```
LinksPlatform's Platform RegularExpressions Transformer CSharpToCpp Class Library
     ./Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.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
   namespace Platform.RegularExpressions.Transformer.CSharpToCpp
        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(0"(namespace[\rrimn]+?)\.([\rrimn]+?)"), "$1::$2", null, 20),
25
                // out TProduct
26
                // TProduct
27
                (new Regex(0"(?<before>(<|, ))(in|out)</pre>
2.8
                    (?<typeParameter>[a-zA-Z0-9]+)(?<after>(>|,))"),
                    "${before}${typeParameter}${after}", null, 10),
                // public abstract class
2.9
                // class
30
                (new Regex(0"(public abstract|static) class"), "class", null, 0),
31
                // class GenericCollectionMethodsBase {
32
                // class GenericCollectionMethodsBase {
                                                          public:
33
                (new Regex(0"class ([a-zA-Z0-9]+)(\s+){"}, "class $1$2{"} + Environment.NewLine + "
                     public:", null, 0),
                // class GenericCollectionMethodsBase<TElement> {
35
                // template <typename TElement> class GenericCollectionMethodsBase { public:
36
                (\text{new Regex}(@"class}([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>([^{1}+)^{1}), "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
39
                    TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
                    tree, TElement* root)
                 (\text{new Regex}(0"\text{static }([a-zA-Z0-9]+) ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>\\(([^{\})\r\n]+)\\)"), 
40
                    "template <typename $3> static $1 $2($4)", null, 0),
                // interface IFactory<out TProduct> {
                // template <typename TProduct> class IFactory { public:
42
                (new Regex(@"interface (?<interface>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9
43
                     ,]+\dot{}>(?<whitespace>[^{]+){"}, "template <typename...> class ${interface};
                    template <typename ${typeParameters}> class
                    ${interface}<${typeParameters}>${whitespace}{" + Environment.NewLine + "
                    public:", null, 0),
                // template <typename TObject, TProperty, TValue>
                // template <typename TObject, typename TProperty, TValue>
45
                (new Regex(0"(?<before>template <((, )?typename [a-zA-Z0-9]+)+,</pre>
46
                    )(?<typeParameter>[a-zA-Z0-9]+)(?<after>(,|>))"), "${before}typename
                    $\{\typeParameter\}\{\text{after}\", \text{null, 10}\},
                // (this
47
                (new Regex(0"\(this "), "(", null, 0),
49
                // Func<TElement> treeCount
50
                // std::function<TElement()> treeCount
                (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", null,
52
                 \hookrightarrow 0),
                // Action<TElement> free
53
                // std::function<void(TElement)> free
54
                (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<void($1)> $2",
                    null, 0),
                // Predicate<TArgument> predicate
56
                // std::function<bool(TArgument)> predicate
```

```
(new Regex(0"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<br/>bool(1)>
                    $2", null, 0),
                // public static readonly EnsureAlwaysExtensionRoot Always = new
59
                    EnsureAlwaysExtensionRoot();
                // inline static EnsureAlwaysExtensionRoot Always;
                (new Regex(0"public static readonly (?<type>[a-zA-ZO-9]+) (?<name>[a-zA-ZO-9]+) =
61
                   new \k<type>\(\);"), "inline static ${type} ${name};", null, 0),
                // public static readonly string ExceptionContentsSeparator = "---";
62
                // inline static const char* ExceptionContentsSeparator = "---";
                (new Regex(@"public static readonly string (?<name>[a-zA-Z0-9_]+) =
64
                    ""(?<string>(\""|[^""\r\n])+)"";"), "inline static const char* ${name} =
                    \"${string}\";", null, 0),
                // private const int MaxPath = 92;
65
                // static const int MaxPath = 92;
66
                (new Regex(@"private (const|static readonly) ([a-zA-Z0-9]+) ([_a-zA-Z0-9]+) =
                    ([^; \r\n]^+);"), "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>(|
70
                    \rightarrow null, 0),
                // protected virtual
71
                // virtual
72
                (new Regex(0"protected virtual"), "virtual", null, 0),
                // protected abstract TElement GetFirst();
74
                // virtual TElement GetFirst() = 0;
75
                (new Regex(@"protected abstract ([^*,\r\n]+);"), "virtual $1 = 0;", null, 0),
                // TElement GetFirst();
77
                // virtual TElement GetFirst() = 0;
78
                (\text{new Regex}(@"([\r\n]+[ ]+)((?!\text{return})[a-zA-Z0-9]+ [a-zA-Z0-9]+\([^\)\r\n]*\))(;[
79
                    ]*[\r\n]+)"), "$1virtual $2 = 0$3", null, 1),
                // public virtual
80
                // virtual
81
                (new Regex(@"public virtual"), "virtual", null, 0),
82
                // protected readonly
83
                //
                (new Regex(@"protected readonly ")
                                                     "", null, 0),
85
                // protected readonly TreeElement[] _elements;
86
                // TreeElement _elements[N];
                (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+)([\\]]+)
88
                    ([_a-zA-ZO-9]+);"), "$2 $4[N];", null, 0),
                // protected readonly TElement Zero;
89
                // TElement Zero;
90
                (new Regex(0"(protected|private) readonly ([a-zA-Z<>0-9]+) ([a-zA-Z0-9]+);"), "$2

    $3;", null, 0),
                // private
92
                //
93
                (new Regex(@"(\W)(private|protected|public|internal) "), "$1", null, 0),
94
                // SizeBalancedTree(int capacity) => a = b;
95
                // SizeBalancedTree(int capacity) { a = b; }
96
                (new Regex(0"(^\s+)(override )?(void
                \rightarrow )?([a-zA-Z0-9]+)\(([^\(\r\n]*)\)\s+=>\s+([^;\r\n]+);"), "$1$2$3$4($5) { $6; }",
                    null, 0),
                // int SizeBalancedTree(int capacity) => a;
98
                // int SizeBalancedTree(int capacity) { return a; }
                (new Regex(0"(^{s+})(override)?([a-zA-Z0-9]+
100
                    )([a-zA-Z0-9]+)\(([^{(rn)}*)\)\s+=>\s+([^{(rn)}*);"), "$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),
102
103
                // => Integer<TElement>.Zero;
104
                // { return Integer<TElement>.Zero; }
                (new Regex(0"\)\s+=>\s+([^;\r\n]+?);"), ") { return $1; }", null, 0),
106
                // () { return avlTree.Count; }
107
                // [&]()-> auto { return avlTree.Count; }
                109
                \rightarrow null, 0)
                // Count => GetSizeOrZero(Root);
110
                               { return GetSizeOrZero(Root);
111
                // GetCount()
                (new Regex(0"([A-Z][a-z]+)\s+=>\s+([^;\r\n]+);"), "Get$1() { return $2; }", null, 0),
112
                // var
113
                // auto
114
                (new Regex(@"(\W)var(\W)"), "$1auto$2", null, 0),
                // unchecked
```

```
117
                  (new Regex(0"[\r\n]{2}\s*?unchecked\s*?$"), "", null, 0),
                 // $"
119
                 // "
120
                 (new Regex(@"\$"""), "\"", null, 0),
                 // Console.WriteLine("...")
122
                 // printf("...\n")
123
                 (new Regex(@"Console\.WriteLine\(""([^""\r\n]+)""\)"), "printf(\"$1\\n\")", null, 0),
124
                    throw new InvalidOperationException
                 // throw std::exception
126
                 (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
127

    std::exception", null, 0),
                 // override void PrintNode(TElement node, StringBuilder sb, int level)
128
                 // void PrintNode(TElement node, StringBuilder sb, int level) override
129
                 (new Regex(0"override ([a-zA-Z0-9 \*\+]+)(\([^\)\r\n]+?\))"), "$1$2 override", null,
130
                  \rightarrow 0),
                 // string
131
132
                 // char*
                 (new Regex(@"(\W)string(\W)"), "$1char*$2", null, 0),
133
                 // sbyte
134
                 // std::int8_t
135
                 (new Regex(@"(\W)sbyte(\W)"), "$1std::int8_t$2", null, 0),
                 // uint
137
                 // std::uint32_t
138
                  (new Regex(@"(\W)uint(\W)"), "$1std::uint32_t$2", null, 0),
139
                 // char*[] args
140
                 // char* args[]
141
                 (\text{new Regex}(@"([_a-zA-Z0-9:\*]?)\[\] ([a-zA-Z0-9]+)"), "$1 $2[]", null, 0),
142
                 // @object
                 // object
144
                 (\text{new Regex}(@"@([_a-zA-Z0-9]+)"), "$1", null, 0),
145
                 // using Platform.Numbers;
147
                 (\text{new Regex}(@"([\r\n]_{2}|^))\s*?using [\.a-zA-ZO-9]+;\s*?$"), "", null, 0),
148
                 // struct TreeElement { }
149
                 // struct TreeElement { };
                 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([^;])"), "$1
151
                     $2$3{$4};$5", null, 0),
                 // class Program { }
// class Program { };
152
153
                 (new Regex(0"(struct|class) ([a-zA-Z0-9]+[^r]*)([^r]+(?<indentLevel>[\t
                  \rightarrow ]*)?)\{([\S\s]+?[\r\n]+\k<indentLevel>)\}([^;]|$)"), "$1 $2$3{$4};$5", null, 0),
                 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
155
                 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
156
                 (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)"), "class $1 : public $2", null,
157
                  \rightarrow 0),
                 // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
158
                 // class IProperty : public ISetter<TValue, TObject>, IProvider<TValue, TObject>
159
                 (new Regex(@"(?<before>class [a-zA-Z0-9]+ : ((public [a-zA-Z0-9]+(<[a-zA-Z0-9]+(>]+(>]-zA-Z0-9])))
160
                      ,]+>)?.
                              )+)?)(?<inheritedType>(?!public)[a-zA-Z0-9]+(<[a-zA-Z0-9])
                      ,]+>)?)(?<after>(, [a-zA-Z0-9]+(?!>)|[ \r\n]+))"), "${before}public
                      ${inheritedType}${after}", null, 10),
                 // Insert scope borders.
                 // ref TElement root
162
                 // ~!root!~ref TElement root
163
                 (\text{new Regex}(@"(?<\text{definition}>(?<= |\()(\text{ref }[a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!\text{ref})))))
                      (?< variable>[a-zA-Z0-9]+)(?=\)|, | =))"), "~!${variable}!~${definition}", null, | =)
                     0)
                 // Inside the scope of ~!root!~ replace:
                 // root
                 // *root
167
                 (new Regex(0"(?<definition>~!(?<pointer>[a-zA-Z0-9]+)!~ref [a-zA-Z0-9]+
168
                      \k<pointer>(?=\)|, | =))(?<before>((?<!~!\k<pointer>!~)(.|\n))*?)(?<prefix>(\W
                      |\cdot|()\rangle k<pointer>(?<suffix>( |\cdot|()|;|,))"),
                      "${definition}${before}${prefix}*${pointer}${suffix}", null, 70),
                 // Remove scope borders.
                     ~!root!~
170
                 //
171
                  (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
                 // ref auto root = ref
173
                 // ref auto root =
174
                 (\text{new Regex}(@"\text{ref}([a-zA-Z0-9]+)([a-zA-Z0-9]+) = \text{ref}(\W)"), "$1* $2 =$3", null, 0),
175
                 // *root = ref left;
                 // root = left;
177
                 (\text{new Regex}(@"\*([a-zA-Z0-9]+) = ref([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", null, 0),
178
                 // (ref
                 // (left)
180
```

```
(\text{new Regex}(@"\ref ([a-zA-Z0-9]+)(\)|\(|,)"), "($1$2", null, 0),
181
                                              ref TElement
                                              TElement*
183
                                      (new Regex(0"(|\cdot|)ref ([a-zA-Z0-9]+)"), "$1$2*", null, 0),
184
                                      // ref sizeBalancedTree.Root
                                      // &sizeBalancedTree->Root
186
                                      (\text{new Regex}(@"\text{ref}([a-zA-Z0-9]+)\.([a-zA-Z0-9]*]+)"), "&$1->$2", null, 0),
187
                                      // ref GetElement(node).Right
188
                                      // &GetElement(node)->Right
                                      (new Regex(0"ref ([a-zA-Z0-9]+)\(([a-zA-Z0-9]*]+)\)\.([a-zA-Z0-9]+)"),
190
                                               "&$1($2)->$3", null, 0),
                                      // GetElement(node).Right
191
                                      // GetElement(node)->Right
192
                                      (\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
193
                                             null, 0),
                                      // [Fact] \npublic static void SizeBalancedTreeMultipleAttachAndDetachTest()
194
                                      // TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
195
                                      (\text{new Regex}(@'\[\text{Fact}\] [\s\n] + (\text{static})?void([a-zA-ZO-9]+)\(\)"), "TEST_METHOD($2)",
196
                                              null,
                                                           0),
                                      // class TreesTests
                                      // TEST_CLASS(TreesTests)
                                      (new Regex(@"class ([a-zA-ZO-9]+)Tests"), "TEST_CLASS($1)", null, 0),
199
                                      // Assert.Equal
                                      // Assert::AreEqual
201
                                      (new Regex(@"Assert\.Equal"), "Assert::AreEqual", null, 0),
202
                                            TElement Root;
203
                                      // TElement Root = 0;
                                      (new Regex(0"(\r?\n[\t]+)([a-zA-Z0-9:_]+(?<!return)) ([_a-zA-Z0-9]+);"), "$1$2 $3 =
205
                                              0;", null, 0),
                                      // TreeElement _elements[N];
206
                                      // TreeElement _elements[N] = { {0} }
                                      (\text{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];
209
                                      // TElement path[MaxPath] = { {0} }
210
                                      (new Regex(0"(\r?\n[\t]+)[a-zA-Z0-9]+ ([a-zA-Z0-9]+) = new
                                                ([a-zA-Z0-9]+)\setminus[([-a-zA-Z0-9]+)\setminus];"), "$1$3 $2[$4] = { {0} };", null, 0),
                                      // Insert scope borders.
212
                                      // auto added = new HashSet<TElement>();
213
                                      // ~!added!~std::unordered_set<TElement> added;
214
                                      (new Regex(0"auto (?<variable>[a-zA-Z0-9]+) = new
                                               HashSet < (? < element > [a-zA-Z0-9] +) > ( ); "),
                                      "~!${variable}!~std::unordered_set<${element}> ${variable};", null, 0),
// Inside the scope of ~!added!~ replace:
216
                                      // added.Add(node)
217
                                      // added.insert(node)
218
                                      (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?<_</pre>
219
                                               !~!\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:
                                      // added.Remove(node)
221
                                      // added.erase(node)
222
223
                                      (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?< |</pre>
                                                !^{\cdot} k< variable>!^{\cdot} (.|n))*?) k< variable>\.Remove\((?< argument>[a-zA-Z0-9]+)\)"),
                                               "${scope}${separator}${before}${variable}.erase(${argument})", null, 10),
                                      // if (added.insert(node))
                                      // if (!added.contains(node)) { added.insert(node);
225
                                      (\text{new Regex}(@"if \setminus ((?<\text{variable}=a-zA-Z0-9]+) \setminus (?<\text{argument}=a-zA-Z0-9]+) \setminus) (?_{\text{local}})
226
                                               \operatorname{separator}[\t]*[\r\n]+)(?\operatorname{indent}[\t]*){"}, "if
                                               (!${variable}.contains(${argument})))${separator}${indent}{" +
                                              Environment.NewLine + "${indent}
                                                                                                                                   ${variable}.insert(${argument});", null, 0),
                                      // Remove scope borders.
227
                                     //
                                              ~!added!′
228
229
                                      (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
                                      // Insert scope borders.
231
                                      // auto random = new System.Random(0);
232
                                      // std::srand(0);
                                      (\text{new Regex}(@"[a-zA-Z0-9]) + ([a-zA-Z0-9]) + ) = \text{new}
234
                                               (System\.)?Random\(([a-zA-Z0-9]+)\);"), "~!$1!~std::srand($3);", null, 0),
                                      // Inside the scope of ~!random!~ replace:
235
                                      // random.Next(1, N)
// (std::rand() % N) + 1
236
237
                                      (new Regex(0"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<br/>before>((?<|
238
                                                !^*[\k<\text{variable}]^*(.\n))*?)\k<\text{variable}^.\next^((?<from>[a-zA-ZO-9]+))
                                               (?<to>[a-zA-Z0-9]+)\)"), "$\{scope\}$\{separator\}$\{before\}(std::rand() % $\{to\}) + (?<to>[a-zA-Z0-9]+)\)"), "$\{scope\}$\{separator\}$\{to]$\{scope\}$\{separator\}$\{scope\}$\{separator\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{scope\}$\{s
                                              ${from}", null, 10),
```

```
// Remove scope borders.
239
                                  // ~!random!
                                  //
241
                                  (new Regex(0"^{-1}(?<pointer>[a-zA-Z0-9]+)!^{-1}), "", null, 5),
242
                                  // Insert method body scope starts.
                                  // void PrintNodes(TElement node, StringBuilder sb, int level) {
244
                                  // void PrintNodes(TElement node, StringBuilder sb, int level) {/*method-start*/
245
                                   (new Regex(0"(?<start>\r?\n[\t]+)(?<prefix>((virtual))?[a-zA-Z0-9:_]+
246
                                           )?) (?\mbox{method}[a-zA-Z][a-zA-Z0-9]*) ((?\mbox{arguments}[^\)]*) \) (?\mbox{override}(
                                           override)?)(?\langle separator\rangle[ \t\r\n]*)\{(?\langle end\rangle[^{~}])"), "$\{start\}$\{prefix\}$\{method\}_{\n}$ is the constant of the constant o
                                           (${arguments})${override}${separator}{/*method-start*/${end}", null,
                                          0),
                                   // Insert method body scope ends.
                                  // {/*method-start*/...}
248
                                  // {/*method-start*/.../*method-end*/}
249
                                   (new Regex(@"\{/\*method-start\*/(?<body>((?<bracket>\{)|(?<-bracket>\})|[^\{\}]*)+) |
250
                                          \}"), "{/*method-start*/${body}/*method-end*/}", null,
                                          0).
                                  // Inside method bodies replace:
251
                                  // GetFirst(
252
253
                                  // this->GetFirst(
                                  //(new Regex(@"(?<separator>(\(|, |([\\]) |return ))(?<!(->|\*
                                           ))(?<method>(?!sizeof)[a-zA-Z0-9]+)\((?!\)\{)"),
                                           "${separator}this->${method}(", null, 1),
                                   (\texttt{new Regex}(@"(?<scope>/\\*method-start\\*/)(?<before>((?<!/\\*method-end\\*/)(.|\\n))*?)(_{|})()
                                           ?<separator>[\W](?<!(::\\.|->)))(?<method>(?!sizeof)[a-zA-Z0-9]+)\((?!\)
                                           \{\) (?\langle after\rangle(.|\n)*?) (?\langle scopeEnd\rangle/\method-end\*/)"),
                                           "${scope}${before}${separator}this->${method}(${after}${scopeEnd}", null, 100),
                                  // Remove scope borders.
256
                                  // /*method-start*/
257
                                   (new Regex(0"/\*method-(start|end)\*/"), "", null, 0),
                                  // throw new ArgumentNullException(argumentName, message);
260
                                  // throw std::invalid_argument(((std::string)"Argument
261
                                          ").append(argumentName).append(" is null: ").append(message).append("."));
                                   (new Regex(@"throw new
                                           ArgumentNullException\((?<argument>[a-zA-Z]*[Aa]rgument[a-zA-Z]*),
                                           (?\langle message \rangle [a-zA-Z] * [Mm] essage [a-zA-Z] *) \rangle;"), "throw"
                                         std::invalid_argument(((std::string)\"Argument \").append(${argument}).append(\"
                                          is null: \").append(${message}).append(\".\"));", null, 0),
                                  // throw new ArgumentException(message, argumentName);
263
                                  // throw std::invalid_argument(((std::string)"Invalid
264
                                          ").append(argumentName).append(" argument: ").append(message).append("."))
                                   (new Regex(@"throw new ArgumentException\(((?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*),
265
                                           (?\langle argument \rangle [a-zA-Z] * [Aa] rgument [a-zA-Z] *) \rangle;"), "throw"
                                          std::invalid_argument(((std::string)\"Invalid \").append(${argument}).append(\"
                                           argument: \").append(${message}).append(\".\"));", null, 0),
                                  // throw new NotSupportedException();
                                  // throw std::logic_error("Not supported exception.");
267
                                   (new Regex(@"throw new NotSupportedException\(\);"), "throw std::logic_error(\"Not
268
                                          supported exception.\");", null, 0),
                                  // throw new NotImplementedException();
                                  // throw std::logic_error("Not implemented exception.");
270
                                   (\texttt{new Regex}(\texttt{@"throw new NotImplementedException}\cdot{(`\);"), "throw std::logic\_error(\"NotImplementedException)\cdot{(`\);"), "throw std::logic\_error(\"NotImplementedException)\cdot{('\);"), "throw std::logic\_error(\"N
271
                                          implemented exception.\");", null, 0),
272
                          }.Cast<ISubstitutionRule>().ToList();
273
274
                          public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
275
                                  // ICounter<int, int> c1;
277
                                  // ICounter<int, int>* c1;
278
                                   (new Regex(0"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^>\r\n]+>)?)
                                           (?<variable>[_a-zA-Z0-9]+);"), "${abstractType}* ${variable};", null, 0),
                                  // (expression)
                                  // expression
281
                                   (\text{new Regex}(@"(\(| )(([a-zA-Z0-9_{*:}]+))(,| |;|))"), "$1$2$3", null, 0),
282
                                  // (method(expression))
                                  // method(expression)
284
                                   (new Regex(0"(?<firstSeparator>(\())
285
                                          ))\((?<method>[a-zA-Z0-9_\->\*:]+)\((?<expression>((?<parenthesis>\()|(?<-parent
                                          | | ;|\)))"), "${firstSeparator}${method}(${expression})${lastSeparator}", null, 0),
                                  // return ref _elements[node];
286
                                  // return &_elements[node];
```

```
(\text{new Regex}(@"\text{return ref}([_a-zA-Z0-9]+))[([_a-zA-Z0-9]*]+))];"), "return &$1[$2];",
288
                  \rightarrow null, 0),
                 // default
289
                 // 0
290
                 (new Regex(@"(\W)default(\W)"), "${1}0$2", null, 0),
291
                 // //#define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
292
                 (\text{new Regex}(@'')/[ t]*\#\text{define}[ t]+[_a-zA-Z0-9]+[ t]*"), "", null, 0),
294
                 // #if USEARRAYPOOL\r\n#endif
295
                 (new Regex(@"#if [a-zA-Z0-9]+\s+#endif"), "", null, 0),
297
                 // [Fact]
298
                 //
                 (new Regex(@"(?<firstNewLine>\r?\n|\A)(?<indent>[\t
                     ]+) \[[a-zA-Z0-9]+(\((?\expression\)((?\expression\))|[\gamma()\r_|\)
                     \n]*)+)(?(parenthesis)(?!)))))?][ \t]*(\r?\n\k<indent>)?"),
                     "${firstNewLine}${indent}", null, 5),
                 // \n ... namespace
301
                 // namespace
302
                 (new Regex(0"(S[\r\n]{1,2})?[\r\n]+namespace"), "$1namespace", null, 0),
303
                 // \n ... class
                 // class
305
                 (\text{new Regex}(0"(S[\r\n]{1,2})?[\r\n]+class"), "$1class", null, 0),
306
             }.Cast<ISubstitutionRule>().ToList();
307
308
             public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
             → base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
310
            public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
311
        }
313
     ./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()
                 const string helloWorldCode = @"using System;
    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"");
23
24
    };";
25
                 var transformer = new CSharpToCppTransformer();
26
                 var actualResult = transformer.Transform(helloWorldCode, new Context(null));
                 Assert.Equal(expectedResult, actualResult);
             }
29
        }
30
    }
```

31

Index

 $./Platform. Regular Expressions. Transformer. CSharp ToCpp. Tests/CSharp ToCpp Transformer Tests. cs, \ 6../Platform. Regular Expressions. Transformer. CSharp ToCpp/CSharp ToCpp Transformer. cs, \ 1...$