

## 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 = $4;", null, 0),
59 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
→ TArgument : class
60 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
61 (new Regex(@"(?<before> [a-zA-Z]+\\((([a-zA-Z *],+ |)) (?<type>[a-zA-Z]+) (?<after>([
→ [a-zA-Z *],+))\\)) [ \r\n]+where \<type> : class)", "${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),

```

```

113 // Predicate<TArgument> predicate
114 // std::function<bool(TArgument)> predicate
115 (new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<bool($1)>
    → $2", null, 0),
116 // var
117 // auto
118 (new Regex(@"(\W)var(\W)"), "$1auto$2", null, 0),
119 // unchecked
120 //
121 (new Regex(@"[\r\n]{2}\s*?unchecked\s*?$"), "", null, 0),
122 // $"Argument {argumentName} is null."
123 // ((std::string)"Argument ").append(argumentName).append(" is null.")
124 (new Regex(@"\$""(?<left>\\""|[""'\r\n])*{(?<expression>[_a-zA-Z0-9]+)}{(?<right>\\
    → \""|[""'\r\n])*}"""),
    → "((std::string)$\"${left}\").append(${expression}).append(\"${right}\")", null,
    → 10),
125 // $"
126 // "
127 (new Regex(@"\$"""), "\"", null, 0),
128 // Console.WriteLine(...)
129 // printf(...)
130 (new Regex(@"Console\.WriteLine\\(\"\"([~""'\r\n]+)""\\)"), "printf(\"$1\\n\")", null, 0),
131 // throw new InvalidOperationException
132 // throw std::runtime_error
133 (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
    → std::runtime_error", null, 0),
134 // void RaiseExceptionIgnoredEvent(Exception exception)
135 // void RaiseExceptionIgnoredEvent(const std::exception& exception)
136 (new Regex(@"\\(|, )(System\\.Exception|Exception)( |\\)"), "$1const
    → std::exception&$3", null, 0),
137 // EventHandler<Exception>
138 // EventHandler<std::exception>
139 (new Regex(@"(\\W)(System\\.Exception|Exception)(\\W)"), "$1std::exception$3", null, 0),
140 // override void PrintNode(TElement node, StringBuilder sb, int level)
141 // void PrintNode(TElement node, StringBuilder sb, int level) override
142 (new Regex(@"override ([a-zA-Z0-9 \\\*\\+]+)\\((~""'\r\n|\\?\\)\\)"), "$1$2 override", null,
    → 0),
143 // string
144 // char*
145 (new Regex(@"(\\W)string(\\W)"), "$1char*$2", null, 0),
146 // sbyte
147 // std::int8_t
148 (new Regex(@"(\\W)sbyte(\\W)"), "$1std::int8_t$2", null, 0),
149 // uint
150 // std::uint32_t
151 (new Regex(@"(\\W)uint(\\W)"), "$1std::uint32_t$2", null, 0),
152 // char*[] args
153 // char* args[]
154 (new Regex(@"(_a-zA-Z0-9:\\\\*\\?)\\[\\] ([a-zA-Z0-9]+)"), "$1 $2[]", null, 0),
155 // @object
156 // object
157 (new Regex(@"@(_a-zA-Z0-9)+"), "$1", null, 0),
158 // using Platform.Numbers;
159 //
160 (new Regex(@"([\\r\\n]{2}|~)\\s*?using [\\_a-zA-Z0-9]+;\\s*?$"), "", null, 0),
161 // struct TreeElement { }
162 // struct TreeElement { };
163 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\\s+){([\\sa-zA-Z0-9;:_]+?)}{([~;])")", "$1
    → $2$3{$4};$5", null, 0),
164 // class Program { }
165 // class Program { };
166 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)[~\\r\\n]*([\\r\\n]+(?<indentLevel>[\\t
    → ]*)?){([\\S\\s]+?\\r\\n|\\k<indentLevel>)}{([~;]|$)"), "$1 $2$3{$4};$5", null, 0),
167 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
168 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
169 (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)"), "class $1 : public $2", null,
    → 0),
170 // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
171 // class IProperty : public ISetter<TValue, TObject>, IProvider<TValue, TObject>
172 (new Regex(@"(?<before>class [a-zA-Z0-9]+ : ((public [a-zA-Z0-9]+(<[a-zA-Z0-9
    → ,]+>)?, )+)?)(?<inheritedType>(?!public) [a-zA-Z0-9]+(<[a-zA-Z0-9
    → ,]+>)?)(?<after>(, [a-zA-Z0-9]+(?!>)|[\\r\\n]+))"), "${before}public
    → ${inheritedType}${after}", null, 10),
173 // Insert scope borders.
174 // ref TElement root
175 // ~!root!~ref TElement root

```

```

(new Regex(@"(?<definition>(?!<pointer>[a-zA-Z0-9]+)~ref [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>(?!<ref>\\|,| |=)(?!<before>((?!~!<pointer>~)(.|\n))*?)?(?!<prefix>(\\W
→ |\\())\k<pointer>(?!<suffix>(\\|;|,))"),
→ "${definition}${before}${prefix}*${pointer}${suffix}", null, 70),
// Remove scope borders.
// ~!root!~
//
(new Regex(@"~!(?!<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
// ref auto root = ref
// ref auto root =
(new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\\W)"), "$1* $2 = $3", null, 0),
// *root = ref left;
// root = left;
(new Regex(@"\\*(\\[a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\\W)"), "$1 = $2$3", null, 0),
// (ref left)
// (left)
(new Regex(@"\\(ref ([a-zA-Z0-9]+)(\\|\\(|,))"), "($1$2", null, 0),
// ref TElement
// TElement*
(new Regex(@"(\\|\\()ref ([a-zA-Z0-9]+) ", "$1$2* ", null, 0),
// ref sizeBalancedTree.Root
// &sizeBalancedTree->Root
(new Regex(@"ref ([a-zA-Z0-9]+)\\.([a-zA-Z0-9\\*]+)"), "&$1->$2", null, 0),
// ref GetElement(node).Right
// &GetElement(node)->Right
(new Regex(@"ref ([a-zA-Z0-9]+)\\((\\[a-zA-Z0-9\\*]+)\\)\\.([a-zA-Z0-9]+)"),
→ "&$1($2)->$3", null, 0),
// GetElement(node).Right
// GetElement(node)->Right
(new Regex(@"(\\[a-zA-Z0-9]+)\\((\\[a-zA-Z0-9\\*]+)\\)\\.([a-zA-Z0-9]+)"), "$1($2)->$3",
→ null, 0),
// [Fact]\\npublic static void SizeBalancedTreeMultipleAttachAndDetachTest()
// TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
(new Regex(@"\\[Fact\\] \\s\\n\\n+(static)?void ([a-zA-Z0-9]+)\\(\\)"), "TEST_METHOD($2)",
→ null, 0),
// class TreesTests
// TEST_CLASS(TreesTests)
(new Regex(@"class ([a-zA-Z0-9]+)Tests"), "TEST_CLASS($1)", null, 0),
// Assert.Equal
// Assert::AreEqual
(new Regex(@"Assert\\.Equal"), "Assert::AreEqual", null, 0),
// TElement Root;
// TElement Root = 0;
(new Regex(@"(\\r?\\n\\t+)([a-zA-Z0-9_]+(?<return>)) ([_a-zA-Z0-9]+);"), "$1$2 $3 =
→ 0;", null, 0),
// TreeElement _elements[N];
// TreeElement _elements[N] = { {0} };
(new Regex(@"(\\r?\\n\\t+)([a-zA-Z0-9]+) ([_a-zA-Z0-9]+)\\[([_a-zA-Z0-9]+)\\];"),
→ "$1$2 $3[$4] = { {0} };", null, 0),
// auto path = new TElement[MaxPath];
// TElement path[MaxPath] = { {0} };
(new Regex(@"(\\r?\\n\\t+)[a-zA-Z0-9]+ ([a-zA-Z0-9]+) = new
→ ([a-zA-Z0-9]+)\\[([_a-zA-Z0-9]+)\\];"), "$1$3 $2[$4] = { {0} };", null, 0),
// Insert scope borders.
// auto added = new HashSet<TElement>();
// ~!added!~std::unordered_set<TElement> added;
(new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
→ HashSet<(?!<element>[a-zA-Z0-9]+)\\(\\);"),
→ "~!${variable}!~std::unordered_set<${element}> ${variable};", null, 0),
// Inside the scope of ~!added!~ replace:
// added.Add(node)
// added.insert(node)
(new Regex(@"(?<scope>~!(?!<variable>[a-zA-Z0-9]+)~)(?!<separator>\\.|\\n)(?!<before>((?!~!<variable>~)(.|\n))*?)\k<variable>\\.Add\\((?!<argument>[a-zA-Z0-9]+)\\)",
→ "${scope}${separator}${before}${variable}.insert(${argument})", null, 10),
// Inside the scope of ~!added!~ replace:
// added.Remove(node)
// added.erase(node)
(new Regex(@"(?<scope>~!(?!<variable>[a-zA-Z0-9]+)~)(?!<separator>\\.|\\n)(?!<before>((?!~!<variable>~)(.|\n))*?)\k<variable>\\.Remove\\((?!<argument>[a-zA-Z0-9]+)\\)",
→ "${scope}${separator}${before}${variable}.erase(${argument})", null, 10),

```

```

236 // if (added.insert(node)) {
237 // if (!added.contains(node)) { added.insert(node);
238 (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),
239 // Remove scope borders.
240 // ~!added!~
241 //
242 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
243 // Insert scope borders.
244 // auto random = new System.Random(0);
245 // std::srand(0);
246 (new Regex(@"[a-zA-Z0-9\.]+ ([a-zA-Z0-9]+) = new
    ↳ (System\.)?Random\((([a-zA-Z0-9]+\)\);", "~!$!~std::srand($3);", null, 0),
247 // Inside the scope of ~!random!~ replace:
248 // random.Next(1, N)
249 // (std::rand() % N) + 1
250 (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),
251 // Remove scope borders.
252 // ~!random!~
253 //
254 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
255 // Insert method body scope starts.
256 // void PrintNodes(TElement node, StringBuilder sb, int level) {
257 // void PrintNodes(TElement node, StringBuilder sb, int level) { /*method-start*/
258 (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*/${end}", null,
    ↳ 0),
259 // Insert method body scope ends.
260 // { /*method-start*/...}
261 // { /*method-start*/... /*method-end*/}
262 (new Regex(@"{ /\*method-start*/(?<body>((?<bracket>\{) | (?<-bracket>\}) | [^\{\}])*+)
    ↳ \}"}", "{ /*method-start*/${body} /*method-end*/", null,
    ↳ 0),
263 // Inside method bodies replace:
264 // GetFirst(
265 // this->GetFirst(
266 (new Regex(@"(?<separator>(\(| |([W]) |return ))(?<!(-->|\\*
    ↳ ))(?<method>(?!sizeof)[a-zA-Z0-9]+\)\(((?!\\) \{) )",
    ↳ "${separator}this->${method}(", null, 1),
267 (new Regex(@"(?<scope> /\*method-start*/)(?<before>((?<!(-->|\\*method-end*/)(.\|\n))*?) (
    ↳ ?<separator>[W] (?<!(-->|\\*method-end*/)(.\|\n))*?) (?<method>(?!sizeof)[a-zA-Z0-9]+\)\(((?!\\)
    ↳ \{) (?<after>(\|\n))*?) (?<scopeEnd> /\*method-end*/) )",
    ↳ "${scope}${before}${separator}this->${method}(${after}${scopeEnd}", null, 100),
268 // Remove scope borders.
269 // /*method-start*/
270 //
271 (new Regex(@" /\*method-(start|end)*/"), "", null, 0),
272 // throw new ArgumentNullException(argumentName, message);
273 // throw std::invalid_argument(((std::string)"Argument
    ↳ ").append(argumentName).append(" is null: ").append(message).append("."));
274 (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),
275 // throw new ArgumentException(message, argumentName);
276 // throw std::invalid_argument(((std::string)"Invalid
    ↳ ").append(argumentName).append(" argument: ").append(message).append("."));
277 (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),
278 // throw new NotSupportedException();
279 // throw std::logic_error("Not supported exception.");
280 (new Regex(@"throw new NotSupportedException\(\);", "throw std::logic_error(\"Not
    ↳ supported exception.\");", null, 0),
281 // throw new NotImplementedException();
282 // throw std::logic_error("Not implemented exception.");

```

```

283         (new Regex(@"throw new NotImplementedException\(\);"), "throw std::logic_error(\"Not
        ↳ implemented exception.\");", null, 0),
284
285     }.Cast<ISubstitutionRule>().ToList();
286
287     public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
288     {
289         // ICounter<int, int> c1;
290         // ICounter<int, int>* c1;
291         (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^\r\n]+>)?)
        ↳ (?<variable>[_a-zA-Z0-9]+);"), "${abstractType}* ${variable};", null, 0),
292         // (expression)
293         // expression
294         (new Regex(@"(\(|\)|)(([a-zA-Z0-9_\*:]*)\(|\)|;|\\)"), "$1$2$3", null, 0),
295         // (method(expression))
296         // method(expression)
297         (new Regex(@"(?<firstSeparator>(\(|
        ↳ ))\((?<method>[a-zA-Z0-9_\*:]*)\((?<expression>((?<parenthesis>\(|(?<-parent
        ↳ hesis>\\)|[a-zA-Z0-9_\*:]*)\((?<parenthesis>(?!))\\)\)(?<lastSeparator>(\(|
        ↳ |;|\\))\)", "${firstSeparator}${method}(${expression})${lastSeparator}", null, 0),
298         // return ref _elements[node];
299         // return &elements[node];
300         (new Regex(@"return ref ([a-zA-Z0-9]+)\([([a-zA-Z0-9_\*:]*)\];"), "return &$1[$2];",
        ↳ null, 0),
301         // null
302         // NULL
303         (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\" | [~""\r\n])*" [~""\r\n]*)(?<=\\W)null
        ↳ (?<after>\\W)", "${before}NULL${after}", null,
        ↳ 10),
304         // default
305         // 0
306         (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\" | [~""\r\n])*" [~""\r\n]*)(?<=\\W)defa
        ↳ ult(?<after>\\W)", "${before}0${after}", null,
        ↳ 10),
307         // #region Always
308         //
309         (new Regex(@"(^\r?\n)[ \t]*#(region|endregion)[^\r\n]*(\r?\n|$)"), "", null, 0),
310         // // #define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
311         //
312         (new Regex(@"\\/\\/ [ \t]*#define[ \t]+[a-zA-Z0-9]+[ \t]*"), "", null, 0),
313         // #if USEARRAYPOOL\r\n#endif
314         //
315         (new Regex(@"#if [a-zA-Z0-9]+\s+#endif"), "", null, 0),
316         // [Fact]
317         //
318         (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),
319         // \n ... namespace
320         // namespace
321         (new Regex(@"(\\S[ \r\n]{1,2})?[ \r\n]+namespace"), "$1namespace", null, 0),
322         // \n ... class
323         // class
324         (new Regex(@"(\\S[ \r\n]{1,2})?[ \r\n]+class"), "$1class", null, 0),
325     }.Cast<ISubstitutionRule>().ToList();
326
327     public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
        ↳ base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
328
329     public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
330 }
331 }

```

## 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(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, 6  
./Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1