

## 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}: ${before}", null, 0),
38             // public: static bool CollectExceptions { get; set; }
39             // public: static bool CollectExceptions;
40             (new Regex(@"(?<before>(private|protected|public): (static )?[\r\n]+
41             → )(?<name>[a-zA-Z0-9]+) {[~;]}*(?<=\\W)get;[~;]}*(?<=\\W)set;[~;]}*"),
42             → "${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-Z<0-9]+)(\[\[ \]]+)(?<name>[_a-zA-Z0-9_]+);"), "${access}${type}
105     ↳ ${name}[N];", null, 0),
106 // protected: readonly TElement Zero;
107 // protected: TElement Zero;
108 (new Regex(@"(?<access>(private|protected|public): )?readonly
109     ↳ (?<type>[a-zA-Z<0-9]+) (?<name>[_a-zA-Z0-9_]+);"), "${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 )?([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+([^\;\r\n]+?);"), "( ) { return $1; }", 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 { }
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] npublic: 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", "Assert::AreEqual", null, 0),
213 // $Argument {argumentName} is null."
214 // ((std::string)"Argument ").append(argumentName).append(" is null.").data()
215 (new Regex(@"\\$""(?<left>\\\"\"|\\\"\"\\r\\n)*\\{(?<expression>[_a-zA-Z0-9]+)\\}(?<right>\\\"
    ↳ \\\"\"|\\\"\"\\r\\n)*\\\"\""),
    ↳ "((std::string)$\\\"${left}\\\".append(${expression}).append(\"${right}\\\".data()",
    ↳ null, 10),
216 // $"
217 // "
218 (new Regex(@"\\$"""), "\\\"", null, 0),
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 // public: static event EventHandler<std::exception> ExceptionIgnored =
→ OnExceptionIgnored; ... };
232 // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
→ const std::exception&> ExceptionIgnored = OnExceptionIgnored; };
233 (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),
234 // Insert scope borders.
235 // auto added = new StringBuilder();
236 // /*~sb~/std::string added;
237 (new Regex(@"(auto|(System\.Text\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new
→ (System\.Text\.)?StringBuilder\(\);)", "/*~${variable}~/std::string
→ ${variable};", null, 0),
238 // static void Indent(StringBuilder sb, int level)
239 // static void Indent(/*~sb~/StringBuilder sb, int level)
240 (new Regex(@"(?<start>, \|() (System\.Text\.)?StringBuilder
→ (?<variable>[a-zA-Z0-9]+)(?<end>, \|))", "${start}/*~${variable}~/std::string&
→ ${variable}${end}", null, 0),
241 // Inside the scope of ~!added!~ replace:
242 // sb.ToString()
243 // sb.data()
244 (new Regex(@"(?<scope>\/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
→ (((?!\/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.ToString\(\)\"",
→ "${scope}${separator}${before}${variable}.data()", null, 10),
245 // sb.AppendLine(argument)
246 // sb.append(argument).append('\\n')
247 (new Regex(@"(?<scope>\/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
→ (((?!\/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.AppendLine\((?<argument>[^\r\n]
→ r\\n)+\\)\"",
→ "${scope}${separator}${before}${variable}.append(${argument}).append('\\n')",
→ null, 10),
248 // sb.Append('\\t', level);
249 // sb.append(level, '\\t');
250 (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),
251 // sb.Append(argument)
252 // sb.append(argument)
253 (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),
254 // Remove scope borders.
255 // /*~sb~/
256 //
257 (new Regex(@"\/\*~(?<pointer>[a-zA-Z0-9]+)~\*/"), "", null, 0),
258 // Insert scope borders.
259 // auto added = new HashSet<TElement>();
260 // ~!added!~std::unordered_set<TElement> added;
261 (new Regex(@"(auto (?<variable>[a-zA-Z0-9]+) = new
→ HashSet<(?<element>[a-zA-Z0-9]+)>\(\);)",
→ "/*~${variable}!~std::unordered_set<${element}> ${variable};", null, 0),
262 // Inside the scope of ~!added!~ replace:
263 // added.Add(node)
264 // added.insert(node)

```

```

265 (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),
266 // Inside the scope of ~!added!~ replace:
267 // added.Remove(node)
268 // added.erase(node)
269 (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),
270 // if (added.insert(node)) {
271 // if (!added.contains(node)) { added.insert(node);
272 (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),
273 // Remove scope borders.
274 // ~!added!~
275 //
276 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
277 // Insert scope borders.
278 // auto random = new System.Random();
279 // std::srand(0);
280 (new Regex(@"[a-zA-Z0-9\.]+ ([a-zA-Z0-9]+) = new
    ↳ (System\.)?Random\((([a-zA-Z0-9]+\)\);", "~!$1!~std::srand($3);", null, 0),
281 // Inside the scope of ~!random!~ replace:
282 // random.Next(1, N)
283 // (std::rand() % N) + 1
284 (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),
285 // Remove scope borders.
286 // ~!random!~
287 //
288 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
289 // Insert method body scope starts.
290 // void PrintNodes(TElement node, StringBuilder sb, int level) {
291 // void PrintNodes(TElement node, StringBuilder sb, int level) { /*method-start*/
292 (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),
293 // Insert method body scope ends.
294 // { /*method-start*/ ... }
295 // { /*method-start*/ ... /*method-end*/ }
296 (new Regex(@"\{ /*method-start*/ (?<body>((?<bracket>\{) | (?<-bracket>\}) | [^\{\}])* )
    ↳ \}"), " /*method-start*/ ${body} /*method-end*/", null,
    ↳ 0),
297 // Inside method bodies replace:
298 // GetFirst(
299 // this->GetFirst(
300 // (new Regex(@"(?<separator>(\(| |([\W]) |return ))(?<!(-->|\\*
    ↳ ))(?<method>(?!sizeof)[a-zA-Z0-9]+\)\((?!\\) \{)",
    ↳ "${separator}this->${method}(", null, 1),
301 (new Regex(@"(?<scope>\/\*method-start\*\/)(?<before>((?<!(\/\*method-end\*\/)(.\|\\n))*?) (
    ↳ ?<separator>[\W] (?<!(::|\\.|-->))) (?<method>(?!sizeof)[a-zA-Z0-9]+\)\((?!\\)
    ↳ \{) (?<after>(\|\\n))*?) (?<scopeEnd>\/\*method-end\*\/)",
    ↳ "${scope}${before}${separator}this->${method} (${after}${scopeEnd}", null, 100),
302 // Remove scope borders.
303 // /*method-start*/
304 //
305 (new Regex(@"\/\*method-(start|end)\*\/"), "", null, 0),
306 // throw new ArgumentNullException(argumentName, message);
307 // throw std::invalid_argument(((std::string)"Argument
    ↳ ").append(argumentName).append(" is null: ").append(message).append("."));
308 (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),
309 // throw new ArgumentException(message, argumentName);
310 // throw std::invalid_argument(((std::string)"Invalid
    ↳ ").append(argumentName).append(" argument: ").append(message).append("."));

```

```

(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),
// throw new NotSupportedException();
// throw std::logic_error("Not supported exception.");
(new Regex(@"throw new NotSupportedException\(\);", "throw std::logic_error\\"Not
    supported exception.\\"");", null, 0),
// throw new NotImplementedException();
// throw std::logic_error("Not implemented exception.");
(new Regex(@"throw new NotImplementedException\(\);", "throw std::logic_error\\"Not
    implemented exception.\\"");", null, 0),
}.Cast<ISubstitutionRule>().ToList();

public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
{
    // ICounter<int, int> c1;
    // ICounter<int, int>* c1;
    (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^\r\n]+>?)
        (?<variable>[_a-zA-Z0-9]+);", "${abstractType}* ${variable};", null, 0),
    // (expression)
    // expression
    (new Regex(@"(\(|\)|\((([a-zA-Z0-9_*.:~\+])\)|\(|\)|\)|\))", "$1$2$3", null, 0),
    // (method(expression))
    // method(expression)
    (new Regex(@"(?<firstSeparator>(\(|
        ))\(((?<method>[a-zA-Z0-9_~\+]*:~\+))\(((?<expression>((?<parenthesis>\(|(?<-parenthesis>\)|[a-zA-Z0-9_~\+]*~\+)(?<parenthesis>(?!))\)|\)|\)|\))\)|\)|\))", "${firstSeparator}${method}(${expression})${lastSeparator}", null, 0),
    // return ref _elements[node];
    // return &_elements[node];
    (new Regex(@"return ref ([_a-zA-Z0-9~\+])\([([_a-zA-Z0-9~\+])\];", "return &$1[$2];",
        null, 0),
    // null
    // NULL
    (new Regex(@"(?<before>\r?\n[~""\r\n]*(""\\"""|["""\r\n])*""["""\r\n]*)(?<=\\W)null
        (?<after>\\W)", "${before}NULL${after}", null,
        10),
    // default
    // 0
    (new Regex(@"(?<before>\r?\n[~""\r\n]*(""\\"""|["""\r\n])*""["""\r\n]*)(?<=\\W)defa
        ult(?<after>\\W)", "${before}0${after}", null,
        10),
    // object x
    // void *x
    (new Regex(@"(?<before>\r?\n[~""\r\n]*(""\\"""|["""\r\n])*""["""\r\n]*)(?<=\\W)([0|
        ]object|System\\.Object) (?<after>\\w)", "${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

```
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
11             ↪ syntax
12             var transformer = new CSharpToCppTransformer();
13             var actualResult = transformer.Transform("", new Context(null));
14             Assert.Equal("", actualResult);
15         }
16
17         [Fact]
18         public void HelloWorldTest()
19         {
20             const string helloWorldCode = @"using System;
21 class Program
22 {
23     public static void Main(string[] args)
24     {
25         Console.WriteLine(""Hello, world!"");
26     }
27 }";
28             const string expectedResult = @"class Program
29 {
30     public: static void Main(const char* args[])
31     {
32         printf(""Hello, world!\n"");
33     }
34 };";
35             var transformer = new CSharpToCppTransformer();
36             var actualResult = transformer.Transform(helloWorldCode, new Context(null));
37             Assert.Equal(expectedResult, actualResult);
38         }
39     }
```



## Index

./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs, 8

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