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

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;

#pragma warning disable CS1591 // Missing XML comment for publicly visible type or member

namespace Platform.RegularExpressions.Transformer.CSharpToCpp
{
    public class CSharpToCppTransformer : Transformer
    {
        public static readonly IList<ISubstitutionRule> FirstStage = new List<SubstitutionRule>()
        {
            // #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
            //
            (new Regex(@"^\s*?\#pragma[\sa-zA-Z0-9\/]+$"), "", null, 0),
            // [MethodImpl(MethodImplOptions.AggressiveInlining)]
            //
            (new Regex(@"$\s+\[MethodImpl\(MethodImplOptions\.AggressiveInlining\)\]"), "", null, 0),
            // [Fact]
            //
            (new Regex(@"$\s+\[Fact\]"), "", null, 0),
            // {\n\n\n
            // {
            (new Regex(@"{\s+[\r\n]+"), "{" + Environment.NewLine, null, 0),
            // Platform.Collections.Methods.Lists
            // Platform::Collections::Methods::Lists
            (new Regex(@"(namespace[^\r\n]+?)\.([^\r\n]+?)"), "$1::$2", null, 20),
            // public abstract class
            // class
            (new Regex(@"(public abstract|static) class"), "class", null, 0),
            // class GenericCollectionMethodsBase {
            // class GenericCollectionMethodsBase { public:
            (new Regex(@"class ([a-zA-Z0-9]+)(\s+){"), "class $1$2{" + Environment.NewLine + "    public:", null, 0),
            // class GenericCollectionMethodsBase<TElement> {
            // template <typename TElement> class GenericCollectionMethodsBase { public:
            (new Regex(@"class ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>([^{]+){"), "template <typename $2> class $1$3{" + Environment.NewLine + "    public:", null, 0),
            // static void
            // TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement> tree, TElement* root)
            // template<typename T> static void
            // TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement> tree, TElement* root)
            (new Regex(@"static ([a-zA-Z0-9]+) ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>\(([^\)]+)\)"), "template <typename $3> static $1 $2($4)", null, 0),
            // (this
            // (
            (new Regex(@"\(this "), "(", null, 0),
            // Func<TElement> treeCount
            // TElement(*treeCount)()
            (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "$1(*$2)()", null, 0),
            // Action<TElement> free
            // void (*free)(TElement)
            (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "void (*$2)($1)", null, 0),
            // private const int MaxPath = 92;
            // static const int MaxPath = 92;
            (new Regex(@"private const ([a-zA-Z0-9]+) ([_a-zA-Z0-9]+) = ([a-zA-Z0-9]+);"), "static const $1 $2 = $3;", null, 0),
            // protected virtual
            // virtual
            (new Regex(@"protected virtual"), "virtual", null, 0),
            // protected abstract TElement GetFirst();
            // virtual TElement GetFirst() = 0;
            (new Regex(@"protected abstract ([^;]+);"), "virtual $1 = 0;", null, 0),
            // public virtual
            // virtual
            (new Regex(@"public virtual"), "virtual", null, 0),
            // protected readonly
            //
            (new Regex(@"protected readonly "), "", null, 0),
            // protected readonly TreeElement[] _elements;
```

```
            // TreeElement _elements[N];
            (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+)([\[\]]+)
            ↪  ([_a-zA-Z0-9]+);"), "$2 $4[N];", null, 0),
            // protected readonly TElement Zero;
            // TElement Zero;
            (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+) ([_a-zA-Z0-9]+);"), "$2
            ↪  $3;", null, 0),
            // private
            //
            (new Regex(@"(\W)(private|protected|public|internal) "), "$1", null, 0),
            // SizeBalancedTree(int capacity) => a = b;
            // SizeBalancedTree(int capacity) { a = b; }
            (new Regex(@"(^\s+)(override )?(void )?([a-zA-Z0-9]+)\(([^\(]+)\)\s+=>\s+([^;]+);"),
            ↪  "$1$2$3$4($5) { $6; }", null, 0),
            // () => Integer<TElement>.Zero,
            // () { return Integer<TElement>.Zero; },
            (new Regex(@"\(\)\s+=>\s+([^\r\n,;]+?),"), "() { return $1; },", null, 0),
            // => Integer<TElement>.Zero;
            // { return Integer<TElement>.Zero; }
            (new Regex(@"\)\s+=>\s+([^\r\n;]+?);"), ") { return $1; }", null, 0),
            // () { return avlTree.Count; }
            // []()-> auto { return avlTree.Count; }
            (new Regex(@", \(\) { return ([^;]+); }"), ", []()-> auto { return $1; }", null, 0),
            // Count => GetSizeOrZero(Root);
            // GetCount() { return GetSizeOrZero(Root); }
            (new Regex(@"([A-Z][a-z]+)\s+=>\s+([^;]+);"), "Get$1() { return $2; }", null, 0),
            // var
            // auto
            (new Regex(@"(\W)var(\W)"), "$1auto$2", null, 0),
            // unchecked
            //
            (new Regex(@"[\r\n]{2}\s*?unchecked\s*?$"), "", null, 0),
            // $"
            // "
            (new Regex(@"\$"""), "\"", null, 0),
            // Console.WriteLine("...")
            // printf("...\n")
            (new Regex(@"Console\.WriteLine\(""([^""]+)""\)"), "printf(\"$1\\n\")", null, 0),
            // throw new InvalidOperationException
            // throw std::exception
            (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
            ↪  std::exception", null, 0),
            // override void PrintNode(TElement node, StringBuilder sb, int level)
            // void PrintNode(TElement node, StringBuilder sb, int level) override
            (new Regex(@"override ([a-zA-Z0-9 \*\+]+)(\([^\)]+?\))"), "$1$2 override", null, 0),
            // string
            // char*
            (new Regex(@"(\W)string(\W)"), "$1char*$2", null, 0),
            // sbyte
            // std::int8_t
            (new Regex(@"(\W)sbyte(\W)"), "$1std::int8_t$2", null, 0),
            // uint
            // std::uint32_t
            (new Regex(@"(\W)uint(\W)"), "$1std::uint32_t$2", null, 0),
            // char*[] args
            // char* args[]
            (new Regex(@"([_a-zA-Z0-9:\*]?)\[\] ([a-zA-Z0-9]+)"), "$1 $2[]", null, 0),
            // using Platform.Numbers;
            //
            (new Regex(@"([\r\n]{2}|^)\s*?using [\.a-zA-Z0-9]+;\s*?$"), "", null, 0),
            // struct TreeElement { }
            // struct TreeElement { };
            (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([^;])"), "$1
            ↪  $2$3{$4};$5", null, 0),
            // class Program { }
            // class Program { };
            (new Regex(@"(struct|class) ([a-zA-Z0-9]+[^\r\n]*)([\r\n]+(?<indentLevel>[\t
            ↪  ]*)?)\{([\S\s]+?[\r\n]+\k<indentLevel>)\}([^;]|$)"), "$1 $2$3{$4};$5", null, 0),
            // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
            // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
            (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)"), "class $1 : public $2", null,
            ↪  0),
        }.Cast<ISubstitutionRule>().ToList();

        public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>()
        {
            // 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),
            // = ref GetLeftReference(root);
            // = GetLeftReference(root);
            (new Regex(@" = ref ([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\);"), " = $1($2);", null, 0),
            // ref this->GetElement(node)
            // this->GetElement(node)
            (new Regex(@"ref this->([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\)"), "this->$1($2)", null,
            ↪  0),
            // ref GetElement(node)
            // GetElement(node)
            (new Regex(@"ref ([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\)"), "$1($2)", null, 0),
            // = ref left;
            // = left;
            (new Regex(@" = ref ([a-zA-Z0-9]+);"), " = $1;", 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),
            // 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()) { }

        public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
    }
}
```

./Platform.RegularExpressions.Transformer.CSharpToCpp/obj/Release/netstandard2.1/Platform.RegularExpressions.

```
//------------------------------------------------------------------------------
// <auto-generated>
//     Generated by the MSBuild WriteCodeFragment class.
// </auto-generated>
//------------------------------------------------------------------------------

using System;
using System.Reflection;

[assembly: System.Reflection.AssemblyConfigurationAttribute("Release")]
[assembly: System.Reflection.AssemblyCopyrightAttribute("Konstantin Diachenko")]
[assembly: System.Reflection.AssemblyDescriptionAttribute("LinksPlatform\'s
↪  Platform.RegularExpressions.Transformer.CSharpToCpp Class Library" +
"")]
[assembly: System.Reflection.AssemblyFileVersionAttribute("0.0.1.0")]
[assembly: System.Reflection.AssemblyInformationalVersionAttribute("0.0.1")]
[assembly: System.Reflection.AssemblyTitleAttribute("Platform.RegularExpressions.Transformer.CSh
↪  arpToCpp")]
[assembly: System.Reflection.AssemblyVersionAttribute("0.0.1.0")]
```

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

```
using Xunit;

namespace Platform.RegularExpressions.Transformer.CSharpToCpp.Tests
{
    public class CSharpToCppTransformerTests
    {
        [Fact]
        public void HelloWorldTest()
        {
            const string helloWorldCode = @"using System;
class Program
{
    public static void Main(string[] args)
    {
        Console.WriteLine(""Hello, world!"");
    }
}";
            const string expectedResult = @"class Program
{
    public:
    static void Main(char* args[])
    {
        printf(""Hello, world!\n"");
    }
};";
            var transformer = new CSharpToCppTransformer();
            var actualResult = transformer.Transform(helloWorldCode, new Context(null));
            Assert.Equal(expectedResult, actualResult);
        }
    }
}
```

./Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/obj/Release/netcoreapp3.0/Platform.RegularExpres

```
//------------------------------------------------------------------------------
// <auto-generated>
//     Generated by the MSBuild WriteCodeFragment class.
// </auto-generated>
//------------------------------------------------------------------------------

using System;
using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("Platform.RegularExpressions.Transformer.C
↪ SharpToCpp.Tests")]
[assembly: System.Reflection.AssemblyConfigurationAttribute("Release")]
[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]
[assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0")]
[assembly: System.Reflection.AssemblyProductAttribute("Platform.RegularExpressions.Transformer.C
↪ SharpToCpp.Tests")]
[assembly: System.Reflection.AssemblyTitleAttribute("Platform.RegularExpressions.Transformer.CSh
↪ arpToCpp.Tests")]
[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]
```

Index