```csharp
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text.RegularExpressions;
5
6   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
7
8   namespace Platform.RegularExpressions.Transformer.CSharpToCpp
9   {
10      public class CSharpToCppTransformer : Transformer
11      {
12          public static readonly IList<ISubstitutionRule> FirstStage = new List<SubstitutionRule>
13          {
14              // // ...
15              //
16              (new Regex(@"(\r?\n)?[ \t]+//+.+"), "", null, 0),
17              // #pragma warning disable CS1591 // Missing XML comment for publicly visible type
18              //   or member
19              //
20              (new Regex(@"^\s*?\#pragma[\sa-zA-Z0-9]+$"), "", null, 0),
21              // {\n\n\n
22              // {
23              (new Regex(@"{\s+[\r\n]+"), "{" + Environment.NewLine, null, 0),
24              // Platform.Collections.Methods.Lists
25              // Platform::Collections::Methods::Lists
26              (new Regex(@"(namespace[^\r\n]+?)\.([^\r\n]+?)"), "$1::$2", null, 20),
27              // out TProduct
28              // TProduct
29              (new Regex(@"(?<before>(<|, ))(in|out) (?<typeParameter>[a-zA-Z0-9]+)(?<after>(>|,))"),
30                  "${before}${typeParameter}${after}", null, 10),
31              // 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(@"class ([a-zA-Z0-9]+)(\s+){"), "class $1$2{" + Environment.NewLine + "    public:", null, 0),
37              // class GenericCollectionMethodsBase<TElement> {
38              // template <typename TElement> class GenericCollectionMethodsBase { public:
39              (new Regex(@"class ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>([^{]+){"), "template <typename $2> class $1$3{" + Environment.NewLine + "    public:", null, 0),
40              // static void TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement> tree, TElement* root)
41              // template<typename T> static void TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement> tree, TElement* root)
42              (new Regex(@"static ([a-zA-Z0-9]+) ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>\((([^\)\r\n]+)\)"),
43                  "template <typename $3> static $1 $2($4)", null, 0),
44              // interface IFactory<out TProduct> {
45              // template <typename TProduct> class IFactory { public:
46              (new Regex(@"interface (?<interface>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9,]+)>(?<whitespace>[^{]+){"), "template <typename...> class ${interface}; template <typename ${typeParameters}> class ${interface}<${typeParameters}>${whitespace}{" + Environment.NewLine + "    public:", null, 0),
47              // template <typename TObject, TProperty, TValue>
48              // template <typename TObject, typename TProperty, TValue>
49              (new Regex(@"(?<before>template <((, )?typename [a-zA-Z0-9]+)+,)(?<typeParameter>[a-zA-Z0-9]+)(?<after>(,|>))"), "${before}typename ${typeParameter}${after}", null, 10),
50              // (this
51              // (
52              (new Regex(@"\(this "), "(", null, 0),
53              // public static readonly EnsureAlwaysExtensionRoot Always = new EnsureAlwaysExtensionRoot();
54              // inline static EnsureAlwaysExtensionRoot Always;
55              (new Regex(@"public static readonly (?<type>[a-zA-Z0-9]+) (?<name>[a-zA-Z0-9_]+) = new \k<type>\(\);"), "inline static ${type} ${name};", null, 0),
                // public static readonly string ExceptionContentsSeparator = "---";
                // inline static const char* ExceptionContentsSeparator = "---";
                (new Regex(@"public static readonly string (?<name>[a-zA-Z0-9_]+) = ""(?<string>(\""|[^""""\r\n])+)"";"), "inline static const char* ${name} = \"${string}\";", null, 0),
```

```csharp
56              // private const int MaxPath = 92;
57              // static const int MaxPath = 92;
58              (new Regex(@"private (const|static readonly) ([a-zA-Z0-9]+) ([_a-zA-Z0-9]+) =
   ↪   ([^;\r\n]+);"), "static const $2 $3 = $4;", null, 0),
59              //  ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
   ↪   TArgument : class
60              //  ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
61              (new Regex(@"(?<before> [a-zA-Z]+\(([a-zA-Z *,]+, |))(?<type>[a-zA-Z]+)(?<after>(|
   ↪   [a-zA-Z *,]+)\))[ \r\n]+where \k<type> : class"), "${before}${type}*${after}",
   ↪   null, 0),
62              // protected virtual
63              // virtual
64              (new Regex(@"protected virtual"), "virtual", null, 0),
65              // protected abstract TElement GetFirst();
66              // virtual TElement GetFirst() = 0;
67              (new Regex(@"protected abstract ([^;\r\n]+);"), "virtual $1 = 0;", null, 0),
68              // TElement GetFirst();
69              // virtual TElement GetFirst() = 0;
70              (new Regex(@"([\r\n]+[ ]+)((?!return)[a-zA-Z0-9]+ [a-zA-Z0-9]+\(([^\)\r\n]*\))(;[
   ↪   ]*[\r\n]+)"), "$1virtual $2 = 0$3", null, 1),
71              // public virtual
72              // virtual
73              (new Regex(@"public virtual"), "virtual", null, 0),
74              // protected readonly
75              //
76              (new Regex(@"protected readonly "), "", null, 0),
77              // protected readonly TreeElement[] _elements;
78              // TreeElement _elements[N];
79              (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+)([\[\]]+)
   ↪   ([_a-zA-Z0-9]+);"), "$2 $4[N];", null, 0),
80              // protected readonly TElement Zero;
81              // TElement Zero;
82              (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+) ([_a-zA-Z0-9]+);"), "$2
   ↪   $3;", null, 0),
83              // private
84              //
85              (new Regex(@"(\W)(private|protected|public|internal) "), "$1", null, 0),
86              // static void NotImplementedException(ThrowExtensionRoot root) => throw new
   ↪   NotImplementedException();
87              // static void NotImplementedException(ThrowExtensionRoot root) { return throw new
   ↪   NotImplementedException(); }
88              (new Regex(@"(^\s+)(template \<[^>\r\n]+\> )?(static )?(override )?([a-zA-Z0-9]+
   ↪   )([a-zA-Z0-9]+)\((([^\(\r\n]*)\))\s+=>\s+throw([^;\r\n]+);"), "$1$2$3$4$5$6($7) {
   ↪   throw$8; }", null, 0),
89              // SizeBalancedTree(int capacity) => a = b;
90              // SizeBalancedTree(int capacity) { a = b; }
91              (new Regex(@"(^\s+)(template \<[^>\r\n]+\> )?(static )?(override )?(void
   ↪   )?([a-zA-Z0-9]+)\((([^\(\r\n]*)\))\s+=>\s+([^;\r\n]+);"), "$1$2$3$4$5$6($7) { $8;
   ↪   }", null, 0),
92              // int SizeBalancedTree(int capacity) => a;
93              // int SizeBalancedTree(int capacity) { return a; }
94              (new Regex(@"(^\s+)(template \<[^>\r\n]+\> )?(static )?(override )?([a-zA-Z0-9]+
   ↪   )([a-zA-Z0-9]+)\((([^\(\r\n]*)\))\s+=>\s+([^;\r\n]+);"), "$1$2$3$4$5$6($7) {
   ↪   return $8; }", null, 0),
95              // () => Integer<TElement>.Zero,
96              // () { return Integer<TElement>.Zero; },
97              (new Regex(@"\(\)\s+=>\s+([^,;\r\n]+?),"), "() { return $1; },", null, 0),
98              // => Integer<TElement>.Zero;
99              // { return Integer<TElement>.Zero; }
100             (new Regex(@"\)\s+=>\s+([^;\r\n]+?);"), ") { return $1; }", null, 0),
101             // () { return avlTree.Count; }
102             // [&]()-> auto { return avlTree.Count; }
103             (new Regex(@", \(\) { return ([^;\r\n]+); }"), ", [&]()-> auto { return $1; }",
   ↪   null, 0),
104             // Count => GetSizeOrZero(Root);
105             // GetCount() { return GetSizeOrZero(Root); }
106             (new Regex(@"(\W)([A-Z][a-zA-Z]+)\s+=>\s+([^;\r\n]+);"), "$1Get$2() { return $3; }",
   ↪   null, 0),
107             // Func<TElement> treeCount
108             // std::function<TElement()> treeCount
109             (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", null,
   ↪   0),
110             // Action<TElement> free
111             // std::function<void(TElement)> free
112             (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<void($1)> $2",
   ↪   null, 0),
```

```csharp
113            // Predicate<TArgument> predicate
114            // std::function<bool(TArgument)> predicate
115            (new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<bool($1)>
    ↪   $2", null, 0),
116            // var
117            // auto
118            (new Regex(@"(\W)var(\W)"), "$1auto$2", null, 0),
119            // unchecked
120            //
121            (new Regex(@"[\r\n]{2}\s*?unchecked\s*?$"), "", null, 0),
122            // throw new InvalidOperationException
123            // throw std::runtime_error
124            (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
    ↪   std::runtime_error", null, 0),
125            // void RaiseExceptionIgnoredEvent(Exception exception)
126            // void RaiseExceptionIgnoredEvent(const std::exception& exception)
127            (new Regex(@"(\(|, )(System\.Exception|Exception)( |\))"), "$1const
    ↪   std::exception&$3", null, 0),
128            // EventHandler<Exception>
129            // EventHandler<std::exception>
130            (new Regex(@"(\W)(System\.Exception|Exception)(\W)"), "$1std::exception$3", null, 0),
131            // override void PrintNode(TElement node, StringBuilder sb, int level)
132            // void PrintNode(TElement node, StringBuilder sb, int level) override
133            (new Regex(@"override ([a-zA-Z0-9 \*\+]+)(\(([^\)\r\n]+?\)))"), "$1$2 override", null,
    ↪   0),
134            // string
135            // const char*
136            (new Regex(@"(\W)string(\W)"), "$1const char*$2", null, 0),
137            // sbyte
138            // std::int8_t
139            (new Regex(@"(\W)sbyte(\W)"), "$1std::int8_t$2", null, 0),
140            // uint
141            // std::uint32_t
142            (new Regex(@"(\W)uint(\W)"), "$1std::uint32_t$2", null, 0),
143            // char*[] args
144            // char* args[]
145            (new Regex(@"([_a-zA-Z0-9:\*]?)\[\] ([a-zA-Z0-9]+)"), "$1 $2[]", null, 0),
146            // @object
147            // object
148            (new Regex(@"@([_a-zA-Z0-9]+)"), "$1", null, 0),
149            // using Platform.Numbers;
150            //
151            (new Regex(@"([\r\n]{2}|^)\s*?using [\.a-zA-Z0-9]+;\s*?$"), "", null, 0),
152            // struct TreeElement { }
153            // struct TreeElement { };
154            (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([^;])"), "$1
    ↪   $2$3{$4};$5", null, 0),
155            // class Program { }
156            // class Program { };
157            (new Regex(@"(struct|class) ([a-zA-Z0-9]+[^\r\n]*)([\r\n]+(?<indentLevel>[\t
    ↪   ]*)?)\{([\S\s]+?[\r\n]+\k<indentLevel>)\}([^;]|$)"), "$1 $2$3{$4};$5", null, 0),
158            // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
159            // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
160            (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)"), "class $1 : public $2", null,
    ↪   0),
161            // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
162            // class IProperty : public ISetter<TValue, TObject>, IProvider<TValue, TObject>
163            (new Regex(@"(?<before>class [a-zA-Z0-9]+ : ((public [a-zA-Z0-9]+(<[a-zA-Z0-9
    ↪   ,]+>)?, )+)?)(?<inheritedType>(?!public)[a-zA-Z0-9]+(<[a-zA-Z0-9
    ↪   ,]+>)?)(?<after>(, [a-zA-Z0-9]+(?!>)|[ \r\n]+))"), "${before}public
    ↪   ${inheritedType}${after}", null, 10),
164            // Insert scope borders.
165            // ref TElement root
166            // ~!root!~ref TElement root
167            (new Regex(@"(?<definition>(?<= |\()(ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!ref))
    ↪   (?<variable>[a-zA-Z0-9]+)(?=\)|, | =))"), "~!${variable}!~${definition}", null,
    ↪   0),
168            // Inside the scope of ~!root!~ replace:
169            // root
170            // *root
171            (new Regex(@"(?<definition>~!(?<pointer>[a-zA-Z0-9]+)!~ref [a-zA-Z0-9]+
    ↪   \k<pointer>(?=\)|, | =))(?<before>((?<!~!\k<pointer>!~)(.|\n))*?)(?<prefix>(\W
    ↪   |\())\k<pointer>(?<suffix>( |\))|;|,))"),
    ↪   "${definition}${before}${prefix}*${pointer}${suffix}", null, 70),
172            // Remove scope borders.
173            // ~!root!~
```

```csharp
174                 //
175                 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
176                 // ref auto root = ref
177                 // ref auto root =
178                 (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)"), "$1* $2 =$3", null, 0),
179                 // *root = ref left;
180                 // root = left;
181                 (new Regex(@"\*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", null, 0),
182                 // (ref left)
183                 // (left)
184                 (new Regex(@"\(ref ([a-zA-Z0-9]+)(\)|\(|,)"), "($1$2", null, 0),
185                 //  ref TElement
186                 //  TElement*
187                 (new Regex(@"( |\()ref ([a-zA-Z0-9]+) "), "$1$2* ", null, 0),
188                 // ref sizeBalancedTree.Root
189                 // &sizeBalancedTree->Root
190                 (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)"), "&$1->$2", null, 0),
191                 // ref GetElement(node).Right
192                 // &GetElement(node)->Right
193                 (new Regex(@"ref ([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"),
    ↪            "&$1($2)->$3", null, 0),
194                 // GetElement(node).Right
195                 // GetElement(node)->Right
196                 (new Regex(@"([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"), "$1($2)->$3",
    ↪            null, 0),
197                 // [Fact]\npublic static void SizeBalancedTreeMultipleAttachAndDetachTest()
198                 // TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
199                 (new Regex(@"\[Fact\][\s\n]+(static )?void ([a-zA-Z0-9]+)\(\)"), "TEST_METHOD($2)",
    ↪            null, 0),
200                 // class TreesTests
201                 // TEST_CLASS(TreesTests)
202                 (new Regex(@"class ([a-zA-Z0-9]+)Tests"), "TEST_CLASS($1)", null, 0),
203                 // Assert.Equal
204                 // Assert::AreEqual
205                 (new Regex(@"Assert\.Equal"), "Assert::AreEqual", null, 0),
206                 // $"Argument {argumentName} is null."
207                 // ((std::string)"Argument ").append(argumentName).append(" is null.").data()
208                 (new Regex(@"\$""(?<left>(\\""|[^""\r\n])*){(?<expression>[_a-zA-Z0-9]+)}(?<right>(\↲
    ↪            \""|[^""\r\n])*)"""),
    ↪            "((std::string)$\"${left}\").append($${expression}).append(\"${right}\").data()",
    ↪            null, 10),
209                 // $"
210                 // "
211                 (new Regex(@"\$"""), "\"", null, 0),
212                 // Console.WriteLine("...")
213                 // printf("...\n")
214                 (new Regex(@"Console\.WriteLine\(""([^""\r\n]+)""\)"), "printf(\"$1\\n\")", null, 0),
215                 // TElement Root;
216                 // TElement Root = 0;
217                 (new Regex(@"(\r?\n[\t ]+)([a-zA-Z0-9:_]+(?<!return)) ([_a-zA-Z0-9]+);"), "$1$2 $3 =
    ↪            0;", null, 0),
218                 // TreeElement _elements[N];
219                 // TreeElement _elements[N] = { {0} };
220                 (new Regex(@"(\r?\n[\t ]+)([a-zA-Z0-9]+) ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9]+)\];"),
    ↪            "$1$2 $3[$4] = { {0} };", null, 0),
221                 // auto path = new TElement[MaxPath];
222                 // TElement path[MaxPath] = { {0} };
223                 (new Regex(@"(\r?\n[\t ]+)[a-zA-Z0-9]+ ([a-zA-Z0-9]+) = new
    ↪            ([a-zA-Z0-9]+)\[([_a-zA-Z0-9]+)\];"), "$1$3 $2[$4] = { {0} };", null, 0),
224                 // Insert scope borders.
225                 // auto added = new StringBuilder();
226                 // /*~sb~*/std::string added;
227                 (new Regex(@"(auto|(System\.Text\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new
    ↪            (System\.Text\.)?StringBuilder\(\);"), "/*~${variable}~*/std::string
    ↪            ${variable};", null, 0),
228                 // static void Indent(StringBuilder sb, int level)
229                 // static void Indent(/*~sb~*/StringBuilder sb, int level)
230                 (new Regex(@"(?<start>, |\()(System\.Text\.)?StringBuilder
    ↪            (?<variable>[a-zA-Z0-9]+)(?<end>,|\))"), "${start}/*~${variable}~*/std::string&
    ↪            ${variable}${end}", null, 0),
231                 // Inside the scope of ~!added!~ replace:
232                 // sb.ToString()
233                 // sb.data()
234                 (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>↲
    ↪            ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.ToString\(\)"),
    ↪            "${scope}${separator}${before}${variable}.data()", null, 10),
```

```
235    // sb.AppendLine(argument)
236    // sb.append(argument).append('\n')
237    (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>
    ↪    ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.AppendLine\((?<argument>[^\),\
    ↪    r\n]+)\))"),
    ↪    "${scope}${separator}${before}${variable}.append(${argument}).append('\\n')",
    ↪    null, 10),
238    // sb.Append('\t', level);
239    // sb.append(level, '\t');
240    (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),
241    // sb.Append(argument)
242    // sb.append(argument)
243    (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>
    ↪    ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.Append\((?<argument>[^\),\r\n]
    ↪    +)\))"), "${scope}${separator}${before}${variable}.append(${argument})", null,
    ↪    10),
244    // Remove scope borders.
245    // /*~sb~*/
246    //
247    (new Regex(@"/\*~(?<pointer>[a-zA-Z0-9]+)~\*/"), "", null, 0),
248    // Insert scope borders.
249    // auto added = new HashSet<TElement>();
250    // ~!added!~std::unordered_set<TElement> added;
251    (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
    ↪    HashSet<(?<element>[a-zA-Z0-9]+)>\(\);"),
    ↪    "~!${variable}!~std::unordered_set<${element}> ${variable};", null, 0),
252    // Inside the scope of ~!added!~ replace:
253    // added.Add(node)
254    // added.insert(node)
255    (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?<
    ↪    !~!\k<variable>!~)(.|\n))*?)\k<variable>\.Add\((?<argument>[a-zA-Z0-9]+)\))"),
    ↪    "${scope}${separator}${before}${variable}.insert(${argument})", null, 10),
256    // Inside the scope of ~!added!~ replace:
257    // added.Remove(node)
258    // added.erase(node)
259    (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?<
    ↪    !~!\k<variable>!~)(.|\n))*?)\k<variable>\.Remove\((?<argument>[a-zA-Z0-9]+)\))"),
    ↪    "${scope}${separator}${before}${variable}.erase(${argument})", null, 10),
260    // if (added.insert(node)) {
261    // if (!added.contains(node)) { added.insert(node);
262    (new Regex(@"if \((?<variable>[a-zA-Z0-9]+)\.insert\((?<argument>[a-zA-Z0-9]+)\)\)(?
    ↪    <separator>[\t ]*[\r\n]+)(?<indent>[\t ]*){"), "if
    ↪    (!${variable}.contains(${argument}))${separator}${indent}{" +
    ↪    Environment.NewLine + "${indent}    ${variable}.insert(${argument});", null, 0),
263    // Remove scope borders.
264    // ~!added!~
265    //
266    (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
267    // Insert scope borders.
268    // auto random = new System.Random(0);
269    // std::srand(0);
270    (new Regex(@"[a-zA-Z0-9\.]+ ([a-zA-Z0-9]+) = new
    ↪    (System\.)?Random\(([a-zA-Z0-9]+)\);"), "~!$1!~std::srand($3);", null, 0),
271    // Inside the scope of ~!random!~ replace:
272    // random.Next(1, N)
273    // (std::rand() % N) + 1
274    (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.|\n)(?<before>((?<
    ↪    !~!\k<variable>!~)(.|\n))*?)\k<variable>\.Next\((?<from>[a-zA-Z0-9]+),
    ↪    (?<to>[a-zA-Z0-9]+)\))"), "${scope}${separator}${before}(std::rand() % ${to}) +
    ↪    ${from}", null, 10),
275    // Remove scope borders.
276    // ~!random!~
277    //
278    (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
279    // Insert method body scope starts.
280    // void PrintNodes(TElement node, StringBuilder sb, int level) {
281    // void PrintNodes(TElement node, StringBuilder sb, int level) {/*method-start*/
282    (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),
```

```
283            // Insert method body scope ends.
284            // {/*method-start*/...}
285            // {/*method-start*/.../*method-end*/}
286            (new Regex(@"\{/\*method-start\*/(?<body>((?<bracket>\{)|(?<-bracket>\})|[^\{\}]*)+)↵
       ↪  \}"), "{/*method-start*/${body}/*method-end*/}", null,
       ↪  0),
287            // Inside method bodies replace:
288            // GetFirst(
289            // this->GetFirst(
290            //(new Regex(@"(?<separator>(\(|, |([\W]) |return ))(?<!(->|\*↵
       ↪  ))(?<method>(?!sizeof)[a-zA-Z0-9]+)\((?!\) \{)"),
       ↪  "${separator}this->${method}(", null, 1),
291            (new Regex(@"(?<scope>/\*method-start\*/)(?<before>((?<!/\*method-end\*/)(.|\n))*?)(↵
       ↪  ?<separator>[\W](?<!(::|\.|->)))(?<method>(?!sizeof)[a-zA-Z0-9]+)\((?!\)↵
       ↪  \{)(?<after>(.|\n)*?)(?<scopeEnd>/\*method-end\*/)"),
       ↪  "${scope}${before}${separator}this->${method}(${after}${scopeEnd}", null, 100),
292            // Remove scope borders.
293            // /*method-start*/
294            //
295            (new Regex(@"/\*method-(start|end)\*/"), "", null, 0),
296            // throw new ArgumentNullException(argumentName, message);
297            // throw std::invalid_argument(((std::string)"Argument
       ↪  ").append(argumentName).append(" is null: ").append(message).append("."));
298            (new Regex(@"throw new
       ↪  ArgumentNullException\((?<argument>[a-zA-Z]*[Aa]rgument[a-zA-Z]*),
       ↪  (?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*)\);"), "throw
       ↪  std::invalid_argument(((std::string)\"Argument \").append(${argument}).append(\"
       ↪  is null: \").append(${message}).append(\".\"));", null, 0),
299            // throw new ArgumentException(message, argumentName);
300            // throw std::invalid_argument(((std::string)"Invalid
       ↪  ").append(argumentName).append(" argument: ").append(message).append("."));
301            (new Regex(@"throw new ArgumentException\((?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*),
       ↪  (?<argument>[a-zA-Z]*[Aa]rgument[a-zA-Z]*)\);"), "throw
       ↪  std::invalid_argument(((std::string)\"Invalid \").append(${argument}).append(\"
       ↪  argument: \").append(${message}).append(\".\"));", null, 0),
302            // throw new NotSupportedException();
303            // throw std::logic_error("Not supported exception.");
304            (new Regex(@"throw new NotSupportedException\(\);"), "throw std::logic_error(\"Not
       ↪  supported exception.\");", null, 0),
305            // throw new NotImplementedException();
306            // throw std::logic_error("Not implemented exception.");
307            (new Regex(@"throw new NotImplementedException\(\);"), "throw std::logic_error(\"Not
       ↪  implemented exception.\");", null, 0),
308
309        }.Cast<ISubstitutionRule>().ToList();
310
311        public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
312        {
313            // ICounter<int, int> c1;
314            // ICounter<int, int>* c1;
315            (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^>\r\n]+>)?)
       ↪  (?<variable>[_a-zA-Z0-9]+);"), "${abstractType}* ${variable};", null, 0),
316            // (expression)
317            // expression
318            (new Regex(@"(\(| )\(([a-zA-Z0-9_\*:]+)\)(,| |;|\))"), "$1$2$3", null, 0),
319            // (method(expression))
320            // method(expression)
321            (new Regex(@"(?<firstSeparator>(\(|
       ↪  ))\((?<method>[a-zA-Z0-9_\->\*:]+)\((?<expression>((?<parenthesis>\()|(?<-parent↵
       ↪  hesis>\))|[a-zA-Z0-9_\->\*:]*)+)(?(parenthesis)(?!))\)\)(?<lastSeparator>(,|
       ↪  |;|\)))"), "${firstSeparator}${method}(${expression})${lastSeparator}", null, 0),
322            // return ref _elements[node];
323            // return &_elements[node];
324            (new Regex(@"return ref ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9\*]+)\];"), "return &$1[$2];",
       ↪  null, 0),
325            // null
326            // NULL
327            (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)null↵
       ↪  (?<after>\W)"), "${before}NULL${after}", null,
       ↪  10),
328            // default
329            // 0
330            (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)defa↵
       ↪  ult(?<after>\W)"), "${before}0${after}", null,
       ↪  10),
331            // #region Always
```

```
332             //
333             (new Regex(@"(^|\r?\n)[ \t]*\#(region|endregion)[^\r\n]*(\r?\n|$)"), "", null, 0),
334             // //#define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
335             //
336             (new Regex(@"\/\/[ \t]*\#define[ \t]+[_a-zA-Z0-9]+[ \t]*"), "", null, 0),
337             // #if USEARRAYPOOL\r\n#endif
338             //
339             (new Regex(@"#if [a-zA-Z0-9]+\s+#endif"), "", null, 0),
340             // [Fact]
341             //
342             (new Regex(@"(?<firstNewLine>\r?\n|\A)(?<indent>[\t
    ↪  ]+)\[[a-zA-Z0-9]+(\((?<expression>((?<parenthesis>\()|(?<-parenthesis>\))|[^()\r
    ↪  \n]*)+)(?(parenthesis)(?!))\))?\][ \t]*(\r?\n\k<indent>)?"),
    ↪  "${firstNewLine}${indent}", null, 5),
343             // \n ... namespace
344             // namespace
345             (new Regex(@"(\S[\r\n]{1,2})?[\r\n]+namespace"), "$1namespace", null, 0),
346             // \n ... class
347             // class
348             (new Regex(@"(\S[\r\n]{1,2})?[\r\n]+class"), "$1class", null, 0),
349         }.Cast<ISubstitutionRule>().ToList();
350
351         public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
    ↪  base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
352
353         public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
354     }
355 }
```

## 1.2 ./Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs

```
1  using Xunit;
2
3  namespace Platform.RegularExpressions.Transformer.CSharpToCpp.Tests
4  {
5      public class CSharpToCppTransformerTests
6      {
7          [Fact]
8          public void HelloWorldTest()
9          {
10             const string helloWorldCode = @"using System;
11 class Program
12 {
13     public static void Main(string[] args)
14     {
15         Console.WriteLine(""Hello, world!"");
16     }
17 }";
18             const string expectedResult = @"class Program
19 {
20     public:
21     static void Main(const char* args[])
22     {
23         printf(""Hello, world!\n"");
24     }
25 };";
26             var transformer = new CSharpToCppTransformer();
27             var actualResult = transformer.Transform(helloWorldCode, new Context(null));
28             Assert.Equal(expectedResult, actualResult);
29         }
30     }
31 }
```

# Index