

1.1 ./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             // static class Ensure ... public static readonly EnsureAlwaysExtensionRoot Always =
33             → new EnsureAlwaysExtensionRoot(); ... } }
34             // static class Ensure ... static EnsureAlwaysExtensionRoot Always; ... }
35             → EnsureAlwaysExtensionRoot Ensure::Always; }
36             (new Regex(@"static class (?<class>[a-zA-Z0-9]+)(?<before>[\\s\\S\\r\\n]+)public static
37             → readonly (?<type>[a-zA-Z0-9]+) (?<name>[a-zA-Z0-9_]+) = new
38             → \\k<type>\\(\\);(?<after>[\\s\\S]+[\\r\\n]+)(?<indent>[ ]+){(?<ending>[a-zA-Z:;
39             → \\r\\n]+}[\\r\\n]+$)"), "static class ${class}${before}static ${type}
40             → ${name};${after}${indent}\\r\\n${indent}${type} ${class}::${name};${ending}",
41             → null, 10),
42             // public abstract class
43             // class
44             (new Regex(@"(public abstract|static) class") , "class", null, 0),
45             // class GenericCollectionMethodsBase {
46             // class GenericCollectionMethodsBase { public:
47             (new Regex(@"class ([a-zA-Z0-9]+)(\\s+){") , "class $1$2{" + Environment.NewLine + "
48             → public:", null, 0),
49             // class GenericCollectionMethodsBase<TElement> {
50             // template <typename TElement> class GenericCollectionMethodsBase { public:
51             (new Regex(@"class ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>([\\r\\n]+){") , "template <typename $2>
52             → class $1$3{" + Environment.NewLine + "      public:", null, 0),
53             // static void
54             → TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
55             → tree, TElement* root)
56             // template<typename T> static void
57             → TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
58             → tree, TElement* root)
59             (new Regex(@"static ([a-zA-Z0-9]+) ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>\\((([\\r\\n]+)\\))") ,
60             → "template <typename $3> static $1 $2($4)", null, 0),
61             // interface IFactory<out TProduct> {
62             // template <typename TProduct> class IFactory { public:
63             (new Regex(@"interface (?<interface>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9
64             → ,]+)>(?<whitespace>[\\r\\n]+){") , "template <typename...> class ${interface};
65             → template <typename ${typeParameters}> class
66             → ${interface}<${typeParameters}>${whitespace}" + Environment.NewLine + "
67             → public:", null, 0),
68             // template <typename TObject, TProperty, TValue>
69             // template <typename TObject, typename TProperty, TValue>
70             (new Regex(@"(?<before>template <((, )?typename [a-zA-Z0-9]+)+,
71             → )(?<typeParameter>[a-zA-Z0-9]+)(?<after>(,|>))") , "${before}typename
72             → ${typeParameter}${after}", null, 10),
73             // (this
74             // (

```

```

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

```

```

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

```

```

181 (new Regex(@"( |\()ref ([a-zA-Z0-9]+) "), "$1$2* ", null, 0),
182 // ref sizeBalancedTree.Root
183 // &sizeBalancedTree->Root
184 (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)"), "&$1->$2", null, 0),
185 // ref GetElement(node).Right
186 // &GetElement(node)->Right
187 (new Regex(@"ref ([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"),
188     ↳ "$1($2)->$3", null, 0),
189 // GetElement(node).Right
190 // GetElement(node)->Right
191 (new Regex(@"([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"), "$1($2)->$3",
192     ↳ null, 0),
193 // [Fact]\npublic static void SizeBalancedTreeMultipleAttachAndDetachTest()
194 // TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
195 (new Regex(@"\[Fact\]\[s\n\]+(static )?void ([a-zA-Z0-9]+)\(\)"), "TEST_METHOD($2)",
196     ↳ null, 0),
197 // class TreesTests
198 // TEST_CLASS(TreesTests)
199 (new Regex(@"class ([a-zA-Z0-9]+)Tests"), "TEST_CLASS($1)", null, 0),
200 // Assert.Equal
201 // Assert::AreEqual
202 (new Regex(@"Assert\.Equal"), "Assert::AreEqual", null, 0),
203 // TElement Root;
204 // TElement Root = 0;
205 (new Regex(@"(\\r?\\n[\\t ]+)([a-zA-Z0-9:_]+(?:!return)) ([_a-zA-Z0-9]+);"), "$1$2 $3 =
206     ↳ 0;", null, 0),
207 // TreeElement _elements[N];
208 // TreeElement _elements[N] = { {0} };
209 (new Regex(@"(\\r?\\n[\\t ]+)([a-zA-Z0-9]+) ([_a-zA-Z0-9]+)\\([([_a-zA-Z0-9]+)\\];"),
210     ↳ "$1$2 $3[$4] = { {0} };", null, 0),
211 // auto path = new TElement[MaxPath];
212 // TElement path[MaxPath] = { {0} };
213 (new Regex(@"(\\r?\\n[\\t ]+)[a-zA-Z0-9]+ ([a-zA-Z0-9]+) = new
214     ↳ ([a-zA-Z0-9]+)\\([([_a-zA-Z0-9]+)\\];", "$1$3 $2[$4] = { {0} };", null, 0),
215 // Insert scope borders.
216 // auto added = new HashSet<TElement>();
217 // ~!added!~std::unordered_set<TElement> added;
218 (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
219     ↳ HashSet<(?<element>[a-zA-Z0-9]+)>\\(\)");
220     ↳ "~!${variable}!~std::unordered_set<${element}> ${variable};", null, 0),
221 // Inside the scope of ~!added!~ replace:
222 // added.Add(node)
223 // added.insert(node)
224 (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>\\.|\\n)(?<before>((?<
225     ↳ !~!\\k<variable>!~)(\\.|\\n))*?)\\k<variable>\\.Add\\((?<argument>[a-zA-Z0-9]+)\\)"),
226     ↳ "${scope}${separator}${before}${variable}.insert(${argument})", null, 10),
227 // Inside the scope of ~!added!~ replace:
228 // added.Remove(node)
229 // added.erase(node)
230 (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>\\.|\\n)(?<before>((?<
231     ↳ !~!\\k<variable>!~)(\\.|\\n))*?)\\k<variable>\\.Remove\\((?<argument>[a-zA-Z0-9]+)\\)"),
232     ↳ "${scope}${separator}${before}${variable}.erase(${argument})", null, 10),
233 // if (added.insert(node)) {
234 // if (!added.contains(node)) { added.insert(node);
235 (new Regex(@"if \\((?<variable>[a-zA-Z0-9]+)\\.insert\\((?<argument>[a-zA-Z0-9]+)\\)\\)(?
236     ↳ <separator>[\\t ]*[\\r\\n]+(?:<indent>[\\t ]*){")", "if
237     ↳ (!${variable}.contains(${argument}))${separator}${indent}{ " +
238     ↳ Environment.NewLine + "${indent}    ${variable}.insert(${argument});", null, 0),
239 // Remove scope borders.
240 // ~!added!~
241 //
242 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
243 // Insert scope borders.
244 // auto random = new System.Random(0);
245 // std::srand(0);
246 (new Regex(@"[a-zA-Z0-9\\.]+ ([a-zA-Z0-9]+) = new
247     ↳ (System\\.)?Random\\((([a-zA-Z0-9]+)\\)");, "~!$1!~std::srand($3);", null, 0),
248 // Inside the scope of ~!random!~ replace:
249 // random.Next(1, N)
250 // (std::rand() % N) + 1
251 (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>\\.|\\n)(?<before>((?<
252     ↳ !~!\\k<variable>!~)(\\.|\\n))*?)\\k<variable>\\.Next\\((?<from>[a-zA-Z0-9]+),
253     ↳ (?<to>[a-zA-Z0-9]+)\\)"), "${scope}${separator}${before}(std::rand() % ${to}) +
254     ↳ ${from}", null, 10),
255 // Remove scope borders.
256 // ~!random!~
257

```

```

238 //
239 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
240 // Insert method body scope starts.
241 // void PrintNodes(TElement node, StringBuilder sb, int level) {
242 // void PrintNodes(TElement node, StringBuilder sb, int level) { /*method-start*/
243 (new Regex(@"(?<start>\r?\n[\t ]+)(?<prefix>((virtual )?[a-zA-Z0-9:_]+
    → )?)(?<method>[a-zA-Z][a-zA-Z0-9]*)\(((?<arguments>[^\s]*)\)(?<override>(
    → override)?)(?<separator>[ \t\r\n]*)\{((?<end>[~])")", "${start}${prefix}${method}_
    → (${arguments})${override}${separator}{ /*method-start*/${end}", null,
    → 0),
244 // Insert method body scope ends.
245 // { /*method-start*/... }
246 // { /*method-start*/... /*method-end*/ }
247 (new Regex(@"\{ /\*method-start\*/ (?<body>((?<bracket>\{) | (?<-bracket>\}) | [^\{\}]* )+ )
    → \}"), "{ /*method-start*/${body} /*method-end*/", null,
    → 0),
248 // Inside method bodies replace:
249 // GetFirst(
250 // this->GetFirst(
251 // (new Regex(@"(?<separator>(\(| |([W]) |return ))(?<!(->|\\*
    → ))(?<method>(?!sizeof)[a-zA-Z0-9]+)\(((?!\\) \{)"),
    → "${separator}this->${method}(", null, 1),
252 (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),
253 // Remove scope borders.
254 // /*method-start*/
255 //
256 (new Regex(@" /\*method-(start|end)\*/"), "", null, 0),
257 // throw new ArgumentNullException(argumentName, message);
258 // throw std::invalid_argument(((std::string)"Argument
    → ").append(argumentName).append(" is null: ").append(message).append("."));
259 (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),
260 // throw new ArgumentException(message, argumentName);
261 // throw std::invalid_argument(((std::string)"Invalid
    → ").append(argumentName).append(" argument: ").append(message).append("."));
262 (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),
263 // throw new NotSupportedException();
264 // throw std::logic_error("Not supported exception.");
265 (new Regex(@"throw new NotSupportedException\(\);"), "throw std::logic_error(\"Not
    → supported exception.\");", null, 0),
266 // throw new NotImplementedException();
267 // throw std::logic_error("Not implemented exception.");
268 (new Regex(@"throw new NotImplementedException\(\);"), "throw std::logic_error(\"Not
    → implemented exception.\");", null, 0),
269
270 }.Cast<ISubstitutionRule>().ToList();
271
272 public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
273 {
274     // ICounter<int, int> c1;
275     // ICounter<int, int>* c1;
276     (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^\r\n]+>)?)
    → (?<variable>[_a-zA-Z0-9]+);"), "${abstractType}* ${variable};", null, 0),
277     // (expression)
278     // expression
279     (new Regex(@"(\(| |)\((([a-zA-Z0-9_\*:]+)\)(, | |; |\\))"), "$1$2$3", null, 0),
280     // (method(expression))
281     // method(expression)
282     (new Regex(@"(?<firstSeparator>(\(|
    → ))\(((?<method>[a-zA-Z0-9_\->]*)+)\(((?<expression>((?<parenthesis>\(| (?<-parent
    → hesis>\)| [a-zA-Z0-9_\->]* )+ )?(parenthesis)(?!))\)\)\((?<lastSeparator>(, |
    → |; |\\))")", "${firstSeparator}${method} (${expression})${lastSeparator}", null, 0),
283     // return ref _elements[node];
284     // return &_elements[node];
285     (new Regex(@"return ref ([_a-zA-Z0-9]+)\([([_a-zA-Z0-9_\*:]+)\];"), "return &$1[$2];",
    → null, 0),
286     // default

```

```

287 // 0
288 (new Regex(@"(\\W)default(\\W)", "${1}0$2", null, 0),
289 // //define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
290 //
291 (new Regex(@"\\/\\/[/ \t]*#define[ \t]+[_a-zA-Z0-9]+[ \t]*"), "", null, 0),
292 // #if USEARRAYPOOL\r\n#endif
293 //
294 (new Regex(@"#if [a-zA-Z0-9]+\s+#endif"), "", null, 0),
295 // [Fact]
296 //
297 (new Regex(@"(?<firstNewLine>\r?\n|\\A)(?<indent>[ \t ]+)\\[ [a-zA-Z0-9]+(\\((?<expressio
↵ n>((?<parenthesis>\()|(?<-parenthesis>\))|[\^()]*)+)(?(parenthesis)(?!))\)\)?\[ [
↵ \t]*((\r?\n\k<indent>)?")", "${firstNewLine}${indent}", null, 5),
298 // \n ... namespace
299 // namespace
300 (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+namespace)", "$1namespace", null, 0),
301 // \n ... class
302 // class
303 (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+class)", "$1class", null, 0),
304 }.Cast<ISubstitutionRule>().ToList();
305
306 public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
↵ base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
307
308 public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
309 }
310 }

```

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

```

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

```

Index

./Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs, 6
./Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1