

LinksPlatform's Platform.RegularExpressions.Transformer.CSharpToCpp Class Library

./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             (new Regex(@"^-s*?\#pragma[sa-zA-Z0-9]+$"), "", null, 0),
20             // [MethodImpl(MethodImplOptions.AggressiveInlining)]
21             //
22             (new Regex(@"$\s+[MethodImpl\(MethodImplOptions\.AggressiveInlining\)\\"]), "",
23             // null, 0),
24             // [Fact]
25             //
26             (new Regex(@"$\s+[Fact\\"]), "", null, 0),
27             // {
28             //
29             (new Regex(@"$\s+[\\r\\n]+"), "{" + Environment.NewLine, null, 0),
30             // Platform.Collections.Methods.Lists
31             // Platform::Collections::Methods::Lists
32             (new Regex(@"(namespace[~\\r\\n]+?)\\.([~\\r\\n]+?)"), "$1::$2", null, 20),
33             // public abstract class
34             // class
35             (new Regex(@"(public abstract|static) class"), "class", null, 0),
36             // class GenericCollectionMethodsBase {
37             // class GenericCollectionMethodsBase { public:
38             (new Regex(@"class ([a-zA-Z0-9]+)(\\s+){")", "class $1$2{" + Environment.NewLine + "
39             // public:", null, 0),
40             // class GenericCollectionMethodsBase<TElement> {
41             // template <typename TElement> class GenericCollectionMethodsBase { public:
42             (new Regex(@"class ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>([~\\r\\n]+){")", "template <typename $2>
43             // class $1$3{" + Environment.NewLine + " public:", null, 0),
44             // static void
45             // TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
46             // tree, TElement* root)
47             // template<typename T> static void
48             // TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
49             // tree, TElement* root)
50             (new Regex(@"static ([a-zA-Z0-9]+) ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>\\((([~\\r\\n]+)\\))"),
51             // "template <typename $3> static $1 $2($4)", null, 0),
52             // (this
53             // (
54             (new Regex(@"\\(this ")", "(", null, 0),
55             // Func<TElement> treeCount
56             // TElement(*treeCount)()
57             (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)", "$1(*$2)()", null, 0),
58             // Action<TElement> free
59             // void (*free)(TElement)
60             (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)", "void (*$2)($1)", null, 0),
61             // private const int MaxPath = 92;
62             // static const int MaxPath = 92;
63             (new Regex(@"private const ([a-zA-Z0-9]+) ([_a-zA-Z0-9]+) = ([a-zA-Z0-9]+);"),
64             // "static const $1 $2 = $3;", null, 0),
65             // protected virtual
66             // virtual
67             (new Regex(@"protected virtual"), "virtual", null, 0),
68             // protected abstract TElement GetFirst();
69             // virtual TElement GetFirst() = 0;
70             (new Regex(@"protected abstract ([~;]+);"), "virtual $1 = 0;", null, 0),
71             // public virtual
72             // virtual
73             (new Regex(@"public virtual"), "virtual", null, 0),
74             // protected readonly
```

```

66 //
67 (new Regex(@"protected readonly "), "", null, 0),
68 // protected readonly TreeElement[] _elements;
69 // TreeElement _elements[N];
70 (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+)([\\[]+)"
71   ↳ ([_a-zA-Z0-9]+);"), "$2 $4[N];", null, 0),
72 // protected readonly TElement Zero;
73 // TElement Zero;
74 (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+) ([_a-zA-Z0-9]+);"), "$2
75   ↳ $3;", null, 0),
76 // private
77 //
78 (new Regex(@"(\W)(private|protected|public|internal) "), "$1", null, 0),
79 // SizeBalancedTree(int capacity) => a = b;
80 // SizeBalancedTree(int capacity) { a = b; }
81 (new Regex(@"(^\s+)(override )?(void )?([a-zA-Z0-9]+)\(((^\([])\s+=>\s+([~;]+);"),
82   ↳ "$1$2$3$4($5) { $6; }", null, 0),
83 // () => Integer<TElement>.Zero,
84 // () { return Integer<TElement>.Zero; },
85 (new Regex(@"\(\)\s+=>\s+([~\r\n;]+?);"), "() { return $1; }", null, 0),
86 // => Integer<TElement>.Zero;
87 // { return Integer<TElement>.Zero; }
88 (new Regex(@"\)\s+=>\s+([~\r\n;]+?);"), "() { return $1; }", null, 0),
89 // () { return avlTree.Count; }
90 // []()-> auto { return avlTree.Count; }
91 (new Regex(@"\(\)\s+=>\s+([~\r\n;]+?);"), "[]()-> auto { return $1; }", null, 0),
92 // Count => GetSizeOrZero(Root);
93 // GetCount() { return GetSizeOrZero(Root); }
94 (new Regex(@"([A-Z][a-z]+)\s+=>\s+([~;]+);"), "Get$1() { return $2; }", null, 0),
95 // var
96 // auto
97 (new Regex(@"(\W)var(\W)"), "$1auto$2", null, 0),
98 // unchecked
99 //
100 (new Regex(@"[\r\n]{2}\s*?unchecked\s*?$"), "", null, 0),
101 // $"
102 // "
103 (new Regex(@"\$"""), "\"", null, 0),
104 // Console.WriteLine("...")
105 // printf("...\n")
106 (new Regex(@"Console\.WriteLine\\(\"([~\""]+)\"\\\""), "printf(\"$1\\n\\n\"", null, 0),
107 // throw new InvalidOperationException
108 // throw std::exception
109 (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
110   ↳ std::exception", null, 0),
111 // override void PrintNode(TElement node, StringBuilder sb, int level)
112 // void PrintNode(TElement node, StringBuilder sb, int level) override
113 (new Regex(@"override ([a-zA-Z0-9 \*+]+)(\([^\)]+?\))"), "$1$2 override", null, 0),
114 // string
115 // char*
116 (new Regex(@"(\W)string(\W)"), "$1char*$2", null, 0),
117 // sbyte
118 // std::int8_t
119 (new Regex(@"(\W)sbyte(\W)"), "$1std::int8_t$2", null, 0),
120 // uint
121 // std::uint32_t
122 (new Regex(@"(\W)uint(\W)"), "$1std::uint32_t$2", null, 0),
123 // char*[] args
124 // char* args[]
125 (new Regex(@"([_a-zA-Z0-9\*+]?)\\[] ([a-zA-Z0-9]+)"), "$1 $2[]", null, 0),
126 // using Platform.Numbers;
127 //
128 (new Regex(@"([\r\n]{2}|^)\s*?using [\.a-zA-Z0-9+;]\s*?$"), "", null, 0),
129 // struct TreeElement { }
130 // struct TreeElement { };
131 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([~;])"), "$1
132   ↳ $2$3{$4};$5", null, 0),
133 // class Program { }
134 // class Program { };
135 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)[~\r\n]*)([\r\n]+(?<indentLevel>[\t
136   ↳ ]*)?)\{([\S\s]+?[~\r\n]+\k<indentLevel>)\}([~;]|$)"), "$1 $2$3{$4};$5", null, 0),
137 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
138 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
139 (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)"), "class $1 : public $2", null,
140   ↳ 0),
141 // Insert scope borders.
142 // ref TElement root

```

```

136 // ~!root!~ref TElement root
137 (new Regex(@"(?<definition>(?!<= \\\() (ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!ref))
    ↳ (?<variable>[a-zA-Z0-9]+)(?<=\\|,| |=)")), "~!${variable}!~!${definition}", null,
    ↳ 0),
138 // Inside the scope of ~!root!~ replace:
139 // root
140 // *root
141 (new Regex(@"(?<definition>~! (?<pointer>[a-zA-Z0-9]+)!~ref [a-zA-Z0-9]+
    ↳ (?<pointer>[a-zA-Z0-9]+)(?<=\\|,| |=)
    ↳ =)) (?<before>((?!~!\\k<pointer>!~)(.|\n))*?) (?<prefix>(\W
    ↳ |\\))\\k<pointer>(?<suffix>( |\\|;|,))"),
    ↳ "${definition}${before}${prefix}*${pointer}${suffix}", null, 70),
142 // Remove scope borders.
143 // ~!root!~
144 //
145 (new Regex(@"~! (?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
146 // ref auto root = ref
147 // ref auto root =
148 (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)"), "$1* $2 =$3", null, 0),
149 // *root = ref left;
150 // root = left;
151 (new Regex(@"\*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", null, 0),
152 // (ref left)
153 // (left)
154 (new Regex(@"\ (ref ([a-zA-Z0-9]+)(\\|\\(|,))"), "($1$2", null, 0),
155 // ref TElement
156 // TElement*
157 (new Regex(@"( |\\()ref ([a-zA-Z0-9]+) "), "$1$2* ", null, 0),
158 // ref sizeBalancedTree2.Root
159 // &sizeBalancedTree2->Root
160 (new Regex(@"ref ([a-zA-Z0-9]+)\\.([a-zA-Z0-9\\*]+)"), "&$1->$2", null, 0),
161 // ref GetElement(node).Right
162 // &GetElement(node)->Right
163 (new Regex(@"ref ([a-zA-Z0-9]+)\\((([a-zA-Z0-9\\*]+)\\)\\.([a-zA-Z0-9]+)"),
    ↳ "&$1($2)->$3", null, 0),
164 // GetElement(node).Right
165 // GetElement(node)->Right
166 (new Regex(@"([a-zA-Z0-9]+)\\((([a-zA-Z0-9\\*]+)\\)\\.([a-zA-Z0-9]+)"), "$1($2)->$3",
    ↳ null, 0),
167 }.Cast<ISubstitutionRule>().ToList();
168
169 public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
170 {
171     // (expression)
172     // expression
173     (new Regex(@"(\\(|)\\(((\\[a-zA-Z0-9_\\*:]*)\\)(,| |;|\\))"), "$1$2$3", null, 0),
174     // (method(expression))
175     // method(expression)
176     (new Regex(@"(?<firstSeparator>(\\(|)
    ↳ ))\\((?<method>[a-zA-Z0-9_\\->\\*:]*)\\((?<expression>((?<parenthesis>\\(|) (?<-parent
    ↳ hesis>\\)|[a-zA-Z0-9_\\->\\*:]*) (?<(parenthesis) (?!)\\)\\) (?<lastSeparator>(,|
    ↳ |;|\\)))"), "${firstSeparator}${method}(${expression})${lastSeparator}", null, 0),
177     // return ref _elements[node];
178     // return &elements[node];
179     (new Regex(@"return ref ([_a-zA-Z0-9]+)\\((([_a-zA-Z0-9\\*]+)\\);"), "return &$1[$2];",
    ↳ null, 0),
180     // default
181     // 0
182     (new Regex(@"(\\W)default(\\W)"), "${1}0$2", null, 0),
183     // // #define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
184     //
185     (new Regex(@"\\[/\\[ \\t]*#define[ \\t]+[_a-zA-Z0-9]+[ \\t]*"), "", null, 0),
186     // #if USEARRAYPOOL\\r\\n#endif
187     //
188     (new Regex(@"#if [a-zA-Z0-9]+\\s+#endif"), "", null, 0),
189     // \\n ... namespace
190     // namespace
191     (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+namespace"), "$1namespace", null, 0),
192     // \\n ... class
193     // class
194     (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+class"), "$1class", null, 0),
195 }.Cast<ISubstitutionRule>().ToList();
196
197 public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
    ↳ base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
198
199 public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }

```

```
200     }
201 }
```

./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, 4
./Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1