```
LinksPlatform's Platform.RegularExpressions.Transformer.CSharpToCpp Class Library
       ./Platform. Regular Expressions. Transformer. CSharp To Cpp/CSharp To Cpp Transformer. cs \\
     using System;
 1
     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
                         // {
21
                         (new Regex(0"{\s+[\r\n]+"), "{" + Environment.NewLine, null, 0),
22
                         // Platform.Collections.Methods.Lists
                         // Platform::Collections::Methods::Lists
                         (new Regex(0"(namespace[\r^n+?)\.([\r^n+?)"), "$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 static bool CollectExceptions { get; set; }
2.9
                         // public static bool CollectExceptions;
30
                         (new Regex(@"(?<before>(private|protected|public)( static?) [^\r\n]+
31
                              )(?<ame>[a-zA-Z0-9]+) {[^;}]*(?<=\W)get;[^;}]*(?<=\W)set;[^;}]*),
                               "${before}${name};", null, 0),
                         // public abstract class
32
                         // class
33
                         (new Regex(@"(public abstract|static) class"), "class", null, 0),
34
                         // class GenericCollectionMethodsBase {
35
                         // class GenericCollectionMethodsBase { public:
36
                         (new Regex(0"class ([a-zA-Z0-9]+)(\s+){"), "class $1$2{"} + Environment.NewLine + "
                                 public:", null, 0),
                         // class GenericCollectionMethodsBase<TElement> {
                         // template <typename TElement> class GenericCollectionMethodsBase { public:
39
                         (\text{new Regex}(@"class}([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>([^{1}+)^{1}), "template < typename $2>
40
                              class $1$3{" + Environment.NewLine + "
                                                                                                   public:", null, 0),
                         // static void
                              TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
                              tree, TElement* root)
                         // template<typename T> static void
42
                              TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>

    tree, TElement* root)

                         (\text{new Regex}(@"\text{static }([a-zA-Z0-9]+) ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>(([^\)\r\n]+)\)"),
                               "template <typename $3> static $1 $2($4)", null, 0),
                         // interface IFactory<out TProduct> {
                         // template <typename TProduct> class IFactory { public:
                         (new Regex(@"interface (?<interface>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9]
46
                               ,]+)>(?<whitespace>[^{]+){"}, "template <typename...> class ${interface};
                               template <typename ${typeParameters}> class
                               ${interface}<${typeParameters}>${whitespace}{" + Environment.NewLine + "
                               public:", null, 0)
                         // template <typename TObject, TProperty, TValue>
// template <typename TObject, typename TProperty, TValue>
                         49
                               ) (?<typeParameter>[a-zA-Z0-9]+) (?<after>(,|>))"), "${before}typename
                              ${typeParameter}${after}", null, 10),
                         // Insert markers
50
                         // private static void BuildExceptionString(this StringBuilder sb, Exception
                               exception, int level)
                         // /*~extensionMethod~BuildExceptionString~*/private static void
                         \hookrightarrow BuildExceptionString(this StringBuilder sb, Exception exception, int level)
                         (new Regex(0"private static [^{r}] + (?^{a}) + (2^{a}) + (2^{a})
53
                               "/*~extensionMethod~${name}~*/$0", null, 0),
```

```
// Move all markers to the beginning of the file.
54
                (new Regex(@"\A(?<before>[^\r\n]+\r?\n(.|\n)+)(?<marker>/\*~extensionMethod~(?<name>_
                    [a-zA-Z0-9]+)^*/"), "${marker}${before}", null,
                    10),
                // /*~extensionMethod~BuildExceptionString~*/...sb.BuildExceptionString(exception.In_
56

    nerException, level +

                   1);
                // /*~extensionMethod~BuildExceptionString~*/...BuildExceptionString(sb,

→ exception.InnerException, level + 1);
                5.8
                    iable > [_a-zA-ZO-9]+) \.\k<name > ("), "${before}${name}(${variable}, ", null, "), "}
                   50).
                // Remove markers
59
                // /*~extensionMethod~BuildExceptionString~*/
60
                (new Regex(0"/*extensionMethod[a-zA-Z0-9]+<math>*/*), "", null, 0),
62
                // (this
63
64
                (new Regex(@"\(this "), "(", null, 0),
                // public static readonly EnsureAlwaysExtensionRoot Always = new
66
                   EnsureAlwaysExtensionRoot();
                // inline static EnsureAlwaysExtensionRoot Always;
                (\text{new Regex}(@"\text{public static readonly} (?<\text{type}=[a-zA-ZO-9]+) (?<\text{name}=[a-zA-ZO-9]+) =
                → new \k<type>\(\);"), "inline static ${type} ${name};", null, 0),
                // public static readonly string ExceptionContentsSeparator = "---"
69
                // inline static const char* ExceptionContentsSeparator = "---";
70
                (new Regex(@"public static readonly string (?<name>[a-zA-ZO-9_]+) =
71
                   ""(?<string>(\""|[^""\r\n])+)"";"), "inline static const char* ${name} =
                   \"${string}\";", null, 0),
                // private const int MaxPath = 92;
                // static const int MaxPath = 92;
7.3
                (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>(|
76
77
                   null, 0),
                // protected virtual
                // virtual
79
                (new Regex(0"protected virtual"), "virtual", null, 0),
80
                // protected abstract TElement GetFirst();
                // virtual TElement GetFirst() = 0;
82
                (new Regex(0"protected abstract ([^{r}\r\n]+);"), "virtual $1 = 0;", null, 0),
83
                  TElement GetFirst();
84
                // virtual TElement GetFirst() = 0;
                (\text{new Regex}(@"([\r\n]+[ ]+)((?!\text{return})[a-zA-Z0-9]+ [a-zA-Z0-9]+\([^\)\r\n]*\))(;[
86
                \rightarrow ]*[\r\n]+)"), "$1virtual $2 = 0$3", null, 1),
                // public virtual
                // virtual
                (new Regex(@"public virtual"), "virtual", null, 0),
                // protected readonly
90
91
                (new Regex(@"protected readonly "), "", null, 0),
                // protected readonly TreeElement[] _elements;
93
                // TreeElement _elements[N];
94
                (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+)([\[\]]+)
                   ([_a-zA-Z0-9]+);"), "$2 $4[N];", null, 0),
                // protected readonly TElement Zero;
                // TElement Zero;
97
                (new Regex(0"(protected|private) readonly ([a-zA-Z<>0-9]+) ([_a-zA-Z0-9]+);"), "$2
98
                \rightarrow $3;", null, 0),
                // private
                //
100
                (new Regex(@"(\W)(private|protected|public|internal) "), "$1", null, 0),
101
                // static void NotImplementedException(ThrowExtensionRoot root) => throw new
102
                → NotImplementedException();
                // static void NotImplementedException(ThrowExtensionRoot root) { return throw new
103
                → NotImplementedException(); }
                (new Regex(@"(^s+)(template <[^>\r\n]+\))?(static )?(override )?([a-zA-Z0-9]+
104
                   ([a-zA-Z0-9]+)(([^{(r)n}*)))
                    throw$8; }"
                                null, 0),
                // SizeBalancedTree(int capacity) => a = b;
                // SizeBalancedTree(int capacity) { a = b; }
106
```

```
(\text{new Regex}(@"(^\s+)(\text{template }\c)^{r})+\c)^{(\text{static })^{(\text{override })^{(\text{void })^{-1}}})
107
                     null, 0),
                // int SizeBalancedTree(int capacity) => a;
108
                // int SizeBalancedTree(int capacity) { return a; }
109
                (new Regex(0"(^\s+)(template \<[^>\r\n]+\>)?(static )?(override )?([a-zA-Z0-9]+
110
                    )([a-zA-Z0-9]+)\(([^\(\r\n]*)\)\s+=>\s+([^;\r\n]+);"), "$1$2$3$4$5$6($7) { return $8; }", null, 0),
                    () => Integer<TElement>.Zero,
                // () { return Integer<TElement>.Zero; }
112
                (\text{new Regex}(@''())s=-)s+([^,;\r\n]+?),"), "() { return $1; },", null, 0),
113
                // => Integer<TElement>.Zero;
                // { return Integer<TElement>.Zero; }
                (new Regex(0"\)\s+=>\s+([^;\r\n]+?);"), ") { return $1; }", null, 0),
116
                // () { return avlTree.Count; }
117
                 // [&]()-> auto { return avlTree.Count;
118
                (new Regex(0", \(\) { return ([^;\r\n]+); }"), ", [&]()-> auto { return $1; }",
119
                    null, 0)
                // Count => GetSizeOrZero(Root);
120
                // GetCount() { return GetSizeOrZero(Root); }
                (new Regex(0"(\W)([A-Z][a-zA-Z]+)\s+=>\s+([^;\r\n]+);"), "$1Get$2() { return $3; }",
                    null, 0),
                // Func<TElement> treeCount
123
                // std::function<TElement()> treeCount
124
                 (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", null,
125
                // Action<TElement> free
                // std::function<void(TElement)> free
127
                (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<void($1)> $2",
128
                    null, 0).
                // Predicate<TArgument> predicate
                // std::function<bool(TArgument)> predicate
130
                (new Regex(0"Predicate<(([a-zA-Z0-9]+)) (([a-zA-Z0-9]+)"), "std::function<br/>bool($1)>
131
                 \rightarrow $2", null, 0),
                // var
132
                 // auto
                 (new Regex(@"(\W)var(\W)"), "$1auto$2", null, 0),
134
                // unchecked
135
136
                (\text{new Regex}(0"[\r\n]{2}\s*?unchecked\s*?$"), "", null, 0),
                // throw new InvalidOperationException
138
                // throw std::runtime_error
139
                 (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
                    std::runtime_error", null, 0)
                // void RaiseExceptionIgnoredEvent(Exception exception)
141
                // void RaiseExceptionIgnoredEvent(const std::exception& exception)
142
                (new Regex(@"(\(|, ))(System\.Exception|Exception)( |\))"), "$1const
143
                    std::exception&$3", null, 0),
                // EventHandler<Exception>
                // EventHandler<std::exception>
145
                 (new Regex(@"(\W)(System\.Exception|Exception)(\W)"), "$1std::exception$3", null, 0),
146
147
                   override void PrintNode(TElement node, StringBuilder sb, int level)
                 // void PrintNode(TElement node, StringBuilder sb, int level) override
                (new Regex(0"override ([a-zA-Z0-9 \*\+]+)(\([^\)\r\n]+?\))"), "$1$2 override", null,
149
                    0)
                // string
150
                // const char*
                (new Regex(@"(\W)string(\W)"), "$1const char*$2", null, 0),
152
                // sbyte
153
                // std::int8_t
154
                 (new Regex(@"(\W)sbyte(\W)"), "$1std::int8_t$2", null, 0),
155
                // uint
156
                // std::uint32_t
157
                 (new Regex(@"(\W)uint(\W)"), "$1std::uint32_t$2", null, 0),
                // char*[] args
159
                // char* args[]
160
                 (\text{new Regex}(@"([_a-zA-Z0-9:\*]?)\[\] ([a-zA-Z0-9]+)"), "$1 $2[]", null, 0),
161
                // @object
                // object
163
                (new Regex(0"0([_a-zA-Z0-9]+)"), "$1", null, 0),
164
                // using Platform.Numbers;
166
                (\text{new Regex}(@"([\r\n]{2}|^)\s*?using [\.a-zA-ZO-9]+;\s*?$"), "", null, 0),
167
                // struct TreeElement {
168
                 // struct TreeElement {
169
                 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([^;])"), "$1
170
                    $2$3{$4};$5", null, 0),
```

```
// class Program {
                  (\text{new Regex}(@^{\text{"}}(\text{struct}|\text{class}) ([a-zA-Z0-9]+[^\n]*)([\n]+(?<\text{indentLevel}>[\t]))
173
                      ]*)?)\{([\S\s]+?[\r\n]+\k<indentLevel>)\}([^;]|$)"), "$1 $2$3{$4};$5", null, 0),
                  // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
174
                  // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
175
                  (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)"), "class $1 : public $2", null,
                  \rightarrow 0),
                  // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
177
                  // class IProperty : public ISetter<TValue, TObject>, IProvider<TValue, TObject> (new Regex(@"(?<before>class [a-zA-ZO-9]+ : ((public [a-zA-ZO-9]+(<[a-zA-ZO-9]+)))
178
179
                      ,]+>)?, )+)?)(?<inheritedType>(?!public)[a-zA-Z0-9]+(<[a-zA-Z0-9]+(^{2}
                      ,]+>)?)(?<after>(, [a-zA-ZO-9]+(?!>)|[ \r\n]+))"), "${before}public
                      ${inheritedType}${after}", null, 10),
                  // Insert scope borders.
180
                  // ref TElement root
181
                  // ~!root!~ref TElement root
182
                  (new Regex(0"(?<definition>(?<= |\()(ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!ref))
                      (?\langle variable \rangle [a-zA-Z0-9]+)(?= \rangle |, | =))"), "^! {\{variable\}!^{\{definition\}}", null, \}}
                  // Inside the scope of ~!root!~ replace:
                  // root
185
                  // *root
186
                  (\text{new Regex}(@"(?<\text{definition}^{"}(?<\text{pointer}[a-zA-Z0-9]+))" [a-zA-Z0-9]+)
                      \k<pointer>(?=\)|, | =))(?<before>((?<!~!\k<pointer>!~)(.|\n))*?)(?<prefix>(\W
                      |\())\k<pointer>(?<suffix>( |\)|;|,))"),
                      "${definition}${before}${prefix}*${pointer}${suffix}", null, 70),
                  // Remove scope borders.
                  //
                      '!root!'
189
190
                  (new Regex(0"^{!}(?<pointer>[a-zA-Z0-9]+)!^{"}), "", null, 5),
                  // ref auto root = ref
192
                  // ref auto root =
193
                  (\text{new Regex}(@"\text{ref}([a-zA-Z0-9]+)([a-zA-Z0-9]+) = \text{ref}(\W)"), "$1* $2 =$3", null, 0),
                  // *root = ref left;
195
                  // root = left;
196
                  (\text{new Regex}(@'')*([a-zA-Z0-9]+) = \text{ref}([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", null, 0),
197
                  // (ref
                          left)
                  // (left)
199
                  (new Regex(0"\(ref ([a-zA-Z0-9]+)(\)|\(|,)"), "($1$2", null, 0),
200
                     ref TElement
                      TElement*
202
                  (\text{new Regex}(@"(|\()\text{ref}([a-zA-Z0-9]+)"), "$1$2*", null, 0),
203
                    ref sizeBalancedTree.Root
204
                  // &sizeBalancedTree->Root
                  (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)"), "&1->", null, 0),
206
                  // ref GetElement(node).Right
207
                  // &GetElement(node)->Right
                  (new Regex(0"ref ([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"),
209
                      "&$1($2)->$3", null, 0),
                  // GetElement(node).Right
210
211
                  // GetElement(node)->Right
                  (\text{new Regex}(@"([a-zA-Z0-9]+))(([a-zA-Z0-9]*)+))).([a-zA-Z0-9]+)"), "$1($2)->$3",
                      null
                  // [Fact]\npublic static void SizeBalancedTreeMultipleAttachAndDetachTest()
213
                  // TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
214
                  (\text{new Regex}(@'\[\text{Fact}\] [\s\n] + (\text{static})?void ([a-zA-Z0-9]+)\(\)"), "TEST_METHOD($2)",
215
                   \rightarrow null, 0),
                  // class TreesTests
217
                  // TEST_CLASS(TreesTests)
                  (new Regex(@"class ([a-zA-ZO-9]+)Tests"), "TEST_CLASS($1)", null, 0),
218
                   / Assert.Equal
219
                  // Assert::AreEqual
220
                  (new Regex(@"Assert\.Equal"), "Assert::AreEqual", null, 0),
221
                  // $"Argument {argumentName} is null."
222
                  // ((std::string) "Argument ").append(argumentName).append(" is null.").data()
                  (new Regex(@"\$""(?<\ieft>(\\""|\[^\"\r\n])*){(?<expression>[_a-zA-Z0-9]+)}(?<right>(\__
224
                       ""|[^""\r\n])*)""")
                      "((std::string)$\"${left}\").append(${expression}).append(\"${right}\").data()",
                      null, 10),
                  // $"
225
                  // "
226
                  (new Regex(@"\$"""), "\"", null, 0),
                  // Console.WriteLine("...")
                  // printf("...\n")
229
                  (new Regex(@"Console\.WriteLine\(""([^""\r\n]+)""\)"), "printf(\"$1\\n\")", null, 0),
230
                  // TElement Root;
```

```
// TElement Root = 0;
232
                                   (\text{new Regex}(@"(\r?\n[\t]+)([a-zA-Z0-9:_]+(?<!\text{return})) ([_a-zA-Z0-9]+);"), "$1$2 $3 =
                                           0;"
                                                    null, 0)
                                   // TreeElement _elements[N];
234
                                   // TreeElement _elements[N] = { {0} };
235
                                   (\text{new Regex}(@"(\r?\n[\t]+)([a-zA-Z0-9]+) ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9]+)\];"),
236
                                           "$1$2 $3[$4] = { {0} }; ", null, 0),
                                   // auto path = new TElement[MaxPath];
                                   // TElement path[MaxPath] = { {0} }
238
                                   (\text{new Regex}(0"(\r?\n[\t]+)[a-zA-ZO-9]+([a-zA-ZO-9]+) = \text{new})
239
                                            ([a-zA-Z0-9]+)\setminus[([_a-zA-Z0-9]+)\setminus];"), "$1$3 $2[$4] = { {0} };", null, 0),
                                   // Insert scope borders.
240
                                   // auto added = new StringBuilder();
                                   // /*~sb~*/std::string added;
242
                                   (new Regex(@"(auto|(System\.Text\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new
243
                                            (System\.Text\.)?StringBuilder\(\);"), "/*~${variable}~*/std::string
                                           ${variable}; ", null, 0),
                                   // static void Indent(StringBuilder sb, int level)
                                   // static void Indent(/*~sb~*/StringBuilder sb, int level)
245
                                   (new Regex(@"(?<start>, |\()(System\.Text\.)?StringBuilder
246
                                            (?<variable>[a-zA-Z0-9]+)(?<end>,|\))"), "${start}/*~${variable}~*/std::string&
                                   $\ \text{variable}$\{\text{end}\}\", null, 0),
// Inside the scope of \[ \text{!added!}\] replace:
247
                                   // sb.ToString()
                                   // sb.data()
249
                                   (new Regex(0"(?<scope>/\*^(?<variable>[a-zA-Z0-9]+)^\*/)(?<separator>.|\n)(?<before>|
250
                                            ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.ToString\(\)"),
                                           "${scope}${separator}${before}${variable}.data()", null, 10),
                                   // sb.AppendLine(argument)
                                   // sb_append(argument).append('\n')
252
                                   (new Regex(0"(?<scope>/\sqrt{*}(?<variable>[a-zA-Z0-9]+)^**/)(?<separator>.|\n)(?<before>|
253
                                            ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.AppendLine\((?<argument>[^\),\_
                                           r\n]+)\)")
                                            \label{lem:standard} $$\{scope\}$\{separator\}$\{before\}$\{variable\}.append($\{argument\}).append('\n')", append('\n')", append('\n'
                                    \hookrightarrow
                                           null, 10)
                                   // sb.Append('\t', level);
254
                                   // sb.append(level, '\t');
255
                                   (\text{new Regex}(@"(?<scope>/)*^(?<variable>[a-zA-Z0-9]+)^**/)(?<separator>.|\n)(?<before>|
                                            ((?<!/*^k<variable>^**/)(.|\n))*?)\k<variable>\.Append\('(?<character>[^'\r\n]_|)
                                                   , (?<count>[^\),\r\n]+)\)")
                                           "${scope}${separator}${before}${variable}.append(${count}, '${character}')",
                                           null, 10),
                                   // sb.Append(argument)
                                   // sb.append(argument)
                                   (new Regex(0"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>)
259
                                            ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.Append\((?<argument>[^\),\r\n]
                                           +)\)", "${scope}${separator}${before}${variable}.append(${argument})", null,
                                          10),
                                   // Remove scope borders.
260
                                   // /*~sb~*/
                                   //
                                   (new Regex(0"/*(?<pointer>[a-zA-Z0-9]+)*/"), "", null, 0),
263
                                   // Insert scope borders.
264
                                        auto added = new HashSet<TElement>();
                                   // ~!added!~std::unordered_set<TElement> added;
266
                                   (new Regex(0"auto (?<variable>[a-zA-Z0-9]+) = new
267
                                           HashSet < (? < element > [a-zA-Z0-9] +) > \setminus (\);"),
                                            "~!${variable}!~std::unordered_set<${element}> ${variable};", null, 0),
                                   // Inside the scope of ~!added!~ replace:
268
                                        added.Add(node)
269
                                   // added.insert(node)
270
                                   (new\ Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?<|))(?<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>.|\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)
271
                                           !^*!\k<\text{variable}:^*)(.|\n))*?)\k<\text{variable}\. Add\((?<\text{argument}=[a-zA-Z0-9]+)\)"),
                                           "${scope}${separator}${before}${variable}.insert(${argument})", null, 10),
                                   // Inside the scope of "!added!" replace:
272
                                   // added.Remove(node)
                                   // added.erase(node)
274
                                   (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?< |</pre>
275
                                            !^{\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);
277
                                   (new Regex(@"if \((?<variable>[a-zA-Z0-9]+)\.insert\((?<argument>[a-zA-Z0-9]+)\)))(?|
278
                                           \operatorname{separator}[\t] *[\r\n] +) (?< \operatorname{indent}[\t] *) {"), "if}
                                            (!${variable}.contains(${argument}))${separator}${indent}{" +
                                                                                                                         ${variable}.insert(${argument});", null, 0),
                                           Environment.NewLine + "${indent}
```

```
// Remove scope borders.
                 // ~!added!
                 11
281
                 (new Regex(0"^{!}(?<pointer>[a-zA-Z0-9]+)!^{"}), "", null, 5),
282
                 // Insert scope borders.
                 // auto random = new System.Random(0);
284
                 // std::srand(0);
285
                 (new Regex(0"[a-zA-Z0-9]) + ([a-zA-Z0-9]) = new
286
                 \hookrightarrow (System\.)?Random\(([a-zA-Z0-9]+)\);"), "~!$1!~std::srand($3);", null, 0), // Inside the scope of ~!random!~ replace:
                 // random.Next(1, N)
288
                 // (std::rand() % N) + 1
289
                 (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?<|</pre>
290
                      !^{\cdot} \k< variable>!^{\cdot} (.|n))*?) \k< variable> \. Next \ ((?< from>[a-zA-Z0-9]+)
                      (?<to>[a-zA-Z0-9]+)))"), "${scope}${separator}${before}(std::rand() % ${to}) +
                      ${from}", null, 10),
                 // Remove scope borders.
291
                    "!random!"
                 //
                 //
293
                 (new Regex(0"^{!}(?<pointer>[a-zA-Z0-9]+)!^{"}), "", null, 5),
294
                 // Insert method body scope starts.
                 // void PrintNodes(TElement node, StringBuilder sb, int level)
296
                 // void PrintNodes(TElement_node, StringBuilder sb, int level) {/*method-start*/
297
                  (new Regex(@"(?<start>\r?\n[\t]+)(?<prefix>((virtual))?[a-zA-Z0-9:_]+
298
                      )?) (?\mbox{method}[a-zA-Z][a-zA-Z0-9]*) ((?\mbox{arguments}[^\)]*) () (?\mbox{override}(
                      override)?)(?<separator>[ \t\r\n]*)\{(?<end>[^~])"), "${start}${prefix}${method}_
                      (${arguments})${override}${separator}{/*method-start*/${end}", null,
                      0),
                  // Insert method body scope ends.
                    '{/*method-start*/...}
300
                 // {/*method-start*/.../*method-end*/}
301
                  (new Regex(@"\{/\*method-start\*/(?<body>((?<bracket>\{)|(?<-bracket>\})|[^\{\}]*)+)|
                     \}"), "{/*method-start*/${body}/*method-end*/}", null,
                     0),
                 // Inside method bodies replace:
303
                 // GetFirst(
304
                 // this->GetFirst(
305
                 //(new Regex(0"(?<separator>(\(|, |([\\\]) |return ))(?<!(->|\*
                      ))(?<method>(?!sizeof)[a-zA-Z0-9]+)\((?!\)\()"),
                      "${separator}this->${method}(", null,
                  (\texttt{new Regex}(@"(?<scope>/\*method-start/*/)(?<before>((?<!//*method-end/*/)(.|\n))*?)(_|)
                      <separator>[\\\](?<!(::|\.|->)))(?<method>(?!sizeof)[a-zA-Z0-9]+)\((?!\)
                      \{\) (?\langle \text{after}\rangle(.|\n)*?) (?\langle \text{scopeEnd}\rangle/\text{method-end}\rangle) \}
                      "${scope}${before}${separator}this->${method}(${after}${scopeEnd}", null, 100),
                 // Remove scope borders.
308
                 // /*method-start*/
309
310
                 (new Regex(0"/\*method-(start|end)\*/"), "", null, 0),
                 // throw new ArgumentNullException(argumentName, message);
312
                 // throw std::invalid_argument(((std::string)"Argument
313
                     ").append(argumentName).append(" is null: ").append(message).append("."));
                  (new Regex(@"throw new
314
                      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);
315
                 // throw std::invalid_argument(((std::string)"Invalid
316
                     ").append(argumentName).append(" argument: ").append(message).append("."))
                  (new Regex(@"throw new ArgumentException\(((?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*),
317
                      (?\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.");
319
                 (new Regex(@"throw new NotSupportedException\(\);"), "throw std::logic_error(\"Not
320
                     supported exception.\");", null, 0),
                 // throw new NotImplementedException();
                 // throw std::logic_error("Not implemented exception.");
322
                  (new Regex(@"throw new NotImplementedException\(\);"), "throw std::logic_error(\"Not
323
                      implemented exception.\");", null, 0),
             }.Cast<ISubstitutionRule>().ToList();
325
             public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
327
                 // ICounter<int, int> c1;
328
```

```
// ICounter<int, int>* c1;
329
                              (new Regex(0"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^>\r\n]+>)?)
                                      (?<variable>[_a-zA-Z0-9]+);"), "${abstractType}* ${variable};", null, 0),
                              // (expression)
331
                              // expression
332
                              (\text{new Regex}(@"((| )([a-zA-Z0-9_*:]+)))(, | ; | , ))"), "$1$2$3", null, 0),
333
                              // (method(expression))
                              // method(expression)
335
                              (new Regex(0"(?<firstSeparator>(\( | )))
336
                                     ))\((?<method>[a-zA-Z0-9_\->\*:]+)\((?<expression>((?<parenthesis>\()|(?<-parent_|
                                    hesis > )) | [a-zA-Z0-9_\-> *:]*)+) (?(parenthesis)(?!)) \) (?(lastSeparator>(, | Parenthesis)(?!)) | (a-zA-Z0-9_\-> *:]*)+) (?(parenthesis)(?!)) | (a-zA-Z0-9_\-> *:]*)+) (a-zA-Z0-9_\-> *:]*)+)+) (a-zA-Z0-9_\-> *:]*)+)+)+)+(a-zA-Z0-9_\-> *:]*)+)+(a-zA-Z0-9_\-> *:]*)+)+(a-zA-Z0-9_\-> *:]*)+(a-zA-Z0-9_\-> *:]*)+(
                              \rightarrow |;|\)))"), "${firstSeparator}${method}(${expression})${lastSeparator}", null, 0), // return ref _elements[node];
                              // return &_elements[node];
338
                              (new Regex(@"return ref ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9\*]+)\];"), "return &$1[$2];",
339
                              \rightarrow null, 0),
                              // null
340
                              // NULL
                              (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*) (?<=\W)null;</pre>
342
                                    (?<after>\W)"), "${before}NULL${after}", null,
                                     10)
                              // default
                              // 0
344
                              (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)defa_</pre>
345
                                    ult(?<after>\W)"), "${before}0${after}", null,
                                    10).
346
                              // #region Always
347
                              (\text{new Regex}(@"(^|\r?^n)[ \t]*\t(\text{region}|\text{endregion})[^\r^n]*(\r?^n|\$)"), "", null, 0),
                              // //#define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
349
350
                              (\text{new Regex}(@")//[ t]*) + (a-zA-ZO-9] + [t]*"), "", null, 0),
                              // #if USEARRAYPOOL\r\n#endif
352
353
                              (new Regex(0"#if [a-zA-Z0-9]+\s+\#endif"), "", null, 0),
354
                              // [Fact]
356
                              (new Regex(0"(?<firstNewLine>\r?\n|\A)(?<indent>[\t
357
                                     ]+)\[[a-zA-Z0-9]+(\((?<expression>((?<parenthesis>\()|(?<-parenthesis>\))|[^{()}\r<sub>|</sub>
                                     \n]*)+)(?(parenthesis)(?!)))))?][ \t]*(\r?\n\k<indent>)?"),
                                     "${firstNewLine}${indent}", null, 5),
                              // \n ... namespace
358
                              // namespace
359
                              (\text{new Regex}(@"(\s[\r\n]{1,2})?[\r\n]+namespace"), "$1namespace", null, 0),
360
                              // \n ... class
361
                              // class
                              (new Regex(0"(\S[\r\n]{1,2})?[\r\n]+class"), "$1class", null, 0),
363
                      }.Cast<ISubstitutionRule>().ToList();
364
365
                      public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
366
                       → base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
                      public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
368
369
       }
370
          ./Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs
 1.2
       using Xunit;
       namespace Platform.RegularExpressions.Transformer.CSharpToCpp.Tests
  4
               public class CSharpToCppTransformerTests
  5
  6
                       [Fact]
                      public void HelloWorldTest()
                              const string helloWorldCode = @"using System;
 10
       class Program
 11
 12
               public static void Main(string[] args)
 13
 14
                      Console.WriteLine(""Hello, world!"");
 1.5
 16
       }";
 17
                              const string expectedResult = @"class Program
 18
 19
               public:
 20
               static void Main(const char* args[])
 21
```

Index

 $./Platform. Regular Expressions. Transformer. CSharp ToCpp. Tests/CSharp ToCpp Transformer Tests. cs, \ 7... A platform. Regular Expressions. Transformer. CSharp ToCpp/CSharp ToCpp Transformer. cs, \ 1... A platform. Regular Expressions. Transformer. CSharp ToCpp/CSharp ToCpp Transformer. cs, \ 1... A platform. Regular Expressions. Transformer. CSharp ToCpp/CSharp ToCpp Transformer. cs, \ 1... A platform. Regular Expressions. Transformer. CSharp ToCpp/CSharp ToCpp Transformer. cs, \ 1... A platform. Regular Expressions. Transformer. CSharp ToCpp/CSharp ToCpp Transformer. cs, \ 1... A platform. Regular Expressions. Transformer. CSharp ToCpp/CSharp ToCpp Transformer. cs, \ 1... A platform. Regular Expressions. Transformer. CSharp ToCpp/CSharp ToCpp Transformer. cs, \ 1... A platform. Regular Expressions. Transformer. CSharp ToCpp/CSharp ToCpp Transformer. cs, \ 1... A platform. Regular Expressions. Transformer. CSharp ToCpp/CSharp ToCpp Transformer. cs, \ 1... A platform. Regular Expressions. Regular Expressions.$