

1.1 ./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

```

```

56 // private const int MaxPath = 92;
57 // static const int MaxPath = 92;
58 (new Regex(@"private (const|static readonly) ([a-zA-Z0-9]+) ([_a-zA-Z0-9]+) =
→ ([~;\r\n]+);"), "static const $2 $3 = $4;", null, 0),
59 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
→ TArgument : class
60 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
61 (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),
62 // protected virtual
63 // virtual
64 (new Regex(@"protected virtual"), "virtual", null, 0),
65 // protected abstract TElement GetFirst();
66 // virtual TElement GetFirst() = 0;
67 (new Regex(@"protected abstract ([~;\r\n]+);"), "virtual $1 = 0;", null, 0),
68 // TElement GetFirst();
69 // virtual TElement GetFirst() = 0;
70 (new Regex(@"([\r\n]+[ ]+)((?!return)[a-zA-Z0-9]+ [a-zA-Z0-9]+\(([\~\r\n]*\)) (
→ ]*[\r\n]+)"), "$1virtual $2 = 0$3", null, 1),
71 // public virtual
72 // virtual
73 (new Regex(@"public virtual"), "virtual", null, 0),
74 // protected readonly
75 //
76 (new Regex(@"protected readonly "), "", null, 0),
77 // protected readonly TreeElement[] _elements;
78 // TreeElement _elements[N];
79 (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+)([\\[\\]]+
→ ([_a-zA-Z0-9]+);"), "$2 $4[N];", null, 0),
80 // protected readonly TElement Zero;
81 // TElement Zero;
82 (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+) ([_a-zA-Z0-9]+);"), "$2
→ $3;", null, 0),
83 // private
84 //
85 (new Regex(@"(\W)(private|protected|public|internal) "), "$1", null, 0),
86 // static void NotImplementedException(ThrowExtensionRoot root) => throw new
→ NotImplementedException();
87 // static void NotImplementedException(ThrowExtensionRoot root) { return throw new
→ NotImplementedException(); }
88 (new Regex(@"(^\\s+)(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) {
→ throw$8; }", null, 0),
89 // SizeBalancedTree(int capacity) => a = b;
90 // SizeBalancedTree(int capacity) { a = b; }
91 (new Regex(@"(^\\s+)(template \\<[~>\\r\\n]+> )?(static )?(override )?(void
→ )?([a-zA-Z0-9]+)\\(((\\~\\(\\r\\n)*\\)\\s+=>\\s+([~;\r\n]+);"), "$1$2$3$4$5$6($7) { $8;
→ }", null, 0),
92 // int SizeBalancedTree(int capacity) => a;
93 // int SizeBalancedTree(int capacity) { return a; }
94 (new Regex(@"(^\\s+)(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) {
→ return $8; }", null, 0),
95 // () => Integer<TElement>.Zero,
96 // () { return Integer<TElement>.Zero; },
97 (new Regex(@"\\(\\)\\s+=>\\s+([~;\r\n]+?);"), "()" { return $1; },",", null, 0),
98 // => Integer<TElement>.Zero;
99 // { return Integer<TElement>.Zero; }
100 (new Regex(@"\\)\\s+=>\\s+([~;\r\n]+?);"), ") { return $1; }", null, 0),
101 // () { return avlTree.Count; }
102 // [&]()-> auto { return avlTree.Count; }
103 (new Regex(@"\\(\\) { return ([~;\r\n]+); }"), ", [&]()-> auto { return $1; }",
→ null, 0),
104 // Count => GetSizeOrZero(Root);
105 // GetCount() { return GetSizeOrZero(Root); }
106 (new Regex(@"(\W)([A-Z][a-zA-Z]+)\s+=>\s+([~;\r\n]+);"), "$1Get$2() { return $3; }",
→ null, 0),
107 // Func<TElement> treeCount
108 // std::function<TElement()> treeCount
109 (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", null,
→ 0),
110 // Action<TElement> free
111 // std::function<void(TElement)> free
112 (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<void($1)> $2",
→ null, 0),

```

```
// Predicate<TArgument> predicate
// std::function<bool(TArgument)> predicate
(new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)", "std::function<bool($1)>
→ $2", null, 0),
// var
// auto
(new Regex(@"(\W)var(\W)", "$1auto$2", null, 0),
// unchecked
//
(new Regex(@"[\r\n]{2}\s*?unchecked\s*?$)", "", null, 0),
// throw new InvalidOperationException
// throw std::runtime_error
(new Regex(@"throw new (InvalidOperationException|Exception)", "throw
→ std::runtime_error", null, 0),
// void RaiseExceptionIgnoredEvent(Exception exception)
// void RaiseExceptionIgnoredEvent(const std::exception& exception)
(new Regex(@"\(|\) (System\.Exception|Exception) (\)|)", "$1const
→ std::exception&$3", null, 0),
// EventHandler<Exception>
// EventHandler<std::exception>
(new Regex(@"(\W) (System\.Exception|Exception) (\W)", "$1std::exception$3", null, 0),
// override void PrintNode(TElement node, StringBuilder sb, int level)
// void PrintNode(TElement node, StringBuilder sb, int level) override
(new Regex(@"override ([a-zA-Z0-9 \*\+]) (\([-]\r\n)+?)", "$1$2 override", null,
→ 0),
// string
// const char*
(new Regex(@"(\W]string(\W)", "$1const char*$2", null, 0),
// sbyte
// std::int8_t
(new Regex(@"(\W)sbyte(\W)", "$1std::int8_t$2", null, 0),
// uint
// std::uint32_t
(new Regex(@"(\W)uint(\W)", "$1std::uint32_t$2", null, 0),
// char*[] args
// char* args[]
(new Regex(@"(_[a-zA-Z0-9:\*]?)\[ \] ([a-zA-Z0-9]+)", "$1 $2[]", null, 0),
// @object
// object
(new Regex(@"@(_[a-zA-Z0-9]+)", "$1", null, 0),
// using Platform.Numbers;
//
(new Regex(@"([\r\n]{2}|^)\s*?using [\.a-zA-Z0-9]+\s*?$)", "", null, 0),
// struct TreeElement { }
// struct TreeElement { };
(new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+)\{([\sa-zA-Z0-9;:_]+?)\}([~;])", "$1
→ $2$3{$4};$5", null, 0),
// class Program { }
// class Program { };
(new Regex(@"(struct|class) ([a-zA-Z0-9]+)[^\r\n]*([\r\n]+(?<indentLevel>[\t
→ ]*)?)\{([\S\s]+?\r\n+\k<indentLevel>\})\}([~;]|$)", "$1 $2$3{$4};$5", null, 0),
// class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
// class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
(new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)", "class $1 : public $2", null,
→ 0),
// class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
// class IProperty : public ISetter<TValue, TObject>, IProvider<TValue, TObject>
(new Regex(@"(?<before>class [a-zA-Z0-9]+ : ((public [a-zA-Z0-9]+(<[a-zA-Z0-9
→ ,]+>)?, )+)? (?<inheritedType>(?!public) [a-zA-Z0-9]+(<[a-zA-Z0-9
→ ,]+>)? (?<after>([a-zA-Z0-9]+(?:!|\r\n)+)))", "${before}public
→ ${inheritedType}${after}", null, 10),
// 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}", null,
→ 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}", null, 70),
// Remove scope borders.
// ~!root!~
```

```

174 //
175 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
176 // ref auto root = ref
177 // ref auto root =
178 (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)"), "$1* $2 =$3", null, 0),
179 // *root = ref left;
180 // root = left;
181 (new Regex(@"\*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", null, 0),
182 // (ref left)
183 // (left)
184 (new Regex(@"\(\ref ([a-zA-Z0-9]+)(\)|\(|,)\)", "($1$2", null, 0),
185 // ref TElement
186 // TElement*
187 (new Regex(@"( |\()ref ([a-zA-Z0-9]+) ", "$1$2* ", null, 0),
188 // ref sizeBalancedTree.Root
189 // &sizeBalancedTree->Root
190 (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)"), "&$1->$2", null, 0),
191 // ref GetElement(node).Right
192 // &GetElement(node)->Right
193 (new Regex(@"ref ([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"),
194     ↳ "$&1($2)->$3", null, 0),
195 // GetElement(node).Right
196 // GetElement(node)->Right
197 (new Regex(@"([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"), "$1($2)->$3",
198     ↳ null, 0),
199 // [Fact]\npublic static void SizeBalancedTreeMultipleAttachAndDetachTest()
200 // TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
201 (new Regex(@"\[Fact\]\[s\n]+(static )?void ([a-zA-Z0-9]+)\(\)"), "TEST_METHOD($2)",
202     ↳ null, 0),
203 // class TreesTests
204 // TEST_CLASS(TreesTests)
205 (new Regex(@"class ([a-zA-Z0-9]+)Tests"), "TEST_CLASS($1)", null, 0),
206 // Assert.Equal
207 // Assert::AreEqual
208 (new Regex(@"Assert\.Equal"), "Assert::AreEqual", null, 0),
209 // $"Argument {argumentName} is null."
210 // ((std::string)"Argument ").append(argumentName).append(" is null.").data()
211 (new Regex(@"\$"(?<left>\\"|["~"\r\n])*{(?<expression>[a-zA-Z0-9]+)}{(?<right>\\"|
212     ↳ "\"|["~"\r\n])*}""),
213     ↳ "((std::string)$\"${left}\").append(${expression}).append(\"${right}\").data()",
214     ↳ null, 10),
215 // $"
216 // "
217 (new Regex(@"\$"""), "\"", null, 0),
218 // Console.WriteLine("...")
219 // printf("...\n")
220 (new Regex(@"Console\.WriteLine\(\"([~""\r\n]+)""\)"), "printf(\"$1\\n\")", null, 0),
221 // TElement Root;
222 // TElement Root = 0;
223 (new Regex(@"(\r?\n[\t ]+)([a-zA-Z0-9:])(?<return>) ([a-zA-Z0-9]+);"), "$1$2 $3 =
224     ↳ 0;", null, 0),
225 // TreeElement _elements[N];
226 // TreeElement _elements[N] = { {0} };
227 (new Regex(@"(\r?\n[\t ]+)([a-zA-Z0-9:])([a-zA-Z0-9]+)\[([a-zA-Z0-9]+)\];"),
228     ↳ "$1$2 $3[$4] = { {0} };", null, 0),
229 // auto path = new TElement[MaxPath];
230 // TElement path[MaxPath] = { {0} };
231 (new Regex(@"(\r?\n[\t ]+)[a-zA-Z0-9]+ ([a-zA-Z0-9]+) = new
232     ↳ ([a-zA-Z0-9]+)\[([a-zA-Z0-9]+)\];"), "$1$3 $2[$4] = { {0} };", null, 0),
233 // Insert scope borders.
234 // auto added = new StringBuilder();
235 // /*~sb~/std::string added;
236 (new Regex(@"(auto|(\System\.Text\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new
237     ↳ (\System\.Text\.)?StringBuilder\(\)");, "/*~${variable}~/std::string
238     ↳ ${variable};", null, 0),
239 // static void Indent(StringBuilder sb, int level)
240 // static void Indent(/*~sb~/StringBuilder sb, int level)
241 (new Regex(@"(?<start>, |\()(\System\.Text\.)?StringBuilder
242     ↳ (?<variable>[a-zA-Z0-9]+)(?<end>,\|\)"), "${start}/*~${variable}~/std::string&
243     ↳ ${variable}${end}", null, 0),
244 // Inside the scope of ~!added!~ replace:
245 // sb.ToString()
246 // sb.data()
247 (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~*/)(?<separator>.\|\\n)(?<before>
248     ↳ ((?!/*~\k<variable>~*/)(.\|\\n))*?)\k<variable>\.ToString\(\)"),
249     ↳ "${scope}${separator}${before}${variable}.data()", null, 10),

```

```

235 // sb.AppendLine(argument)
236 // sb.append(argument).append('\n')
237 (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('\n')",
    ↳ null, 10),
238 // sb.Append('\t', level);
239 // sb.append(level, '\t');
240 (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),
241 // sb.Append(argument)
242 // sb.append(argument)
243 (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),
244 // Remove scope borders.
245 // /*~sb~*/
246 //
247 (new Regex(@"/*~(?<pointer>[a-zA-Z0-9]+)~*/"), "", null, 0),
248 // Insert scope borders.
249 // auto added = new HashSet<TElement>();
250 // ~!added!~std::unordered_set<TElement> added;
251 (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
    ↳ HashSet<(?<element>[a-zA-Z0-9]+)>\(\(\)\);",
    ↳ "~!${variable}!~std::unordered_set<${element}> ${variable};", null, 0),
252 // Inside the scope of ~!added!~ replace:
253 // added.Add(node)
254 // added.insert(node)
255 (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),
256 // Inside the scope of ~!added!~ replace:
257 // added.Remove(node)
258 // added.erase(node)
259 (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),
260 // if (added.insert(node)) {
261 // if (added.contains(node)) { added.insert(node);
262 (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),
263 // Remove scope borders.
264 // ~!added!~
265 //
266 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
267 // Insert scope borders.
268 // auto random = new System.Random(0);
269 // std::srand(0);
270 (new Regex(@"[a-zA-Z0-9\.] + ([a-zA-Z0-9]+) = new
    ↳ (System\.)?Random\(\([a-zA-Z0-9]+\)\);", "~!$!~std::srand($3);", null, 0),
271 // Inside the scope of ~!random!~ replace:
272 // random.Next(1, N)
273 // (std::rand() % N) + 1
274 (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),
275 // Remove scope borders.
276 // ~!random!~
277 //
278 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
279 // Insert method body scope starts.
280 // void PrintNodes(TElement node, StringBuilder sb, int level) {
281 // void PrintNodes(TElement node, StringBuilder sb, int level) {/*method-start*/
282 (new Regex(@"(?<start>\r?\n[\t ]+)(?<prefix>((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),

```

```

283 // Insert method body scope ends.
284 // {/*method-start*/...}
285 // {/*method-start*/.../*method-end*/}
286 (new Regex(@"{\/\*method-start\/(?(<body>((?<bracket>\{)|(?(<-bracket>\})|[\^\{\}]*))+)
    ↳ \}"}), "{/*method-start*/{body}/*method-end*/}", null,
    ↳ 0),
287 // Inside method bodies replace:
288 // GetFirst(
289 // this->GetFirst(
290 // (new Regex(@"(?<separator>(\(| |([\\W]) |return ))(?!(->|\\*
    ↳ ))(?(method>(?!sizeof)[a-zA-Z0-9]+)\\((?!\\) \{)"),
    ↳ "{$separator}this->${method}(", null, 1),
291 (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),
292 // Remove scope borders.
293 // /*method-start*/
294 //
295 (new Regex(@"\/\*method-(start|end)\/"), "", null, 0),
296 // throw new ArgumentNullException(argumentName, message);
297 // throw std::invalid_argument(((std::string)"Argument
    ↳ ").append(argumentName).append(" is null: ").append(message).append("."));
298 (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),
299 // throw new ArgumentException(message, argumentName);
300 // throw std::invalid_argument(((std::string)"Invalid
    ↳ ").append(argumentName).append(" argument: ").append(message).append("."));
301 (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),
302 // throw new NotSupportedException();
303 // throw std::logic_error("Not supported exception.");
304 (new Regex(@"throw new NotSupportedException\\(\\)");", "throw std::logic_error("\\Not
    ↳ supported exception.\\)");", null, 0),
305 // throw new NotImplementedException();
306 // throw std::logic_error("Not implemented exception.");
307 (new Regex(@"throw new NotImplementedException\\(\\)");", "throw std::logic_error("\\Not
    ↳ implemented exception.\\)");", null, 0),
308 }.Cast<ISubstitutionRule>().ToList();
309
310 public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
311 {
312 // ICounter<int, int> c1;
313 // ICounter<int, int>* c1;
314 (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^\r\n]+>)?)
    ↳ (?<variable>[_a-zA-Z0-9]+);", "${abstractType}* ${variable}";", null, 0),
315 // (expression)
316 // expression
317 (new Regex(@"\\(|)\\(((\\[a-zA-Z0-9_\\*:]*)\\(| |;|\\))", "$1$2$3", null, 0),
318 // (method(expression))
319 // method(expression)
320 (new Regex(@"(?<firstSeparator>\\(|
    ↳ ))\\((?<method>[a-zA-Z0-9_\\->\\*:]*)\\((?<expression>((?<parenthesis>\\(|(?(<-parent
    ↳ hesis>\\)|[a-zA-Z0-9_\\->\\*:]*)+(?(parenthesis)(?!))\\)\\) (?(lastSeparator>\\(|
    ↳ |;|\\)))", "${firstSeparator}${method}(${expression})${lastSeparator}", null, 0),
321 // return ref _elements[node];
322 // return &elements[node];
323 (new Regex(@"return ref ([_a-zA-Z0-9]+)\\(((\\[a-zA-Z0-9_\\*:]*)\\(| |;|\\))", "return &$1[$2];",
    ↳ null, 0),
324 // null
325 // NULL
326 (new Regex(@"(?<before>\\r?\\n[^\r\n]*(""(\\\\"|^\r\n))*"^\r\n)*)(?<=\\W)null
    ↳ (?<after>\\W)", "${before}NULL${after}", null,
    ↳ 10),
327 // default
328 // 0
329 (new Regex(@"(?<before>\\r?\\n[^\r\n]*(""(\\\\"|^\r\n))*"^\r\n)*)(?<=\\W)defa
    ↳ ult(?<after>\\W)", "${before}0${after}", null,
    ↳ 10),
330 // #region Always
331 //

```

```

332         (new Regex(@"(?:\r?\n)[ \t]*#(region|endregion)[^r\n]*(\r?\n|$)"), "", null, 0),
333         // //define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
334         //
335         (new Regex(@"\\/\\/ [ \t]*#define [ \t]+[_a-zA-Z0-9]+[ \t]*"), "", null, 0),
336         // #if USEARRAYPOOL\r\n#endif
337         //
338         (new Regex(@"#if [a-zA-Z0-9]+\s#endif"), "", null, 0),
339         // [Fact]
340         //
341         (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),
342         // \n ... namespace
343         // namespace
344         (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+namespace"), "$1namespace", null, 0),
345         // \n ... class
346         // class
347         (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+class"), "$1class", null, 0),
348     }.Cast<ISubstitutionRule>().ToList();
349
350     public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
351         ↪ base(FirstStage.Concat(extraRules).Concat>LastStage).ToList()) { }
352
353     public CSharpToCppTransformer() : base(FirstStage.Concat>LastStage).ToList()) { }
354 }

```

1.2 ./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 HelloWorldTest()
9          {
10             const string helloWorldCode = @"using System;
11 class Program
12 {
13     public static void Main(string[] args)
14     {
15         Console.WriteLine(""Hello, world!"");
16     }
17 }";
18             const string expectedResult = @"class Program
19 {
20     public:
21     static void Main(const char* args[])
22     {
23         printf(""Hello, world!\n"");
24     }
25 };";
26             var transformer = new CSharpToCppTransformer();
27             var actualResult = transformer.Transform(helloWorldCode, new Context(null));
28             Assert.Equal(expectedResult, actualResult);
29         }
30     }
31 }

```

Index

./Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs, 7
./Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1