

# 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
```

```

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

```

```

// Insert scope borders.
// ref TElement root
// ~!root!~ref TElement root
(new Regex(@"(?<definition>(?!<=|\\() (ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!ref))
    => (?<variable>[a-zA-Z0-9]+)(?=\\|,|=)")), "~!${variable}!~${definition}", null,
    => 0),
// Inside the scope of ~!root!~ replace:
// root
// *root
(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),
// Remove scope borders.
// ~!root!~
//
(new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
// ref auto root = ref
// ref auto root =
(new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\\W)", "$1* $2 =$3", null, 0),
// *root = ref left;
// root = left;
(new Regex(@"\\*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\\W)", "$1 = $2$3", null, 0),
// (ref left)
// (left)
(new Regex(@"\\(ref ([a-zA-Z0-9]+)(\\|\\(|,))", "($1$2", null, 0),
// ref TElement
// TElement*
(new Regex(@"(\\|\\()ref ([a-zA-Z0-9]+) ", "$1$2* ", null, 0),
// ref sizeBalancedTree2.Root
// &sizeBalancedTree2->Root
(new Regex(@"ref ([a-zA-Z0-9]+)\\.([a-zA-Z0-9\\*]+)", "&$1->$2", null, 0),
// ref GetElement(node).Right
// &GetElement(node)->Right
(new Regex(@"ref ([a-zA-Z0-9]+)\\((([a-zA-Z0-9\\*]+)\\)\\.([a-zA-Z0-9]+)",
    => "&$1($2)->$3", null, 0),
// GetElement(node).Right
// GetElement(node)->Right
(new Regex(@"([a-zA-Z0-9]+)\\((([a-zA-Z0-9\\*]+)\\)\\.([a-zA-Z0-9]+)", "$1($2)->$3",
    => null, 0),
}.Cast<ISubstitutionRule>().ToList();

public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
{
    // (expression)
    // expression
    (new Regex(@"(\\(|)\\((([a-zA-Z0-9\\*:]*)\\)(,| |;|\\|))", "$1$2$3", null, 0),
    // (method(expression))
    // method(expression)
    (new Regex(@"(?<firstSeparator>(\\(|
        ))\\(((?<method>[a-zA-Z0-9\\->\\*:]*)\\(((?<expression>((?<parenthesis>(\\(|(?<-parent
        hesis>\\)|[a-zA-Z0-9\\->\\*:]*)\\)?(parenthesis)(?!))\\)\\)?<lastSeparator>(,|
        |;|\\|)))", "${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[$2];",
        => null, 0),
    // default
    // 0
    (new Regex(@"(\\W)default(\\W)", "${1}0$2", 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),
    // \\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()) { }

```

```
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