```
LinksPlatform's Platform RegularExpressions Transformer CSharpToCpp Class Library
        ./Platform. Regular Expressions. Transformer. CSharp To Cpp/CSharp To Cpp Transformer. cs \\
     using System;
     using System.Collections.Generic;
using System.Ling;
 2
     using System. Text. Regular Expressions;
 4
      #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
 6
     namespace Platform.RegularExpressions.Transformer.CSharpToCpp
 8
             public class CSharpToCppTransformer : Transformer
10
11
                   public static readonly IList<ISubstitutionRule> FirstStage = new List<SubstitutionRule>
12
13
14
                          //
15
                          (new Regex(0"(\r?\n)?[\t]+//+.+"), "", null, 0),
16
                          // #pragma warning disable CS1591 // Missing XML comment for publicly visible type
                                or member
18
                          (new Regex(0"^\s*?\pragma[\sa-zA-Z0-9]+$"), "", null, 0),
19
                          // \{ n n n
                          // {
                          (new Regex(0"{\s+[\r\n]+"), "{" + Environment.NewLine, null, 0),
22
                          // Platform.Collections.Methods.Lists
                          // Platform::Collections::Methods::Lists
                          (new Regex(@"(namespace[^\r\n]+?)\.([^\r\n]+?)"), "$1::$2", null, 20),
25
                          // public abstract class
// class
26
27
                          (new Regex(0"(public abstract|static) class"), "class", null, 0),
2.8
                          // class GenericCollectionMethodsBase {
2.9
                          // class GenericCollectionMethodsBase { public:
30
                          (new Regex(@"class ([a-zA-ZO-9]+)(\s+){"), "class $1$2{" + Environment.NewLine + "
                                  public:", null, 0),
                          // class GenericCollectionMethodsBase<TElement> {
32
                          // template <typename TElement> class GenericCollectionMethodsBase { public:
33
                          (new Regex(@"class ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>([^{[]+){"}}, "template <typename $2>
                                class $1$3{" + Environment.NewLine + "
                                                                                                        public:", null, 0),
                          // static void
                                TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
                                tree, TElement* root)
                          // template<typename T> static void
36
                                TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>

    tree, TElement* root)

                           (\text{new Regex}(@"\text{static}([a-zA-Z0-9]+)([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>\(([^\)]+)\)"),
                                 "template <typename $3> static $1 $2($4)", null, 0),
                          // (this
                          // (
3.9
                          (new Regex(@"\(this "), "(", null, 0),
40
                              Func<TElement> treeCount
                          // std::function<TElement()> treeCount
42
                          (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", null, function<$1()> $2", null, function<$
43
                           \rightarrow 0).
                          // Action<TElement> free
                          // std::function<void(TElement)> free
                          (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<void($1)> $2",
                           \rightarrow null, 0),
                          // private const int MaxPath = 92;
47
                          // static const int MaxPath = 92;
48
                          (new Regex(@"private (const|static readonly) ([a-zA-Z0-9]+) ([_a-zA-Z0-9]+) =
                                 ([^*;]+);"), "static const $2 $3 = $4;", null, 0),
                          // protected virtual
50
                          // virtual
5.1
                          (new Regex(0"protected virtual"), "virtual", null, 0),
                          // protected abstract TElement GetFirst();
                          // virtual TElement GetFirst() = 0;
54
                          (new Regex(@"protected abstract ([^;]+);"), "virtual $1 = 0;", null, 0),
55
                          // public virtual
56
                          // virtual
57
                          (new Regex(@"public virtual"), "virtual", null, 0),
58
                          // protected readonly
                                                                                       "", null, 0),
                          (new Regex(0"protected readonly ")
61
                          // protected readonly TreeElement[] _elements;
62
                          // 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;
65
                 // TElement Zero;
                 (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+) ([a-zA-Z0-9]+);"), "$2
67
                     $3;", null, 0),
                 // private
68
                 //
                 (new Regex(@"(\W)(private|protected|public|internal) "), "$1", null, 0),
70
                 // SizeBalancedTree(int capacity) => a = b;
71
                 // SizeBalancedTree(int capacity) { a = b; }
72
                 (new Regex(0"(^s+)(override)?(void)?([a-zA-Z0-9]+)\(([^s(]*)\)\s+=>\s+([^s]+);"),
73
                     "$1$2$3$4($5) { $6; }", null, 0)
                 // int SizeBalancedTree(int capacity) => a;
                 // int SizeBalancedTree(int capacity) { return a; }
75
                 (new Regex(0"(^{s+})(override)?([a-zA-Z0-9]+
                     )([a-zA-Z0-9]+)\(([^{(]*)}\)\s+=>\s+([^{;}]+);"), "$1$2$3$4($5) { return $6; }",
                     null, 0),
                    () => Integer<TElement>.Zero,
77
                 // () {    return Integer<TElement>.Zero;    }
                 (new Regex(0"\(\)\s+=>\s+([^{r}, ;]+?),"), "() { return $1; },", null, 0),
79
                 // => Integer<TElement>.Zero;
80
                 // { return Integer<TElement>.Zero; }
                 (\text{new Regex}(@"))s+=>\s+([^\r\n;]+?);"), ") { return $1; }", null, 0),
82
                 // () { return avlTree.Count; }
83
                 // [&]()-> auto { return avlTree.Count;
84
                 (new Regex(0", \(\) { return ([^;]+); }"), ", [&]()-> auto { return $1; }", null, 0),
                 // Count => GetSizeOrZero(Root)
86
                 // GetCount() { return GetSizeOrZero(Root); }
87
                 (new Regex(0"([A-Z][a-z]+)\s+=>\s+([^;]+); "), "Get$1() { return $2; }", null, 0),
                 // var
89
                 // auto
90
                 (\text{new Regex}(0"(\W) \text{var}(\W)"), "$1auto$2", null, 0),
91
                 // unchecked
93
                 (new Regex(0"[\r\n]{2}\s*?unchecked\s*?$"), "", null, 0),
94
                 // "
96
                 (new Regex(@"\$"""), "\"", null, 0),
97
                   / Console.WriteLine("...")
98
                 // printf("...\n")
                 (new Regex(@"Console\.WriteLine\(""([^""]+)""\)"), "printf(\"$1\\n\")", null, 0),
100
                 // throw new InvalidOperationException
101
                 // throw std::exception
102
                 (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
103

    std::exception", null, 0),
                 // override void PrintNode(TElement node, StringBuilder sb, int level)
104
                 // void PrintNode(TElement node, StringBuilder sb, int level) override
105
                 (new Regex(0"override ([a-zA-Z0-9 \*\+]+)(\([^\)]+?\)))"), "$1$2 override", null, 0),
                 // string
107
                 // char*
108
                 (new Regex(@"(\W)string(\W)"), "$1char*$2", null, 0),
109
                 // sbyte
110
                 // std::int8_t
111
                 (new Regex(@"(\W)sbyte(\W)"), "$1std::int8_t$2", null, 0),
112
113
                 // std::uint32_t
114
                 (\text{new Regex}(@"(\W)\text{uint}(\W)"), "$1std::uint32_t$2", null, 0),
115
                 // char*[] args
                 // char* args[]
117
                 (\text{new Regex}(\bar{\mathbb{Q}}''([_a-zA-ZO-9:\*]?)\[\]([_a-zA-ZO-9]+)"), "$1 $2[]", null, 0),
118
119
                 // using Platform.Numbers;
120
                 (new Regex(0"([\r\n]{2}|^)\s*?using [\.a-zA-Z0-9]+;\s*?$"), "", null, 0),
121
                 // struct TreeElement { }
122
                 // struct TreeElement { };
                 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([^;])"), "$1
124
                    $2$3{$4};$5", null, 0),
                 // class Program { }
// class Program { };
125
126
                 (\texttt{new Regex}(\texttt{@"(struct|class)} ([\texttt{a-zA-Z0-9}]+[^\r\n]*)([\r\n]+(?<\texttt{indentLevel}>[\t]))
                     ]*)?)\{([\S\s]+?[\r\n]+\k<indentLevel>)\}([^;]|$)"), "$1 $2$3{$4};$5", null, 0),
                 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
128
                 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
129
                 (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)"), "class $1 : public $2", null,
130
                     0).
                 // Insert scope borders.
```

```
// ref TElement root
132
                            // ~!root!~ref TElement root
                           (new Regex(0"(?<definition>(?<= |\()(ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!ref))
134
                                   (?< variable>[a-zA-Z0-9]+)(?=\)|, | =))"), "~!${variable}!~${definition}", null, "..."
                                  0).
                           // Inside the scope of ~!root!~ replace:
                           // root
136
                           // *root
137
                            (new Regex(@"(?<definition>~!(?<pointer>[a-zA-Z0-9]+)!~ref [a-zA-Z0-9]+)
                                   (?<pointer>[a-zA-Z0-9]+)(?=\)|,
                                  =)) (?<before>((?<!~!\k<pointer>!~)(.|\n))*?)(?<prefix>(\W
                                   |\cdot()\rangle = (?\langle suffix\rangle (\cdot |\cdot\rangle |; |,))
                                  "${definition}${before}${prefix}*${pointer}${suffix}", null, 70),
                           // Remove scope borders.
139
                           // ~!root!~
140
                           (new Regex(0"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
142
                           // ref auto root = ref
143
                           // ref auto root =
144
                           (\text{new Regex}(@"\text{ref}([a-zA-Z0-9]+)([a-zA-Z0-9]+) = \text{ref}(\W)"), "$1* $2 =$3", null, 0),
                           // *root = ref left;
146
                           // root = left;
147
                           (\text{new Regex}(@"\*([a-zA-Z0-9]+) = \text{ref}([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", null, 0),
                           // (ref left)
149
                           // (left)
150
                            (\text{new Regex}(@"\(\text{ref }([a-zA-Z0-9]+)(\)|\(|,)"), "($1$2", null, 0),
151
                                 ref TElement
152
                           // TElement*
153
                           (\text{new Regex}(@"( | \ ([a-zA-Z0-9]+) "), "$1$2* ", null, 0),
154
                           // ref sizeBalancedTree.Root
                           // &sizeBalancedTree->Root
156
                           (\text{new Regex}(@"\text{ref}([a-zA-Z0-9]+)).([a-zA-Z0-9]*]+)"), "&$1->$2", null, 0),
157
                              / ref GetElement(node).Right
                           // &GetElement(node)->Right
159
                           (new Regex(0"ref ([a-zA-\bar{Z}0-9]+)\(([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"),
160
                                  "&$1($2)->$3", null, 0),
                           // GetElement(node).Right
161
                           // GetElement(node)->Right
                           (\text{new Regex}(@"([a-zA-Z0-9]+)\(([a-zA-Z0-9]*]+)\)\.([a-zA-Z0-9]+)"), "$1($2)->$3", "a=2A-Z0-9]+)"], "a=2A-Z0-9]+)"], "a=2A-Z0-9]+)"], "a=2A-Z0-9]+)"], "a=2A-Z0-9]+)"], "a=2A-Z0-9]+)"], "a=2A-Z0-9]+)"], "a=2A-Z0-20-9]+)"], "a=2A-Z0-20-9]+]"], "a=2A-Z0-20-9]+]"], "a=2A-Z0-20-9]+]"], "a=2A-Z0-20-9]+]"], "a=2A-Z0-20-9]+]"], "a=2A-Z0-20-9]+]"], "a=2A-Z0-20-9]+]"], "a=2A-Z0-20-9]+]"], "a=2A-Z0-20-9]+[], "a
163
                                 null, 0),
                                 [Fact]\npublic static void SizeBalancedTreeMultipleAttachAndDetachTest()
164
                           // TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
165
                           (new Regex(@"\[Fact\] [\s\n] + (static )?void ([a-zA-Z0-9]+)\(\)"), "TEST_METHOD($2)",
                              \rightarrow null, 0),
                           // class TreesTests
167
                           // TEST_CLASS(TreesTests)
168
                           (new Regex(@"class ([a-zA-ZO-9]+)Tests"), "TEST_CLASS($1)", null, 0),
                           // Assert.Equal
170
                           // Assert::AreEqual
171
                           (new Regex(@"Assert\.Equal"), "Assert::AreEqual", null, 0),
172
                           // TElement Root;
                           // TElement Root = 0;
174
                           (\text{new Regex}(@"(\r?\n[\t]+)([a-zA-Z0-9:_]+(?<!\text{return}))([_a-zA-Z0-9]+);"), "$1$2 $3 =
175
                                  0;", null, 0),
                           // TreeElement _elements[N];
                           // TreeElement _elements[N] = { {0} }
                            (\text{new Regex}(@"(\r?\n[\t]+)([a-zA-Z0-9]+) ([_a-zA-Z0-9]+)\];"), 
178
                                  "$1$2 $3[$4] = { {0} }; ", null, 0),
                                auto path = new TElement[MaxPath];
179
                           // TElement path[MaxPath] = { {0} }
180
                           (\text{new Regex}(0^{-}(\r?\n[\t]+)[a-zA-ZO-9]+([a-zA-ZO-9]+) = \text{new})
                                  ([a-zA-Z0-9]+)\setminus[([-a-zA-Z0-9]+)\setminus];"), "$1$3 $2[$4] = { {0} };", null, 0),
                           // Insert scope borders.
182
                           // auto added = new HashSet<TElement>();
183
                           // ~!added!~std::unordered_set<TElement> added;
                           (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
185
                                  HashSet < (? < element > [a-zA-Z0-9] +) > ( ( ); " )
                                   "~!${variable}!~std::unordered_set<${element}> ${variable};", null, 0),
                           // Inside the scope of "!added!" replace:
                           // added.Add(node)
187
                           // added.insert(node)
188
                           (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?<|</pre>
                                  !~!\k<variable>!~)(.|\n))*?)\k<variable>\.Add\((?<argument>[a-zA-Z0-9]+)\)"),
                                  "${scope}${separator}${before}${variable}.insert(${argument})", null, 10),
                           // Inside the scope of ~!added!~ replace:
190
                           // added.Remove(node)
191
                           // added.erase(node)
```

```
(new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?<,</pre>
193
                                            !^*(x\sim x)^*((((x\sim x)^*))^*((x\sim x)^*)
                                            "${scope}${separator}${before}${variable}.erase(${argument})", null, 10),
                                   // if (added.insert(node)) {
                                   // if (!added.contains(node)) { added.insert(node);
                                   (new Regex(@"if \(((?\langle variable \rangle [a-zA-ZO-9] + ) \rangle.insert(((?<math>\langle variable \rangle [a-zA-ZO-9] + ) \rangle))))
196
                                            \operatorname{separator}[\t] *[\r\n] +) (?(\operatorname{indent}[\t] *) {"}, "if
                                            (!${variable}.contains(${argument}))${separator}${indent}{" +
                                           Environment.NewLine + "${indent}
                                                                                                                           ${variable}.insert(${argument});", null, 0),
                                   // Remove scope borders.
                                          ~!added!
                                   //
198
199
                                   (\text{new Regex}(@"^{!}(?\leq \text{pointer})[a-zA-zO-9]+)!^{"}), "", \text{null}, 5),
200
                                   // Insert scope borders.
                                   // auto random = new System.Random(0);
202
                                   // std::srand(0);
203
                                   (\text{new Regex}(@"[a-zA-Z0-9]) + ([a-zA-Z0-9]) + ) = \text{new}
                                            (System\.)?Random\(([a-zA-Z0-9]+)\);"), "~!$1!~std::srand($3);", null, 0),
                                   // Inside the scope of ~!random!~ replace:
205
                                   // random.Next(1, N)
206
                                   // (std::rand() % N) + 1
207
                                   (new\ Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?<|))(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator>.|\n)(?<separator)(?<separator>.|\n)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?<separator)(?
                                             !^*!\k<\text{variable}!^*)(.\|\n))*?)\k<\text{variable}\.\Next\((?<\text{from})[a-zA-ZO-9]+)]
                                            (?<to>[a-zA-Z0-9]+))"), "${scope}${separator}${before}(std::rand() % ${to}) +
                                           ${from}", null, 10),
                                   // Remove scope borders.
                                   //
                                          "!random!"
210
211
                                   (new Regex(0"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
212
                                   // Insert method body scope starts.
                                   // void PrintNodes(TElement node, StringBuilder sb, int level) {
214
                                   // void PrintNodes(TElement node, StringBuilder sb, int level) {/*method-start*/
215
                                   (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),
                                   // Insert method body scope ends.
217
                                   // {/*method-start*/...}
218
                                   // {/*method-start*/.../*method-end*/}
219
                                   (\text{new Regex}(@''_{/\star}) | (?<\text{body}((?<\text{bracket})) | (?<-\text{bracket})) | (?({}) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (?({})) | (
220
                                            \}"), "{/*method-start*/${body}/*method-end*/}", null,
                                           0)
                                   // Inside method bodies replace:
221
                                   // GetFirst(
222
                                   // this->GetFirst(
                                   //(new Regex(@"(?<separator>(\(|, |([\\]) |return ))(?<!(->|\*
                                            ))(?<method>(?!sizeof)[a-zA-Z0-9]+)\((?!\)\{)"),
                                            "${separator}this->${method}(", null,
                                    (\texttt{new Regex}(@"(?<scope>/\\*method-start\\*/)(?<before>((?<!/\\*method-end\\*/)(.|\\n))*?)(_{|})()
                                            <separator>[\W](?<!(::|\.|->)))(?<method>(?!sizeof)[a-zA-Z0-9]+)\((?!\)
                                             "${scope}${before}${separator}this->${method}(${after}${scopeEnd}", null, 100),
                                   // Remove scope borders.
226
                                   // /*method-start*/
227
                                   (new Regex(0"/\*method-(start|end)\*/"), "", null, 0),
229
230
                          }.Cast<ISubstitutionRule>().ToList();
231
                          public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
232
233
                                   // (expression)
234
                                   // expression
235
                                    (\text{new Regex}(@"((| ))(([a-zA-Z0-9_{*:}]+))(,| |;|))"), "$1$2$3", null, 0),
                                        (method(expression))
237
                                   // method(expression)
238
                                   (new Regex(@"(?<firstSeparator>(\())
                                            ))\((?<method>[a-zA-Z0-9_\->\*:]+)\((?<expression>((?<parenthesis>\()|(?<-parent
                                           |;|\)))"), "${firstSeparator}${method}(${expression})${lastSeparator}", null, 0),
                                   // return ref _elements[node];
240
                                   // return &_elements[node];
241
                                   (new Regex(0"return ref ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9\*]+)\];"), "return &$1[$2];",
242
                                           null, 0),
                                   // default
                                   // 0
244
```

```
(new Regex(@"(\W)default(\W)"), "${1}0$2", null, 0),
^{245}
                 // //#define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
246
                 //
247
                 (\text{new Regex}(@'')//[ t]*\#\text{define}[ t]+[_a-zA-Z0-9]+[ t]*"), "", null, 0),
248
                 // #if USEARRAYPOOL\r\n#endif
250
                 (new Regex(0"#if [a-zA-Z0-9]+\s+\#endif"), "", null, 0),
251
                 // [Fact]
252
                 //
253
                 (new Regex(0"(?<firstNewLine>\r?\n|\A)(?<indent>[\t ]+)\[[a-zA-Z0-9]+(\((?<expressio_1)))
254
                    n>((?<parenthesis>\()|(?<-parenthesis>\))|[^()]*)+)(?(parenthesis)(?!))\))?\][
                     t]*(r?\n\k<indent>)?"), "${firstNewLine}${indent}", null, 5),
                 // \n ... namespace
255
                 // namespace
256
                 (new Regex(0"(S[\r\n]{1,2})?[\r\n]+namespace"), "$1namespace", null, 0),
257
                 // \n ... class
258
                 // class
259
                 (\text{new Regex}(0"(\s[\r\n]{1,2})?[\r\n]+class"), "$1class", null, 0),
             }.Cast<ISubstitutionRule>().ToList();
261
            public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
263
             → base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
264
            public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
        }
266
267
      ./Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs
1.2
    using Xunit;
    namespace Platform.RegularExpressions.Transformer.CSharpToCpp.Tests
 3
 4
        public class CSharpToCppTransformerTests
 5
 6
             [Fact]
            public void HelloWorldTest()
 9
                 const string helloWorldCode = @"using System;
10
    class Program
11
12
        public static void Main(string[] args)
13
14
             Console.WriteLine(""Hello, world!"");
15
16
    }":
17
                 const string expectedResult = @"class Program
18
    {
19
        public:
20
        static void Main(char* args[])
21
22
            printf(""Hello, world!\n"");
24
    };";
25
26
                 var transformer = new CSharpToCppTransformer();
                 var actualResult = transformer.Transform(helloWorldCode, new Context(null));
27
                 Assert.Equal(expectedResult, actualResult);
28
             }
29
        }
```

31 }

Index