1.1 ./csharp/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 : TextTransformer
    {
        public static readonly IList<ISubstitutionRule> FirstStage = new List<SubstitutionRule>
        {
            // // ...
            //
            (new Regex(@"(\r?\n)?[ \t]+//+.+"), "", 0),
            // #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
            //
            (new Regex(@"^\s*?\#pragma[\sa-zA-Z0-9]+$"), "", 0),
            // {\n\n\n
            // {
            (new Regex(@"{\s+[\r\n]+"), "{" + Environment.NewLine, 0),
            // Platform.Collections.Methods.Lists
            // Platform::Collections::Methods::Lists
            (new Regex(@"(namespace[^\r\n]+?)\.([^\r\n]+?)"), "$1::$2", 20),
            // nameof(numbers)
            // "numbers"
            (new Regex(@"(?<before>\W)nameof\(([^)\n]+\.)?(?<name>[a-zA-Z0-9_]+)(<[^)\n]+>)?\)"), "${before}\"${name}\"", 0),
            // Insert markers
            // EqualityComparer<T> _equalityComparer = EqualityComparer<T>.Default;
            // EqualityComparer<T> _equalityComparer = EqualityComparer<T>.Default;/*~_comparer~*/
            (new Regex(@"(?<declaration>EqualityComparer<(?<type>[^>\n]+)> (?<comparer>[a-zA-Z0-9_]+) = EqualityComparer<\k<type>>\.Default;)"), "${declaration}/*~${comparer}~*/", 0),
            // /*~_equalityComparer~*/..._equalityComparer.Equals(Minimum, value)
            // /*~_equalityComparer~*/...Minimum == value
            (new Regex(@"(?<before>/\*~(?<comparer>[a-zA-Z0-9_]+)~\*/(.|\n)+\W)\k<comparer>\.Equals\((?<left>[^,\n]+), (?<right>[^)\n]+)\)"), "${before}${left} == ${right}", 50),
            // Remove markers
            // /*~_equalityComparer~*/
            //
            (new Regex(@"\r?\n[^\n]+/\*~[a-zA-Z0-9_]+~\*/"), "", 10),
            // Insert markers
            // Comparer<T> _comparer = Comparer<T>.Default;
            // Comparer<T> _comparer = Comparer<T>.Default;/*~_comparer~*/
            (new Regex(@"(?<declaration>Comparer<(?<type>[^>\n]+)> (?<comparer>[a-zA-Z0-9_]+) = Comparer<\k<type>>\.Default;)"), "${declaration}/*~${comparer}~*/", 0),
            // /*~_comparer~*/..._comparer.Compare(Minimum, value) <= 0
            // /*~_comparer~*/...Minimum <= value
            (new Regex(@"(?<before>/\*~(?<comparer>[a-zA-Z0-9_]+)~\*/(.|\n)+\W)\k<comparer>\.Compare\((?<left>[^,\n]+),
            (?<right>[^)\n]+)\)\s*(?<comparison>[<>=]=?)\s*0(?<after>\D)"),
            "${before}${left} ${comparison} ${right}${after}", 50),
            // Remove markers
            // private static readonly Comparer<T> _comparer = Comparer<T>.Default;/*~_comparer~*/
            //
            (new Regex(@"\r?\n[^\n]+/\*~[a-zA-Z0-9_]+~\*/"), "", 10),
            // Comparer<TArgument>.Default.Compare(maximumArgument, minimumArgument) < 0
            // maximumArgument < minimumArgument
            (new Regex(@"Comparer<[^>\n]+>\.Default\.Compare\(\s*(?<first>[^,)\n]+),\s*(?<second>[^\)\n]+)\s*\)\s*(?<comparison>[<>=]=?)\s*0(?<after>\D)"), "${first}
            ${comparison} ${second}${after}", 0),
            // public static bool operator ==(Range<T> left, Range<T> right) => left.Equals(right);
            //
            (new Regex(@"\r?\n[^\n]+bool operator ==\((?<type>[^\n]+) (?<left>[a-zA-Z0-9]+), \k<type> (?<right>[a-zA-Z0-9]+)\) =>
            (\k<left>|\k<right>)\.Equals\((\k<left>|\k<right>)\);"), "", 10),
            // public static bool operator !=(Range<T> left, Range<T> right) => !(left == right);
```

```
58              //
59              (new Regex(@"\r?\n[^\n]+bool operator !=\((?<type>[^\n]+) (?<left>[a-zA-Z0-9]+),
                ↪  \k<type> (?<right>[a-zA-Z0-9]+)\) => !\((\k<left>|\k<right>) ==
                ↪  (\k<left>|\k<right>)\);"), "", 10),
60              // public override bool Equals(object obj) => obj is Range<T> range ? Equals(range)
                ↪  : false;
61              //
62              (new Regex(@"\r?\n[^\n]+override bool Equals\((System\.)?[Oo]bject
                ↪  (?<this>[a-zA-Z0-9]+)\) => \k<this> is [^\n]+ (?<other>[a-zA-Z0-9]+) \?
                ↪  Equals\(\k<other>\) : false;"), "", 10),
63              // out TProduct
64              // TProduct
65              (new Regex(@"(?<before>(<|, ))(in|out)
                ↪  (?<typeParameter>[a-zA-Z0-9]+)(?<after>(>|,))"),
                ↪  "${before}${typeParameter}${after}", 10),
66              // public ...
67              // public: ...
68              (new Regex(@"(?<newLineAndIndent>\r?\n?[
                ↪  \t]*)(?<before>[^\{\(\r\n]*)(?<access>private|protected|public)[
                ↪  \t]+(?![^\{\(\r\n]*(interface|class|struct)[^\{\(\r\n]*[\{\(\r\n])"),
                ↪  "${newLineAndIndent}${access}: ${before}", 0),
69              // public: static bool CollectExceptions { get; set; }
70              // public: inline static bool CollectExceptions;
71              (new Regex(@"(?<access>(private|protected|public): )(?<before>(static )?[^\r\n]+
                ↪  )(?<name>[a-zA-Z0-9]+) {[^;}]*(?<=\W)get;[^;}]*(?<=\W)set;[^;}]*}"),
                ↪  "${access}inline ${before}${name};", 0),
72              // public abstract class
73              // class
74              (new Regex(@"((public|protected|private|internal|abstract|static)
                ↪  )*(?<category>interface|class|struct)"), "${category}", 0),
75              // class GenericCollectionMethodsBase<TElement> {
76              // template <typename TElement> class GenericCollectionMethodsBase {
77              (new Regex(@"class ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>([^{]+){"), "template <typename $2>
                ↪  class $1$3{", 0),
78              // static void
                ↪  TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
                ↪  tree, TElement* root)
79              // template<typename T> static void
                ↪  TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
                ↪  tree, TElement* root)
80              (new Regex(@"static ([a-zA-Z0-9]+) ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>\(([^\)\r\n]+)\)"),
                ↪  "template <typename $3> static $1 $2($4)", 0),
81              // interface IFactory<out TProduct> {
82              // template <typename TProduct> class IFactory { public:
83              (new Regex(@"interface (?<interface>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9
                ,]+)>(?<whitespace>[^{]+){"), "template <typename...> class ${interface};
                ↪  template <typename ${typeParameters}> class
                ↪  ${interface}<${typeParameters}>${whitespace}{" + Environment.NewLine + "
                ↪  public:", 0),
84              // template <typename TObject, TProperty, TValue>
85              // template <typename TObject, typename TProperty, TValue>
86              (new Regex(@"(?<before>template <(, )?typename [a-zA-Z0-9]+)+,
                ↪  )(?<typeParameter>[a-zA-Z0-9]+)(?<after>(,|>))"), "${before}typename
                ↪  ${typeParameter}${after}", 10),
87              // Insert markers
88              // private: static void BuildExceptionString(this StringBuilder sb, Exception
                ↪  exception, int level)
89              // /*~extensionMethod~BuildExceptionString~*/private: static void
                ↪  BuildExceptionString(this StringBuilder sb, Exception exception, int level)
90              (new Regex(@"private: static [^\r\n]+ (?<name>[a-zA-Z0-9]+)\(this [^\)\r\n]+\)"),
                ↪  "/*~extensionMethod~${name}~*/$0", 0),
91              // Move all markers to the beginning of the file.
92              (new Regex(@"\A(?<before>[^\r\n]+\r?\n(.|\n)+)(?<marker>/\*~extensionMethod~(?<name>」
                ↪  [a-zA-Z0-9]+)~\*/)"), "${marker}${before}",
                ↪  10),
93              // /*~extensionMethod~BuildExceptionString~*/...sb.BuildExceptionString(exception.In」
                ↪  nerException, level +
                ↪  1);
94              // /*~extensionMethod~BuildExceptionString~*/...BuildExceptionString(sb,
                ↪  exception.InnerException, level + 1);
95              (new Regex(@"(?<before>/\*~extensionMethod~(?<name>[a-zA-Z0-9]+)~\*/(.|\n)+\W)(?<var」
                ↪  iable>[_a-zA-Z0-9]+)\.\k<name>\("), "${before}${name}(${variable}, ",
                ↪  50),
96              // Remove markers
97              // /*~extensionMethod~BuildExceptionString~*/
98              //
```

```
99              (new Regex(@"/\*~extensionMethod~[a-zA-Z0-9]+~\*/"), "", 0),
100             // (this
101             // (
102             (new Regex(@"\(this "), "(", 0),
103             // public: static readonly EnsureAlwaysExtensionRoot Always = new
                ↪  EnsureAlwaysExtensionRoot();
104             // public:inline static EnsureAlwaysExtensionRoot Always;
105             (new Regex(@"(?<access>(private|protected|public): )?static readonly
                ↪  (?<type>[a-zA-Z0-9]+) (?<name>[a-zA-Z0-9_]+) = new \k<type>\(\);"),
                ↪  "${access}inline static ${type} ${name};", 0),
106             // public: static readonly string ExceptionContentsSeparator = "---";
107             // public: inline static const char* ExceptionContentsSeparator = "---";
108             (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly) string
                ↪  (?<name>[a-zA-Z0-9_]+) = ""(?<string>(\""|[^""\r\n])+)"";"), "${access}inline
                ↪  static const char* ${name} = \"${string}\";", 0),
109             // private: const int MaxPath = 92;
110             // private: inline static const int MaxPath = 92;
111             (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly)
                ↪  (?<type>[a-zA-Z0-9]+) (?<name>[_a-zA-Z0-9]+) = (?<value>[^;\r\n]+);"),
                ↪  "${access}inline static const ${type} ${name} = ${value};", 0),
112             //  ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
                ↪  TArgument : class
113             //  ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
114             (new Regex(@"(?<before> [a-zA-Z]+\(([a-zA-Z *,]+, |))(?<type>[a-zA-Z]+)(?<after>(|
                ↪  [a-zA-Z *,]+)\))[ \r\n]+where \k<type> : class"), "${before}${type}*${after}",
                ↪  0),
115             // protected: abstract TElement GetFirst();
116             // protected: virtual TElement GetFirst() = 0;
117             (new Regex(@"(?<access>(private|protected|public): )?abstract
                ↪  (?<method>[^;\r\n]+);"), "${access}virtual ${method} = 0;", 0),
118             // TElement GetFirst();
119             // virtual TElement GetFirst() = 0;
120             (new Regex(@"([\r\n]+[ ]+)((?!return)[a-zA-Z0-9]+ [a-zA-Z0-9]+\([^\)\r\n]*\))(;[
                ↪  ]*[\r\n]+)"), "$1virtual $2 = 0$3", 1),
121             // protected: readonly TreeElement[] _elements;
122             // protected: TreeElement _elements[N];
123             (new Regex(@"(?<access>(private|protected|public): )?readonly
                ↪  (?<type>[a-zA-Z<>0-9]+)([\[\]]+) (?<name>[_a-zA-Z0-9]+);"), "${access}${type}
                ↪  ${name}[N];", 0),
124             // protected: readonly TElement Zero;
125             // protected: TElement Zero;
126             (new Regex(@"(?<access>(private|protected|public): )?readonly
                ↪  (?<type>[a-zA-Z<>0-9]+) (?<name>[_a-zA-Z0-9]+);"), "${access}${type} ${name};",
                ↪  0),
127             // internal
128             //
129             (new Regex(@"(\W)internal\s+"), "$1", 0),
130             // static void NotImplementedException(ThrowExtensionRoot root) => throw new
                ↪  NotImplementedException();
131             // static void NotImplementedException(ThrowExtensionRoot root) { return throw new
                ↪  NotImplementedException(); }
132             (new Regex(@"(^\s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static
                ↪  )?(override )?([a-zA-Z0-9]+
                ↪  )([a-zA-Z0-9]+)\(([^\(\r\n]*)\)\s+=>\s+throw([^;\r\n]+);"),
                ↪  "$1$2$3$4$5$6$7$8($9) { throw$10; }", 0),
133             // SizeBalancedTree(int capacity) => a = b;
134             // SizeBalancedTree(int capacity) { a = b; }
135             (new Regex(@"(^\s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static
                ↪  )?(override )?(void )?([a-zA-Z0-9]+)\(([^\(\r\n]*)\)\s+=>\s+([^;\r\n]+);"),
                ↪  "$1$2$3$4$5$6$7$8($9) { $10; }", 0),
136             // int SizeBalancedTree(int capacity) => a;
137             // int SizeBalancedTree(int capacity) { return a; }
138             (new Regex(@"(^\s+)(private|protected|public)?(: )?(template \<[^>\r\n]+\> )?(static
                ↪  )?(override )?([a-zA-Z0-9]+
                ↪  )([a-zA-Z0-9]+)\(([^\(\r\n]*)\)\s+=>\s+([^;\r\n]+);"), "$1$2$3$4$5$6$7$8($9) {
                ↪  return $10; }", 0),
139             // () => Integer<TElement>.Zero,
140             // () { return Integer<TElement>.Zero; },
141             (new Regex(@"\(\)\s+=>\s+(?<expression>[^(),;\r\n]+(\(((?<parenthesis>\()|(?<-parent
                ↪  hesis>\))|[^();\r\n]*?)*?\))?[^(),;\r\n]*)(?<after>,|\);)"), "() { return
                ↪  ${expression}; }${after}", 0),
142             // => Integer<TElement>.Zero;
143             // { return Integer<TElement>.Zero; }
144             (new Regex(@"\)\s+=>\s+([^;\r\n]+?);"), ") { return $1; }", 0),
145             // () { return avlTree.Count; }
146             // [&]()-> auto { return avlTree.Count; }
```

```
147             (new Regex(@"(?<before>, |\()\(\) { return (?<expression>[^;\r\n]+); }"),
                ↪  "${before}[&]()-> auto { return ${expression}; }", 0),
148             // Count => GetSizeOrZero(Root);
149             // GetCount() { return GetSizeOrZero(Root); }
150             (new Regex(@"(\W)([A-Z][a-zA-Z]+)\s+=>\s+([^;\r\n]+);"), "$1Get$2() { return $3; }",
                ↪  0),
151             // ArgumentInRange(const char* message) { const char* messageBuilder() { return
                ↪  message; }
152             // ArgumentInRange(const char* message) { auto messageBuilder = [&]() -> const char*
                ↪  { return message; };
153             (new Regex(@"(?<before>\W[_a-zA-Z0-9]+\([^\)\n]*\)[\s\n]*{[\s\n]*([^{}]|\n)*?(\r?\n)
                    ?[ \t]*)(?<returnType>[_a-zA-Z0-9*:]+[_a-zA-Z0-9*: ]*)
                ↪  (?<methodName>[_a-zA-Z0-9]+)\((?<arguments>[^\)\n]*)\)\s*{(?<body>([^}]|\n)+?)}"
                ↪  ), "${before}auto ${methodName} = [&]() -> ${returnType} {${body}};",
                ↪  10),
154             // Func<TElement> treeCount
155             // std::function<TElement()> treeCount
156             (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", 0),
157             // Action<TElement> free
158             // std::function<void(TElement)> free
159             (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<void($1)> $2",
                ↪  0),
160             // Predicate<TArgument> predicate
161             // std::function<bool(TArgument)> predicate
162             (new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<bool($1)>
                ↪  $2", 0),
163             // var
164             // auto
165             (new Regex(@"(\W)var(\W)"), "$1auto$2", 0),
166             // unchecked
167             //
168             (new Regex(@"[\r\n]{2}\s*?unchecked\s*?$"), "", 0),
169             // throw new InvalidOperationException
170             // throw std::runtime_error
171             (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
                ↪  std::runtime_error", 0),
172             // void RaiseExceptionIgnoredEvent(Exception exception)
173             // void RaiseExceptionIgnoredEvent(const std::exception& exception)
174             (new Regex(@"(\(|, )(System\.Exception|Exception)( |\))"), "$1const
                ↪  std::exception&$3", 0),
175             // EventHandler<Exception>
176             // EventHandler<std::exception>
177             (new Regex(@"(\W)(System\.Exception|Exception)(\W)"), "$1std::exception$3", 0),
178             // override void PrintNode(TElement node, StringBuilder sb, int level)
179             // void PrintNode(TElement node, StringBuilder sb, int level) override
180             (new Regex(@"override ([a-zA-Z0-9 \*\+]+)(\([^\)\r\n]+?\))"), "$1$2 override", 0),
181             // return (range.Minimum, range.Maximum)
182             // return {range.Minimum, range.Maximum}
183             (new Regex(@"(?<before>return\s*)\((?<values>[^\)\n]+)\)(?!\()(?<after>\W)"),
                ↪  "${before}{${values}}${after}", 0),
184             // string
185             // const char*
186             (new Regex(@"(\W)string(\W)"), "$1const char*$2", 0),
187             // System.ValueTuple
188             // std::tuple
189             (new Regex(@"(?<before>\W)(System\.)?ValueTuple(?!\s*=)(?<after>\W)"),
                ↪  "${before}std::tuple${after}", 0),
190             // sbyte
191             // std::int8_t
192             (new Regex(@"(?<before>\W)((System\.)?SB|sb)yte(?!\s*=)(?<after>\W)"),
                ↪  "${before}std::int8_t${after}", 0),
193             // short
194             // std::int16_t
195             (new Regex(@"(?<before>\W)((System\.)?Int16|short)(?!\s*=)(?<after>\W)"),
                ↪  "${before}std::int16_t${after}", 0),
196             // int
197             // std::int32_t
198             (new Regex(@"(?<before>\W)((System\.)?I|i)nt(32)?(?!\s*=)(?<after>\W)"),
                ↪  "${before}std::int32_t${after}", 0),
199             // long
200             // std::int64_t
201             (new Regex(@"(?<before>\W)((System\.)?Int64|long)(?!\s*=)(?<after>\W)"),
                ↪  "${before}std::int64_t${after}", 0),
202             // byte
203             // std::uint8_t
```

```
            (new Regex(@"(?<before>\W)((System\.)?Byte|byte)(?!\s*=)(?<after>\W)"),
            ↪  "${before}std::uint8_t${after}", 0),
            // ushort
            // std::uint16_t
            (new Regex(@"(?<before>\W)((System\.)?UInt16|ushort)(?!\s*=)(?<after>\W)"),
            ↪  "${before}std::uint16_t${after}", 0),
            // uint
            // std::uint32_t
            (new Regex(@"(?<before>\W)((System\.)?UI|ui)nt(32)?(?!\s*=)(?<after>\W)"),
            ↪  "${before}std::uint32_t${after}", 0),
            // ulong
            // std::uint64_t
            (new Regex(@"(?<before>\W)((System\.)?UInt64|ulong)(?!\s*=)(?<after>\W)"),
            ↪  "${before}std::uint64_t${after}", 0),
            // char*[] args
            // char* args[]
            (new Regex(@"([_a-zA-Z0-9:\*]?)\[\] ([a-zA-Z0-9]+)"), "$1 $2[]", 0),
            // @object
            // object
            (new Regex(@"@([_a-zA-Z0-9]+)"), "$1", 0),
            // float.MinValue
            // std::numeric_limits<float>::min()
            (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MinValue(?<after>\W
            ↪  )"), "${before}std::numeric_limits<${type}>::min()${after}",
            ↪  0),
            // double.MaxValue
            // std::numeric_limits<float>::max()
            (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MaxValue(?<after>\W
            ↪  )"), "${before}std::numeric_limits<${type}>::max()${after}",
            ↪  0),
            // using Platform.Numbers;
            //
            (new Regex(@"([\r\n]{2}|^)\s*?using [\.a-zA-Z0-9]+;\s*?$"), "", 0),
            // struct TreeElement { }
            // struct TreeElement { };
            (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([^;])"), "$1
            ↪  $2$3{$4};$5", 0),
            // class Program { }
            // class Program { };
            (new Regex(@"(struct|class) ([a-zA-Z0-9]+[^\r\n]*)([\r\n]+(?<indentLevel>[\t
            ↪  ]*)?)\{([\S\s]+?[\r\n]+\k<indentLevel>)\}([^;]|$)"), "$1 $2$3{$4};$5", 0),
            // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
            // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
            (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)"), "class $1 : public $2", 0),
            // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
            // class IProperty : public ISetter<TValue, TObject>, IProvider<TValue, TObject>
            (new Regex(@"(?<before>class [a-zA-Z0-9]+ : ((public [a-zA-Z0-9]+(<[a-zA-Z0-9
            ↪  ,]+>)?, )+)?)(?<inheritedType>(?!public)[a-zA-Z0-9]+(<[a-zA-Z0-9
            ↪  ,]+>)?)(?<after>(, [a-zA-Z0-9]+(?!>)|[ \r\n]+))"), "${before}public
            ↪  ${inheritedType}${after}", 10),
            // 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}", 0),
            // Inside the scope of ~!root!~ replace:
            // root
            // *root
            (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}", 70),
            // Remove scope borders.
            // ~!root!~
            //
            (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", 5),
            // ref auto root = ref
            // ref auto root =
            (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)"), "$1* $2 =$3", 0),
            // *root = ref left;
            // root = left;
            (new Regex(@"\*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", 0),
            // (ref left)
            // (left)
            (new Regex(@"\(ref ([a-zA-Z0-9]+)(\)|\(|,)"), "($1$2", 0),
            //  ref TElement
```

```
263                 //  TElement*
264                 (new Regex(@"( |\()ref ([a-zA-Z0-9]+) "), "$1$2* ", 0),
265                 // ref sizeBalancedTree.Root
266                 // &sizeBalancedTree->Root
267                 (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)"), "&$1->$2", 0),
268                 // ref GetElement(node).Right
269                 // &GetElement(node)->Right
270                 (new Regex(@"ref ([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"),
                    ↪  "&$1($2)->$3", 0),
271                 // GetElement(node).Right
272                 // GetElement(node)->Right
273                 (new Regex(@"([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"), "$1($2)->$3", 0),
274                 // [Fact]\npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
275                 // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
276                 (new Regex(@"\[Fact\][\s\n]+(public: )?(static )?void ([a-zA-Z0-9]+)\(\)"), "public:
                    ↪  TEST_METHOD($3)", 0),
277                 // class TreesTests
278                 // TEST_CLASS(TreesTests)
279                 (new Regex(@"class ([a-zA-Z0-9]+)Tests"), "TEST_CLASS($1)", 0),
280                 // Assert.Equal
281                 // Assert::AreEqual
282                 (new Regex(@"(Assert)\.Equal"), "$1::AreEqual", 0),
283                 // Assert.Throws
284                 // Assert::ExpectException
285                 (new Regex(@"(Assert)\.Throws"), "$1::ExpectException", 0),
286                 // $"Argument {argumentName} is null."
287                 // ((std::string)"Argument ").append(argumentName).append(" is null.").data()
288                 (new Regex(@"\$""(?<left>(\\""|[^""\r\n])*){(?<expression>[_a-zA-Z0-9]+)}(?<right>(\
                    ↪  \""|[^""\r\n])*)"""),
                    ↪  "((std::string)$\"${left}\").append(${expression}).append(\"${right}\").data()",
                    ↪  10),
289                 // $"
290                 // "
291                 (new Regex(@"\$"""), "\"", 0),
292                 // Console.WriteLine("...")
293                 // printf("...\n")
294                 (new Regex(@"Console\.WriteLine\(""([^""\r\n]+)""\)"), "printf(\"$1\\n\")", 0),
295                 // TElement Root;
296                 // TElement Root = 0;
297                 (new Regex(@"(\r?\n[\t ]+)(private|protected|public)?(:
                    ↪  )?([a-zA-Z0-9:_]+(?<!return)) ([_a-zA-Z0-9]+);"), "$1$2$3$4 $5 = 0;", 0),
298                 // TreeElement _elements[N];
299                 // TreeElement _elements[N] = { {0} };
300                 (new Regex(@"(\r?\n[\t ]+)(private|protected|public)?(: )?([a-zA-Z0-9]+)
                    ↪  ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9]+)\];"), "$1$2$3$4 $5[$6] = { {0} };", 0),
301                 // auto path = new TElement[MaxPath];
302                 // TElement path[MaxPath] = { {0} };
303                 (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} };", 0),
304                 // bool Equals(Range<T> other) { ... }
305                 // bool operator ==(const Key &other) const { ... }
306                 (new Regex(@"(?<before>\r?\n[^\n]+bool )Equals\((?<type>[^\n{]+)
                    ↪  (?<variable>[a-zA-Z0-9]+)\)(?<after>(\s|\n)*{)"), "${before}operator ==(const
                    ↪  ${type} &${variable}) const${after}", 0),
307                 // private: static readonly ConcurrentBag<std::exception> _exceptionsBag = new
                    ↪  ConcurrentBag<std::exception>();
308                 // private: inline static std::mutex _exceptionsBag_mutex; \n\n private: inline
                    ↪  static std::vector<std::exception> _exceptionsBag;
309                 (new Regex(@"(?<begin>\r?\n?(?<indent>[ \t]+))(?<access>(private|protected|public):
                    )?static readonly ConcurrentBag<(?<argumentType>[^;\r\n]+)>
                    ↪  (?<name>[_a-zA-Z0-9]+) = new ConcurrentBag<\k<argumentType>>\(\);"),
                    ↪  "${begin}private: inline static std::mutex ${name}_mutex;" + Environment.NewLine
                    ↪  + Environment.NewLine + "${indent}${access}inline static
                    ↪  std::vector<${argumentType}> ${name};", 0),
310                 // public: static IReadOnlyCollection<std::exception> GetCollectedExceptions() {
                    ↪  return _exceptionsBag; }
311                 // public: static std::vector<std::exception> GetCollectedExceptions() { return
                    ↪  std::vector<std::exception>(_exceptionsBag); }
312                 (new Regex(@"(?<access>(private|protected|public): )?static
                    ↪  IReadOnlyCollection<(?<argumentType>[^;\r\n]+)> (?<methodName>[_a-zA-Z0-9]+)\(\)
                    ↪  { return (?<fieldName>[_a-zA-Z0-9]+); }"), "${access}static
                    ↪  std::vector<${argumentType}> ${methodName}() { return
                    ↪  std::vector<${argumentType}>(${fieldName}); }", 0),
313                 // public: static event EventHandler<std::exception> ExceptionIgnored =
                    ↪  OnExceptionIgnored; ... };
```

```
314             // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
                ↪  const std::exception&)> ExceptionIgnored = OnExceptionIgnored; };
315             (new Regex(@"(?<begin>\r?\n(\r?\n)?(?<halfIndent>[
                    \t]+)\k<halfIndent>)(?<access>(private|protected|public): )?static event
                ↪  EventHandler<(?<argumentType>[^;\r\n]+)> (?<name>[_a-zA-Z0-9]+) = (?<defaultDele
                ↪  gate>[_a-zA-Z0-9]+);(?<middle>(.|\n)+?)(?<end>\r?\n\k<halfIndent>};)"),
                ↪  "${middle}" + Environment.NewLine + Environment.NewLine +
                ↪  "${halfIndent}${halfIndent}${access}static inline
                ↪  Platform::Delegates::MulticastDelegate<void(void*, const ${argumentType}&)>
                ↪  ${name} = ${defaultDelegate};${end}", 0),
316             // Insert scope borders.
317             // class IgnoredExceptions { ... private: inline static std::vector<std::exception>
                ↪  _exceptionsBag;
318             // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: inline static
                ↪  std::vector<std::exception> _exceptionsBag;
319             (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n]+\r\n[\t
                    ]*{)(?<middle>((?!class).|\n)+?)(?<vectorFieldDeclaration>(?<access>(private|pro
                ↪  tected|public): )inline static std::vector<(?<argumentType>[^;\r\n]+)>
                ↪  (?<fieldName>[_a-zA-Z0-9]+);)"),
                ↪  "${classDeclarationBegin}/*~${fieldName}~*/${middle}${vectorFieldDeclaration}",
                ↪  0),
320             // Inside the scope of ~!_exceptionsBag!~ replace:
321             // _exceptionsBag.Add(exception);
322             // _exceptionsBag.push_back(exception);
323             (new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<befor
                ↪  e>((?<!/\*~\k<fieldName>~\*/)(.|\n))*?)\k<fieldName>\.Add"),
                ↪  "${scope}${separator}${before}${fieldName}.push_back", 10),
324             // Remove scope borders.
325             // /*~_exceptionsBag~*/
326             //
327             (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
328             // Insert scope borders.
329             // class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
330             // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: static std::mutex
                ↪  _exceptionsBag_mutex;
331             (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n]+\r\n[\t
                ↪  ]*{)(?<middle>((?!class).|\n)+?)(?<mutexDeclaration>private: inline static
                ↪  std::mutex (?<fieldName>[_a-zA-Z0-9]+)_mutex;)"),
                ↪  "${classDeclarationBegin}/*~${fieldName}~*/${middle}${mutexDeclaration}", 0),
332             // Inside the scope of ~!_exceptionsBag!~ replace:
333             // return std::vector<std::exception>(_exceptionsBag);
334             // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
                ↪  std::vector<std::exception>(_exceptionsBag);
335             (new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<befor
                ↪  e>((?<!/\*~\k<fieldName>~\*/)(.|\n))*?){(?<after>((?!lock_guard)[^{};\r\n])*\k<f
                ↪  ieldName>[^;}\r\n]*;)"), "${scope}${separator}${before}{
                ↪  std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", 10),
336             // Inside the scope of ~!_exceptionsBag!~ replace:
337             // _exceptionsBag.Add(exception);
338             // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
                ↪  _exceptionsBag.Add(exception);
339             (new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<befor
                ↪  e>((?<!/\*~\k<fieldName>~\*/)(.|\n))*?){(?<after>((?!lock_guard)([^{};]|\n))*?\r
                ↪  ?\n(?<indent>[ \t]*)\k<fieldName>[^;}\r\n]*;)"),
                ↪  "${scope}${separator}${before}{" + Environment.NewLine +
                ↪  "${indent}std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", 10),
340             // Remove scope borders.
341             // /*~_exceptionsBag~*/
342             //
343             (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
344             // Insert scope borders.
345             // class IgnoredExceptions { ... public: static inline
                ↪  Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
                ↪  ExceptionIgnored = OnExceptionIgnored;
346             // class IgnoredExceptions {/*~ExceptionIgnored~*/ ... public: static inline
                ↪  Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
                ↪  ExceptionIgnored = OnExceptionIgnored;
347             (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n]+\r\n[\t
                    ]*{)(?<middle>((?!class).|\n)+?)(?<eventDeclaration>(?<access>(private|protected
                ↪  |public): )static inline
                ↪  Platform::Delegates::MulticastDelegate<(?<argumentType>[^;\r\n]+)>
                ↪  (?<name>[_a-zA-Z0-9]+) = (?<defaultDelegate>[_a-zA-Z0-9]+);)"),
                ↪  "${classDeclarationBegin}/*~${name}~*/${middle}${eventDeclaration}", 0),
348             // Inside the scope of ~!ExceptionIgnored!~ replace:
349             // ExceptionIgnored.Invoke(NULL, exception);
350             // ExceptionIgnored(NULL, exception);
```

```
351             (new Regex(@"(?<scope>/\*~(?<eventName>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before
                ↪  >((?<!/\*~\k<eventName>~\*/)(.|\n))*?)\k<eventName>\.Invoke"),
                ↪  "${scope}${separator}${before}${eventName}", 10),
352             // Remove scope borders.
353             // /*~ExceptionIgnored~*/
354             //
355             (new Regex(@"/\*~[a-zA-Z0-9]+~\*/"), "", 0),
356             // Insert scope borders.
357             // auto added = new StringBuilder();
358             // /*~sb~*/std::string added;
359             (new Regex(@"(auto|(System\.Text\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new
                ↪  (System\.Text\.)?StringBuilder\(\);"), "/*~${variable}~*/std::string
                ↪  ${variable};", 0),
360             // static void Indent(StringBuilder sb, int level)
361             // static void Indent(/*~sb~*/StringBuilder sb, int level)
362             (new Regex(@"(?<start>, |\()(System\.Text\.)?StringBuilder
                ↪  (?<variable>[a-zA-Z0-9]+)(?<end>,|\))"), "${start}/*~${variable}~*/std::string&
                ↪  ${variable}${end}", 0),
363             // Inside the scope of ~!added!~ replace:
364             // sb.ToString()
365             // sb.data()
366             (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>
                ↪  ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.ToString\(\)"),
                ↪  "${scope}${separator}${before}${variable}.data()", 10),
367             // sb.AppendLine(argument)
368             // sb.append(argument).append('\n')
369             (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>
                ↪  ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.AppendLine\((?<argument>[^\),\
                ↪  r\n]+)\)"),
                ↪  "${scope}${separator}${before}${variable}.append(${argument}).append(1, '\\n')",
                ↪  10),
370             // sb.Append('\t', level);
371             // sb.append(level, '\t');
372             (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>
                ↪  ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.Append\('(?<character>[^'\r\n]
                ↪  +)', (?<count>[^\),\r\n]+)\)"),
                ↪  "${scope}${separator}${before}${variable}.append(${count}, '${character}')", 10),
373             // sb.Append(argument)
374             // sb.append(argument)
375             (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>
                ↪  ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.Append\((?<argument>[^\),\r\n]
                ↪  +)\)"), "${scope}${separator}${before}${variable}.append(${argument})",
                ↪  10),
376             // Remove scope borders.
377             // /*~sb~*/
378             //
379             (new Regex(@"/\*~[a-zA-Z0-9]+~\*/"), "", 0),
380             // Insert scope borders.
381             // auto added = new HashSet<TElement>();
382             // ~!added!~std::unordered_set<TElement> added;
383             (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
                ↪  HashSet<(?<element>[a-zA-Z0-9]+)>\(\);"),
                ↪  "~!${variable}!~std::unordered_set<${element}> ${variable};", 0),
384             // Inside the scope of ~!added!~ replace:
385             // added.Add(node)
386             // added.insert(node)
387             (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})", 10),
388             // Inside the scope of ~!added!~ replace:
389             // added.Remove(node)
390             // added.erase(node)
391             (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})", 10),
392             // if (added.insert(node)) {
393             // if (!added.contains(node)) { added.insert(node);
394             (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});", 0),
395             // Remove scope borders.
396             // ~!added!~
397             //
398             (new Regex(@"~![a-zA-Z0-9]+!~"), "", 5),
399             // 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);", 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}", 10),
            // Remove scope borders.
            // ~!random!~
            //
            (new Regex(@"~![a-zA-Z0-9]+!~"), "", 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>((private|protected|public): )?(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*/${end}",
            ↪  0),
            // Insert method body scope ends.
            // {/*method-start*/...}
            // {/*method-start*/.../*method-end*/}
            (new Regex(@"\{/\*method-start\*/(?<body>((?<bracket>\{)|(?<-bracket>\})|[^\{\}]*)+)
            ↪  \}"), "{/*method-start*/${body}/*method-end*/}",
            ↪  0),
            // Inside method bodies replace:
            // GetFirst(
            // this->GetFirst(
            //(new Regex(@"(?<separator>(\(|, |([\W]) |return ))(?<!(->|\*
            ↪  ))(?<method>(?!sizeof)[a-zA-Z0-9]+)\((?!\) \{)"),
            ↪  "${separator}this->${method}(", 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}", 100),
            // Remove scope borders.
            // /*method-start*/
            //
            (new Regex(@"/\*method-(start|end)\*/"), "", 0),
            // Insert scope borders.
            // const std::exception& ex
            // const std::exception& ex/*~ex~*/
            (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?exception&?
            ↪  (?<variable>[_a-zA-Z0-9]+))(?<after>\W)"),
            ↪  "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
            // Inside the scope of ~!ex!~ replace:
            // ex.Message
            // ex.what()
            (new Regex(@"(?<scope>/\*~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before
            ↪  >((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.Message"),
            ↪  "${scope}${separator}${before}${variable}.what()", 10),
            // Remove scope borders.
            // /*~ex~*/
            //
            (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
            // throw new ArgumentNullException(argumentName, message);
            // throw std::invalid_argument(((std::string)"Argument
            ↪  ").append(argumentName).append(" is null: ").append(message).append("."));
            (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(\".\"));", 0),
            // throw new ArgumentException(message, argumentName);
            // throw std::invalid_argument(((std::string)"Invalid
            ↪  ").append(argumentName).append(" argument: ").append(message).append("."));
            (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(\".\"));", 0),
```

```
446             // throw new ArgumentOutOfRangeException(argumentName, argumentValue,
                ↪  messageBuilder());
447             // throw std::invalid_argument(((std::string)"Value
                ↪  [").append(std::to_string(argumentValue)).append("] of argument
                ↪  [").append(argumentName).append("] is out of range:
                ↪  ").append(messageBuilder()).append("."));
448             (new Regex(@"throw new ArgumentOutOfRangeException\((?<argument>[a-zA-Z]*[Aa]rgument
                    [a-zA-Z]*([Nn]ame[a-zA-Z]*)?),
                ↪  (?<argumentValue>[a-zA-Z]*[Aa]rgument[a-zA-Z]*([Vv]alue[a-zA-Z]*)?),
                ↪  (?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\))?)\);"), "throw
                ↪  std::invalid_argument(((std::string)\"Value
                ↪  [\").append(std::to_string(${argumentValue})).append(\"] of argument
                ↪  [\").append(${argument}).append(\"] is out of range:
                ↪  \").append(${message}).append(\".\"));", 0),
449             // throw new NotSupportedException();
450             // throw std::logic_error("Not supported exception.");
451             (new Regex(@"throw new NotSupportedException\(\);"), "throw std::logic_error(\"Not
                ↪  supported exception.\");", 0),
452             // throw new NotImplementedException();
453             // throw std::logic_error("Not implemented exception.");
454             (new Regex(@"throw new NotImplementedException\(\);"), "throw std::logic_error(\"Not
                ↪  implemented exception.\");", 0),
455         }.Cast<ISubstitutionRule>().ToList();
456 
457         public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
458         {
459             // ICounter<int, int> c1;
460             // ICounter<int, int>* c1;
461             (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^>\r\n]+>)?)
                ↪  (?<variable>[_a-zA-Z0-9]+);"), "${abstractType}* ${variable};", 0),
462             // (expression)
463             // expression
464             (new Regex(@"(\(| )\(([a-zA-Z0-9_\*:]+)\)(,| |;|\))"), "$1$2$3", 0),
465             // (method(expression))
466             // method(expression)
467             (new Regex(@"(?<firstSeparator>(\(|
                ↪  ))\((?<method>[a-zA-Z0-9_\->\*:]+)\((?<expression>((?<parenthesis>\()|(?<-parent
                ↪  hesis>\))|[a-zA-Z0-9_\->\*:]*)+)(?(parenthesis)(?!))\)\)(?<lastSeparator>(,|
                ↪  |;|\)))"), "${firstSeparator}${method}(${expression})${lastSeparator}", 0),
468             // return ref _elements[node];
469             // return &_elements[node];
470             (new Regex(@"return ref ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9\*]+)\];"), "return &$1[$2];",
                ↪  0),
471             // null
472             // nullptr
473             (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)null
                ↪  (?<after>\W)"), "${before}nullptr${after}",
                ↪  10),
474             // default
475             // 0
476             (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)defa
                ↪  ult(?<after>\W)"), "${before}0${after}",
                ↪  10),
477             // object x
478             // void *x
479             (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)([O|
                ↪  o]bject|System\.Object) (?<after>\w)"), "${before}void *${after}",
                ↪  10),
480             // <object>
481             // <void*>
482             (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(?<!
                ↪  \w )([O|o]bject|System\.Object)(?<after>\W)"), "${before}void*${after}",
                ↪  10),
483             // ArgumentNullException
484             // std::invalid_argument
485             (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(Sys
                ↪  tem\.)?ArgumentNullException(?<after>\W)"),
                ↪  "${before}std::invalid_argument${after}", 10),
486             // struct Range<T> : IEquatable<Range<T>> {
487             // struct Range<T> {
488             (new Regex(@"(?<before>(struct|class) (?<type>[a-zA-Z0-9]+(<[^\n]+>)?)) :
                ↪  IEquatable<\k<type>>(?<after>(\s|\n)*{)"), "${before}${after}", 0),
489             // #region Always
490             //
491             (new Regex(@"(^|\r?\n)[ \t]*\#(region|endregion)[^\r\n]*(\r?\n|$)"), "", 0),
492             // //#define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
```

```
            //
            (new Regex(@"\/\/[ \t]*\#define[ \t]+[_a-zA-Z0-9]+[ \t]*"), "", 0),
            // #if USEARRAYPOOL\r\n#endif
            //
            (new Regex(@"#if [a-zA-Z0-9]+\s+#endif"), "", 0),
            // [Fact]
            //
            (new Regex(@"(?<firstNewLine>\r?\n|\A)(?<indent>[\t
            ↪  ]+)\[[a-zA-Z0-9]+(\((?<expression>((?<parenthesis>\()|(?<-parenthesis>\))|[^()\r
            ↪  \n]*)+)(?(parenthesis)(?!))\))?\][ \t]*(\r?\n\k<indent>)?"),
            ↪  "${firstNewLine}${indent}", 5),
            // \n ... namespace
            // namespace
            (new Regex(@"(\S[\r\n]{1,2})?[\r\n]+namespace"), "$1namespace", 0),
            // \n ... class
            // class
            (new Regex(@"(\S[\r\n]{1,2})?[\r\n]+class"), "$1class", 0),
            // \n\n
            // \n
            (new Regex(@"\r?\n[ \t]*\r?\n"), Environment.NewLine, 50),
        }.Cast<ISubstitutionRule>().ToList();

        public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
        ↪  base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }

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

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

```
using Xunit;

namespace Platform.RegularExpressions.Transformer.CSharpToCpp.Tests
{
    public class CSharpToCppTransformerTests
    {
        [Fact]
        public void EmptyLineTest()
        {
            // This test can help to test basic problems with regular expressions like incorrect
            ↪  syntax
            var transformer = new CSharpToCppTransformer();
            var actualResult = transformer.Transform("");
            Assert.Equal("", actualResult);
        }

        [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(const char* args[])
    {
        printf(""Hello, world!\n"");
    }
};";
            var transformer = new CSharpToCppTransformer();
            var actualResult = transformer.Transform(helloWorldCode);
            Assert.Equal(expectedResult, actualResult);
        }
    }
}
```

Index