n
523 // \\n\\n
524 (new Regex(@"\\r?\\n[ \\t]*\\r?\\n[ \\t]*\\r?\\n"), Environment.NewLine +
  ↳ Environment.NewLine, 50),
525 // {\\n\\n
526 // {\\n
527 (new Regex(@"{[ \\t]*\\r?\\n[ \\t]*\\r?\\n"), "{" + Environment.NewLine, 10),
528 // \\n\\n}
529 // {\\n
530 (new Regex(@"\\r?\\n[ \\t]*\\r?\\n(?<end>[ \\t]*)"), Environment.NewLine + "${end}", 10),
531 }.Cast<ISubstitutionRule>().ToList();
532
533 public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
  ↳ base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
534
535 public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
536 }
537 }

```

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

```

1 using Xunit;
2
3 namespace Platform.RegularExpressions.Transformer.CSharpToCpp.Tests
4 {
5     public class CSharpToCppTransformerTests
6     {
7         [Fact]

```

```

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

```

Index

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