

# 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

```

```

58 (new Regex(@"private (const|static readonly) ([a-zA-Z0-9]+) ([_a-zA-Z0-9]+) =
    ↳ ([~;]+);"), "static const $2 $3 = $4;", null, 0),
59 // protected virtual
60 // virtual
61 (new Regex(@"protected virtual"), "virtual", null, 0),
62 // protected abstract TElement GetFirst();
63 // virtual TElement GetFirst() = 0;
64 (new Regex(@"protected abstract ([~;]+);"), "virtual $1 = 0;", null, 0),
65 // TElement GetFirst();
66 // virtual TElement GetFirst() = 0;
67 (new Regex(@"([\r\n]+[ ]+)((?!return)[a-zA-Z0-9]+ [a-zA-Z0-9]+\([~\)]*\))([
    ↳ ]*[\r\n]+)"), "$1virtual $2 = 0$3", null, 1),
68 // public virtual
69 // virtual
70 (new Regex(@"public virtual"), "virtual", null, 0),
71 // protected readonly
72 //
73 (new Regex(@"protected readonly "), "", null, 0),
74 // protected readonly TreeElement[] _elements;
75 // TreeElement _elements[N];
76 (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+)([\\[\\]]+
    ↳ ([_a-zA-Z0-9]+);"), "$2 $4[N];", null, 0),
77 // protected readonly TElement Zero;
78 // TElement Zero;
79 (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+) ([_a-zA-Z0-9]+);"), "$2
    ↳ $3;", null, 0),
80 // private
81 //
82 (new Regex(@"(\W)(private|protected|public|internal) "), "$1", null, 0),
83 // SizeBalancedTree(int capacity) => a = b;
84 // SizeBalancedTree(int capacity) { a = b; }
85 (new Regex(@"(^s+)(override )?(void )?([a-zA-Z0-9]+\(((~\(|*)~\)s+=>s+([~;]+);"),
    ↳ "$1$2$3$4($5) { $6; }", null, 0),
86 // int SizeBalancedTree(int capacity) => a;
87 // int SizeBalancedTree(int capacity) { return a; }
88 (new Regex(@"(^s+)(override )?([a-zA-Z0-9]+
    ↳ )([a-zA-Z0-9]+\(((~\(|*)~\)s+=>s+([~;]+);"), "$1$2$3$4($5) { return $6; }",
    ↳ null, 0),
89 // () => Integer<TElement>.Zero,
90 // () { return Integer<TElement>.Zero; },
91 (new Regex(@"\(\)\s+=>s+([~\r\n;]+?);"), "()" { return $1; }", null, 0),
92 // => Integer<TElement>.Zero;
93 // { return Integer<TElement>.Zero; }
94 (new Regex(@"\)\s+=>s+([~\r\n;]+?);"), ") { return $1; }", null, 0),
95 // () { return avlTree.Count; }
96 // [&]()-> auto { return avlTree.Count; }
97 (new Regex(@"\(\)\{ return ([~;]+); }"), ", [&]()-> auto { return $1; }", null, 0),
98 // Count => GetSizeOrZero(Root);
99 // GetCount() { return GetSizeOrZero(Root); }
100 (new Regex(@"([A-Z][a-z]+)\s+=>s+([~;]+);"), "Get$1() { return $2; }", null, 0),
101 // var
102 // auto
103 (new Regex(@"(\W)var(\W)"), "$1auto$2", null, 0),
104 // unchecked
105 //
106 (new Regex(@"[\r\n]{2}\s*?unchecked\s*?$"), "", null, 0),
107 // $"
108 // "
109 (new Regex(@"\$"""), "\"", null, 0),
110 // Console.WriteLine(...)
111 // printf("...\n")
112 (new Regex(@"Console\.WriteLine\(\"\"([~\""]+)"\""), "printf(\"$1\\n\")", null, 0),
113 // throw new InvalidOperationException
114 // throw std::exception
115 (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
    ↳ std::exception", null, 0),
116 // override void PrintNode(TElement node, StringBuilder sb, int level)
117 // void PrintNode(TElement node, StringBuilder sb, int level) override
118 (new Regex(@"override ([a-zA-Z0-9 \*+]+\([~\)]+?~\)")), "$1$2 override", null, 0),
119 // string
120 // char*
121 (new Regex(@"(\W)string(\W)"), "$1char*$2", null, 0),
122 // sbyte
123 // std::int8_t
124 (new Regex(@"(\W)sbyte(\W)"), "$1std::int8_t$2", null, 0),
125 // uint
126 // std::uint32_t

```

```

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

```

```

190 (new Regex(@"Assert\.Equal"), "Assert::AreEqual", null, 0),
191 // TElement Root;
192 // TElement Root = 0;
193 (new Regex(@"(\r?\n[\t ]+)([a-zA-Z0-9:_]+(?<!return)) ([_a-zA-Z0-9]+);"), "$1$2 $3 =
194     ↳ 0;", null, 0),
195 // TreeElement _elements[N];
196 // TreeElement _elements[N] = { {0} };
197 (new Regex(@"(\r?\n[\t ]+)([a-zA-Z0-9]+) ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9]+)\];"),
198     ↳ "$1$2 $3[$4] = { {0} };", null, 0),
199 // auto path = new TElement[MaxPath];
200 // TElement path[MaxPath] = { {0} };
201 (new Regex(@"(\r?\n[\t ]+)[a-zA-Z0-9]+ ([a-zA-Z0-9]+) = new
202     ↳ ([a-zA-Z0-9]+)\[([_a-zA-Z0-9]+)\];", "$1$3 $2[$4] = { {0} };", null, 0),
203 // Insert scope borders.
204 // auto added = new HashSet<TElement>();
205 // ~!added!~std::unordered_set<TElement> added;
206 (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
207     ↳ HashSet<(?<element>[a-zA-Z0-9]+)>\(\);",
208     ↳ "~!${variable}!~std::unordered_set<${element}> ${variable};", null, 0),
209 // Inside the scope of ~!added!~ replace:
210 // added.Add(node)
211 // added.insert(node)
212 (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.\|\\n)(?<before>((?<
213     ↳ !~!k<variable>!~)(. \|\\n)*?)\k<variable>\.Add\((?<argument>[a-zA-Z0-9]+)\)",
214     ↳ "$${scope}$$separator$$before$$variable}.insert($${argument})", null, 10),
215 // Inside the scope of ~!added!~ replace:
216 // added.Remove(node)
217 // added.erase(node)
218 (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.\|\\n)(?<before>((?<
219     ↳ !~!k<variable>!~)(. \|\\n)*?)\k<variable>\.Remove\((?<argument>[a-zA-Z0-9]+)\)",
220     ↳ "$${scope}$$separator$$before$$variable}.erase($${argument})", null, 10),
221 // if (added.insert(node)) {
222 // if (!added.contains(node)) { added.insert(node);
223 (new Regex(@"if \((?<variable>[a-zA-Z0-9]+)\.insert\((?<argument>[a-zA-Z0-9]+)\)\)(?
224     ↳ <separator>[\t ]*[\r\n]+)(?<indent>[\t ]*){", "if
225     ↳ (!${variable}).contains($${argument}))$${separator}$${indent}${" +
226     ↳ Environment.NewLine + "${indent}    ${variable}.insert($${argument});", null, 0),
227 // Remove scope borders.
228 // ~!added!~
229 //
230 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
231 // Insert scope borders.
232 // auto random = new System.Random();
233 // std::srand(0);
234 (new Regex(@"[a-zA-Z0-9\.] + ([a-zA-Z0-9]+) = new
235     ↳ (System\.)?Random\((([a-zA-Z0-9]+)\);", "~!$1!~std::srand($3);", null, 0),
236 // Inside the scope of ~!random!~ replace:
237 // random.Next(1, N)
238 // (std::rand() % N) + 1
239 (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.\|\\n)(?<before>((?<
240     ↳ !~!k<variable>!~)(. \|\\n)*?)\k<variable>\.Next\((?<from>[a-zA-Z0-9]+),
241     ↳ (?<to>[a-zA-Z0-9]+)\)", "$${scope}$$separator$$before}{std::rand() % $${to}) +
242     ↳ $${from}", null, 10),
243 // Remove scope borders.
244 // ~!random!~
245 //
246 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
247 // Insert method body scope starts.
248 // void PrintNodes(TElement node, StringBuilder sb, int level) {
249 // void PrintNodes(TElement node, StringBuilder sb, int level) { /*method-start*/
250 (new Regex(@"(?<start>\r?\n[\t ]+)(?<prefix>((virtual )?[a-zA-Z0-9:_]+
251     ↳ )?)(?<method>[a-zA-Z][a-zA-Z0-9]*)\(((?<arguments>[^\)]*)\)(?<override>(
252     ↳ override)?)(?<separator>[\t\r\n]*)\{(?<end>[~])", "$${start}$$prefix$$method}
253     ↳ ($${arguments})$${override}$${separator}{/*method-start*/$${end}", null,
254     ↳ 0),
255 // Insert method body scope ends.
256 // { /*method-start*/...}
257 // { /*method-start*/.../*method-end*/}
258 (new Regex(@"{[/\*method-start*/(?<body>((?<bracket>\{)|(?<-bracket>\})|[\^\\\{\}]*)+)
259     ↳ \}"), "{/*method-start*/$${body}/*method-end*/}", null,
260     ↳ 0),
261 // Inside method bodies replace:
262 // GetFirst(
263 // this->GetFirst(

```

```

242 // (new Regex(@"(?<separator>\(|,|([\W])|return ))(?<!(-->|\*
    ↳ ))(?<method>(?!sizeof)[a-zA-Z0-9]+\((?!\\) \{") ,
    ↳ $"{separator}this->${method}(", null, 1),
243 (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),
244 // Remove scope borders.
245 // /*method-start*/
246 //
247 (new Regex(@"/*method-(start|end)\*/"), "", null, 0),
248 }.Cast<ISubstitutionRule>().ToList();
249
250 public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
251 {
252     // ICounter<int, int> c1;
253     // ICounter<int, int>* c1;
254     (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[~>\r\n]+>)?
    ↳ (?<variable>[_a-zA-Z0-9]+);"), $"{abstractType}* ${variable};", null, 0),
255     // (expression)
256     // expression
257     (new Regex(@"(\(|\)|((([a-zA-Z0-9_]*:)+))\(|\)|;|\\)")), "$1$2$3", null, 0),
258     // (method(expression))
259     // method(expression)
260     (new Regex(@"(?<firstSeparator>\(|\)|
    ↳ ))\((?<method>[a-zA-Z0-9_\\>]*:)+\)\((?<expression>((?<parenthesis>\(|\)|(?<-parent
    ↳ hesis>\\)|[a-zA-Z0-9_\\>]*:)+)(?(parenthesis)(?!))\\)\)(?<lastSeparator>(|
    ↳ |;|\\)")) , $"{firstSeparator}${method}(${expression})${lastSeparator}", null, 0),
261     // return ref _elements[node];
262     // return &elements[node];
263     (new Regex(@"return ref ([_a-zA-Z0-9]+\)|([_a-zA-Z0-9_]*+);"), "return &$1[$2];",
    ↳ null, 0),
264     // default
265     // 0
266     (new Regex(@"(\\W)default(\\W)"), "${1}0$2", null, 0),
267     // // #define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
268     //
269     (new Regex(@"\\/[\\t]*#define[ \\t]+[_a-zA-Z0-9]+[ \\t]*"), "", null, 0),
270     // #if USEARRAYPOOL\\r\\n#endif
271     //
272     (new Regex(@"#if [a-zA-Z0-9]+\\s+endif"), "", null, 0),
273     // [Fact]
274     //
275     (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),
276     // \\n ... namespace
277     // namespace
278     (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+namespace"), "$1namespace", null, 0),
279     // \\n ... class
280     // class
281     (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+class"), "$1class", null, 0),
282 }.Cast<ISubstitutionRule>().ToList();
283
284 public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
    ↳ base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
285
286 public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
287 }
288 }

```

## 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, 5  
./Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1