

## 1.1 ./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs

```

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::$2", null, 20),
27             // out TProduct
28             // TProduct
29             (new Regex(@"(?<before>(<|, ))(in|out)
30             → (?<typeParameter>[a-zA-Z0-9]+)(?<after>(>|,))"),
31             → "${before}${typeParameter}${after}", null, 10),
32             // public ...
33             // public: ...
34             (new Regex(@"(?<newLineAndIndent>\r?\n?[
35             → \t]*) (?<before>[^\{\\(\r\n)]*(?<access>private|protected|public) [
36             → \t]+(?![^\{\\(\r\n)]*(interface|class|struct) [^\{\\(\r\n)]*[\{\\(\r\n)]")],
37             → "${newLineAndIndent}${access}: ${before}", null, 0),
38             // public: static bool CollectExceptions { get; set; }
39             // public: inline static bool CollectExceptions;
40             (new Regex(@"(?<access>(private|protected|public): )(?<before>(static )?[^\r\n]+
41             → )(?<name>[a-zA-Z0-9]+) {[^;]}*(?<=\\W)get; [^;]}*(?<=\\W)set; [^;]}*"),
42             → "${access}inline ${before}${name};", null, 0),
43             // public abstract class
44             // class
45             (new Regex(@"((public|protected|private|internal|abstract|static)
46             → )*(?<category>interface|class|struct)", "${category}", null, 0),
47             // class GenericCollectionMethodsBase<TElement> {
48             // template <typename TElement> class GenericCollectionMethodsBase {
49             (new Regex(@"class ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>([^\{]+){", "template <typename $2>
50             → class $1$3{", null, 0),
51             // static void
52             → TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
53             → tree, TElement* root)
54             // template<typename T> static void
55             → TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
56             → tree, TElement* root)
57             (new Regex(@"static ([a-zA-Z0-9]+) ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>\\(((^\\)\\r\\n)+)\\)",
58             → "template <typename $3> static $1 $2($4)", null, 0),
59             // interface IFactory<out TProduct> {
60             // template <typename TProduct> class IFactory { public:
61             (new Regex(@"interface (?<interface>[a-zA-Z0-9]+)<(?!<typeParameters>[a-zA-Z0-9
62             → ,]+)>(?<whitespace>[^\{]+){", "template <typename...> class ${interface};
63             → template <typename ${typeParameters}> class
64             → ${interface}<${typeParameters}>${whitespace}{ + Environment.NewLine + "
65             → public:", null, 0),
66             // template <typename TObject, TProperty, TValue>
67             // template <typename TObject, typename TProperty, TValue>
68             (new Regex(@"(?<before>template <((, )?typename [a-zA-Z0-9]+)+,
69             → )(?<typeParameter>[a-zA-Z0-9]+)(?<after>(>|,))"), "${before}typename
70             → ${typeParameter}${after}", null, 10),
71             // Insert markers
72             // private: static void BuildExceptionString(this StringBuilder sb, Exception
73             → exception, int level)
74             // /*~extensionMethod~BuildExceptionString~*/private: static void
75             → BuildExceptionString(this StringBuilder sb, Exception exception, int level)

```

```

53 (new Regex(@"private: static [^\r\n]+ (?<name>[a-zA-Z0-9]+)\(this [^\]\r\n]+\)"),
54     ↳ "/*~extensionMethod~${name}~*/$0", null, 0),
55 // Move all markers to the beginning of the file.
56 (new Regex(@"\A(?<before>[^\r\n]+\r?\n(.|\n)+)(?<marker>/\*~extensionMethod~(?<name>[a-zA-Z0-9]+)~\*/)"), "${marker}${before}", null,
57     ↳ 10),
58 // /*~extensionMethod~BuildExceptionString~*/...sb.BuildExceptionString(exception.InnerException, level +
59     ↳ 1);
60 // /*~extensionMethod~BuildExceptionString~*/...BuildExceptionString(sb,
61     ↳ exception.InnerException, level + 1);
62 (new Regex(@"(?<before>/\*~extensionMethod~(?<name>[a-zA-Z0-9]+)~\*/(.|\n)+\W)(?<variable>[_a-zA-Z0-9]+\)\. \k<name>\(", "${before}${name}(${variable}", null,
63     ↳ 50),
64 // Remove markers
65 // /*~extensionMethod~BuildExceptionString~*/
66 //
67 (new Regex(@"/*~extensionMethod~[a-zA-Z0-9]+~\*/"), "", null, 0),
68 // (this
69 // (
70 (new Regex(@"(this "), "(", null, 0),
71 // public: static readonly EnsureAlwaysExtensionRoot Always = new
72     ↳ EnsureAlwaysExtensionRoot();
73 // public: inline static EnsureAlwaysExtensionRoot Always;
74 (new Regex(@"(?<access>(private|protected|public): )?static readonly
75     ↳ (?<type>[a-zA-Z0-9]+) (?<name>[_a-zA-Z0-9_]+) = new \k<type>\(\);"),
76     ↳ "${access}inline static ${type} ${name};", null, 0),
77 // public: static readonly string ExceptionContentsSeparator = "---";
78 // public: inline static const char* ExceptionContentsSeparator = "---";
79 (new Regex(@"(?<access>(private|protected|public): )?static readonly string
80     ↳ (?<name>[a-zA-Z0-9_]+) = ""(?:<string>(\\"|\\r\\n))+"";"), "${access}inline
81     ↳ static const char* ${name} = \"${string}\";", null, 0),
82 // private: const int MaxPath = 92;
83 // private: static const int MaxPath = 92;
84 (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly)
85     ↳ (?<type>[a-zA-Z0-9]+) (?<name>[_a-zA-Z0-9_]+) = (?<value>[^\r\n]+);"),
86     ↳ "${access}static const ${type} ${name} = ${value};", null, 0),
87 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
88     ↳ TArgument : class
89 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
90 (new Regex(@"(?<before> [a-zA-Z]+)(([a-zA-Z *,,]+, |))(?<type>[a-zA-Z]+)(?<after>(|
91     ↳ [a-zA-Z *,,]+)))[\r\n]+where \k<type> : class", "${before}${type}*${after}",
92     ↳ null, 0),
93 // protected: abstract TElement GetFirst();
94 // protected: virtual TElement GetFirst() = 0;
95 (new Regex(@"(?<access>(private|protected|public): )?abstract
96     ↳ (?<method>[^\r\n]+);"), "${access}virtual ${method} = 0;", null, 0),
97 // TElement GetFirst();
98 // virtual TElement GetFirst() = 0;
99 (new Regex(@"([^\r\n]+[ ]+)((?!return)[a-zA-Z0-9]+ [a-zA-Z0-9]+\([^\]\r\n]*\))\(\);[
100     ↳ ]*\r\n+)", "$1virtual $2 = 0$3", null, 1),
101 // protected: readonly TreeElement[] _elements;
102 // protected: TreeElement _elements[N];
103 (new Regex(@"(?<access>(private|protected|public): )?readonly
104     ↳ (?<type>[a-zA-Z0-9]+)([^\]]+) (?<name>[_a-zA-Z0-9_]+);"), "${access}${type}
105     ↳ ${name}[N];", null, 0),
106 // protected: readonly TElement Zero;
107 // protected: TElement Zero;
108 (new Regex(@"(?<access>(private|protected|public): )?readonly
109     ↳ (?<type>[a-zA-Z0-9]+) (?<name>[_a-zA-Z0-9_]+);"), "${access}${type} ${name};",
110     ↳ null, 0),
111 // internal
112 //
113 (new Regex(@"(\W)internal\s+"), "$1", null, 0),
114 // static void NotImplementedException(ThrowExtensionRoot root) => throw new
115     ↳ NotImplementedException();
116 // static void NotImplementedException(ThrowExtensionRoot root) { return throw new
117     ↳ NotImplementedException(); }
118 (new Regex(@"(^\s+)(private|protected|public)?(?: )?(template \<[^^\r\n]+\> )?(static
119     ↳ )?(override )?(?<name>[a-zA-Z0-9_]+
120     ↳ )([a-zA-Z0-9_]+\(\([^\]\r\n]*\)\s+=>\s+throw([^\r\n]+);"),
121     ↳ "$1$2$3$4$5$6$7$8($9) { throw$10; }", null, 0),
122 // SizeBalancedTree(int capacity) => a = b;
123 // 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>\()|(?<-parent
   ↳ hesis>~\)|[~() ,;\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(@"(?<before>, |)\(\)\{ return (?<expression>[~;\r\n]+); }"),
   ↳ "${before}&[&]() -> auto { return ${expression}; }", 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]+)"), "$1", null, 0),
156 // using Platform.Numbers;
157 //
158 (new Regex(@"([\r\n]{2}|~)\s*?using [\.a-zA-Z0-9+;~*?$]"), "", null, 0),
159 // struct TreeElement { }

```

```

160 // struct TreeElement { };
161 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([~;])", "$1
    ↳ $2$3{$4};$5", null, 0),
162 // class Program { }
163 // class Program { };
164 (new Regex(@"(struct|class) ([a-zA-Z0-9]+[^\r\n]*)([\r\n]+(?<indentLevel>[\t
    ↳ ]*)?)\{([\S\s]+?[\r\n]+\k<indentLevel>)\}([~;]|$)", "$1 $2$3{$4};$5", null, 0),
165 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
166 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
167 (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)", "class $1 : public $2", null,
    ↳ 0),
168 // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
169 // class IProperty : public ISetter<TValue, TObject>, IProvider<TValue, TObject>
170 (new Regex(@"(?<before>class [a-zA-Z0-9]+ : ((public [a-zA-Z0-9]+(<[a-zA-Z0-9
    ↳ ,]+>)?, )+)?(?<inheritedType>(?!public) [a-zA-Z0-9]+(<[a-zA-Z0-9
    ↳ ,]+>)?(?<after>(, [a-zA-Z0-9]+(?!>)|[\r\n]+)))", "${before}public
    ↳ ${inheritedType}${after}", null, 10),
171 // Insert scope borders.
172 // ref TElement root
173 // ~!root!~ref TElement root
174 (new Regex(@"(?<definition>(?!<= |\\() (ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<ref>
    ↳ (?<variable>[a-zA-Z0-9]+)(?!\\|, | =))", "~!${variable}!~${definition}", null,
    ↳ 0),
175 // Inside the scope of ~!root!~ replace:
176 // root
177 // *root
178 (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),
179 // Remove scope borders.
180 // ~!root!~
181 //
182 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
183 // ref auto root = ref
184 // ref auto root =
185 (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\\W)", "$1* $2 =$3", null, 0),
186 // *root = ref left;
187 // root = left;
188 (new Regex(@"\\*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\\W)", "$1 = $2$3", null, 0),
189 // (ref left)
190 // (left)
191 (new Regex(@"\\(ref ([a-zA-Z0-9]+)(\\|\\(|,))", "($1$2", null, 0),
192 // ref TElement
193 // TElement*
194 (new Regex(@"( |\\()ref ([a-zA-Z0-9]+) ", "$1$2* ", null, 0),
195 // ref sizeBalancedTree.Root
196 // &sizeBalancedTree->Root
197 (new Regex(@"ref ([a-zA-Z0-9]+)\\.([a-zA-Z0-9\\*]+)", "&$1->$2", null, 0),
198 // ref GetElement(node).Right
199 // &GetElement(node)->Right
200 (new Regex(@"ref ([a-zA-Z0-9]+)\\((([a-zA-Z0-9\\*]+)\\)\\.([a-zA-Z0-9]+)",
    ↳ "&$1($2)->$3", null, 0),
201 // GetElement(node).Right
202 // GetElement(node)->Right
203 (new Regex(@"([a-zA-Z0-9]+)\\((([a-zA-Z0-9\\*]+)\\)\\.([a-zA-Z0-9]+)", "$1($2)->$3",
    ↳ null, 0),
204 // [Fact] public: static void SizeBalancedTreeMultipleAttachAndDetachTest()
205 // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
206 (new Regex(@"\\[Fact\\][\\s\\n]+(public: )?(static )?void ([a-zA-Z0-9]+)\\(\\)", "public:
    ↳ TEST_METHOD($3)", null, 0),
207 // class TreesTests
208 // TEST_CLASS(TreesTests)
209 (new Regex(@"class ([a-zA-Z0-9]+)Tests", "TEST_CLASS($1)", null, 0),
210 // Assert.Equal
211 // Assert::AreEqual
212 (new Regex(@"(Assert)\\.Equal", "$1::AreEqual", null, 0),
213 // Assert.Throws
214 // Assert::ExpectException
215 (new Regex(@"(Assert)\\.Throws", "$1::ExpectException", null, 0),
216 // $"Argument {argumentName} is null."
217 // ((std::string)"Argument ").append(argumentName).append(" is null.").data()
218 (new Regex(@"\\$" "(?<left>(\\\"|\\\"\\\"\\r\\n))*{(?<expression>[_a-zA-Z0-9]+)}(?<right>(\\
    ↳ \\\"|\\\"\\\"\\r\\n))*" ",
    ↳ "\\((std::string)$\\\"${left}\\\".append(${expression}).append(\"\\\"${right}\\\".data()",
    ↳ null, 10),

```

```

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

```

```

(new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n}+\r\n[\t
]*)" (?<middle>((?!class)\.|\n)+?) (?<mutexDeclaration>private: inline static
std::mutex (?<fieldName>[_a-zA-Z0-9]+)_mutex;)" ),
"${classDeclarationBegin}/*${fieldName}~*/${middle}${mutexDeclaration}", null,
0),
// 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) (?<before>
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); ${after}" , null, 10),
// Inside the scope of ~!_exceptionsBag!~ replace:
// _exceptionsBag.Add(exception);
// std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
_exceptionsBag.Add(exception);
(new Regex(@"(?<scope>/\~*(?<fieldName>[_a-zA-Z0-9]+)~\~/) (?<separator>.\|\n) (?<before>
e>((?!/\~*\k<fieldName>~\~/)(\|\n))*?) { (?<after>((?!lock_guard)([~{};]\|\n))*? \r
?\n (?<indent>[\t ]*) \k<fieldName>[~;}\r\n]*);)" ),
"${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');

```

```

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

```



```

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

```



```

397         (new Regex(@"(?<before>\r?\n[~""\r\n]*("""\r\n|~""\r\n)*""[~""\r\n]*)(?<=\\W)null
    ↳   (?<after>\\W)"), "${before}nullptr${after}", null,
    ↳   10),
398     // default
399     // 0
400     (new Regex(@"(?<before>\r?\n[~""\r\n]*("""\r\n|~""\r\n)*""[~""\r\n]*)(?<=\\W)defa
    ↳   ult(?<after>\\W)"), "${before}0${after}", null,
    ↳   10),
401     // object x
402     // void *x
403     (new Regex(@"(?<before>\r?\n[~""\r\n]*("""\r\n|~""\r\n)*""[~""\r\n]*)(?<=\\W)([O|
    ↳   object|System\\.Object) (?<after>\\w)"), "${before}void *${after}", null,
    ↳   10),
404     // <object>
405     // <void*>
406     (new Regex(@"(?<before>\r?\n[~""\r\n]*("""\r\n|~""\r\n)*""[~""\r\n]*)(?<=\\W)([O|
    ↳   object|System\\.Object) (?<after>\\W)"), "${before}void*${after}", null,
    ↳   10),
407     // ArgumentException
408     // std::invalid_argument
409     (new Regex(@"(?<before>\r?\n[~""\r\n]*("""\r\n|~""\r\n)*""[~""\r\n]*)(?<=\\W)(Sys
    ↳   tem\\.)?ArgumentException(?<after>\\W)"),
    ↳   "${before}std::invalid_argument${after}", null, 10),
410     // #region Always
411     //
412     (new Regex(@"(\\|\\r?\\n)[ \\t]*#(region|endregion)[^\\r\\n]*(\\r?\\n|$)"), "", null, 0),
413     // //define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
414     //
415     (new Regex(@"\\|\\|/[ \\t]*#define[ \\t]*+_a-zA-Z0-9+_[ \\t]*"), "", null, 0),
416     // #if USEARRAYPOOL\\r\\n#endif
417     //
418     (new Regex(@"#if [a-zA-Z0-9]+\\s+#endif"), "", null, 0),
419     // [Fact]
420     //
421     (new Regex(@"(?<firstNewLine>\\r?\\n|\\A)(?<indent>[\\t
    ↳   ]+)[[a-zA-Z0-9]+(\\((?<expression>(\\(?<parenthesis>\\)|(?<-parenthesis>\\))|(~()\\r
    ↳   \\n)*+)(?<parenthesis>(?!))\\)?\\][ \\t]*(\\r?\\n\\k<indent>)?"),
    ↳   "${firstNewLine}${indent}", null, 5),
422     // \\n ... namespace
423     // namespace
424     (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+namespace"), "$1namespace", null, 0),
425     // \\n ... class
426     // class
427     (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+class"), "$1class", null, 0),
428     }.Cast<ISubstitutionRule>().ToList();
429
430     public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
    ↳   base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
431
432     public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
433 }
434 }

```

## 1.2 ./csharp/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 EmptyLineTest()
9         {
10             // This test can help to test basic problems with regular expressions like incorrect
    ↳   syntax
11             var transformer = new CSharpToCppTransformer();
12             var actualResult = transformer.Transform("", new Context(null));
13             Assert.Equal("", actualResult);
14         }
15
16         [Fact]
17         public void HelloWorldTest()
18         {
19             const string helloWorldCode = @"using System;
20 class Program
21 {
22     public static void Main(string[] args)
23     {

```

```
24         Console.WriteLine("Hello, world!");
25     }
26 };
27     const string expectedResult = @"class Program
28 {
29     public: static void Main(const char* args[])
30     {
31         printf("Hello, world!\n");
32     }
33 };";
34     var transformer = new CSharpToCppTransformer();
35     var actualResult = transformer.Transform(helloWorldCode, new Context(null));
36     Assert.Equal(expectedResult, actualResult);
37 }
38 }
39 }
```

## Index

./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs, 9

./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1