

# 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

```

```

52 (new Regex(@"\(this ", "(", null, 0),
53 // Func<TElement> treeCount
54 // std::function<TElement()> treeCount
55 (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)", "std::function<$1()> $2", null,
56     ↪ 0),
57 // Action<TElement> free
58 // std::function<void(TElement)> free
59 (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)", "std::function<void($1)> $2",
60     ↪ null, 0),
61 // private const int MaxPath = 92;
62 // static const int MaxPath = 92;
63 (new Regex(@"private (const|static readonly) ([a-zA-Z0-9]+) ([_a-zA-Z0-9]+) =
64     ↪ ([~;]+);", "static const $2 $3 = $4;", null, 0),
65 // protected virtual
66 // virtual
67 (new Regex(@"protected virtual", "virtual", null, 0),
68 // protected abstract TElement GetFirst();
69 // virtual TElement GetFirst() = 0;
70 (new Regex(@"protected abstract ([~;]+);", "virtual $1 = 0;", null, 0),
71 // TElement GetFirst();
72 // virtual TElement GetFirst() = 0;
73 (new Regex(@"([r\n]+[ ]+)((?!return)[a-zA-Z0-9]+ [a-zA-Z0-9]+\([^\)]*\))([
74     ↪ ]*\[r\n]+)", "$1virtual $2 = 0$3", null, 1),
75 // public virtual
76 // virtual
77 (new Regex(@"public virtual", "virtual", null, 0),
78 // protected readonly
79 //
80 (new Regex(@"protected readonly ", "", null, 0),
81 // protected readonly TreeElement[] _elements;
82 // TreeElement _elements[N];
83 (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+)([\\[\\]]+
84     ↪ ([_a-zA-Z0-9]+);", "$2 $4[N];", null, 0),
85 // protected readonly TElement Zero;
86 // TElement Zero;
87 (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+) ([_a-zA-Z0-9]+);", "$2
88     ↪ $3;", null, 0),
89 // private
90 //
91 (new Regex(@"(\W)(private|protected|public|internal) "), "$1", null, 0),
92 // SizeBalancedTree(int capacity) => a = b;
93 // SizeBalancedTree(int capacity) { a = b; }
94 (new Regex(@"(^s+)(override )?(void )?([a-zA-Z0-9]+\(((^\[\\]*\\)\s+=>\s+([~;]+);)",
95     ↪ "$1$2$3$4($5) { $6; }", null, 0),
96 // int SizeBalancedTree(int capacity) => a;
97 // int SizeBalancedTree(int capacity) { return a; }
98 (new Regex(@"(^s+)(override )?([a-zA-Z0-9]+
99     ↪ )([a-zA-Z0-9]+\(((^\[\\]*\\)\s+=>\s+([~;]+);", "$1$2$3$4($5) { return $6; }",
100     ↪ null, 0),
101 // () => Integer<TElement>.Zero,
102 // () { return Integer<TElement>.Zero; },
103 (new Regex(@"\(\)\s+=>\s+([~r\n;]+?);", "()" { return $1; }", null, 0),
104 // => Integer<TElement>.Zero;
105 // { return Integer<TElement>.Zero; }
106 (new Regex(@"\)\s+=>\s+([~r\n;]+?);", ") { return $1; }", null, 0),
107 // () { return avlTree.Count; }
108 // [&]()-> auto { return avlTree.Count; }
109 (new Regex(@"\(\)\s+{ return ([~;]+); }", " [&]()-> auto { return $1; }", null, 0),
110 // Count => GetSizeOrZero(Root);
111 // GetCount() { return GetSizeOrZero(Root); }
112 (new Regex(@"([A-Z][a-z]+\s+=>\s+([~;]+);", "Get$1() { return $2; }", null, 0),
113 // var
114 // auto
115 (new Regex(@"(\W)var(\W)", "$1auto$2", null, 0),
116 // unchecked
117 //
118 (new Regex(@"[r\n]{2}\s*?unchecked\s*?$", "", null, 0),
119 // $"
120 // "
121 (new Regex(@"\$"""), "\"", null, 0),
122 // Console.WriteLine(...)
123 // printf(...)
124 (new Regex(@"Console\.WriteLine\(\"([~\"']+)\")", "printf(\"$1\\n\")", null, 0),
125 // throw new InvalidOperationException
126 // throw std::exception
127 (new Regex(@"throw new (InvalidOperationException|Exception)", "throw
128     ↪ std::exception", null, 0),

```

```

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

```

```

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

```

```

238 // Insert method body scope ends.
239 // {/*method-start*/...}
240 // {/*method-start*/.../*method-end*/}
241 (new Regex(@"{\/*method-start*/(?<body>((?<bracket>\{)|(?!<-bracket>\})|[\^\{\}])*+)"
    ↳ \}"), "/*method-start*/${body}/*method-end*/", null,
    ↳ 0),
242 // Inside method bodies replace:
243 // GetFirst(
244 // this->GetFirst(
245 // (new Regex(@"(?<separator>(\(| |([\\W]) |return ))(?<!(?!<->|\\*
    ↳ ))(?<method>(?!sizeof)[a-zA-Z0-9]+)\\((?!\\) \{)"),
    ↳ "${separator}this->${method}(", null, 1),
246 (new Regex(@"(?<scope>\\/*method-start\\/*) (?<before>((?!\\/*method-end\\/*)(.|\\n))*?) (
    ↳ ?<separator>[\\W] (?<!(?!<->|\\*|.|\\n->)) (?<method>(?!sizeof)[a-zA-Z0-9]+)\\((?!\\)
    ↳ \{) (?<after>(|\\n))*?) (?<scopeEnd>\\/*method-end\\/*)"),
    ↳ "${scope}${before}${separator}this->${method}(${after}${scopeEnd}", null, 100),
247 // Remove scope borders.
248 // /*method-start*/
249 //
250 (new Regex(@"\\/*method-(start|end)\\/*/"), "", null, 0),
251 // throw new ArgumentNullException(argumentName, message);
252 // throw std::invalid_argument(((std::string)"Argument
    ↳ ").append(argumentName).append(" is null: ").append(message).append("."));
253 (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),
254 // throw new ArgumentException(message, argumentName);
255 // throw std::invalid_argument(((std::string)"Invalid
    ↳ ").append(argumentName).append(" argument: ").append(message).append("."));
256 (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),
257 // throw new NotSupportedException();
258 // throw std::logic_error("Not supported exception.");
259 (new Regex(@"throw new NotSupportedException\\(\\)");", "throw std::logic_error("\\Not
    ↳ supported exception.\\");", null, 0),
260 // throw new NotImplementedException();
261 // throw std::logic_error("Not implemented exception.");
262 (new Regex(@"throw new NotImplementedException\\(\\)");", "throw std::logic_error("\\Not
    ↳ implemented exception.\\");", null, 0),
263
264 }.Cast<ISubstitutionRule>().ToList();
265
266 public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
267 {
268 // ICounter<int, int> c1;
269 // ICounter<int, int>* c1;
270 (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^\r\n]+>)?)
    ↳ (?<variable>[_a-zA-Z0-9]+);", "${abstractType}* ${variable};", null, 0),
271 // (expression)
272 // expression
273 (new Regex(@"(\\(| |)(([a-zA-Z0-9_\\*:]*)\\(| |;|\\))", "$1$2$3", null, 0),
274 // (method(expression))
275 // method(expression)
276 (new Regex(@"(?<firstSeparator>\\(|
    ↳ ))\\((?<method>[a-zA-Z0-9_\\*:]*)\\((?<expression>((?<parenthesis>\\(|(?<-parent
    ↳ hesis>\\)|[a-zA-Z0-9_\\*:]*)\\((?<parenthesis>(?!))\\)\\) (?<lastSeparator>\\(|
    ↳ |;|\\))")", "${firstSeparator}${method}(${expression})${lastSeparator}", null, 0),
277 // return ref _elements[node];
278 // return &elements[node];
279 (new Regex(@"return ref ([_a-zA-Z0-9]+)\\([[_a-zA-Z0-9\\*:]*)\\);", "return &$1[$2];",
    ↳ null, 0),
280 // default
281 // 0
282 (new Regex(@"(\\W)default(\\W)"), "${1}0$2", null, 0),
283 // // #define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
284 //
285 (new Regex(@"\\[/[ \t]*\\#define[ \t]+[_a-zA-Z0-9]+[ \t]*")", "", null, 0),
286 // #if USEARRAYPOOL\\r\\n\\#endif
287 //
288 (new Regex(@"#if [a-zA-Z0-9]+\\s+\\#endif")", "", null, 0),
289 // [Fact]
290 //

```

```

291         (new Regex(@"(?<firstNewLine>\r?\n|\A)(?<indent>[\t ]+)\[[a-zA-Z0-9]+(\(((?<expressio
    ↪ n>((?<parenthesis>\(|(?<-parenthesis>\)|[^(]*)+)(?(parenthesis)(?!))\))?\)[
    ↪ \t]*(\r?\n\k<indent>)?"), "${firstNewLine}${indent}", null, 5),
292         // \n ... namespace
293         // namespace
294         (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+namespace"), "$1namespace", null, 0),
295         // \n ... class
296         // class
297         (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+class"), "$1class", null, 0),
298     }.Cast<ISubstitutionRule>().ToList();
299
300     public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
    ↪     base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
301
302     public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
303 }
304 }

```

## 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