```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::$2", null, 20),
27              // out TProduct
28              // TProduct
                (new Regex(@"(?<before>(<|, ))(in|out)
                →  (?<typeParameter>[a-zA-Z0-9]+)(?<after>(>|,))"),
                →  "${before}${typeParameter}${after}", null, 10),
29              // public ...
30              // public: ...
31              (new Regex(@"(?<newLineAndIndent>\r?\n?[
                →  \t]*)(?<before>[^\{\(\(\r\n]*)(?<access>private|protected|public)[
                →  \t]+(?![^\{\(\(\r\n]*(interface|class|struct)[^\{\(\(\r\n]*[\{\(\(\r\n])"),
                →  "${newLineAndIndent}${access}: ${before}", null, 0),
32              // public: static bool CollectExceptions { get; set; }
33              // public: static bool CollectExceptions;
34              (new Regex(@"(?<before>(private|protected|public): (static )?[^\r\n]+
                →  )(?<name>[a-zA-Z0-9]+) {[^;}]*?(?<=\W)get;[^;}]*?(?<=\W)set;[^;}]*}"),
                →  "${before}${name};", null, 0),
35              // public abstract class
36              // class
37              (new Regex(@"((public|protected|private|internal|abstract|static)
                →  )*(?<category>interface|class|struct)"), "${category}", null, 0),
38              // class GenericCollectionMethodsBase<TElement> {
39              // template <typename TElement> class GenericCollectionMethodsBase {
40              (new Regex(@"class ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>([^{]+){"), "template <typename $2>
                →  class $1$3{", null, 0),
41              // static void
                →  TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
                →  tree, TElement* root)
42              // template<typename T> static void
                →  TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
                →  tree, TElement* root)
43              (new Regex(@"static ([a-zA-Z0-9]+) ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>\((([^\)\r\n]+)\)"),
                →  "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              // Insert markers
51              // private: static void BuildExceptionString(this StringBuilder sb, Exception
                →  exception, int level)
52              // /*~extensionMethod~BuildExceptionString~*/private: static void
                →  BuildExceptionString(this StringBuilder sb, Exception exception, int level)
```

```
53      (new Regex(@"private: static [^\r\n]+ (?<name>[a-zA-Z0-9]+)\(this [^\)\r\n]+\)"),
        "/*~extensionMethod~${name}~*/$0", null, 0),
54      // Move all markers to the beginning of the file.
55      (new Regex(@"\A(?<before>[^\r\n]+\r?\n(.|\n)+)(?<marker>/\*~extensionMethod~(?<name>
        [a-zA-Z0-9]+)~\*/)"), "${marker}${before}", null,
        10),
56      // /*~extensionMethod~BuildExceptionString~*/...sb.BuildExceptionString(exception.In
        nerException, level +
        1);
57      // /*~extensionMethod~BuildExceptionString~*/...BuildExceptionString(sb,
        exception.InnerException, level + 1);
58      (new Regex(@"(?<before>/\*~extensionMethod~(?<name>[a-zA-Z0-9]+)~\*/(.|\n)+\W)(?<var
        iable>[_a-zA-Z0-9]+)\.\k<name>\("), "${before}${name}(${variable}, ", null,
        50),
59      // Remove markers
60      // /*~extensionMethod~BuildExceptionString~*/
61      //
62      (new Regex(@"/\*~extensionMethod~[a-zA-Z0-9]+~\*/"), "", null, 0),
63      // (this
64      // (
65      (new Regex(@"\(this "), "(", null, 0),
66      // public: static readonly EnsureAlwaysExtensionRoot Always = new
        EnsureAlwaysExtensionRoot();
67      // public:inline static EnsureAlwaysExtensionRoot Always;
68      (new Regex(@"(?<access>(private|protected|public): )?static readonly
        (?<type>[a-zA-Z0-9]+) (?<name>[a-zA-Z0-9_]+) = new \k<type>\(\);"),
        "${access}inline static ${type} ${name};", null, 0),
69      // public: static readonly string ExceptionContentsSeparator = "---";
70      // public: inline static const char* ExceptionContentsSeparator = "---";
71      (new Regex(@"(?<access>(private|protected|public): )?static readonly string
        (?<name>[a-zA-Z0-9_]+) = ""(?<string>(\""|[^""\r\n])+)"";"), "${access}inline
        static const char* ${name} = \"${string}\";", null, 0),
72      // private: const int MaxPath = 92;
73      // private: static const int MaxPath = 92;
74      (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly)
        (?<type>[a-zA-Z0-9]+) (?<name>[_a-zA-Z0-9]+) = (?<value>[^;\r\n]+);"),
        "${access}static const ${type} ${name} = ${value};", null, 0),
75      //  ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
        TArgument : class
76      //  ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
77      (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}*${after}",
        null, 0),
78      // protected: abstract TElement GetFirst();
79      // protected: virtual TElement GetFirst() = 0;
80      (new Regex(@"(?<access>(private|protected|public): )?abstract
        (?<method>[^;\r\n]+);"), "${access}virtual ${method} = 0;", null, 0),
81      // TElement GetFirst();
82      // virtual TElement GetFirst() = 0;
83      (new Regex(@"([\r\n]+[ ]+)((?!return)[a-zA-Z0-9]+ [a-zA-Z0-9]+\(([^\)\r\n]*\))(;[
        ]*[\r\n]+)"), "$1virtual $2 = 0$3", null, 1),
84      // protected: readonly TreeElement[] _elements;
85      // protected: TreeElement _elements[N];
86      (new Regex(@"(?<access>(private|protected|public): )?readonly
        (?<type>[a-zA-Z<>0-9]+)([\[\]]+) (?<name>[_a-zA-Z0-9]+);"), "${access}${type}
        ${name}[N];", null, 0),
87      // protected: readonly TElement Zero;
88      // protected: TElement Zero;
89      (new Regex(@"(?<access>(private|protected|public): )?readonly
        (?<type>[a-zA-Z<>0-9]+) (?<name>[_a-zA-Z0-9]+);"), "${access}${type} ${name};",
        null, 0),
90      // internal
91      //
92      (new Regex(@"(\W)internal\s+"), "$1", null, 0),
93      // static void NotImplementedException(ThrowExtensionRoot root) => throw new
        NotImplementedException();
94      // static void NotImplementedException(ThrowExtensionRoot root) { return throw new
        NotImplementedException(); }
95      (new Regex(@"(^\s+)(private|protected|public)?(: )?(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$8($9) { throw$10; }", null, 0),
96      // SizeBalancedTree(int capacity) => a = b;
97      // SizeBalancedTree(int capacity) { a = b; }
```

```
 98        (new Regex(@"(^\s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static
    ↪   )?(override )?(void )?([a-zA-Z0-9]+)\((([^\(\r\n]*)\)\s+=>\s+([^;\r\n]+);"),
    ↪   "$1$2$3$4$5$6$7$8($9) { $10; }", null, 0),
 99        // int SizeBalancedTree(int capacity) => a;
100        // int SizeBalancedTree(int capacity) { return a; }
101        (new Regex(@"(^\s+)(private|protected|public)?(: )?(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$8($9) {
    ↪   return $10; }", null, 0),
102        // () => Integer<TElement>.Zero,
103        // () { return Integer<TElement>.Zero; },
104        (new Regex(@"\(\)\s+=>\s+(?<expression>[^,;\r\n]+((?<parenthesis>\()|(?<-parenthesis⌋
    ↪   >\)))|[^;\r\n]*)*[^,;\r\n]*)(?<after>,|\);)"), "() { return ${expression};
    ↪   }${after}", null, 0),
105        // => Integer<TElement>.Zero;
106        // { return Integer<TElement>.Zero; }
107        (new Regex(@"\)\s+=>\s+([^;\r\n]+?);"), ") { return $1; }", null, 0),
108        // () { return avlTree.Count; }
109        // [&]()-> auto { return avlTree.Count; }
110        (new Regex(@", \(\) { return ([^;\r\n]+); }"), ", [&]()-> auto { return $1; }",
    ↪   null, 0),
111        // Count => GetSizeOrZero(Root);
112        // GetCount() { return GetSizeOrZero(Root); }
113        (new Regex(@"(\W)([A-Z][a-zA-Z]+)\s+=>\s+([^;\r\n]+);"), "$1Get$2() { return $3; }",
    ↪   null, 0),
114        // Func<TElement> treeCount
115        // std::function<TElement()> treeCount
116        (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", null,
    ↪   0),
117        // Action<TElement> free
118        // std::function<void(TElement)> free
119        (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<void($1)> $2",
    ↪   null, 0),
120        // Predicate<TArgument> predicate
121        // std::function<bool(TArgument)> predicate
122        (new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<bool($1)>
    ↪   $2", null, 0),
123        // var
124        // auto
125        (new Regex(@"(\W)var(\W)"), "$1auto$2", null, 0),
126        // unchecked
127        //
128        (new Regex(@"[\r\n]{2}\s*?unchecked\s*?$"), "", null, 0),
129        // throw new InvalidOperationException
130        // throw std::runtime_error
131        (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
    ↪   std::runtime_error", null, 0),
132        // void RaiseExceptionIgnoredEvent(Exception exception)
133        // void RaiseExceptionIgnoredEvent(const std::exception& exception)
134        (new Regex(@"(\(|, )(System\.Exception|Exception)( |\))"), "$1const
    ↪   std::exception&$3", null, 0),
135        // EventHandler<Exception>
136        // EventHandler<std::exception>
137        (new Regex(@"(\W)(System\.Exception|Exception)(\W)"), "$1std::exception$3", null, 0),
138        // override void PrintNode(TElement node, StringBuilder sb, int level)
139        // void PrintNode(TElement node, StringBuilder sb, int level) override
140        (new Regex(@"override ([a-zA-Z0-9 \*\+]+)(\(([^\)\r\n]+?\)))"), "$1$2 override", null,
    ↪   0),
141        // string
142        // const char*
143        (new Regex(@"(\W)string(\W)"), "$1const char*$2", null, 0),
144        // sbyte
145        // std::int8_t
146        (new Regex(@"(\W)sbyte(\W)"), "$1std::int8_t$2", null, 0),
147        // uint
148        // std::uint32_t
149        (new Regex(@"(\W)uint(\W)"), "$1std::uint32_t$2", null, 0),
150        // char*[] args
151        // char* args[]
152        (new Regex(@"([_a-zA-Z0-9:\*]?)\[\] ([a-zA-Z0-9]+)"), "$1 $2[]", null, 0),
153        // @object
154        // object
155        (new Regex(@"@([_a-zA-Z0-9]+)"), "$1", null, 0),
156        // using Platform.Numbers;
157        //
158        (new Regex(@"([\r\n]{2}|^)\s*?using [\.a-zA-Z0-9]+;\s*?$"), "", null, 0),
159        // struct TreeElement { }
```

```csharp
                    // struct TreeElement { };
                    (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([^;])"), "$1
                    ↪  $2$3{$4};$5", null, 0),
                    // class Program { }
                    // class Program { };
                    (new Regex(@"(struct|class) ([a-zA-Z0-9]+[^\r\n]*)([\r\n]+(?<indentLevel>[\t
                    ↪  ]*)?)\{([\S\s]+?[\r\n]+\k<indentLevel>)\}([^;]|$)"), "$1 $2$3{$4};$5", null, 0),
                    // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
                    // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
                    (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)"), "class $1 : public $2", null,
                    ↪  0),
                    // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
                    // class IProperty : public ISetter<TValue, TObject>, IProvider<TValue, TObject>
                    (new Regex(@"(?<before>class [a-zA-Z0-9]+ : ((public [a-zA-Z0-9]+(<[a-zA-Z0-9
                    ↪  ,]+>)?, )+)?)(?<inheritedType>(?!public)[a-zA-Z0-9]+(<[a-zA-Z0-9
                    ↪  ,]+>)?)(?<after>(, [a-zA-Z0-9]+(?!>)|[ \r\n]+))"), "${before}public
                    ↪  ${inheritedType}${after}", null, 10),
                    // Insert scope borders.
                    // ref TElement root
                    // ~!root!~ref TElement root
                    (new Regex(@"(?<definition>(?<= |\()(ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!ref))
                    ↪  (?<variable>[a-zA-Z0-9]+)(?=\)|, | =))"), "~!${variable}!~${definition}", null,
                    ↪  0),
                    // Inside the scope of ~!root!~ replace:
                    // root
                    // *root
                    (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),
                    // Remove scope borders.
                    // ~!root!~
                    //
                    (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
                    // ref auto root = ref
                    // ref auto root =
                    (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)"), "$1* $2 =$3", null, 0),
                    // *root = ref left;
                    // root = left;
                    (new Regex(@"\*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", null, 0),
                    // (ref left)
                    // (left)
                    (new Regex(@"\(ref ([a-zA-Z0-9]+)(\))|\((|,)"), "($1$2", null, 0),
                    //  ref TElement
                    //  TElement*
                    (new Regex(@"( |\()ref ([a-zA-Z0-9]+) "), "$1$2* ", null, 0),
                    // ref sizeBalancedTree.Root
                    // &sizeBalancedTree->Root
                    (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)"), "&$1->$2", null, 0),
                    // ref GetElement(node).Right
                    // &GetElement(node)->Right
                    (new Regex(@"ref ([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"),
                    ↪  "&$1($2)->$3", null, 0),
                    // GetElement(node).Right
                    // GetElement(node)->Right
                    (new Regex(@"([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"), "$1($2)->$3",
                    ↪  null, 0),
                    // [Fact]\npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
                    // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
                    (new Regex(@"\[Fact\][\s\n]+(public: )?(static )?void ([a-zA-Z0-9]+)\(\)"), "public:
                    ↪  TEST_METHOD($3)", null, 0),
                    // class TreesTests
                    // TEST_CLASS(TreesTests)
                    (new Regex(@"class ([a-zA-Z0-9]+)Tests"), "TEST_CLASS($1)", null, 0),
                    // Assert.Equal
                    // Assert::AreEqual
                    (new Regex(@"Assert\.Equal"), "Assert::AreEqual", null, 0),
                    // $"Argument {argumentName} is null."
                    // ((std::string)"Argument ").append(argumentName).append(" is null.").data()
                    (new Regex(@"\$""(?<left>(\\""|[^""\r\n])*){(?<expression>[_a-zA-Z0-9]+)}(?<right>(\
                    ↪  \""|[^""\r\n])*)"""),
                    ↪  "((std::string)$\"${left}\").append(${expression}).append(\"${right}\").data()",
                    ↪  null, 10),
                    // $"
                    // "
                    (new Regex(@"\$"""), "\"", null, 0),
```

```csharp
219            // Console.WriteLine("...")
220            // printf("...\n")
221            (new Regex(@"Console\.WriteLine\(""([^""\r\n]+)""\)"), "printf(\"$1\\n\")", null, 0),
222            // TElement Root;
223            // TElement Root = 0;
224            (new Regex(@"(\r?\n[\t ]+)(private|protected|public)?(:
    ↪ )?([a-zA-Z0-9:_]+(?<!return)) ([_a-zA-Z0-9]+);"), "$1$2$3$4 $5 = 0;", null, 0),
225            // TreeElement _elements[N];
226            // TreeElement _elements[N] = { {0} };
227            (new Regex(@"(\r?\n[\t ]+)(private|protected|public)?(: )?([a-zA-Z0-9]+)
    ↪ ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9]+)\];"), "$1$2$3$4 $5[$6] = { {0} };", null, 0),
228            // auto path = new TElement[MaxPath];
229            // TElement path[MaxPath] = { {0} };
230            (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),
231            // private: static readonly ConcurrentBag<std::exception> _exceptionsBag = new
    ↪ ConcurrentBag<std::exception>();
232            // private: static std::mutex _exceptionsBag_mutex; \n\n private: static
    ↪ std::vector<std::exception> _exceptionsBag;
233            (new Regex(@"(?<begin>\r?\n?(?<indent>[ \t]+))(?<access>(private|protected|public):
    ↪ )?static readonly ConcurrentBag<(?<argumentType>[^;\r\n]+)>
    ↪ (?<name>[_a-zA-Z0-9]+) = new ConcurrentBag<\k<argumentType>>\(\);"),
    ↪ "${begin}private: static std::mutex ${name}_mutex;" + Environment.NewLine +
    ↪ Environment.NewLine + "${indent}${access}static std::vector<${argumentType}>
    ↪ ${name};", null, 0),
234            // public: static IReadOnlyCollection<std::exception> GetCollectedExceptions() {
    ↪ return _exceptionsBag; }
235            // public: static std::vector<std::exception> GetCollectedExceptions() { return
    ↪ std::vector<std::exception>(_exceptionsBag); }
236            (new Regex(@"(?<access>(private|protected|public): )?static
    ↪ IReadOnlyCollection<(?<argumentType>[^;\r\n]+)> (?<methodName>[_a-zA-Z0-9]+)\(\)
    ↪ { return (?<fieldName>[_a-zA-Z0-9]+); }"), "${access}static
    ↪ std::vector<${argumentType}> ${methodName}() { return
    ↪ std::vector<${argumentType}>(${fieldName}); }", null, 0),
237            // public: static event EventHandler<std::exception> ExceptionIgnored =
    ↪ OnExceptionIgnored; ... };
238            // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
    ↪ const std::exception&)> ExceptionIgnored = OnExceptionIgnored; };
239            (new Regex(@"(?<begin>\r?\n(\r?\n)?(?<halfIndent>[
    ↪ \t]+)\k<halfIndent>)(?<access>(private|protected|public): )?static event
    ↪ EventHandler<(?<argumentType>[^;\r\n]+)> (?<name>[_a-zA-Z0-9]+) = (?<defaultDele⌋
    ↪ gate>[_a-zA-Z0-9]+);(?<middle>(.|\n)+?)(?<end>\r?\n\k<halfIndent>};)"),
    ↪ "${middle}" + Environment.NewLine + Environment.NewLine +
    ↪ "${halfIndent}${halfIndent}${access}static inline
    ↪ Platform::Delegates::MulticastDelegate<void(void*, const ${argumentType}&)>
    ↪ ${name} = ${defaultDelegate};${end}", null, 0),
240            // Insert scope borders.
241            // class IgnoredExceptions { ... private: static std::vector<std::exception>
    ↪ _exceptionsBag;
242            // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: static
    ↪ std::vector<std::exception> _exceptionsBag;
243            (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n]+\r\n[\t
    ↪ ]*{)(?<middle>((?!class).|\n)+?)(?<vectorFieldDeclaration>(?<access>(private|pro⌋
    ↪ tected|public): )static std::vector<(?<argumentType>[^;\r\n]+)>
    ↪ (?<fieldName>[_a-zA-Z0-9]+);)"),
    ↪ "${classDeclarationBegin}/*~${fieldName}~*/${middle}${vectorFieldDeclaration}",
    ↪ null, 0),
244            // Inside the scope of ~!_exceptionsBag!~ replace:
245            // _exceptionsBag.Add(exception);
246            // _exceptionsBag.push_back(exception);
247            (new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<befor⌋
    ↪ e>((?<!/\*~\k<fieldName>~\*/)(.|\n))*?)\k<fieldName>\.Add"),
    ↪ "${scope}${separator}${before}${fieldName}.push_back", null, 10),
248            // Remove scope borders.
249            // /*~_exceptionsBag~*/
250            //
251            (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", null, 0),
252            // Insert scope borders.
253            // class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
254            // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: static std::mutex
    ↪ _exceptionsBag_mutex;
255            (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n]+\r\n[\t
    ↪ ]*{)(?<middle>((?!class).|\n)+?)(?<mutexDeclaration>private: static std::mutex
    ↪ (?<fieldName>[_a-zA-Z0-9]+)_mutex;)"),
    ↪ "${classDeclarationBegin}/*~${fieldName}~*/${middle}${mutexDeclaration}", null,
    ↪ 0),
```

```csharp
            // Inside the scope of ~!_exceptionsBag!~ replace:
            // return std::vector<std::exception>(_exceptionsBag);
            // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
            //  std::vector<std::exception>(_exceptionsBag);
            (new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<befor
               e>((?<!/\*~\k<fieldName>~\*/)(.|\n))*?){(?<after>((?!lock_guard)[^{};\r\n])*\k<f
               ieldName>[^;}\r\n]*;)"), "${scope}${separator}${before}{
               std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", null, 10),
            // Inside the scope of ~!_exceptionsBag!~ replace:
            // _exceptionsBag.Add(exception);
            // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
            //  _exceptionsBag.Add(exception);
            (new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<befor
               e>((?<!/\*~\k<fieldName>~\*/)(.|\n))*?){(?<after>((?!lock_guard)([^{};]|\n))*?\r
               ?\n(?<indent>[ \t]*)\k<fieldName>[^;}\r\n]*;)"),
               "${scope}${separator}${before}{" + Environment.NewLine +
               "${indent}std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", null,
               10),
            // Remove scope borders.
            // /*~_exceptionsBag~*/
            //
            (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", null, 0),
            // Insert scope borders.
            // class IgnoredExceptions { ... public: static inline
            //  Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
            //  ExceptionIgnored = OnExceptionIgnored;
            // class IgnoredExceptions {/*~ExceptionIgnored~*/ ... public: static inline
            //  Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
            //  ExceptionIgnored = OnExceptionIgnored;
            (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n]+\r\n[\t
               ]*{)(?<middle>((?!class).|\n)+?)(?<eventDeclaration>(?<access>(private|protected
               |public): )static inline
               Platform::Delegates::MulticastDelegate<(?<argumentType>[^;\r\n]+)>
               (?<name>[_a-zA-Z0-9]+) = (?<defaultDelegate>[_a-zA-Z0-9]+);)"),
               "${classDeclarationBegin}/*~${name}~*/${middle}${eventDeclaration}", null, 0),
            // Inside the scope of ~!ExceptionIgnored!~ replace:
            // ExceptionIgnored.Invoke(NULL, exception);
            // ExceptionIgnored(NULL, exception);
            (new Regex(@"(?<scope>/\*~(?<eventName>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before
               >((?<!/\*~\k<eventName>~\*/)(.|\n))*?)\k<eventName>\.Invoke"),
               "${scope}${separator}${before}${eventName}", null, 10),
            // Remove scope borders.
            // /*~ExceptionIgnored~*/
            //
            (new Regex(@"/\*~[a-zA-Z0-9]+~\*/"), "", null, 0),
            // Insert scope borders.
            // auto added = new StringBuilder();
            // /*~sb~*/std::string added;
            (new Regex(@"(auto|(System\.Text\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new
               (System\.Text\.)?StringBuilder\(\);"), "/*~${variable}~*/std::string
               ${variable};", null, 0),
            // static void Indent(StringBuilder sb, int level)
            // static void Indent(/*~sb~*/StringBuilder sb, int level)
            (new Regex(@"(?<start>, |\()(System\.Text\.)?StringBuilder
               (?<variable>[a-zA-Z0-9]+)(?<end>,|\))"), "${start}/*~${variable}~*/std::string&
               ${variable}${end}", null, 0),
            // Inside the scope of ~!added!~ replace:
            // sb.ToString()
            // sb.data()
            (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>
               ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.ToString\(\)"),
               "${scope}${separator}${before}${variable}.data()", null, 10),
            // sb.AppendLine(argument)
            // sb.append(argument).append('\n')
            (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(1, '\\n')",
               null, 10),
            // sb.Append('\t', level);
            // sb.append(level, '\t');
            (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),
```

```
297              // sb.Append(argument)
298              // sb.append(argument)
299              (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),
300              // Remove scope borders.
301              // /*~sb~*/
302              //
303              (new Regex(@"/\*~[a-zA-Z0-9]+~\*/"), "", null, 0),
304              // Insert scope borders.
305              // auto added = new HashSet<TElement>();
306              // ~!added!~std::unordered_set<TElement> added;
307              (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
    ↪    HashSet<(?<element>[a-zA-Z0-9]+)>\(\);"),
    ↪    "~!${variable}!~std::unordered_set<${element}> ${variable};", null, 0),
308              // Inside the scope of ~!added!~ replace:
309              // added.Add(node)
310              // added.insert(node)
311              (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),
312              // Inside the scope of ~!added!~ replace:
313              // added.Remove(node)
314              // added.erase(node)
315              (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),
316              // if (added.insert(node)) {
317              // if (!added.contains(node)) { added.insert(node);
318              (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),
319              // Remove scope borders.
320              // ~!added!~
321              //
322              (new Regex(@"~![a-zA-Z0-9]+!~"), "", null, 5),
323              // Insert scope borders.
324              // auto random = new System.Random(0);
325              // std::srand(0);
326              (new Regex(@"[a-zA-Z0-9\.]+ ([a-zA-Z0-9]+) = new
    ↪    (System\.)?Random\(([a-zA-Z0-9]+)\);"), "~!$1!~std::srand($3);", null, 0),
327              // Inside the scope of ~!random!~ replace:
328              // random.Next(1, N)
329              // (std::rand() % N) + 1
330              (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),
331              // Remove scope borders.
332              // ~!random!~
333              //
334              (new Regex(@"~![a-zA-Z0-9]+!~"), "", null, 5),
335              // Insert method body scope starts.
336              // void PrintNodes(TElement node, StringBuilder sb, int level) {
337              // void PrintNodes(TElement node, StringBuilder sb, int level) {/*method-start*/
338              (new Regex(@"(?<start>\r?\n[\t ]+)(?<prefix>((private|protected|public): )?(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*/${end}", null,
    ↪    0),
339              // Insert method body scope ends.
340              // {/*method-start*/...}
341              // {/*method-start*/.../*method-end*/}
342              (new Regex(@"\{/\*method-start\*/(?<body>((?<bracket>\{)|(?<-bracket>\})|[^\{\}]*)+)↵
    ↪    \}"), "{/*method-start*/${body}/*method-end*/}", null,
    ↪    0),
343              // Inside method bodies replace:
344              // GetFirst(
345              // this->GetFirst(
346              //(new Regex(@"(?<separator>(\(|, |([\W]) |return ))(?<!(->|\*
    ↪    ))(?<method>(?!sizeof)[a-zA-Z0-9]+)\((?!\) \{)"),
    ↪    "${separator}this->${method}(", null, 1),
```

```csharp
347          (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),
348          // Remove scope borders.
349          // /*method-start*/
350          //
351          (new Regex(@"/\*method-(start|end)\*/"), "", null, 0),
352          // Insert scope borders.
353          // const std::exception& ex
354          // const std::exception& ex/*~ex~*/
355          (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?exception&?
     ↪   (?<variable>[_a-zA-Z0-9]+))(?<after>\W)"),
     ↪   "${before}${variableDefinition}/*~${variable}~*/${after}", null, 0),
356          // Inside the scope of ~!ex!~ replace:
357          // ex.Message
358          // ex.what()
359          (new Regex(@"(?<scope>/\*~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before
     ↪   >((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.Message"),
     ↪   "${scope}${separator}${before}${variable}.what()", null, 10),
360          // Remove scope borders.
361          // /*~ex~*/
362          //
363          (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", null, 0),
364          // throw new ArgumentNullException(argumentName, message);
365          // throw std::invalid_argument(((std::string)"Argument
     ↪   ").append(argumentName).append(" is null: ").append(message).append("."));
366          (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),
367          // throw new ArgumentException(message, argumentName);
368          // throw std::invalid_argument(((std::string)"Invalid
     ↪   ").append(argumentName).append(" argument: ").append(message).append("."));
369          (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),
370          // throw new NotSupportedException();
371          // throw std::logic_error("Not supported exception.");
372          (new Regex(@"throw new NotSupportedException\(\);"), "throw std::logic_error(\"Not
     ↪   supported exception.\");", null, 0),
373          // throw new NotImplementedException();
374          // throw std::logic_error("Not implemented exception.");
375          (new Regex(@"throw new NotImplementedException\(\);"), "throw std::logic_error(\"Not
     ↪   implemented exception.\");", null, 0),
376      }.Cast<ISubstitutionRule>().ToList();
377
378      public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
379      {
380          // ICounter<int, int> c1;
381          // ICounter<int, int>* c1;
382          (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^>\r\n]+>)?)
     ↪   (?<variable>[_a-zA-Z0-9]+);"), "${abstractType}* ${variable};", null, 0),
383          // (expression)
384          // expression
385          (new Regex(@"(\(| )\(([a-zA-Z0-9_\*:]+)\)(,| |;|\))"), "$1$2$3", null, 0),
386          // (method(expression))
387          // method(expression)
388          (new Regex(@"(?<firstSeparator>(\(|
     ↪   ))\((?<method>[a-zA-Z0-9_\->\*:]+)\((?<expression>((?<parenthesis>\()|(?<-parent
     ↪   hesis>\))|[a-zA-Z0-9_\->\*:]*)+)(?(parenthesis)(?!))\)\)(?<lastSeparator>(,|
     ↪   |;|\)))"), "${firstSeparator}${method}(${expression})${lastSeparator}", null, 0),
389          // return ref _elements[node];
390          // return &_elements[node];
391          (new Regex(@"return ref ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9\*]+)\];"), "return &$1[$2];",
     ↪   null, 0),
392          // null
393          // nullptr
394          (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)null
     ↪   (?<after>\W)"), "${before}nullptr${after}", null,
     ↪   10),
395          // default
396          // 0
```

```csharp
                    (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)defa↵
        ↪    ult(?<after>\W)"), "${before}0${after}", null,
        ↪    10),
                    // object x
                    // void *x
                    (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)([0|↵
        ↪    o]bject|System\.Object) (?<after>\w)"), "${before}void *${after}", null,
        ↪    10),
                    // #region Always
                    //
                    (new Regex(@"(^|\r?\n)[ \t]*\#(region|endregion)[^\r\n]*(\r?\n|$)"), "", null, 0),
                    // //#define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
                    //
                    (new Regex(@"\/\/[ \t]*\#define[ \t]+[_a-zA-Z0-9]+[ \t]*"), "", null, 0),
                    // #if USEARRAYPOOL\r\n#endif
                    //
                    (new Regex(@"#if [a-zA-Z0-9]+\s+#endif"), "", null, 0),
                    // [Fact]
                    //
                    (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),
                    // \n ... namespace
                    // namespace
                    (new Regex(@"(\S[\r\n]{1,2})?[\r\n]+namespace"), "$1namespace", null, 0),
                    // \n ... class
                    // class
                    (new Regex(@"(\S[\r\n]{1,2})?[\r\n]+class"), "$1class", null, 0),
            }.Cast<ISubstitutionRule>().ToList();

            public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
        ↪    base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }

            public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
        }
    }
```

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

```csharp
using Xunit;

namespace Platform.RegularExpressions.Transformer.CSharpToCpp.Tests
{
    public class CSharpToCppTransformerTests
    {
        [Fact]
        public void EmptyLineTest()
        {
            // This test can help to test basic problems with regular expressions like incorrect
        ↪    syntax
            var transformer = new CSharpToCppTransformer();
            var actualResult = transformer.Transform("", new Context(null));
            Assert.Equal("", actualResult);
        }

        [Fact]
        public void HelloWorldTest()
        {
            const string helloWorldCode = @"using System;
class Program
{
    public static void Main(string[] args)
    {
        Console.WriteLine(""Hello, world!"");
    }
}";
            const string expectedResult = @"class Program
{
    public: static void Main(const char* args[])
    {
        printf(""Hello, world!\n"");
    }
};";
            var transformer = new CSharpToCppTransformer();
            var actualResult = transformer.Transform(helloWorldCode, new Context(null));
            Assert.Equal(expectedResult, actualResult);
        }
    }
```

```
39     }
```

Index