

# LinksPlatform's Platform.RegularExpressions.Transformer.CSharpToCpp Class Library

## 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

```

```

(new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)", "std::function<bool($1)>
→ $2", null, 0),
// public static readonly EnsureAlwaysExtensionRoot Always = new
→ EnsureAlwaysExtensionRoot();
// inline static EnsureAlwaysExtensionRoot Always;
(new Regex(@"public static readonly (?<type>[a-zA-Z0-9]+) (?<name>[a-zA-Z0-9_]+) =
→ new \k<type>\(\);", "inline static ${type} ${name};", null, 0),
// public static readonly string ExceptionContentsSeparator = "---";
// inline static const char* ExceptionContentsSeparator = "---";
(new Regex(@"public static readonly string (?<name>[a-zA-Z0-9_]+) =
→ ""(?<string>(\\"|["'])+"";", "inline static const char* ${name} =
→ \"${string}\";", null, 0),
// private const int MaxPath = 92;
// static const int MaxPath = 92;
(new Regex(@"private (const|static readonly) ([a-zA-Z0-9]+) ([_a-zA-Z0-9]+) =
→ ([^;]+);", "static const $2 $3 = $4;", null, 0),
// ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
→ TArgument : class
// ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument& argument)
(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),
// protected virtual
// virtual
(new Regex(@"protected virtual", "virtual", null, 0),
// protected abstract TElement GetFirst();
// virtual TElement GetFirst() = 0;
(new Regex(@"protected abstract ([^;]+);", "virtual $1 = 0;", null, 0),
// TElement GetFirst();
// virtual TElement GetFirst() = 0;
(new Regex(@"([r\n]+[ ]+)((?!return)[a-zA-Z0-9]+ [a-zA-Z0-9]+\([^\\]*\\)) (;[
→ ]*\[r\n]+)", "$1virtual $2 = 0$3", null, 1),
// public virtual
// virtual
(new Regex(@"public virtual", "virtual", null, 0),
// protected readonly
//
(new Regex(@"protected readonly ", "", null, 0),
// protected readonly TreeElement[] _elements;
// TreeElement _elements[N];
(new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+)(\[\\]+)
→ ([_a-zA-Z0-9]+);", "$2 $4[N];", null, 0),
// protected readonly TElement Zero;
// TElement Zero;
(new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+) ([_a-zA-Z0-9]+);", "$2
→ $3;", null, 0),
// private
//
(new Regex(@"(\\W)(private|protected|public|internal) ", "$1", null, 0),
// SizeBalancedTree(int capacity) => a = b;
// SizeBalancedTree(int capacity) { a = b; }
(new Regex(@"(^\\s+)(override )?(void )?([a-zA-Z0-9]+)\\(((\\[\\])*\\)\\s+>\\s+([^;]+);",
→ "$1$2$3$4($5) { $6; }", null, 0),
// int SizeBalancedTree(int capacity) => a;
// int SizeBalancedTree(int capacity) { return a; }
(new Regex(@"(^\\s+)(override )?([a-zA-Z0-9]+
→ )([a-zA-Z0-9]+)\\(((\\[\\])*\\)\\s+>\\s+([^;]+);", "$1$2$3$4($5) { return $6; }",
→ null, 0),
// () => Integer<TElement>.Zero,
// () { return Integer<TElement>.Zero; },
(new Regex(@"\\(\\)\\s+>\\s+([\\r\\n;]+?)", "() { return $1; }", null, 0),
// => Integer<TElement>.Zero;
// { return Integer<TElement>.Zero; }
(new Regex(@"\\)\\s+>\\s+([\\r\\n;]+?);", ") { return $1; }", null, 0),
// () { return avlTree.Count; }
// [&]()-> auto { return avlTree.Count; }
(new Regex(@"\\, \\(\\) { return ([^;]+); }", ", [&]()-> auto { return $1; }", null, 0),
// Count => GetSizeOrZero(Root);
// GetCount() { return GetSizeOrZero(Root); }
(new Regex(@"([A-Z][a-z]+)\\s+>\\s+([^;]+);", "Get$1() { return $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),

```

```

119 // $"
120 // "
121 (new Regex(@"\$"""), @"\\"", null, 0),
122 // Console.WriteLine(...)
123 // printf(...\n")
124 (new Regex(@"Console\.WriteLine\(\"([^\"]+)\\""), "printf(\"$1\\n\\n\"", null, 0),
125 // throw new InvalidOperationException
126 // throw std::exception
127 (new Regex(@"throw new (InvalidOperationException|Exception)", "throw
→ std::exception", null, 0),
128 // override void PrintNode(TElement node, StringBuilder sb, int level)
129 // void PrintNode(TElement node, StringBuilder sb, int level) override
130 (new Regex(@"override ([a-zA-Z0-9 \*+]+)\([^\)]+\)", "$1$2 override", null, 0),
131 // string
132 // char*
133 (new Regex(@"(\W)string(\W)"), "$1char*$2", null, 0),
134 // sbyte
135 // std::int8_t
136 (new Regex(@"(\W)sbyte(\W)"), "$1std::int8_t$2", null, 0),
137 // uint
138 // std::uint32_t
139 (new Regex(@"(\W)uint(\W)"), "$1std::uint32_t$2", null, 0),
140 // char*[] args
141 // char* args[]
142 (new Regex(@"([_a-zA-Z0-9:*\?]?)\[\] ([a-zA-Z0-9]+)"), "$1 $2[]", null, 0),
143 // @object
144 // object
145 (new Regex(@"@([_a-zA-Z0-9]+)"), "$1", null, 0),
146 // using Platform.Numbers;
147 //
148 (new Regex(@"([\r\n]{2}|~)\s*?using \.a-zA-Z0-9+;\s*?($)", "", null, 0),
149 // struct TreeElement { }
150 // struct TreeElement { };
151 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([~;])", "$1
→ $2$3{$4};$5", null, 0),
152 // class Program { }
153 // class Program { };
154 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)[^\r\n]*([\r\n]+(?<indentLevel>[\t
→ ]*)?)\{([\S\s]+?[\r\n]+\k<indentLevel>)\}([~;]|$)", "$1 $2$3{$4};$5", null, 0),
155 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
156 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
157 (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)", "class $1 : public $2", null,
→ 0),
158 // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
159 // class IProperty : public ISetter<TValue, TObject>, IProvider<TValue, TObject>
160 (new Regex(@"(?<before>class [a-zA-Z0-9]+ : ((public [a-zA-Z0-9]+(<[a-zA-Z0-9
→ ,]+>)?, )+)?(?<inheritedType>(?!public) [a-zA-Z0-9]+(<[a-zA-Z0-9
→ ,]+>)?(?<after>([a-zA-Z0-9]+(?>)|[\r\n]+)))", "${before}public
→ ${inheritedType}${after}", null, 10),
161 // Insert scope borders.
162 // ref TElement root
163 // ~!root!~ref TElement root
164 (new Regex(@"(?<definition>(?!<= |\\() (ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<ref>))
→ (?<variable>[a-zA-Z0-9]+)(?=\\|, | =))", "~!${variable}!~${definition}", null,
→ 0),
165 // Inside the scope of ~!root!~ replace:
166 // root
167 // *root
168 (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),
169 // Remove scope borders.
170 // ~!root!~
171 //
172 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
173 // ref auto root = ref
174 // ref auto root =
175 (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)"), "$1* $2 =$3", null, 0),
176 // *root = ref left;
177 // root = left;
178 (new Regex(@"\*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", null, 0),
179 // (ref left)
180 // (left)
181 (new Regex(@"\\(ref ([a-zA-Z0-9]+)(\\|\\(|,))", "($1$2", null, 0),
182 // ref TElement
183 // TElement*

```

```

184 (new Regex(@"( |\()ref ([a-zA-Z0-9]+) ", "$1$2* ", null, 0),
185 // ref sizeBalancedTree.Root
186 // &sizeBalancedTree->Root
187 (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)", "&$1->$2", null, 0),
188 // ref GetElement(node).Right
189 // &GetElement(node)->Right
190 (new Regex(@"ref ([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)",
191   ↳ "&$1($2)->$3", null, 0),
192 // GetElement(node).Right
193 // GetElement(node)->Right
194 (new Regex(@"([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)", "$1($2)->$3",
195   ↳ null, 0),
196 // [Fact]\npublic static void SizeBalancedTreeMultipleAttachAndDetachTest()
197 // TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
198 (new Regex(@"\[Fact\]\[s\n\]+(static )?void ([a-zA-Z0-9]+)\(\)", "TEST_METHOD($2)",
199   ↳ null, 0),
200 // class TreesTests
201 // TEST_CLASS(TreesTests)
202 (new Regex(@"class ([a-zA-Z0-9]+)Tests", "TEST_CLASS($1)", null, 0),
203 // Assert.Equal
204 // Assert::AreEqual
205 (new Regex(@"Assert\.Equal", "Assert::AreEqual", null, 0),
206 // TElement Root;
207 // TElement Root = 0;
208 (new Regex(@"(\\r?\\n[\\t ]+)([a-zA-Z0-9:_]+(?:!return)) ([_a-zA-Z0-9]+);", "$1$2 $3 =
209   ↳ 0;", null, 0),
210 // TreeElement _elements[N];
211 // TreeElement _elements[N] = { {0} };
212 (new Regex(@"(\\r?\\n[\\t ]+)([a-zA-Z0-9]+) ([_a-zA-Z0-9]+)\\([([_a-zA-Z0-9]+)\\];",
213   ↳ "$1$2 $3[$4] = { {0} };", null, 0),
214 // auto path = new TElement[MaxPath];
215 // TElement path[MaxPath] = { {0} };
216 (new Regex(@"(\\r?\\n[\\t ]+)[a-zA-Z0-9]+ ([a-zA-Z0-9]+) = new
217   ↳ ([a-zA-Z0-9]+)\\([([_a-zA-Z0-9]+)\\];", "$1$3 $2[$4] = { {0} };", null, 0),
218 // Insert scope borders.
219 // auto added = new HashSet<TElement>();
220 // ~!added!~std::unordered_set<TElement> added;
221 (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
222   ↳ HashSet<(?<element>[a-zA-Z0-9]+)>\\(\)");
223   ↳ "~!${variable}!~std::unordered_set<${element}> ${variable};", null, 0),
224 // Inside the scope of ~!added!~ replace:
225 // added.Add(node)
226 // added.insert(node)
227 (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>\\.|\\n)(?<before>((?<
228   ↳ !~!\\k<variable>!~)(\\.|\\n))*?)\\k<variable>\\.Add\\((?<argument>[a-zA-Z0-9]+)\\)",
229   ↳ "${scope}${separator}${before}${variable}.insert(${argument})", null, 10),
230 // Inside the scope of ~!added!~ replace:
231 // added.Remove(node)
232 // added.erase(node)
233 (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>\\.|\\n)(?<before>((?<
234   ↳ !~!\\k<variable>!~)(\\.|\\n))*?)\\k<variable>\\.Remove\\((?<argument>[a-zA-Z0-9]+)\\)",
235   ↳ "${scope}${separator}${before}${variable}.erase(${argument})", null, 10),
236 // if (added.insert(node)) {
237 // if (!added.contains(node)) { added.insert(node);
238 (new Regex(@"if \\((?<variable>[a-zA-Z0-9]+)\\.insert\\((?<argument>[a-zA-Z0-9]+)\\)\\)(?
239   ↳ <separator>[\\t ]*[\\r\\n]+(?:<indent>[\\t ]*){", "if
240   ↳ (!${variable}.contains(${argument}))${separator}${indent}{ " +
241   ↳ Environment.NewLine + "${indent}    ${variable}.insert(${argument});", null, 0),
242 // Remove scope borders.
243 // ~!added!~
244 //
245 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
246 // Insert scope borders.
247 // auto random = new System.Random(0);
248 // std::srand(0);
249 (new Regex(@"[a-zA-Z0-9\\.]+ ([a-zA-Z0-9]+) = new
250   ↳ (System\\.)?Random\\((([a-zA-Z0-9]+)\\);", "~!$1!~std::srand($3);", null, 0),
251 // Inside the scope of ~!random!~ replace:
252 // random.Next(1, N)
253 // (std::rand() % N) + 1
254 (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>\\.|\\n)(?<before>((?<
255   ↳ !~!\\k<variable>!~)(\\.|\\n))*?)\\k<variable>\\.Next\\((?<from>[a-zA-Z0-9]+),
256   ↳ (?<to>[a-zA-Z0-9]+)\\)", "${scope}${separator}${before}(std::rand() % ${to}) +
257   ↳ ${from}", null, 10),
258 // Remove scope borders.
259 // ~!random!~
260

```

```

241 //
242 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
243 // Insert method body scope starts.
244 // void PrintNodes(TElement node, StringBuilder sb, int level) {
245 // void PrintNodes(TElement node, StringBuilder sb, int level) { /*method-start*/
246 (new Regex(@"(?<start>\r?\n[\t ]+)(?<prefix>((virtual )?[a-zA-Z0-9:_]+
    → )?)(?<method>[a-zA-Z][a-zA-Z0-9]*)\(((?<arguments>[^\s]*)\)(?<override>(
    → override)?)(?<separator>[ \t\r\n]*)\{((?<end>[~])")", "${start}${prefix}${method}
    → (${arguments})${override}${separator}{ /*method-start*/${end}", null,
    → 0),
247 // Insert method body scope ends.
248 // { /*method-start*/... }
249 // { /*method-start*/... /*method-end*/ }
250 (new Regex(@"\{ /\*method-start*/(?<body>((?<bracket>\{) | (?<-bracket>\}) | [^\{\}]*)+ )
    → \}"), "{ /*method-start*/${body} /*method-end*/", null,
    → 0),
251 // Inside method bodies replace:
252 // GetFirst(
253 // this->GetFirst(
254 // (new Regex(@"(?<separator>(\(| |([\\W]) |return ))(?<!(->|\\*
    → ))(?<method>(?!sizeof)[a-zA-Z0-9]+)\(((?!\\) \{)"),
    → "${separator}this->${method}(", null, 1),
255 (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),
256 // Remove scope borders.
257 // /*method-start*/
258 //
259 (new Regex(@"\/\*method-(start|end)\/"), "", null, 0),
260 // throw new ArgumentNullException(argumentName, message);
261 // throw std::invalid_argument(((std::string)"Argument
    → ").append(argumentName).append(" is null: ").append(message).append("."));
262 (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),
263 // throw new ArgumentException(message, argumentName);
264 // throw std::invalid_argument(((std::string)"Invalid
    → ").append(argumentName).append(" argument: ").append(message).append("."));
265 (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),
266 // throw new NotSupportedException();
267 // throw std::logic_error("Not supported exception.");
268 (new Regex(@"throw new NotSupportedException\(\);"), "throw std::logic_error(\"Not
    → supported exception.\");", null, 0),
269 // throw new NotImplementedException();
270 // throw std::logic_error("Not implemented exception.");
271 (new Regex(@"throw new NotImplementedException\(\);"), "throw std::logic_error(\"Not
    → implemented exception.\");", null, 0),
272
273 }.Cast<ISubstitutionRule>().ToList();
274
275 public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
276 {
277     // ICounter<int, int> c1;
278     // ICounter<int, int>* c1;
279     (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^\r\n]+>)?)
    → (?<variable>[_a-zA-Z0-9]+);"), "${abstractType}* ${variable};", null, 0),
280     // (expression)
281     // expression
282     (new Regex(@"(\(| |)(([a-zA-Z0-9_\*:]+)\(| |;|\\))"), "$1$2$3", null, 0),
283     // (method(expression))
284     // method(expression)
285     (new Regex(@"(?<firstSeparator>(\(|
    → ))\(((?<method>[a-zA-Z0-9_\->]*)+)\(((?<expression>((?<parenthesis>\(| (?<-parent
    → hesis>\)| [a-zA-Z0-9_\->]*)+)(?<parenthesis>(?!))\)\)\((?<lastSeparator>(|
    → |;|\\))")", "${firstSeparator}${method}(${expression})${lastSeparator}", null, 0),
286     // return ref _elements[node];
287     // return &_elements[node];
288     (new Regex(@"return ref ([a-zA-Z0-9]+)\([([a-zA-Z0-9_\*:]+)\];"), "return &$1[$2];",
    → null, 0),
289     // default

```

```

290 // 0
291 (new Regex(@"(\W)default(\W)", "${1}0$2", null, 0),
292 // //define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
293 //
294 (new Regex(@"\\/[/[ \t]*\#define[ \t]+[_a-zA-Z0-9]+[ \t]*"), "", null, 0),
295 // #if USEARRAYPOOL\r\n#endif
296 //
297 (new Regex(@"#if [a-zA-Z0-9]+\s+#endif", "", null, 0),
298 // [Fact]
299 //
300 (new Regex(@"(?<firstNewLine>\r?\n|\A)(?<indent>[\t ]+)\[[a-zA-Z0-9]+(\((?<expression>
↪ n>((?<parenthesis>\(|(?<-parenthesis>\)|[^\()]*+)(?<parenthesis>(?!))\))?\)[
↪ \t]*\(\r?\n\k<indent>?)")", "${firstNewLine}${indent}", null, 5),
301 // \n ... namespace
302 // namespace
303 (new Regex(@"(\S[\r\n]{1,2})?[\r\n]+namespace)", "$1namespace", null, 0),
304 // \n ... class
305 // class
306 (new Regex(@"(\S[\r\n]{1,2})?[\r\n]+class)", "$1class", null, 0),
307 }.Cast<ISubstitutionRule>().ToList();
308
309 public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
↪ base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
310
311 public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
312 }
313 }

```

## 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