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

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

```csharp
132            // ref TElement root
133            // ~!root!~ref TElement root
134            (new Regex(@"(?<definition>(?<= |\()(ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!ref))
    ↪  (?<variable>[a-zA-Z0-9]+)(?=\)|, | =))"), "~!${variable}!~${definition}", null,
    ↪  0),
135            // Inside the scope of ~!root!~ replace:
136            // root
137            // *root
138            (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),
139            // Remove scope borders.
140            // ~!root!~
141            //
142            (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
143            // ref auto root = ref
144            // ref auto root =
145            (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)"), "$1* $2 =$3", null, 0),
146            // *root = ref left;
147            // root = left;
148            (new Regex(@"\*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", null, 0),
149            // (ref left)
150            // (left)
151            (new Regex(@"\(ref ([a-zA-Z0-9]+)(\)|\(|,)"), "($1$2", null, 0),
152            //  ref TElement
153            //  TElement*
154            (new Regex(@"( |\()ref ([a-zA-Z0-9]+) "), "$1$2* ", null, 0),
155            // ref sizeBalancedTree.Root
156            // &sizeBalancedTree->Root
157            (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)"), "&$1->$2", null, 0),
158            // ref GetElement(node).Right
159            // &GetElement(node)->Right
160            (new Regex(@"ref ([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"),
    ↪  "&$1($2)->$3", null, 0),
161            // GetElement(node).Right
162            // GetElement(node)->Right
163            (new Regex(@"([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"), "$1($2)->$3",
    ↪  null, 0),
164            // [Fact]\npublic static void SizeBalancedTreeMultipleAttachAndDetachTest()
165            // TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
166            (new Regex(@"\[Fact\][\s\n]+(static )?void ([a-zA-Z0-9]+)\(\)"), "TEST_METHOD($2)",
    ↪  null, 0),
167            // class TreesTests
168            // TEST_CLASS(TreesTests)
169            (new Regex(@"class ([a-zA-Z0-9]+)Tests"), "TEST_CLASS($1)", null, 0),
170            // Assert.Equal
171            // Assert::AreEqual
172            (new Regex(@"Assert\.Equal"), "Assert::AreEqual", null, 0),
173            // TElement Root;
174            // TElement Root = 0;
175            (new Regex(@"(\r?\n[\t ]+)([a-zA-Z0-9:_]+(?<!return)) ([_a-zA-Z0-9]+);"), "$1$2 $3 =
    ↪  0;", null, 0),
176            // TreeElement _elements[N];
177            // TreeElement _elements[N] = { {0} };
178            (new Regex(@"(\r?\n[\t ]+)([a-zA-Z0-9]+) ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9]+)\];"),
    ↪  "$1$2 $3[$4] = { {0} };", null, 0),
179            // auto path = new TElement[MaxPath];
180            // TElement path[MaxPath] = { {0} };
181            (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),
182            // Insert scope borders.
183            // auto added = new HashSet<TElement>();
184            // ~!added!~std::unordered_set<TElement> added;
185            (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
    ↪  HashSet<(?<element>[a-zA-Z0-9]+)>\(\);"),
    ↪  "~!${variable}!~std::unordered_set<${element}> ${variable};", null, 0),
186            // Inside the scope of ~!added!~ replace:
187            // added.Add(node)
188            // added.insert(node)
189            (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),
190            // Inside the scope of ~!added!~ replace:
191            // added.Remove(node)
192            // added.erase(node)
```

```csharp
            (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),
            // if (added.insert(node)) {
            // if (!added.contains(node)) { added.insert(node);
            (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),
            // Remove scope borders.
            // ~!added!~
            //
            (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
            // Insert scope borders.
            // auto random = new System.Random(0);
            // std::srand(0);
            (new Regex(@"[a-zA-Z0-9\.]+ ([a-zA-Z0-9]+) = new
            ↪  (System\.)?Random\(([a-zA-Z0-9]+)\);"), "~!$1!~std::srand($3);", null, 0),
            // Inside the scope of ~!random!~ replace:
            // random.Next(1, N)
            // (std::rand() % N) + 1
            (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),
            // Remove scope borders.
            // ~!random!~
            //
            (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
            // Insert method body scope starts.
            // void PrintNodes(TElement node, StringBuilder sb, int level) {
            // void PrintNodes(TElement node, StringBuilder sb, int level) {/*method-start*/
            (new Regex(@"(?<start>\r?\n[\t ]+)(?<prefix>((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*/${end}", null,
            ↪  0),
            // Insert method body scope ends.
            // {/*method-start*/...}
            // {/*method-start*/.../*method-end*/}
            (new Regex(@"\{/\*method-start\*/(?<body>((?<bracket>\{)|(?<-bracket>\})|[^\{\}]*)+)
            ↪  \}"), "{/*method-start*/${body}/*method-end*/}", null,
            ↪  0),
            // Inside method bodies replace:
            // GetFirst(
            // this->GetFirst(
            //(new Regex(@"(?<separator>(\(|, |([\W]) |return ))(?<!(->|\*
            ↪  ))(?<method>(?!sizeof)[a-zA-Z0-9]+)\((?!\) \{)"),
            ↪  "${separator}this->${method}(", null, 1),
            (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),
            // Remove scope borders.
            // /*method-start*/
            //
            (new Regex(@"/\*method-(start|end)\*/"), "", 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
```

```csharp
245             (new Regex(@"(\W)default(\W)"), "${1}0$2", null, 0),
246             // //#define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
247             //
248             (new Regex(@"\/\/[ \t]*\#define[ \t]+[_a-zA-Z0-9]+[ \t]*"), "", null, 0),
249             // #if USEARRAYPOOL\r\n#endif
250             //
251             (new Regex(@"#if [a-zA-Z0-9]+\s+#endif"), "", null, 0),
252             // [Fact]
253             //
254             (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),
255             // \n ... namespace
256             // namespace
257             (new Regex(@"(\S[\r\n]{1,2})?[\r\n]+namespace"), "$1namespace", null, 0),
258             // \n ... class
259             // class
260             (new Regex(@"(\S[\r\n]{1,2})?[\r\n]+class"), "$1class", null, 0),
261         }.Cast<ISubstitutionRule>().ToList();
262
263         public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
    ↳  base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
264
265         public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
266     }
267 }
```

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

```csharp
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