

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

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

```

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 )?([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:]+\s*(?!return)) ([_a-zA-Z0-9]+\s*);", "$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]+\s*
→ ([_a-zA-Z0-9]+\s*\s*([_a-zA-Z0-9]+\s*))");", "$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]+\s*([_a-zA-Z0-9]+\s*) = new
→ ([_a-zA-Z0-9]+\s*\s*([_a-zA-Z0-9]+\s*))");", "$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]+\s*) = 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]+\s*)\(\)
→ { return (?<fieldName>[_a-zA-Z0-9]+\s*); }", "${access}static
→ std::vector<${argumentType}> ${methodName}() { return
→ std::vector<${argumentType}>({${fieldName}}); }", null, 0),
237 // public: static event EventHandler<std::exception> ExceptionIgnored =
→ OnExceptionIgnored; ... };
238 // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
→ const std::exception&> ExceptionIgnored = OnExceptionIgnored; };
239 (new Regex(@"(?<begin>\r?\n(\r?\n)?(?<halfIndent>[
→ \t]+\s*)k<halfIndent>)(?<access>(private|protected|public): )?static event
→ EventHandler<(?<argumentType>[^\r\n]+)> (?<name>[_a-zA-Z0-9]+\s*) = (?<defaultDele
→ gate>[_a-zA-Z0-9]+\s*);(?<middle>(\s*\n)+)?(?<end>\r?\n\s*k<halfIndent>});)",
→ "${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)\s*\n)+)?(?<vectorFieldDeclaration>(?(<access>(private|pro
→ tected|public): )static std::vector<(?<argumentType>[^\r\n]+)>
→ (?<fieldName>[_a-zA-Z0-9]+\s*);)",
→ "${classDeclarationBegin}/*~${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>\/\s*(?<fieldName>[_a-zA-Z0-9]+\s*)~\/)(?<separator>.\s*)(?<before
→ e>((?!\/\s*k<fieldName>\s*)\s*\n))*?)\s*k<fieldName>\s*.Add)",
→ "${scope}${separator}${before}${fieldName}.push_back", null, 10),
248 // Remove scope borders.
249 // /*~_exceptionsBag~*/
250 //
251 (new Regex(@"\/\s*~[_a-zA-Z0-9]+\s*~\/"), "", 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)\s*\n)+)?(?<mutexDeclaration>private: static std::mutex
→ (?<fieldName>[_a-zA-Z0-9]+\s*)_mutex;)",
→ "${classDeclarationBegin}/*~${fieldName}~*/${middle}${mutexDeclaration}", null,
→ 0),
256 // Inside the scope of ~!_exceptionsBag!~ replace:

```

```

257 // return std::vector<std::exception>(_exceptionsBag);
258 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
    ↳ std::vector<std::exception>(_exceptionsBag);
259 (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);${after}", null, 10),
260 // Inside the scope of ~!_exceptionsBag!~ replace:
261 // _exceptionsBag.Add(exception);
262 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \\r\\n
    ↳ _exceptionsBag.Add(exception);
263 (new Regex(@"(?<scope>/\~*(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↳ e((?!/\~*\k<fieldName>~\*/)(.\\n))*?){(?<after>((?!lock_guard)[^{};\\r\\n])*\\r
    ↳ ?\\n(?<indent>[ \\t]*)\\k<fieldName>[~;]\\r\\n*};)",
    ↳ "${scope}${separator}${before}{ " + Environment.NewLine +
    ↳ "${indent}std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", null,
    ↳ 10),
264 // Remove scope borders.
265 // /*~_exceptionsBag~*/
266 //
267 (new Regex(@"/*~[_a-zA-Z0-9]+~\*/)", "", null, 0),
268 // Insert scope borders.
269 // class IgnoredExceptions { ... public: static inline
    ↳ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
    ↳ ExceptionIgnored = OnExceptionIgnored;
270 // class IgnoredExceptions { /*~ExceptionIgnored~*/ ... public: static inline
    ↳ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
    ↳ ExceptionIgnored = OnExceptionIgnored;
271 (new Regex(@"(?<classDeclarationBegin>\\r?\\n(?<indent>[ \\t ]*)class [^\\r\\n]+\\r\\n[ \\t
    ↳ ]*}{(?<middle>((?!class).\\n)+)?(?<eventDeclaration>(?<access>(private|protected
    ↳ |public): )static inline
    ↳ Platform::Delegates::MulticastDelegate<(?<argumentType>[~;\\r\\n]+)>
    ↳ (?<name>[_a-zA-Z0-9]+) = (?<defaultDelegate>[_a-zA-Z0-9]+);)",
    ↳ "${classDeclarationBegin}/*~${name}~*/${middle}${eventDeclaration}", null, 0),
272 // Inside the scope of ~!ExceptionIgnored!~ replace:
273 // ExceptionIgnored.Invoke(NULL, exception);
274 // ExceptionIgnored(NULL, exception);
275 (new Regex(@"(?<scope>/\~*(?<eventName>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↳ >((?!/\~*\k<eventName>~\*/)(.\\n))*?)\\k<eventName>\\.Invoke)",
    ↳ "${scope}${separator}${before}${eventName}", null, 10),
276 // Remove scope borders.
277 // /*~ExceptionIgnored~*/
278 //
279 (new Regex(@"/*~[_a-zA-Z0-9]+~\*/)", "", null, 0),
280 // Insert scope borders.
281 // auto added = new StringBuilder();
282 // /*~sb~*/std::string added;
283 (new Regex(@"(auto|(System\\.Text\\.)?StringBuilder) (?<variable>[_a-zA-Z0-9]+) = new
    ↳ (System\\.Text\\.)?StringBuilder\\(\\);)", "/*~${variable}~*/std::string
    ↳ ${variable};", null, 0),
284 // static void Indent(StringBuilder sb, int level)
285 // static void Indent(/*~sb~*/StringBuilder sb, int level)
286 (new Regex(@"(?<start>, \\|() (System\\.Text\\.)?StringBuilder
    ↳ (?<variable>[_a-zA-Z0-9]+)(?<end>, \\|))", "${start}/*~${variable}~*/std::string&
    ↳ ${variable}${end}", null, 0),
287 // Inside the scope of ~!added!~ replace:
288 // sb.ToString()
289 // sb.data()
290 (new Regex(@"(?<scope>/\~*(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↳ ((?!/\~*\k<variable>~\*/)(.\\n))*?)\\k<variable>\\.ToString\\(\\)",
    ↳ "${scope}${separator}${before}${variable}.data()", null, 10),
291 // sb.AppendLine(argument)
292 // sb.append(argument).append('\\n')
293 (new Regex(@"(?<scope>/\~*(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↳ ((?!/\~*\k<variable>~\*/)(.\\n))*?)\\k<variable>\\.AppendLine\\((?<argument>[^\n], \\
    ↳ r\\n)+\\)",
    ↳ "${scope}${separator}${before}${variable}.append(${argument}).append(1, '\\n')",
    ↳ null, 10),
294 // sb.Append('\\t', level);
295 // sb.append(level, '\\t');
296 (new Regex(@"(?<scope>/\~*(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↳ ((?!/\~*\k<variable>~\*/)(.\\n))*?)\\k<variable>\\.Append\\('(?<character>[^\r\\n]
    ↳ +)', (?<count>[^\n], \\r\\n)+\\)",
    ↳ "${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();
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),

```



```

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),
    // (expression)
    // expression
383     (new Regex(@"\\(| )\\(((a-zA-Z0-9_\\*:]*)\\(| |;|\\)\\)"), "$1$2$3", null, 0),
384     // (method(expression))
    // method(expression)
385     (new Regex(@"(?<firstSeparator>\\(|
    ↳ ))\\(((?<method>[a-zA-Z0-9_\\->\\*:]*)\\(((?<expression>((?<parenthesis>\\(|(?<-parent
    ↳ hesis>\\)|[a-zA-Z0-9_\\->\\*:]*)\\((?<parenthesis>(?!))\\)\\)(?<lastSeparator>\\(|
    ↳ |;|\\)\\)\\)", "${firstSeparator}${method}(${expression})${lastSeparator}", null, 0),
386     // return ref _elements[node];
    // return &elements[node];
387     (new Regex(@"return ref ([a-zA-Z0-9]+)\\([([a-zA-Z0-9_\\*:]*)\\]");", "return &$1[$2];",
    ↳ null, 0),
388     // null
    // nullptr
389     (new Regex(@"(?<before>\\r?\\n[~""\\r\\n]*(""\\\\"|~""\\r\\n)*""[~""\\r\\n]*)(?<=\\W)null
    ↳ (?<after>\\W)"), "${before}nullptr${after}", null,
    ↳ 10),
390     // default
    // 0

```



```

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

```

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;
class Program
20 {
21     public static void Main(string[] args)
22     {
23         Console.WriteLine("Hello, world!");
24     }
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
35             var transformer = new CSharpToCppTransformer();
36             var actualResult = transformer.Transform(helloWorldCode, new Context(null));
37             Assert.Equal(expectedResult, actualResult);
38         }
39     }
40 }

```


Index

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

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